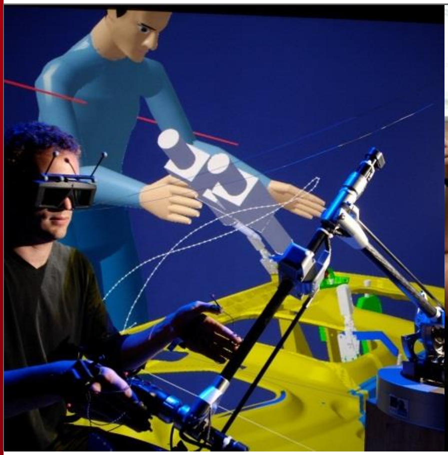
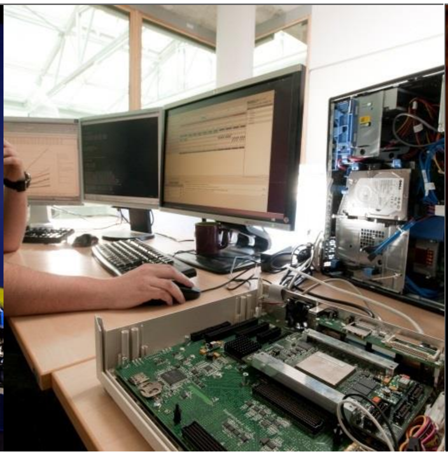
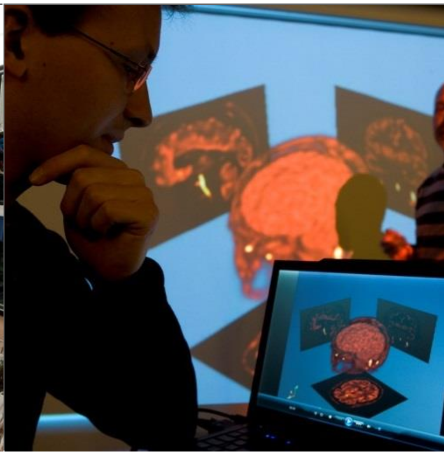
FROM RESEARCH TO INDUSTRY

[PSSM] – WEBEX DECEMBER 18TH

Jérémie TATIBOUET (CEA LIST)
Arnaud CUCCURU (CEA LIST)

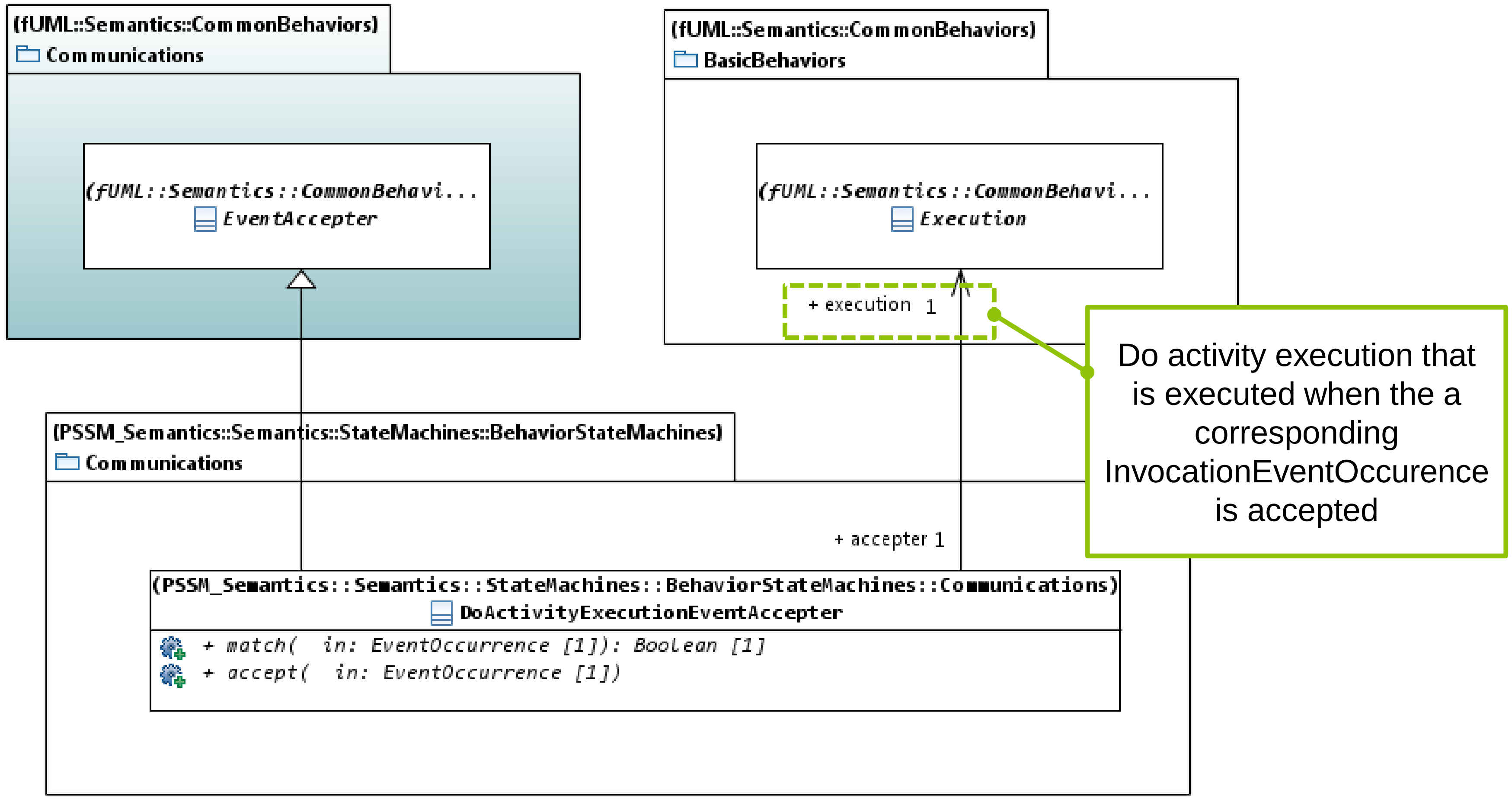
SUMMARY

Contributions from last meeting

- Minor change(s)
  - Bug 484578 – ConnectionPointReferenceActivation must be renamed
  - Bug 484582 – DoActivity must start before regions are entered
  - Bug 484583 – isReactive operation must be renamed
  - Bug 484584 – StateActivation and TransitionActivation “state” attribute must be renamed
- Major change(s)
  - Bug 484580 – DoActivity must be started asynchronously

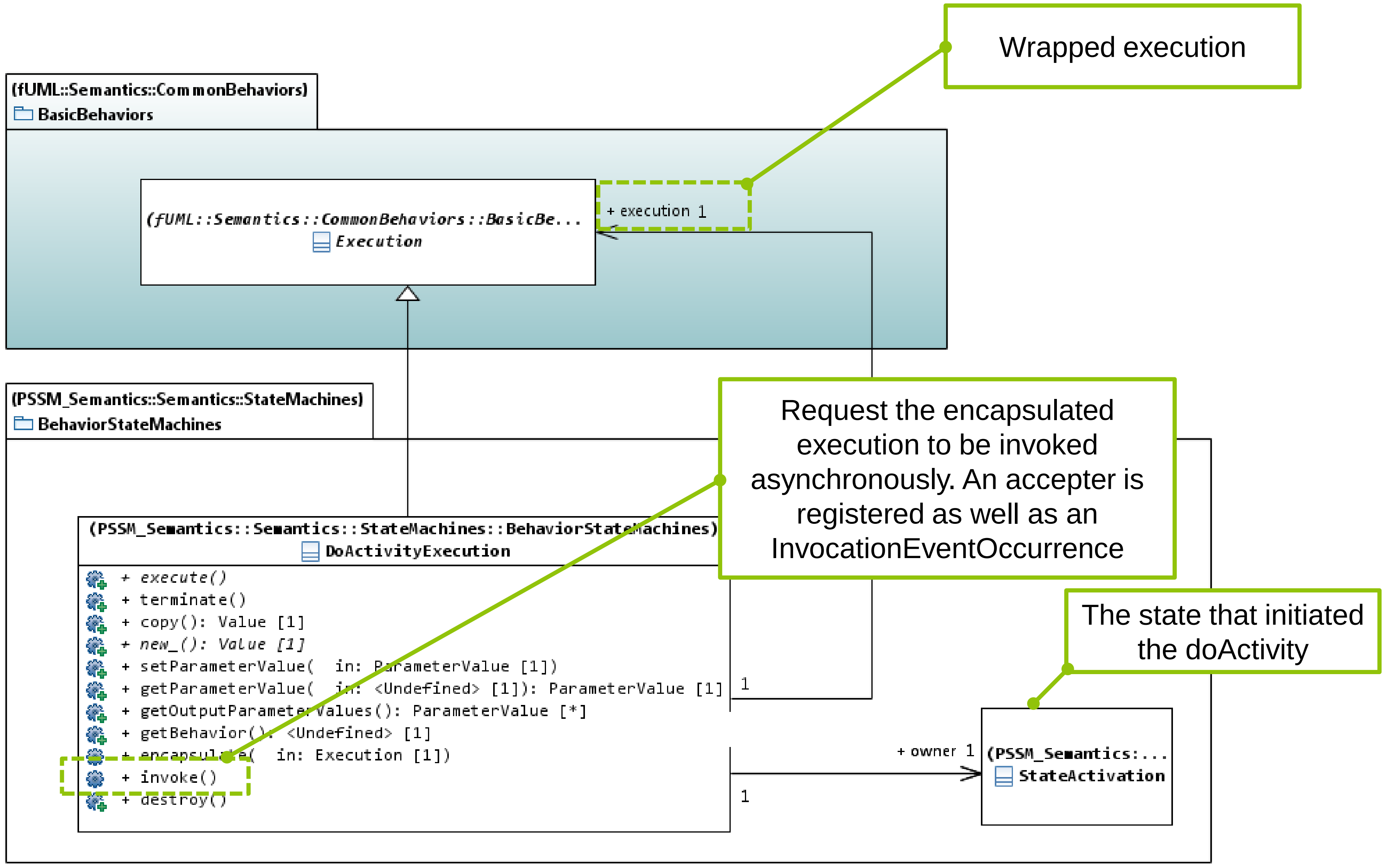
BUG 484584 - PROPOSAL

DO ACTIVITY MUST BE STARTED ASYNCHRONOUSLY


(fUML::Semantics::CommonBehaviors)
Communications
(fUML::Semantics::CommonBehavi...
EventAccepter
(fUML::Semantics::CommonBehaviors)
BasicBehaviors
(fUML::Semantics::CommonBehavi...
Execution
+ execution 1
(PSSM_Semantics::Semantics::StateMachines::BehaviorStateMachines)
Communications
+ accepter 1
(PSSM_Semantics::Semantics::StateMachines::BehaviorStateMachines::Communications)
DoActivityExecutionEventAccepter
+ match(in: EventOccurrence [1]): Boolean [1]
+ accept(in: EventOccurrence [1])
Do activity execution that is executed when the a corresponding InvocationEventOccurence is accepted

BUG 484584 - PROPOSAL

DO ACTIVITY MUST BE STARTED ASYNCHRONOUSLY



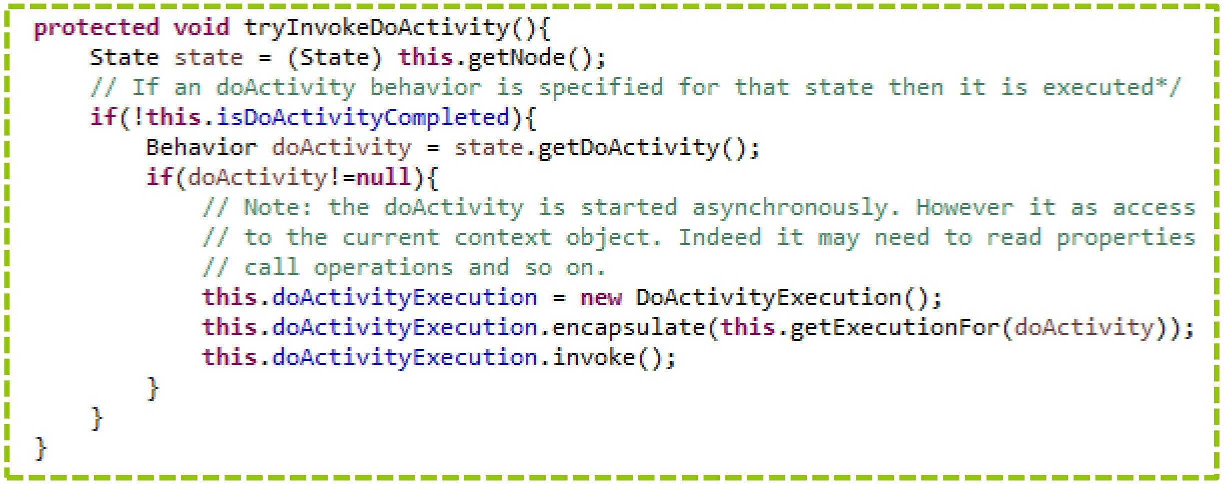
Wrapped execution
(fUML::Semantics::CommonBehaviors)
BasicBehaviors
(fUML::Semantics::CommonBehaviors::BasicBe...
Execution
+ execution 1
(PSSM_Semantics::Semantics::StateMachines)
BehaviorStateMachines
(PSSM_Semantics::Semantics::StateMachines::BehaviorStateMachines)
DoActivityExecution
+ execute()
+ terminate()
+ copy(): Value [1]
+ new_(): Value [1]
+ setParameterValue(in: ParameterValue [1])
+ getParameterValue(in: <Undefined> [1]): ParameterValue [1]
+ getOutputParameterValues(): ParameterValue [*]
+ getBehavior(): <Undefined> [1]
+ encapsulate(in: Execution [1])
+ invoke()
+ destroy()
1
1
+ owner 1
(PSSM_Semantics:...
StateActivation
Request the encapsulated execution to be invoked asynchronously. An accepter is registered as well as an InvocationEventOccurrence
The state that initiated the doActivity

FROM RESEARCH TO INDUSTRY cea tech

BUG 484584 – IMPLEMENTATION

DO ACTIVITY MUST BE STARTED ASYNCHRONOUSLY

List



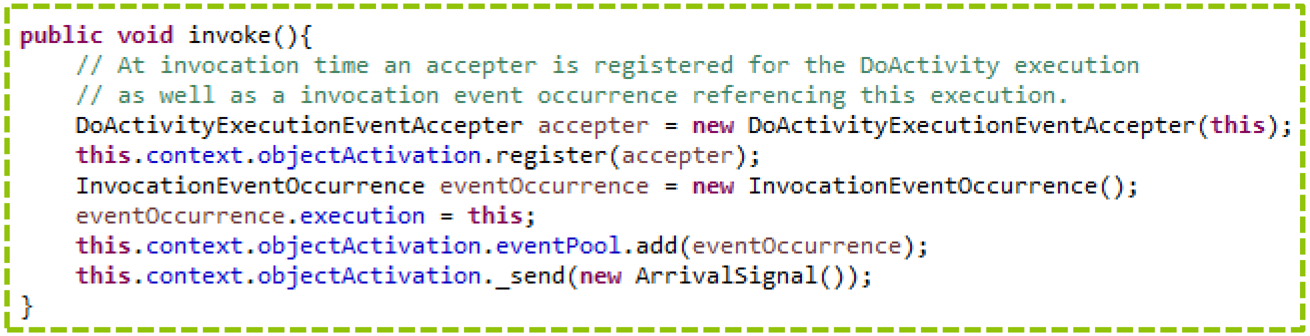
```
protected void tryInvokeDoActivity(){
    State state = (State) this.getNode();
    // If an doActivity behavior is specified for that state then it is executed*/
    if(!this.isDoActivityCompleted){
        Behavior doActivity = state.getDoActivity();
        if(doActivity!=null){
            // Note: the doActivity is started asynchronously. However it as access
            // to the current context object. Indeed it may need to read properties
            // call operations and so on.
            this.doActivityExecution = new DoActivityExecution();
            this.doActivityExecution.encapsulate(this.getExecutionFor(doActivity));
            this.doActivityExecution.invoke();
        }
    }
}
```

StateActivation

- tryInvokeDoActivity()
    - Encapsulate the execution corresponding to the “doActivity”
    - Request the execution to happen asynchronously

BUG 484584 – IMPLEMENTATION

DO ACTIVITY MUST BE STARTED ASYNCHRONOUSLY



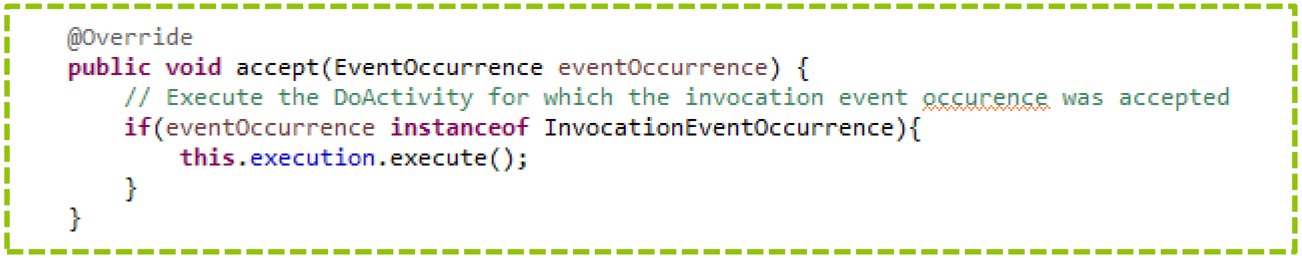
```java
public void invoke(){
    // At invocation time an accepter is registered for the DoActivity execution
    // as well as a invocation event occurrence referencing this execution.
    DoActivityExecutionEventAccepter accepter = new DoActivityExecutionEventAccepter(this);
    this.context.objectActivation.register(accepter);
    InvocationEventOccurrence eventOccurrence = new InvocationEventOccurrence();
    eventOccurrence.execution = this;
    this.context.objectActivation.eventPool.add(eventOccurrence);
    this.context.objectActivation._send(new ArrivalSignal());
}
```

DoActivityExecution

- Invoke()
  - Register a DoActivityExecutionEventAccepter
  - Register an InvocationEventOccurrence
  - Notify the arrival of a new Event

BUG 484584 – IMPLEMENTATION

DO ACTIVITY MUST BE STARTED ASYNCHRONOUSLY



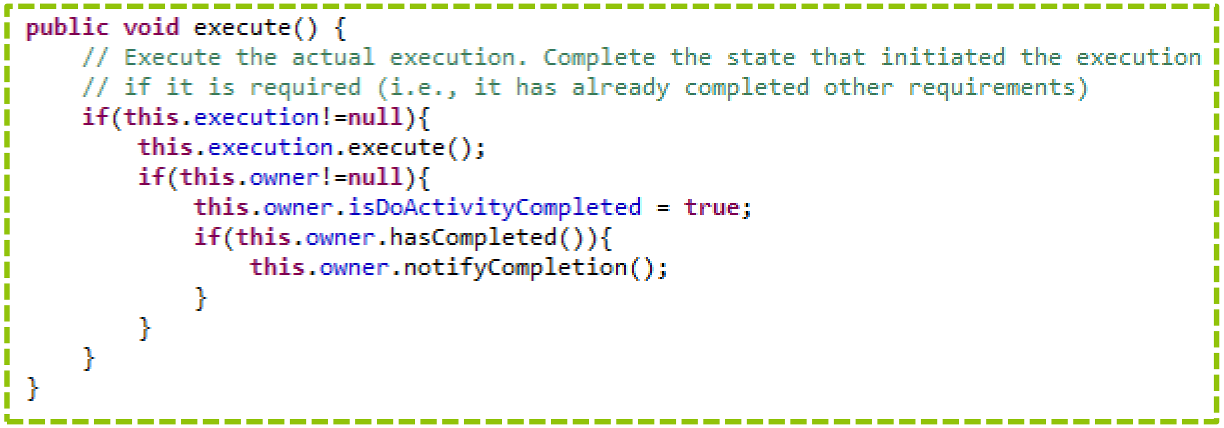
```java
@Override
public void accept(EventOccurrence eventOccurrence) {
    // Execute the DoActivity for which the invocation event occurence was accepted
    if(eventOccurrence instanceof InvocationEventOccurrence){
        this.execution.execute();
    }
}
```

DoActivityExecutionEventAccepter

- accept()
  - When a particular InvocationEventOccurrence is accepted by the accepter then the execution that is referenced by this latter gets executed

BUG 484584 – IMPLEMENTATION

DO ACTIVITY MUST BE STARTED ASYNCHRONOUSLY



```java
public void execute() {
    // Execute the actual execution. Complete the state that initiated the execution
    // if it is required (i.e., it has already completed other requirements)
    if(this.execution!=null){
        this.execution.execute();
        if(this.owner!=null){
            this.owner.isDoActivityCompleted = true;
            if(this.owner.hasCompleted()){
                this.owner.notifyCompletion();
            }
        }
    }
}
```

DoActivityExecution

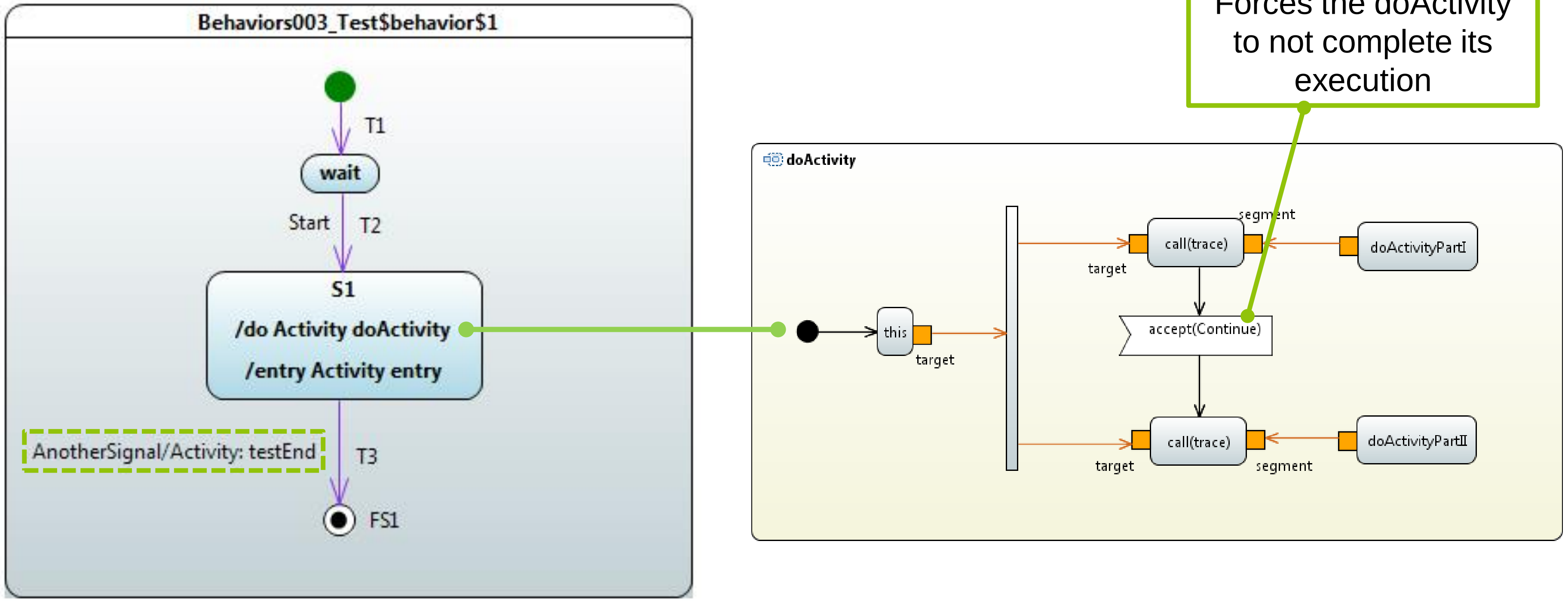
- execute()
  - When the execution terminates then the state which launched this doActivity gets notified that the this latter finished its execution
  - If the state has completed then a completion event is generated

BUG 484584 – TESTING

DO ACTIVITY MUST BE STARTED ASYNCHRONOUSLY

```
A State may also have an associated doActivity Behavior. This Behavior commences execution
when the State is entered (but only after the State entry Behavior has completed) and
executes concurrently with any other Behaviors that may be associated with the State, until
it completes (in which case a completion event is generated) or the State is exited, in
which case execution of the doActivity Behavior is aborted. (p.323)
```

Forces the doActivity to not complete its execution



```
S1(entry)::S1(doActivityPartI)
```