DE LA RECHERCHE À L'INDUSTRIE

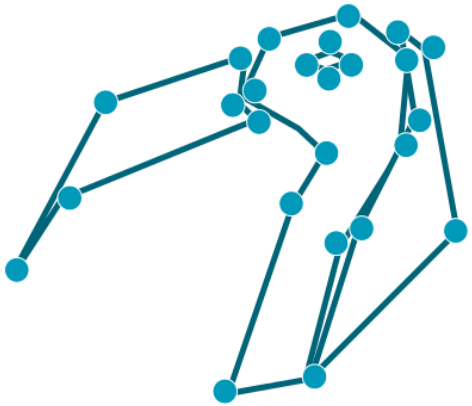
cea

www.cea.fr

- OMG TECHNICAL MEETING –

June 15 – 19, 2015, Berlin

Precise Semantics of State-Machines



Jérémie Tatibouët
CEA LIST / DILS / LISE
jeremie.tatibouet@cea.fr

Arnaud Cuccuru
CEA LIST / DILS / LISE
arnaud.cuccuru@cea.fr

INSTITUT CARNOT CEA LIST

digiteo

Objective

- “Solicit specifications containing more precise semantics for UML state machines to enable execution, allow model checking, and reduce ambiguities in UML models.”

Requirements

- “Proposals shall build on the precise semantics of Foundational UML (fUML)”
- “Be consistent with the Precise with the Precise Semantics of UML Composite Structures (PSCS)”

CEA

- French government-funded technological research organization

Proposal

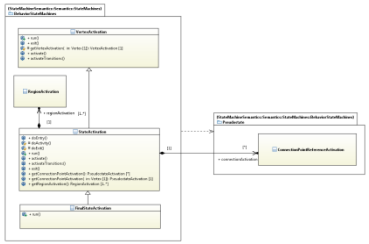
- Extension of fUML semantic model for a subset of UML state-machines
- This extension is consistent with PSCS
- Proposal components:
  1. A requirement model – extracted from UML 2.5 specification, chapter 14
  2. A semantic model (UML model)
  3. A test suite (UML model)

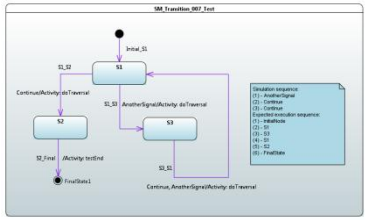
OUR WORKFLOW

Semantic model

Tests

Requirements





		A	B	C
		Id : String : Strin...	name : String [0..1]	text : String : String [1]
0	Event pool ownership	Event_001	Event pool ownership	The event pool for a StateMachine execution belongs to either its c...
1	Region activation	Region_001	*Region activation*	*A Region becomes active (i.e., it begins executing) wi...*
2	Region default activation	Region_002	Region default activation	A default activation of a Region occurs if the Region is entered im...
3	Region explicit activation	Region_003	Region explicit activation	an explicit activation occurs when a Region is entered by a Transiti...
4	State configuration representa...	State_Config_001	State configuration representation	a particular "state" of an executing StateMachine instance is repres...
5	Active state configuration unig...	State_Config_002	Active state configuration uniqueness	An executing StateMachine instance can only be in exactly one sta...
6	Active state	State_Config_003	Active state	A State is said to be active if it is part of the active state configurati...
7	State configuration stability	State_Config_004	State configuration stability	A state configuration is said to be stable when no further Transitio...

Add a verification relationship between a the test and the requirement

Refine the semantic model

Test case

CURRENT SCOPE OF THE PROPOSAL [1]

States

- *Entry:* if specified is executed when the state is entered
- *Exit:* if specified is executed when the state is exited
- *Final state* (specialization of a *State*) is supported
- States that are composite are supported
- *doActivity* is not supported
- State behaviors cannot be parameterized
- States that are composites cannot have multiple regions
  - This is also true for state-machines

Pseudo states

- InitialNode, EntryPoint and ExitPoint are supported
- Join, Fork, Junction, DeepHistory, ShallowHistory, Terminate or Choice are not supported

Transitions

- *Guards* are supported
- *Triggers* referencing *SignalEvent* are supported
  - Multiple *triggers* can be placed on the same *transition*
- Automated *transitions* (no *guard* and no *trigger*) are supported
- *Effect* on transitions are supported
- *Triggers* referencing *CallEvent* are not supported
  - Need to be integrated first into Foundational UML
- *Effects* placed on *transitions* cannot be parameterized
  - Note: we have an extension which provides the possibility to do this

Transitions selections

- During runtime we maintain a *StateMachineConfiguration*
  - Provides a snapshot of *active states* during execution
  - It is dynamically updated during the execution
- *Transitions* that are able to fire are calculated from the configuration
- The selection of *transitions* able to fire respects priority rules implied by the hierarchy
- The case of *transitions* able to fire at the same time in different *regions* is not supported

Event handling

- The *run-to-completion* step is handled as a chain of transitions that leads the *StateMachineConfiguration* in a stable state
  - No further transitions from the configuration are enabled to fire
  - All the entry behaviors of the configuration have completed
- Currently the possibility to defer an event is not handled
  - Consequence: the event is lost if no transition is ready to fire on it

ISSUES OF THE PROPOSAL [1]

Issue 1 -

- The current implementation introduces a change within fUML
  - Related to the difference in accepter selection logic between activities and state-machines
  - Impact: ObjectActivation, ClassifierBehaviorExecution and ActivityExecution
  - Solution: could be re-encoded in the StateMachineEventAccepter (replacing TransitionEventAccepter) class of the semantic model (cf. discussion with Ed).

Issue 2 -

- Open question – Do we need to maintain the StateMachineConfiguration during execution or should it be computed dynamically ?
  - Cons
    - Need to be updated when the set of active states changes
    - …
  - Pros
    - Can be reused for history pseudo states (e.g. deepHistory)
    - Provides an easy way to compute fireable transitions
    - Provides an easy way to check if the executed state-machine is in a stable state

QUESTIONS