Experimentations, transfer and development

*Final workshop of the ANR project GEMOC, March 17th, 2016*

Jérôme Le Noir (Thales Research & Technology)
Industrial Context

Can we coordinate this heterogeneous model?
Mode Automata & DataFlow Model Coordination
xDSML Development and Composition

Breakthrough #1: modular and explicit definition of the behavioral semantics of modeling languages

Breakthrough #2: explicit mapping used as behavioral interface of modeling languages

Breakthrough #3: integration of modeling languages for heterogeneous model coordination

- Data
- Control
- Communication

Event-driven behavioral interface

Concurrent execution of heterogeneous domain-specific models

AS

DSA

MoCC

AS

DSA

MoCC

MoCC

DSA

AS

xDSML

Semantics

DSE

(Domain Specific Event)
xCapella Mode Automata: DSA

Breakthrough #1: modular and explicit definition of the behavioral semantics of modeling languages

MoCC → DSA → AS

Semantics

Modular and explicit definition of the behavioral semantics of modeling languages.

Exec. Function and data

ModeAutomata.ecore

Exec. function

Exec. data

@Aspect (className=ModeRuntimeData)
class ModeRuntimeDataAspect {
  def public String onEnter() {
    var AbstractMode mode = _self.eContainer as AbstractMode
    var ModeMachine machine = mode.eContainer as ModeMachine
    for (ElementExtension ext : machine.ownedExtensions) {
      if (ext instanceof MachineRuntimeData) {
        ext as MachineRuntimeData).current = mode
      }
    }
    return "";
  }
}
xCapella Mode Automata: DSE & MoCC

Breakthrough #1: modular and explicit definition of the behavioral semantics of modeling languages

Breakthrough #2: explicit mapping used as behavioral interface of modeling languages

Package statemode
  context AbstractMode
  def : entering : Event = self.ownedExtensions->select(E | EoclIsTypeOf(ModeRuntimeData))->first().onEnter()
  def : leaving : Event = self.ownedExtensions->select(E | EoclIsTypeOf(ModeRuntimeData))->first().onLeave()

Event-driven behavioral interface
Mode automata Animator

Decoration of current mode
Breath life into an industrial modeling workbench with the GEMOC approach
xCapella Mode Automata
xCapella Mode Automata & DataFlow Coordination Specification

Breakthrough #1: modular and explicit definition of the behavioral semantics of modeling languages.

```
Operator MatchingandCoordinationSharedEventsActivate (dse_entering : i_Mode::entering, dse_activate : i_DF::activate)
  MatchingCorrespondance: when
  "(dse_entering.oclAsType(Mode_).enterActions->first().oclAsType(BroadcastEventAction).name) = dse_activate.name";
  CoordinationRule: facilities.RendezVous(dse_entering,dse_activate)
end operator;
```

```
Operator MatchingandCoordinationSharedEventsDeactivate (dse_leaving : i_Mode::leaving, dse_deactivate : i_DF::deactivate)
  MatchingCorrespondance: when
  "(dse_leaving.oclAsType(Mode_).enterActions->first().oclAsType(BroadcastEventAction).name) = dse_deactivate.name";
  CoordinationRule: facilities.RendezVous(dse_leaving,dse_deactivate)
end operator;
```

Breakthrough #3: integration of modeling languages for heterogeneous model coordination.

Event-driven behavioral interface
Mode Automata & Data Flow Model Coordination
Transfer : GEMOC Studio

- GEMOC studio : http://gemoc.org/studio-download
- GEMOC studio update site: http://gemoc.org/updatesite/studio
  1. Model debugging: (sequential) execution, trace management and animation
     • Requires: Xtend/Java, generative approach for trace management, generic execution engine, generic animation framework
  2. Concurrency modeling and analysis: (concurrent) execution, and analysis tools
     • Requires: model debugging + MoccML, ECL/GEL, Timesquare, concurrent execution engine
  3. Behavioral coordination of, possibly heterogeneous, models: coordination engine
     • Requires: concurrency + BCOoL, coordination execution engine
Transfer : Experimentations

GEMOC Studio examples (deployed) :
  TFSM
  SigPML

Public GEMOC experimentations :
  Activity Diagram (fUML) (cf. https://github.com/gemoc/activitydiagram)
  Farming modeling (cf. https://github.com/gemoc/farmingmodeling)

Internal experimentation :
  xCapella, an executable extension of Capella (PoC)
Generic technologies (EPL) which have proven helpful and are looking for an interest to bring it at full maturity level.

- Try executable Arduino Designer $0.1.x$
- Read the GEMOC Publications
- Join the GEMOC Initiative
- Get in touch with us to breath life into your designer