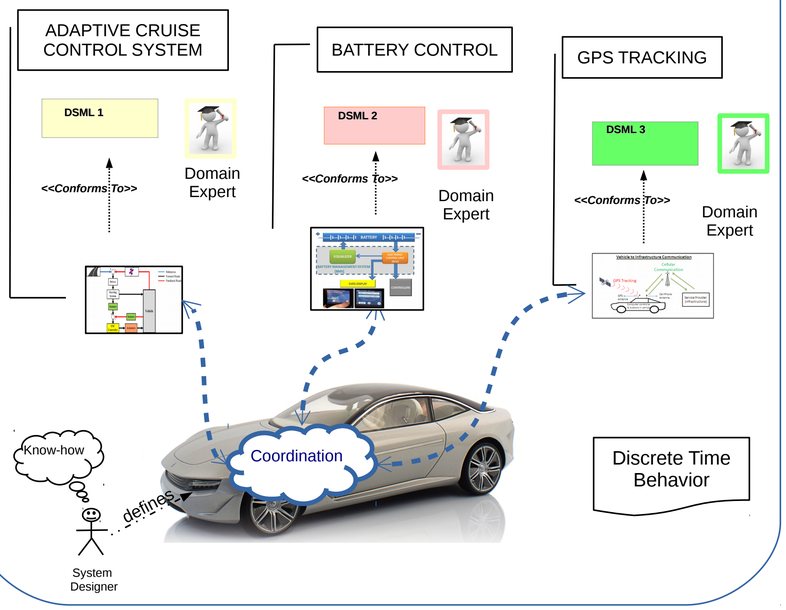
A model based environment for automatic model coordination

Matias Vara Larsen[1,2,3,4] Julien DeAntoni[1,2,3,4] Benoit Combemale[3] Frédéric Mallet[1,2,3,4]

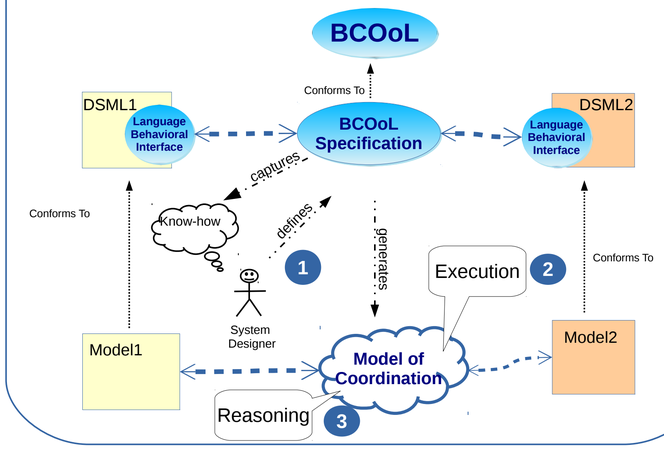
Introduction

In Model Driven Engineering, the coordination of behavioral models is necessary to understand and validate the emerging system behavior. In this context, we propose the Behavioral Coordination Operator Language (BCOoL) that enables the system designer to specify coordination patterns between languages in order to automate the coordination between models. This research is part of the GEMOC initiative.

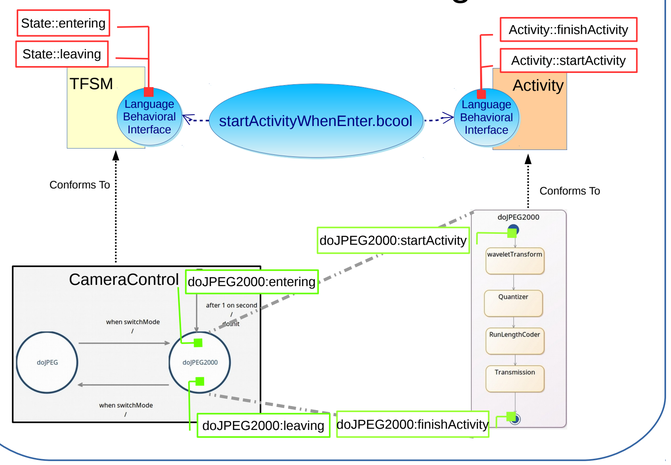
Coordination of Heterogeneous Models



Proposed Approach



The Camera Encoder Algorithm



BCOoL implementation

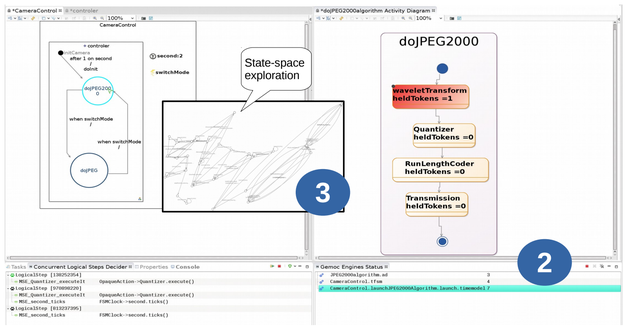
BCOoL is developed as a set of plugins for Eclipse as part of the GEMOC studio. BCOoL is itself based on EMF and its abstract syntax has been developed using Ecore. While the GEMOC studio supports several facilities, e.g., editing, graphical representation, animation and execution of domain-specific tools, BCOoL adds coordination facilities.

```
// We import a libraries that gather some predefined event relations that we will use in the coordination rule
// The library is defined in a MOCCML/CCSL
ImportLib "platform:/resource/org.gemoc.sample.bcool.tfsm_activityDiagram_composition/modeAutomataCoordination.moccml"
// a BCOoL spec imports the interfaces
ImportInterface "platform:/plugin/org.modelexecution.operationalsemantics.gemoc.ecl/model/ActivitiyDiagramV2.ecl" as ad
ImportInterface "platform:/plugin/org.gemoc.sample.tfsm.eclmoc2as/ecl/TFSM.ecl" as tfsm

Spec activity_tfsm
// The operator specifies What Events from the Interfaces will be coordinated
Operator  StartActivityWhenEnter (activityStart : ad::startActivity , activityStop : ad::finishActivity, enterState : tfsm::entering, leaveState : tfsm::leaving)
        // The matching correspondence specifies When instances of Events must be coordinated
        // Activities and States are selected by name
        MatchingCorrespondance:
            when (
                (activityStart.name = activityStop.name )
                and
                (enterState.name = leaveState.name)
                and
                (activityStart.name = enterState.name )
            );
        CoordinationRule:
        // The coordination rule specifies how the selected instances are coordinated
        // The event relation LoopFromStartToFinishNonPreemtive is defined in the library previously imported.
            facilities.LoopFromStartToFinishNonPeemptive (enterState, leaveState, activityStart, activityStop)
end operator;
```

1

Textual Concrete Syntax by using Xtext



Heterogeneous Execution of the Coordinated Models

Contact | matias.vara_larsen@inria.fr | http://timesquare.inria.org/BCOoL

1 2 3 4