The Behavioral Coordination Operator Language

GLACE Day, May 10th, 2016

Julien Deanton
University of Nice, I3S CNRS, INRIA AOSTE
Julien.deanton@polytech.unice.fr
Conforms To

Discrete Time Behavior

xDMSL 1

ADAPTIVE CRUISE CONTROL SYSTEM
Heterogeneous models and languages
Emerging System Behavior difficult to apprehend
Heterogeneous models and languages
Emerging System Behavior difficult to apprehend
Outline

• State of the Art:
  – Coordination Languages and ADLs
  – Coordination Frameworks

• Our proposal:
  – The Behavioral Coordination Operator Language

• Conclusion
Running Example: the Coffee Machine

Timed Finite State Machine (TFSM)

State-based language

Activity

Conforms To

Action-based language

Conforms To

[Diagram of a coffee machine with a state transition diagram]

when coin / !selectCoffee;

[Diagram of a state transition diagram with actions]

+ CoffeeCoin

when releaseCoffee / doLock

Locked → Unlocked

selectCoffee

releaseCoffee

start

coin
Running Example: the Coffee Machine

Timed Finite State Machine (TFSM)

State-based language

Action-based language

Activity

Conforms To

Conforms To

+ CoffeeCoin

when releaseCoffee
\( \rightarrow \) doLock

when coin
\( \rightarrow \) !selectCoffee;

selectCoffee

releaseCoffee

start

coin

CoffeeAlgorithm

selectCoffee

makeCoffee

releaseCoffee
Running Example: the Coffee Machine

Timed Finite State Machine (TFSM)

State-based language

Action-based language

Conforms To

Activity

e.g., When the event selectCoffee occurs, the Action selectCoffee is started
Coordination Languages & ADLs

Coordination Languages proposed to model the coordination by using a dedicated Language, e.g., Linda or Esper.

ADLs proposed component based languages to coordinate, possibly heterogeneous behaviors, e.g., Wright, Rapide or MetaH.
Coordination Languages proposed to model the coordination by using a dedicated Language, e.g., Linda or Esper.

ADLs proposed component based languages to coordinate, possibly heterogeneous behaviors, e.g., Wright, Rapide or MetaH.
Coordination Languages proposed to model the coordination by using a dedicated Language, e.g., Linda or Esper.

ADLs proposed component based languages to coordinate, possibly heterogeneous behaviors, e.g., Wright, Rapide or MetaH.
Coordination Languages & ADLs

Coordination Languages proposed to model the coordination by using a dedicated Language, e.g., Linda or Esper.

ADLs proposed component based languages to coordinate, possibly heterogeneous behaviors, e.g., Wright or MetaH.

The coordination is modeled explicitly

The coordination is defined rule by rule
Coordination Patterns

Know-how

System Designer

TFSM

Activity

Model of Coordination

What?, When?, How?

Conforms To

Conforms To

Model of Coordination

What?, When?, How?

Model Behavioral Interface

Model Behavioral Interface

CoffeeAlgorithm

selectCoffee

makeCoffee

releaseCoffee

when releaseCoffee / doLock

when coin / !selectCoffee;

Locked

Unlocked

selectCoffee

releaseCoffee

start

coin
Coordination frameworks identified a systematic way to coordinate models, i.e., a coordination pattern.
Coordination Frameworks

Coordination frameworks identified a systematic way to coordinate models, i.e., a coordination pattern.
Coordination Frameworks

Ptolemy/Modhel’X Framework

Syntax

Semantics (SDF)

Semantics (FSM)

Conforms To

Actors:
- Actor 0: D0: SDF
- Actor 1
- Actor 2: D1: FSM

Activities:
- selectCoffee
- releaseCoffee
- start
- coin

Locked

Conforms To

Activity

CoffeeAlgorithm
- selectCoffee
- makeCoffee
- releaseCoffee
Coordination Frameworks

Ptolemy/Modheli'X Framework

Syntax

Semantics (SDF)

Semantics (FSM)

Conforms To

Activity

Conforms To

Conforms To

Actor 0

D0: SDF

Actor 1

Coordination

Actor 2

D1: FSM

A3

A4

Composite Actor

Director

Locked

selectCoffee

releaseCoffee

start

coin

Locked

selectCoffee

releaseCoffee

start

coin

TFSM

selectCoffee; releaseCoffee; start; coin

locked

selectCoffee

releaseCoffee

start

coin

locked

selectCoffee

releaseCoffee

start

coin

locked

selectCoffee

releaseCoffee

start

coin

locked
Coordination Frameworks

Coordination frameworks identified a systematic way to coordinate models, i.e., a coordination pattern.
Coordination Frameworks

TFSM
- Syntax
- Behavioral Semantics

Know-how
System Designer

Activity
- Syntax

Director: TFSM
Conforms To

Language Behavioral Interface
Coordination Frameworks e.g. Ptolemy

to encode

generates

Behavioral Interface

Director: Activity
Conforms To

Model of Coordination

Model Behavioral Interface

when releaseCoffee /
doLock

locked

when coin /
!selectCoffee;

selectCoffee 

releaseCoffee

start

CoffeeAlgorithm
Coordination frameworks identified a systematic way to coordinate models, i.e., a coordination pattern.

What?, When? and How? the methods from the semantics Interfaces are invoked (Java)
Coordination frameworks identified a systematic way to coordinate models, i.e., a *coordination pattern*.

- The coordination is generated automatically.
- The coordination pattern is hidden/encoded in the tool in a GPL.
Take-Away Lessons

• Coordination Languages & ADLs:
  - 😊 The coordination is modeled explicitly
  - 😞 The coordination is defined rule by rule

• Coordination Frameworks:
  - 😊 The coordination is generated automatically based on a coordination pattern
  - 😞 The coordination pattern is hidden/encoded in the tool in a GPL
Outline

- State of the Art
  - Coordination Languages
  - Coordination Frameworks

- Our proposal:
  - The Behavioral Coordination Operator Language

- Conclusion
Model of Coordination

BCOoL Specification

<<Conforms To>>

BCOoL

Language Behavioral Interface

TFSM

Syntax

Behavioral Semantics

Activity

Behavioral Semantics

Syntax

Language Behavioral Interface

Know-how

System Designer

defines

generates

System Designer

when coin
! selectCoffee;
when releaseCoffee / doLock

selectCoffee
releaseCoffee
start
coin

Model of Coordination

Model Behavioral Interface

Model Behavioral Interface

CoffeeAlgorithm

selectCoffee
makeCoffee
releaseCoffee
BCOoL Specification

<<Conforms To>>

DSE (Domain Specific Event)

MoCC

DSA

AS

Model of Coordination

Behavioral Interface

Behavioral Semantics

System Designer

Know-how

defines

generates

Model Behavioral Interface

Model Behavioral Interface

DSE (Domain Specific Event)

MoCC

DSA

AS

<<Conforms To>>

activity

Behavioral Semantics

CoffeeAlgorithm

selectCoffee

makeCoffee

releaseCoffee

when releaseCoffee
  / doLock

when coin
  / !selectCoffee;

selectCoffee

releaseCoffee

start

coin
Model of Coordination

BCOoL Specification

<<Conforms To>>

when coin / !selectCoffee;

when releaseCoffee / doLock

selectCoffee  
releaseCoffee

Model Behavioral Interface

CoffeeAlgorithm

makeCoffee

Model Behavioral Interface
when coin / !selectCoffee;

CoffeeCoin

when releaseCoffee / doLock

Locked

Unlocked

selectCoffee

releaseCoffee

start

coin

Model of Coordination

BCOoL Specification

State::entering

State::leaving

FSMEvent::occurs

Model

Behavioral Interface

Model

Behavioral Interface

BCOoL

<<Conforms To>>

DSE (Domain Specific Event)

DSE

MoCC

DSA

AS

Behavioral Semantics

AS

MoCC

DSA

DSE

(TFSM)

Behavioral Semantics

TFSM

Behavioral Semantics

MoCC

DSA

DSE

(TFSM)
Model of Coordination

Model of Behavioral Interface

Behavioral Interface

selectCoffee
releaseCoffee
start!

when releaseCoffee
Locked
Unlocked

when coin

Model of Coordination

Model of Behavioral Interface

BCoOL Specification

BCoOL

DSE (Domain-Specific Event)

MoCC

MoCC

DSA

MoCC

MoCC

BCoOL

FSMEvent::occurs

Alternates

State::entering

State::leaving

FSMEvent::occurs

IncomingTransitions

OutgoingTransitions

OwnedStates

0..*

Transition

InitialState

0..*

Target

0..*

Source

0..*

IncomingEvents

0..*

GeneratedEvents

0..*

SolicitingTransitions

<<Conforms To>>

coffeeAlgorithm

releaseCoffee
makeCoffee
selectCoffee
Model of Coordination

BCOoL Specification

<<Conforms To>>
Model of Coordination

FSMEvent::occurs

Locked::entering
Locked::leaving

releaseCoffee::occurs
selectCoffee::occurs

when coin / !selectCoffee;
locked / !doLock

State::entering
State::leaving

<<Conforms To>>

BCOoL Specification

Gemoc
**Model of Coordination**

- **FSM**
  - States: `Locked, Unlocked`
  - Transitions:
    - `Locked:entering`
    - `Locked:leaving`
    - `Locked:entering`
    - `Locked:leaving`
  - Events:
    - `selectCoffee:occurs`
    - `releaseCoffee:occurs`

- **BCOoL**
  - **Action::startAction**
  - **Action::finishAction**
  - **State::entering**
  - **State::leaving**
  - **FSMEvent::occurs**

- **Behavioral Semantics**
  - MoCC
  - DSA
  - TFSM
  - DSE

- **Specification**
  - **Conforms To**
  - Specification of `Interactions` and `Actions`
Model of Coordination

Know-how

System Designer

<<Conforms To>>

BCOoL Specification

defines

generates

Model of Coordination

when coin / !selectCoffee;

selectCoffee
releaseCoffee
start
coin

CoffeeAlgorithm

selectCoffee
makeCoffee

releaseCoffee
A Model of Coordination Know-how System

**BCOoL** Specification

<<Conforms To>>

specification

defines

generates

System Designer

Know-how

when coin / selectCoffee;
when releaseCoffee / doLock
BCOoL Metamodel

SyncFSMEventsAndActions.bcool
SyncFSMEventsAndActions.bcoool

ImportInterface tfsm;
ImportInterface Activity;
**BCOoL Metamodel**

**SyncFSMEventsAndActions.bcool**

```plaintext
ImportInterface tfsm;
ImportInterface Activity;

Operator RendezVousWhenSameName
(FSMEvent::occurs, Action::startAction)

End Operator;
```
SyncFSMEventsAndActions.bcool

ImportInterface tfsm;
ImportInterface Activity;

Operator RendezVousWhenSameName
(FSMEvent::occurs, Action::startAction)

When(occurs.name = startAction.name);

End Operator;

OCL Boolean Expression between model elements

BCOoL Metamodel

TFSM

Behavioral Semantics

MoCC
DSE
(Domain Specific Event)
DSEs

CorrespondenceMatching
Condition
(from OCL)

BCoolSpecification
2..* operators
1..* importInterfaceStatement

importBehavioralInterface

Operator

Condition
(from OCL)

DSA

Activity

BCOoL

Specification

FSMEvent::occurs

Action::startAction

<<Conforms To>>
Causal and Temporal relations between Events e.g., Rendezvous, Precedes, etc.

SyncFSMEventsAndActions.bcool

ImportInterface tfsm;
ImportInterface Activity;

Operator RendezVousWhenSameName
(FSMEvent::occurs,Action::startAction)

When(occurs.name = startAction.name);

CoordinationRule:
  RendezVous (occurs, startAction)
End Operator;
BCOoL Metamodel

Defined in MoCCML
(Model of Concurrency and Communication Modeling Language)

SyncFSMEventsAndActions.bcool

ImportInterface tfsm;
ImportInterface Activity;

Operator RendezVousWhenSameName
(FSMEvent::occurs, Action::startAction)

When (occurs.name = startAction.name);

CoordinationRule:
RendezVous (occurs, startAction)
End Operator;
SyncFSMEventsAndActions.bcool

ImportInterface tfsm;
ImportInterface Activity;

Operator FSMEventsandActions
(FSMEvent::occurs, Action::startAction)
  When (occurs.name = startAction.name);
CoordinationRule:
  Rendezvous (occurs, startAction)
End Operator;

selectCoffee:startAction
makeCoffee:startAction
makeCoffee:finishAction
releaseCoffee:startAction
releaseCoffee:finishAction
SyncFSMEventsAndActions.bcool

ImportInterface tfsm;
ImportInterface Activity;

Operator FSMEventsandActions (FSMEvent::occurs, Action::startAction)
   When (occurs.name = startAction.name);
CoordinationRule:
   Rendezvous (occurs, startAction)
End Operator;
SyncFSMEventsAndActions.bcool

```plaintext
ImportInterface tfsm;
ImportInterface Activity;

Operator FSMEventsandActions (FSMEvent::occurs, Action::startAction)
  When(occurs.name = startAction.name);
  CoordinationRule:
  Rendezvous (occurs, startAction)
End Operator;
```

```
When (releaseCoffee) {
  doLock
}
when coin {
  !selectCoffee;
}
when releaseCoffee {
  !releaseCoffee;
}
```

- `selectCoffee::startAction`
- `makeCoffee::startAction`
- `releaseCoffee::startAction`
- `selectCoffee::occurs`
- `releaseCoffee::occurs`
SyncFSMEventsAndActions.bcool

```
ImportInterface tfsm;
ImportInterface Activity;

Operator FSMEventsAndActions
(FSMEvent::occurs, Action::startAction)
  When (occurs.name = startAction.name);
CoordinationRule:
  Rendezvous (occurs, startAction)
End Operator;
```
SyncFSMEventsAndActions.bcool

ImportInterface tfsm;
ImportInterface Activity;

Operator FSMEventsandActions
(FSMEvent::occurs, Action::startAction)
When (occurs.name = startAction.name);
CoordinationRule:
Rendezvous (occurs, startAction)
End Operator;

when releaseCoffee
I doLock

selectCoffee::startAction

Rendezvous

selectCoffee::occurs

releaseCoffee::startAction

Rendezvous

releaseCoffee::occurs
SyncFSMEventsAndActions.bcool

ImportInterface tfsm;
ImportInterface Activity;

Operator FSMEventsandActions
(FSMEvent::occurs, Action::startAction)
  When(occurs.name = startAction.name);
CoordinationRule:
  Rendezvous (occurs, startAction)
End Operator;

Heterogeneous Execution In the GEMOC Studio
Implemented into the GEMOC studio

Editing facilities by using Xtext
Implemented into the GEMOC studio

Schedule space exploration

Coordinated Heterogeneous Execution
Conclusion

- BCOoL is a dedicated metalanguage to capture coordination patterns.
- It automates the coordination of models by relying on a formal language.
- It is associated to the GEMOC language/modeling workbench to execute and analyze the coordinated system.

Future work:
- Using the explicit coordination to generate master on co-simulation bus
- Understanding the interconnection with physical model (continuous time)

1http://timesquare.inria.fr/BCOoL
Thanks

http://timesquare.inria.fr/BCOoL

http://gemoc.org/ins