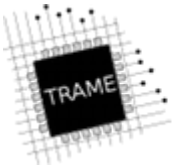
Automatic UML transfo. 4 Context-aware Verification



Frederic.Jouault
Ciprian.Teodorov
Jerome.Delatour
Luka.Le_Roux
Philippe.Dhaussy

@eseo.fr @ensta-bretagne.fr

Outline

- Context
  - UML
  - Context-based verification & OBP
- Transformation Overview
- UML Subset
- tUML
- ABCD
- Conclusion & Perspectives

Context

- Complex systems with many interacting components
- Verification by Exhaustive Reachability Analysis
  - Automatic
  - Safety and Liveness
  - Counter-example

UML

- A modeling language
  - Graphical, with more than a dozen diagrams
  - Able to model structural and behavioral aspects
  - Semi-formal
- Usable for
  - Requirements
  - Specification
  - Design
- Facilitates dialogue between experts
- Extensible via profiles, notably:
  - MARTE for real-time & embedded systems
  - SysML for systems engineering

Context-aware Verification

- Explicit environment modeling (**contexts**)
  - Operating modes
  - External actors
  - Environmental conditions
- **Requirements** associated to **contexts**

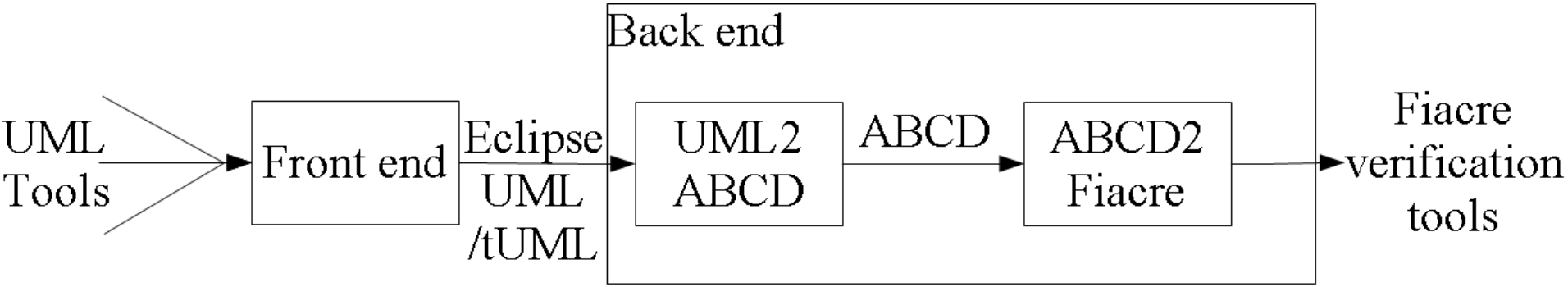
www.obpcdl.org

System-under-study || Context

in Fiacre in CDL

Transformation Overview

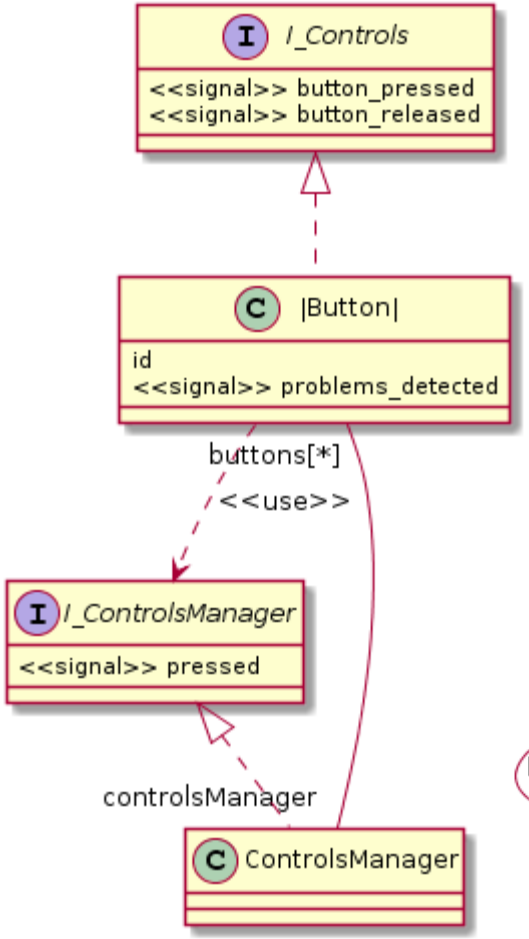
- Model in UML, convert to Fiacre, verify
  - Front end: handles UML tools variability
  - Back end: handles Fiacre model generation
  - Eclipse UML 2.4 metamodel used as reference & pivot abstract syntax

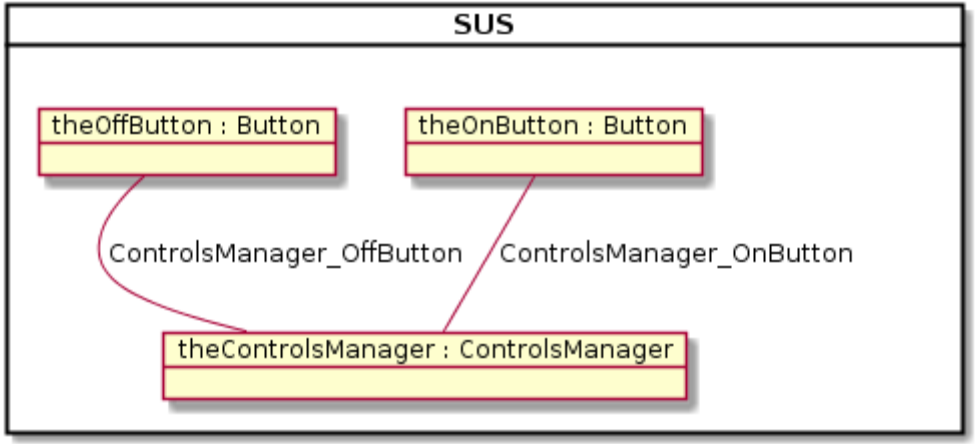


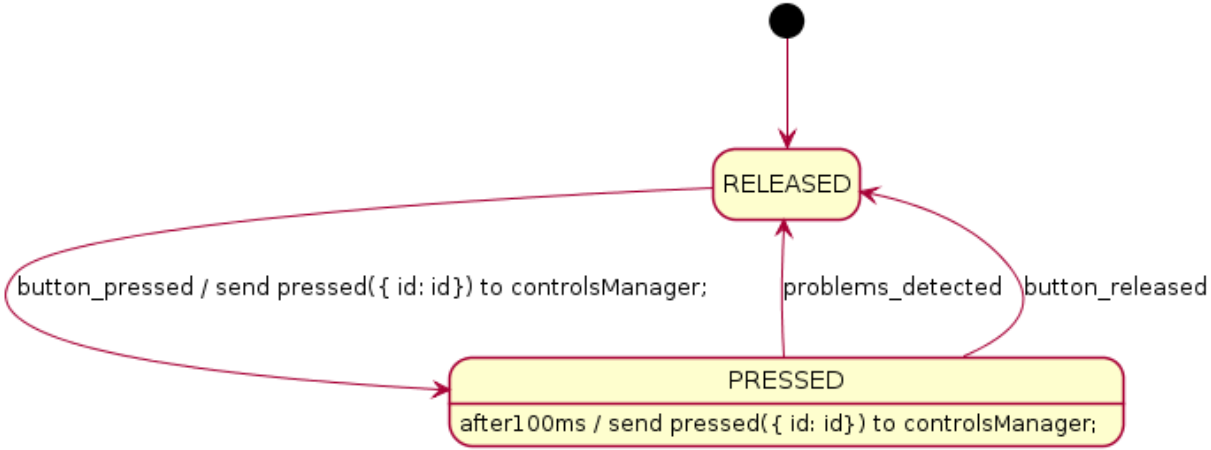
UML Subset

- Only three diagrams, typical of embedded systems modeling
  - Class
  - Composite structure
  - State
- **Precise** and **unambiguous** semantics
- We model at **general design** level
  - Active objects
  - Communication via asynchronous signals

UML Subset Example







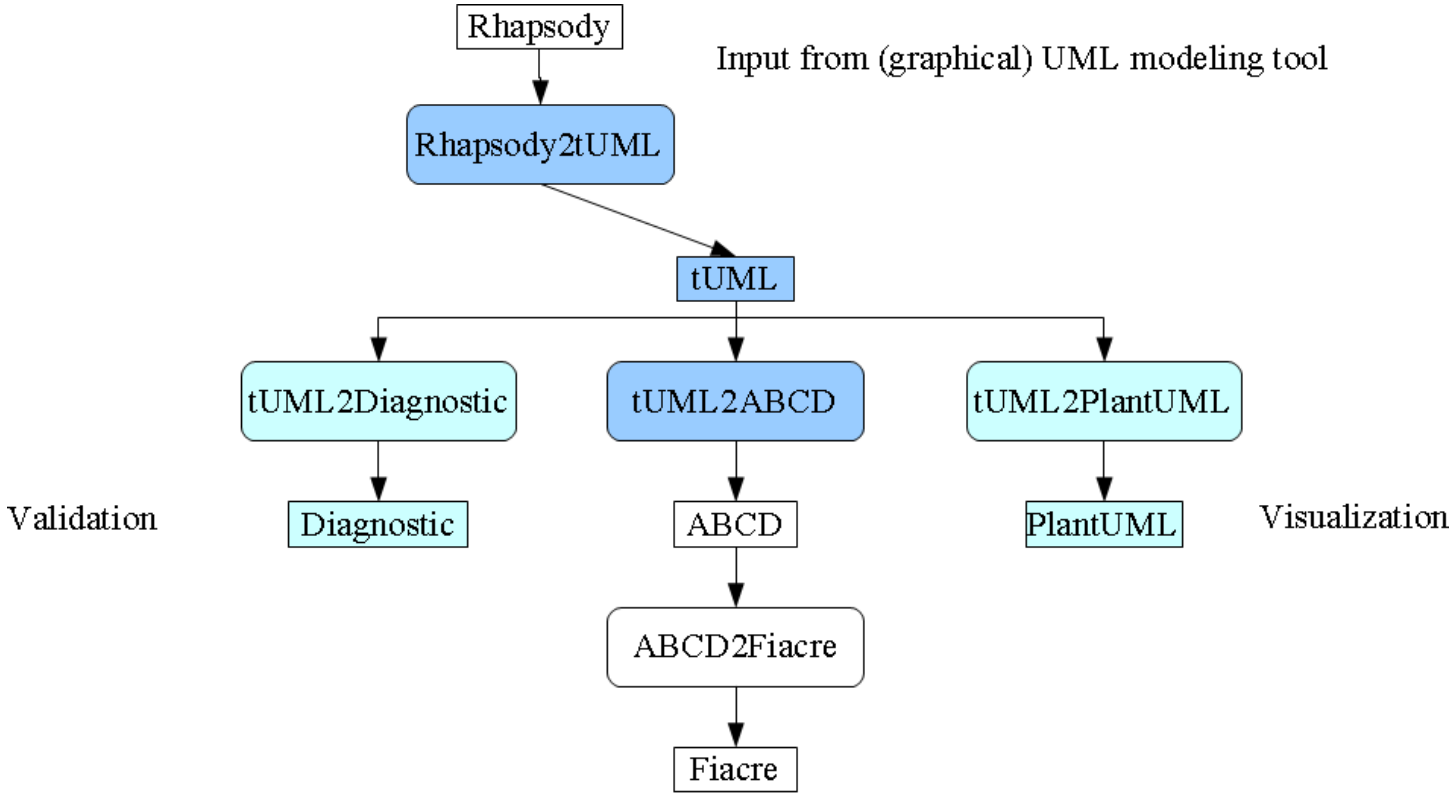
tUML

- Problem with graphical UML tools
  - No global model view
  - Hidden “details” (accessible as “properties” via several clicks)
- Approach: a textual syntax for this UML subset
  - A single homogeneous rendering of the complete model
  - Visible “details”
  - Close to UML metamodel
  - Reduced redundancy

tUML Example

```
class |Button| behavesAs SM implements I_Controls
      receives
            problems_detected_R(problems_detected) {
      private controlsManager[1-1] : ControlsManager in ControlsManager_Button;
      private id[1-1] : Integer;

      stateMachine SM {
            region MainRegion {
                  Initial -> RELEASED;
                  RELEASED -> PRESSED : button_pressed_SE /
                        opaqueBehavior = 'send pressed(id) to controlsManager;' in ABCD;;
                  PRESSED -> PRESSED : after100ms /
                        opaqueBehavior = 'send pressed(id) to controlsManager;' in ABCD;;
                  PRESSED -> RELEASED : problems_detected_SE /;
                  PRESSED -> RELEASED : button_released_SE /;
                  initial pseudoState Initial;
            }
      }
}
```

tUML Tools


Rhapsody
Input from (graphical) UML modeling tool
Rhapsody2tUML
tUML
tUML2Diagnostic
tUML2ABCD
tUML2PlantUML
Validation
Diagnostic
ABCD
PlantUML
Visualization
ABCD2Fiacre
Fiacre

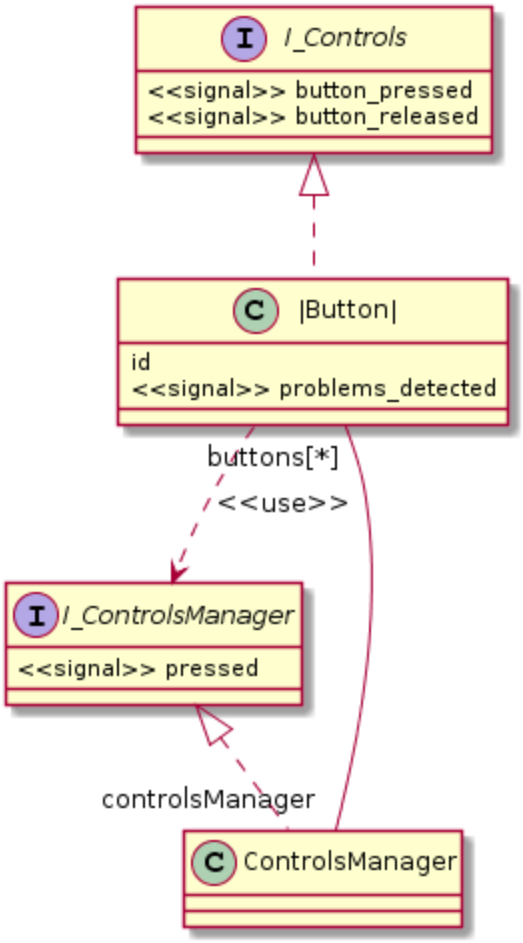
ABCD

- Intermediate format
- Halfway between UML & Fiacre
  - Reuses Fiacre syntax, but notably adds UML-like event-pool

- **A**rchitecture
- **B**ehaviors
- **C**ommunications
- **D**ata

ABCD Example

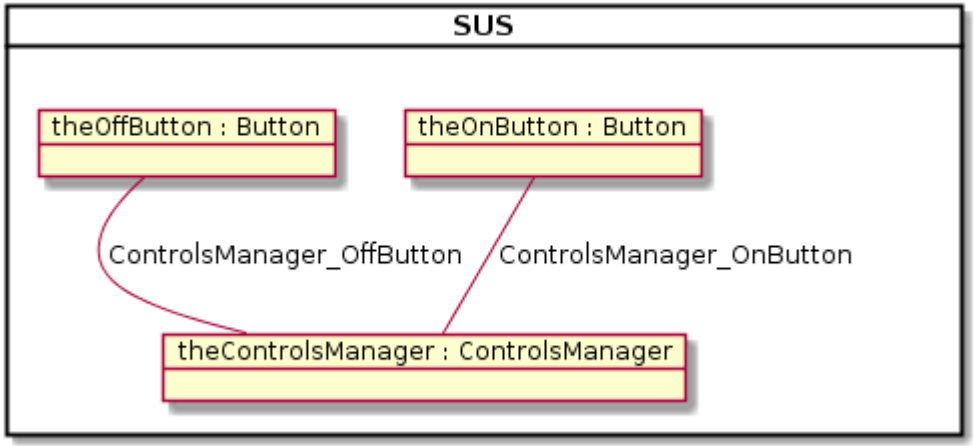
Class Diagram




```
type ButtonEvents is union
        problems_detected   //localy defined event
|       button_pressed
|       button_released
end union

type ControlsManagerEvents is union
        pressed of int
end union

channeltype ToControlsManager is eventpool buffersize: 2 of ControlsManagerEvents
channeltype ToButton is eventpool buffersize: 2 of ButtonEvents
```

ABCD Example

Composite Structure Diagram

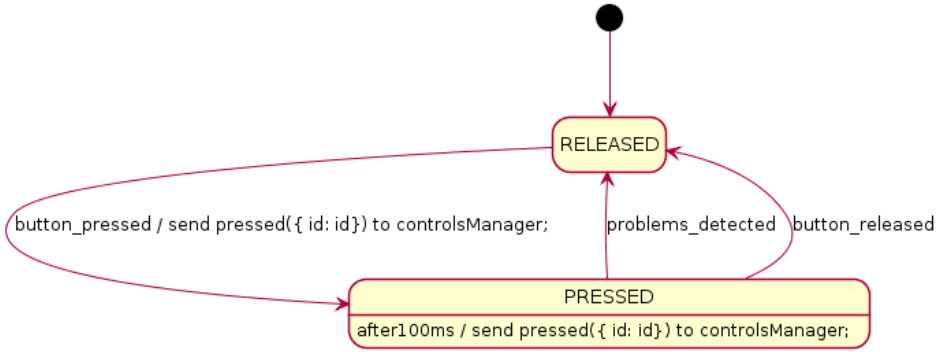



```
channel itsOnButton : ToButton
channel itsOffButton : ToButton
channel itsControlsManager : ToControlsManager
par
        [itsOnButton]          theOnButton:Button(1)        [itsControlsManager]
||      [itsOffButton]         theOffButton:Button(2)       [itsControlsManager]
||      [itsControlsManager]   theControlsManager:ControlsManager
end par
```

ABCD Example

State Diagram

```
process Button
        port
                myEventPool : in ButtonEvents;
                itsControlsManager : out ControlsManagerEvents;
        param
                id : int
is
init to RELEASED
from RELEASED
        receive ButtonEvents:button_pressed from myEventPool;
        send ControlsManagerEvents:pressed(id) to itsControlsManager;
        to PRESSED
from PRESSED
        select
                wait[100,100];
                send ControlsManagerEvents:pressed(id) to itsControlsManager;
                to PRESSED
        []      receive ButtonEvents:problems_detected from myEventPool;
                to RELEASED
        []      receive ButtonEvents:button_released from myEventPool;
                to RELEASED
        end select
```



ABCD vs. Fiacre

```
from UnsetSetting
  defer = {evtRequestEngagement}
select
...
[] receive evtSpeedA(speed) from eP;
...
end select;
```

```
from UnsetSetting
select
...
[] case (not (empty myEventPool)) of
   true ->
      myEventPool :=
            preprocessDeferred_ToActuation(
                  myEventPool, [true, false, false, ...]);
      case (first myEventPool) of
            evtSpeedA (speed) ->
            myEventPool := (dequeue myEventPool)
            end case
   end case;
 ...
end select;
```

preprocessDeferred_ToActuation

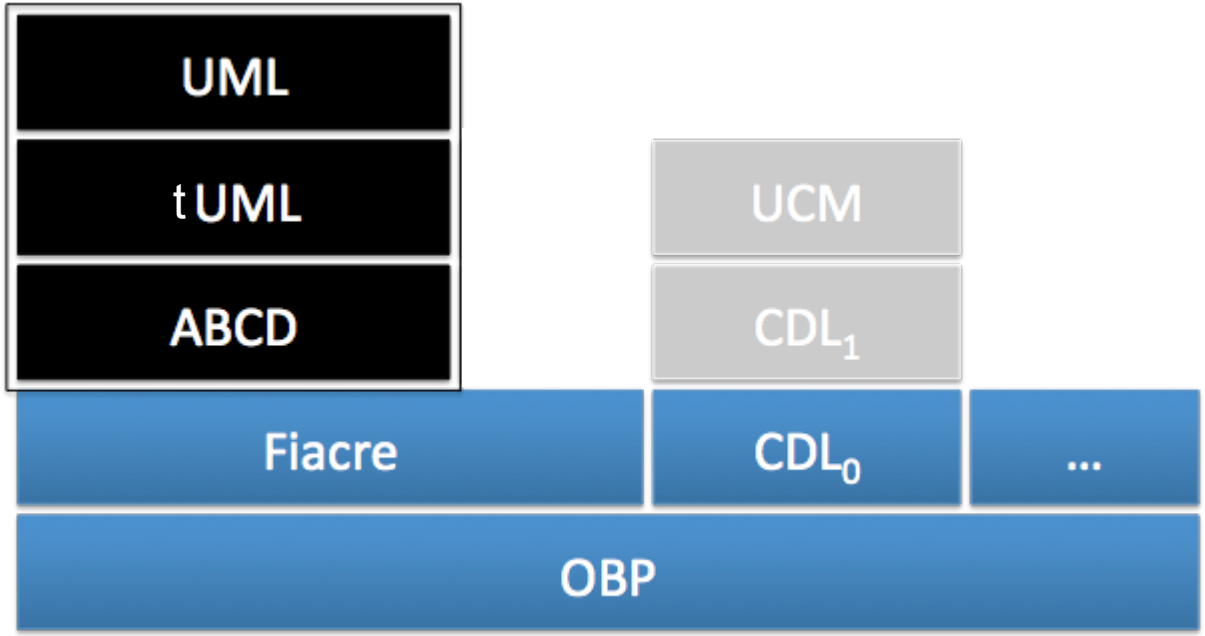
```
function preprocessDeferred_ToActuation
(eventPool: ToActuation, deferMask:
deferMask_ToActuation) : ToActuation is
var
deferred: ToActuation := {| | },
result: ToActuation := {| |},
continue: bool := true
begin
while ((not (empty eventPool)) and continue) do
                case (first eventPool) of
                evtRequestEngagment -> if deferMask[1] then

                deferred := (enqueue(deferred, (first eventPool)))

                eventPool := (dequeue eventPool)
                                else

                continue := false
                                end if
                | evtChangeActuation -> if deferMask[1] then

                deferred := (enqueue(deferred, (first eventPool)))

                eventPool := (dequeue eventPool)
                                else

                continue := false
                                end if
                | evtResetActuation -> if deferMask[2] then

                deferred := (enqueue(deferred, (first eventPool)))

                eventPool := (dequeue eventPool)
                                else

                continue := false
                                end if
                | evtPause -> if deferMask[3] then

                deferred := (enqueue(deferred, (first eventPool)))

                eventPool := (dequeue eventPool)
                                else

                continue := false
                                end if
                | evtPauseEnd -> if deferMask[4] then

                deferred := (enqueue(deferred, (first eventPool)))

                eventPool := (dequeue eventPool)
                                else

                continue := false
                                end if
                | evtSetA -> if deferMask[5] then

                deferred := (enqueue(deferred, (first eventPool)))

                eventPool := (dequeue eventPool)
                                else

                continue := false
                                end if
                | evtIncA -> if deferMask[6] then

                deferred := (enqueue(deferred, (first eventPool)))

                eventPool := (dequeue eventPool)
                                else

                continue := false
                                end if
                | evtDecA -> if deferMask[7] then

                deferred := (enqueue(deferred, (first eventPool)))

                eventPool := (dequeue eventPool)
                                else

                continue := false
                                end if
                | evtSpeedA (any) -> if deferMask[8] then

                deferred := (enqueue(deferred, (first eventPool)))

                eventPool := (dequeue eventPool)
                                else

                continue := false
                                end if
                end case
                end while;
if (not (empty eventPool)) then
                result := (enqueue(result, (first eventPool)));
                eventPool := (dequeue eventPool)
end if;
while (not (empty deferred)) do
                result := (enqueue(result, (first deferred)));
                deferred := (dequeue deferred)
                end while;
while (not (empty eventPool)) do
                result := (enqueue(result, (first eventPool)));
                eventPool := (dequeue eventPool)
                end while;
return result
end
```

Conclusion

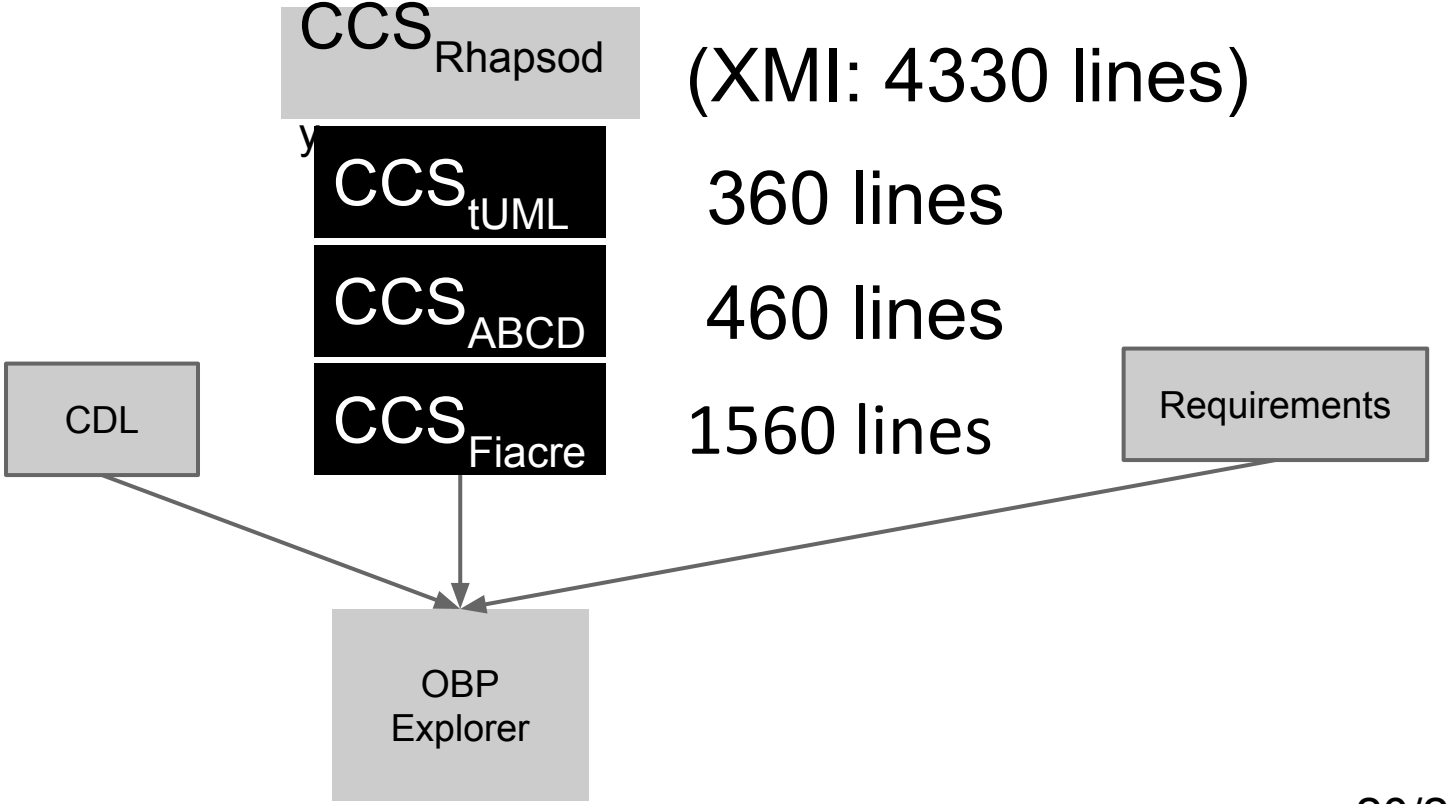
- Our tool
  - helps creating correct models
  - enables verification of UML models
    - By transformation to Fiacre
    - Supporting contexts by using OBP
  - can be extended to additional input formalisms
    - By targeting either UML/tUML or ABCD

Conclusion

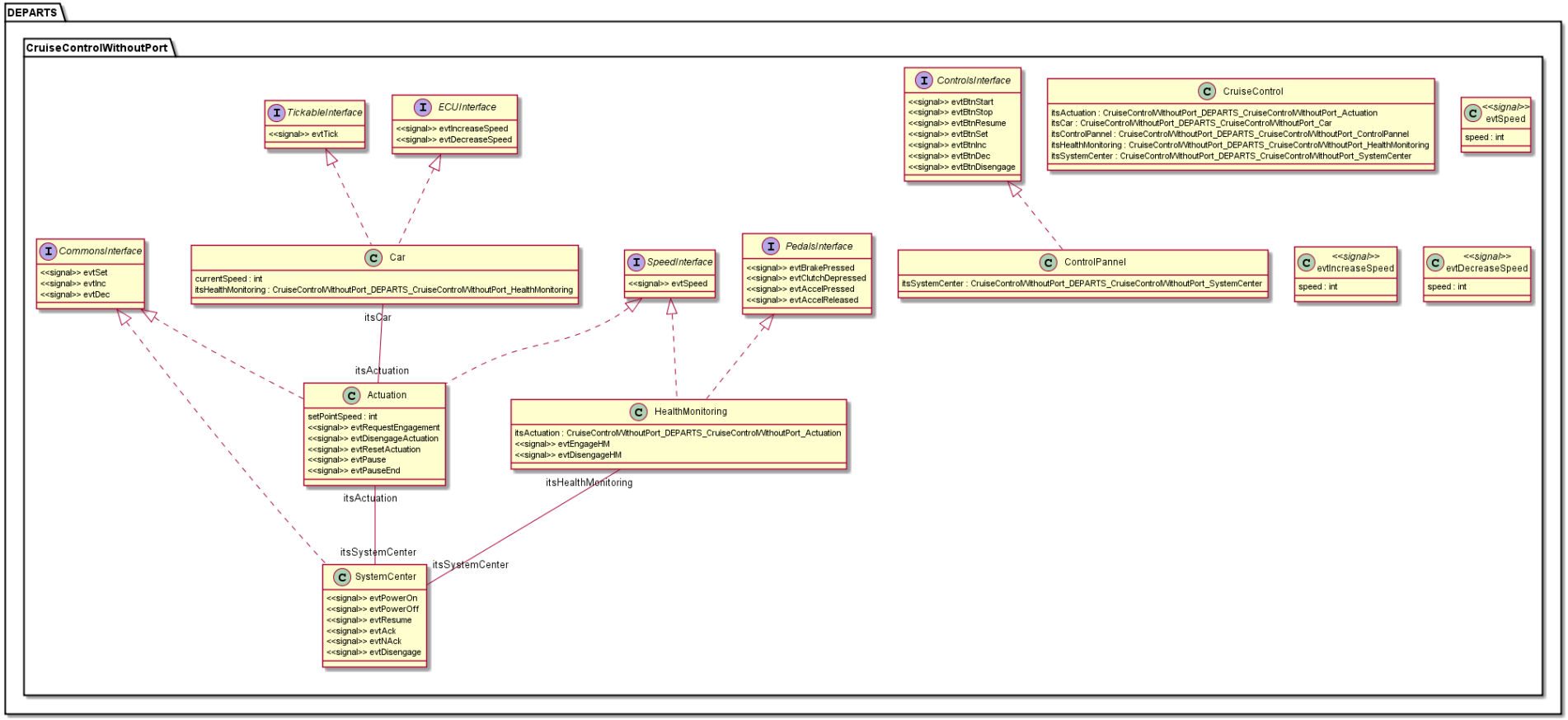
UML 2 Fiacre



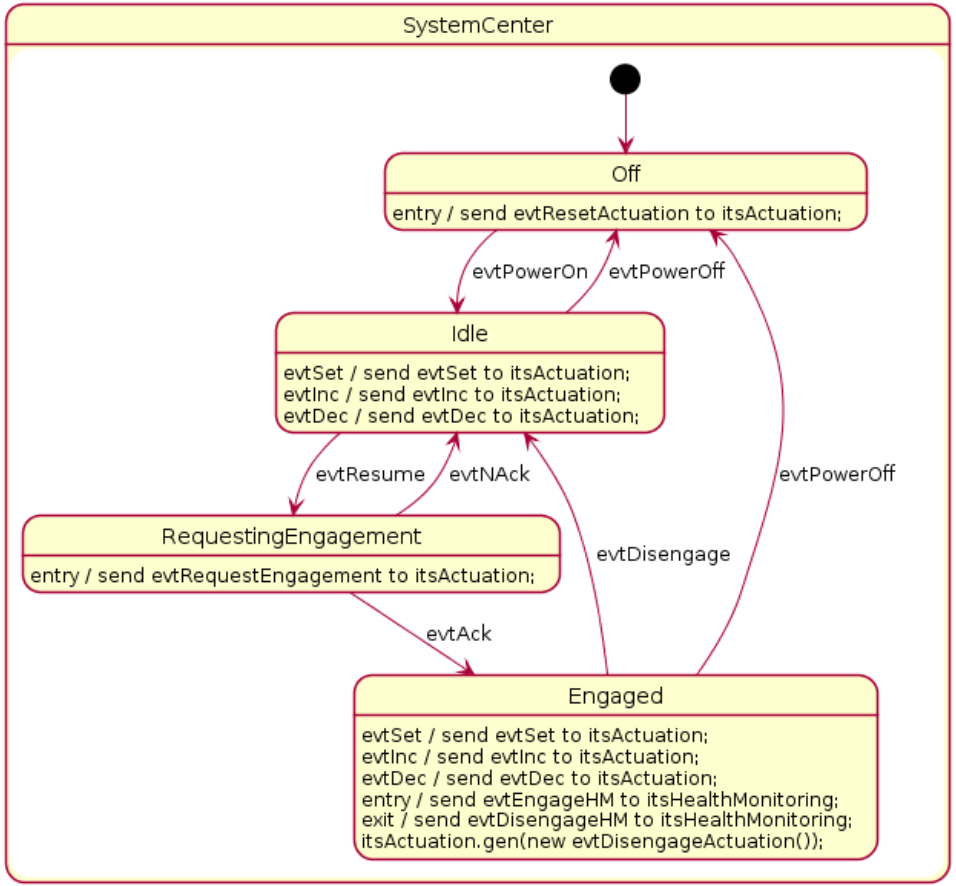
Application to
Cruise-Control System (CCS)



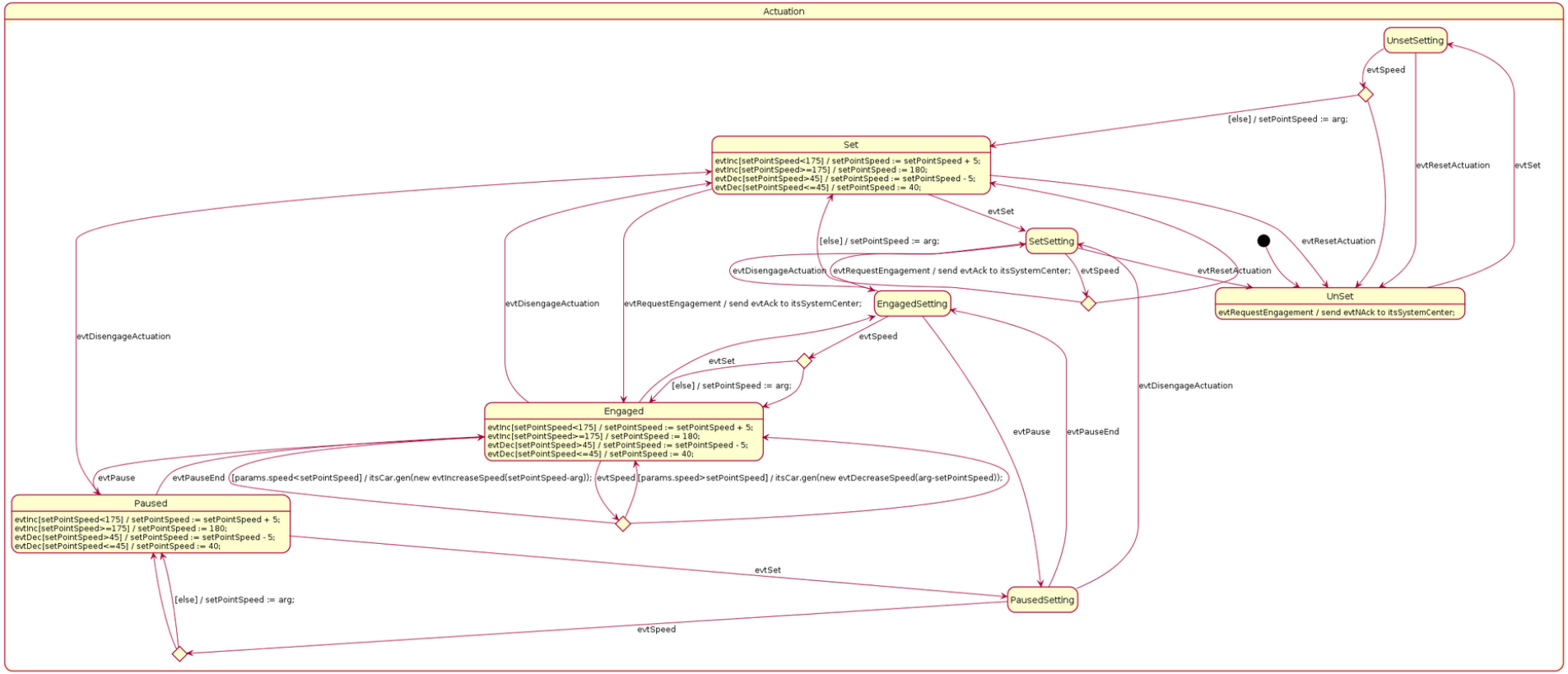
Application to CCS - Class Diagram

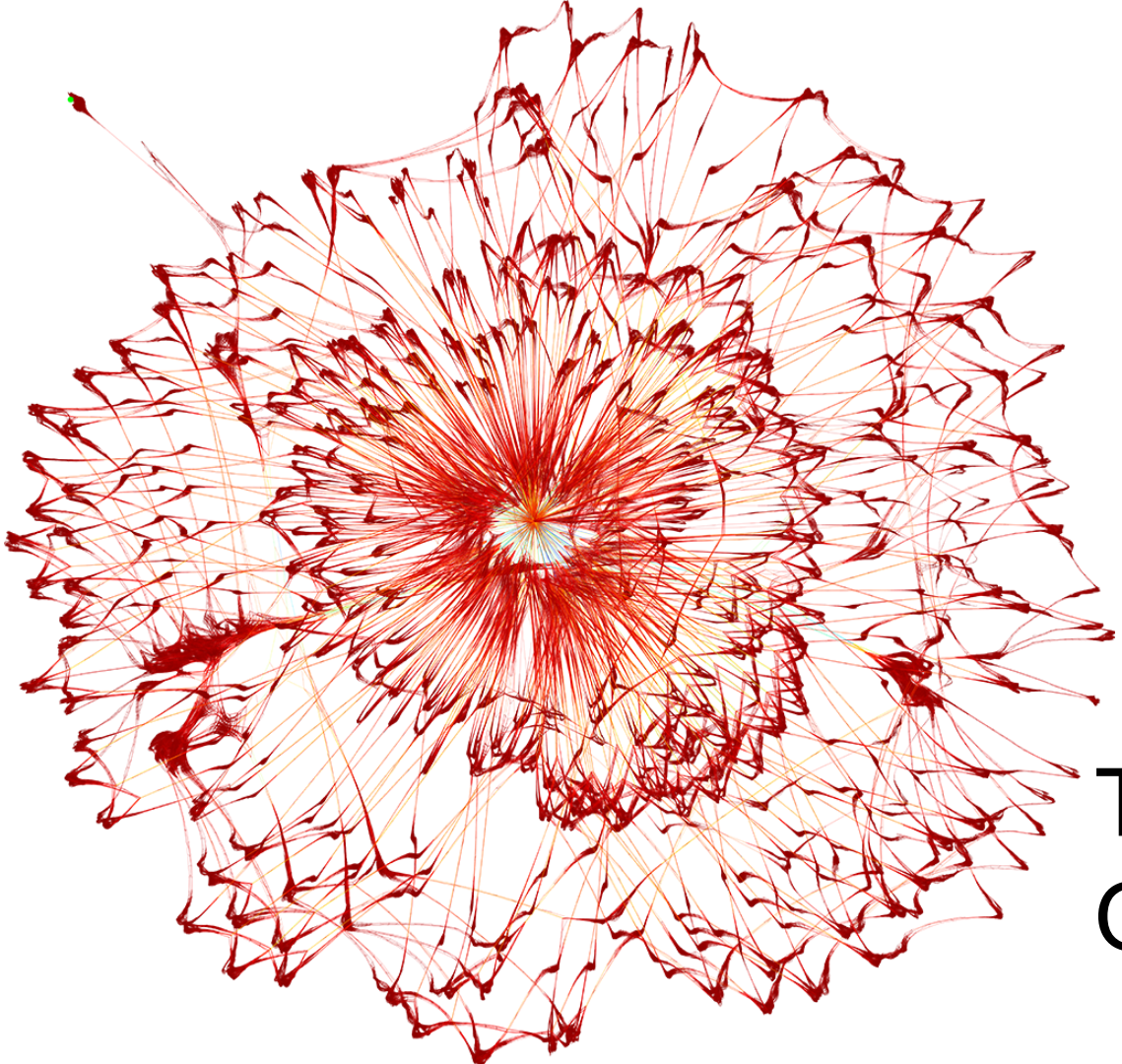


Application to CCS - State Diagram



Application to CCS - State Diagram





Thanks!
Questions?

MDD4DRES 2014

6th International School on Model-Driven Development for Distributed, Realtime, Embedded Systems
Aber-Wrach (France), September, 1-5, 2014

SYLLABUS OVERVIEW (see http://www.mdd4dres.org/program/ for details)
Lectures given by F. Bordeleau, M. DiNatale, M. Egea, R. France, P. Fritzson, J.M. Jézéquel, G. Karsai, P.A. Muller, B. Schätz, B. Selic, J. Sztipanovits, H. Vangheluwe, J.L. Voirin, ...

And a Work in Progress Session and Tools Session

Apply online http://www.mdd4dres.org before August 23th

Get a flyer (http://www.mdd4dres.org/flyer.pdf) and help us spread the world!
You can even get a printed flyer at Neptune 2014.