

Modelling and Evaluating Real-Time Software Architectures

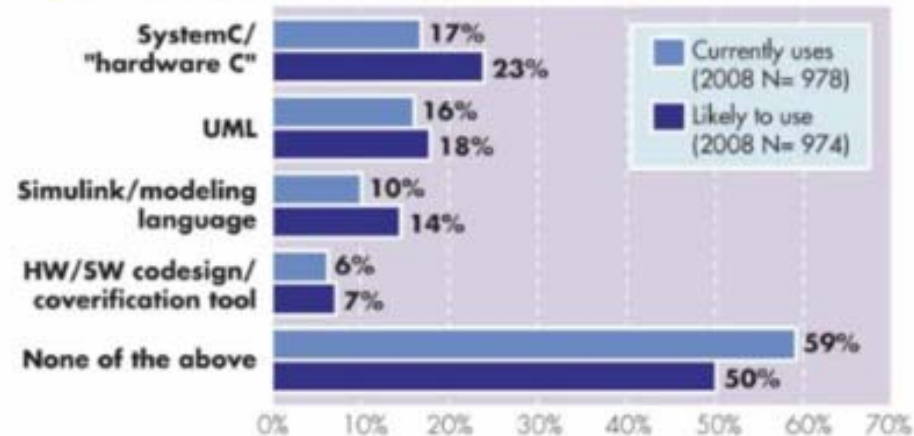
José L. Fernández Sánchez

Gloria Mármol Acitores

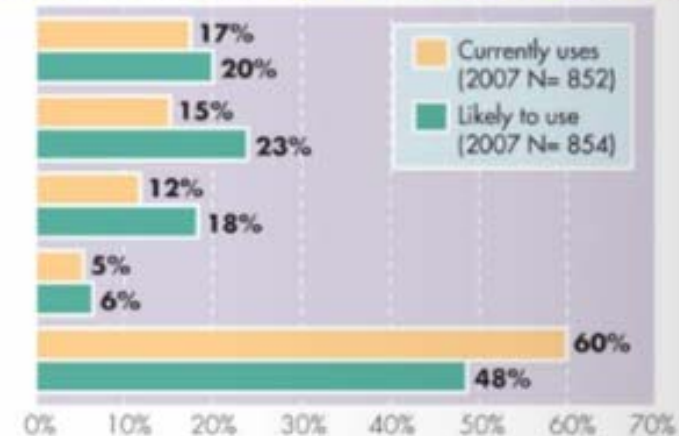
Madrid Technical University (UPM)

The facts

My current embedded project uses ...



My next embedded project is likely to use ...



Michael Barr noticed that “UML adoption remains extremely low at 16% with no expected upturn. This is disappointing after so many years of pushing by some many people and companies”. An insider's view of the 2008 embedded Market Study. Richard Nass & Michael Barr. Embedded Systems Design Europe. October 2008.

Effort devoted to trial and error in embedded systems development is still at 25%. Training in well established practices is indeed lacking and everybody has his personal definitions and best practices. Jim Turley. “Development teams are getting bigger and richer”. Embedded Systems Programming October 2005

Proposed approach

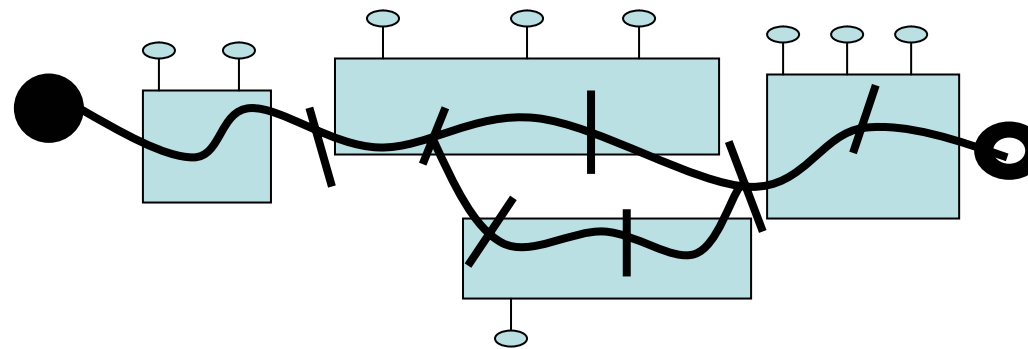
- We think that the balance between the formality level of a method and its easy adoption is the main issue for its success.
- Here we show how a model based architecting approach combined with the schedulability analysis and simulation, can be applied seamlessly to the design and evaluation of small and medium real-time systems, identifying and solving concurrency problems at the architecture phase, thus saving later testing and debugging efforts.
- Making easy its adoption by small and medium embedded systems industries is one of the main goals of the presented approach.

Scope of the presentation

- We present PPOOA-Cheddar as a complete solution for architecting real-time systems.
- We mean a complete solution that supporting:
 - a vocabulary of building elements (PPOOA-UML),
 - a process for architecting a real-time system (PPOOA_AP)
 - models of the system architecture views using PPOOA-UML notation,
 - responsiveness and schedulability assessment using analytical techniques and execution simulation (Cheddar-University of Brest).
- To illustrate the approach an example of an elevator control system is presented

PPOOA (I)

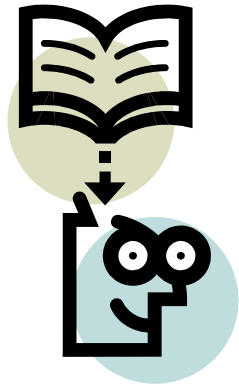
- PPOOA, “Processes Pipelines in Object Oriented Architectures” is an architectural style for concurrent object oriented architectures. It can be used when individual paths of execution are required to be concurrent and different processes may be positioned along the path to control the action.



PPOOA (II)

- A model based approach for architecting real-time systems
 - Based on **UML** notation
 - Supports a diversity of components and coordination mechanisms (for synchronization and communication) not found in UML.
 - Improves **behavioural view modelling** (“causal flow of activities” modelling) and allows performance assessment.
 - An **architecting process (PPOOA_AP)**, defining the steps to build the architecture
 - A **CASE tool (PPOOA-Visio)**

The PPOOA solution for the designer's problem



How to find the best architecture that fits the customer demands?

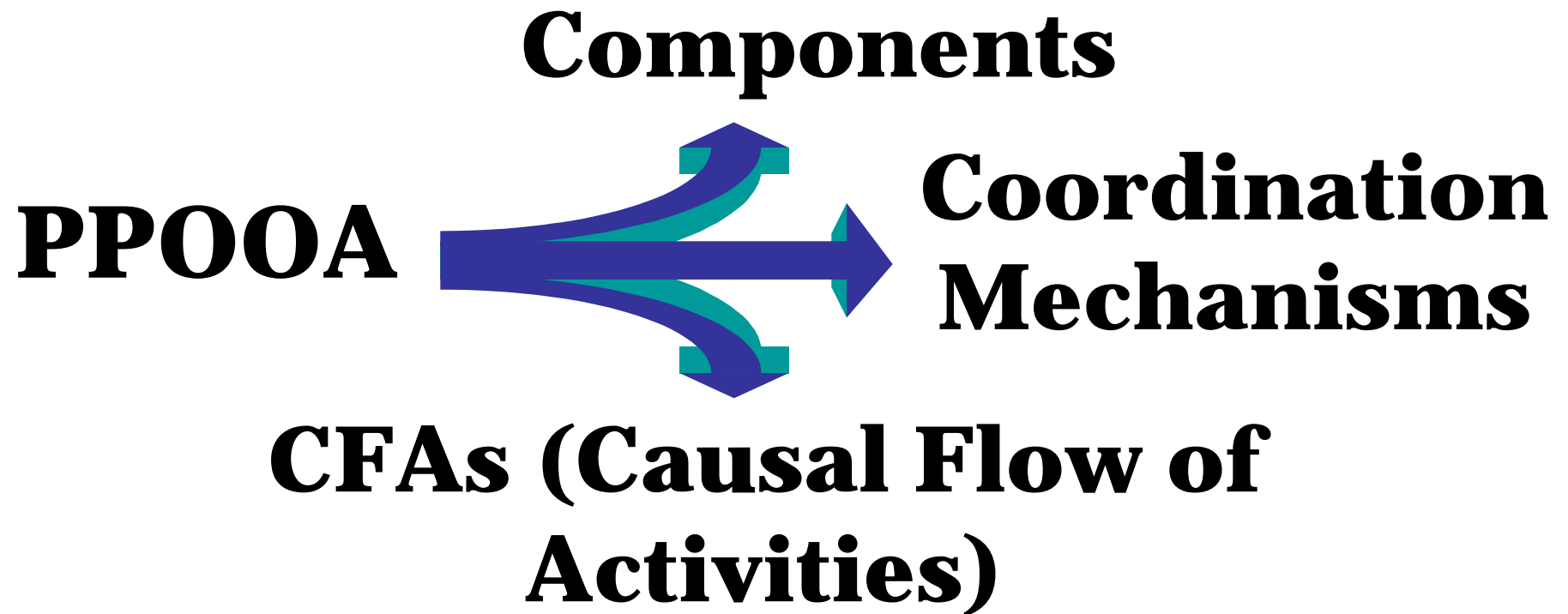
- Common building elements (PPOOA-UML vocabulary) and
- a well defined and easy to be adopted architecting process (PPOOA_AP)



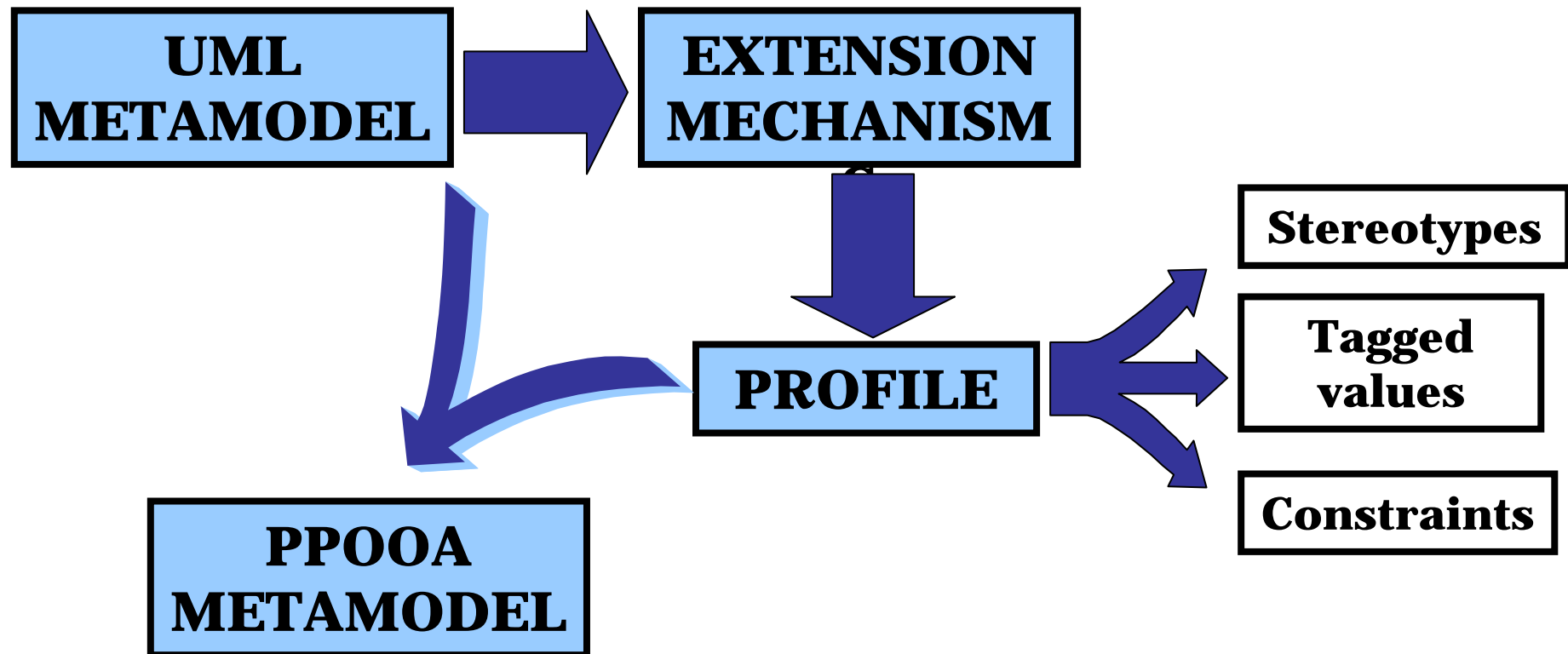
How to document and modify the architecture?

- A CASE tool (PPOOA-Visio) implementing the building elements and their representation

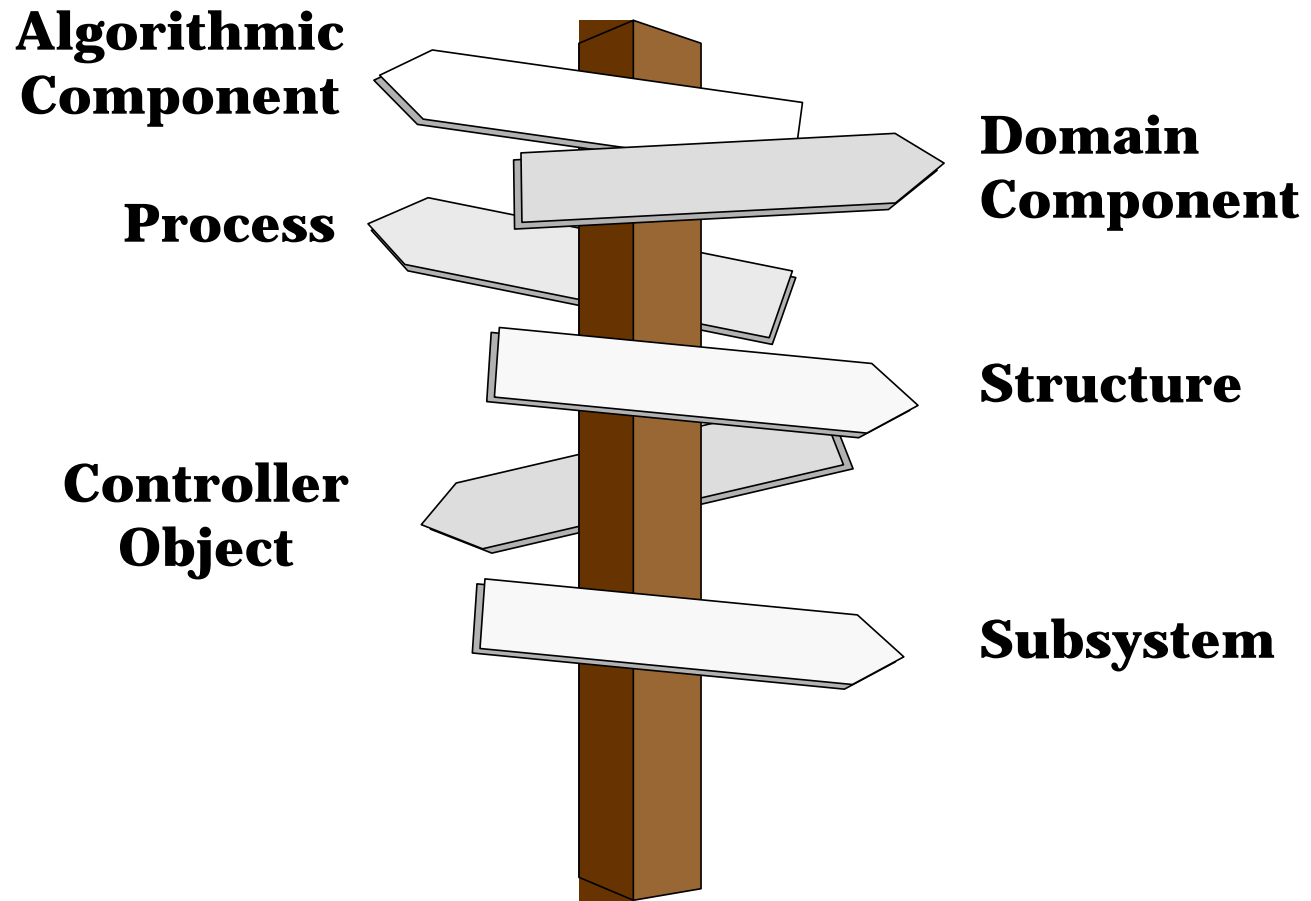
PPOOA building elements



PPOOA metamodel creation



Components



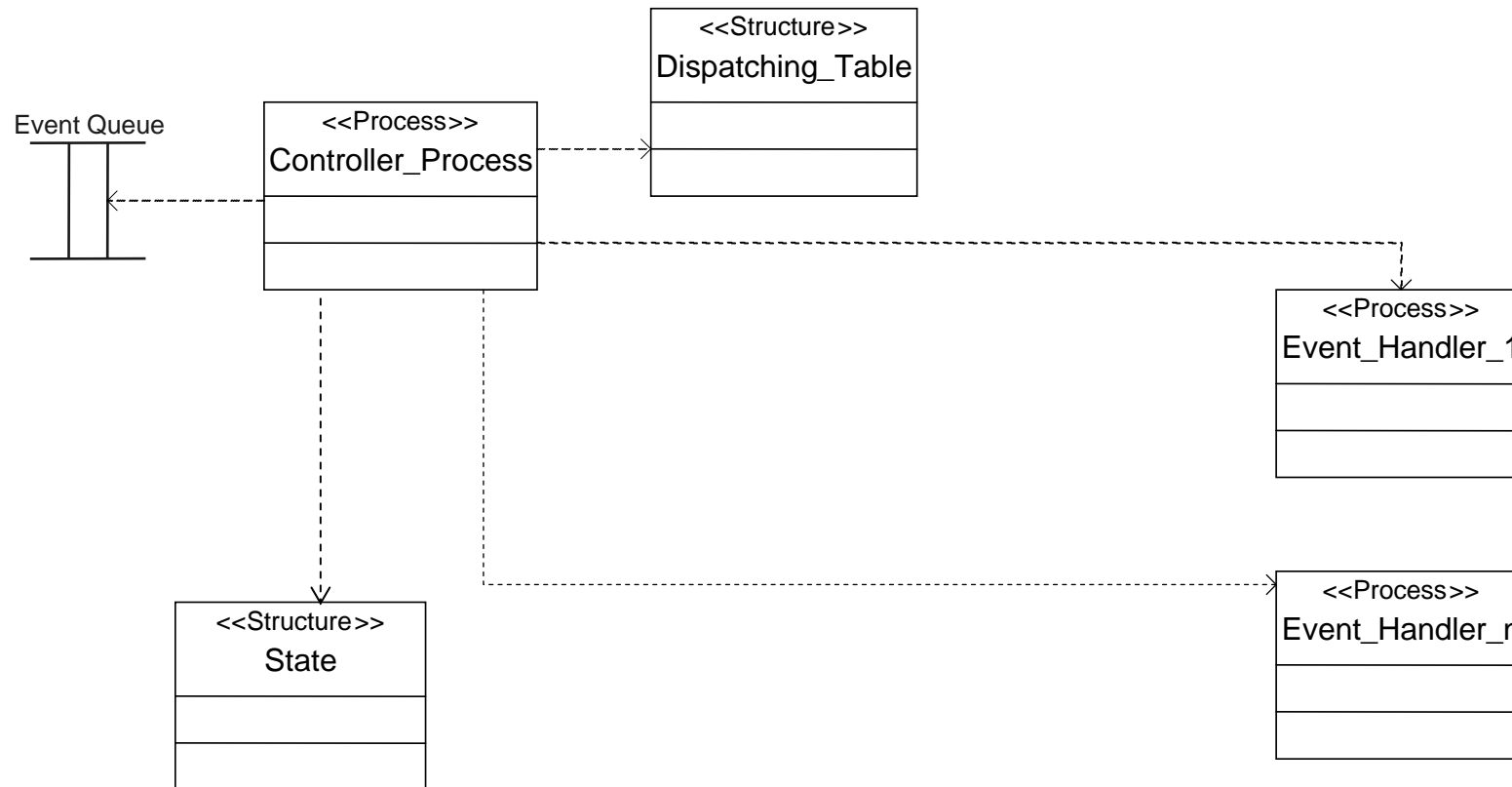
Process

- The process is a building element of the architecture that implements an activity or group of activities that can be executed at the same time as other processes. Its execution can be scheduled.
- The PPOOA style supports two different types of processes: periodic and aperiodic

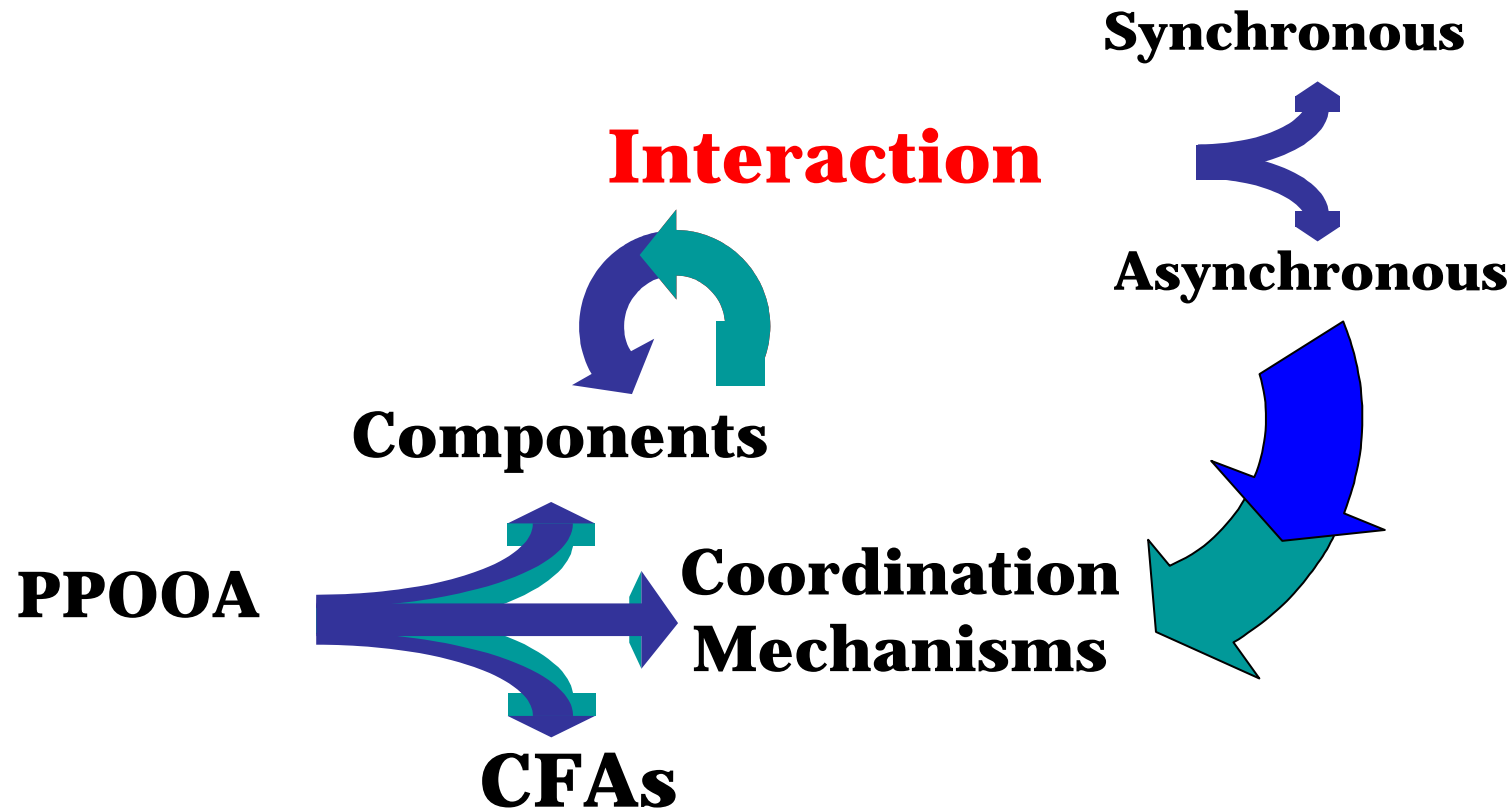
Controller Object

- The controller object is responsible for initiating and managing directly a group of activities that can be repetitive, alternative or parallel. These activities can be executed depending on a number of events or other circumstances.
- Typically, a controller object receives an event. An event is something that can happen, like a OS signal, a hardware interrupt or it represents a computed event, like airplane entering forbidden region or an alarm. When one of these events occurs, the system schedules associated event handlers.
- A controller object manages two issues: the dispatching of handlers when the event is fired, and the set of handlers associated with the event.

A Controller Object implementation



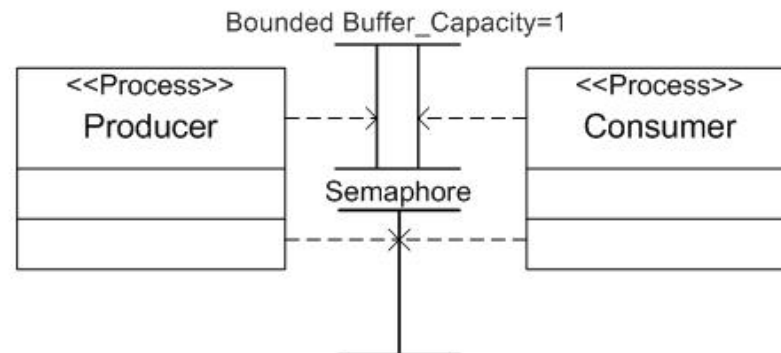
Interaction between Components (I)



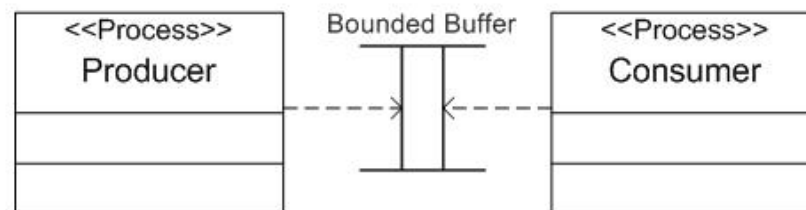
Interaction between Components (II)



Synchronous communication between passive components through operations

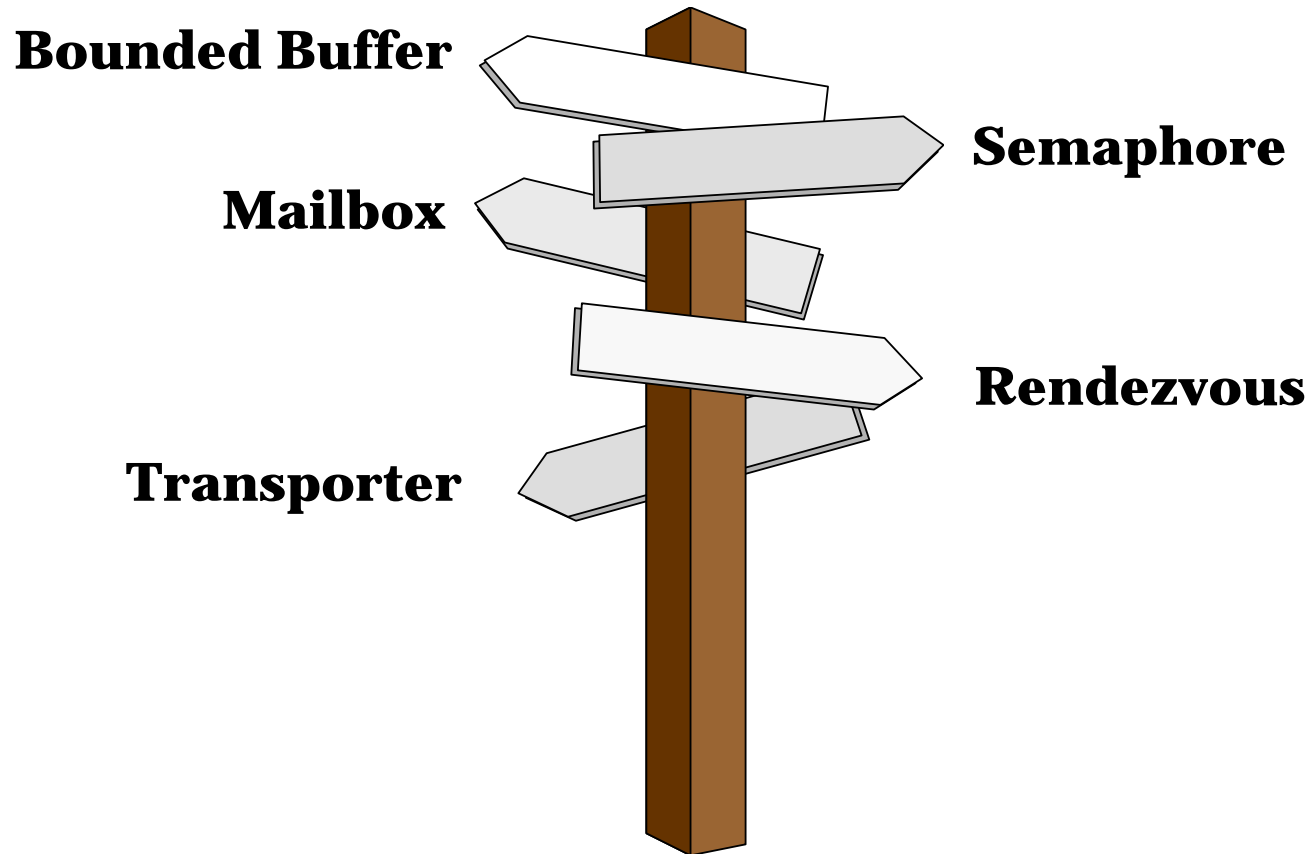


Tightly coupled message communication between active components without reply



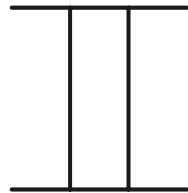
Loosely coupled or asynchronous communication between active components

Coordination Mechanisms included in PPOOA vocabulary

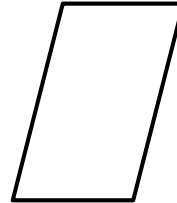


PPOOA icons for coordination mechanisms

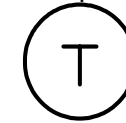
Bounded Buffer



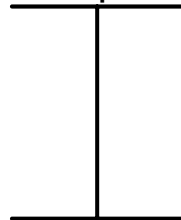
Rendezvous



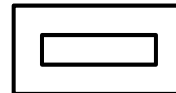
Transporter



Semaphore



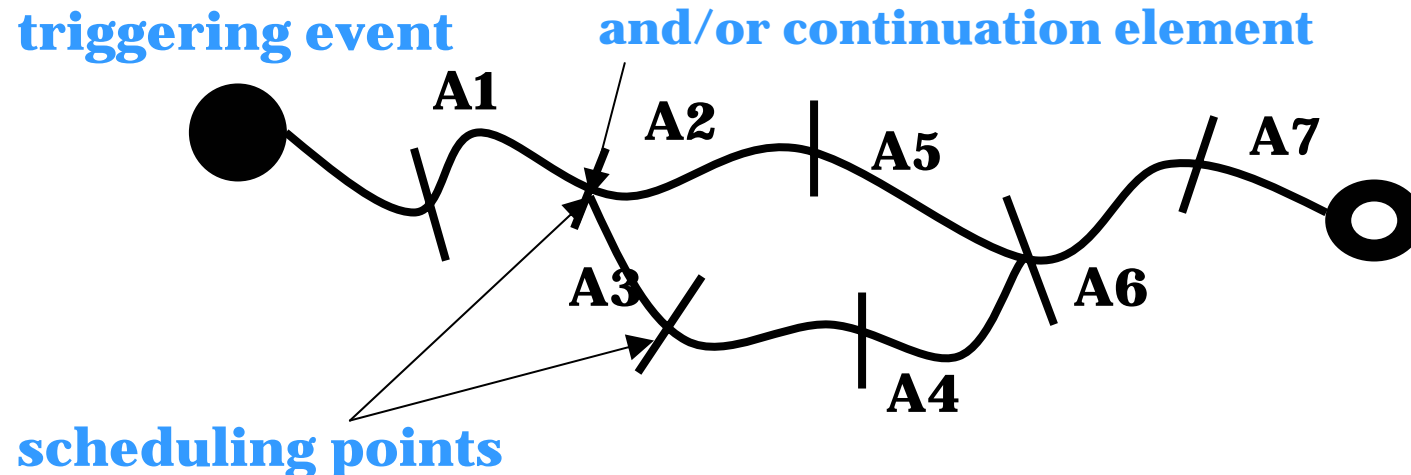
Mailbox



CFA (Response)

DEFINITION

CFA means Causal Flow of Activities. Therefore, a CFA is a chain of activities that is triggered by an event.



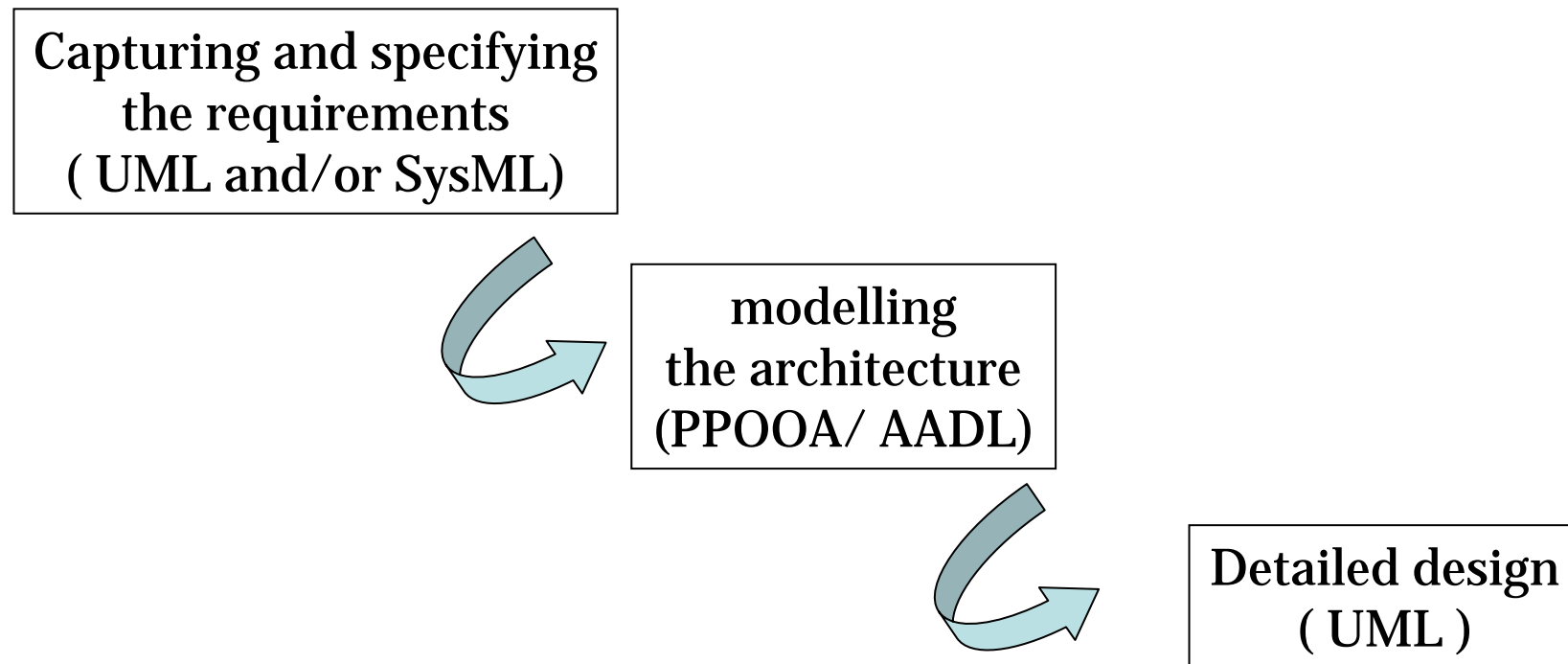
CFA Building Elements

- **Event**: something that occurs in the system or in the environment and something to which the system must react to and handle.
- **Action**: Computation block where no decision of assigning resources is taken. In PPOOA it represents an operation, task to perform, resource usage, etc.
- **Operators** that allow branching, parallel execution, etc.

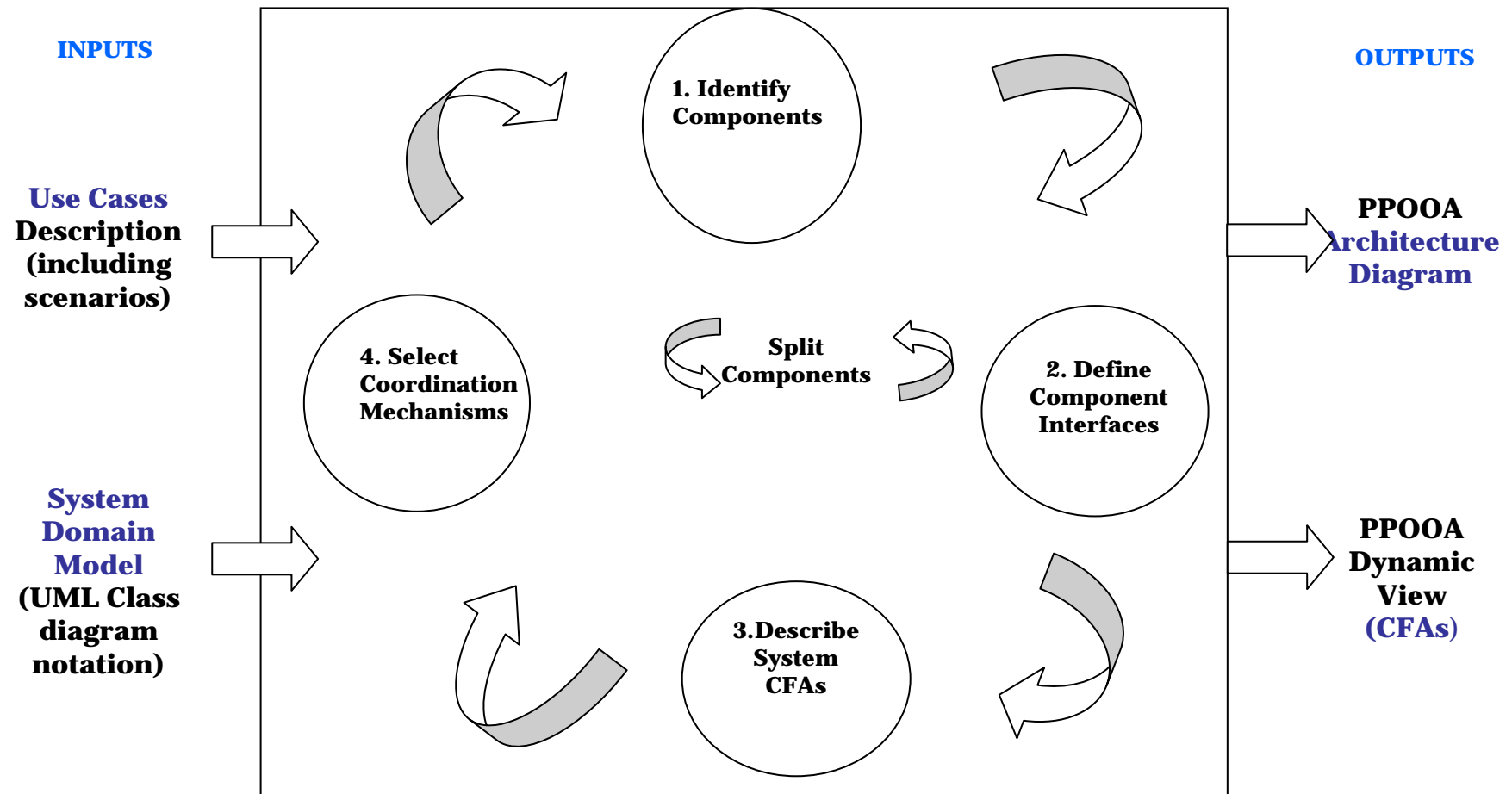
Why a new architecting process?

- Traditional Component Based Development and Object Oriented architecting approaches focus upon producing encapsulations and abstractions for system componentry.
- The effort of the resulting architecture on the ability of a system to meet its timing constraints requires additional understanding well beyond functionalities and their combined computational timing requirements.
- Concurrency modelling and synchronization behavior become a dominant concern early in the architecture development whenever time is a critical factor
- Object definition and collaboration strategies should reflect meaningful timing constraints.

PPOOA in the MDD (Model Driven Development) lifecycle



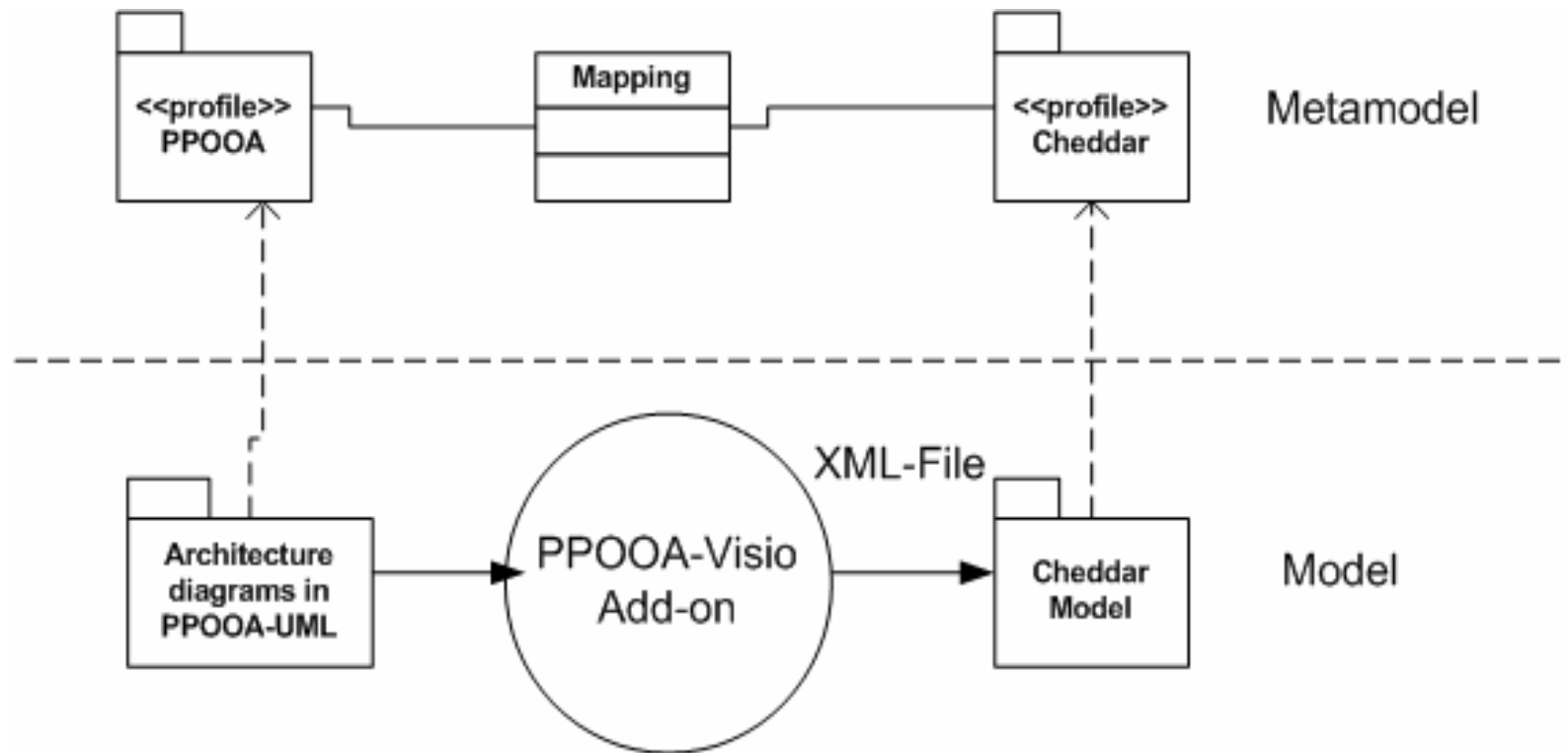
PPOOA Architecting Process



Cheddar

- Cheddar is a framework implemented in Ada by the University of Brest. It provides tools to check if a real-time system meets its temporal requirements.
- Cheddar provides real-time feasibility tests in the case of monoprocessor, multiprocessor and distributed systems.
 - The first feasibility test consists in comparing the processor utilization factor to a given bound.
 - The second feasibility test consists in comparing the worst response time of each system task with its deadline.
- Cheddar provides a simulation engine which allows the performance engineer to describe and run simulations of specific real-time systems

PPOOA-Cheddar implementation. Transforming architecture models



PPOOA-Cheddar implementation.

Building elements mapping

PPOOA	Cheddar
Process or Controller	Task
Bounded buffer	Buffer
Domain component, algorithm component or structure	Resource
Domain component, algorithm component or structure + Semaphore	Shared resource

Working with PPOOA-Cheddar tools

1. Create architecture models with PPOOA-Visio tool
2. Execute the PPOOA-XML add-on
3. The add-on automatically identifies the architecture building elements and their relations and generates an XML file of the architecture
4. The XML file is used as input to Cheddar
5. The engineer has to assign time parameters to the architecture elements
6. Cheddar runs the simulation and schedulability feasibility tests

Performance evaluation results

- Cheddar offers a simulation engine which allows the performance engineer to describe and run simulations of the architected system. When the simulation is executed, Cheddar determines for each system task and during the simulation time:
 - The number of task preemptions
 - The number of context switches
 - The blocking times
 - The missed deadlines.
- The Cheddar tool offers real-time feasibility checks based on scheduling theory for example “Rate Monotonic Analysis” (RMA), without the need of running the system. Cheddar indicates if the task set is schedulable.

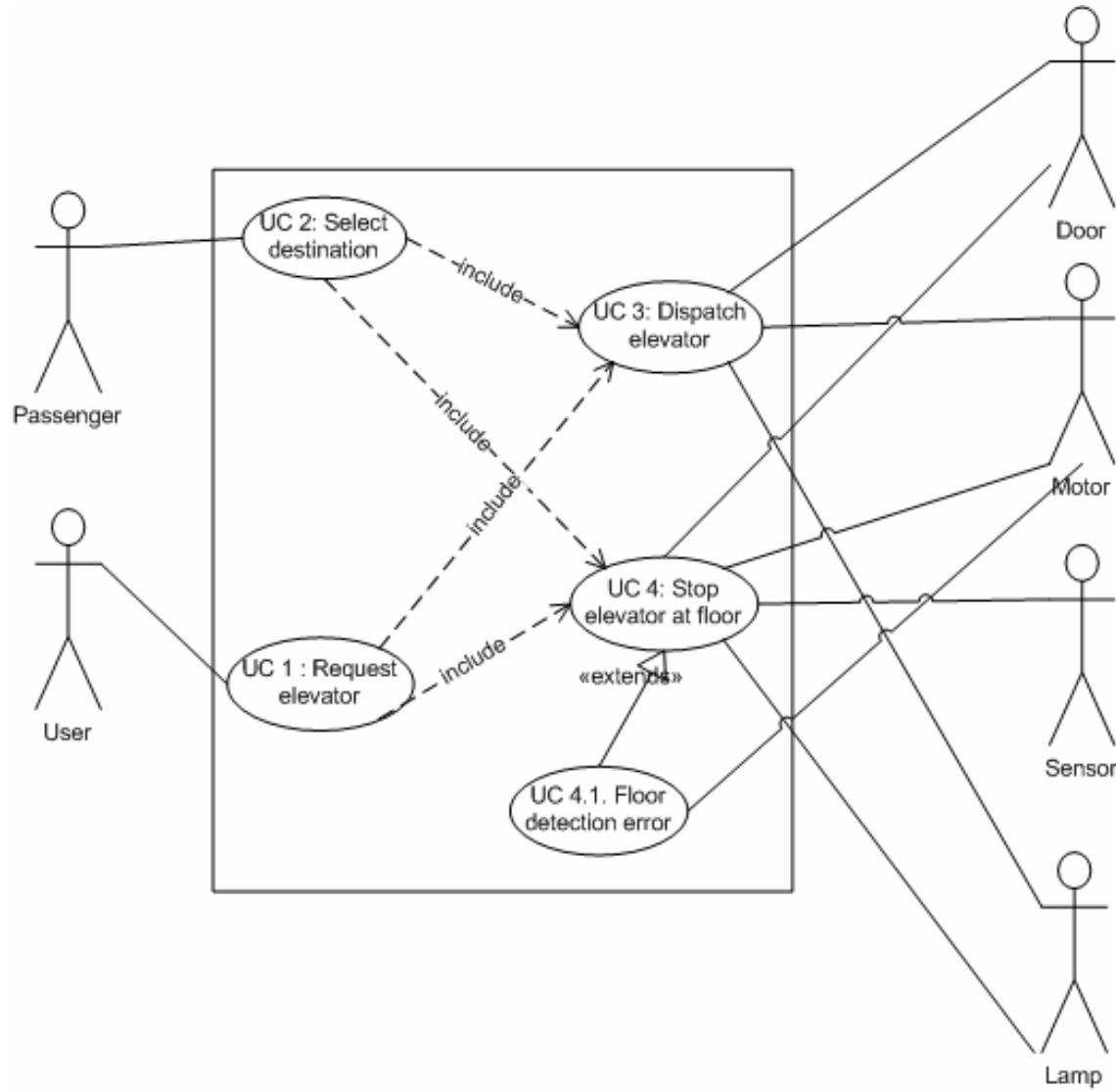
Example

Elevator Control System

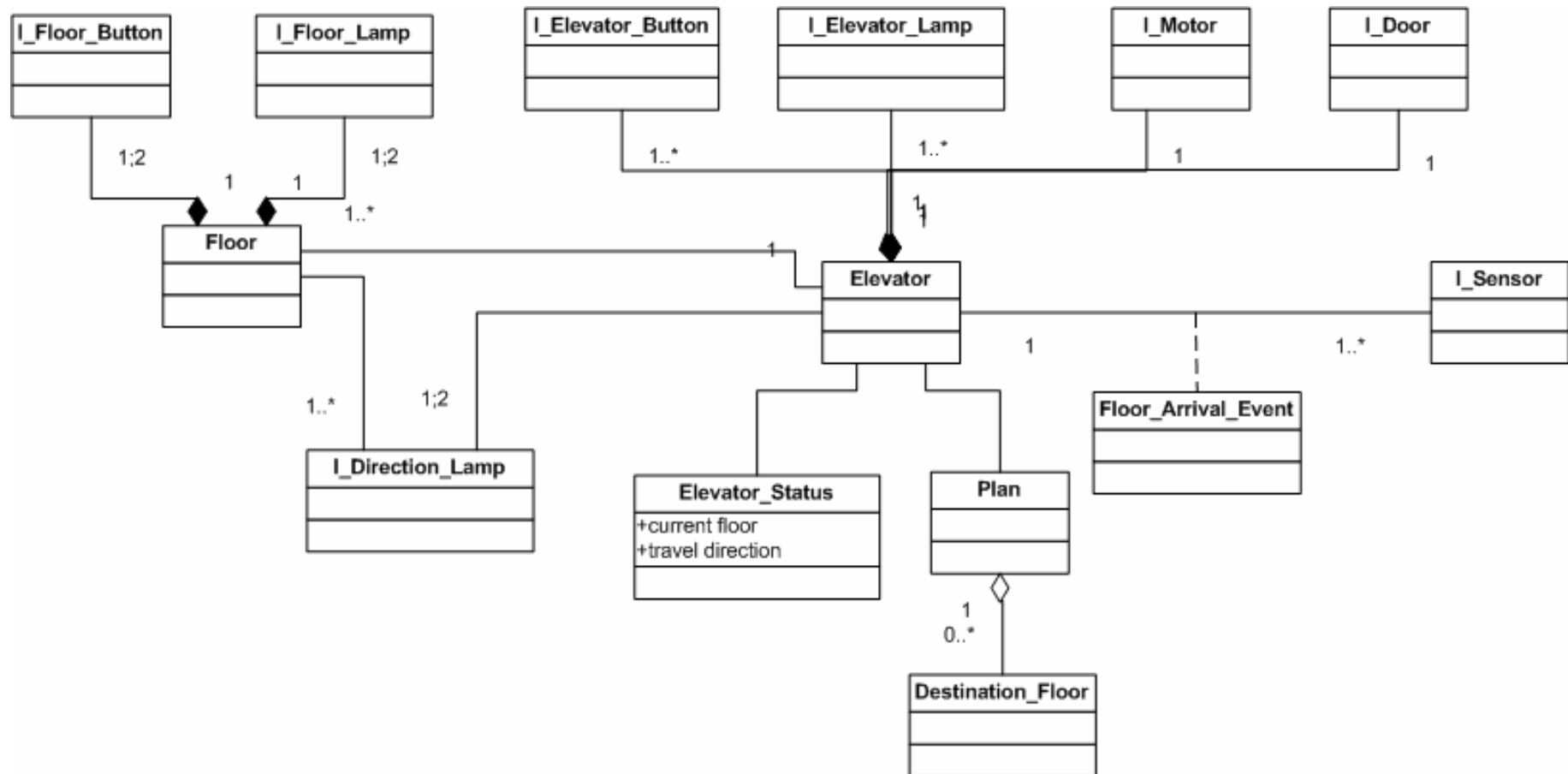
Elevator Control System (ECS)

- The system controls a single elevator which responds to requests from elevator passengers and users at various floors. Based on these requests and the information received from floor arrival sensors, the system builds a plan to control the motion and stops of the elevator.
- We consider a ten floor building, so for the house elevator, there are:
 - 10 arrival sensors, one at each floor in the elevator shaft to detect the arrival of the elevator at the corresponding floor.
 - 10 elevator buttons. An elevator passenger presses a button to select a destination.
 - The elevator motor controlled by commands to move up, move down and stop.
 - The elevator door controlled by commands to open and close it.
 - Up and down floor buttons. A user in a floor of the house presses a floor button to request the elevator.
 - A corresponding pair of floor lamps which indicate the directions which have been requested.

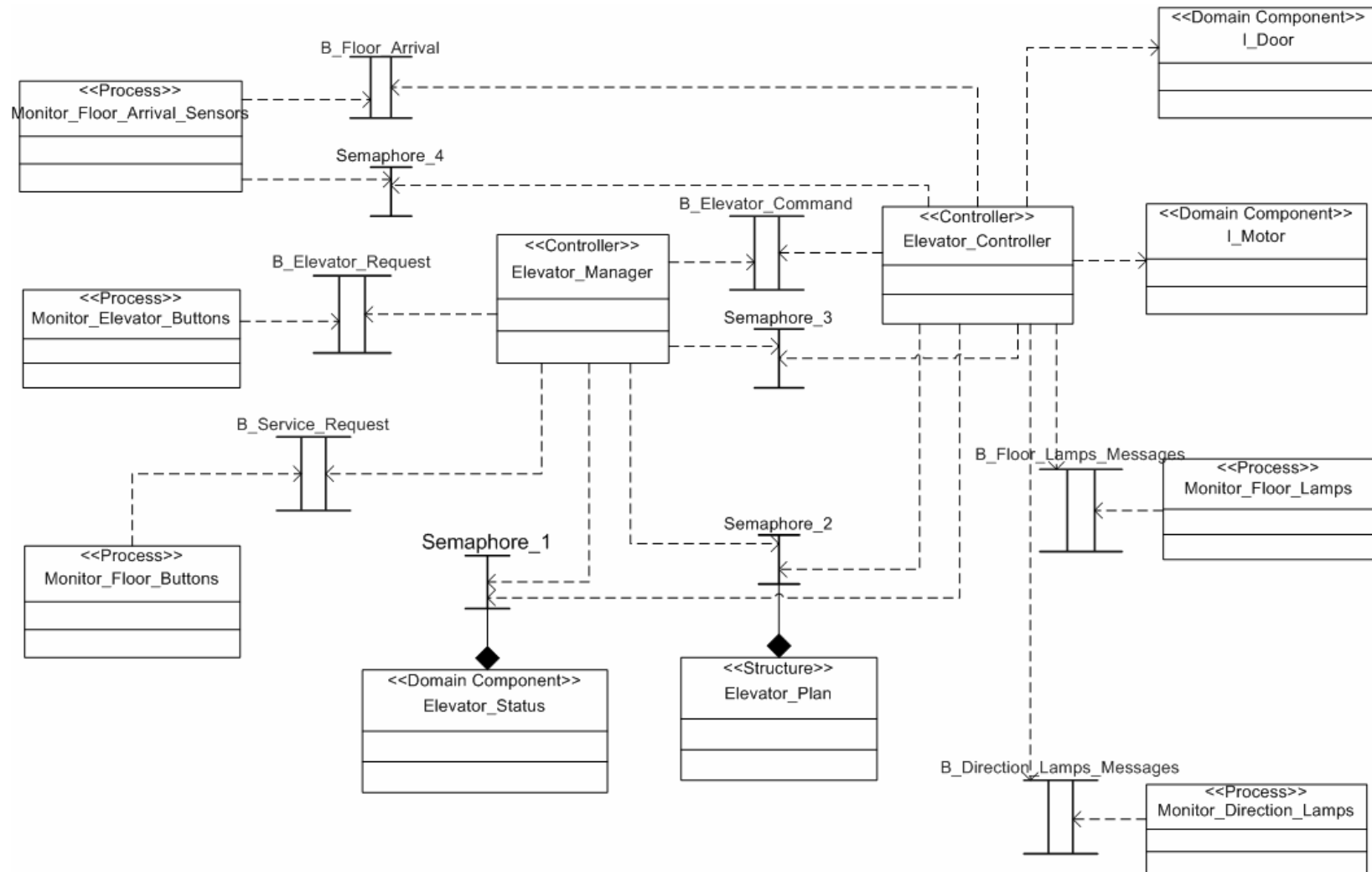
ECS use cases



ECS Domain Model



ECS Architecture Diagram

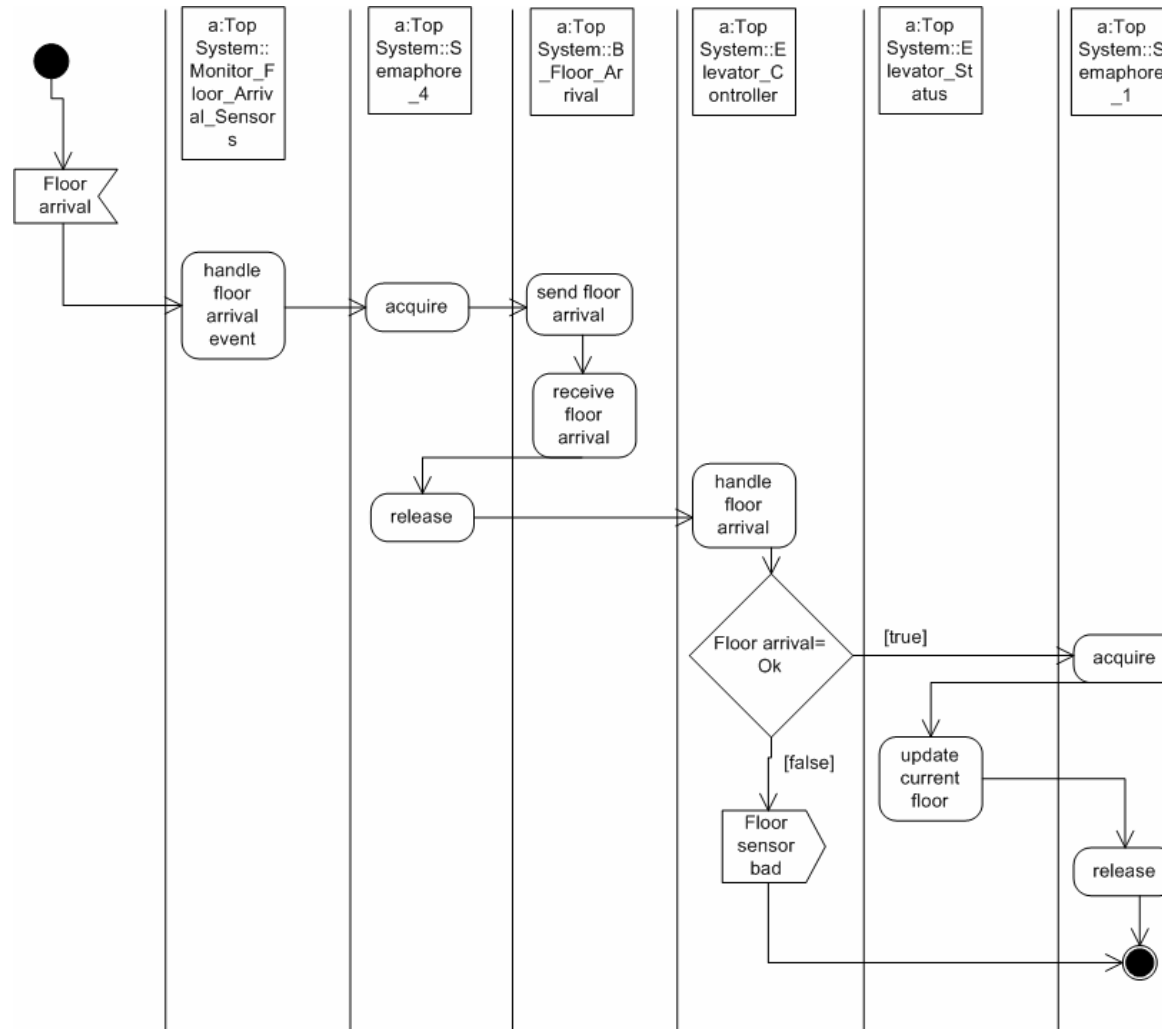


System responses of the ECS

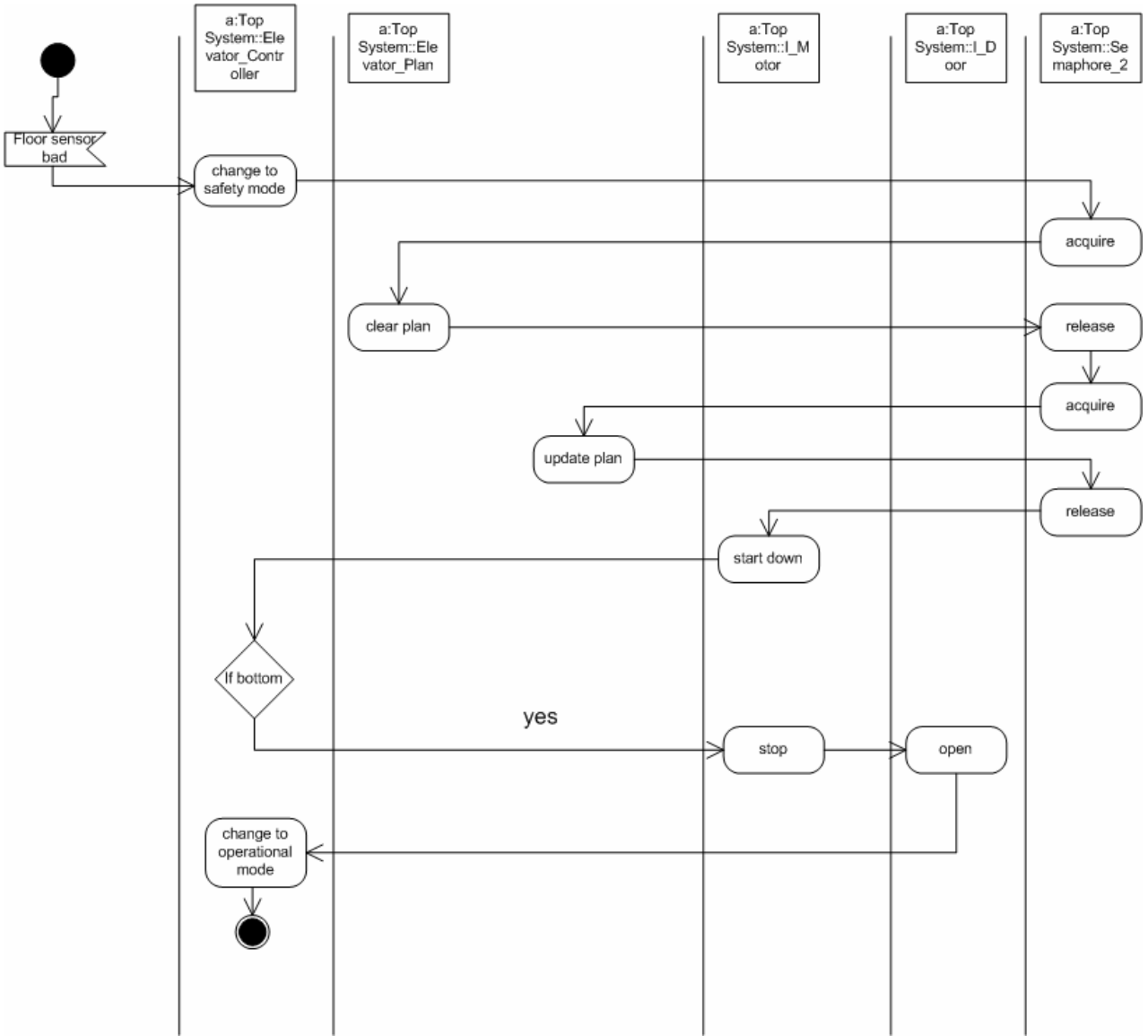
For the elevator control system we identified the following system responses and represented them as CFAs:

- CFA 1: Request floor from elevator
- CFA 2: Request elevator from floor
- CFA 3: Update current floor
- CFA 4: Stop elevator at floor
- CFA 5: Dispatch elevator to next destination
- CFA 6: Bad floor arrival sensor event

CFA 3: Update Current Floor



CFA 6: Bad floor arrival sensor event



Using Cheddar

Simulation and analysis of the
ECS architecture

Working with PPOOA-Cheddar tools

1. The ECS architecture models were created using PPOOA-Visio tool
2. We executed the PPOOA-XML add-on
3. The add-on automatically identified the architecture building elements and their relations and generates an XML file of the architecture
4. The XML file was used as input to Cheddar
5. The engineer had to assign time parameters to the architecture elements and system responses
6. Cheddar ran the simulation and schedulability feasibility tests

Numeric inputs to Cheddar

- The software engineer had to estimate the execution period and capacity of the system tasks, see table below.
- Also he or she had to estimate the time instant each task begins using each buffer and resource. The CFAs or system responses are the inputs used for this usage estimation.

Task name	Type	Capacity	Period	Deadline
Elevator_Controller	periodic	20	50	50
Elevator_Manager	periodic	50	100	100
Monitor_Floor_Lamps	periodic	5	1000	1000
Monitor_Elevator_Buttons	periodic	2	500	500
Monitor_Direction_Lamps	periodic	5	500	500
Monitor_Floor_Arr_Sensors	periodic	2	1000	1000
Monitor_Floor_Buttons	periodic	4	200	200

ECS Performance Evaluation

The screenshot displays the Cheddar scheduling simulator interface. The top window shows three task timelines for 'my_processor':

- Task name=Elevator_Controller**: Period= 50; Capacity= 20; Deadline= 50; Start time= 0; Priority= 7; Cpu=my_processor. The timeline shows a single execution bar from time 0 to 50.
- Task name=Elevator_Manager**: Period= 100; Capacity= 50; Deadline= 100; Start time= 0; Priority= 6; Cpu=my_processor. The timeline shows a single execution bar from time 0 to 100.
- Task name=Monitor_Direction_Lamps**: Period= 500; Capacity= 5; Deadline= 500; Start time= 0; Priority= 4; Cpu=my_processor. The timeline shows a single execution bar from time 0 to 500.

The bottom window displays the following text:

```
Scheduling feasibility, Processor my_processor :
1) Feasibility test based on the processor utilization factor :

- The base period is 1000 (see [1], page 6).
- 59 units of time are unused in the base period.
- Processor utilization factor with deadline is 0.94100 (see [1], page 6).
- Processor utilization factor with period is 0.94100 (see [1], page 6).
- In the preemptive case, with RM, we can not prove that the task set is schedulable because the processor
  utilization factor 0.94100 is more than 0.72863 (see [1], page 16, theorem 8).

2) Feasibility test based on worst case task response time :

- Bound on task response time : (see [2], page 3, equation 4).
  Monitor_Floor_Arrival_Sensors => 198
  Monitor_Floor_Lamps => 196
  Monitor_Direction_Lamps => 191
  Monitor_Elevator_Buttons => 96
  Monitor_Floor_Buttons => 94
  Elevator_Manager => 90
  Elevator_Controller => 20
- All task deadlines will be met : the task set is schedulable.
```

To Conclude

- The evaluation of real-time systems at earlier phases of their development is possible and suitable. The architecture structural and behavioural views may be used to perform unbounded blocking detection and schedulability analysis to identify possible missed deadlines.
- The accuracy of this analysis is lower than the accuracy that may be obtained in later phases of development when the code and execution platform are available, but the earliest detection of problems and the evaluation of design alternatives are very important benefits of this approach.
- A tool based approach for architecting and schedulability analysis and execution simulation has been presented here. The tool integration is based on the tools metamodels mapping and on the data sharing integration approach.
- It is easy to adopt because it is based on the use of diagrams that appeal to practitioners and help them tackle software architectures for RTS.