Making Ideas a Reality

A Real Time Modeling Example: The HIDOORS project

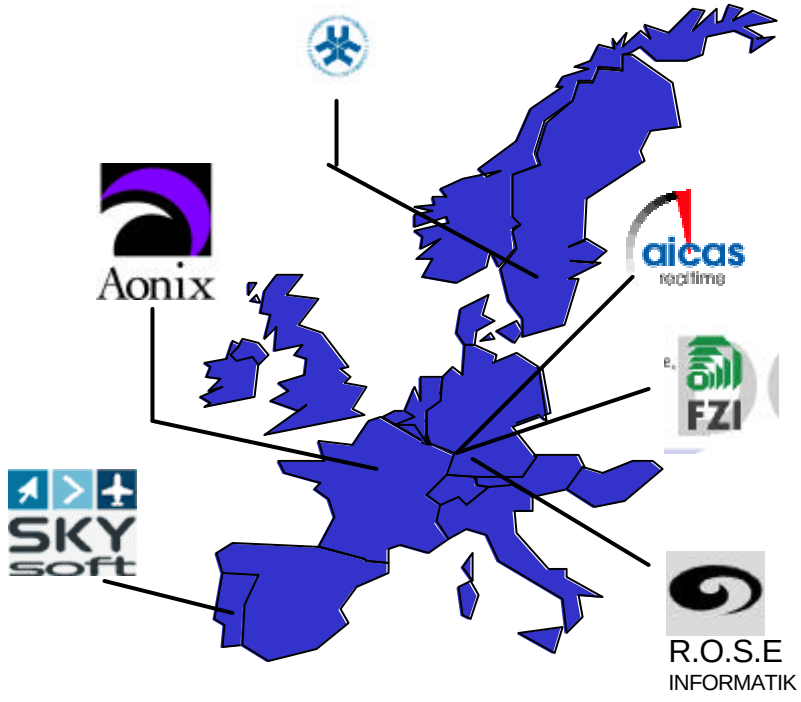
Ravi Jadhav, jadhav@aonix.com
Aonix

Plan

- Introduction
  What is “HIDOORS” ?
- The “Hidoors” profile
  Goals
  UML - SPT profile
  An example
- ARINC-653
  Communication mechanisms
  Examples
- MDA and Code Generation
  MDA, profiles and automatic code generation
- Conclusion

What is HIDOORS ?

- HIDOORS = High Integrity Distributed Object Oriented Realtime Systems
- Website: www.hidoors.org
- European project (IST)
- Duration: 30 months
- Start: January 2002
- Consortium:
  - FZI university
  - Linköpings university
  - AICAS
  - AONIX
  - R.O.S.E Informatik
  - SKYSOFT



A project divided into 2 parts

- A real time Java Platform
  A “Jamaica” virtual machine dedicated to critical and embedded RT systems
  (RTSJ implementation)

- Real Time modeling
  How to model critical and embedded RT systems ?
  In this presentation, we focus on that part

Goals of the Hidoors profile

- Goal:
  - To be compliant with the SPT profile
  - To provide concepts enabling to specify a RMA view (Rate Monotonic Analysis) of the model
  - To provide concepts enabling to specify a task view (and inter-tasks communication) of the model
  - To increase the abstraction power particularly for specifying communication between tasks
  - …

Goals of a profile

- To give a deep and non ambiguous semantics to models
- To reduce model complexity and to increase the expression power

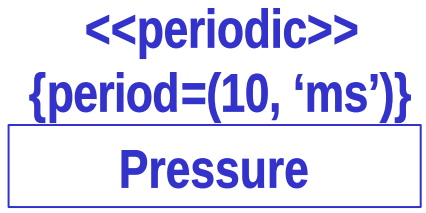
⇒ model specification more easy

⇒ model readability improved

⇒ automatic code generation more efficient

UML Profile

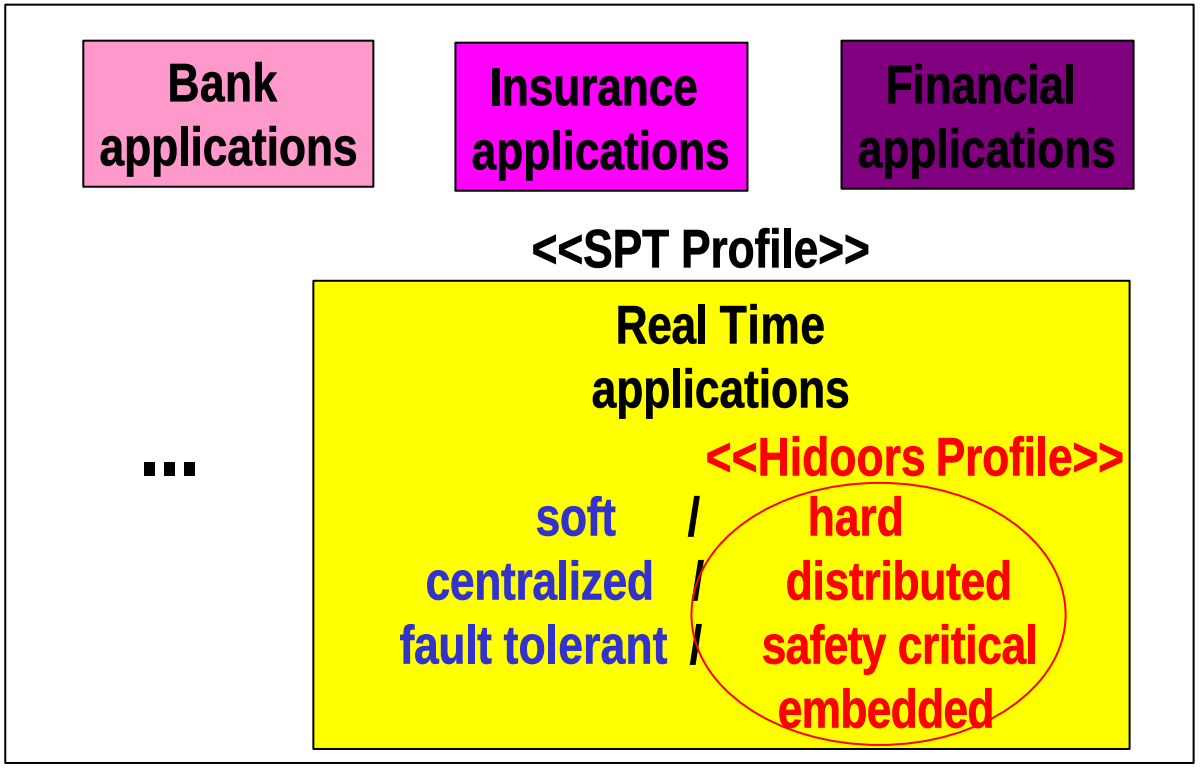
- A profile is a kind of UML “customization”
- A profile provides the context of use of UML for a given domain or project
- A profile is defined by:
  - A subset of UML
  - Some UML extensions (stereotypes, tagged values and constraints)
  - Some rules



UML profiles for Real Time

- A profile dedicated to RT systems has been adopted by the OMG in march 2002: "profile for Schedulability, Performance and Time" (**SPT**)
- The problem of the SPT profile is that it is too general and does not make any distinctions between RT applications
- A profile for Hidoors ("Hidoors Profile") has been defined as a sub profile of SPT to address critical and embedded RT applications

A Profile for critical RT systems

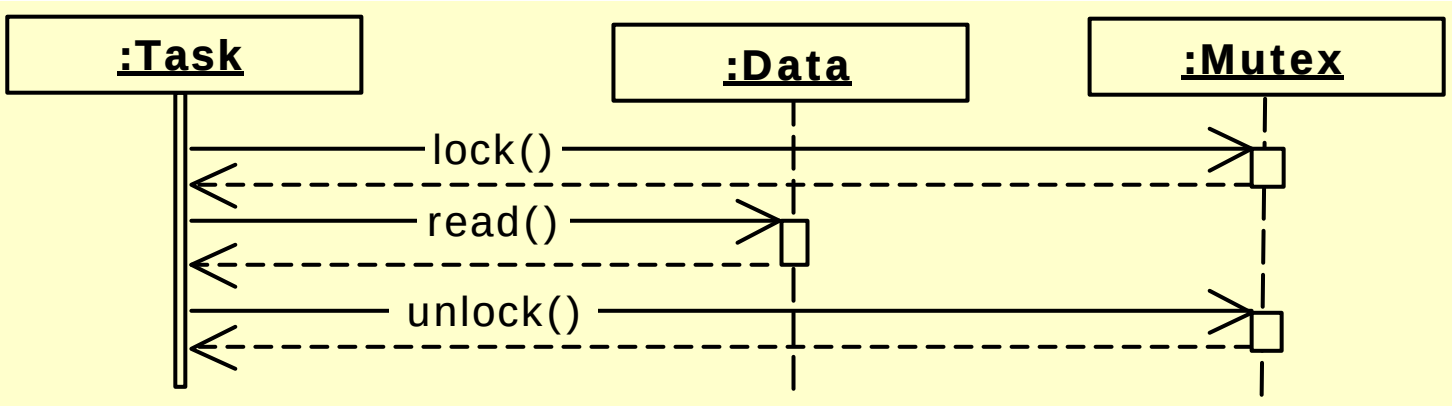


U
M
L

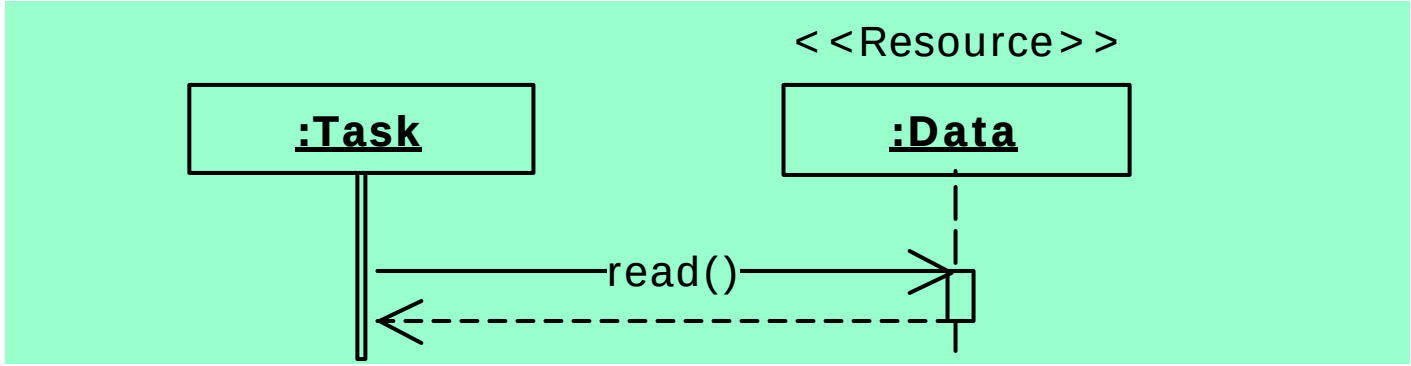
An example

- Ex: exclusive access to a shared resource

With no Profile



With Profile


<<Resource>>
:Task
:Data
read()

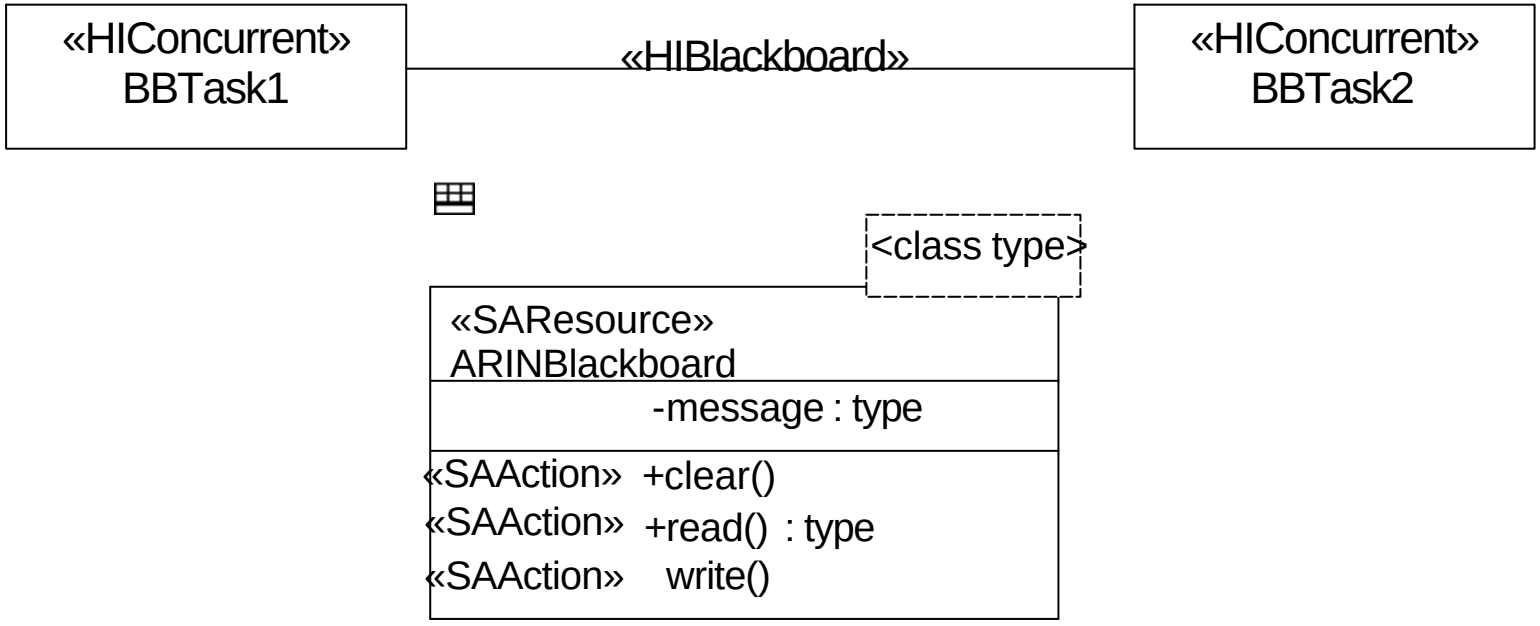
Inter-task communication / ARINC 653

- The SPT profile does not supply any high level concepts to specify communication between tasks

  => Creation of new concepts to specify this communication – 3 kinds of communication (from ARINC 653) :
  - asynchronous by backboard (<<HIBlackboard>>)
  - asynchronous by buffer (<<HIBuffer>>)
  - synchronous by event (<<HIEvent>>)

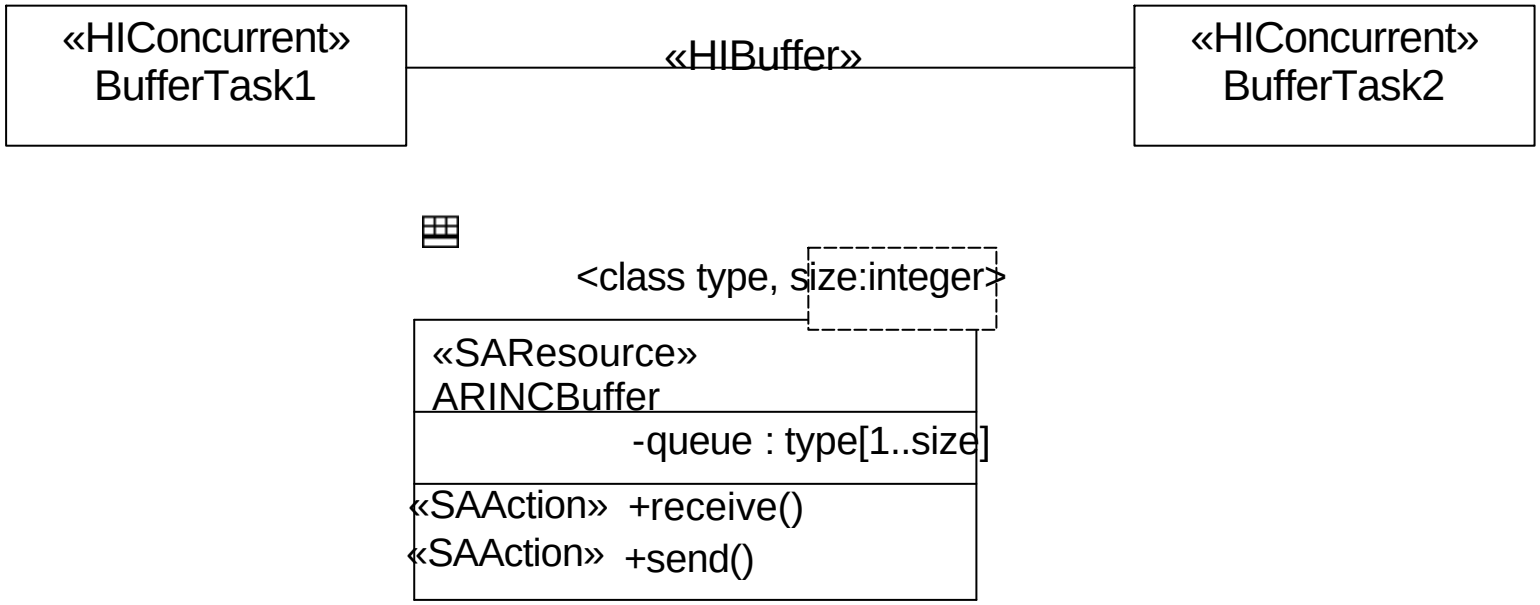
Blackboard

Blackboard: No queueing of messages. A message is put in a board and is either received or gets overwritten by the next message.



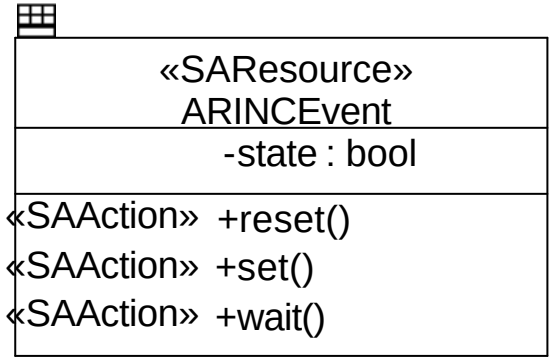
Buffer

Buffer: Messages are transmitted via queues with predefined capacity in FIFO order



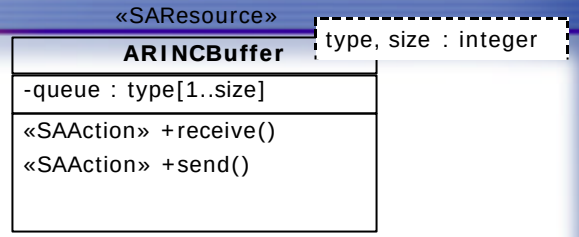
Events

Events: For notification of processes, which wait for them.
Two values: „up“ and „down“

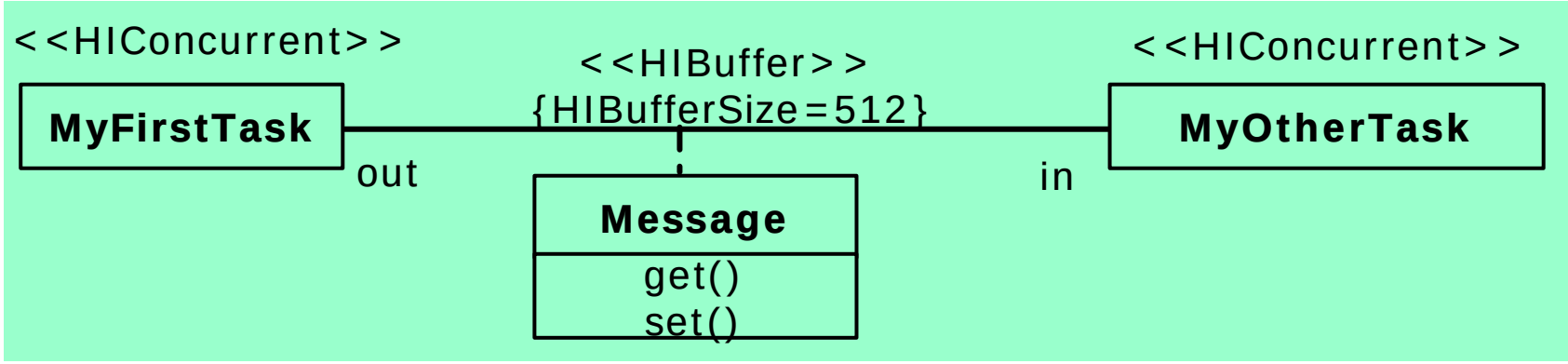


Example: inter-task communication

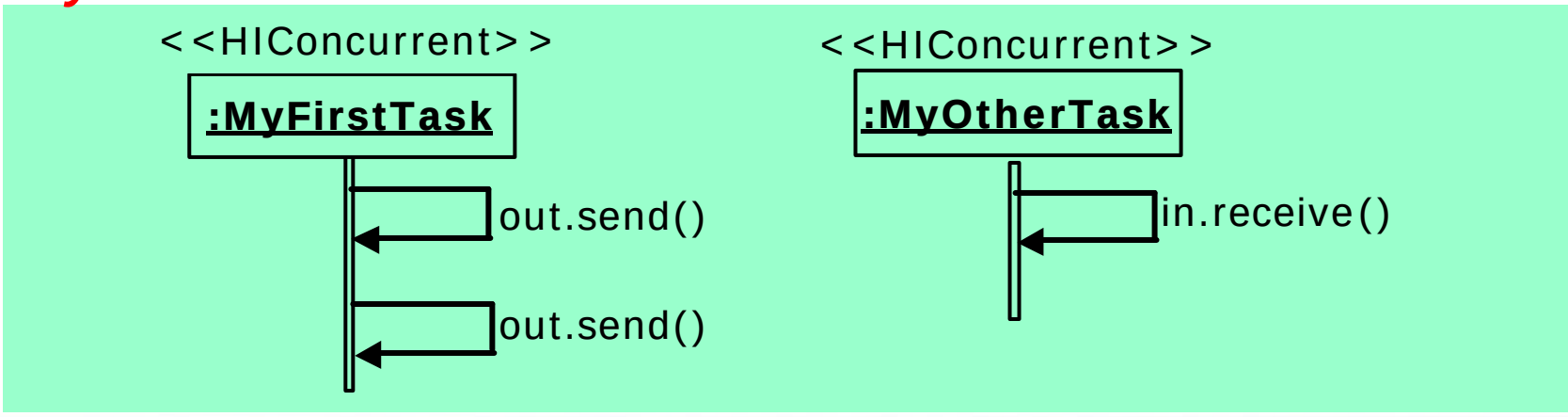
- Example: communication by buffer



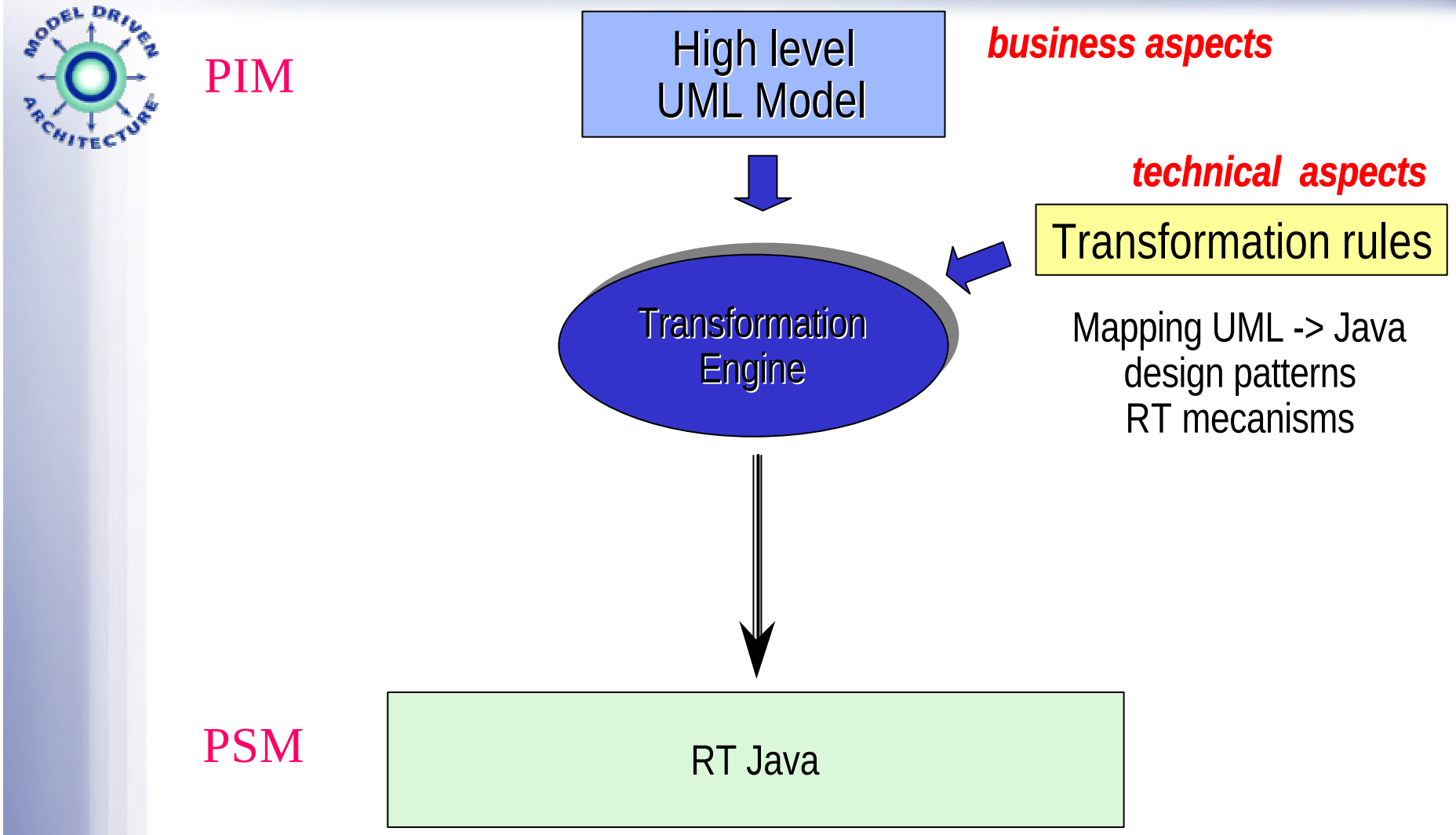
Static view



Dynamic view



A process based on a MDA approach



Why a MDA approach?

- This is the approach which is advised by OMG (ww.omg.org/mda) to improve software quality and to reduce development costs
- This is the approach we used to work with since a long time now, and with success !
- This is a natural approach which follows the trend of languages (independence towards the platform/OS and abstraction to reduce complexity)

MDA, Profile and Automatic Code Generation

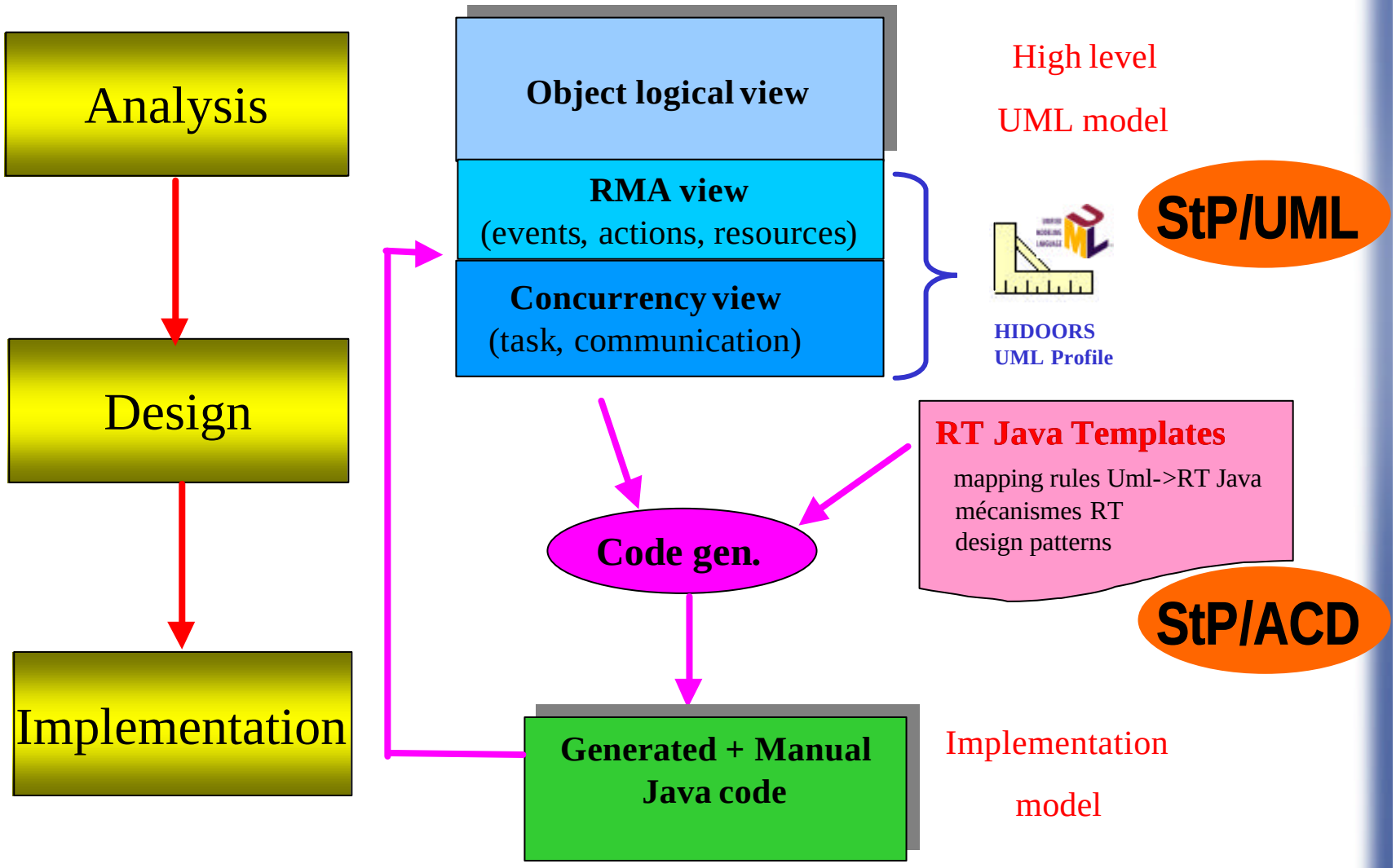
2 essential aspects in the MDA approach:

- Abstraction of UML models
  => Profile

- Model transformation
  => Automatic code generation

The more the model is abstract, the more the code generator plays a central role in the development process

The development process



Code Generation

ACD: template based generator

```
template genClass(MClass)
[MClass.access] class [MClass.name] {
  [loop(MClass->MAttribute As Att)]
  [Att.access] [Att.type] [Att.name];
  [end loop]
}
end template
```

```
public class Car {
  ...
  private double weight;
  private short color;
  ...
}
```

Code Generation 2

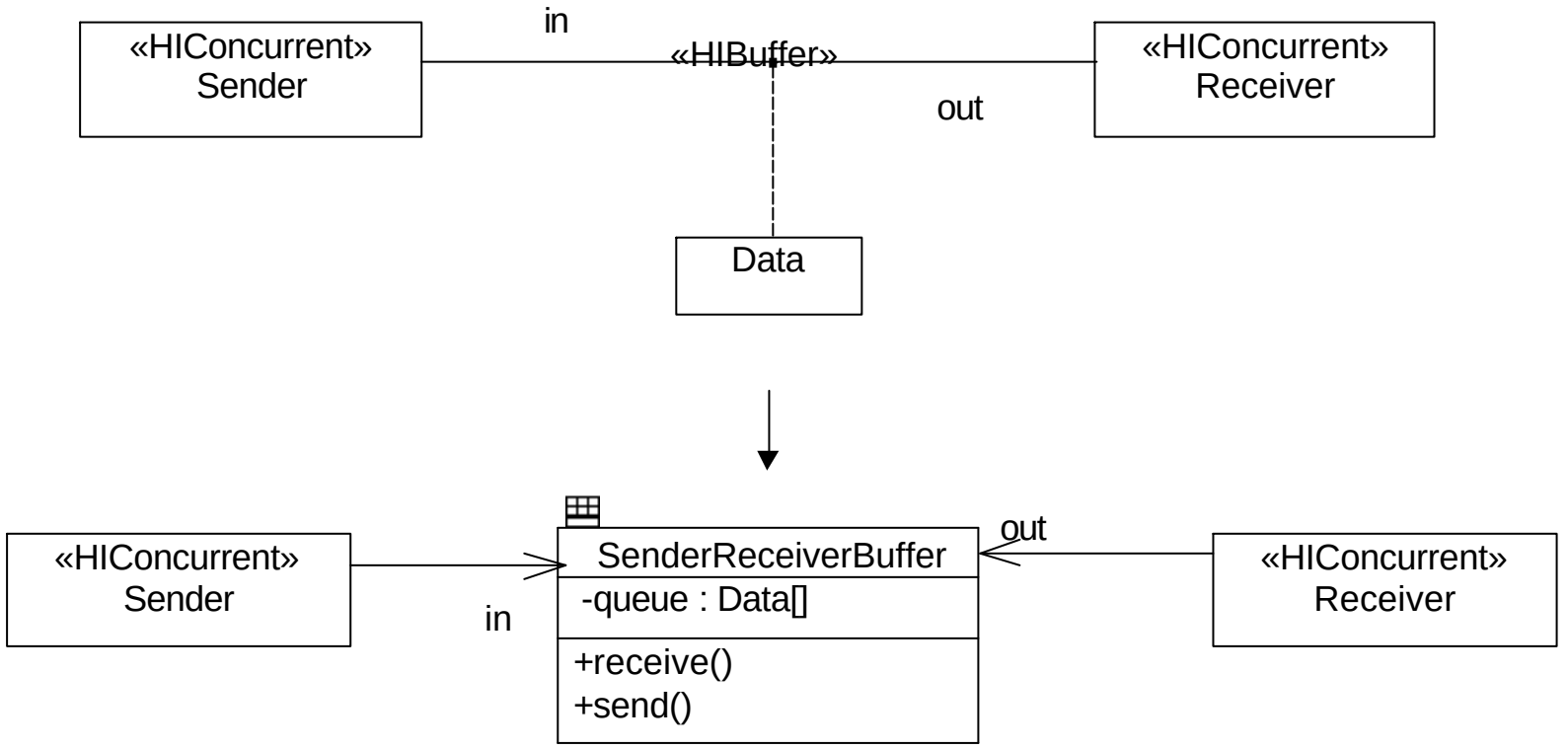
Evaluates HIDOORS RT profile

```
proc initHIDOORSAssocs(MAssociation)
  loop(MAssociation->MAssociationEnd As FromRole->MClass As Class1
    Where [Class1.stereotype] == "HIConcurrent")
    loop(MAssociation->MAssociationEnd As ToRole->MClass As Class2
      Where [ToRole.id] != [FromRole.id] && [Class2.stereotype] ==
      "HIConcurrent")
      ..
    end loop
  end loop
end proc

proc enrichHIDOORSAssocs(MAssociation)
  switch (toLower([MAssociation.stereotype]))
    case "hibuffer" :
      ..
    case "hiblackboard" :
      ..
end proc
```

Code Generation 3

Maps highlevel modeling onto simpler associations (model transformation => MDA)



Code Generation 4

```
public class Receiver {

  // ------------------------------------------------------------
  // instance attributes
  // ------------------------------------------------------------
  private SenderReceiverBuffer out;

  //#ACD# M(UDAT::UID_65c15e75-0000067a-3ee5acfa-000aca45-0000000b)
  //user defined code to be added here ...

  //#end ACD#
  ...
}

public class Sender {

  // ------------------------------------------------------------
  // instance attributes
  // ------------------------------------------------------------
  private SenderReceiverBuffer in;

  //#ACD# M(UDAT::UID_65c15e75-0000067a-3ee5acf3-000626c6-00000004)
  //user defined code to be added here ...

  //#end ACD#
  ...
}
```

Code Generation 5

```
public class SenderReceiverBuffer {

  // ------------------------------------------------------------
  // instance attributes
  // ------------------------------------------------------------
  /**
   * The buffer array holding the messages.
   */
  private Data[] queue = null;

  // ------------------------------------------------------------
  // methods
  // ------------------------------------------------------------
  /**
   * Obtains the next message from the message FIFO queue.
   */
  public void receive() {
    ...
  }

  /**
   * Puts a message at the last position in the message FIFO queue.
   */
  public void send() {
    ...
  }
```

Conclusion

Standard UML has no special realtime features

Profiles give new semantic to an UML-Modell

RT profile defines necessary timing constraints for model elements (duration, priority, preemptive or not)

Tools evaluate these information or complete them

Java code generation uses higher communication patterns, transforms HIDOORS associations in ordinary associations, which need not to be modeled manually