A unified model

Henrik I Christensen & Lars Petersson
Centre for Autonomous Systems
Royal Institute of Technology
Stockholm, Sweden
hic@nada.kth.se

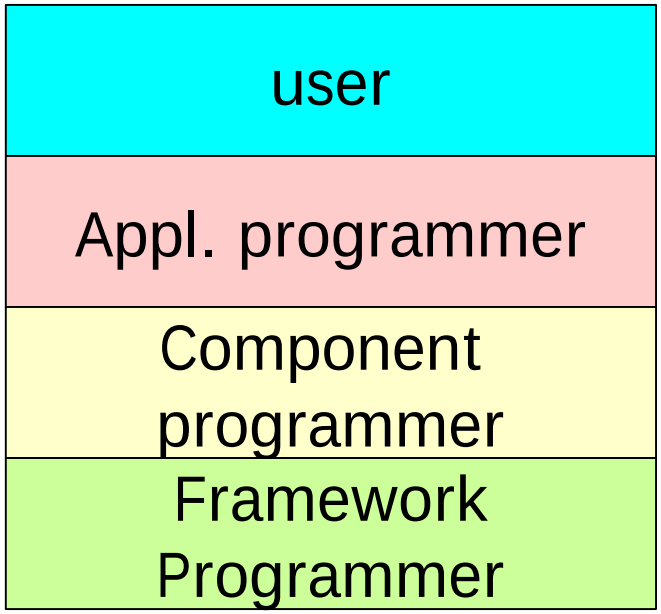
Outline

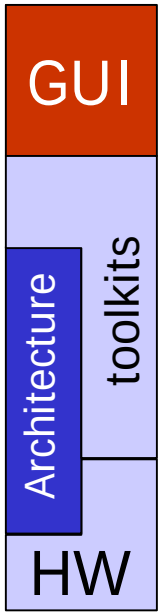
- Dimensions of Systems & Programmers
- What are the common parts?
- What must be different
- A model, and some examples from real systems
- Did we learn anything?
- Where could we go from here?

Who is the end-user?

- There are at least four of them!
- The system user ($)
- The application programmer
- The module programmer
- The components / framework provider

The layers





Ideal Requirements

- Plug-and-play composition of systems
- Self-documenting
- Hard-real-time performance
- Fully portable
  - RTLinux, ECOS, Linux, QNX,
- Language independent
  - As a minimum C, C++, Java

What is out there?

- Saphira
- TeamBots
- Nrobot
  - (¤#!%&!¤%)
- Mobility
- DAMN
- Open Cntl Platf
- NASREM/RCS

- MissionLab
- ISR
- Rex
- GenoM
- OROCOS
- ....

Analysis

- Does any of these systems satisfy our requirements?
- NO!
- Lack of real-time, portability, ...

What is needed

- A components/module model
- A communication framework
- A composition model
- A portable (low level) architecture
- A rich set of toolkits

- A large number of programmers!

Spec of “behaviour”

- Control models – State space models/ODE
- Discrete Models – DES, Petri nets
- Hybrid Models – HDS
- Process Algebras
- Object Oriented Models – UML, Beans, ...

- Which one is the best model?

Integration issues

- Distribution models
- Communication / interactions
- Synchronisation
- Real-time behaviour
- Delays
- Efficiency
  - Run-time, programming, debugging, learning curve
- Fault tolerance

A unified model

- Three components
  - A module specification
  - A communication model
  - A coordination model

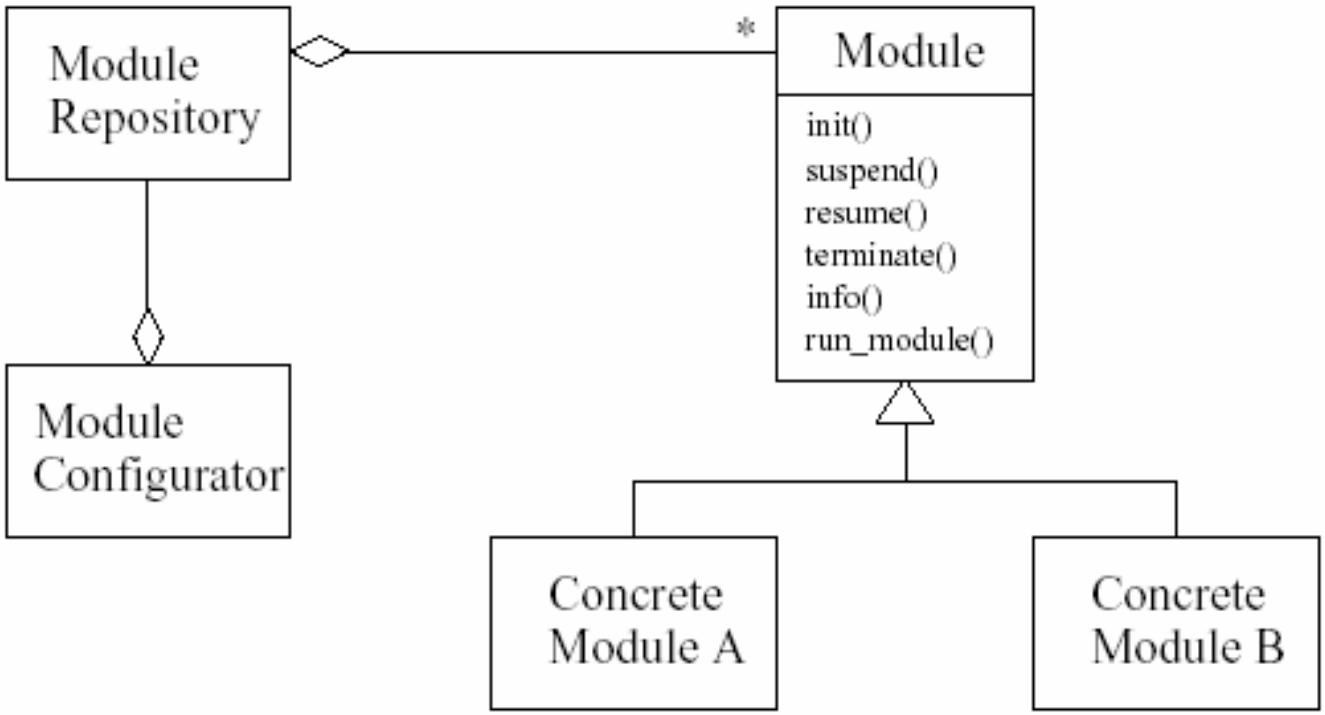
The coordination model

- Coordination model is user dependent
  - Framework programmer
    - Modelling language
  - Module programmer
    - Control laws/DES ! HDS
  - Application Programmer
    - Macro language / Process algebra
  - End user
    - Natural language “derivative”
      - Gestures & speech

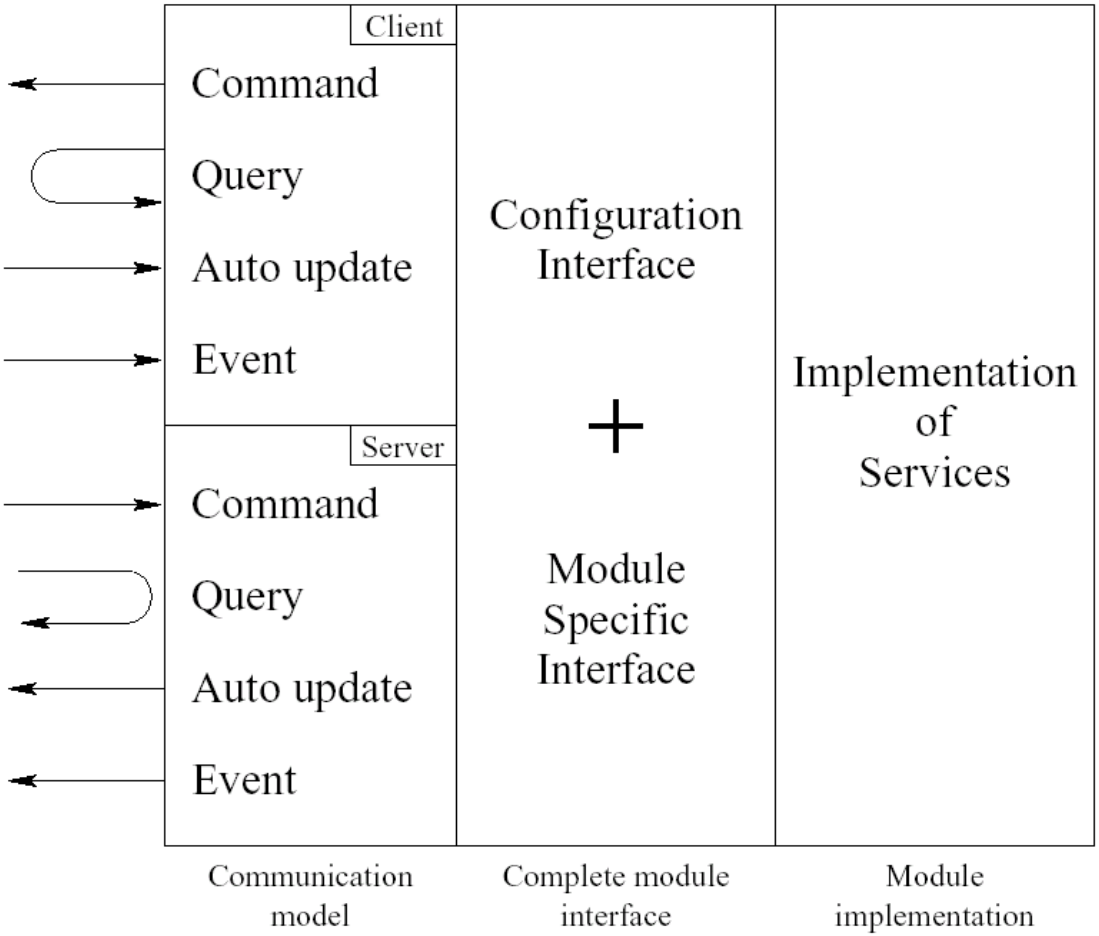
Communication Model

- Four basic interactions [Schlegel et al.]
  - Command
    - Client -> server [no ack]
  - Query
    - Blocking send/recv
  - AutoUpdate
    - Updating at discrete or timed intervals
  - Event
    - Server -> client [no ack]

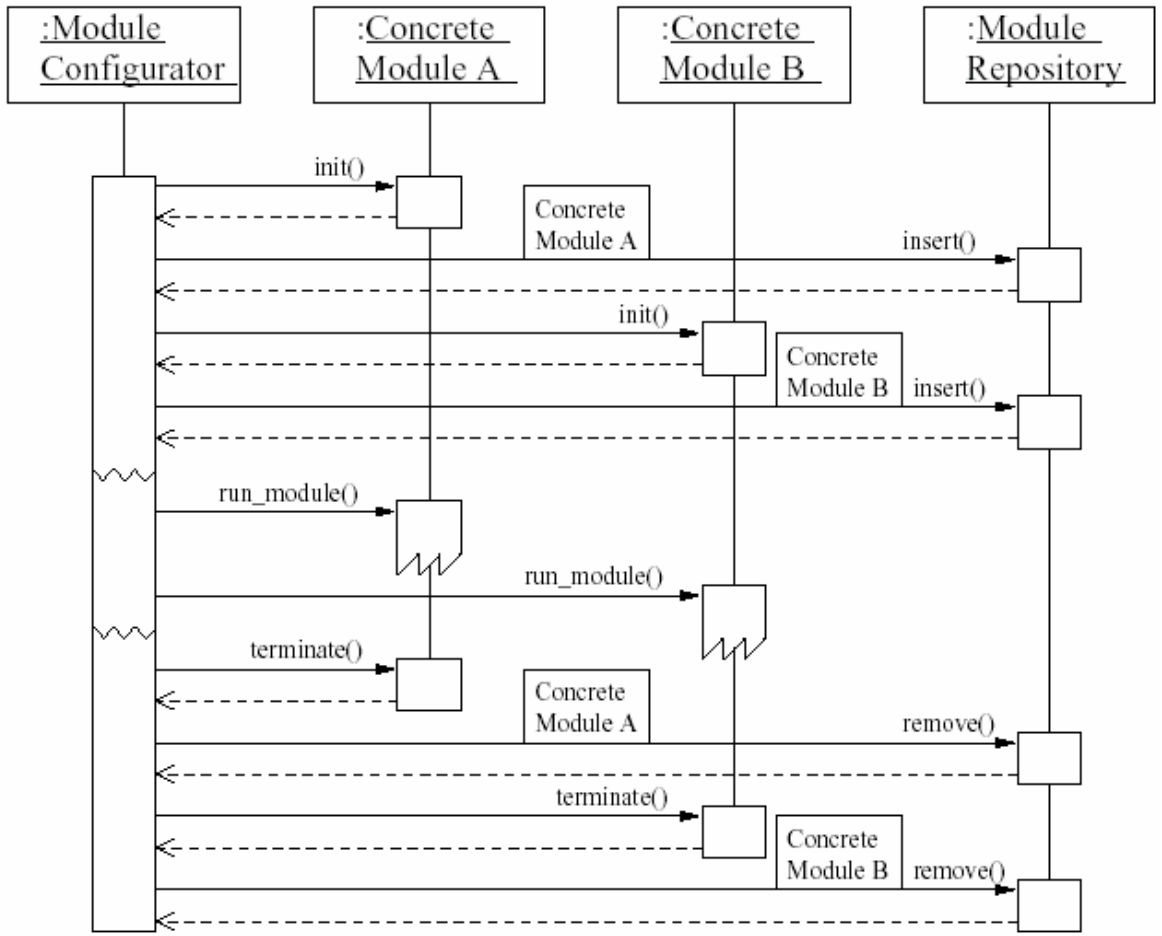
Module Layout



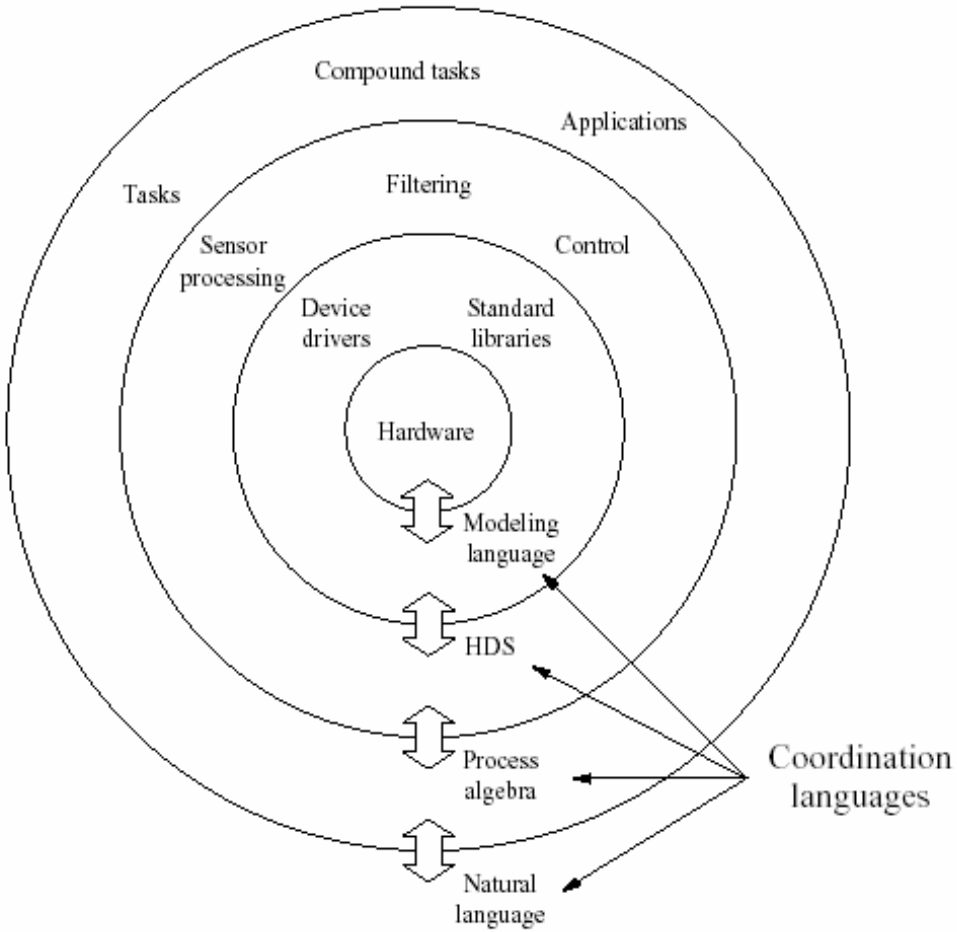
Module Interface



Configuration of a module



System layout



A example system

- A first step towards an implementation
  - DCS [Petersson, Austin, Christensen 01]
  - A standard module modul
  - Implemented in C++
  - A distributed communication model
  - A process algebra for coordination
  - Implemented for mobile manipulation

Glue/Architecture

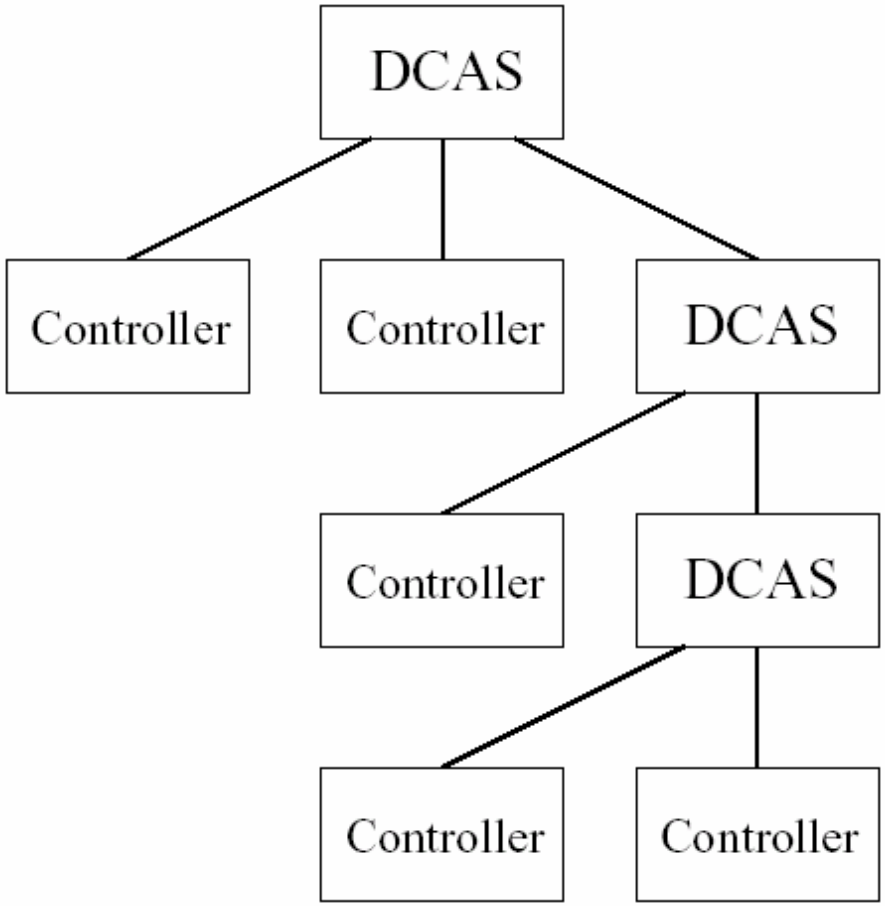
- A “supervisor” for launching of systems
- A local process coordinator
  - For of group of processes
- A NameServer
  - For location independent access to info and run-time configuration
- TimerServer – high res time coordination
- GenComm – A communication library

Example task


```
process CompliantMotion(host1, host2){
   host{host1}
   instantiated_processes{
      forcedata instof ForceSensor(host1);
      compliant instof CompliantCtrl(host1);
      puma instof Puma560(host2);
      hit instof HitDetector(host1);
   }
   external_io{
      // This process has no external I/O
   }
   internal_connections(forcedata, lbl1){
      forcedata, Out1 -> compliant, In1;
      forcedata, Out1 -> hit, In1;
   }
   internal_connections(compliant, lbl1){
      compliant, Out1 -> puma, In1;
   }
   internal_event_actions{
      // Perform compliant motion until the process
      // 'hit' detects abnormal forces and preempts
      // the running group of processes

      hit # (forcedata[lbl1], compliant[lbl1], puma)
   }
}
```

Process model

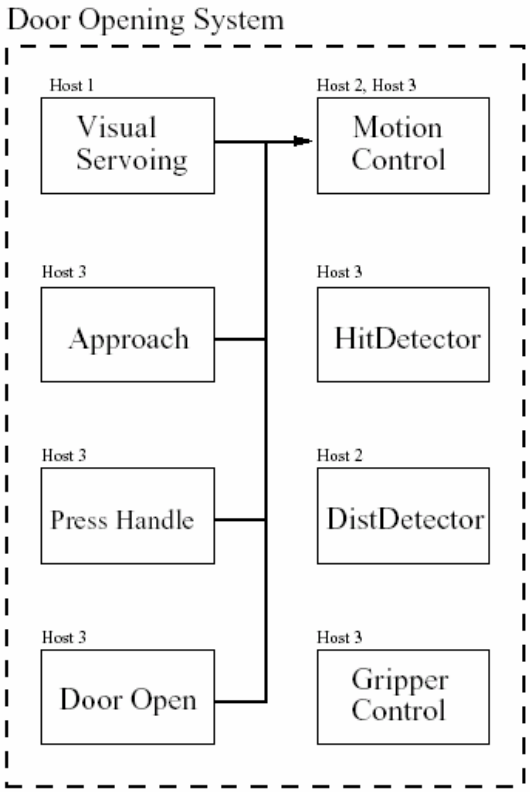


Example task

- Specification

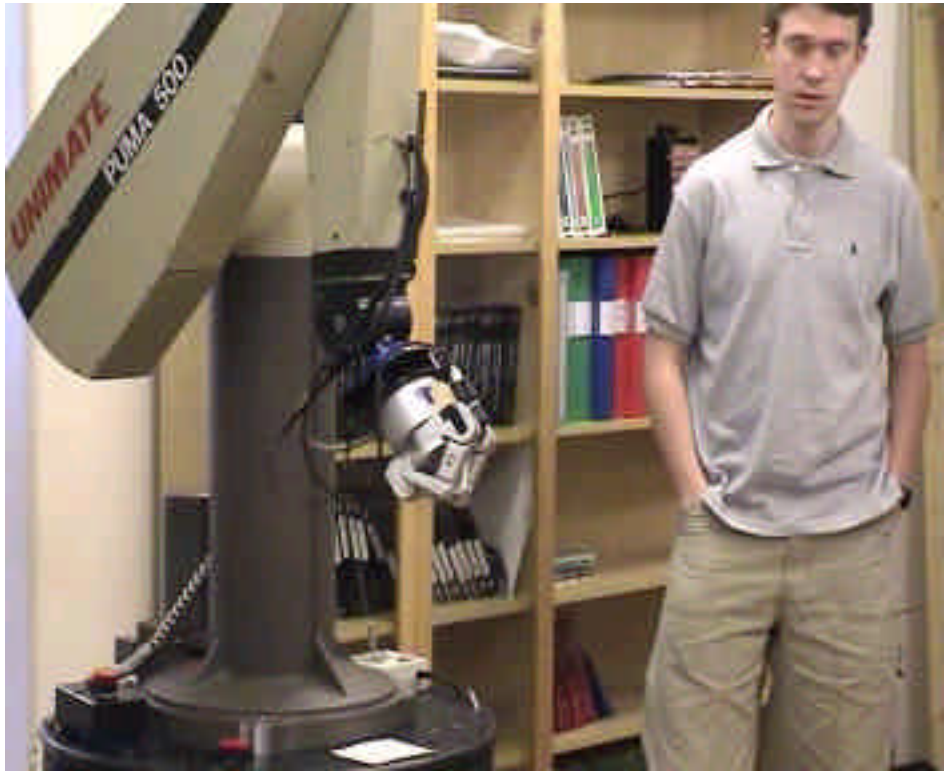
```
MotionControl , ((VisualServoing # DistDetector) ;
                 ((OpenGripper, Approach) # HitDetector) ;
                 CloseGripper ; PressHandle ; DoorOpen)
```

- Model



A more complex task

- Grasping of objects on a table (Jun 2001)



Considerations

- Specification
  - XML based specification of system
- A well defined name / object hierarchy
- A more articulated communication model
  - A la IDL for interfaces, Corba impl
- Use of standard for languages (IEC 1131)
- A rich set of toolkits?

Summary

- There is a need for an open source framework
  - Standard model for modules/components
  - A standard communication model
  - A rich set of coordination mechanisms
  - A well articulated set of toolkits
  - Support of a standard set of platforms

- OROCOS is a step in this direction! Stay tuned!