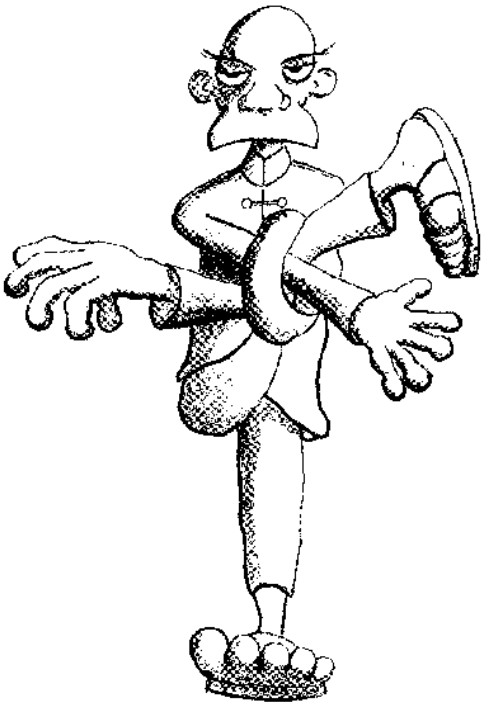
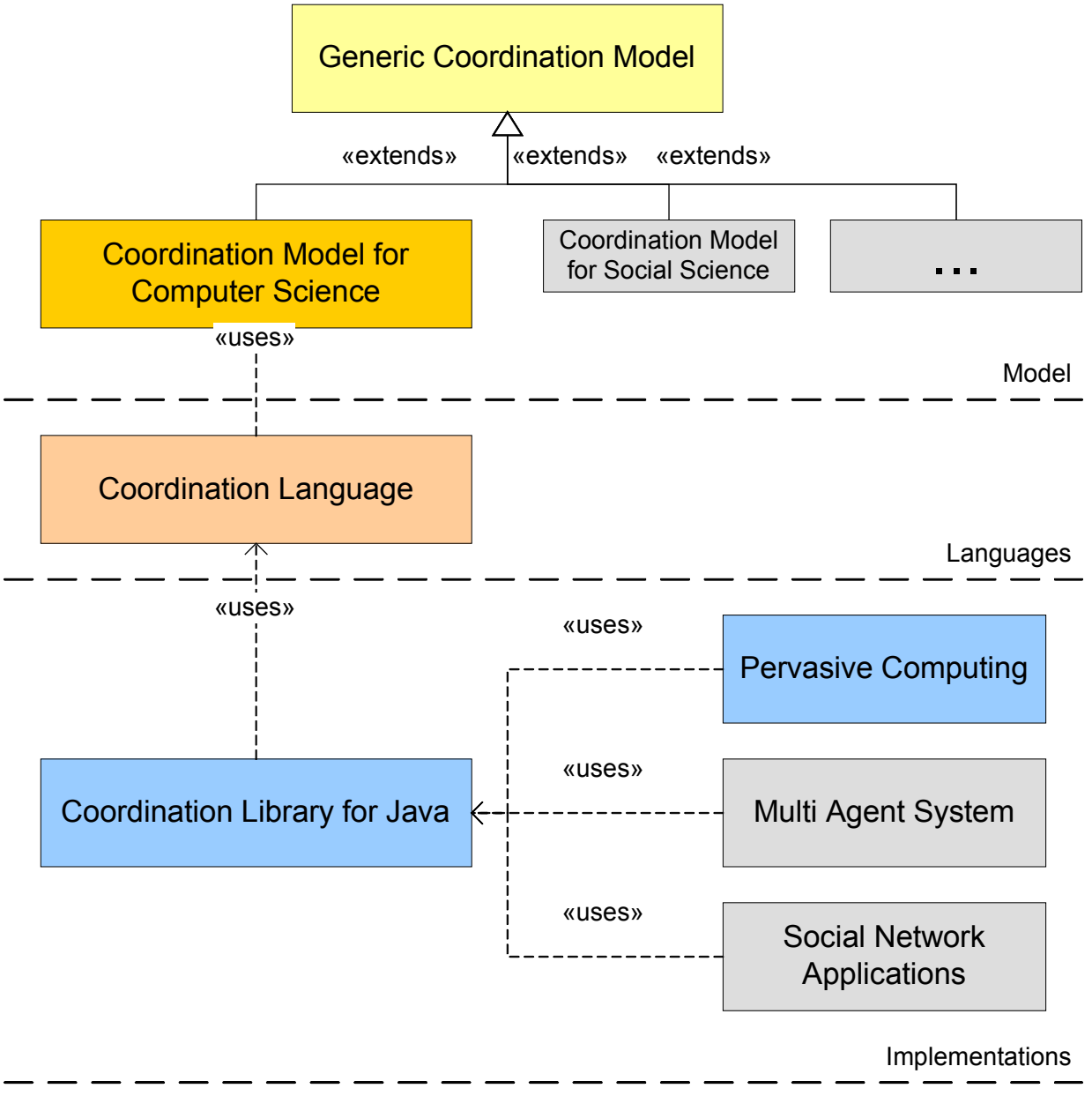
Coordination Model for Computer Science

(inclusive pervasive computing)

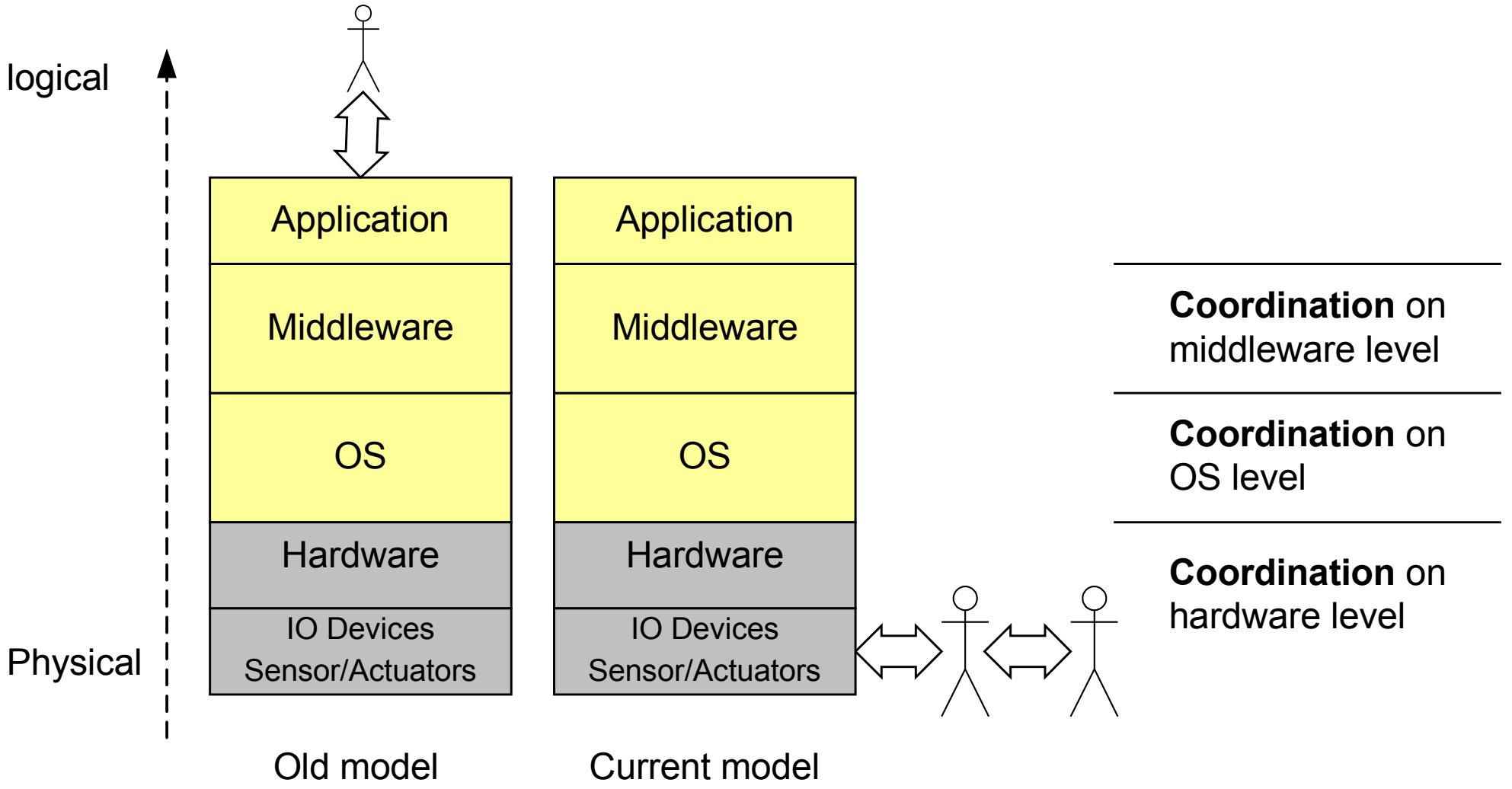
Benjamin Hadorn
PAI research group, University of Fribourg
26.02.2010



Overview


Generic Coordination Model
«extends»
«extends»
«extends»
Coordination Model for Computer Science
Coordination Model for Social Science
...
«uses»
Model
Coordination Language
Languages
«uses»
«uses»
Pervasive Computing
«uses»
Coordination Library for Java
Multi Agent System
«uses»
Social Network Applications
Implementations

The Computer System



Coordination Levels

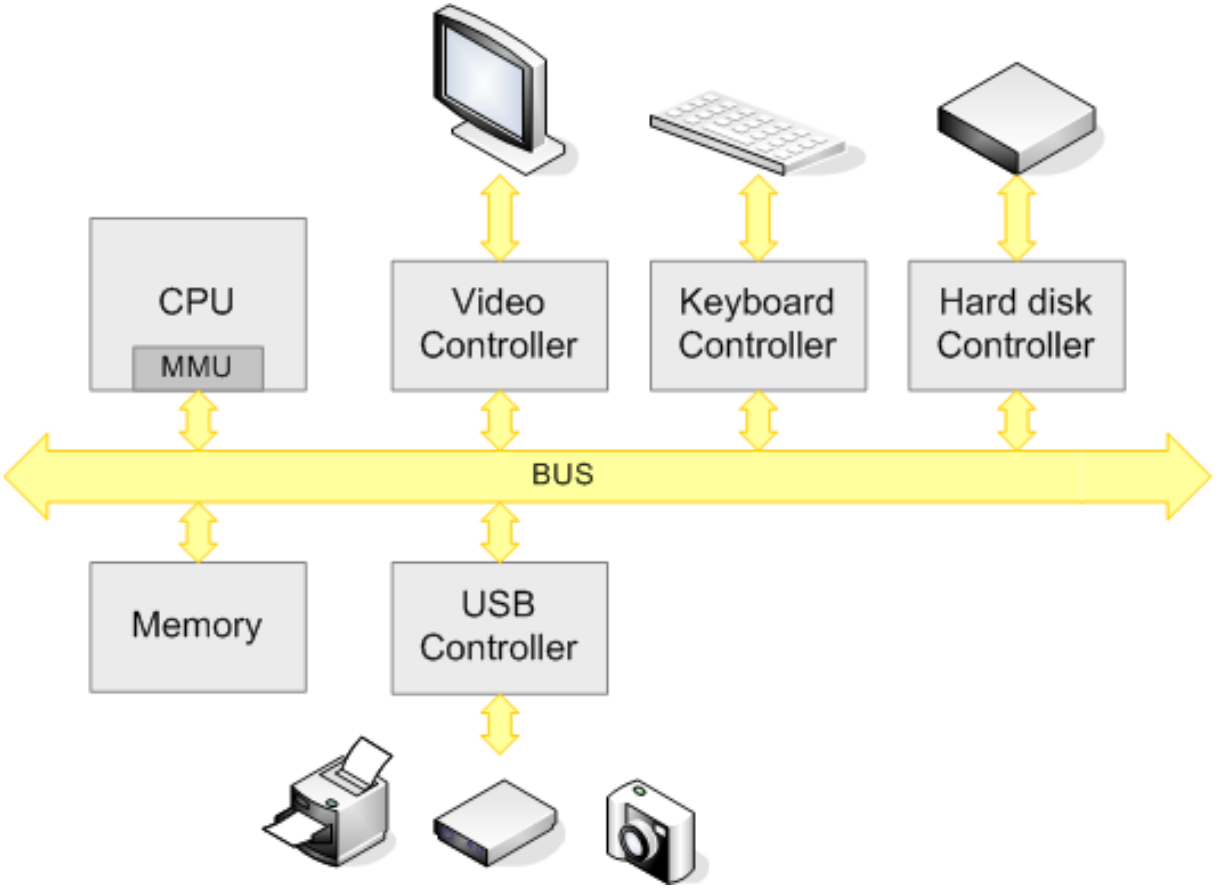
- Coordination on the Hardware Level
  - wired coordination. e.g. medium access on ethernet
- Coordination on the Operation System level
  - Manages the dependencies between processes and hardware devices or system objects.
- Coordination on the Middleware level:
  - Manages the interdependence of processes by mediating communication spaces and channels
  - Manages the dependencies between human activities using context and activity aware computing.

Physical Layer (I)

- Properties (from generic Model)
  - entities exist only in one place at the time
  - Overlapping not possible
- Entities
  - Hardware
  - Network
  - The user (human)
  - Elements of programming languages
  - GUI design

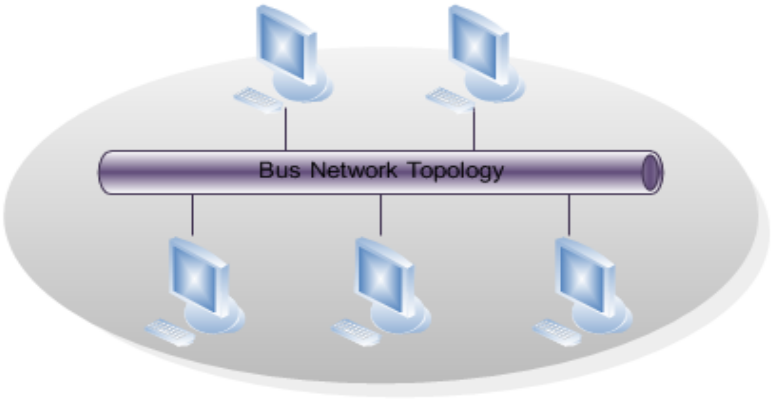
Physical Entities (I)

- Hardware devices
  - Internal components like processor, memory
  - IO devices like screen, keyboard

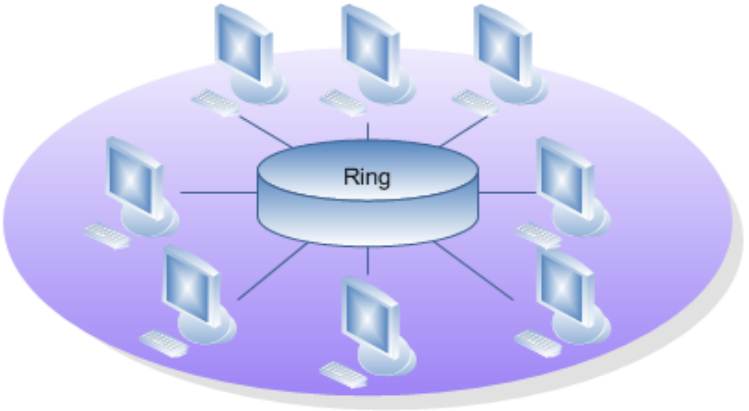


Physical Entities (II)

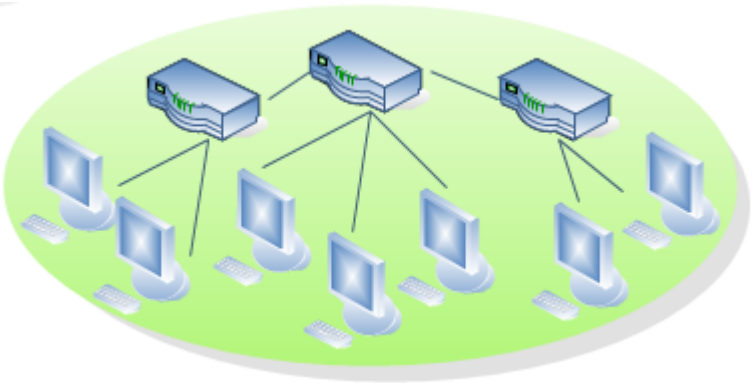
- Network Topologies



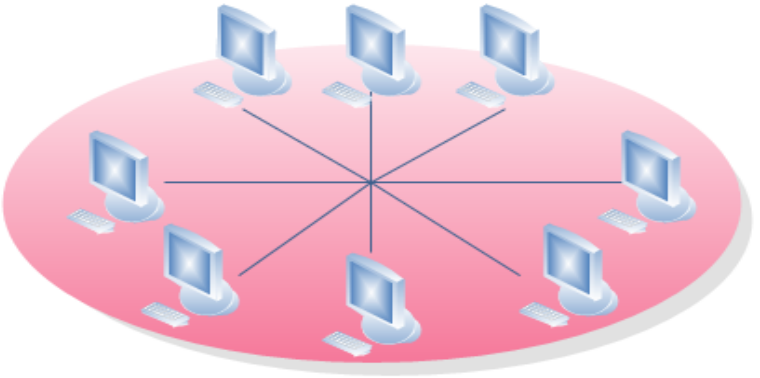
Bus Network



Ring Network



Tree Network

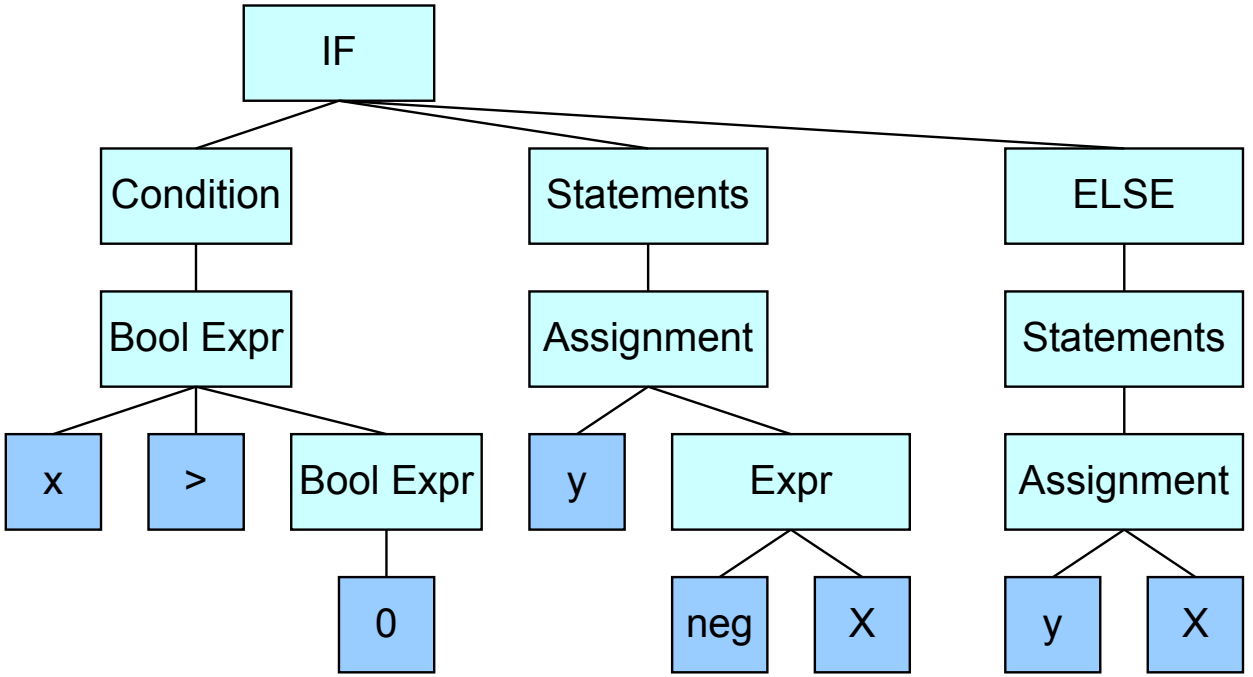


Star Network

Physical Entities (III)

- All elements of a programming language can be seen as physical entities (Java. Lisb, Prolog...)
  - representation by abstract syntax tree (AST)
- Entities like
  - Variables, classes, functions, operators
  - example:

```
if (x > 0)
{
  y:= -x;
}
else
{
  y:= x;
}
```



Virtual Layer

- Entities
  - Semantic of a program code
    - Software components
    - Services, Applications
  - virtual memory
    - can exist in RAM, cache, on hard drive
    - mapping to real memory addresses by MMU
  - virtual machines
    - one virtual machine can run on several hardware platforms

Task Structure

- Manages the task dependency. 2 Examples:
  - Scheduling algorithm of the OS
    - assigns processes (activity) to the processor (entity)
    - dependencies to resources are kept in the process table entry
      - Which entitiy plays which role (port, tool etc.)
  - Planning Software (like MS Project)
    - activities of humans (entities)
    - task organisation (project management)
      - who is doing what?
    - resource assignment
      - who needs what?

Physical and Social Laws

- Physical Law
  - The speed of transmitting information
  - Constraints given by the programming language
    - instanciated objects can not change these constraints
- Social Law
  - Security settings
    - Permissions
    - Restrictions
  - Ethics

Physical and Social Laws for Pervasive System (I)

- Physical
  - Restrictions by sensors
  - Measuring of gravitation, temperature, pressure, ...
- Social
  - Used to allow or restrict interaction between humans and the pervasive system
  - Depend on
    - the location of the user (environment)
    - the context of the user
    - situation analysis
  - The law controls the objective coordination (generic model)

Observer (I)

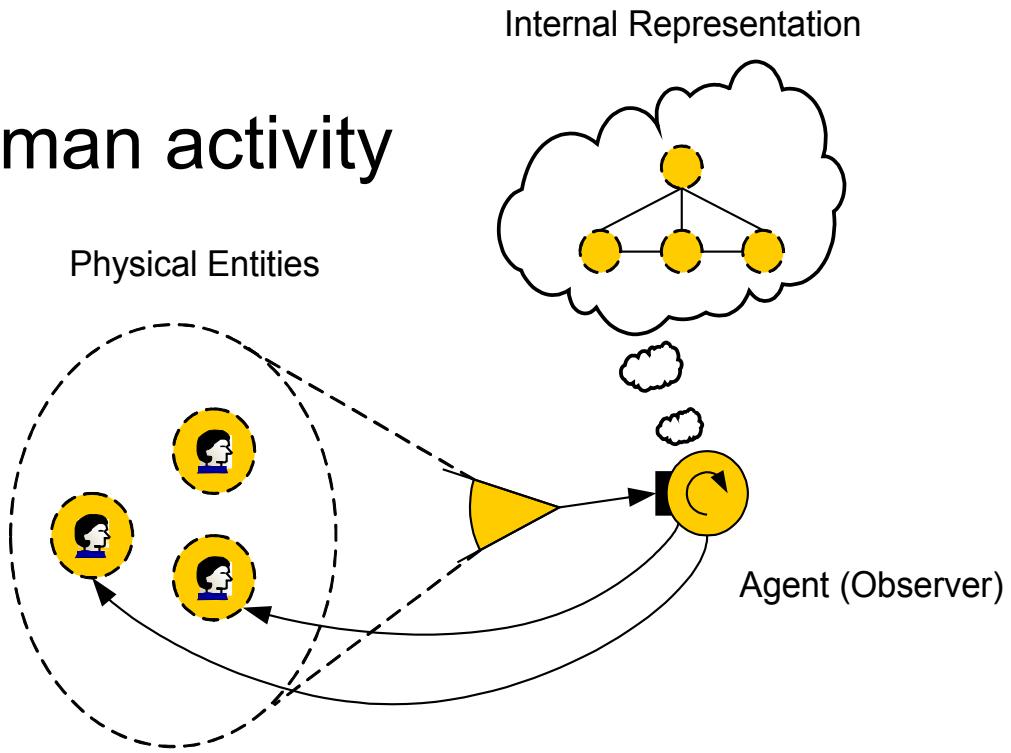
- The observation of a computing system is done by
  - humans (users, system engineers)
    - common view are given by the senses of the humans
    - monitoring tools help to capture data (extending the view)
  - sensors
    - Energy monitoring (e.g temperature)
    - Observing environment and human activities (pervasive computing)
    - Observing the states of a machine (industrial computing)
  - observer-subject pattern
    - a software component acts as an observer, listening to changes of an other software component (subject)

Observer (II)

- Traditional entity classification (done by humans)
  - Actor is
    - the user (user centered view)
    - computer (computer centered view)
  - Tools:
    - IO devices
    - Processor helping doing some work (could be seen as an actor cooperating with the user)
  - Artifacts: Files and data generated by the processor
  - Port: Interface to network (internet)

Pervasive Observer (I)

- Paradigm change in pervasive computing
  - Duality within a computing system
    - real world
    - internal representation (software)
    - Observation is relative
  - System observes the human activity
    - Actor: human, car, train
    - Tools: PDA, Phone,...
    - Ports: Phone, WiFi,...



Pervasive Observer (II)

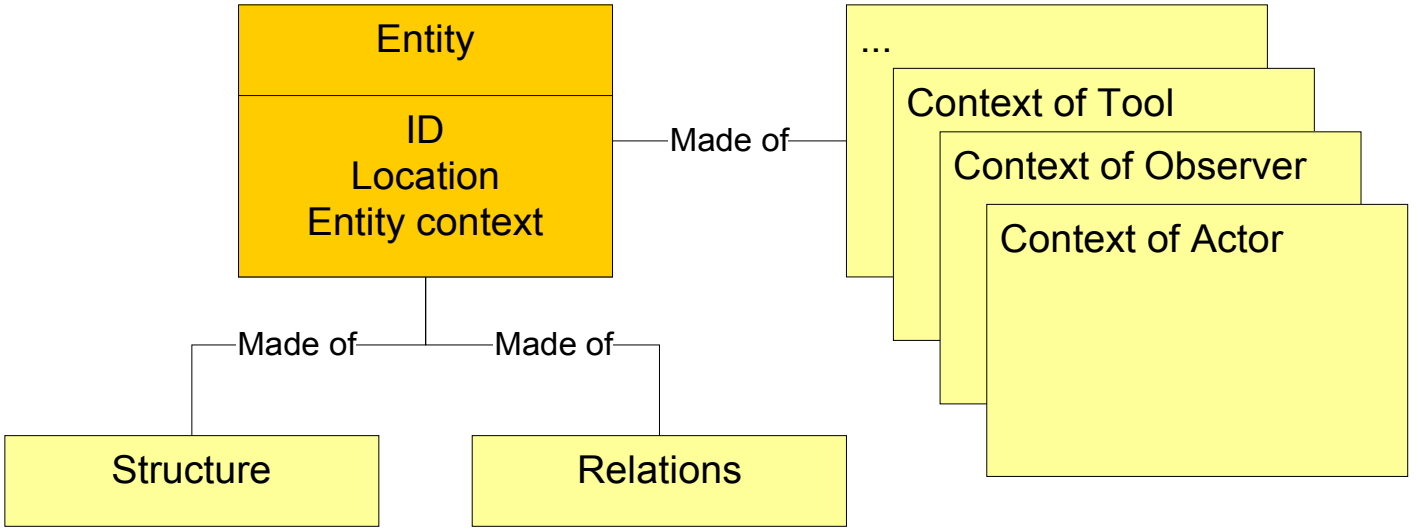
- The real world is observed by the computing system
  - Sensors are observers having a view onto a part of the world
  - Humans and their activities are tracked by sensors
  - Context information is gathered by sensors
- The observed entities are represented by an software entity (stub)
  - helps to communicate with the entity
  - storage of context data

Pervasive System (I)

- Internal representation of the real world
  - entity space
  - context
  - relations
- Internal observers
  - help to react on context changes
  - evaluate situations
  - report alerts to a higher level (application)

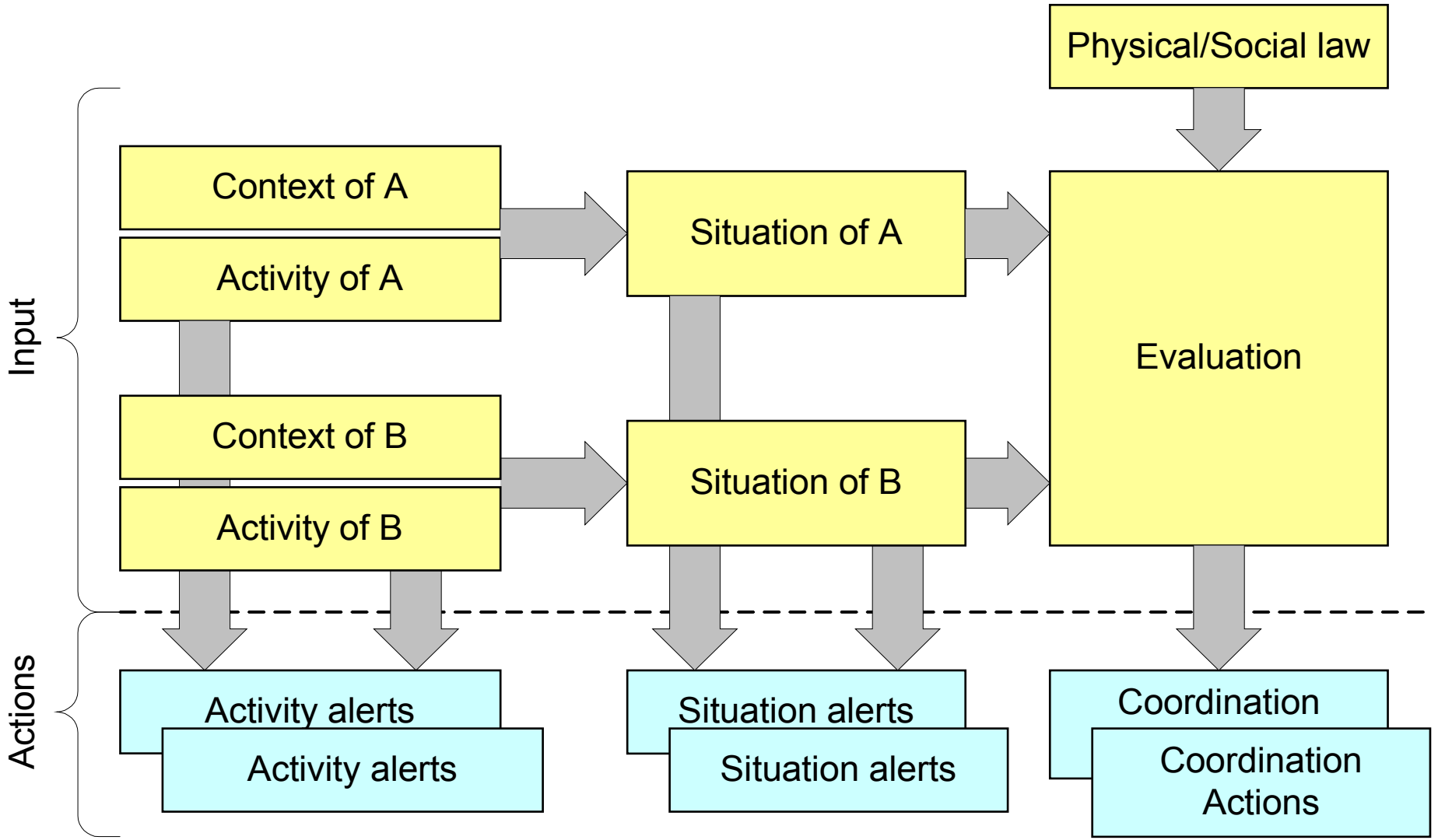
Relative information storage

- Entity contains for each possible role a context
  - The observer will ask the entity what it is (actor, observer etc.) and receive the context
- Relations contain all dependency information
  - Social, spacial and task dependencies



Pervasive Coordination

- Events of coordination



Communication

- Definition (from generic model)

  Communication is a process of transferring information form one entity to one or more entities.
- Human Computer Interaction (HCI)
- Network communication (like Bus, Ethernet, WiFi)
- Programming languages
  - variable assignment, function calling, events
- Interprocess communication
  - shared memory
  - message passing
  - others: tuple spaces, shared files (e.g pipe)

Ethernet Communication Pattern

- Ethernet
  - Peer to peer: TCP/IP
  - Broadcast and multicast
    - Ethernet using UDP
    - muticast address 224.x.x.x
  - Generic communication
    - Linda tuple space, JavaSpace

Programming Language (I)

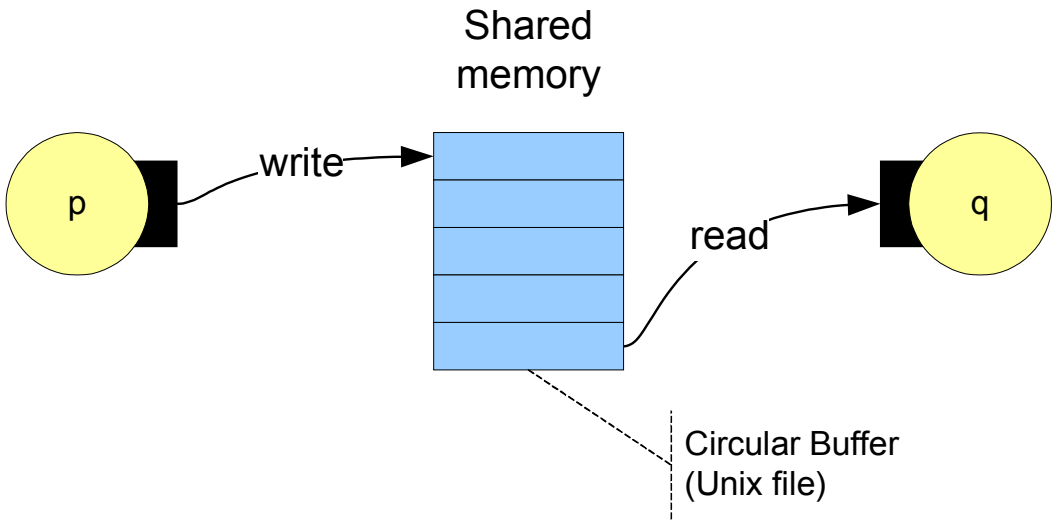
- x = y;
  - syntactical: y communicates with x
  - semantical: y send its value to x. The value of x equals the value of y afterwards.
- x *= y;
  - syntactical: same as x = y;
  - semantical: x is multiplied with the value of y
- x = x * y;
  - syntactical: x and y produce a result r which is communicates with x
  - semantical: same as x *= y;

Programming Language (II)

- Calling functions
  - A function can be seen as an entity.
    - If a function f() is called within an other function g():
      g() { f(); } then g() communicates with f()
  - f(x↓)
    - syntactical: call of f() by passing x.
    - semantical: f(x) receives the content of x.
  - f(x↓,y↑)
    - syntactical: call of f() by passing x and receiving y.
    - semantical: f(x,y) receives the content of x and returns a content for y.
  - f(x↓↑)
    - syntactical: call of f() by passing x and receiving x
    - semantical: x is passed to function and contains the changes afterwards (defails depend on the implementation).

Interprocess Communication

- Communication between processes
  - Shared memory



  - Message passing
  - Shared file
    - Unix provides the pipe |
    - The pipe implements the producer-consumer coordination process
      - p | q : process p produces data, q receives the data
      - Unix mediates a channel (a file) using the FIFO protocol

Events and JavaSpace

- Events.

```
myButton.addActionListener(
   (ActionListener)EventHandler.create(ActionListener.class,
                                       frame, "toFront"));
```

  - An event or signal is sent and captured by an action listener (Java).
  - The signal handling works similar (Unix, C)
- JavaSpace
  - entities communicate indirectly over a blackboard space
  - distributed space storing message objects
  - receiving messages using a matching template