

# Software development methodologies for Cyber-Physical Systems

**Professor Yolande Berbers**

**DistriNet, Department of Computer Science,  
K.U.Leuven, Belgium**



## need for research in Cyber-Physical Systems

- lot of effort devoted to underlying technologies
  - sensor technology
  - wireless network technology, batteries
  - RFID's
  - tiny processors
  - ...
- some effort devoted in generic software layers
  - tiny OS's for sensor networks
  - basic middleware for sensor networks
  - networking protocols for wireless networks
  - ...
- **less effort for SW development methodologies**
  - **current applications are ad hoc (logistics, some AAL appl, ...)**
  - **SW for Cyber-Physical Systems IS NOT BUSINESS AS USUAL**
    - **adapted methodologies are needed**

## Specific characteristics of Cyber-Physical systems: NOT BUSINESS AS USUAL

- world is made up of **many Physical Things**
  - possibly with sensors and actuators
- Things: not general purpose, have a small and specific role
  - **identity** and **semantics** play a crucial role
- **location** and **geometry** (related to the Physical): essential
- **granularity** is generally small, but varying
- **scale** is huge
- environment full of things is extremely **dynamic**
  - Things come and go
  - context is changing constantly
  - applications will form dynamically
- **uncertainty** becomes the rule
- **security** is a huge problem (not covered in this talk)

## General approach we will follow

- **Dual World model**
  - integrates the notion of semantics, location and geometry
- **Windowing mechanism**
  - allows filtering, zooming and/or focusing on context
  - addresses granularity and scalability problems
- **Combination of**
  - **Service oriented paradigm**
    - uses bottom up approach
    - has successfully been applied in business transactions
  - **Goal oriented paradigm**
    - follows a top down approach
    - is mostly used in highly interactive environments

## Dual World Model

- Spans the digital and the physical world
- need to model capabilities and properties of Physical Things
  - Things sense and act on physical environment
  - Things provide services with respect to user context and env. state
  - Things collaborate and communicate with one another
- need to reason on these models
  - determine relations
  - challenge is to combine semantic and spatio temporal reasoning
    - e.g a thing belonging to category T is in front of the user
- need to store the models on the Things
  - but limited storage capacity
  - but varying network connectivity (from no access to full access)

## Dual world model: modeling language

- necessary properties
  - should describe both semantics and spatio temporal properties
    - ‘near’ can have different meaning in function of context
  - must be minimal: sufficiently complete, but not too complex to enable efficient reasoning
  - must enable partial descriptions: storage of Things is limited
- input
  - ontologies have been used for description of semantics
    - but not for spatio temporal properties
  - gaming theory has interesting elements
    - logical relations between volumes (intersections, inclusions)
  - spatial databases also use space as property
  - less interesting input for temporal properties

## Dual world model: reasoning

- requirements
  - reasoning in terms of positioning, nearness, shapes and geometry
  - reasoning in terms of time
  - support inference with respect to vagueness and fuzziness
- input
  - Allen's temporal interval algebra
  - region connection calculus

## Dual world model: storage of models

- requirements
  - Things must carry their own model (not always connected)
  - Things must carry their interface to allow dynamic interaction
  - several storage strategies are necessary
  - difference between active and passive Things, granularity
- strategies
  - offer different levels of description less and more complete
  - combine local storage, neighboring storage, internet storage
    - Things with greater storage capacity can help small Things
  - use space efficient storage mechanisms
  - store in function of dynamic usage patterns

## Window on the World

- huge overload of information
  - both for humans and for Things (Things have limited capabilities!)
  - scale is magnitude higher than what we already are used to
  - need for methods to limit the view of the available information
- window = limited view
  - on context and environment
  - can be in space (only see what is 'near' to me, what is reachable)
  - can be in semantics (only see Things with a particular property)
  - can be in granularity (potential use of magnifying glass)
- requirements
  - should allow for filtering
  - should allow for zooming
  - should be accessible by humans and by Things

## Window on the world: defining window scope and variability

- necessary dimensions for qualitative and quantitative granularities and scales of information relevance
  - time (e.g. age of information)
  - space (e.g. nearness) (100 m may not mean same in diff context)
  - granularity (e.g. scale of accuracy)
  - semantic distance (e.g. relevant related concepts, right properties)
- uncertainty in all dimensions needs to be taken into account
- need to manipulate and or aggregate different windows
  - matching of 2 views (requester and provider)

## Window on the world: ambient window of interaction

- window of interaction
  - access point
  - presents an interface to interact with the environment
  - like peephole display, slides over environment, annotates Things
  - allows interaction with virtual counterparts of the physical objects
- takes into account
  - accessibility properties of user
  - user location (possible large display nearby)
  - goals of the user (current activities of user)
- interface should be
  - multimodal
  - interface must be generated at run time
- input from HCI models
  - task models, domain models, user models, ...

## Window on the world: digital user interface for Things

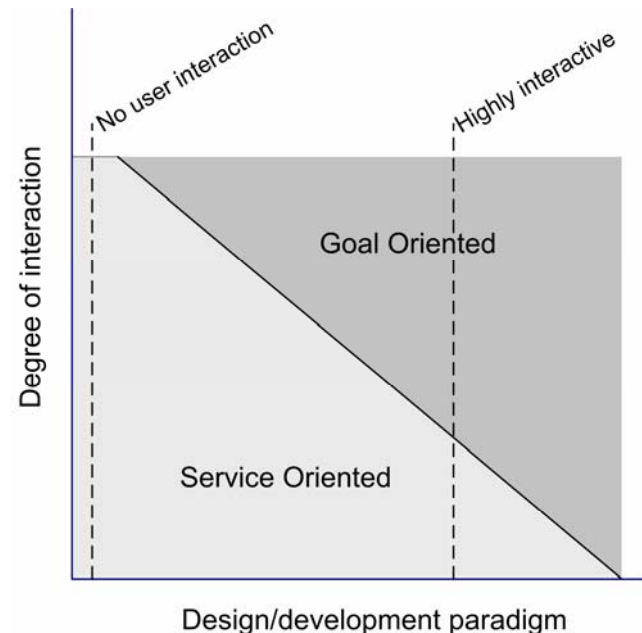
- for every Thing in neighborhood
  - interactive digital user interface can be generated
  - at run time
  - using nearby devices
  - to interact with Thing itself (if that is possible)
- need for user interface generator
  - can use different types of visualization (graphical, speech, ...)
  - is adapted to both the user and the Thing

## Window on the world: Quality of experience

- quality of experience
  - reliability, responsiveness, personalization
- should take the preferences of the user into account
  - user profiling
- should optimize the possibilities
  - depending on processor, memory, display, battery, ... possibilities
- strategies must be integrated in the runtime system

## Software development for Cyber-Physical Systems

- each Thing has something to offer
- each person has a goal he/she wants to pursue
- paradigms are different
  - Things offer services: service oriented paradigm is appropriate
  - people have goals: goal oriented paradigm is appropriate
- a combination of both paradigms will be necessary



## Service oriented methodology for Cyber-Physical Systems

- differences with classical SOA
  - not all services of Things can be registered somewhere (scale)
    - discovery will be different
  - semantic web: addition of spatio-temporal properties
    - see Dual World
  - complexity of applications
  - dynamic aspect: applications will be composed at runtime
  - larger diversity, heterogeneity, volatility
- need for adapted
  - discovery mechanisms
  - selection mechanisms
  - composition language

## Service oriented architecture: discovery mechanisms

- need for Windows on the world to discover those services that are interesting
  - semantically
  - nearby
  - timely
- UDDI needs to be extended
- intelligent (efficient) discovery ways are necessary
  - not first discovering everything, and then filtering
  - make use of processing power available in neighborhood
  - collaborate with other Things

## Service oriented architecture: selection mechanisms

- need for policies
  - based on location of users and Things
  - based on preferences of users
  - based on functional properties of Things
  - based on non-functional properties of Things
- declarative mechanisms to specify and enforce policies

## Service oriented architecture: composition language

- need for richer composition operators, that can deal with
  - scale
  - volatility
  - uncertainty
- extensions to current workflow languages (BPEL and alike)

## Goal oriented methodology for Cyber-Physical Systems

- end user needs to configure and even ‘program’
  - large heterogeneity of devices
  - user wants to use devices in many different settings
    - not only the preprogrammed ones
  - user does not want to be bothered with low level details
  - user knows WHAT he/she wants to achieve
    - not HOW this will be achieved
- need for goal oriented software development methodology
  - modeling of tasks
  - dynamic task composition
  - robust user interfaces

## Goal oriented methodology: modeling of tasks

- language to describe goals
- models to describe tasks that will fulfill these goals
  - e.g. based on ConcurTaskTree, timed Petri-Nets, process algebras
- task ontologies, possibly stored on the Things

## Goal oriented methodology: dynamic task composition

- combine partial task ontologies into greater tasks

## Goal oriented methodology: robust user interfaces

- minimize unexpected behavior for the user
- enable the user to correct mistakes (undo/redo)
- provide feedback to user for unexpected results
- provide feedback to user when user performs actions
- let user feel in control, in spite of inherent complexity
- make tools that let the user simulate what he wants to do in a virtual environment

## Conclusion

- Fulfilling the promises of Cyber-Physical Systems
  - huge challenge
  - efforts needed in technology
  - efforts needed in OS's and middleware and protocols
  - also research needed in software development methodologies
- biggest challenge
  - users that 'program' new applications on the fly
  - new applications that are dynamically composed in function of the activities of the users (e.g. in Ambient Assisted Living)
- **a lot of work needs to be done !!**