Networked components

(the architecture of ambient intelligence)

Johan Lukkien

(thanks to Johan Muskens, TU/e and
Jan Nesvadba, Philips Research)

SAN @ TU/e

- System Architecture & Networking
  - one of eight research groups of Computer Science
    - as off 2002

- Staff: 7 full-time, 4 part-time
- Temporary: 12

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Computer Science, System Architecture and Networking

General research area of SAN

- Networked, resource-constrained embedded systems
  - techniques, concepts, approaches, results, prototypes
- Corresponding research fields
  - distributed systems, networking, parallel computing
  - architecture, in particular, software
    - compositional systems
    - non-functional requirements
  - real-time techniques
    - resource sharing (QoS), resource allocation (budgets)
    - performance analysis
  - embedded VLSI techniques
- Strong participation in (inter)national projects

Agenda

- Ambient Intelligence
- System and software aspects from AmI
- ITEA projects ROBOCOP/Space4U
- ITEA project CANDELA

Ambient Intelligence by aspect

IST Advisary Group, 2001
pictures from the Philips site in 2001

- Ambient Intelligence implies a seamless environment
  - of computing,
  - advanced networking technology
  - and specific interfaces.
- It is aware
  - of the specific characteristics of human presence
  - and personalities,
- takes care of needs
- and is capable
  - of responding intelligently to spoken or gestured indications of desire,
  - and even can engage in intelligent dialogue.
- Ambient Intelligence should also be
  - unobtrusive,
  - often invisible:
    - everywhere and yet in our consciousness – nowhere unless we need it.
- Interaction should be
  - relaxing and enjoyable for the citizen,
  - and not involve a steep learning curve.

Ambient Intelligence by aspect

IST Advisary Group, 2001

pictures from the Philips site in 2001

Do we want this?

- Only the nice part
  - not the hassle
  - not the danger
  - .... imposes additional challenge
    - must be able to tranparently control this environment
- Moore's and Metcalf's laws indicate this direction
  - something like this is happening anyway
  - so let's steer it
- Excellent as a 'man on the moon' concept
  - just imagine the steps that must be taken to make it happen

Yet, what do we have now?

- .... a seamless environment
  - of computing,
  - advanced networking technology
  - and specific interfaces ???

- We have this environment, except the ‘seamless’ and perhaps the specific interfaces
  - bits and pieces exist
  - but without the ‘seamless’, forget about the other AmI stuff

- The ‘intelligence’ is still sub-optimal

Computing

- Typical AI technologies for adaptivity and intelligence
  - data mining
  - neural networks, genetic algorithms

- Context awareness
  - models of user and environment
    - interpretation context for information
    - knowledge base for determining responses

- Distributed systems technology
  - global tasks through cooperation
    - no complete information, low resources locally

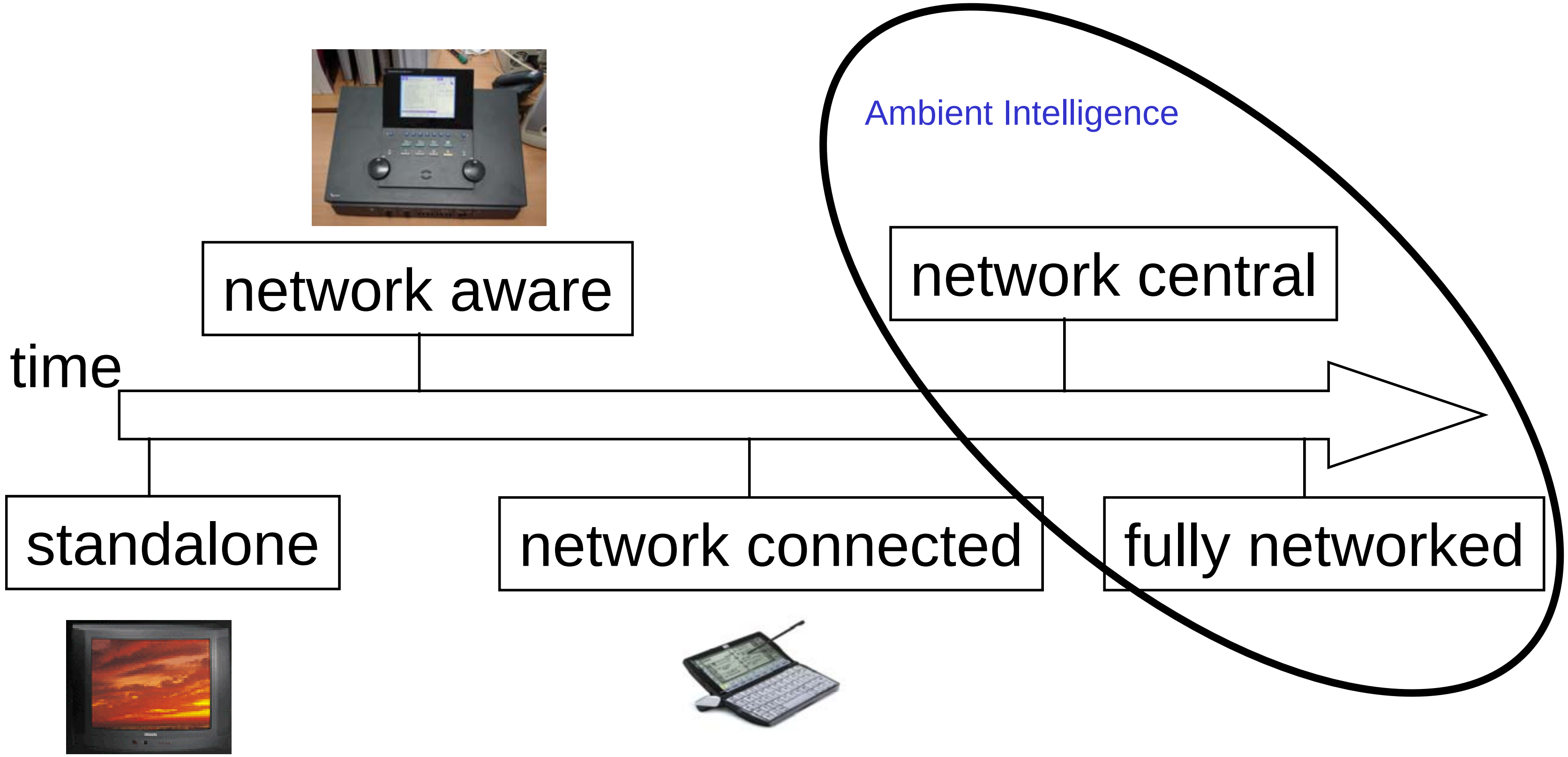
Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Computer Science, System Architecture and Networking

Advanced networking technology

- What for?
  - ambient intelligence is not going to be centralized!
    - at least terminals will be distributed
  - though a centralized computer (in the cupboard is conceivable)....
  - .... the combination of computing/storage/interaction are deployed everywhere (embedded systems)
    - AmI requires these to become part of the ambient environment
    - users cannot configure and control this increasing number of systems

- Then, what technology?
  - Wireless technology
    - general, packet switched
    - specific, e.g. extreme low power
      - wireless sensors, smart dus, smart paint
  - Intelligence in the network
    - adaptive protocols for content delivery
    - QoS control

Evolution of embedded networking



Network central

- Devices have a stand-alone function
  - PDA, TV set
- In addition, they serve as ‘platform’ for networked applications
  - specific device capabilities available on the network
    - display, internal hardware, internal memory
  - support for hosting (networked) applications
    - components that can be down- and re-loaded routinely

Consequence

- Highly distributed applications
  - composed of the cooperation of small services
- Highly mobile code
  - services and applications realized in independent components that can be down- and re-loaded routinely
- Embedded knowledge
  - decisions taken without direct user involvement
    - at most some steering
  - need a reference that separates good from bad decisions
    - model, learning, feedback-control loops
  - intelligence at many levels
    - protocols, algorithms, system

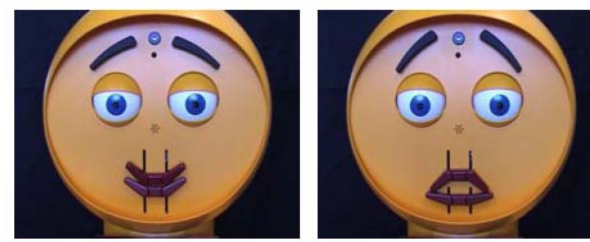
Fully networked

- No stand-alone function
  - dedicated, single function components
    - e.g. networked storage, internet radio
  - cheap devices, elementary behavior
    - sensing, actuating, computing, communicating (sensor networks)
- Applications arise from cooperation

Specific interfaces

- *Move interfaces from the computer domain to the domain of the user*
  - sensing and actuating replaces keyboard and mouse
    - metaphores, analogies
  - screens and displays replaced by other types of feedback
    - natural interaction
  - media content analysis
    - speech recognition
    - video segmentation

Marble answering machine by Durrell Bishop

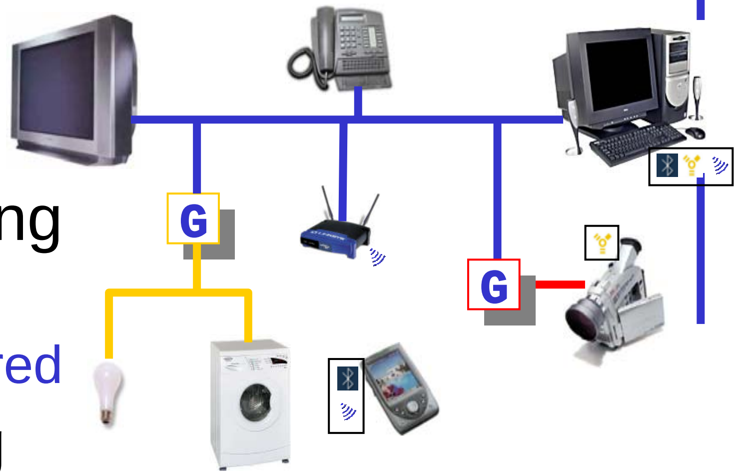


Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Computer Science, System Architecture and Networking

Missing: integration

- How come?
  - consumer products are bought over a long period
    - diversity!
    - must be designed to integrate in an evolving environment
    - we're better at designing complete systems ("solutions") than in "design for evolution"
      - this may be expensive
  - fixed decisions must be replaced by policies
    - the key is postponing decisions, seeing a device or function as part of a whole – software not prepared
    - towards humans/application << 1
  - applications involving several devices are complicated
    - particularly, real-time, reliability and other extra-functional properties

Example: simple AmI scenario



- Person enters house
  - visual recognition
- Light is switched on by speaking
  - speech recognition
  - preferences of person remembered
- TV is switched on by speaking
  - speech recognition, different control target
- Relevant messages pop up
  - TV-on event is recognized; TV-display is used for messages
- Someone rings – picture on screen, name spoken
  - visual recognition – found place where to display
  - speech synthesis

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Computer Science, System Architecture and Networking

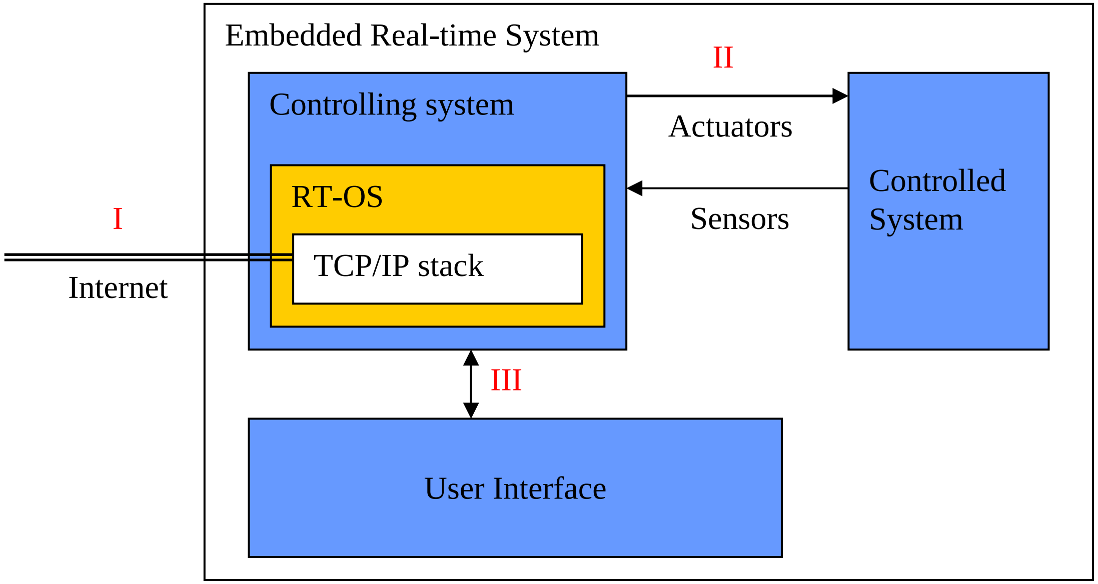
Scenario thinking

- We can realize such a scenario
  - given enough time and money we can make just about anything
- But:

> *Can we make system components such that future, as yet unforeseen, cooperations and adaptations are simply realizable, and actually work?*

- Note:
  - need to re-think the role networking / distribution plays in system design
  - do not only look at typical end-user functionality, but also at *hidden aspects*

Example: classical embedded real-time system



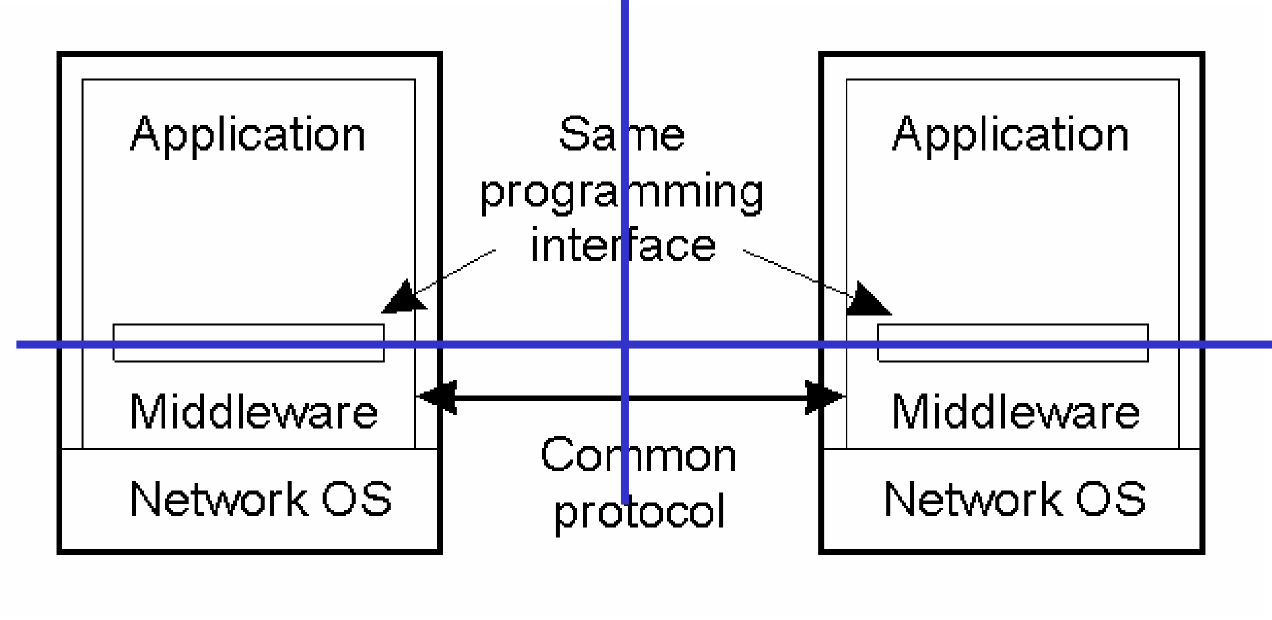
- Network options
  - I : just a connected device/system
  - II : remote sensing/control (could combine with e.g. fieldbus technology in practice)
  - III : remote monitoring and control
  - IV: ......
- Making the protocols open increases the possible applications

Requirements on software architecture

- Interoperability (who would not want this....)

Interoperability views

Interoperability focussed
on protocol; no language or platform
binding besides message structure
and semantics



Hide network details
by extending platform
services
Binding mostly through
libraries

Requirements on software architecture

- Interoperability (who would not want this....)
- Loose coupling
  - clear interfaces and dependencies
  - late binding
    - even at run-time:
      - advertisement, discovery
      - based on descriptions
  - avoidance of language, OS, ISA binding
- Composability
  - including extra-functional properties
- Software upgrade
  - routinely: context aware updates
  - trading
- Location transparency

Concluding

- Inside devices: component framework
  - routine
  - dynam

    - det s and performance
  - maintain integrity
- On the network: *“service oriented approach”*
  - devices ex s: “information faces”
    - discover
    - control a hrough open protocols
    - basic, or tionality is exposed
  - applications coordinate services

Separation of components and coordination

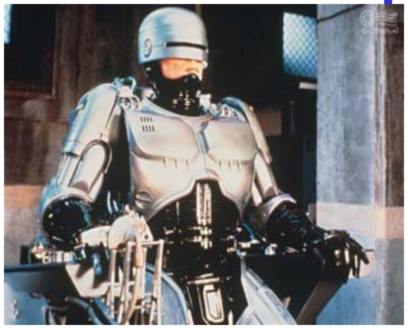
Agenda

- System aspects following from Ambient Intelligence
- ITEA projects ROBOCOP/Space4U
  - component model & runtime environment
  - download framework
  - context aware configuration
  - system integrity management
- ITEA project CANDELA

Background

Research is part of the following projects

- Robocop (2001 - 2003)
  - Define an open, component-based framework for the middle-ware layer in high-volume consumer devices (robustness/reliability, upgrading/extension, and trading)



- Space4U (2003 - 2005)
  - Extend and validate the Architecture
    - Fault Management
    - Power Management
    - Terminal Managment

The Robocop Project

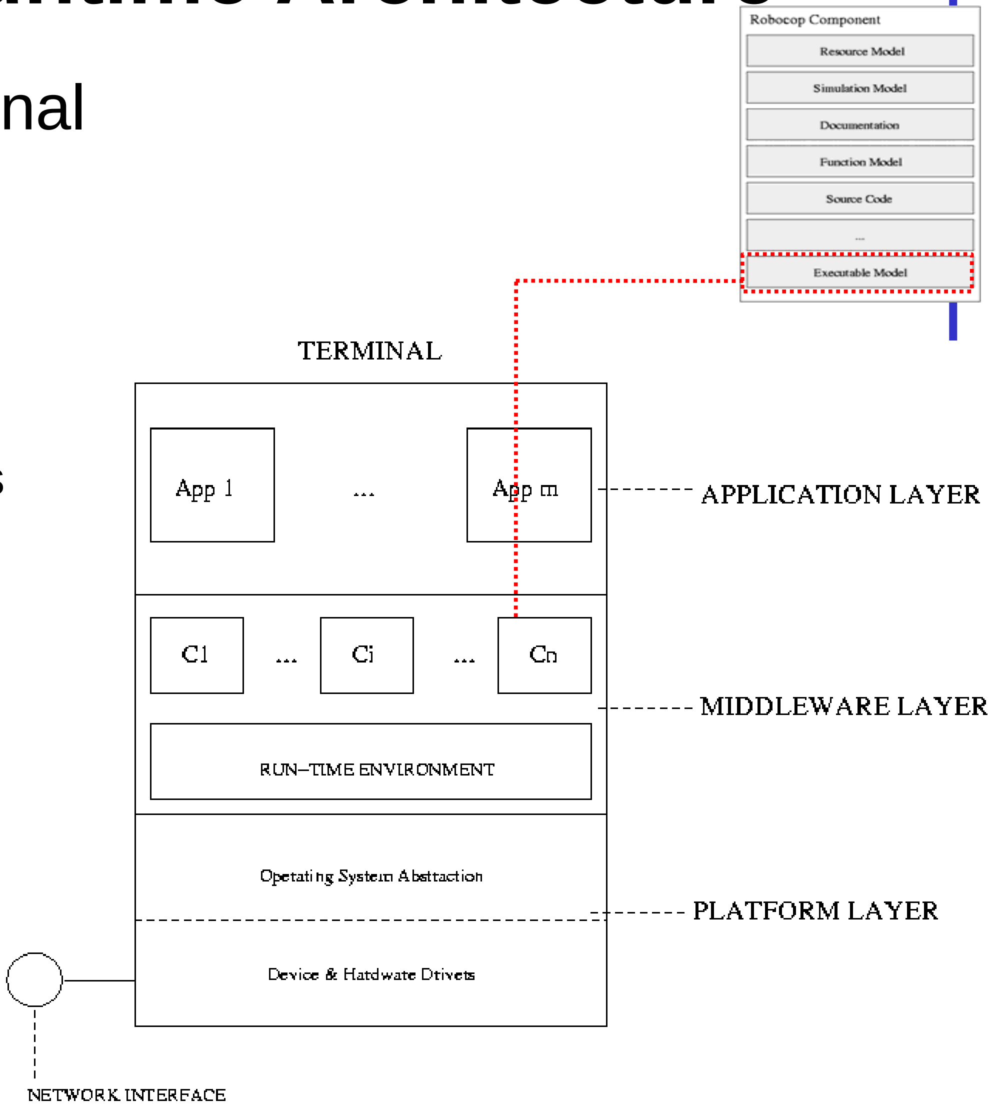
- Project timeline: 2 years
  - July 2001 through June 2003
- Partners
  - 5 Countries
    - Finland, France, Netherlands, Spain, Switzerland
  - Categorization
    - 5 Industrial (Philips, Nokia, CSEM, IKERLAN, FAGOR)
    - 2 SME (SAIA Burgess, Visual Tools)
    - 2 Universities (Univ. Madrid, TU/e)
- Financials
  - 111 FTE, 21.5 M Euro

Space4U

- Same Partners
- 86 FTE

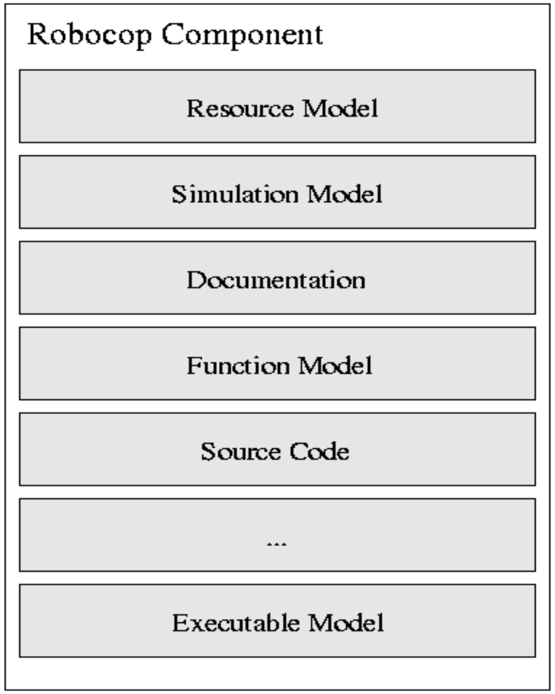
ROBOCOP: Runtime Architecture

- Run-time view of a terminal
  - Application Layer
    - Applications
  - Middleware Layer
    - Run Time Environment
    - Executable Components
  - Platform Layer
    - OS Abstraction
    - Device & HW drivers



Component Packaging

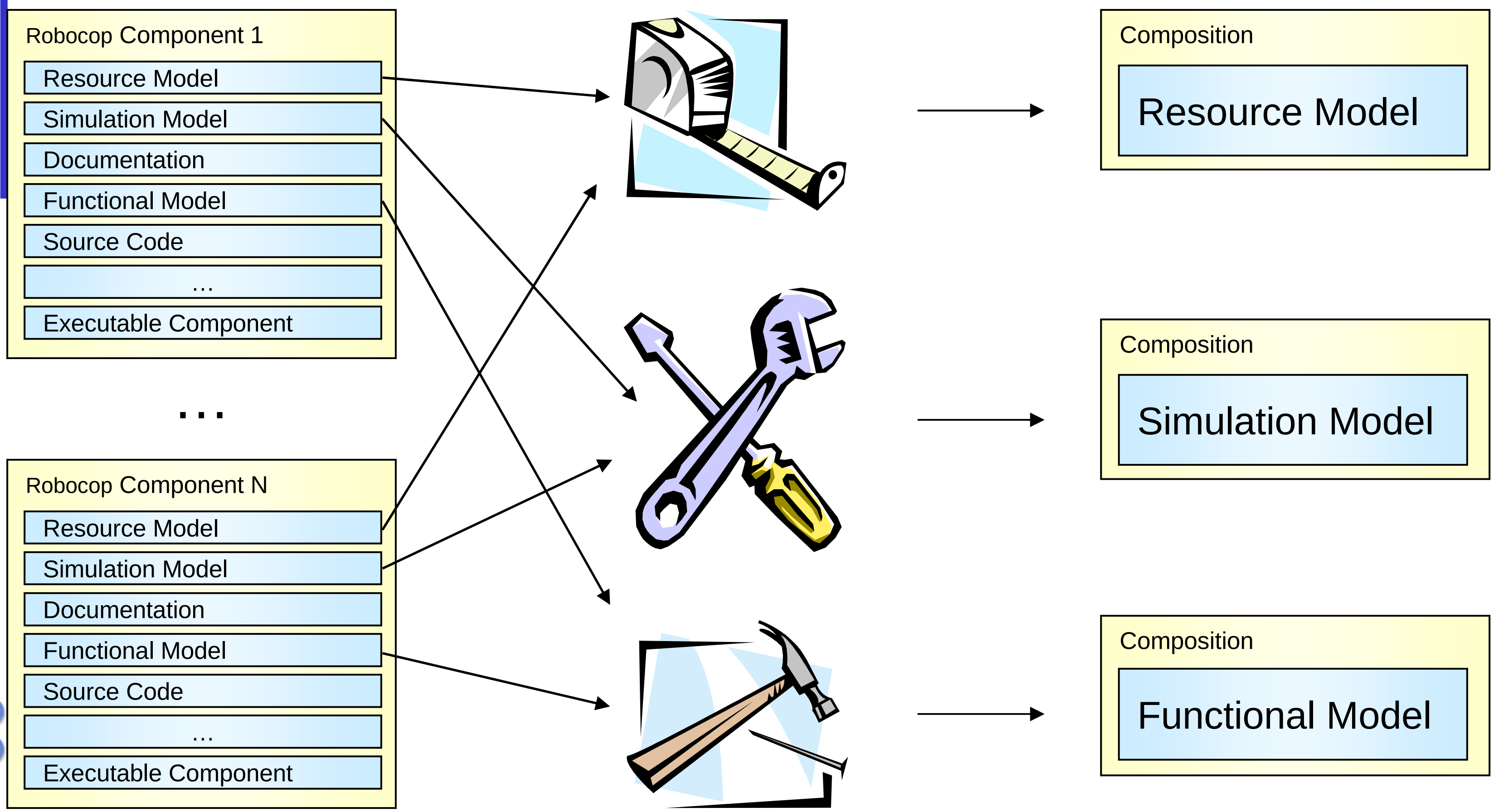
- A Robocop component is a set of related models



Component Packaging (Motivation)

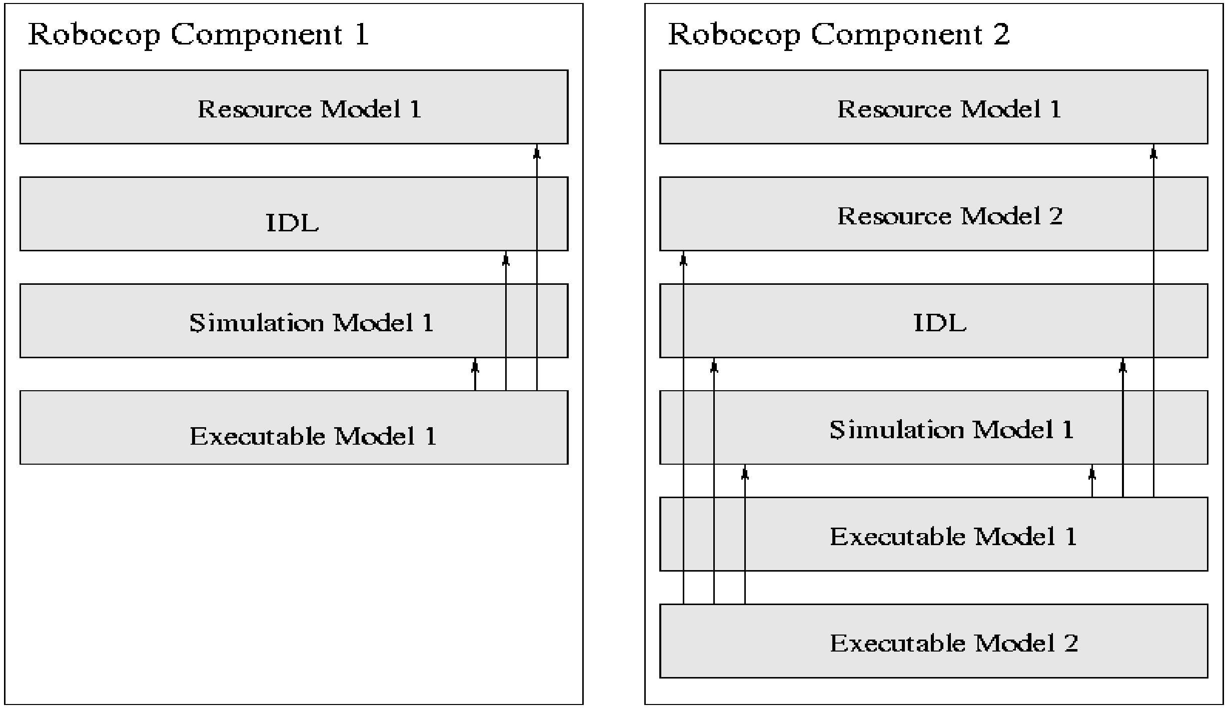
- Trading
  - Different views for different stakeholders
    - Executable for consumer
    - Source code, documentation for developer
    - ...
  - Desirable to trade more than binaries
- Analysis
  - During Development
    - Simulations / Analysis for feasibility tests
  - At Run Time
    - Admission tests during downloading of components

Component Packaging (Vision: Tool Based Composition)

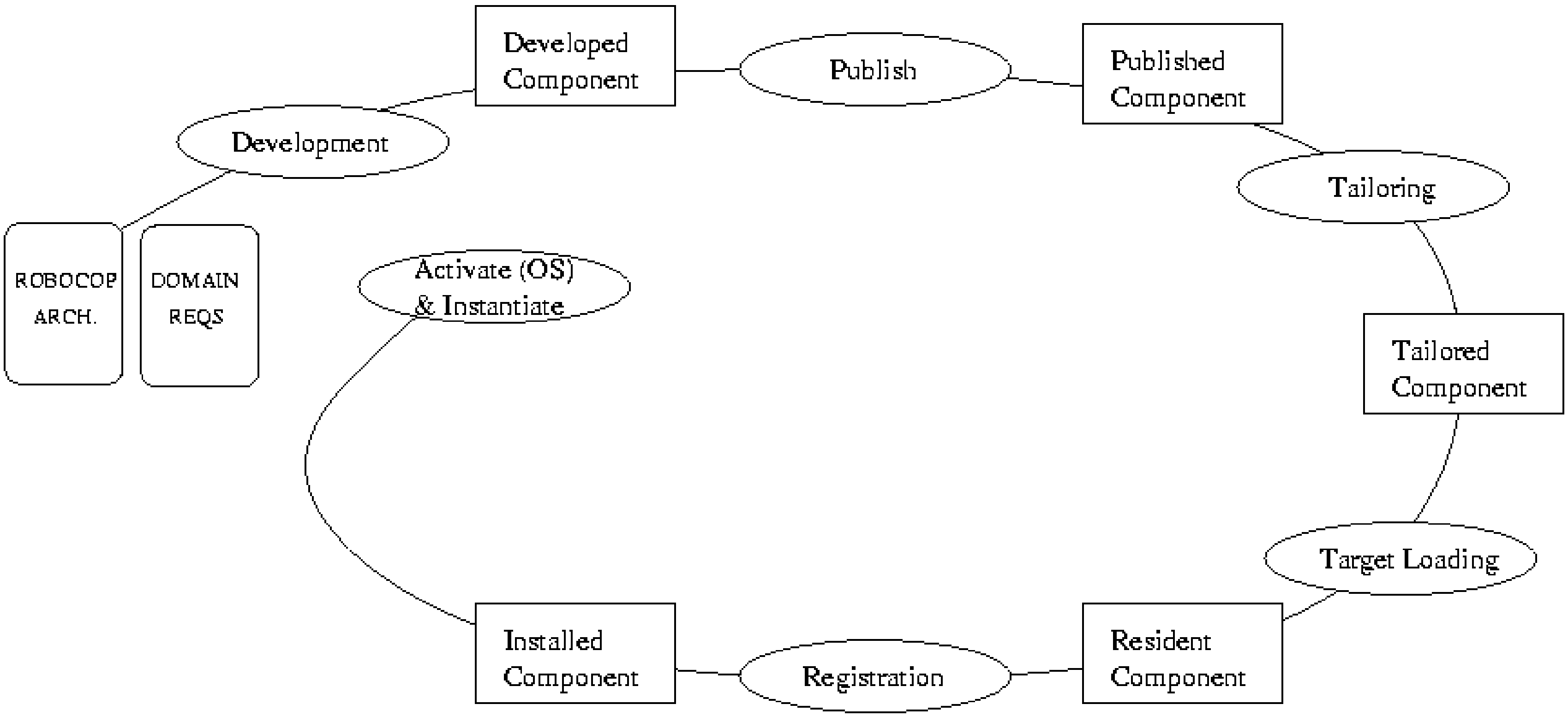


Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Computer Science, System Architecture and Networking

Component Packaging (Example)



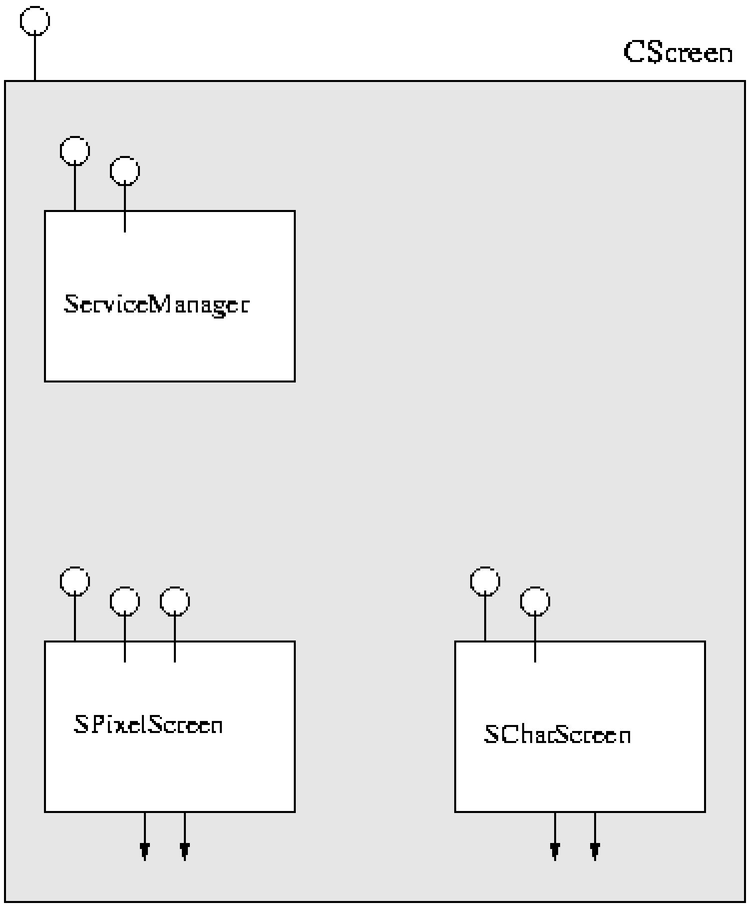
Component Life-cycle

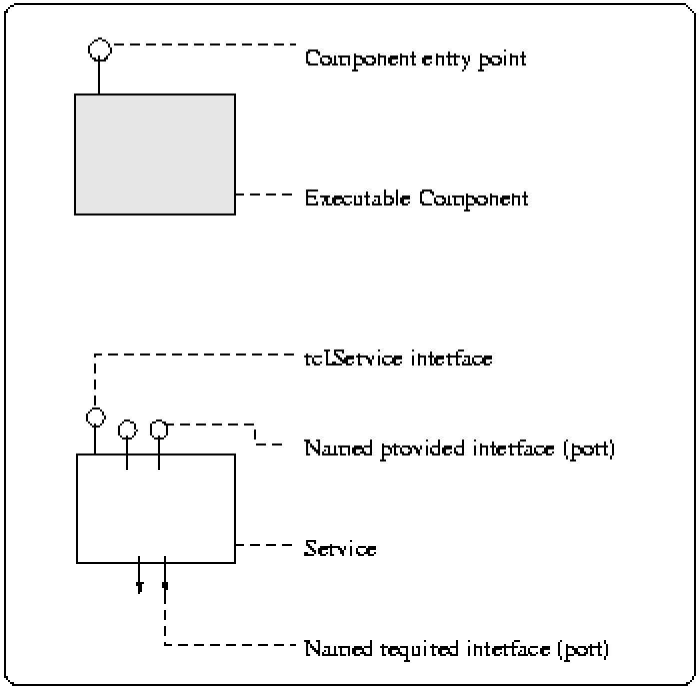


Executable Component (or model)

- Executable Components implement a number of Services
- Executable Components are instantiated in OS terms
  - Static in process (LIB)
  - Dynamic in process (DLL)
  - Dynamic out process (EXE)
- Executable Components have a fixed entry point for
  - Registration to Run Time
  - Retrieving Service Manager
    - Service Manager is used for instantiating services

Executable Component (Example)





Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Computer Science, System Architecture and Networking

Services

- Services offer their functionality through a set of ports (named interfaces)
- Services have explicit dependencies: required ports (named interfaces)
- An Interface is a set of operations
- Services are instantiated at Run Time
  - Service ≈ Class in object oriented programming
- Service Instance is an entity with its own data and a unique identity
  - Service Instance ≈ Object in object oriented programming

Run-time Environment

- Responsibility

Registry

  - Registration of components and services
  - Handle requests for services instances (and services managers)
  - Offer support for QoS (Optional)
- Implementation
  - Three tables
    - Association between Component (ID) and Location
    - Association between Component (ID) and Service (ID)
    - Complies relation between Services (IDs)

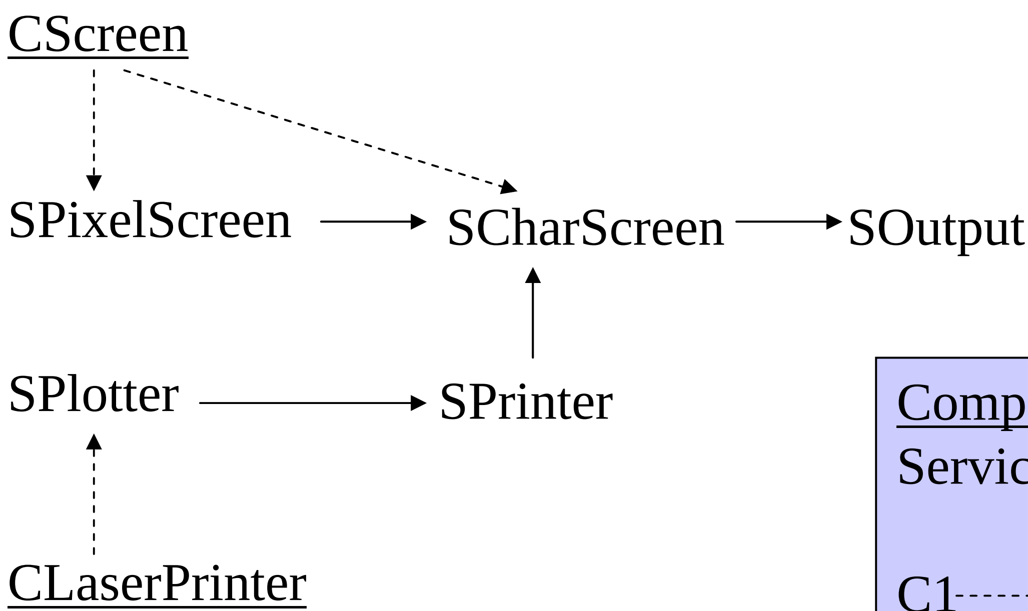
Run Time Environment (Example Registry Content)

Component Location	
component guid	component url
CScreenGUID	file://CScreen.so
CLaserPrinterGUID	file://CLaserPrinter

Component-Service Rel.	
component guid	service guid
CScreenGUID	SCharScreenGUID
CScreenGUID	SPixelScreenGUID
CLaserPrinterGUID	SPlotterGUID

Complies Rel.	
service guid	complies with
SCharScreenGUID	SOutputGUID
SPixelScreenGUID	SCharScreenGUID
SPrinterGUID	SCharScreenGUID
SPlotterGUID	SPrinterGUID

Run Time Environment (Example Registry Content)



Components are underlined
Services are not underlined

C1 - - - -> S1: C1 implements S1
S1 ——> S2: S1 complies with S2

Run-time Environment (Service Instantiation)

- Client can do 2 things:
  - Implicit Service Instantiation
    - Request for Service Instance
    - RRE will return reference to Service Instance
  - Explicit Service Instantiation
    - Request Service Manager
    - RRE will return reference to Service Manager
    - Use Service Manager to create Service Instance
    - Service Manager will return reference to Service Instance

Download Framework

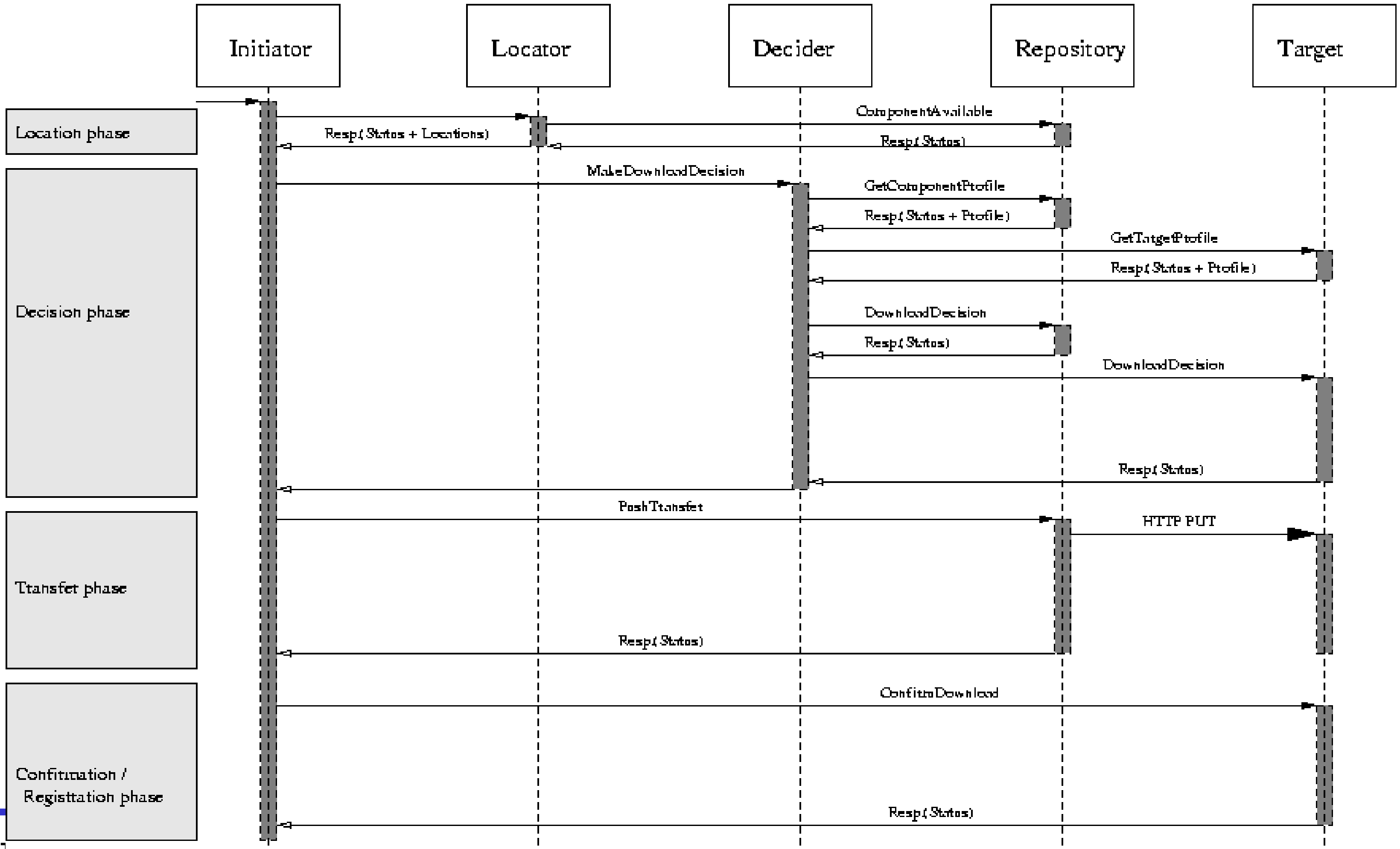
- Responsibility
  - Transfer Robocop components from repository to a target terminal.
- Implementation
  - 5 roles together accomplish the download
    - Initiator
    - Locator
    - Decider
    - Repository
    - Target (needs to be on the terminal)

Download Framework (Key Features)

- Low Resource Footprint on Target
  - Only the target role needs to be resident on the target terminal
- Supports external initiation of download
  - Initiator can be resident on a external server
- Supports decision on suitability of a component for a specific target
  - Decider role

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Computer Science, System Architecture and Networking

Download Framework (Procedure)



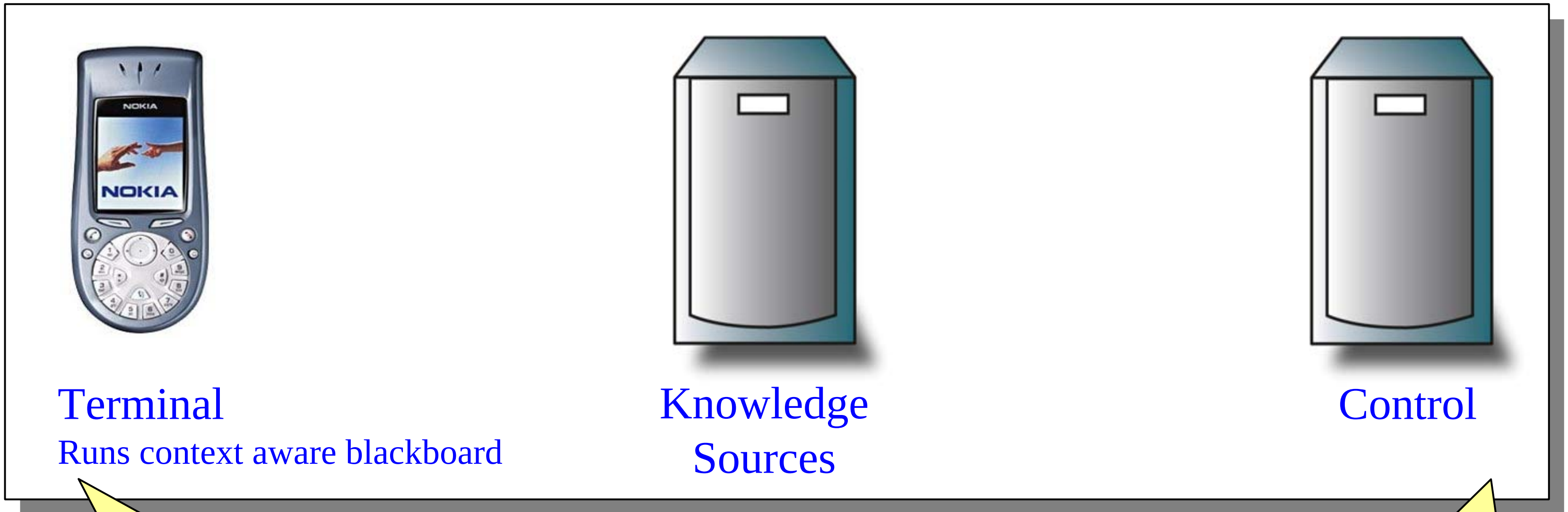
Context Aware Configuration

- Goal
  - Run-time adaptation of the software configuration of the terminal based on the context in which it is used.
- Approach
  - Adapt configuration by using a number of knowledge sources that verify a specific condition and know what to do if this condition is TRUE.

Context Aware Configuration (Basic Idea)

Approach: Update configuration based on context using 3 roles !

Responsibilities: of the individual roles …



•Maintain context information
•Facilities for updating

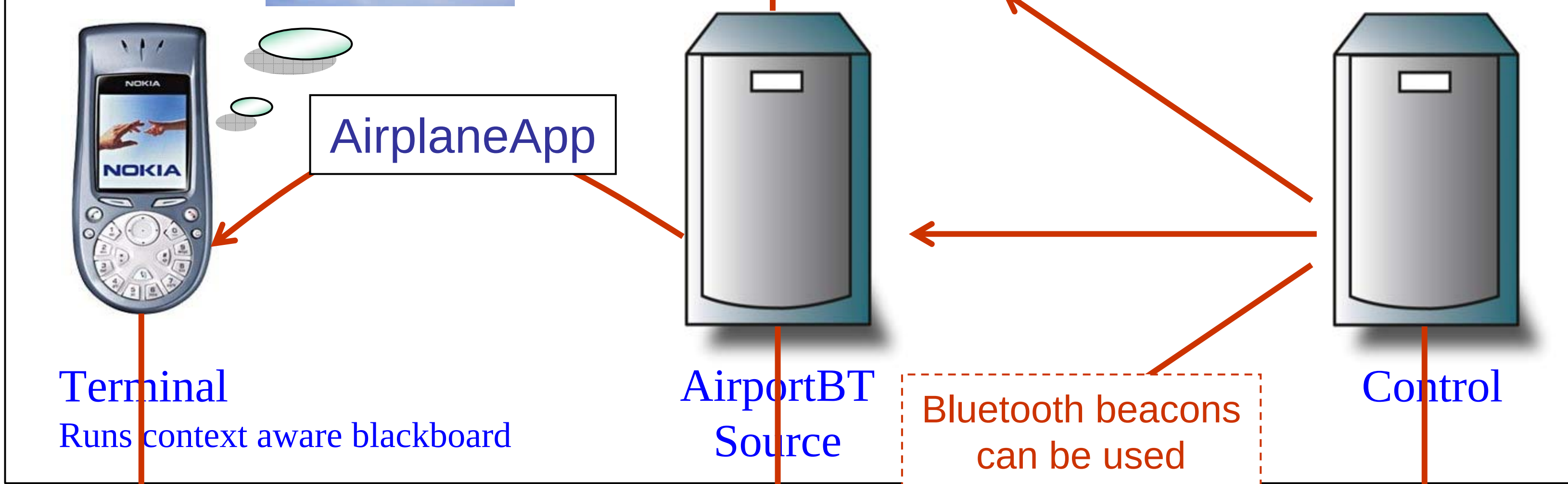
• Detect context changes
• Adapt configuration based on rules

• Control Knowledge Sources

Context Aware Configuration (In Practice)

Scenario

If the condition is true, the AirportBT Source Executes the corresponding action:
Download airplane arrival application & update blackboard



Terminal is used at the airport.

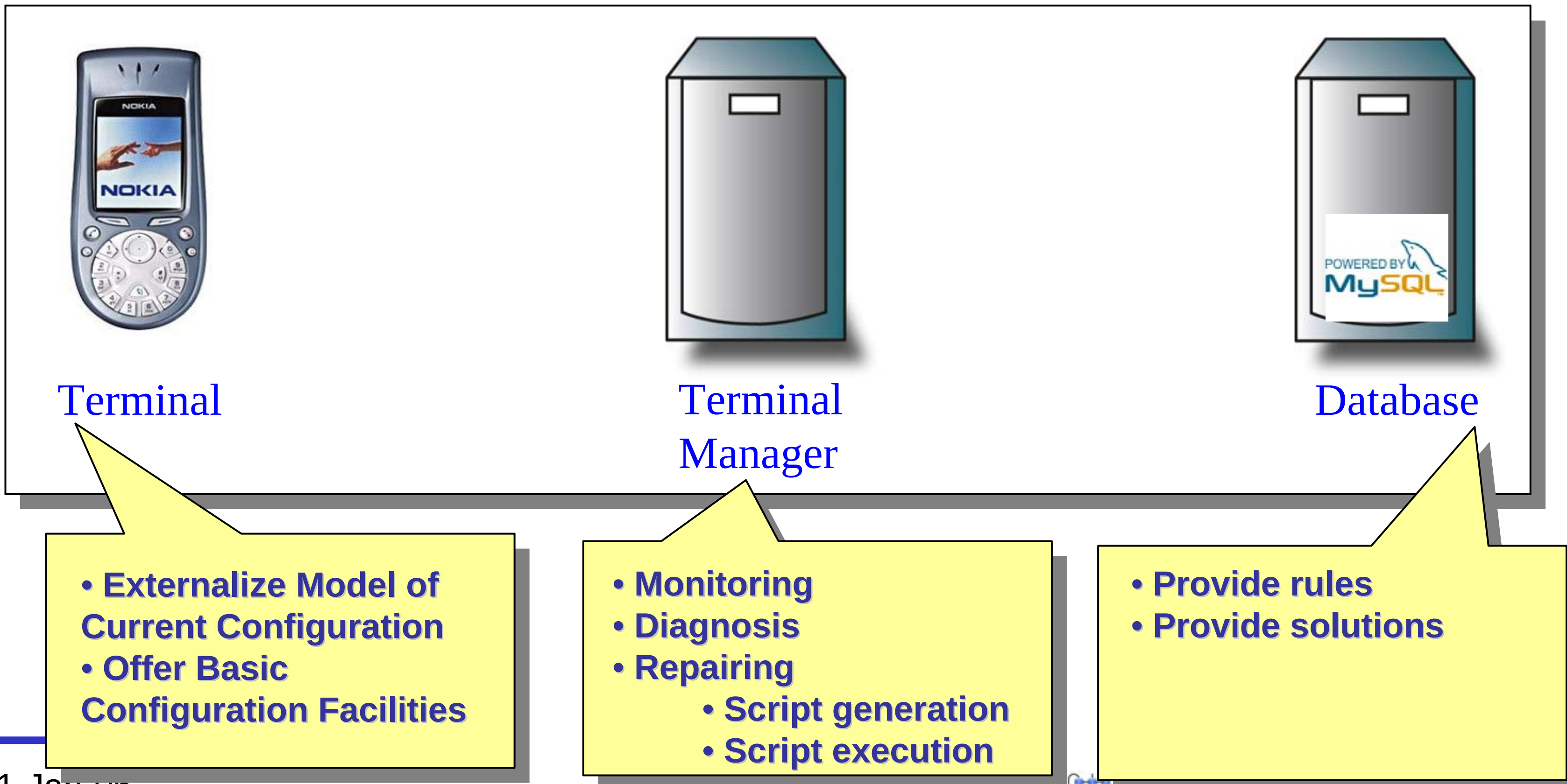
The AirportBT Source verifies a condition:
Am I at the Airport?

Control triggers a Number of knowledge sources

System Integrity Management (Basic Idea)

Approach: Maintaining software integrity using 3 roles !

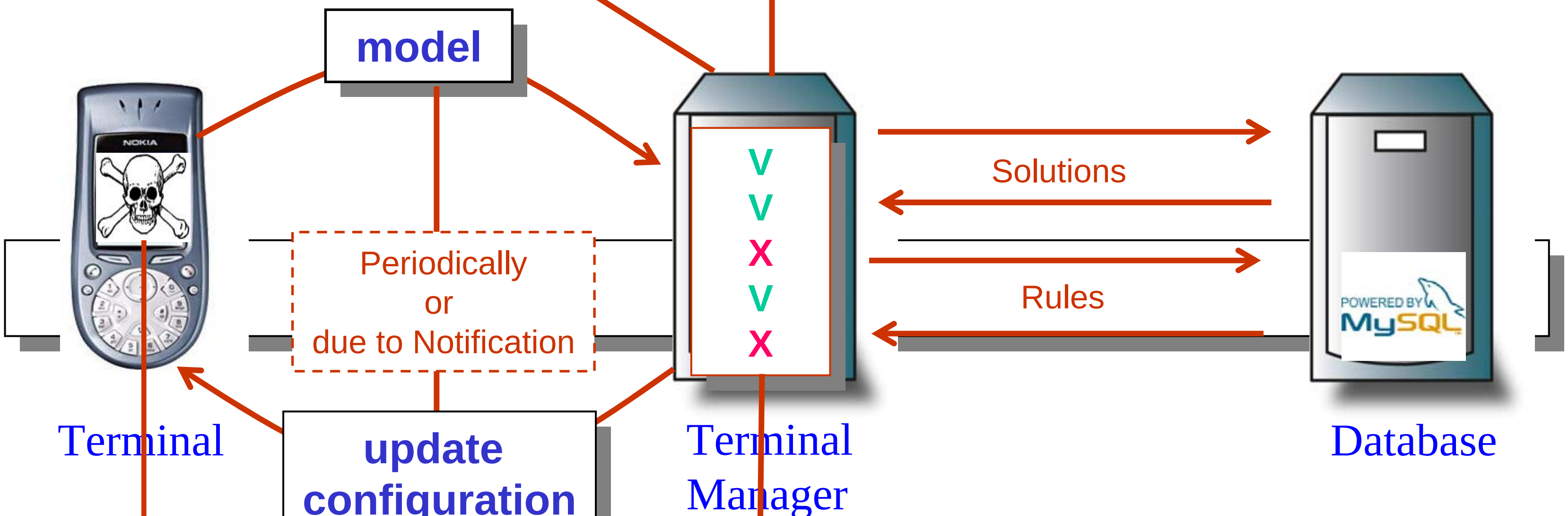
Responsibilities: of the individual roles …



System Integrity Management (In

Terminal Manager will execute repair script using the basic configuration facilities offered by the terminal.

Terminal Manager will generate a repair script based on the outcome of the checks . This might require some knowledge from the database.



Terminal is not working properly

Self Model is retrieved by Terminal Manager

Terminal Manager will do a number of checks. These checks might require knowledge from the database

Agenda

- System aspects following from Ambient Intelligence
- ITEA projects ROBOCOP/Space4U
- ITEA project CANDELA

Remember

- Inside devices: component framework
  - routine
  - dynam
    - det s and performance
  - maintain integrity

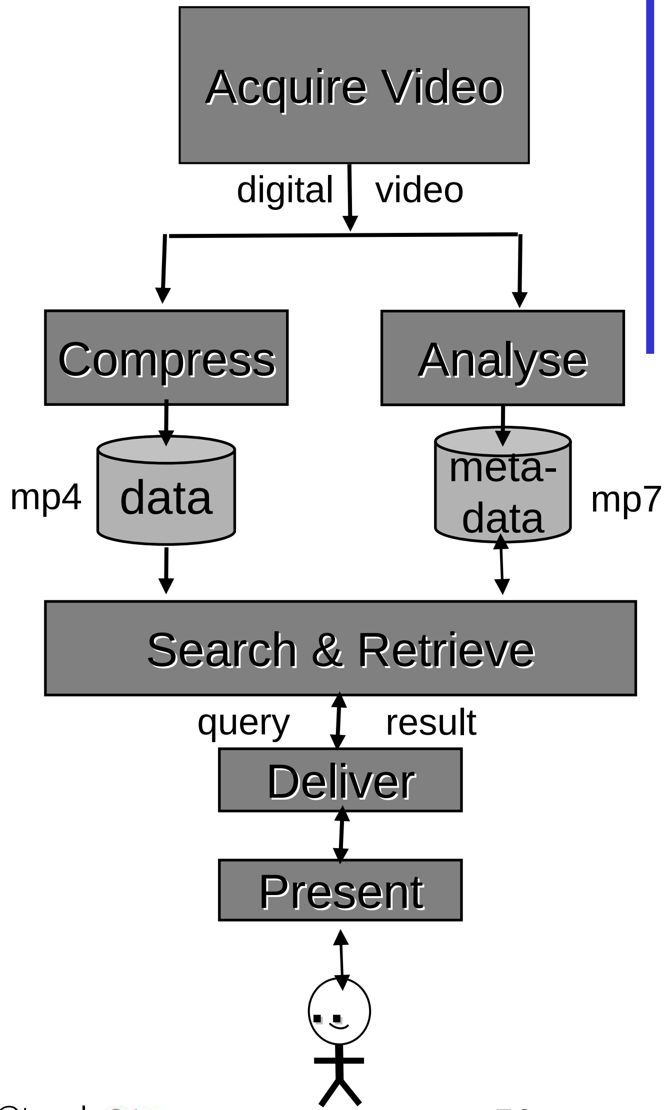
- On the network: *"service oriented approach"*
  - devices ex es: "information faces"
    - discover
    - control a hrough open protocols
    - basic, or tionality is exposed
  - applications coordinate services

Separation of components and coordination

CANDELA

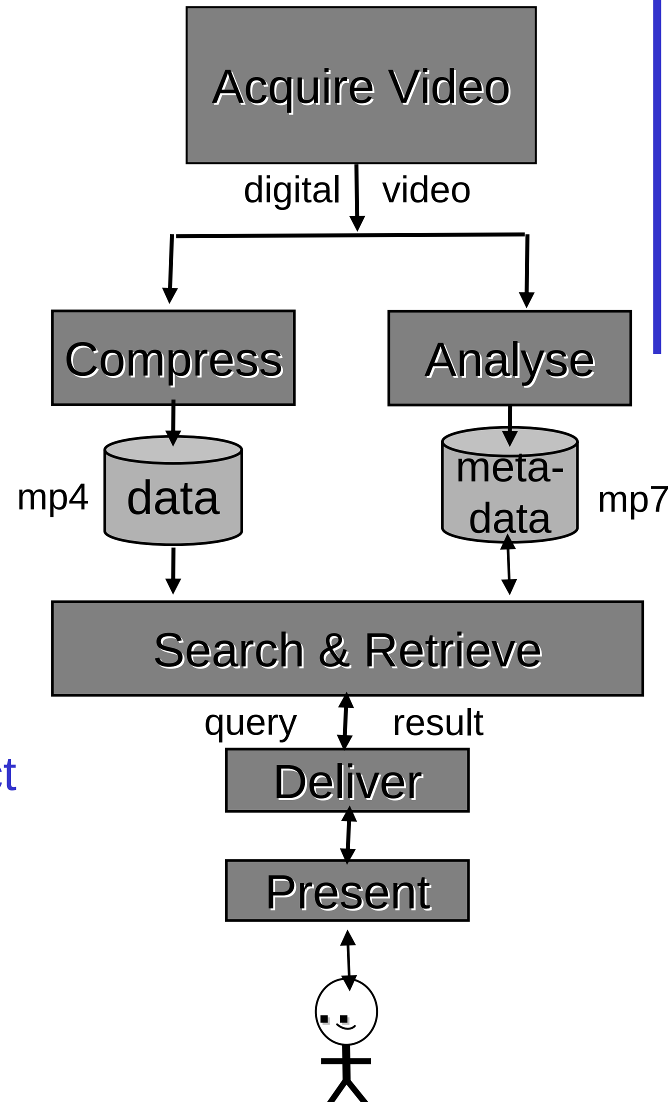
- Content Analysis and Networked Delivery Architectures

- Scope
  - "understand" video
    - improve display
    - improve delivery
    - use as system input



CANDELA – TU/e

- *Context*: home network
- *Focus*:
  - *System architecture* for *Distributed* Video Content Analysis processing
    - devices expose video processing and other capabilities as *service units*
    - video applications (e.g. VCA, viewing, storing) through compositions of these
  - Online processing, flexibility, reliability
- *Connections:*
  - embedded within CASSANDRA project in Philips Research
  - project MultimediaN, iShare



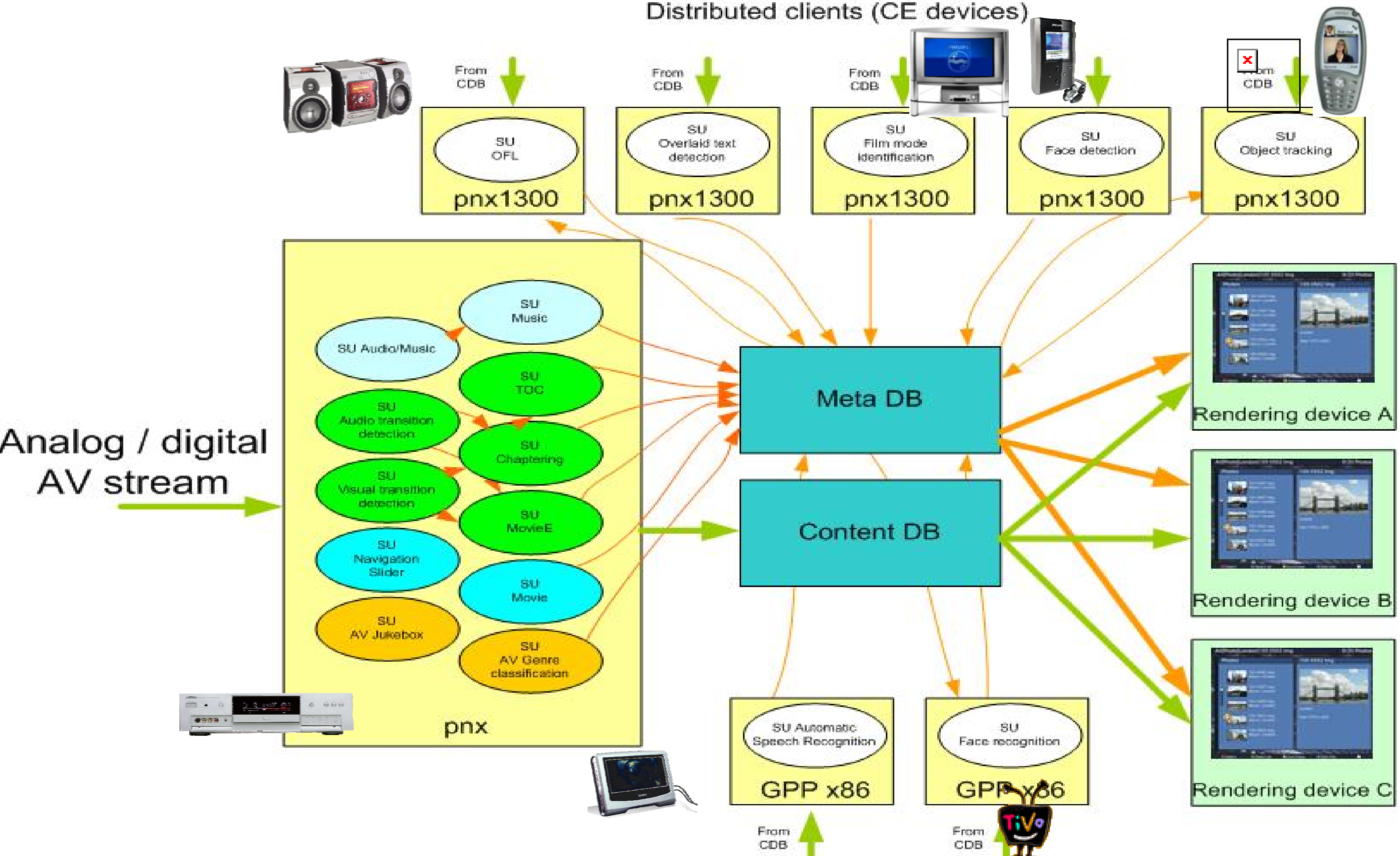
Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Computer Science, System Architecture and Networking

Sharing services

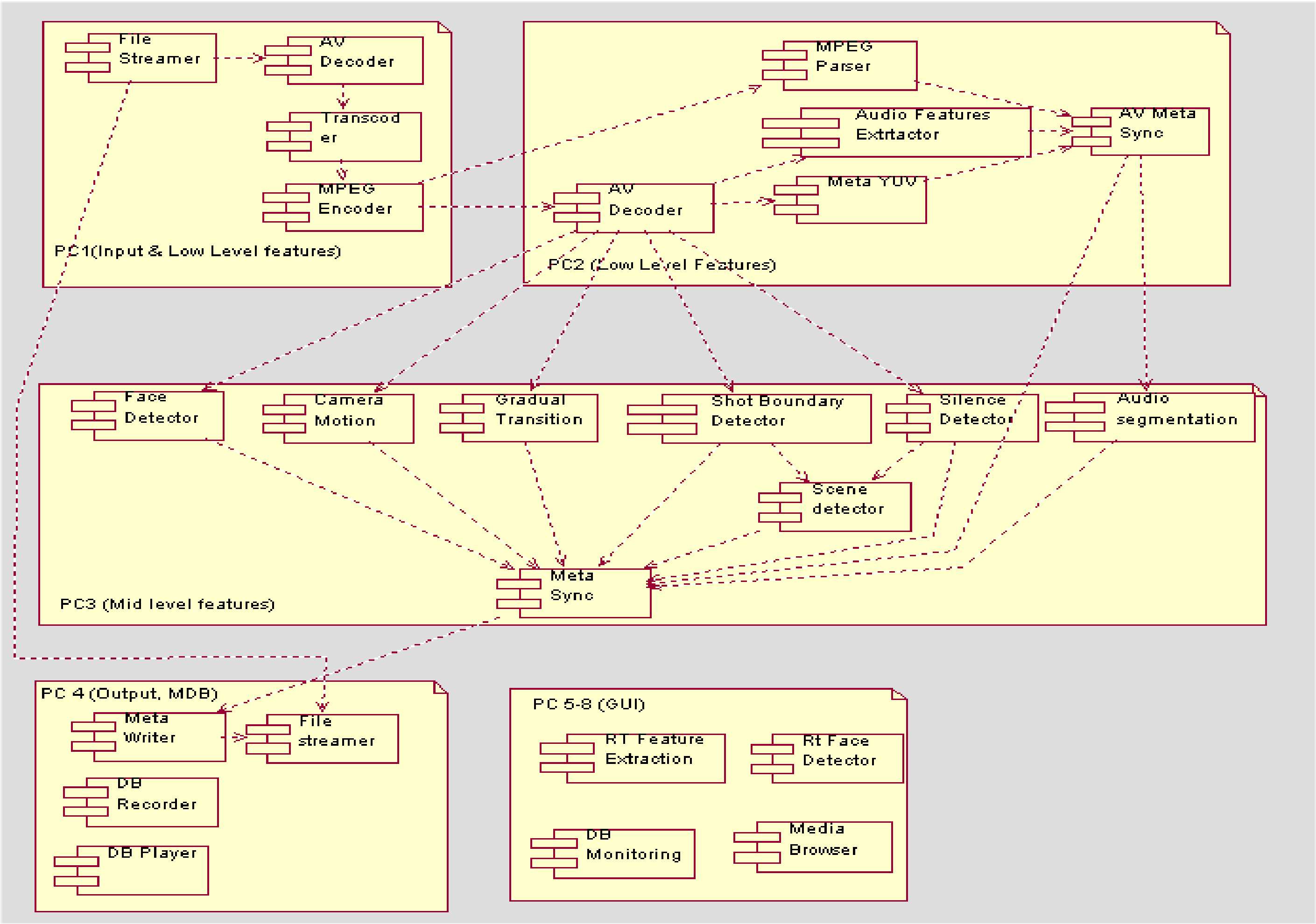
- Devices share
  - functionality
    - e.g. media processing functions
  - resources
    - e.g. database, computational platform
  - content
    - e.g. video
- Accesssible as networked services (“service units”)
- Applications (“use cases”) are compositions of services

CASSANDRA distributed and decentralized content analysis in Connected Home / Planet

Home Network Topologie
Distributed clients (CE devices)



Architecture: deployment view



Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Computer Science, System Architecture and Networking

Focus: reliability

- Applications rely on the cooperation of several devices
  - devices may disappear
  - more points of failures

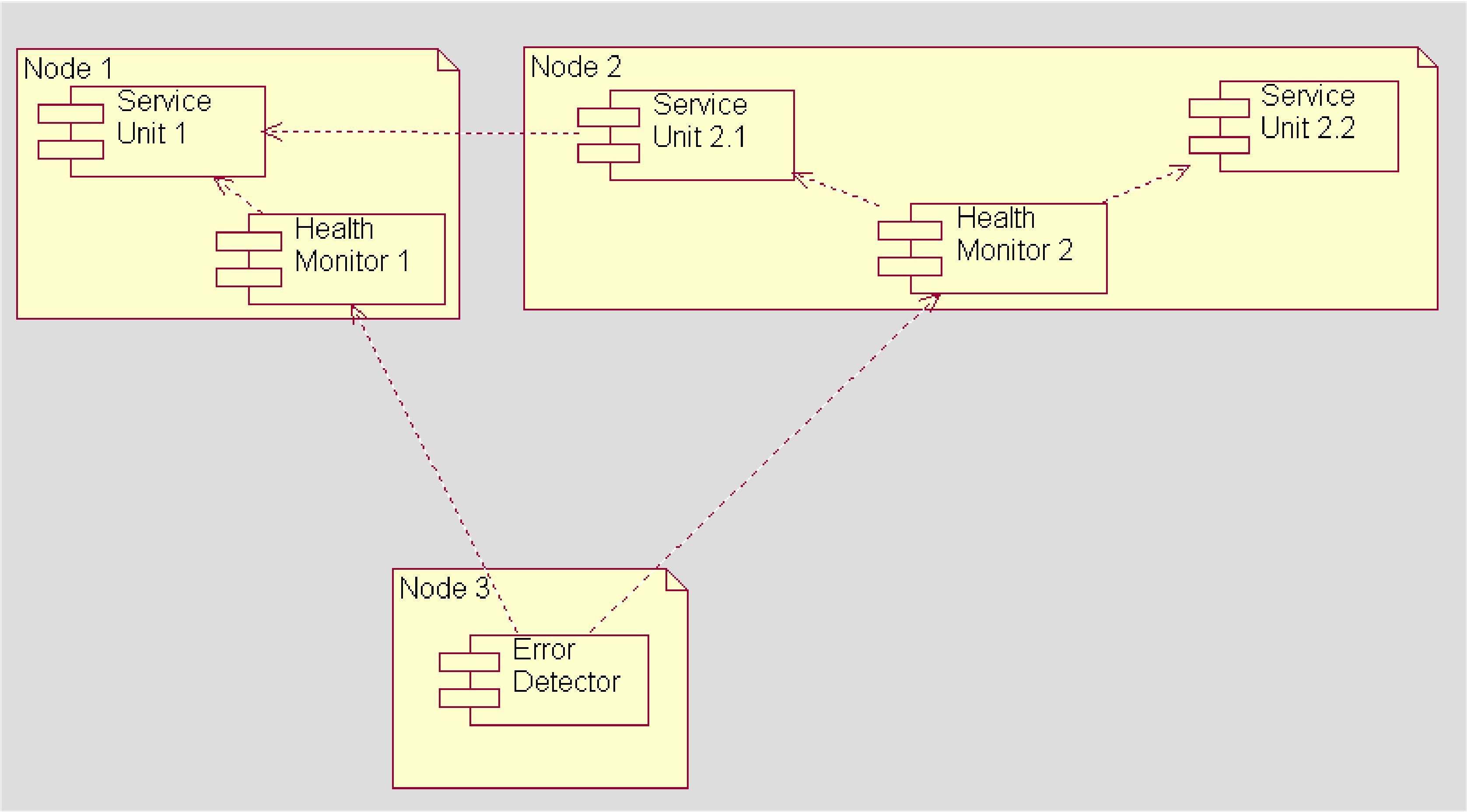
- Considered errors
  - Service units
    - Logical (catchable by a service unit itself)
    - Physical (failstop of process running the service)
  - Host (device) breakdown
  - Communication (network) failure

Johan J. Lukkien, j.j.lukkien@tue.nl
TU/e Computer Science, System Architecture and Networking

Error detection: method

- External (watchdog): external observer
- Internal: adjust the component model such that malfunctioning can be determined
  - Interfaces for communication are equipped with detection capabilities
    - Data
    - Control
  - Components become stoppable
  - Each component defines the error collection that it
    - Can catch itself
    - Cannot handle itself
    - Can report on

Error detection: diagram



Conclusion

- Typical AmI functions requires a service-oriented software architecture
  - both inside and outside terminals
- Service composition must include extra-functional properties
- Functionality (services) and coordination (compositions) should be decoupled

Some conclusions

- Ambient intelligence requires open, flexible systems
  - cooperation beyond the protocol level
- New perspectives in design
  - service orientation
  - component based
- Embedded intelligence plays at multiple levels
  - intelligence 'stack'
- Embedded intelligence raises strong issues of privacy