Infrastructure and Architectural Principles for Plastic User Interfaces

Joëlle Coutaz, Gaëlle Calvary, Alexandre Demeure (Univ. of Grenoble)

Lionel Balme (Immotronic, Grenoble)

Stéphane Lavirotte, Gaëtan Rey, Jean-Yves Tigli (Univ. of Nice)

Workshop on Ambient Intelligence Infrastructures (WAmil), Pisa, November 2012

Outline

- Contributions of our research to AmI infrastructures : Plastic user interfaces as requirements

- Lessons learned

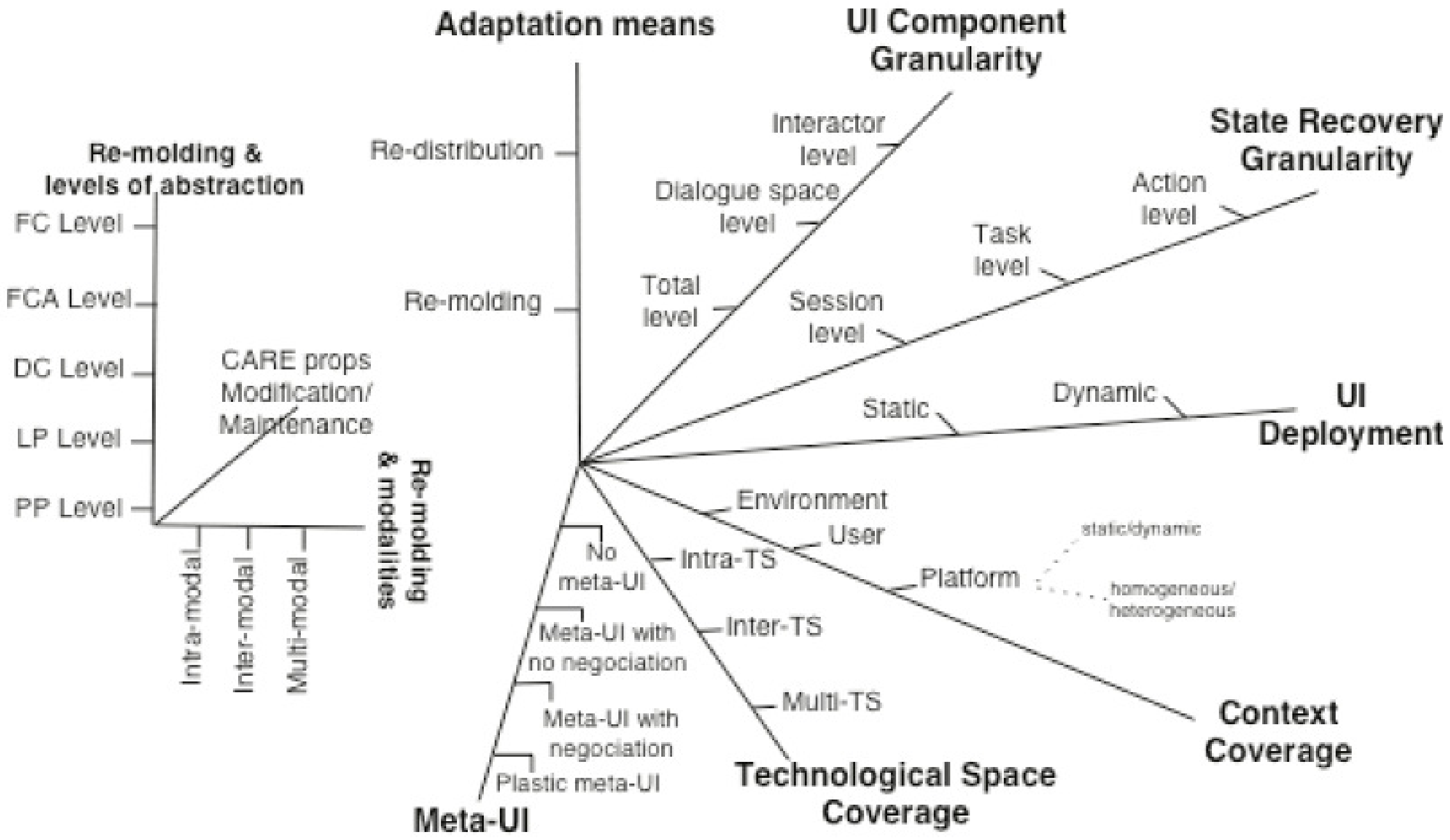
- Perspectives

Plastic User Interfaces: Definition

- User interfaces that are able to adapt to the context of use while preserving utility, usability, value

- Context of use: user, platform, physical environment

Plastic User Interfaces: Problem space



Adaptation means

Re-distribution

Re-molding

UI Component Granularity

Interactor level

Dialogue space level

Total level

State Recovery Granularity

Action level

Task level

Session level

UI Deployment

Static

Dynamic

Re-molding & levels of abstraction

FC Level

FCA Level

DC Level

LP Level

PP Level

CARE props Modification/ Maintenance

Re-molding & modalities

Intra-modal

Inter-modal

Multi-modal

Context Coverage

Environment

User

Platform

static/dynamic

homogeneous/ heterogeneous

Technological Space Coverage

Intra-TS

Inter-TS

Multi-TS

Meta-UI

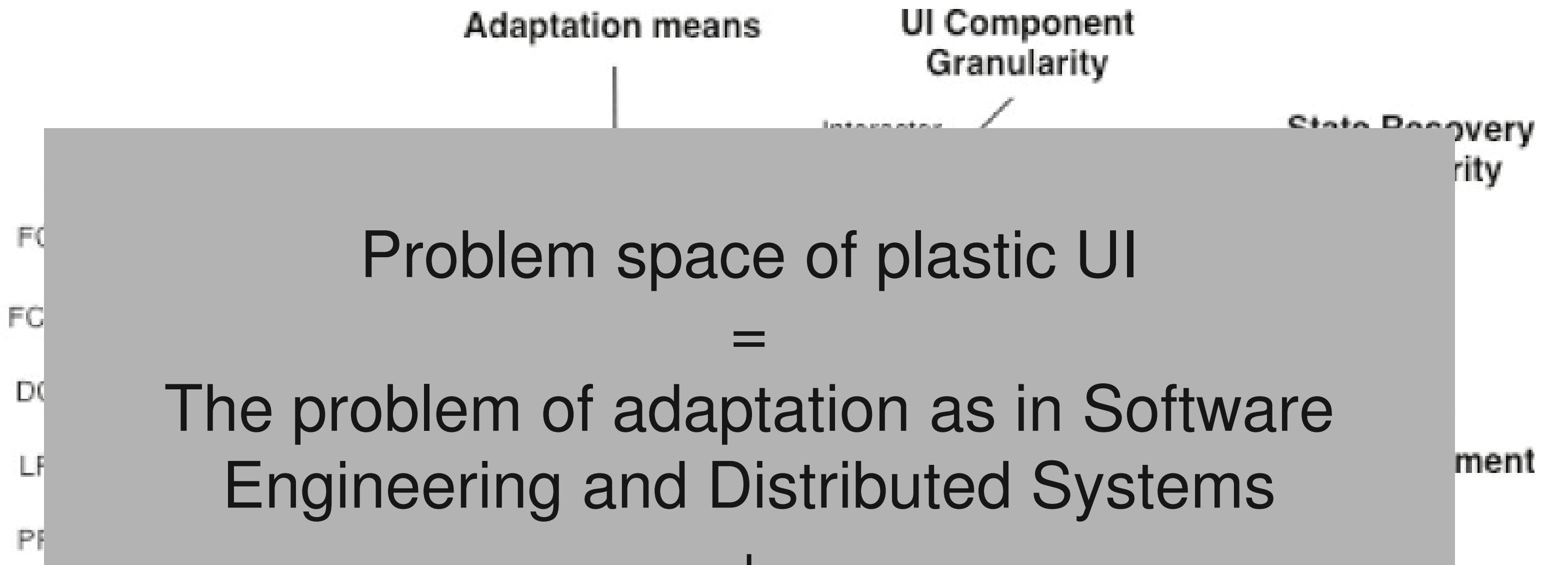
No meta-UI

Meta-UI with no negociation

Meta-UI with negociation

Plastic meta-UI

Plastic User Interfaces: Problem space



Problem space of plastic UI

=

The problem of adaptation as in Software Engineering and Distributed Systems

+

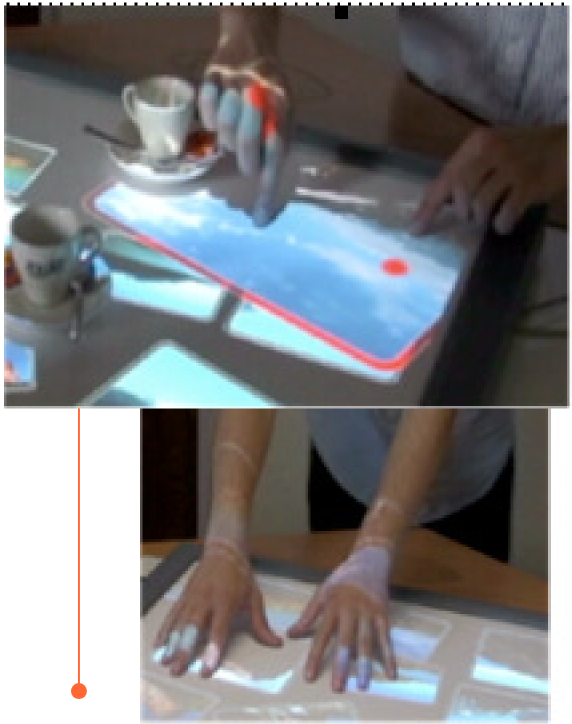
Adaptation “that is visible”

+

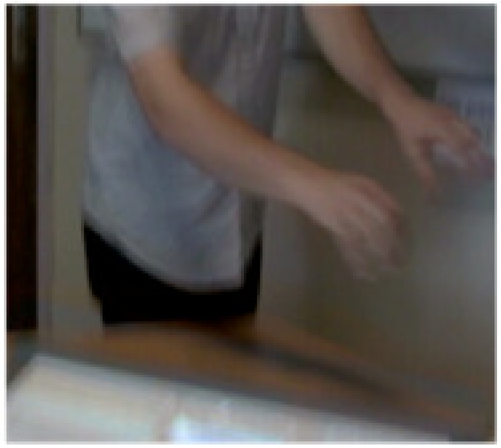
Human control (not too much, not too little) => Meta-UI

Running example: Photo-browser

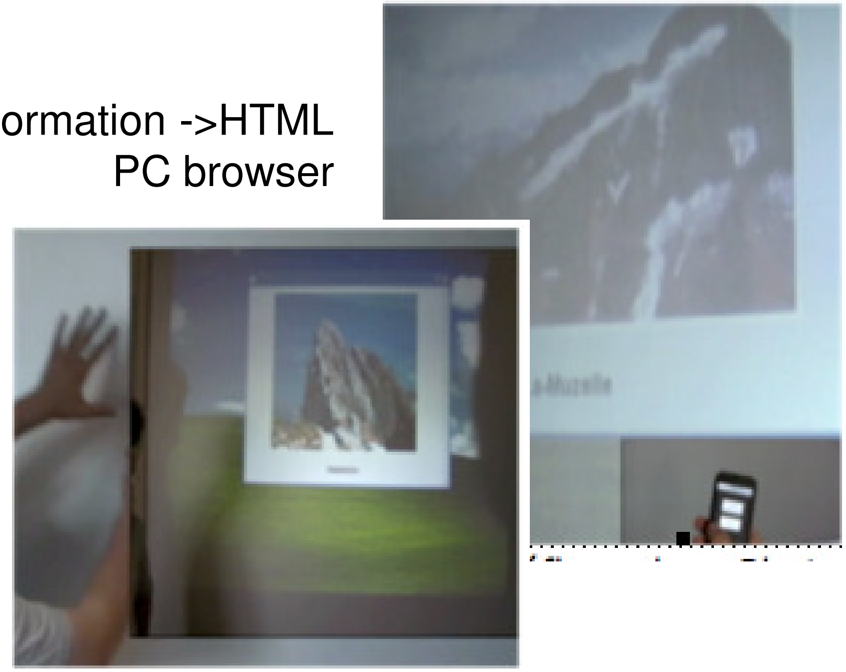
- Dynamicity of the platform
- Heterogeneity of the software components
- Dynamic transformation of some UI components
- UI adaptation via redistribution and remolding
- Gesture-based Meta-UI for human control



Tcl-Tk component
MERL table



On-the-fly transformation ->HTML
PC browser



Java remote controller
Android gPhone

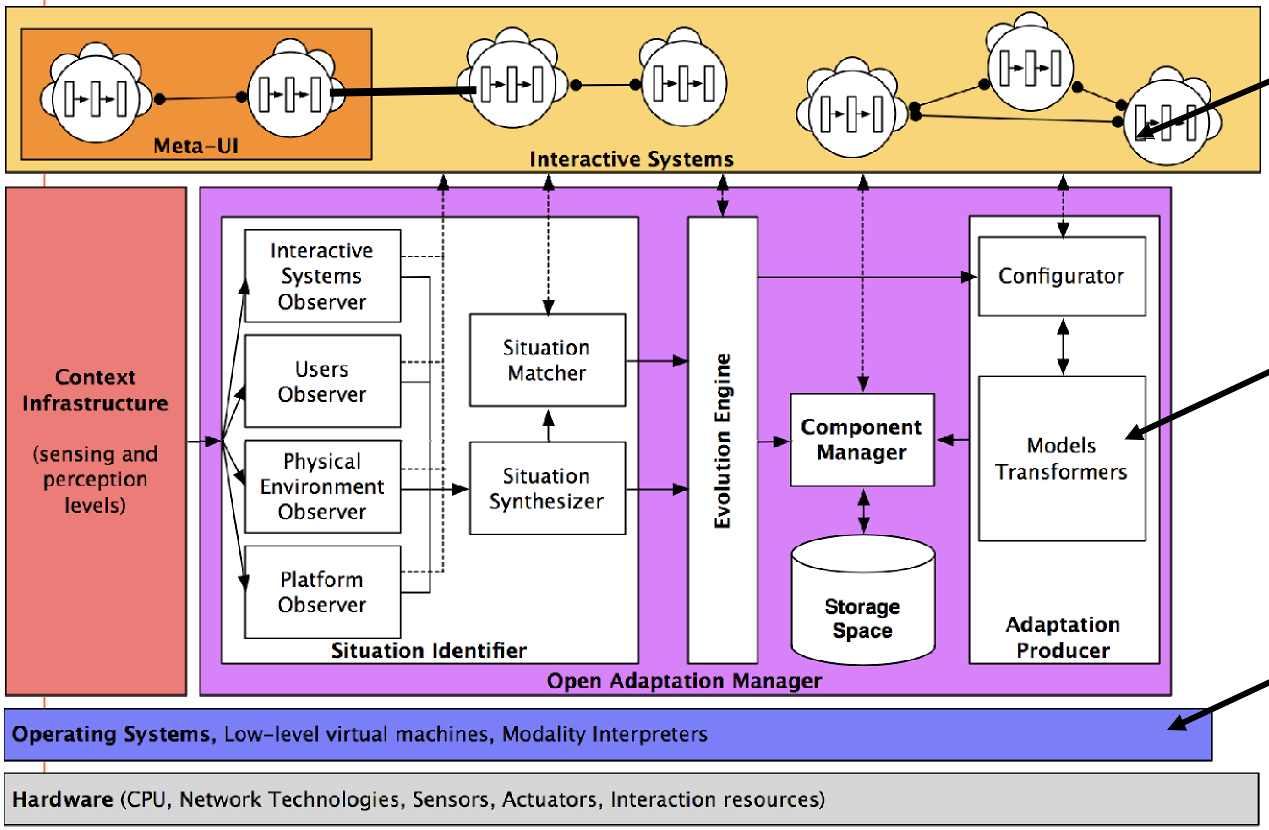
Running example: Photo-browser

Plastic User Interfaces: Principles

- An interactive system as a graph of models that expresses different aspects of the system (e.g., task model, AUI, CUI, FUI) = blurring the distinction between design and run time phases

Plastic User Interfaces: Principles

- An interactive system as a graph of models that expresses different aspects of the system (e.g., task model, AUI, CUI, FUI)
- A mix of close and open adaptativeness on top of a baseline middleware



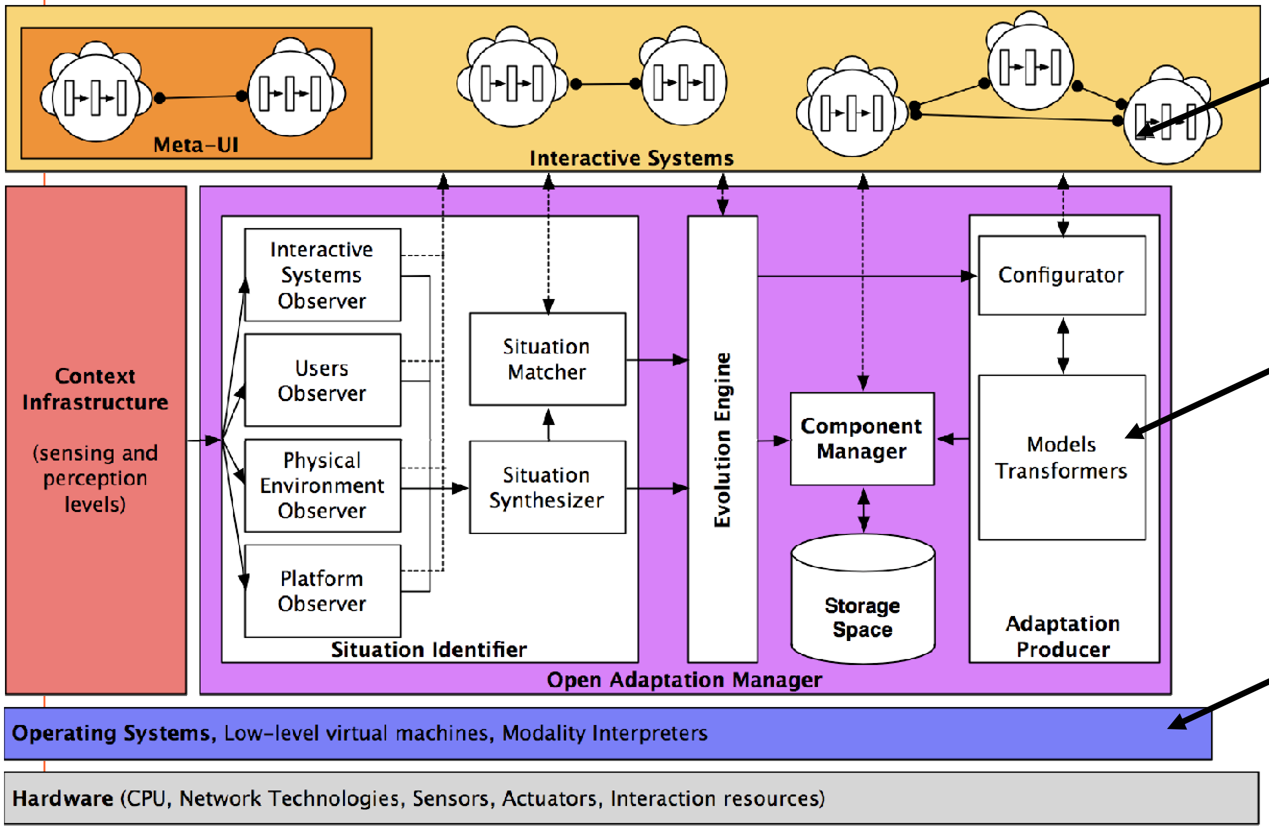
Embedded expression of UI adaptation planned at design time

Externatized UI adaptation supported by an infrastructure

Baseline Middleware

Plastic User Interfaces: Principles

- An interactive system as a graph of models that expresses different aspects of the system (e.g., task model, AUI, CUI, FUI)
- A mix of close and open adaptativeness on top of a baseline middleware



Embedded expression of UI adaptation planned at design time

Externatized UI adaptation supported by an infrastructure

Baseline Middleware

Photo-browser on top of WCOMP, a service-oriented middleware (univ. Nice)

- Components are encapsulated as UPnP devices
- An application is a configuration of UPnP proxies
- The meta-UI recognizes human gestures and translates gestures into configuration scripts
- Scripts are dynamically interpreted by a specific component of WCOMP (the AA designer) -> reconfiguration of the application components

```
component1 = *?type=lamp
componen2 = *?type=switch
Advice light_switch (component1, component2):
component2.^StateChange -> (component1.setState)
```

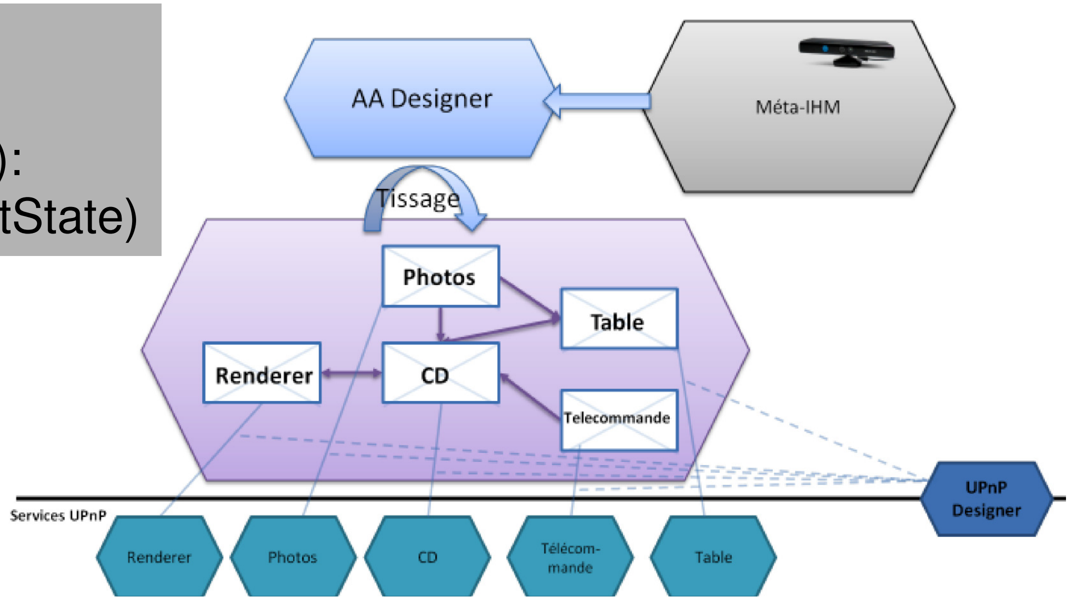


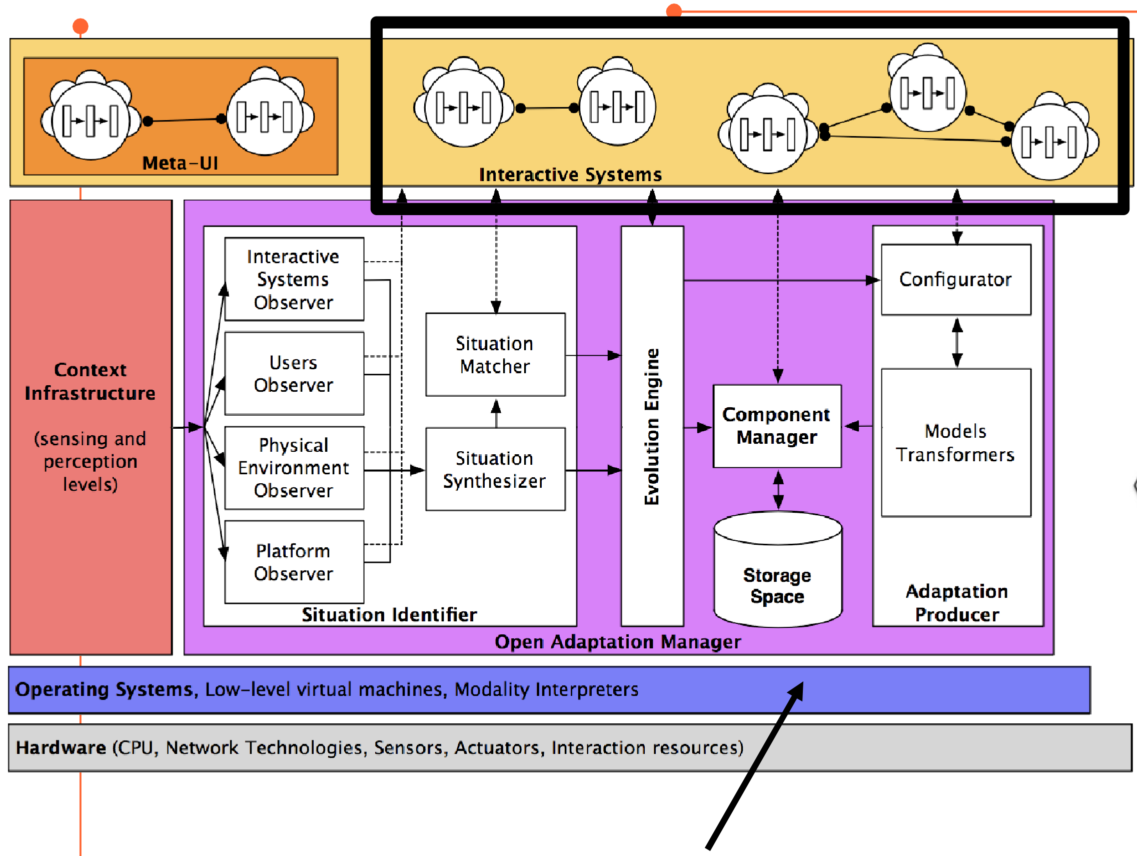
Photo-browser on top of WCOMP, a service-oriented middleware (univ. Nice)

Photo-Browser implemented on top of WCOMP

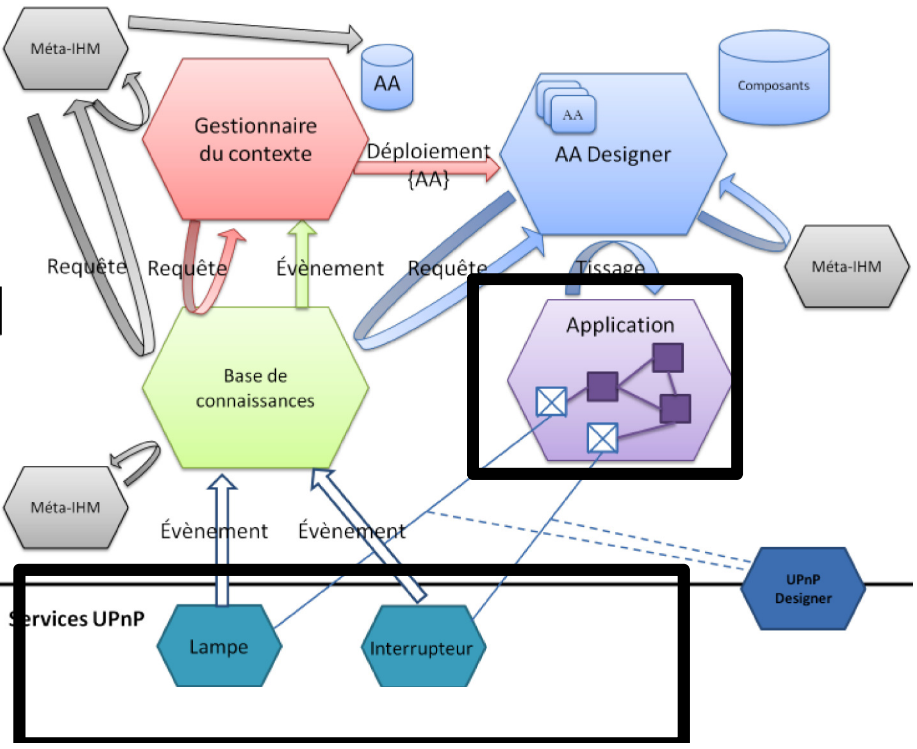
(C) 2010

UJF - LIG - IIHM

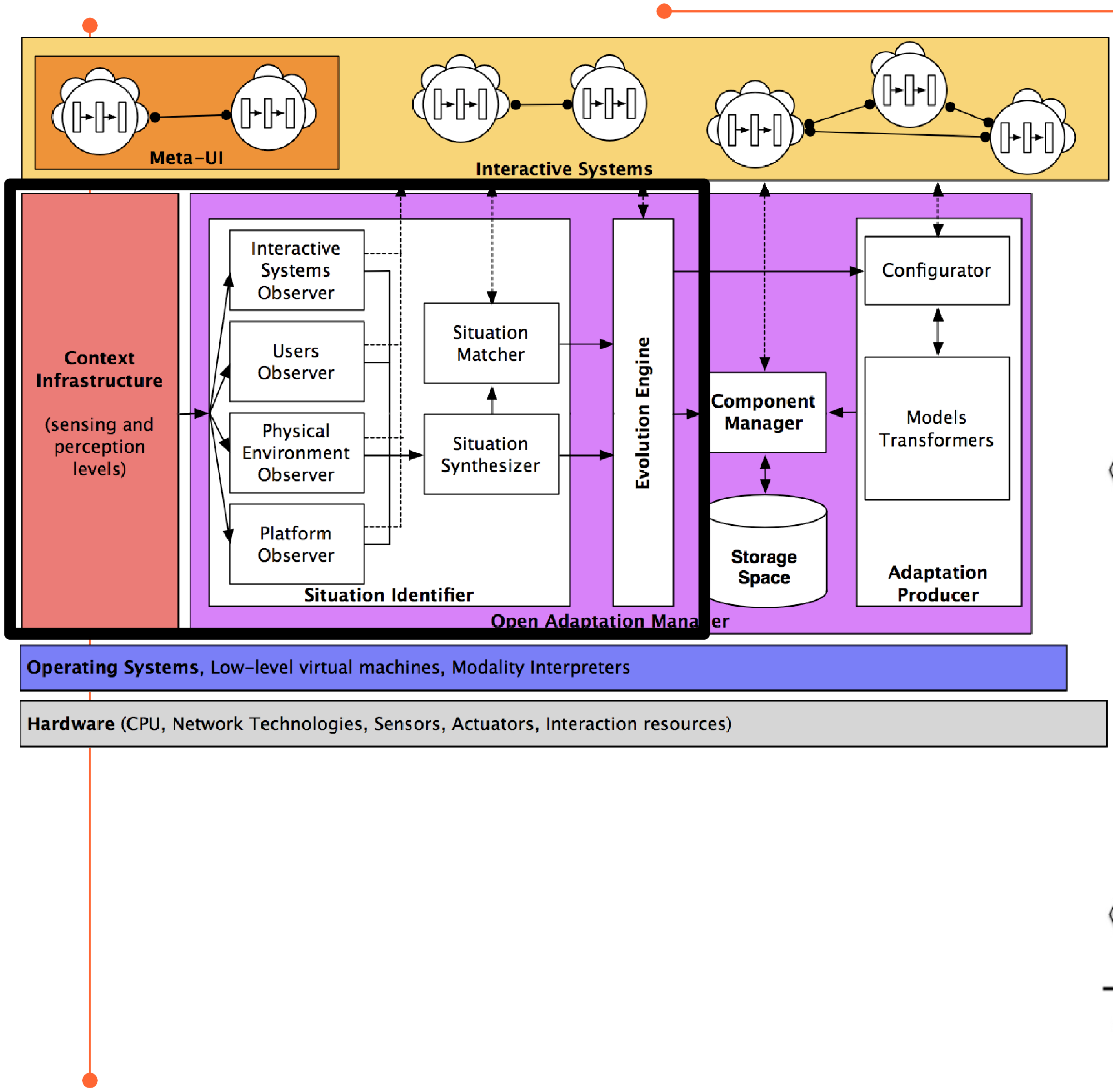
Generalization : from functional decomposition to implemenattion on top of a component-oriented middleware

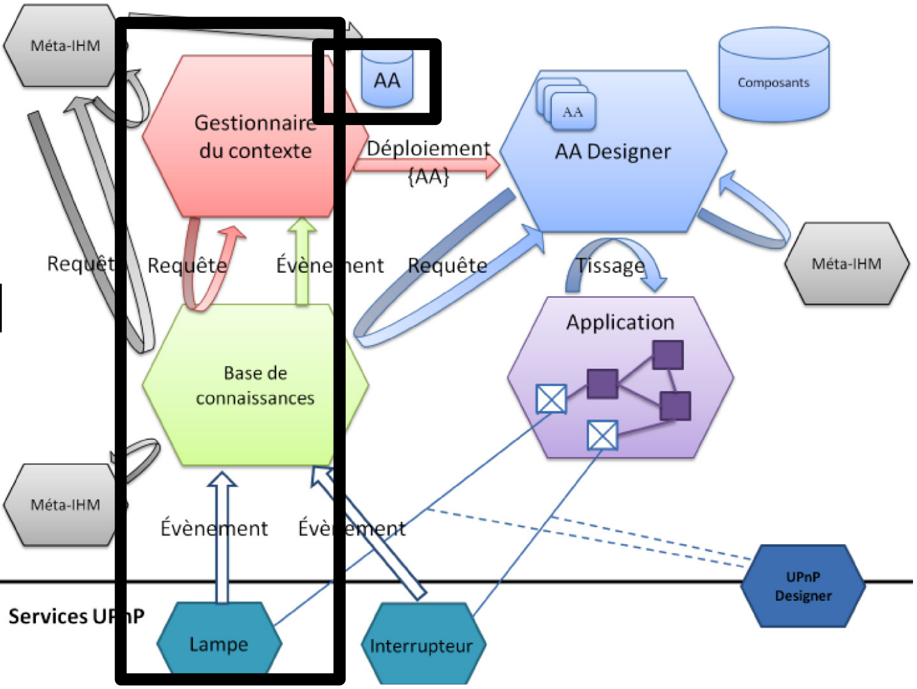


Baseline Middleware
WCOMP

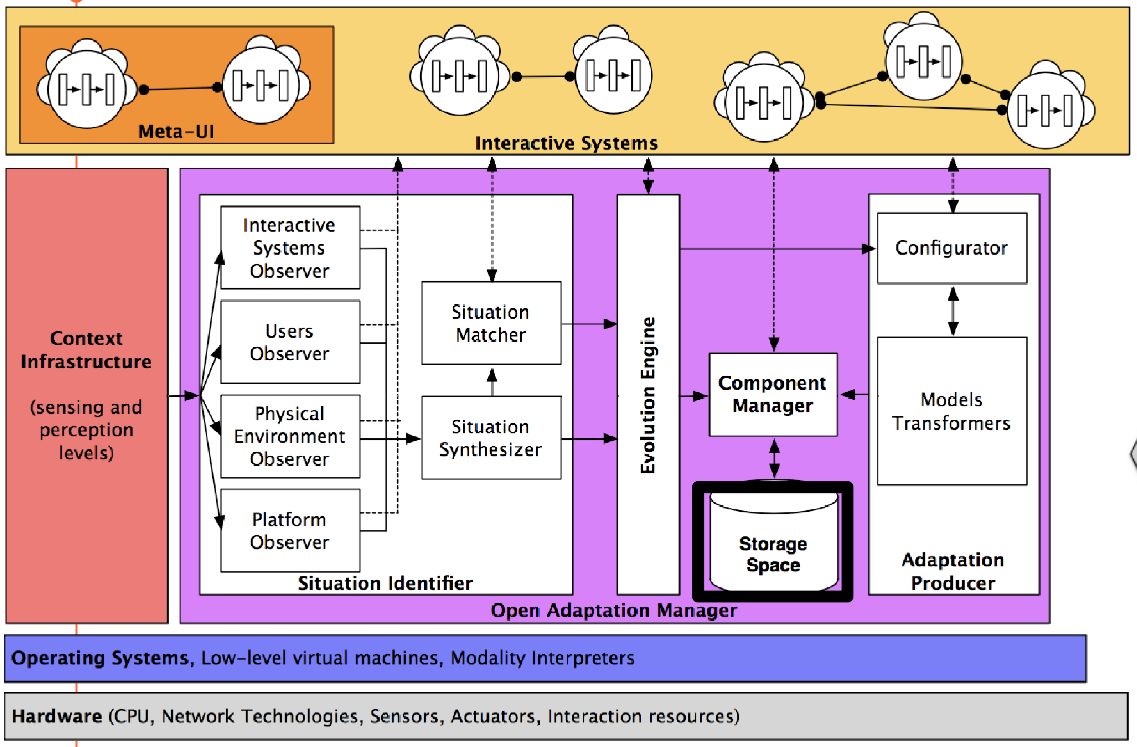


Generalization : from functional decomposition to implementation on top of a component-oriented middleware





Generalization : from functional decomposition to implementation on top of a component-oriented middleware



Meta-UI

Interactive Systems

Context Infrastructure

(sensing and perception levels)

Interactive Systems Observer

Users Observer

Physical Environment Observer

Platform Observer

Situation Matcher

Situation Synthesizer

Situation Identifier

Evolution Engine

Component Manager

Storage Space

Configurator

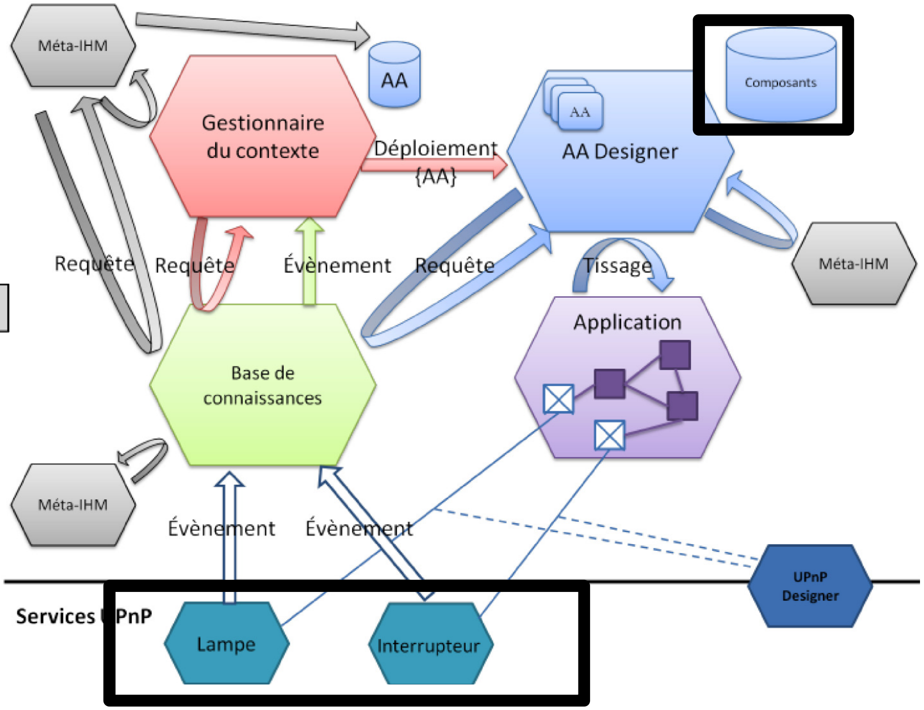
Models Transformers

Adaptation Producer

Open Adaptation Manager

Operating Systems, Low-level virtual machines, Modality Interpreters

Hardware (CPU, Network Technologies, Sensors, Actuators, Interaction resources)



Méta-IHM

AA

Gestionnaire du contexte

Déploiement {AA}

AA

AA Designer

Composants

Méta-IHM

Requête

Requête

Évènement

Requête

Tissage

Application

Base de connaissances

Méta-IHM

Évènement

Évènement

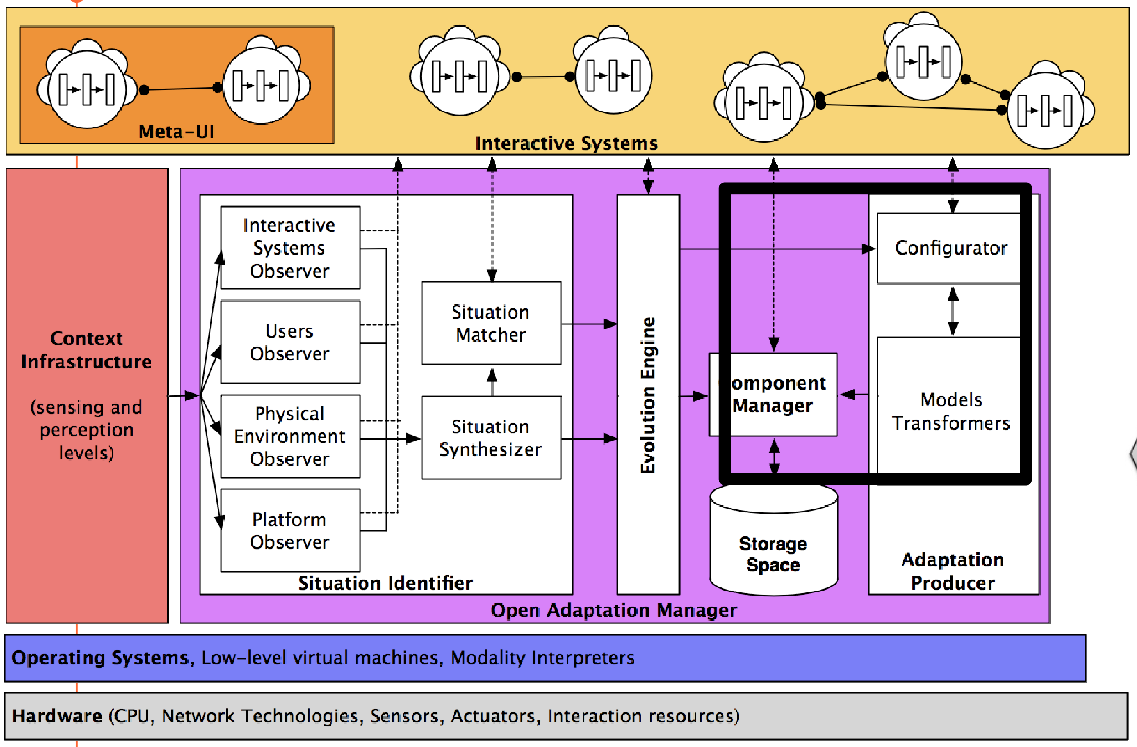
UPnP Designer

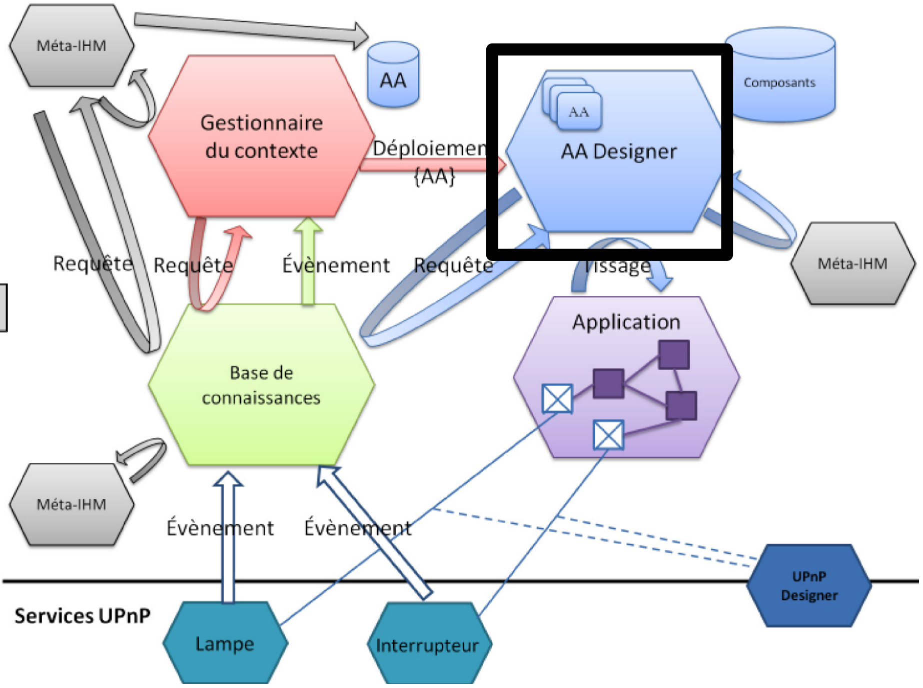
Services UPnP

Lampe

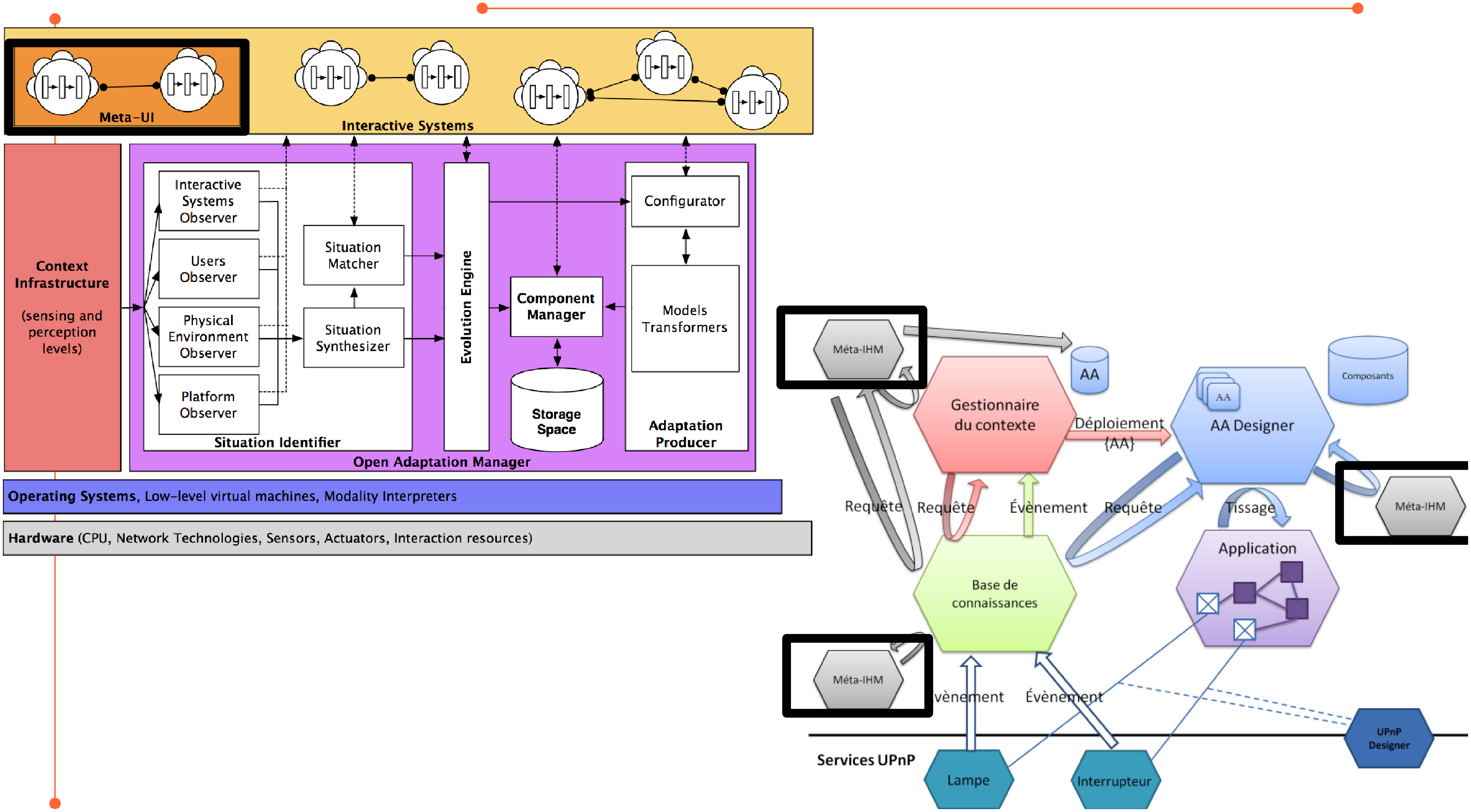
Interrupteur

Generalization : from functional decomposition to implementation on top of a component-oriented middleware





Generalization : from functional decomposition to implementation on top of a component-oriented middleware



Outline

- Contributions of our research to AmI infrastructures : Plastic user interfaces as requirements
- Lessons learned
- Perspectives

Lessons learned

- A “good” component-oriented middleware is key

- “Good” means support for incremental growth, heterogeneity, and dynamicity at run time (not pre-planned at design time)
  - Incremental integration/replacement of a large variety of protocols for sensors and actuators: ZigBee, EnOcean, Wateco
  - Dynamic discovery of heterogeneous devices and services
  - Service/Component dynamic deployment (life cycle management)
  - Notion of container for hierarchical composition and reuse (e.g., as in Fractal and WCOMP)
  - An ADL for expressing reconfiguration + interpretation at run time
  - Semantic interoperability

- On top of a good middleware
  - Knowledge base
  - Context manager
  - Simulator as a dual existence of the real world
  - Data capture
  - Meta-UI for every “system-oriented” component !

Perspectives

- End-User Development for the Home
- Baseline middleware: OSGi + Rose