Distributed Computing: From Principles to Pervasive Systems

Frédéric Le Mouël

Who am I ? – FLM

- Frédéric Le Mouël (frederic.le-mouel@insa-lyon.fr)
  - Associate Professor – INSA Lyon
  - Researcher – INRIA Rhône Alpes
  - http://perso.citi.insa-lyon.fr/flemouel/ @flemouel
- Research at CITI Laboratory / INRIA Amazones Team
  - Middleware, Component- and Service-Oriented Programming and Architectures, Pervasive Systems, Ambient Intelligence, Adaptation
- Teaching at Telecommunication Department
  - Modeling and Software Engineering, Middleware, Object-Oriented Programming, Java, Dynamic Web, Pervasive Systems

How to succeed in this lecture ?

- No mystery: Reading !
  - Two books
  - ~5-10 reference publications
- To acquire
  - Background
  - Critical sense

First part: Definitions and Concepts

Pervasive ≠ Peer-to-Peer ?

- Different application domains

- Common aspect:
  - Both are Highly Dynamic Systems
  - Devices, Data, Applications appearance/removal
  - Distance is different
    - Pervasive Systems ⇨ Proximity
    - Peer-to-Peer Systems ⇨ Large-scale

How to build such systems ?

- Common goal:
  - Building an application respecting user's needs and adapted and adapting to these highly changing environments

⇨Which are the base blocks on such development and runtime lifecycle ?

Lifecycle

Deployment
Runtime

Development

User's needs

Application

(examples: Information Systems, Location Systems, etc.)

Middleware

(examples: Models, Dev Toolkits, Architecture, Simulators, etc)

Hardware

(examples: Devices: PDA, mobile phones, laptops, etc Networks: 802.11*, etc)

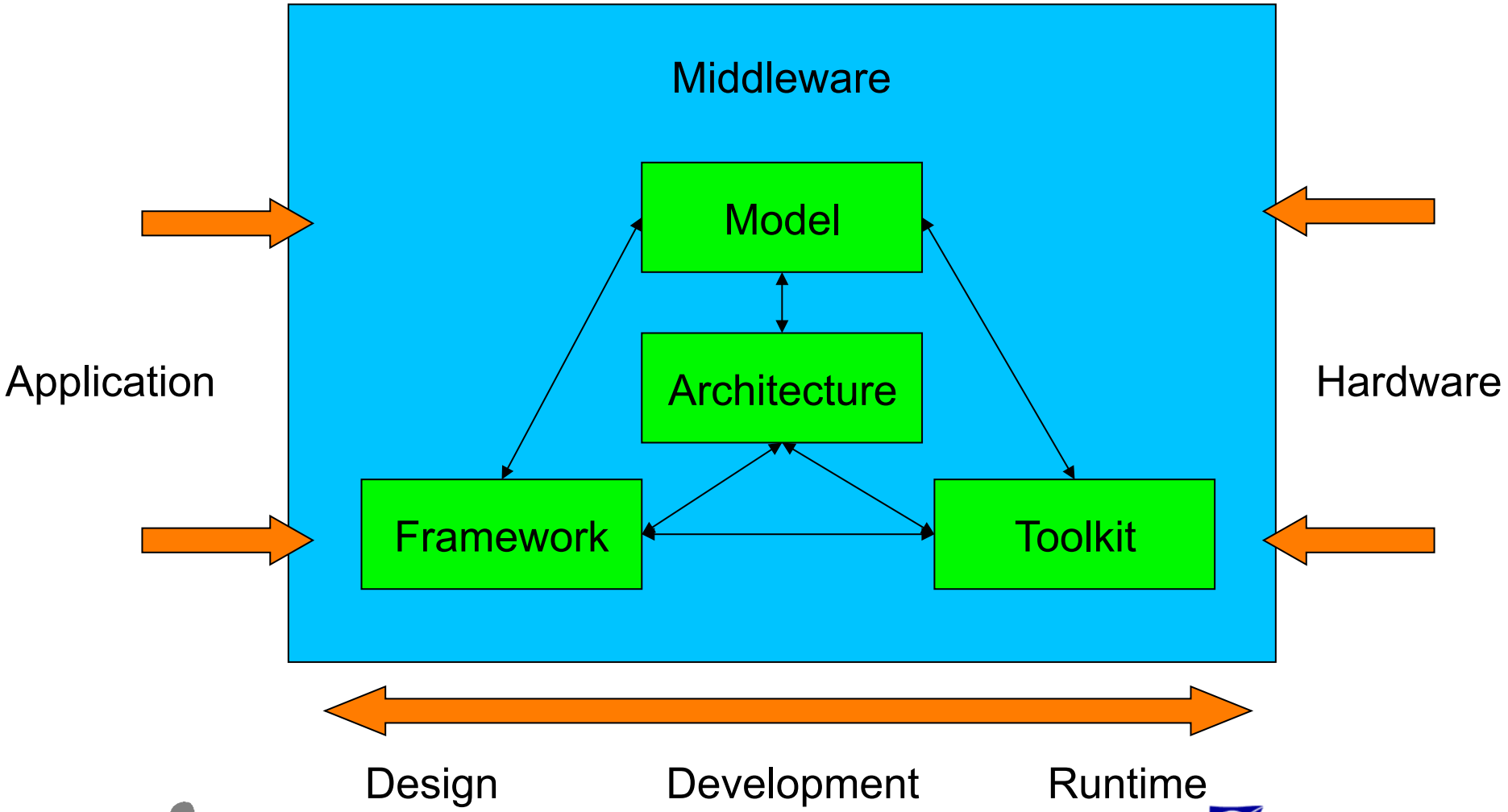
Pervasive or P2P Dynamism

Specific

Generic (reusable)

Specific

Middleware


Middleware
Model
Architecture
Framework
Toolkit
Application
Hardware
Design
Development
Runtime

Middleware

- Model, Architecture, Framework and Toolkit concepts
  - Transversal concepts (not only linked to middleware)

⇨Detailed in this lecture in terms of middleware for Distributed and Pervasive Systems

Definition: Framework

- "A software framework is a reusable design for a software system (or subsystem). This is expressed as a set of abstract classes and the way their instances collaborate for a specific type of software." [Johnson 1988]
- "A framework is a set of cooperating classes that make up a reusable design for a specific class of software." [Gamma 1995]

- Classes ⇨ System base blocks
- Specific class of software ⇨ Skeleton of application family

Definition: Toolkit

- "A toolkit is a set of related and reusable classes designed to provide useful, general-purpose functionality. An example of a toolkit is a set of collection classes for lists, associate tables, stacks and the like." [Gamma 1995]

- Lists, etc ⇨ Implementation of base blocks

Definition: Model

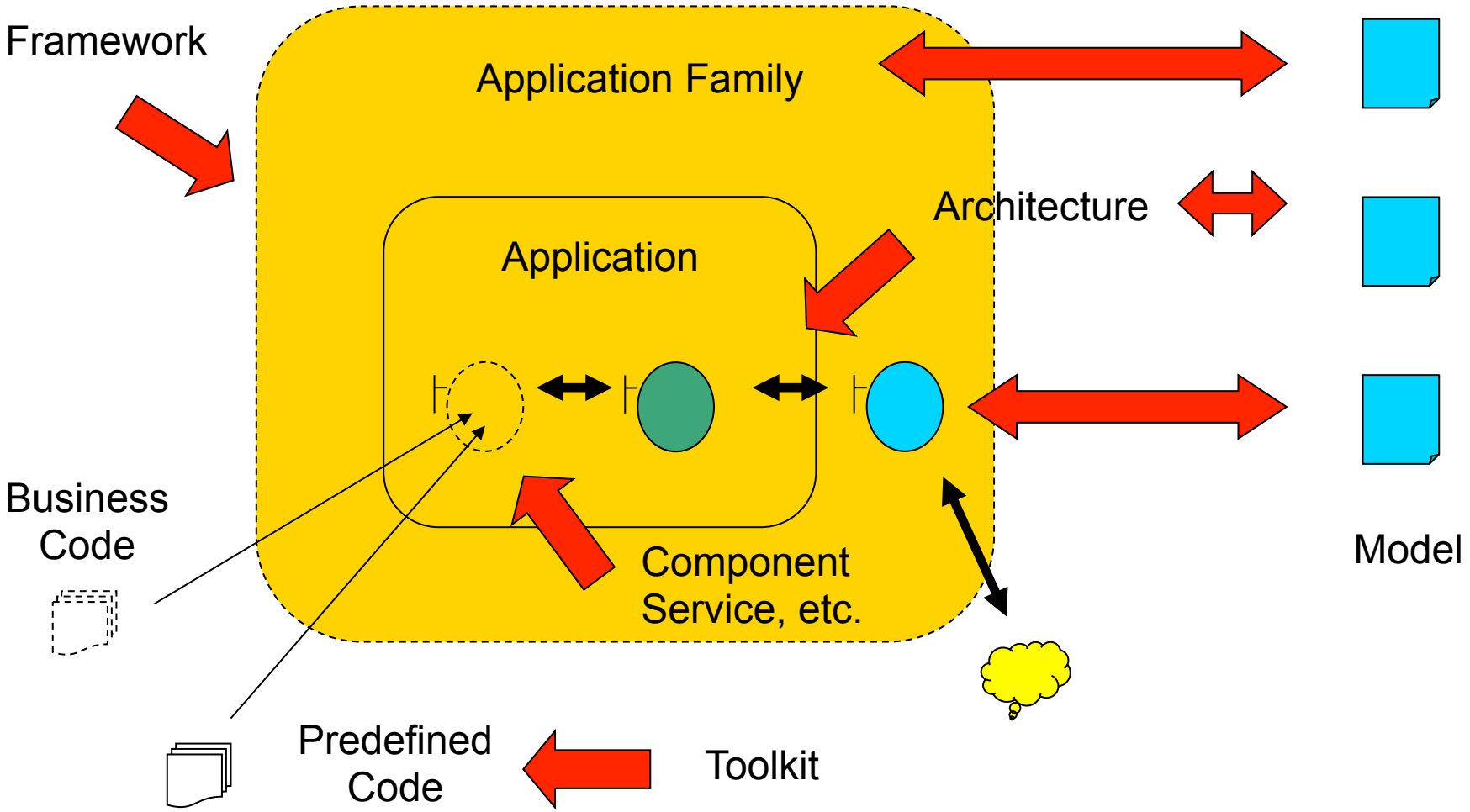
- “Metamodeling is the construction of a collection of "concepts" (things, terms, etc.) within a certain domain. A model is an abstraction of phenomena in the real world, and a metamodel is yet another abstraction, highlighting properties of the model itself.” [Wikipedia, OMG 2001, Schmidt 2006]

- Abstraction ⇨ Characteristics of the system
- Phenomena ⇨ Behavior of the system
- Properties ⇨ Deterministic behavior, guarantees ?

Definition: Architecture

- "The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships between them." [Len 2003]

Summary


Framework
Application Family
Architecture
Application
Business
Code
Component
Service, etc.
Model
Predefined
Code
Toolkit

Next step

- Frameworks, toolkits, architectures, models for:
  - Pervasive systems
  - Peer-to-peer systems

Bibliography

- Johnson, R. E. and B. Foote (1988). “Designing reusable classes”. Journal of object-oriented programming 1(2): 22-35
- Pree, W. (1994). “Meta patterns - a means for capturing the essentials of reusable object-oriented design”. in M. Tokoro and R. Pareschi (eds), Springer-Verlag, proceedings of the ECOOP, Bologna, Italy: 150-162
- Gamma, E.; Helm, R.; Johnson, R. and Vlissides, J. (1995). “Design Patterns”. Addison Wesley Professional Computing Series.
- OMG (2001). “Model Driven Architecture (MDA) document”. Architecture Board ORMSC.
- Len, B.;Clements, P.; Kazman, R. (2003). “Software Architecture In Practice, Second Edition”. Boston: Addison-Wesley, p. 21-24.
- Schmidt, D.C. (2006). “Model-Driven Engineering”. IEEE Computer 39 (2).

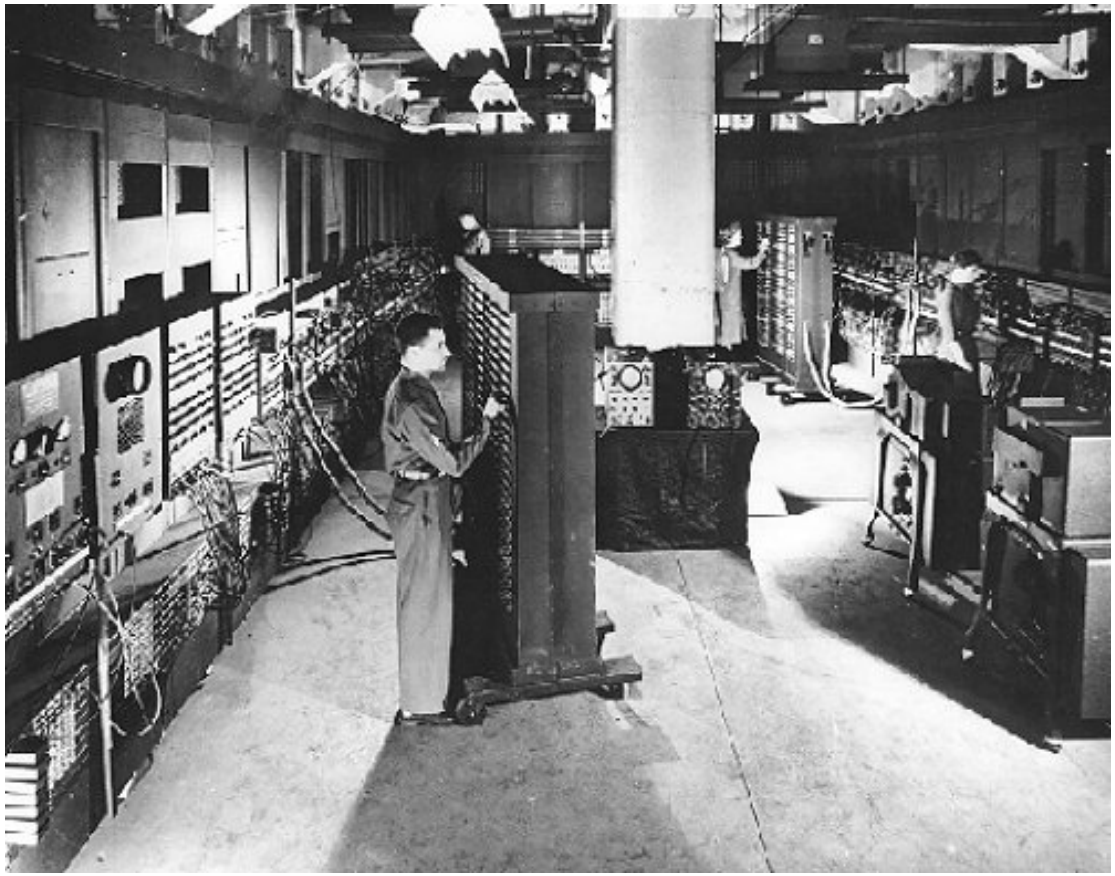
Pervasive Systems

Introduction

New vision for Information Technologies

- Working Environment
  - Before: a virtual environment where you log in, execute applications and then log out
  - Now: a physical environment where you are always connected to execute tasks

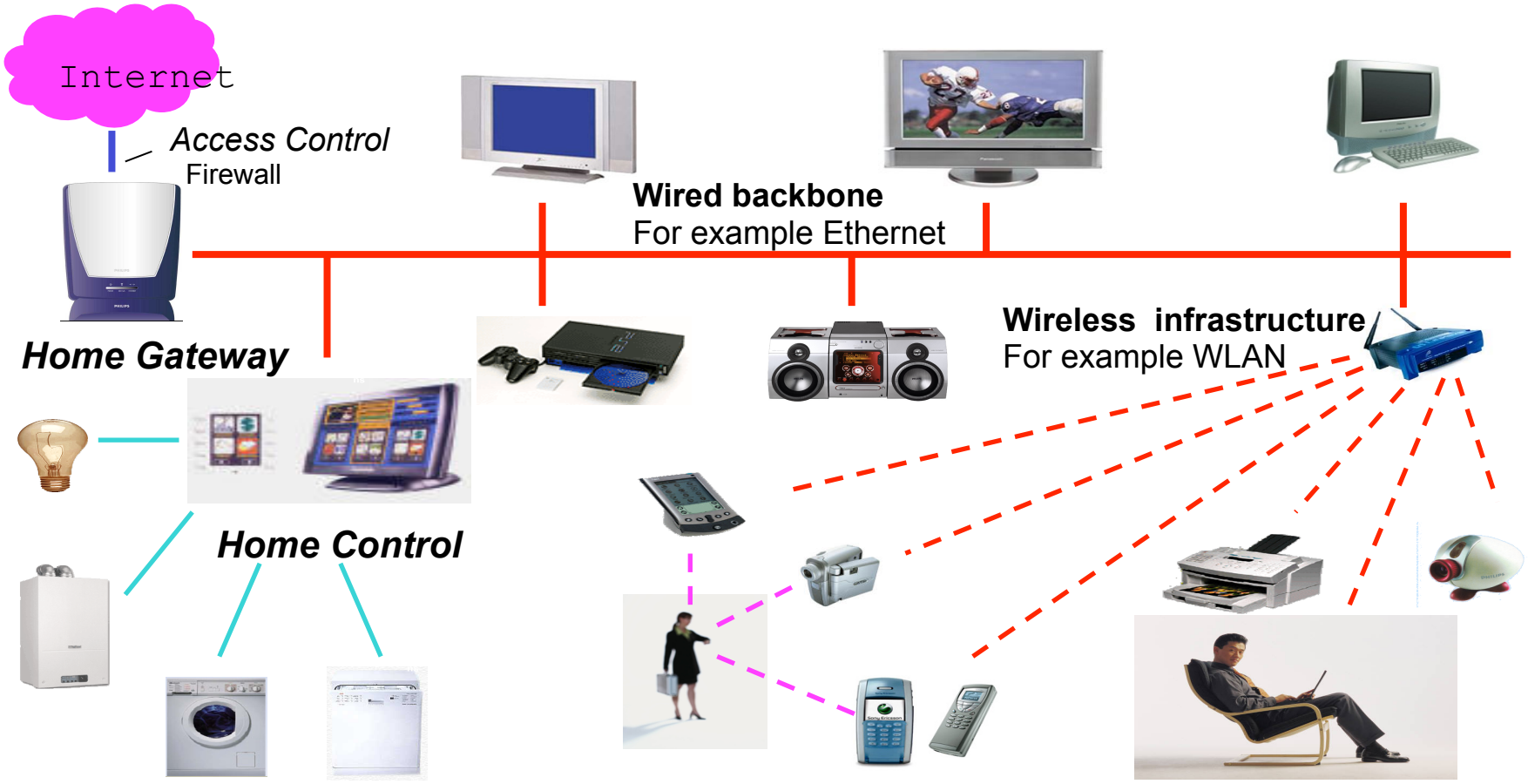
Before:
Room full with a computer



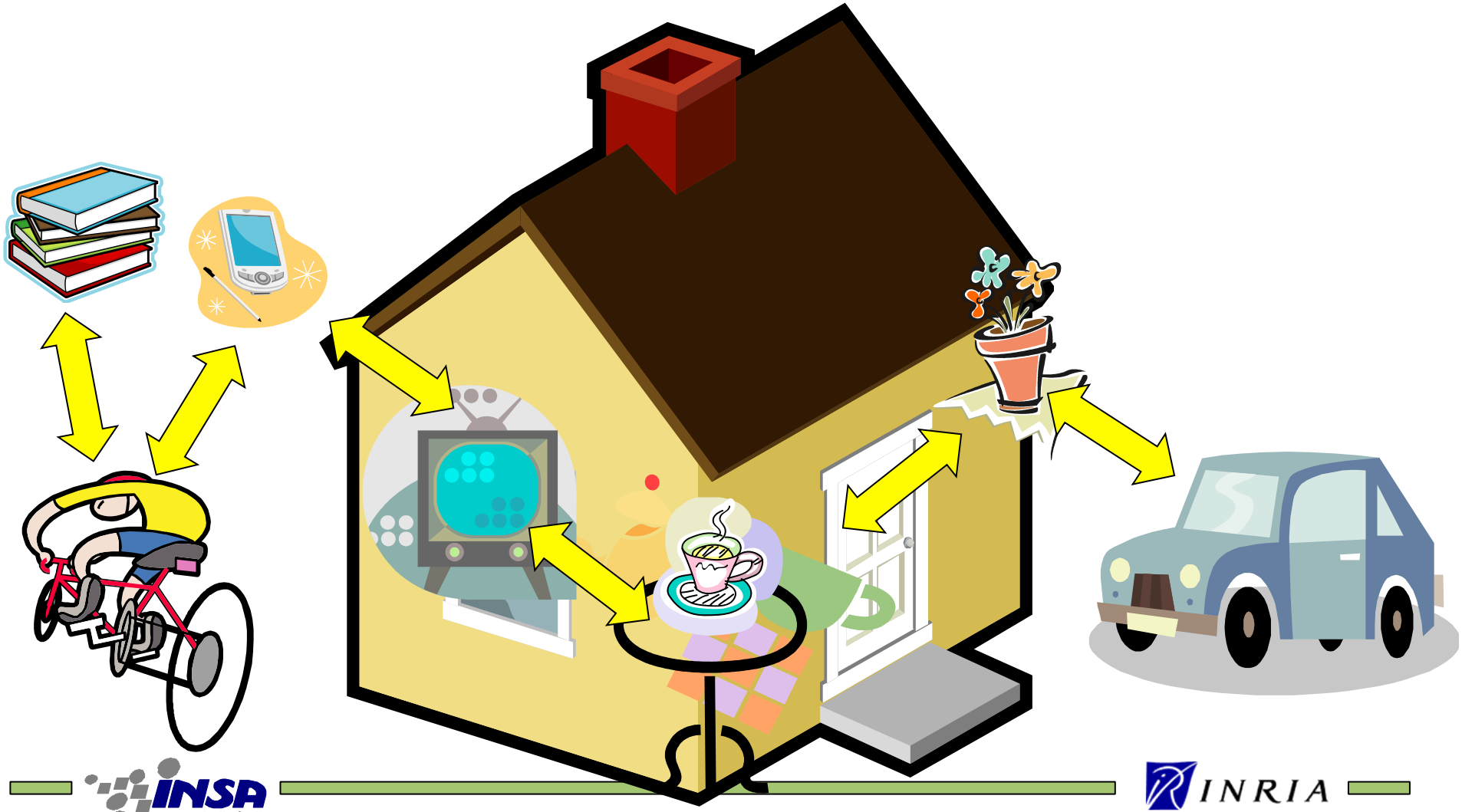
Electronic
Numerical
Integrator
and Computer
(ENIAC)
1946

© Computer Science History

Now: Room with several visible devices



Tomorrow: Everyday life full of invisible appliances



CITI — INSA LYON — INRIA

Motivation

- Today
  - Computers
  - Internet wired connection

- Tomorrow
  - Every object will be smart (Embedded processors + memory)
  - Wireless connection (802.11*, Bluetooth, etc. + Internet New Generation, IPv6)

Pervasive Environments

- [M. Weiser, 1991]
  - « *A new way of thinking about computers in the world, one that takes into account the natural human environment and allows the computers themselves to vanish in the background* »
  - « *The most profound technologies are those that disappear . They weave themselves into the fabric of everyday life until they are indistinguishable from it* »

Pervasive Environments

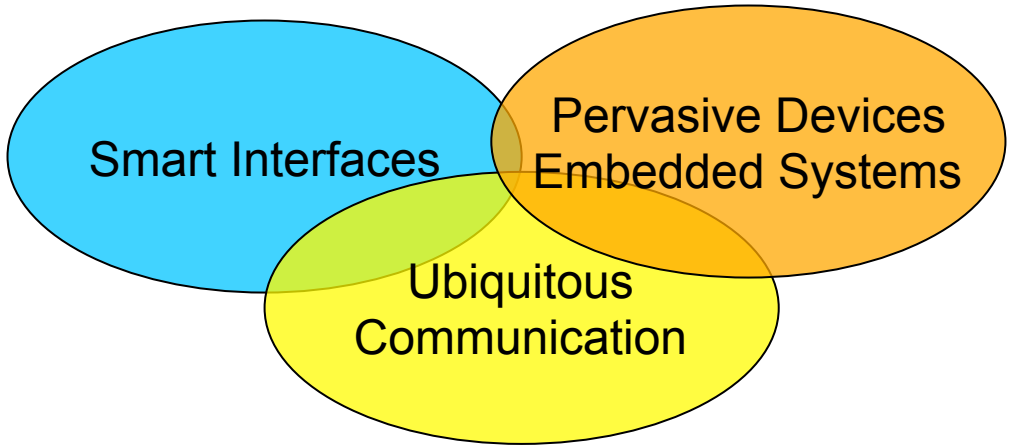
- [M. Satyanarayanan, 2001]
  - « *One saturated with computing and communication capability, yet so gracefully integrated with users that it becomes 'a technology that disappears'* »

Pervasive Environments

- [NIST, 2001]
  - « *Pervasive computing is a term for the strongly emerging trend toward: numerous, casually accessible, often invisible computing devices, frequently mobile or embedded in the environment, connected to an increasingly ubiquitous network infrastructure, composed of a wired core and wireless edges* »

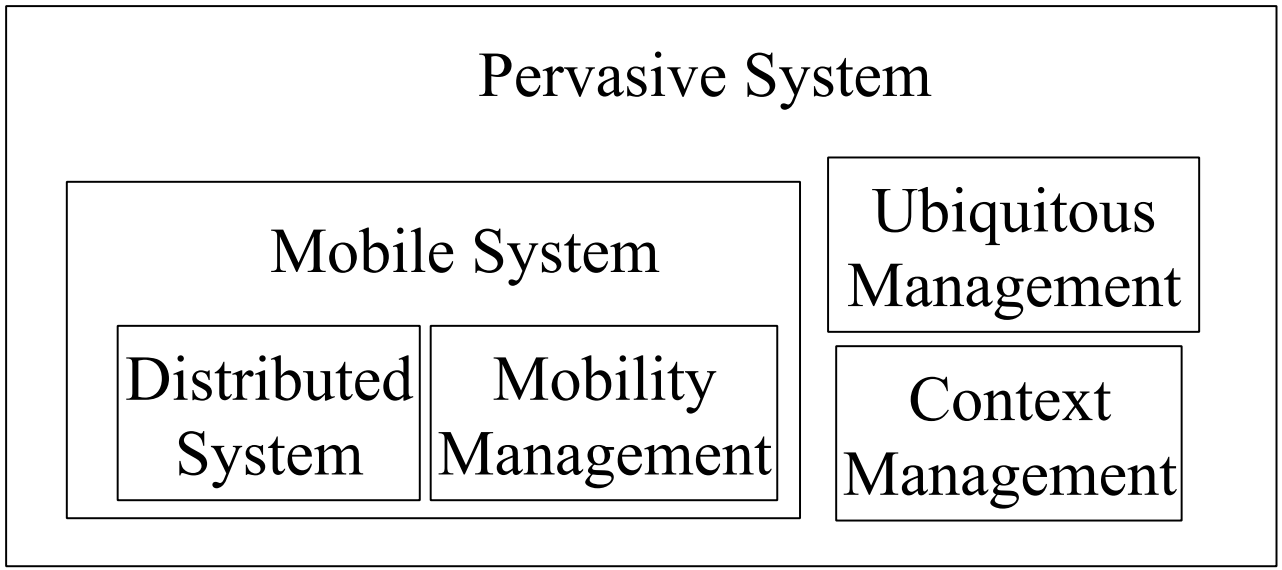
Pervasive Sub-topics

- Interconnection of 3 technological domains:
  - Smart Interfaces
  - Pervasive Devices, Embedded Systems
  - Ubiquitous Communication, Connectivity



System view of a Pervasive System

- Adapted from [Saha & Mukherjee, 2003]



Definitions

- Ubiquitous
  - Accessible from everywhere
- Mobile
  - Which integrates mobile devices
- Context-awareness
  - Which takes into account the execution environment
- Pervasive
  - Which associates ubiquitous, mobility and context-awareness

Pervasive System Properties

- Scalability
- Invisibility
- Context-awareness
- Smartness
- Pro-action

Scalability

- Management of a great amount of
  - Devices
  - Applications
  - Users
- Performance

Example: Web Server scalable ?

- Development of systems, middlewares, models, applications that are independent and can resist to a high number of devices, users, etc.

Invisibility

- Transparency for human beings
- Minimal intervention of human beings

- Adaptation to environment changes
- Self-learning

- Example: auto-configuration of gateway

Context-awareness

- Virtual representation of the physical environment
- Perception of changes of the environment

- Environment model
- Environment monitoring

- Examples :
  - User Profile, Application Meta-data, Self-descriptive Devices
  - Temperature, Location Sensors

Smartness

- Smart = showing mental alertness and calculation and resourcefulness [wordreference.com, Merriam-Webster, dictionary.com]
- “Intelligent” use of perceived changes
  - Reaction and/or anticipation model (rules, etc.)
  - Inference motor
- Example: Smart House - Power reduction by switch on/off the lights

Pro-action

- Ability to interact, “disturb” the user in order to suggest a better action

! To balance with invisibility !

- Context, environment evaluation
- Several contexts (past, current, future)
- Disturbance model to evaluate the cost/gain between Invisibility/Pro-action

- Example: Information filtering/classification -> Spam

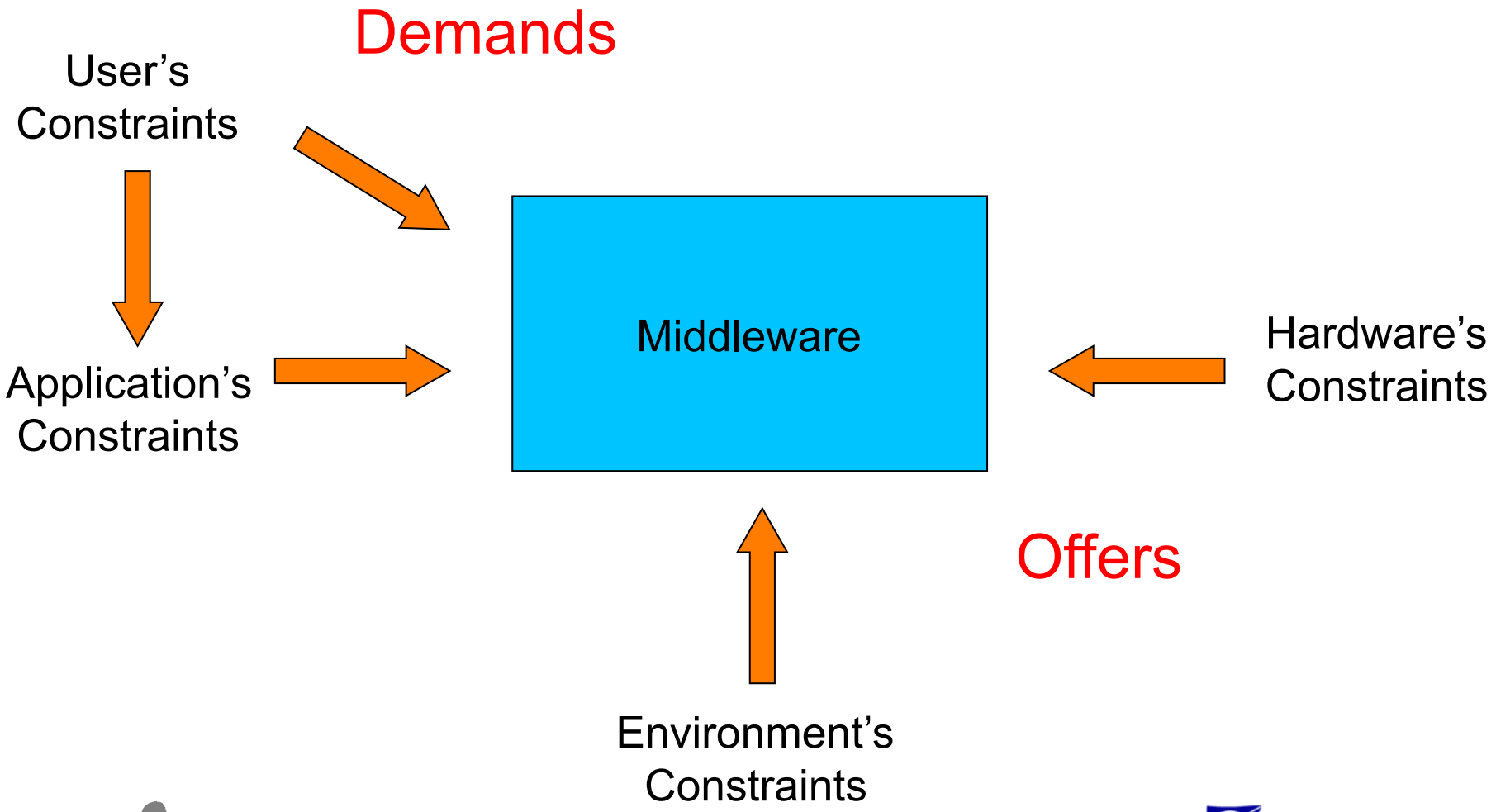
Bibliographie

- M. Weiser « The computer for the twenty-first century », Scientific American, sept 1991:94-104
- M. Satyanarayanan « Pervasive computing : Visions and challenges », IEEE Personal Communications, aug. 2001:10-17
- National Institute of Standards and Technology « Pervasive Computing Program », Pervasive Computing 2001
- D. Saha & A. Mukherjee « Pervasive computing : a paradigm for the 21st century », IEEE Computer journal, march 2003:25-31
- F. Mattern. « Ubiquitous & Pervasive Computing: A Technology-driven Motivation », Summer school on ubiquitous and pervasive computing, 2002
- F. Laforest « Cours Systèmes d'Information Pervasifs », Master Mastria, INSA de Lyon, 2007

Pervasive Systems

Middleware

Middleware


Demands
User's Constraints
Application's Constraints
Middleware
Hardware's Constraints
Offers
Environment's Constraints

Hardware's constraints

- Autonomy ⇦ Battery
- Limited resources (CPU, memory, screen, etc.)
  - Power ⇔/ Battery, energy dissipation
  - Capacity ⇦ Weight, size
- Low impact robustness ⇦ Weight, size
- Low security confidence ⇦ Easy access, lost

Environment's constraints

- Mobility / Nomadism
  - Transmission signal
    - Connections / disconnections
    - Variable signal strength

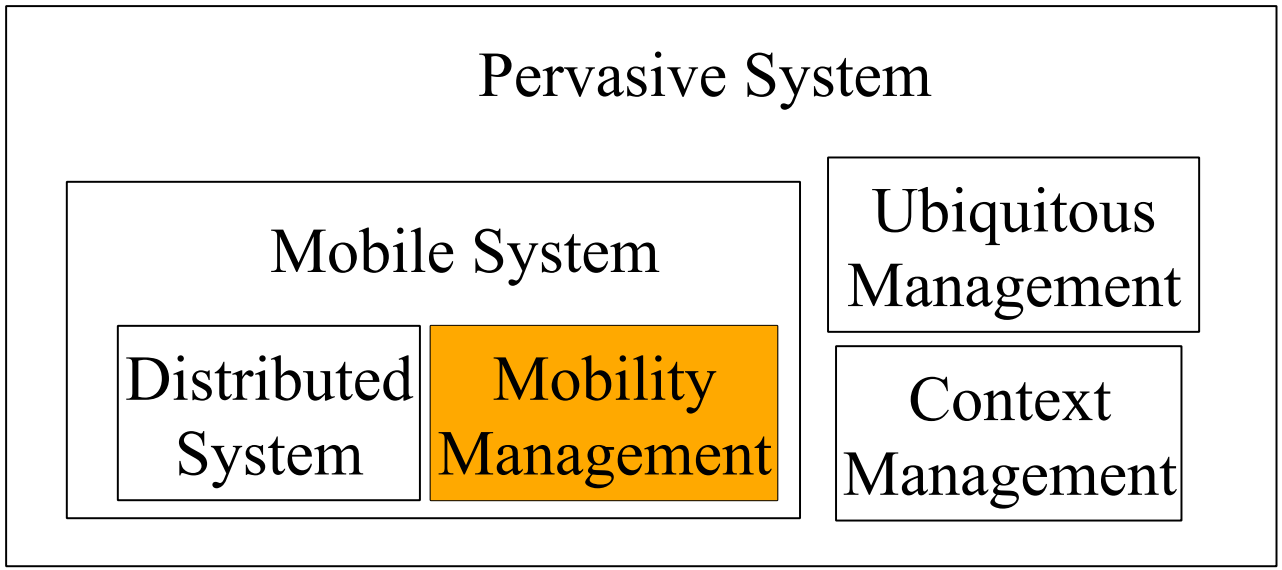
  ⇨ Interferences, cells scope
  - Services availability
    - Different quality of service

  ⇨ Devices appearance / removal

User and Application's constraints

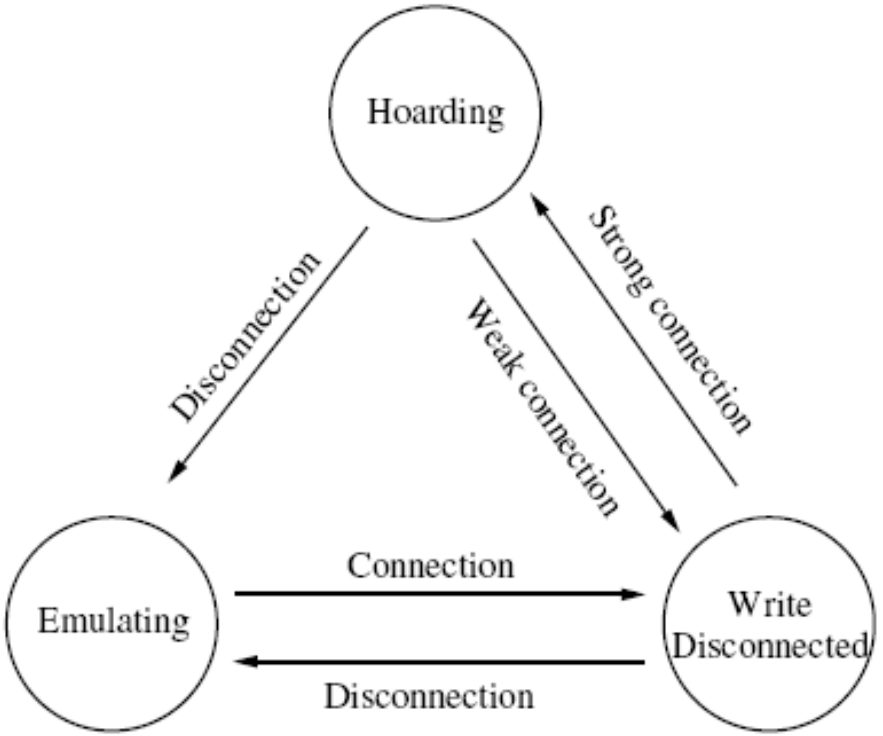
- User's constraints
  - User Profile
  - QoS required
- Application's constraints
  - Power demand
  - Storage demand

System view of a Pervasive System

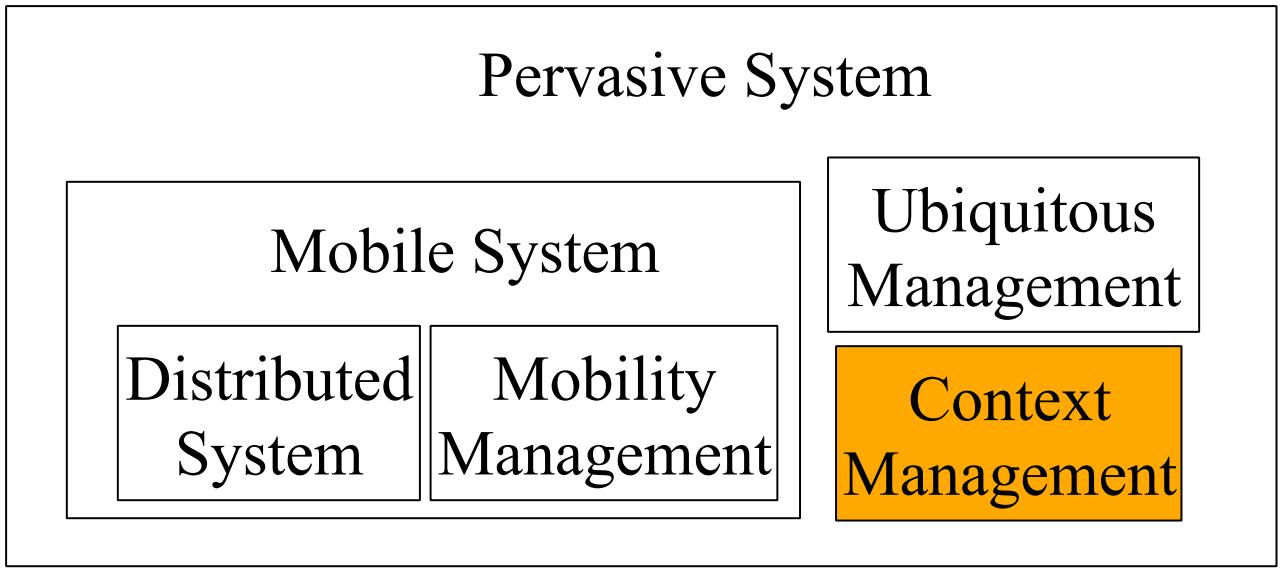


Coda

- File system
  - Hoarding mode (connected)
    - Loading
    - Prefetching
  - Emulating (disconnected)
    - Local work
    - Logs
  - Write disconnected (weak connection)
    - Loading
    - Reconciliation



System view of a Pervasive System



Definition

- [Salber,Dey,Abowd 99]
  - « Environmental information or context covers information that is part of an application's operating environment and that can be sensed by the application. This typically includes the location, identity, activity and state of people, groups and objects. »

Context data

- 4 axes
  - User
    - Profile, preferences, location, etc.
  - Application
    - Size, format, encoding, langue, versions, etc.
  - Hardware
    - Screen size, resolution, color depth, memory, etc.
  - Network
    - Bandwidth, signal strength, etc.

Context modeling

- 3 approaches
  - Attribute/Value
  - CC/PP Extension
  - Ontology

Attribute/Value Pairs

- Context = pairs (attribute, value)
  - User= Toto
  - Localisation = CITI

- Pairs are independent

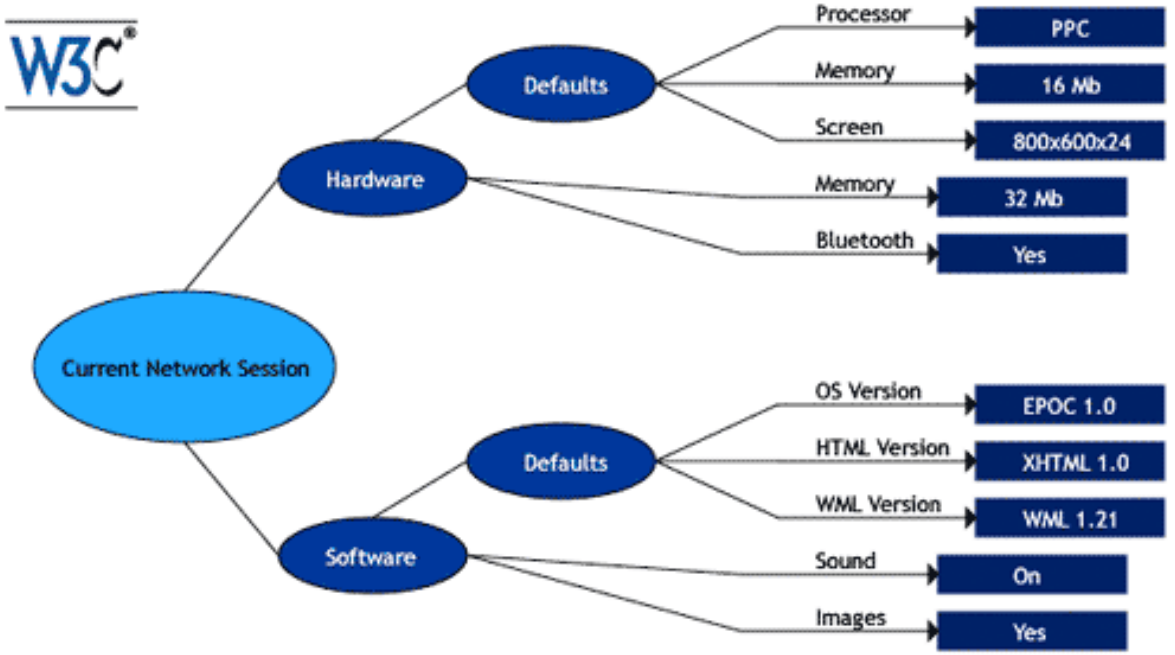
+ Easy

- Consistency

- Poor semantic expressiveness

CC/PP Extensions

- Composite Capabilities / Preferences Profile (W3C)
  - Hardware and User
  - RDF file
  - Context = Extensions proposal



+ Standard

- Extensions =>

  Complex, hard to read

CC/PP example: XML file

```
<?xml version="1.0"?>
<!-- Checked by SiRPAC 1.16, 18-Jan-2001 -->
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ccpp="http://www.w3.org/2000/07/04-ccpp#">
  <rdf:Description rdf:about="HWDefault">
    <rdf:type rdf:resource="HardwarePlatform" />
    <display>320x200</display>
    <memory>16Mb</memory>
  </rdf:Description>
</rdf:RDF>
```

Ontology

- Model of
  - Class
  - Classes relationships
  - Instances

+ Semantically expressive

+ Large-scale environments

- Complex, ontology matching

Ontology example

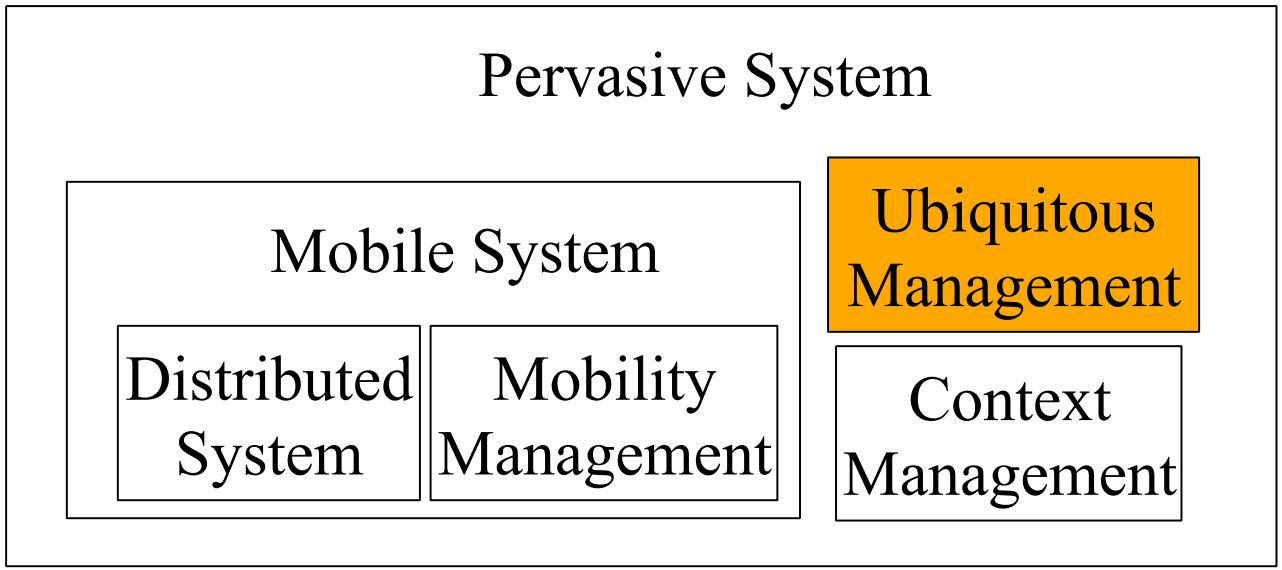
- Exemple CoOL [Strang & al. 03]

```
<instance xmlns=http://demo.heywow.com/schema/cool
xmlns:a=http://demo.heywow.com/schema/aspects
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<contextInformation>
<entity system="urn:phonenumber">+49-179-1234567</entity>
<characterizedBy>
<aspect name="GaussKruegerCoordinate">
<observedState xsi:type="a:o2GaussKruegerType">367032533074</
observedState>
<units>10m</units>
</aspect>
<certaintyOfObserver>90</certaintyOfObserver>
</characterizedBy>
</contextInformation>
</instance>
```

Synthesis

	Expressiveness and powerful	Easiness	Conflict Management
Attribute / Value	-	+	-
CC/PP	+	+	-
Ontology	+	-	+

System view of a Pervasive System



Ubiquitous Management

- Ways of accessing everything everywhere

- 2 steps
  - Discovery
  - Communication/Dissemination
    - Data results
    - Software itself

Service Discovery Protocols

- Service search
  - Where are the services ?
  - Where to store this knowledge ?

Service Discovery Protocols

- 2 implementation approaches
  - Service registry: centralized
  - Flooding: distributed
- Different search criteria
  - Location
  - Semantic
  - Mobility

Service Registry

- Organized set of available services
- Set provided by a dedicated host

- Examples:
  - SLP
  - Jini
  - Salutation

SLP registry: centralized data

- Service Location Protocol: Agent based
- Service Agent: the provided service
- Directory Agent
  - Registers the SAs in a LDAP registry
  - Multicast
- User Agent: the requester of the service
  - Multicast request

Jini registry: centralized soft

- Sun Jini: Java based
- Service provider
  - Identity and group broadcast
  - Renew registering
- Jini registry:
  - Stores RMI interface, proxy to service provider
  - Leasing mechanism, limited lifetime storing
- Service requester
  - Lookup request, receives proxy and location
  - Direct RMI proxy use

Salutation registry: neighbours decentralization

- Each host:
  - Stored a subset of available services
- Service provider:
  - Registers in local registry
  - And in neighbours registries
- Service requester:
  - Lookup in local registry
  - Then broadcasts to neighbours registries

UPnP: flooding discovery

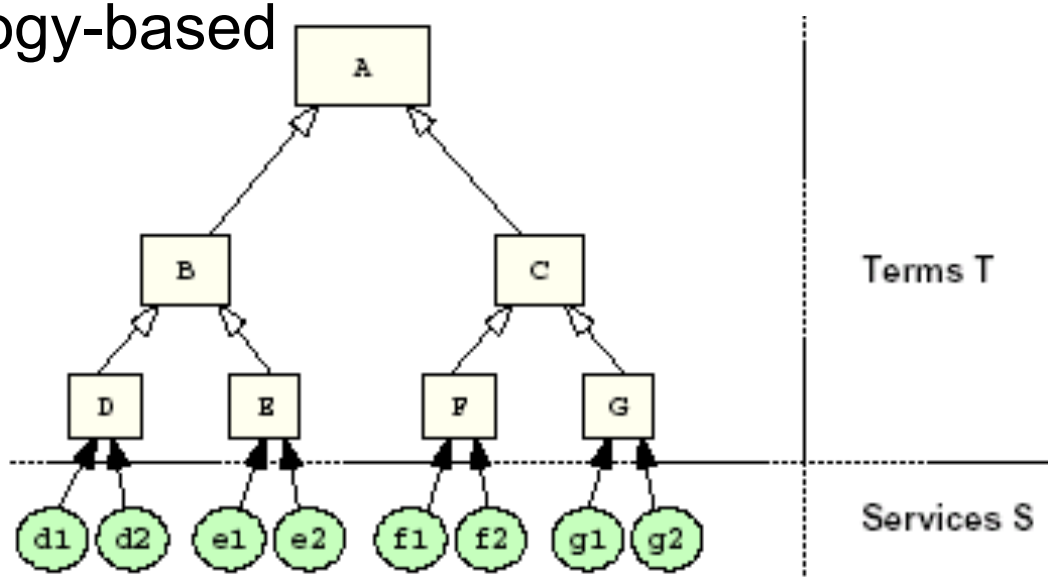
- Universal Plug and Play: Industrial consortium
- Each host
  - Available service list
  - Zero conf (DHCP, autoIP, multicast DNS)
  - Communication (point-to-point, streaming)
  - Automatic discovery
    - Multicast: XML messages, Arrival ANNOUNCE, Services available OPTIONS

Bluetooth registry: geographic service location

- Each host: SDP server
  - Stored local available services (service record: services attributes, class with unique UUID)
- Service provider:
  - Registers in local registry
- Service requester:
  - UUID lookup
  - Broadcast lookup for navigating into neighbours registries

Multi-layers clusters: geographic and semantic discovery

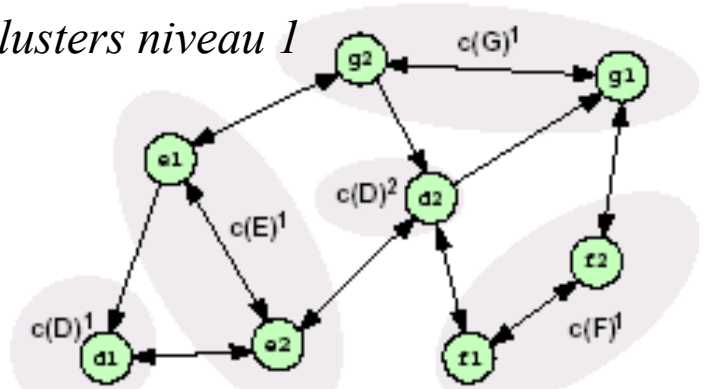
- Hosts grouping according to proximity
  - Geographic: direct routing (single hop)
  - Semantic: providing similar services
    - Ontology-based



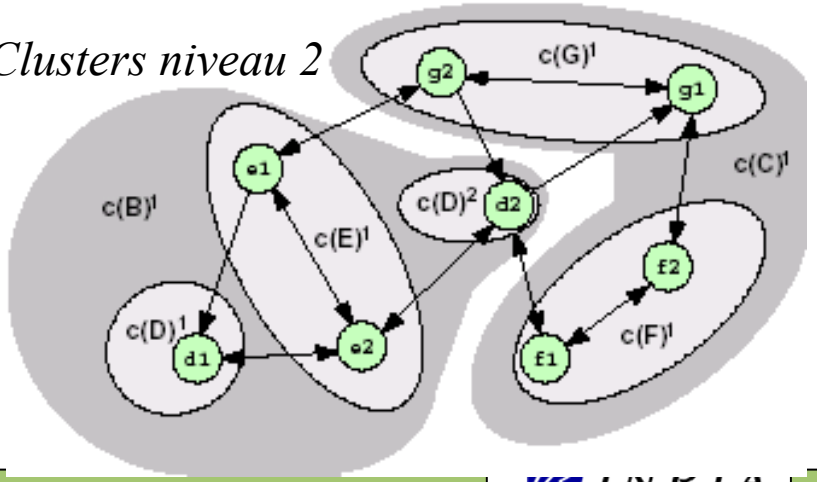
Multi-layers clusters: geographic and semantic discovery

- Service request:
  - Lookup in the closest cluster
    - Level 1: direct access or same service
    - Level 2: 2 hops or same category service
    - Level 3, etc.

Clusters niveau 1



Clusters niveau 2



Mobile Services Lookup

- During use, services can move
  - User mobility
  - Load balancing
- 3 accessing approaches
  - Location server
  - Poste restante
  - Repeaters

Mobile service discovery: location server

- Location server:
  - Stores pairs (service/location)
  - Each service warns the server of its location changes
- Service requester:
  - Asks the location server for the service location
  - Directly accesses to the service

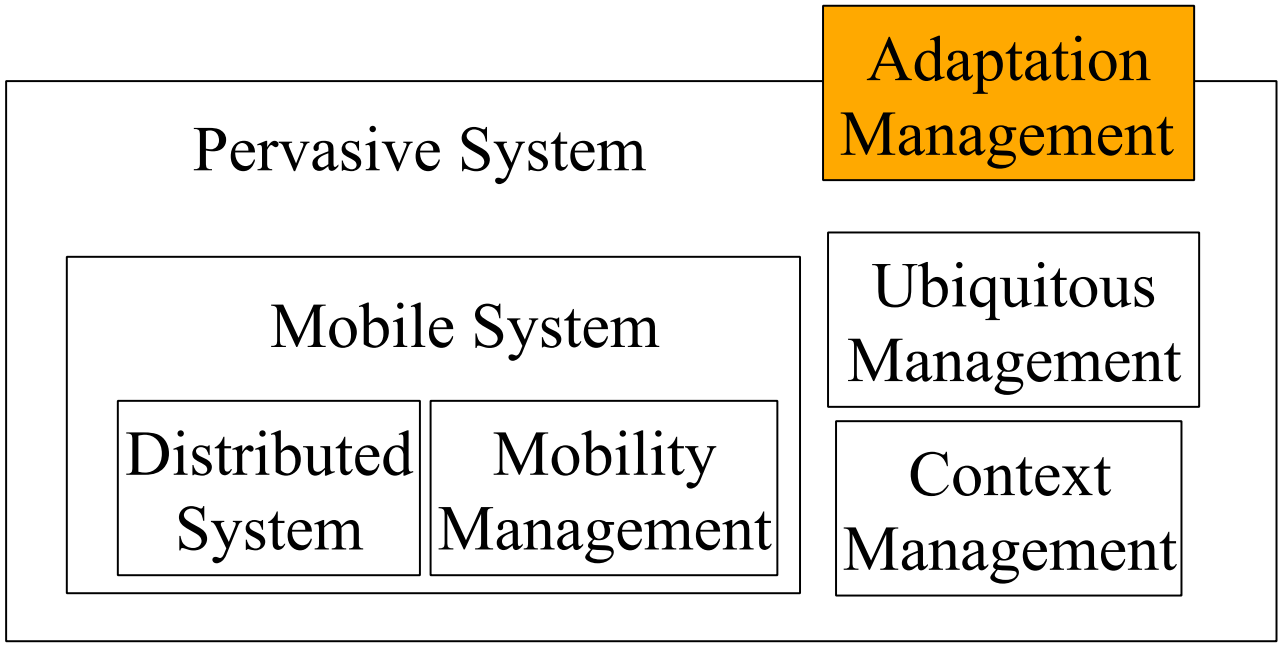
Mobile service discovery: Poste restante

- Static proxy:
  - As the same service interface
  - Used as Poste restante for the service
  - The Service periodically reads and answers its messages
- Service requester:
  - Asynchronous communication
  - Send a message to the service
  - Proxy interception

Mobile service discovery: Repeaters

- Service provider:
  - Generates a repeater on each host it is leaving
  - A repeater knows the next location of the service
  - A repeater forwards the messages to the next service location
- Service requester:
  - Sends its request to a service at its last known location
- Several service moving: Repeaters chain

System view of a Pervasive System



Adaptation Management

- Dynamic adaptation
- Adaptation can focus:
  - The user interface (close to application)
  - The data
  - The Services

Data adaptation

- Data adaptation: modification of data to respect the display rules of a target terminal
- Adaptation location:
  - On the client (not for light-client)
  - On the server (heavy-server)
  - On a active proxy network (load balancing, consistency to implement)

Adaptation according to data type

- Text source
  - Format conversion (html -> txt, doc -> pdf…)
  - Summary
  - Traduction
  - Compressing/uncompressing
  - Vocal synthesis
- Image source
  - Format conversion (jpeg -> png)
  - Modifications of resolution, colours number, depth…
  - Compressing/uncompressing (e.g. semantic jpeg or raw zip)

Adaptation according to data type

- Audio source
  - Format conversion
  - Textual synthesis or vocal recognition
  - Compressing/uncompressing (e.g. semantic MP3 or raw zip)
- Video source
  - Format conversion (resolution, nb images/sec)
  - Spatial Decomposition/recomposition (zoom…)
  - Compressing/uncompressing (e.g. semantic MPEG4 or brute zip)

Content adaptation operators

- Coding (Wav->MP3)
- Format (HTML ->WML)
- Modality replacement (image by descriptive text)
- Selection (size selection of images)
- Integration (multi-servers data)

Documents adaptation by WebServices

- [Berhe, Brunie 2004]
- WebServices-based adaptation architecture
  - Local Proxies
  - Content Proxies
  - Content Servers
  - Adaptation Service Proxies
  - Adaptation Services Repository
  - Profile Manager

Documents adaptation by WebServices

- 4 profile types:
  - Document
    - Physical meta-data (type, size, format…)
    - Storage meta-data (versions, repartition…)
    - Semantic meta-data (keywords…)
  - Client : user and terminal CC/PP
    - User : language, interests…
    - Terminal : hardware (screen size, memory...) and software (available, versions,…)
  - Network
    - Latency, bandwidth …
  - Service
    - WSDL : adaptation type, media type, performance, cost...

Documents adaptation by WebServices

- Local proxy
  - Receives user requests
  - Calculates the client profile
  - Sends the request to content proxy
  - Compares the answer profile with the client profile
  - Deduces an adaptation plan and applies it
  - Integrates adapted received data
  - Collaborates with other local proxies for cache management

Documents adaptation by WebServices

- Adaptation plan
  - Determines the adaptation constraints: attributes conditions
  - Determines the adaptation operators needed
  - Selects an optimal adaptation strategy
  - Looks for adaptation services that can realize the needed adaptation operators
  - Negotiates with the adaptation services (costs, performances)
  - Uses selected adaptation services

Documents adaptation by WebServices

- Notion of adaptation path
  - Sequence of adaptation operators that verifies the constraints
  - The path can be balanced by costs, performances…
- Notion of adaptation graph
  - When several adaptations can be applied in parallel on a subset of data (eg. image et meta-data on images – DICOM format)

Bibliographic references - Content adaptation

- E. Mory et al. Adaptation de contenu multimédia aux terminaux mobiles. RTSI - ISI n° spécial systèmes d ’ information pervasifs n°9, 2004, Hermès: 39-60
- G. Berhe, L. Brunie, LIRIS Adaptation de contenus multimédia pour les systèmes d ’ information pervasifs RTSI - ISI n° spécial systèmes d ’ information pervasifs n°9, 2004, Hermès: 39-60

Services adaptation

- Services adaptation have generally 3 parts [Cremene 04]:
  - Modifiable part: the adaptable service
  - Monitoring part: continuous evaluation of the service and its context
  - Control part: definition of reconfiguration orders, according to the service logic

Adaptation in reflexive systems

- Reflexivity
  - Ability of a system to represent itself, to monitor itself and to act on itself
  - Meta level that describe the components of a system
- Introspection
  - System property allowing to know its internal state. Allows to reason and take decision about itself
- Intercession
  - System property allowing to change its behaviour by modifying its own functionality

Adaptation in reflexive systems

- Adaptation targets
  - Entity (methods, objects, components, services…)
  - Link between entity (links between base entities and/or between base and meta entities)
  - Set of entities
- Adaptation moment
  - Compilation : code generation according to meta-entity
  - Loading : alteration of compiled code or modification of dependencies in a set of entities
  - Execution : dynamic access to the meta level, by using proxy, or by the execution platform

RAM

- [Bouraqadi et al 01] Reflexion for Adaptable Mobility
- Code mobility = non functional aspect, so at meta level
- Cluster = unity of mobile code with:
  - A set of applicative objects
  - A meta interface for policies of the cluster (migration, …)
  - A table of instantiated bindings (references to other clusters)
- Strong code mobility

DynamicTAO: reflexive middleware

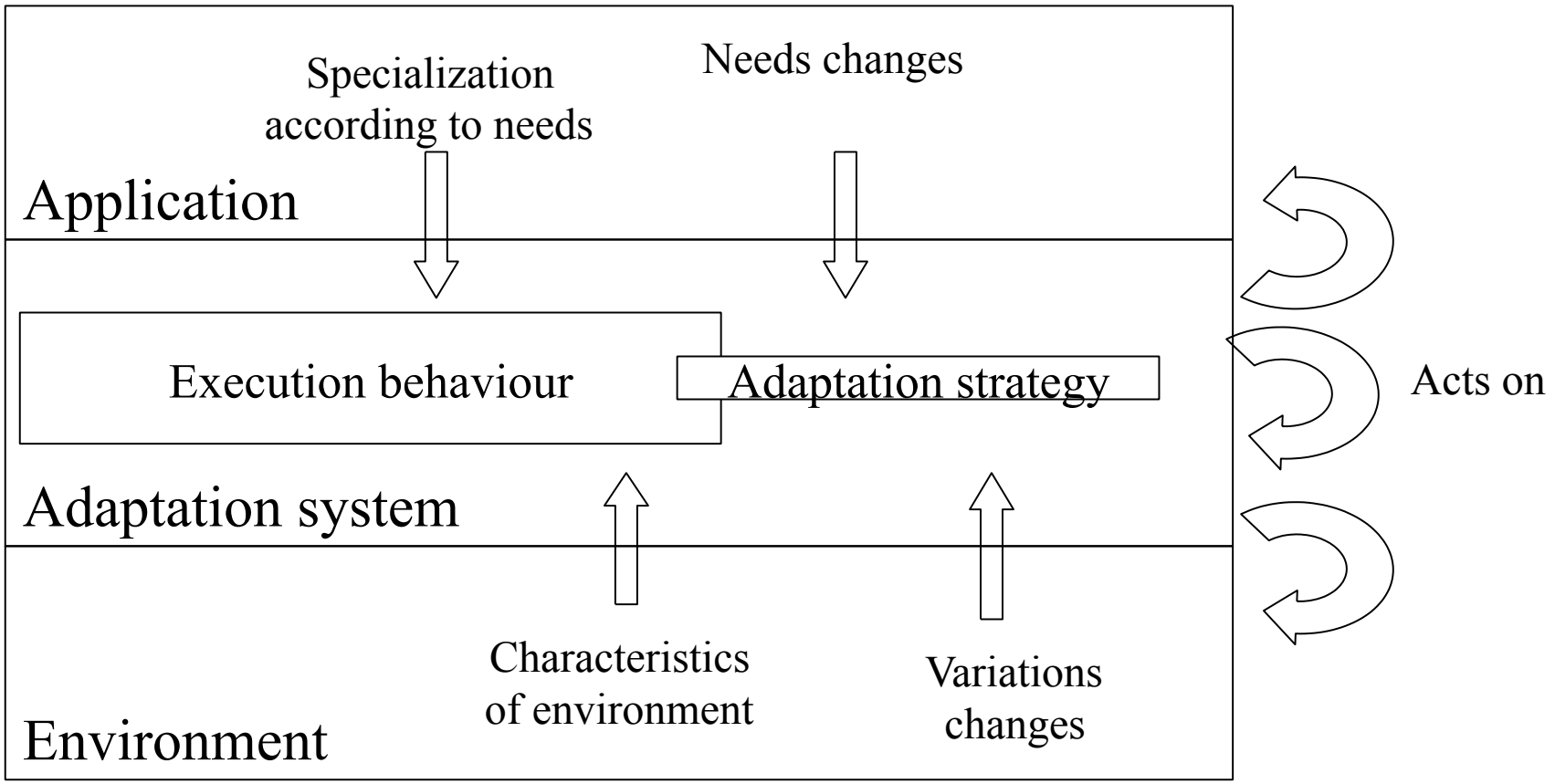
- [Kon et al. 2000] ORB reflexive based on CORBA
- Set of Component Configurators
  - A TAO Configurator maintains middleware strategies (concurrency, scheduling…)
    - Component calls interception -> strategies
    - Dynamic component load of implementations (even for strategies)
- Component dynamic reconfiguration

Dream : Dynamic REflective Asynchronous Middleware

- [Leclerc et al. 2005] « component-based framework for constructing, statically or dynamically, resource-aware, configurable message-oriented middleware (MOM) ».
- Fractal extension (Java based)
  - Dynamic assembly of components
    - Primitive components
    - Composite components
  - For each component, management interfaces:
    - BindingControler : components dependencies management
    - ContentControler: adding and removing components
    - LifeCycleControler : run, stop components

AeDEN – Software entity adaptation

- [Le Mouel 2003]

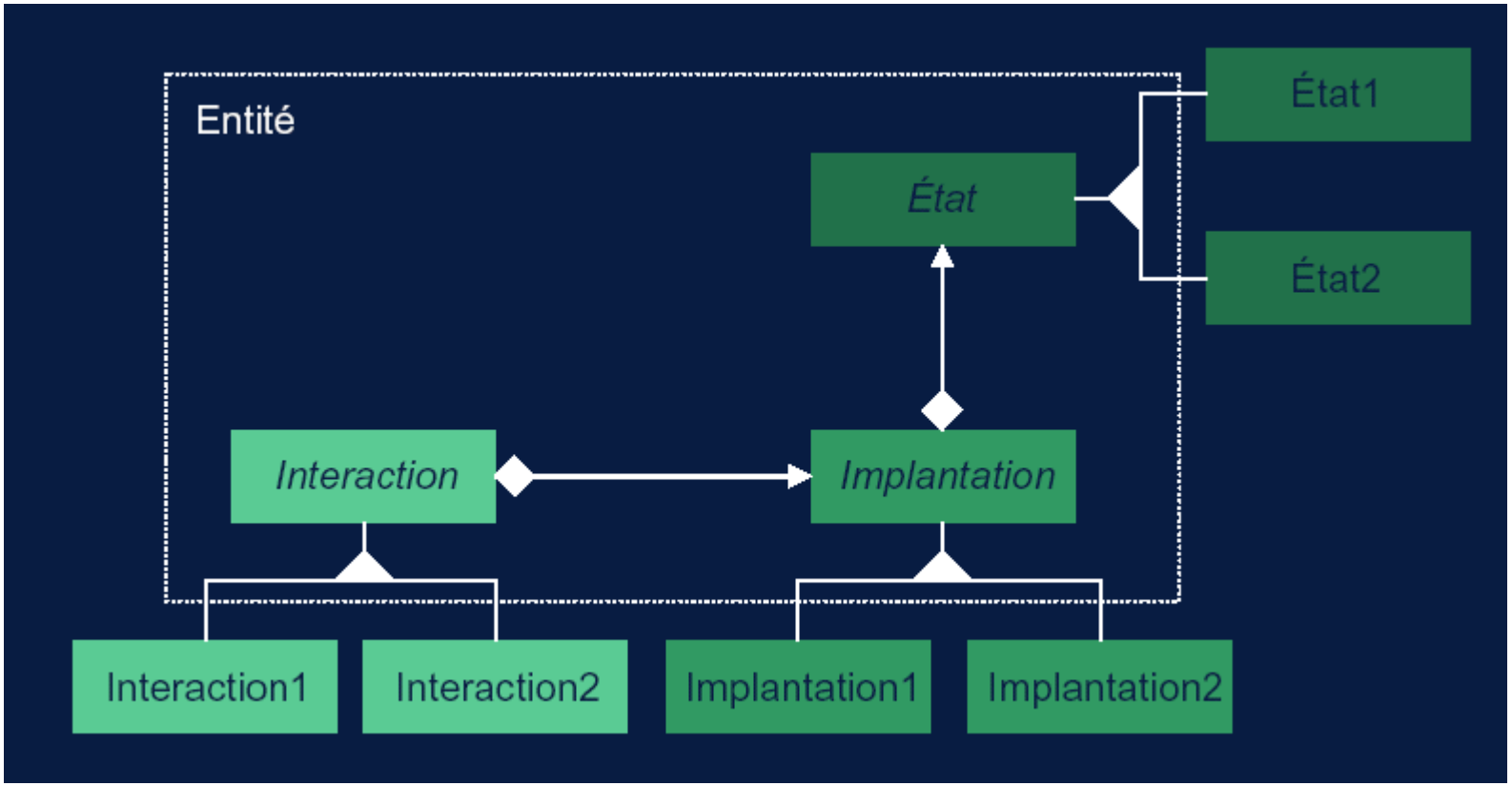


AeDEN - entity

- Entity = software conception unity
  - Abstract and specializable
  - A functionality <=> A entity
  - A service <=> A specialized entity
- 3 aspects:
  - AInteraction (communication with other entities)
  - AImplementation (business : expected treatments)
  - AState (internal state of the business part)
- Different implementation available for each aspect

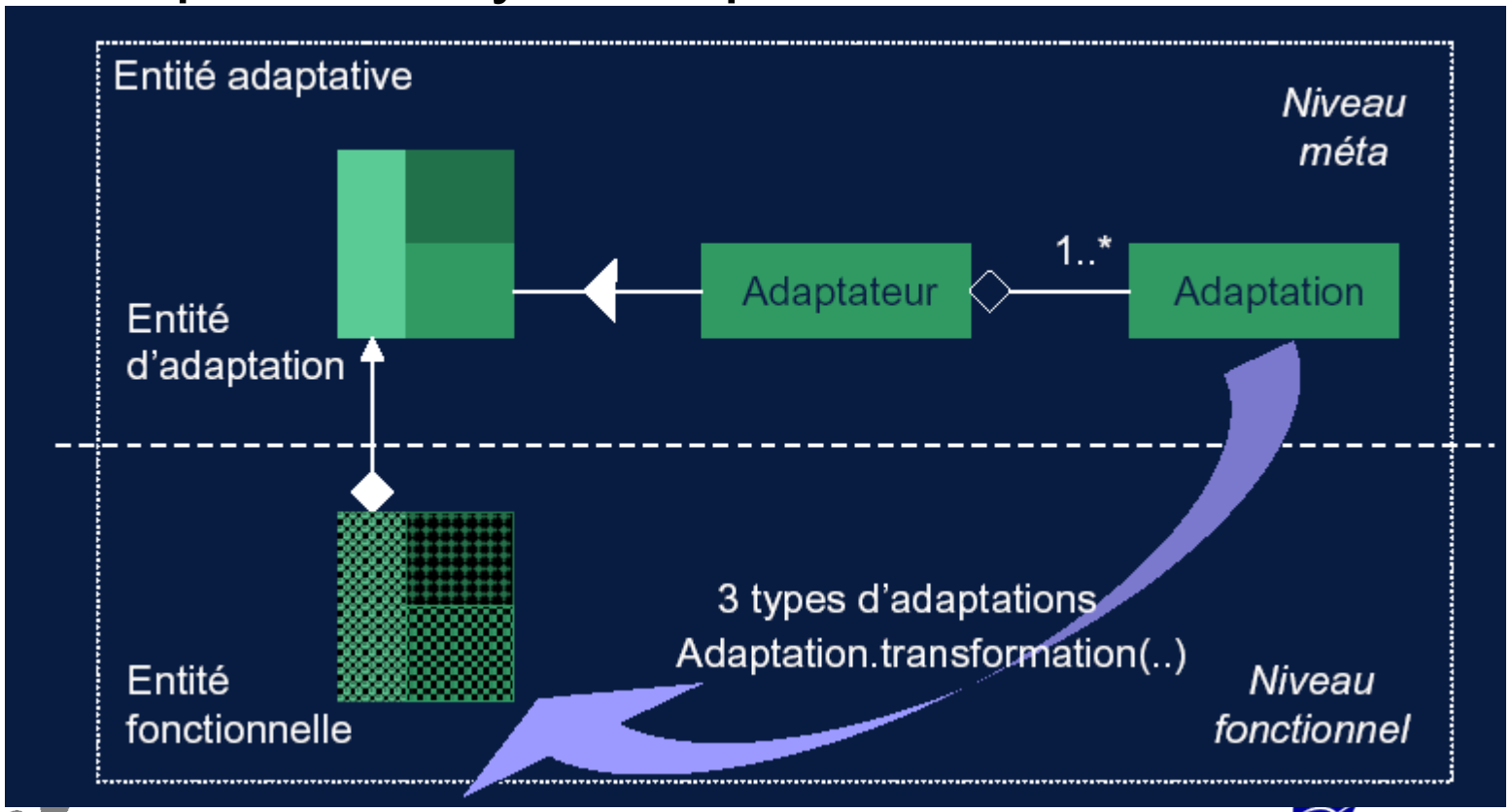
AeDEN - entity

- Abstract entity + possible specializations



AeDEN – Adaptive entity

- Entity + Adaptation entity
  - Adaptations by introspection and intercession

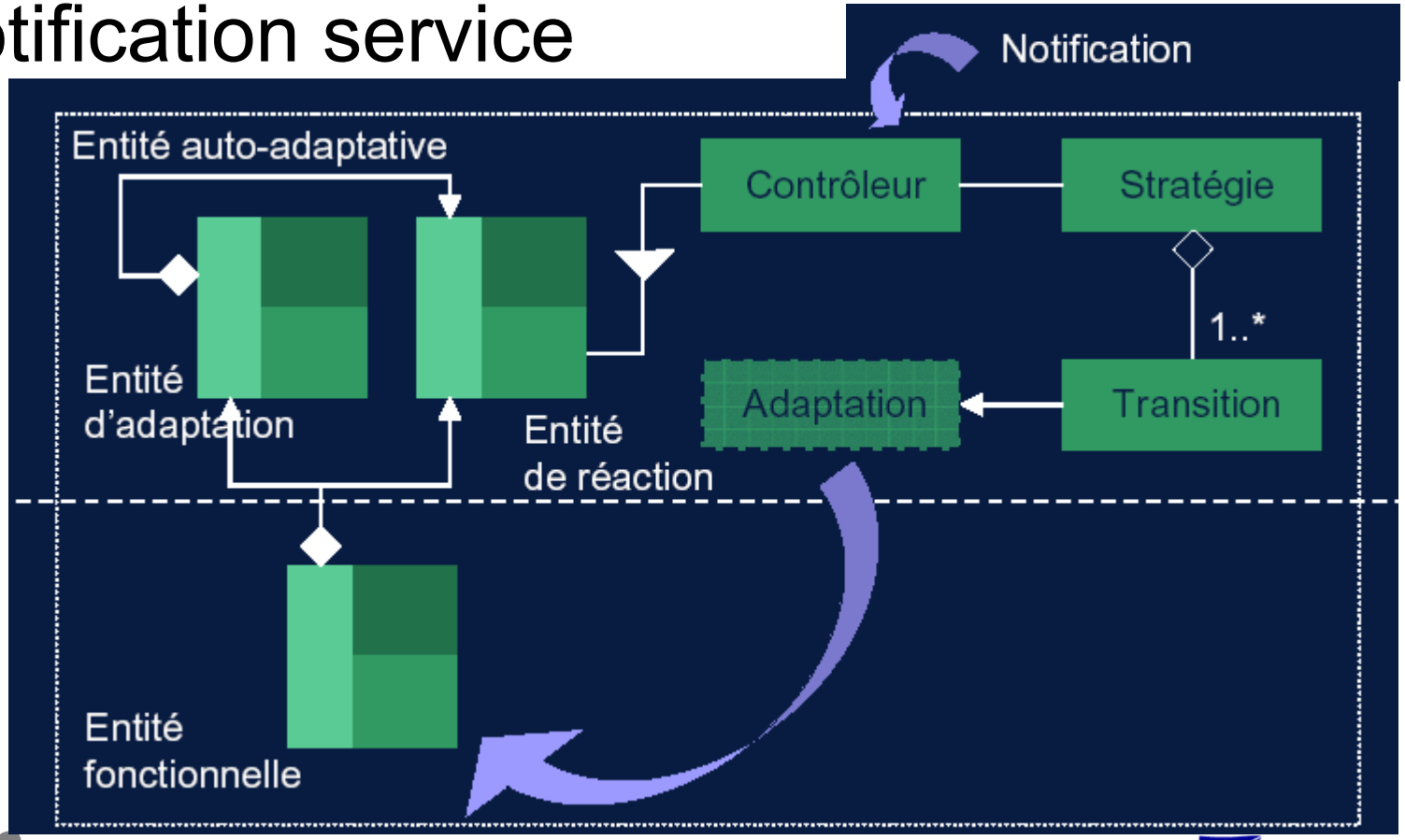


AeDEN - introspection and intercession

- Each entity has the following methods:
  - getInteraction(), setInteraction()
  - getImplementation(), setImplementation()
  - getState(), setState()
- Each adaptive entity has the following methods:
  - getFunctionalInteraction(), setFunctionalInteraction()
  - getFunctionalImplementation(), setFunctionalImplementation()
  - getFunctionalState(), setFunctionalState()

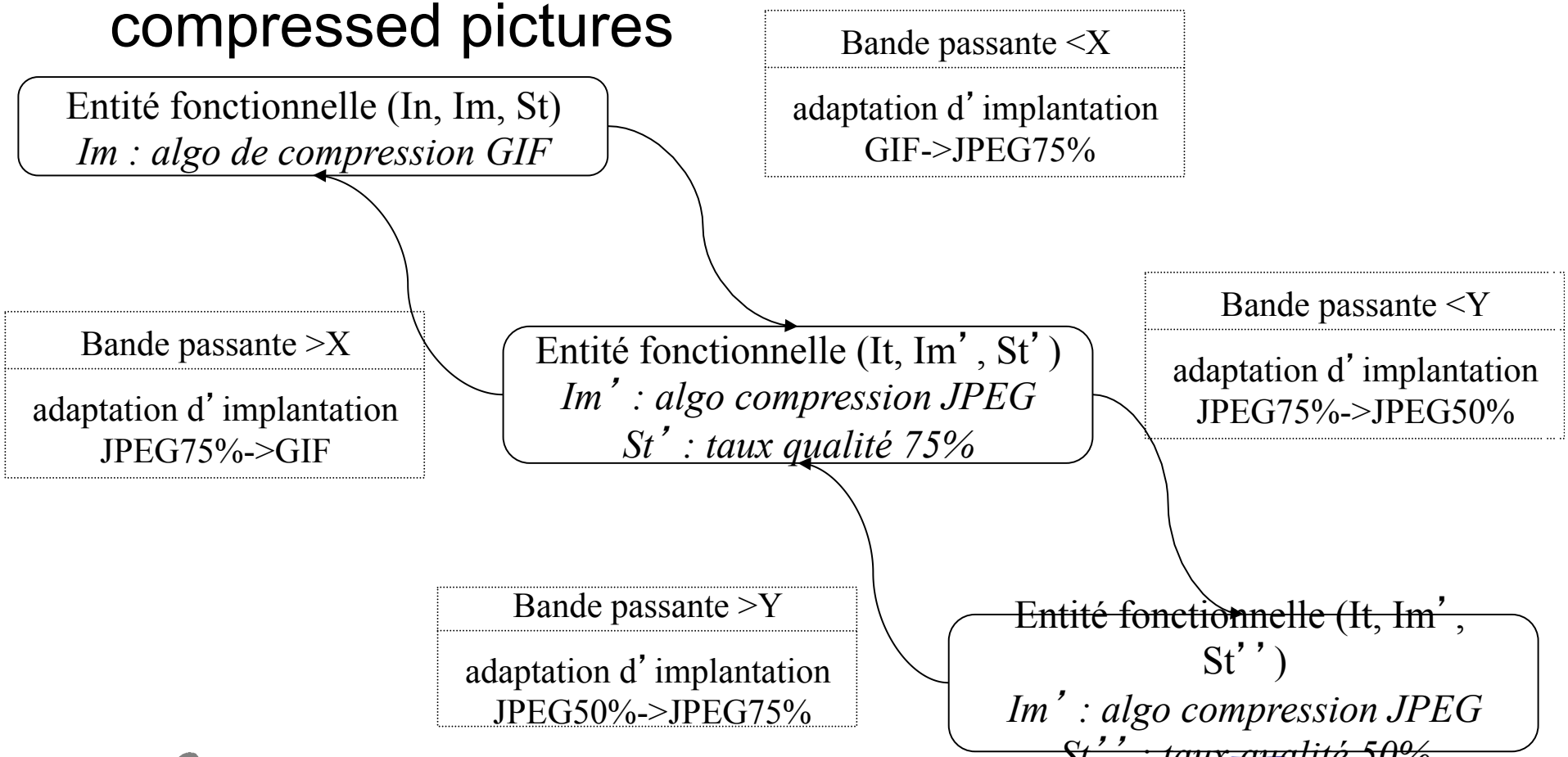
AeDEN - Adaptive and reactive entity

- Adaptation entity + reaction entity linked to a notification service



AeDEN – Strategy example

- Adaptation strategy for the transmission of compressed pictures



Adaptation conclusion

- General needs [Le Mouel 2003]:
  - Genericity : use by different kind of applications
  - Modularity : Splitting and decorrelation
  - Context-aware
  - Evolution : integration of new technologies and new functionalities
  - Dynamicity : reaction to changes without stopping the system
  - Efficiency : performance and stability

Bibliographic references - Adaptation services

- M. Cremene et al. « Adaptation dynamique de services », Decor ’2004, Grenoble, octobre 2004, pp 53-64
- CAAD : T. Chaari, F. Laforest, A. Celentano design of context-aware applications based on web services Rapport de recherche LIRIS, octobre 2004 - RR-2004-033
- RAM : N.M.N. Bouraqadi-Saadani et al. « A reflexive infrastructure for coarse grained strong mobility and its tool-based implementation. » Int. Workshop on experiences with reflexive systems. Sept. 2001
- DynamicTAO : F. Kon et al. « Monitoring, security and dynamic configuration with the dynamicTAO reflective ORB » Middleware 2000
- AeDEN : F. Le Mouël « Environnement adaptatif d’exécution distribuée d’applications dans un contexte mobile » mémoire de thèse de doctorat en informatique, Université Rennes I, 1er décembre 2003
- M. Leclercq, V. Quma, J.-B. Stefani, DREAM: A Component Framework for Constructing Resource-Aware, Configurable Middleware, IEEE distributed systems online, vol. 6, no. 9, September 2005

Synthesis

- **Roadmap to build a Pervasive System – Questions?**
  - Which kind of functionalities of a classic distributed system do I need (load balancing, fault tolerance, etc) ? Can I reuse one, or do I have to plan to redevelop some of these functionalities ?
  - How do I model my context ? What do I have to include in my context ?
  - What are the changes expected in my system ? Where and how do I have to adapt to these changes ?
  - What are the middleware frameworks and toolkits available to reach these goals ?