Middleware for Ubiquituous Computing

The WComp Solution

M. Riveill – http://rangiroa.polytech.unice.fr/riveill
University of Nice Sophia Antipolis,

In collaboration with S. Lavirotte, G. Rey, J-Y. Tigli



RAINBOW RESEARCH GROUP

+ 7-8 PhD students, + 10 MSc students, + 5 ingineers

Outline

- Part I
  - Requirements, trends, open issues associated with middleware for ubiquitous computing
- Part II
  - Our solution, called WComp
- Part III
  - An illustration – interaction control from the context

AMBIANT COMPUTING

A large number of computerized devices










AMBIANT COMPUTING

Creating applications based on these devices










CREATING RELEVANT APPLICATIONS

- Taking into account various information





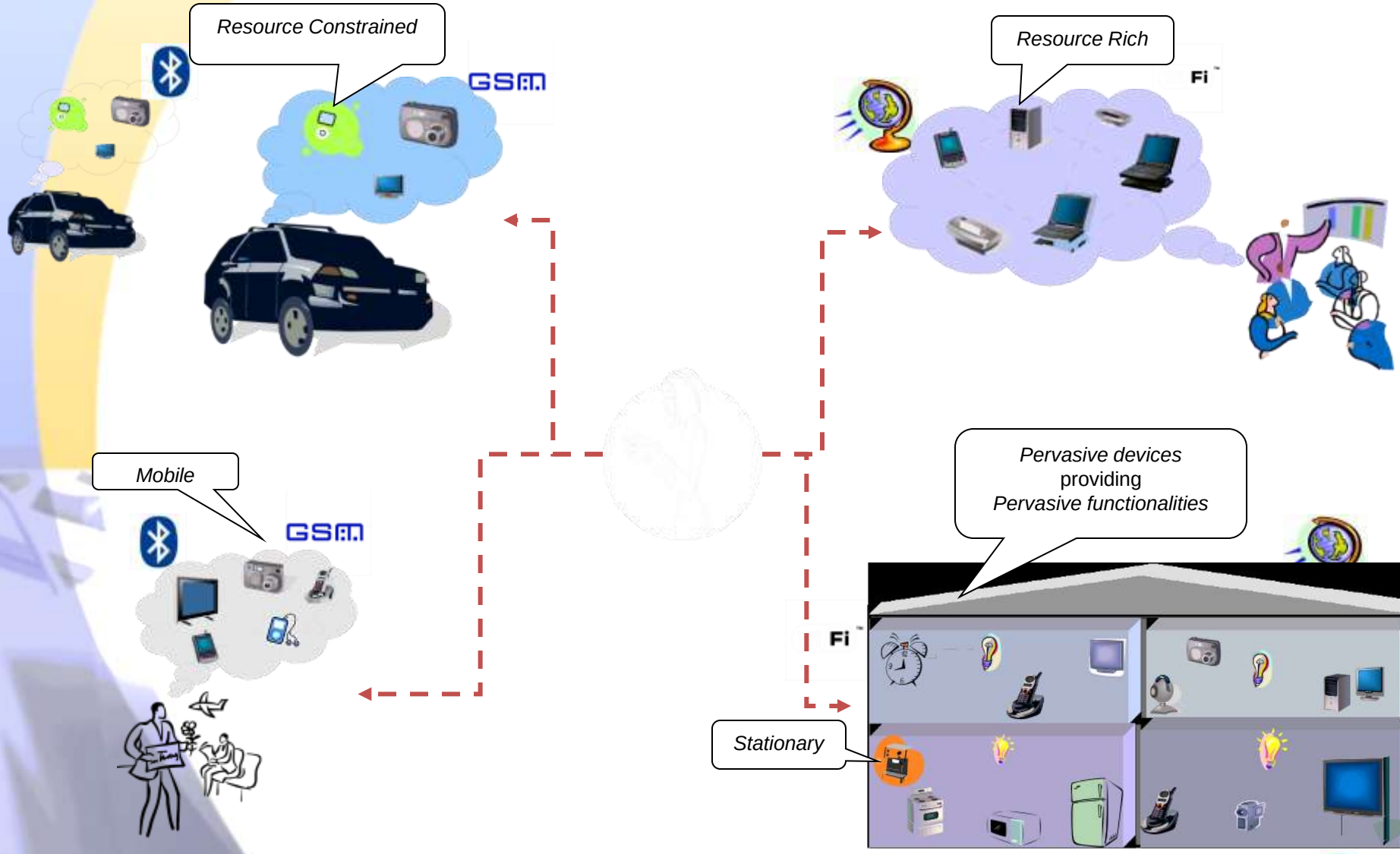




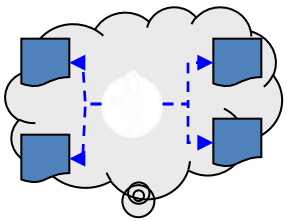

ADAPTING APPLICATIONS

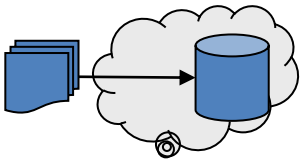
Taking into account dynamic information

AMBIANT COMPUTING



MANY PROBLEMS, MANY PROTOCOLS





WSDL
SAWSDL
OWL-S

SLP Desc
UPnP Descp

Ambiant computing

Semantic Middleware

Service Composition

Efficient Semantic Service Registry

Service Publication | Service Location | Service Matching

Interoperable Service Specification

Interoperable Communication Middleware

Multi-Protocol Management

Service Discovery Protocol | Service Access Protocol

Multi-Network Management

Pervasive Device Software Platform

PART 1: REQUIREMENTS, TRENDS, OPEN ISSUES ASSOCIATED WITH MIDDLEWARE FOR UBIQUITOUS COMPUTING

I.1 What does Ubiquitous Computing mean ?

- [Mark Weiser 1991]

« Silicon-based information technology, is far from having become part of the environment. »

Scientific American,
Vol. 265 N.9, pp. 66-75, 1991

Ubiquitous applications used everyday life connected objets and devices

I.1 First ubiquitous applications

- Smart Objets and Devices are well-known at Design time, Embedding for smart control
  - Embedded systems for cars, airplanes, etc
- First Ubiquitous Application are generally Ad-Hoc applications without middleware, creating new computing devices
  - Hi-tech, silicon-based gadgetry, e.g. PDAs, cell phones, mp3 players, active displays

1.1 First requirements

- First requirements:
  - System requirement: Ubiquitous Applications applications are continuously interacting with a real world
  - Design requirement: Smart objects and devices must be able to communicate spontaneous information from the environment to the application
  - Software requirement: Software application must be event-driven

I.2 New Constraints for ubiquitous computing : Heterogeneity of devices

- Technological Heterogeneity of smart objects and devices
  - Numerous software and network technologies

- But also Semantic Heterogeneity
  - Various Smart Objects and Devices (sensors, mobile phones, …, coffee machine, mug …)
  - Variation of capabilities between them (ex. from J2ME to JSRs in mobile phones)

I.2 New Constraints for ubiquitous computing : Mobility

- WSI user-centered reference Model
  - Spheres of interaction of devices, from Personal Area Network to World Wide Web

S. Arbanowski, M. Lipka, K. Mössner, K. Ott, R. Pabst, P. Pulli, A. Schieder, M. A. Uusitalo. The **WSI Reference Model** for the Wireless World. *Proceedings of IST Mobile Summit 2003*.

- Users and Devices are Mobile

1.3 Main Ubiquitous Computing Characteristics

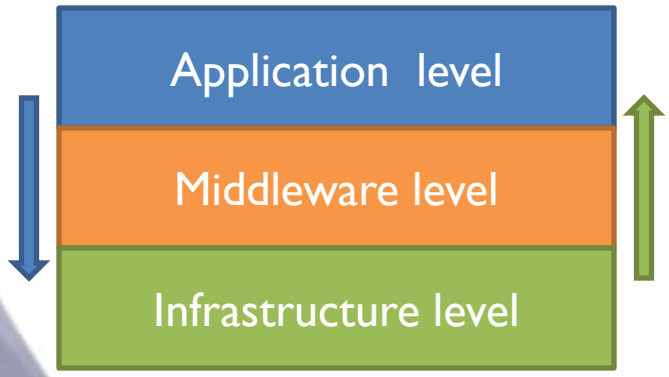
- Three main characteristics are :
  - Use embedded devices in a real environment
  - Deal with Multiple Heterogeneous Devices
  - Deal with Highly Dynamic variation at Runtime

I.4 Requirements for software composition between smart objects and device

- Main requirements for composition are :
  - *Interating with Real World* ➔ Event based interaction in the composition
  - *Heterogeneous Devices* ➔ Discover at runtime, new smart objects and devices
  - *Mobility* ➔ Deal with dynamic appareance and dispareance of smart objects and devices
  - *Mobility* ➔ Deal with dynamic composition (at runtime)
  - *Mobility* ➔ Distribution must be explicit to deal with the evolution of the infrastructure (we distinguish local and distributed composition)

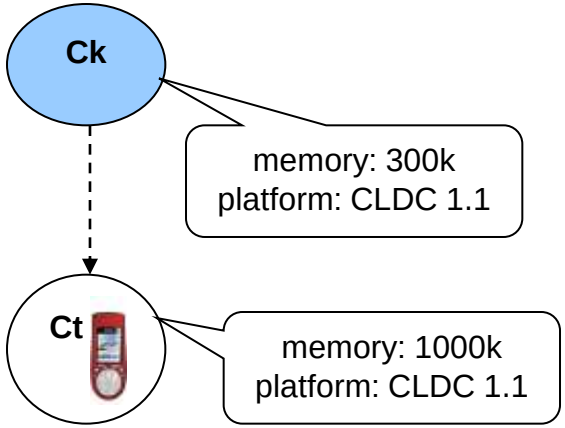
I.5 New challenges and open issues in adaptation

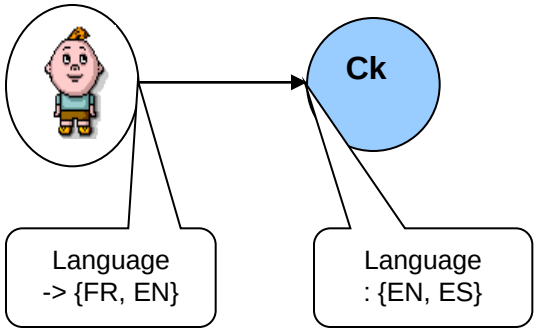
- Ubiquitous Computing applications are continuously interacting with a real world, partly unknown at design time and, always changing at runtime in uncountable manner
- We witness to a kind of inversion in the classical software methodology where the software applications levels are much more stable and stationary than the software infrastructure level.



ADAPTATION

- On an application to its execution environment
  - Taking into account the execution context
  - Taking account of user need
- Dynamic adaptation
  - aims to take into account "instantaneously" changes in the runtime environment
  - Not only to continue to run the application
  - But also, and especially to make the best new configuration





I.6 Multi-Domain adaptation as open issue

- Ubiquitous Middleware must continuously adapt at runtime, application requirements to changing computing environment (due to mobility) in multiple domains :
  - HMI,
  - Power,
  - QoS, Network bandwidth,
  - Devices availability, …

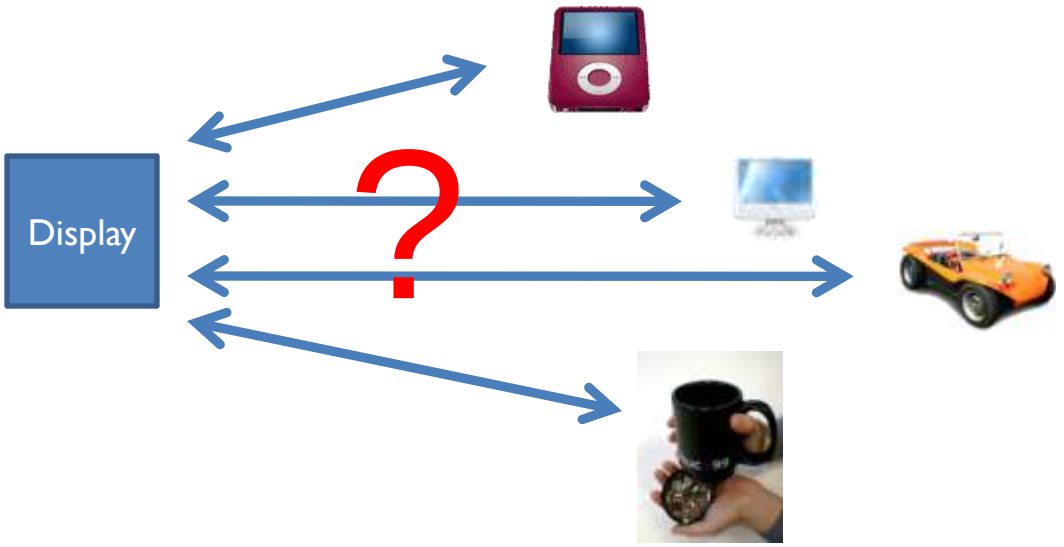
I.7 Reactive adaptation as open issue

- Reactive adaptation is defined as the ability for the Ubiquitous applications to perceive the environment and adapt to changes in that environment in a timely fashion.

- Ubiquitous Middleware must provide reactive adaptation mecanism to changing operational environment.

I.8 SEMANTIC ADAPTATION AS OPEN ISSUE

- Ubiquitous Middleware must match at run-time the current operational environment and application requirements.

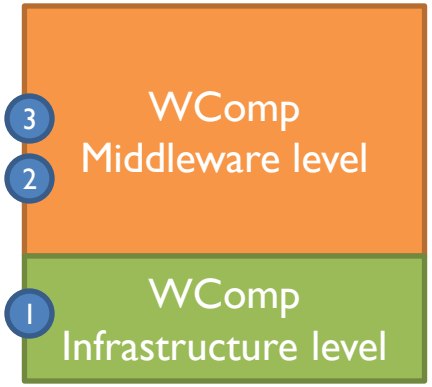


Can match with ?

1.9 Autonomic Computing as an objective

- Self-Configuration: Configuration automatique des composants
- Self-Healing: Découverte automatiqu et correction aux fautes
- Self-Optimization: Contrôle des ressources pour optimiser le fonctionnement
- Self-Protection: Identification proactive et protection contre les attaques de tous ordres

PART II: OUR SOLUTION, CALLED *WCOMP*



1. WComp Infrastructure : based on Web services for Device
2. Composition: WComp Local and Distributed composition (LCA and SLCA models),
3. Adaptation: WComp Reactive adaptation using Aspects of Assembly (AA)

Our approach

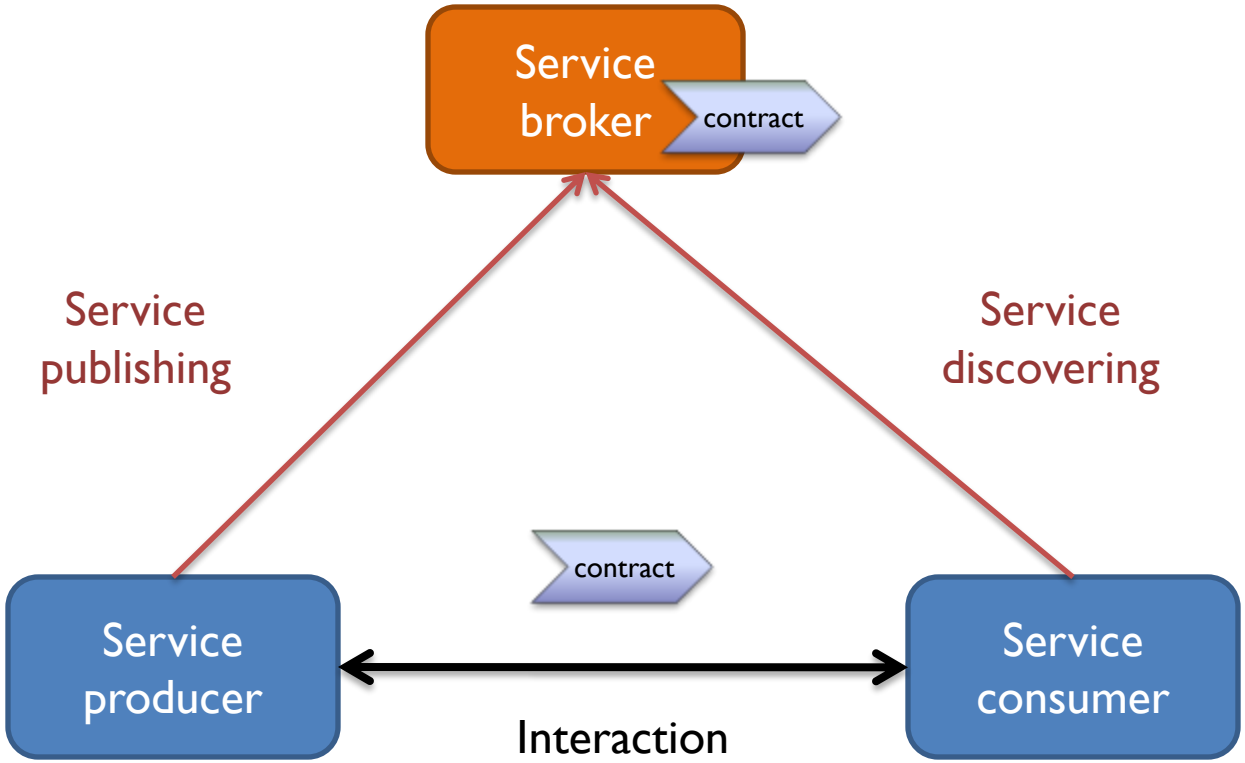
- Taking into account the needs of users
  - Adapting an application to the person using it and not vice versa
- Approach composition
  - Use of 'bricks' software: components or services
    - Availability of 'bricks' depends on changes in the runtime and the availability of devices
  - Knowledge of application architecture
    - Evolution Rules unknown a priori
- Objectives
  - Produce software tools (middleware) for
    - Self-adaptation of the application
    - Without explicit programming of evolution

II.1 Wcomp Infrastructure

- Main requirements :
  - Decentralized and Contextual discovery
  - Managing Apperance and Disapperance of the service
  - Event based interactions
- Solution : WComp Infrastructure based on service for Device
  - From Service to Service for Device
  - From Web service to Web service for Device

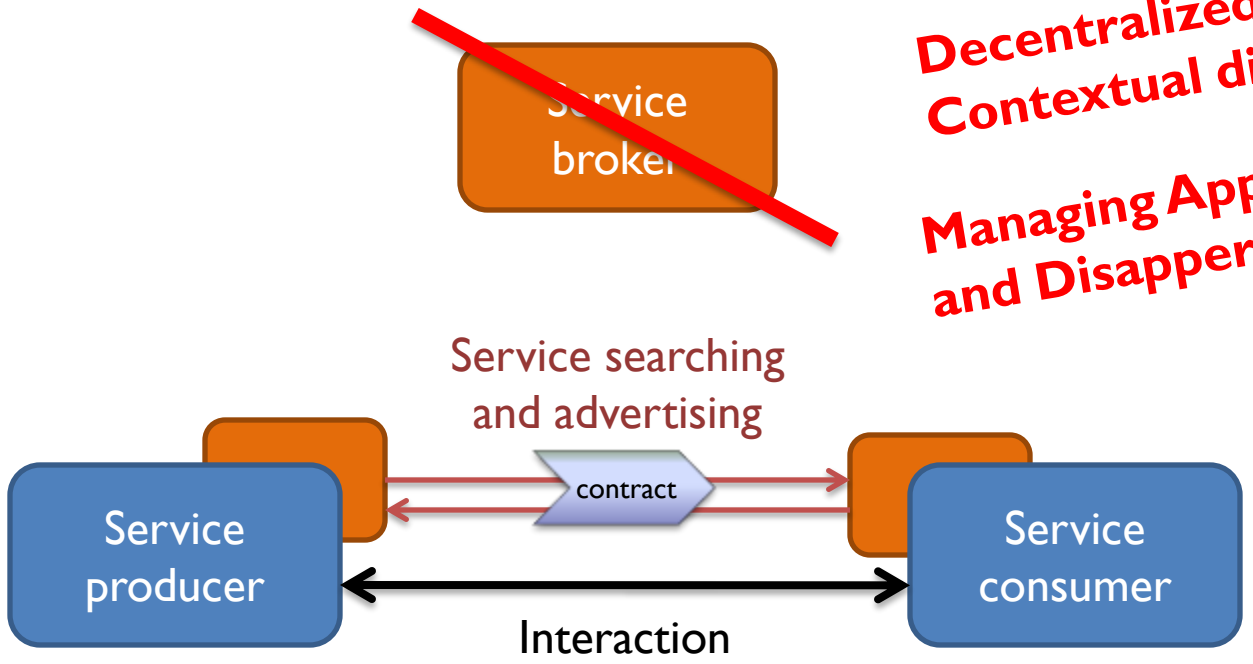
II.1 From a Service Oriented Approach

- Standard service cycle of use



II.1 Web Services for Devices

- New requirements for WComp Infrastructure
  - Decentralized and Contextual discovery
  - Managing Apperance and Disapperance of the service



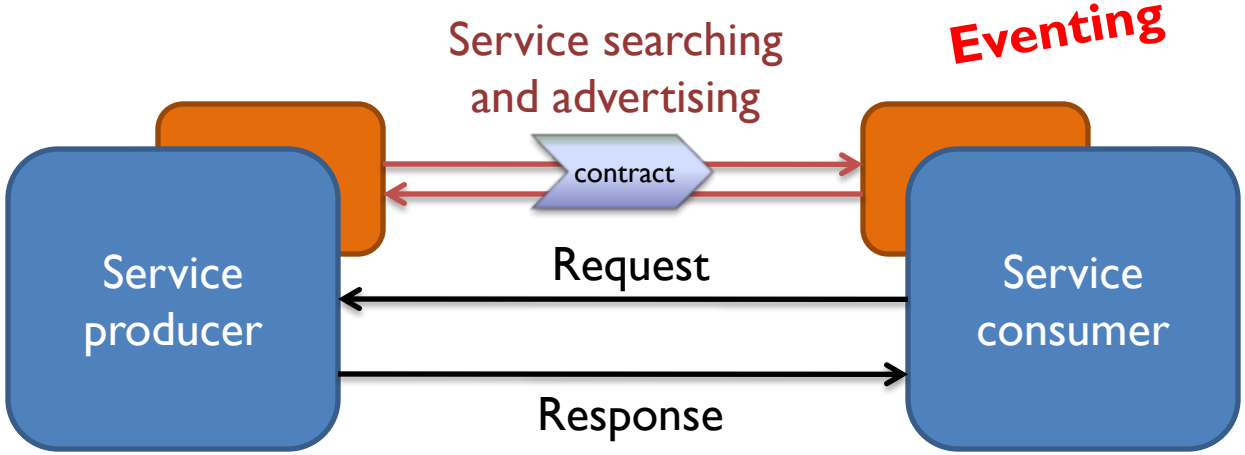
II.1 Web Services for Devices

- New requirements for WComp Infrastructure
  - New ways of interacting: Eventing

Decentralized and Contextual discovery

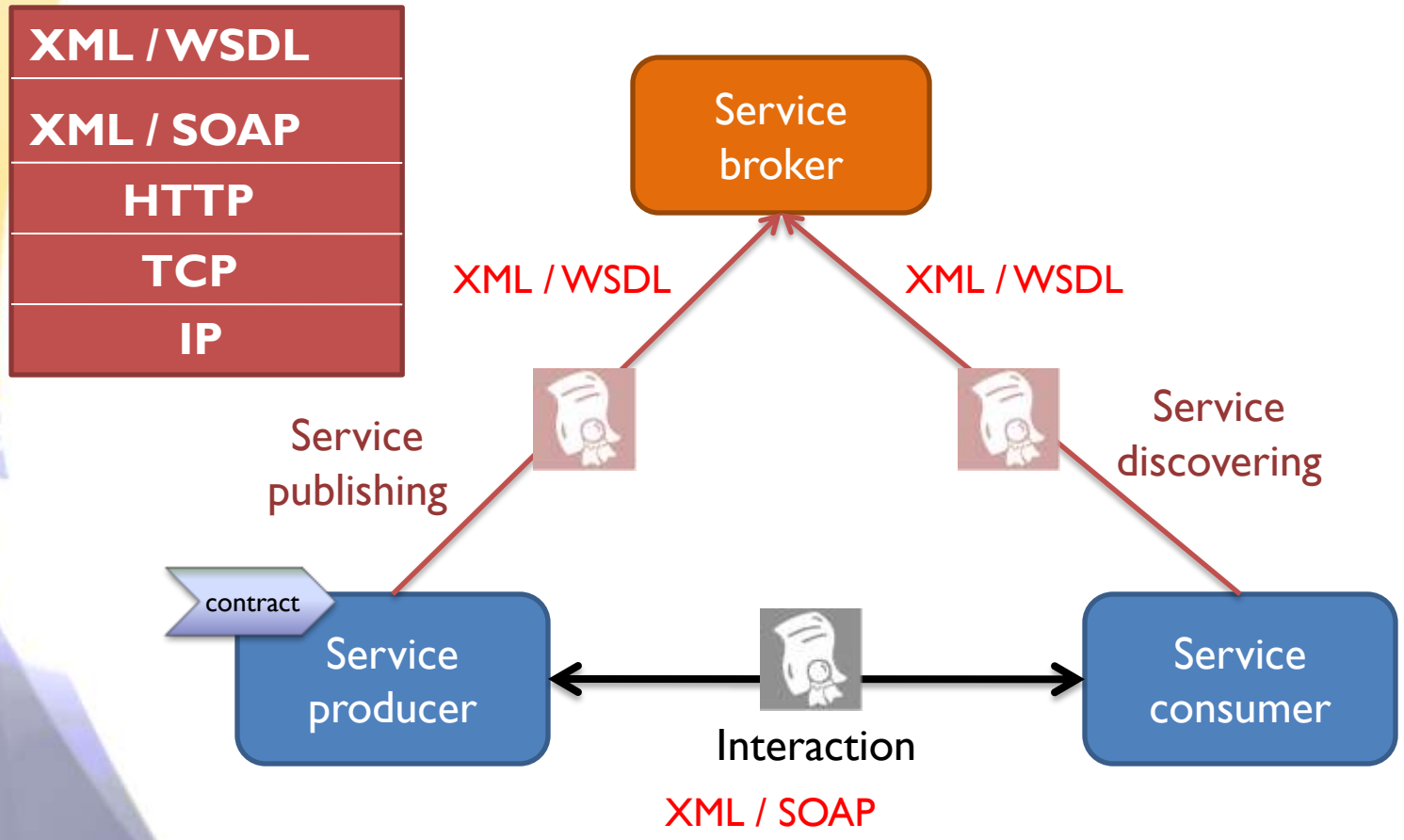
Managing Apperance and Disapperance

Eventing



II.1 From Web Service Oriented Approach

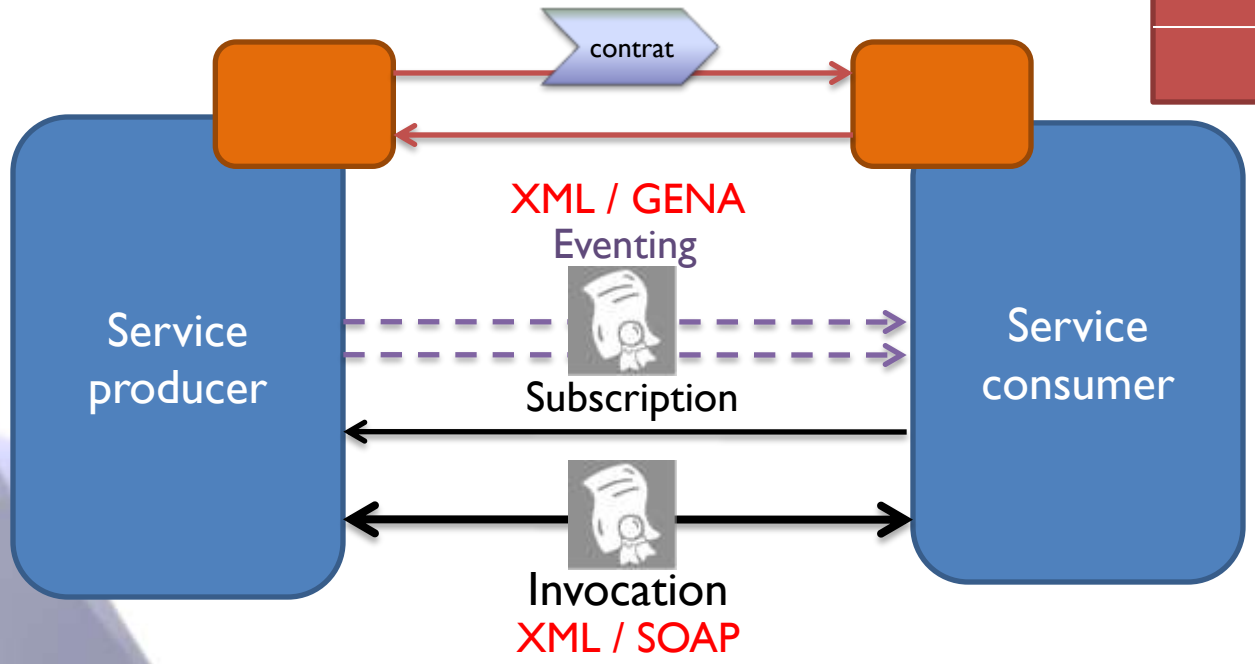
- Web Services using Web technologies



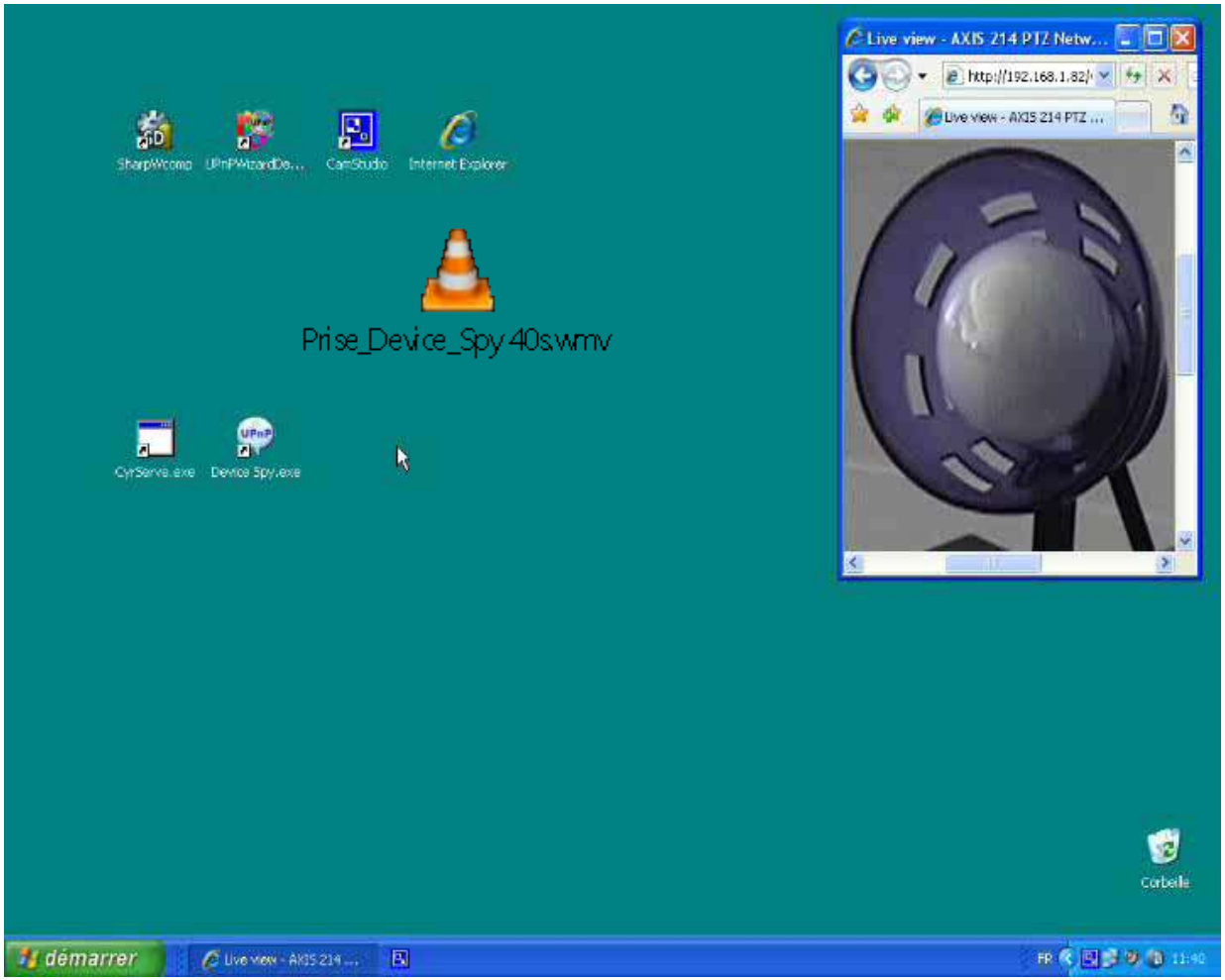
II.1 To Web Service for device

- Example : UPnP (like DPWS now)

XML / SSDP
XML / SOAP/GENA
HTTP/HTTPU/HTTMU
TCP / UDP
IP



II.1 Demo: Services for Physical Devices in WComp (40 s)



II.1 Demo : Services for Virtual Devices in Wcomp (2 min.)

II.2 WComp Local and Distributed composition

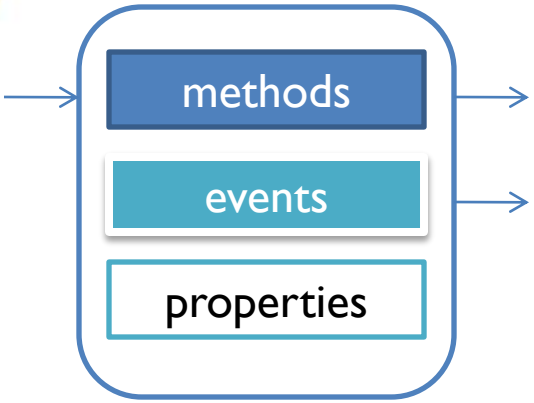
- Main requirements :
  - Contrarly to most middleware approaches, distribution must be explicit to deal with the evolution of the infrastructure (location based)
  - Composition must be event based
  - At runtime ….
- Solution :
  - Event based Local Composition : LCA (Lightweight Component Model) for each application execution node.
  - Event based Distributed Composition : SLCA (SLCA (Service Lightweight Component Model) to enable reusability.

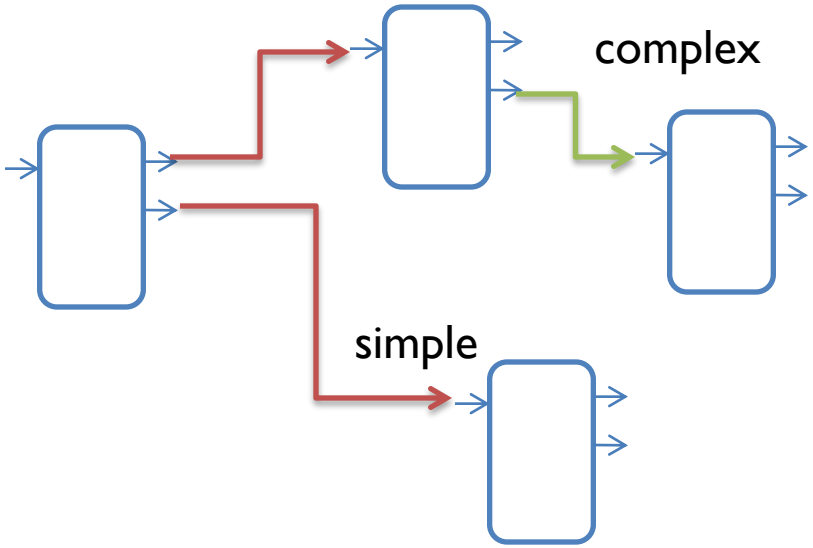
II.2 Main Features of LCA Model :

- Goal:
  - Allow to compose Services for Device between them towards a multiple devices ubiquitous application.
- Principles
  - LightWeight Components Approach
    - Like OpenCom, JavaBeans, PicoContainer
  - On the same execution node
  - For each execution node, a container dynamically manage the assembly of components
  - Event-based interaction between components
  - Blackbox LightWeight Components

II.2 LCA COMPONENTS, PORTS AND CONNECTORS

LCA components





Connectors

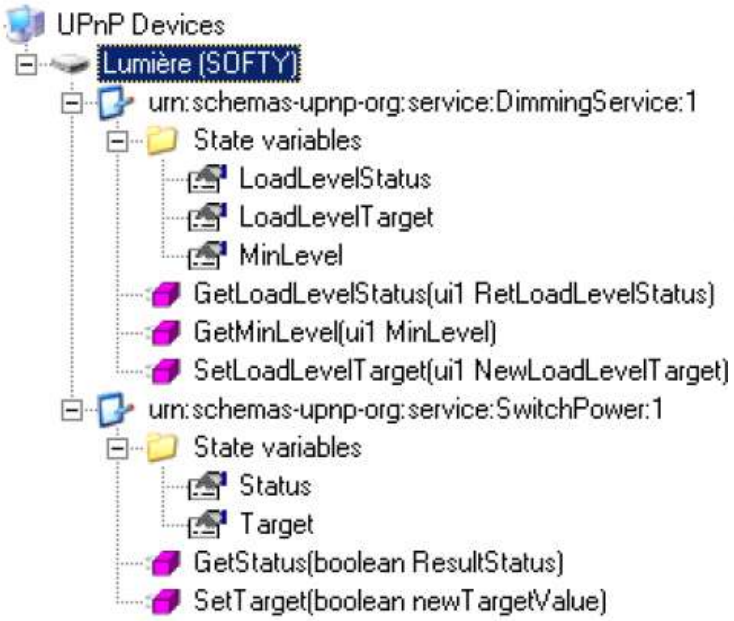
Simple Event based Connector

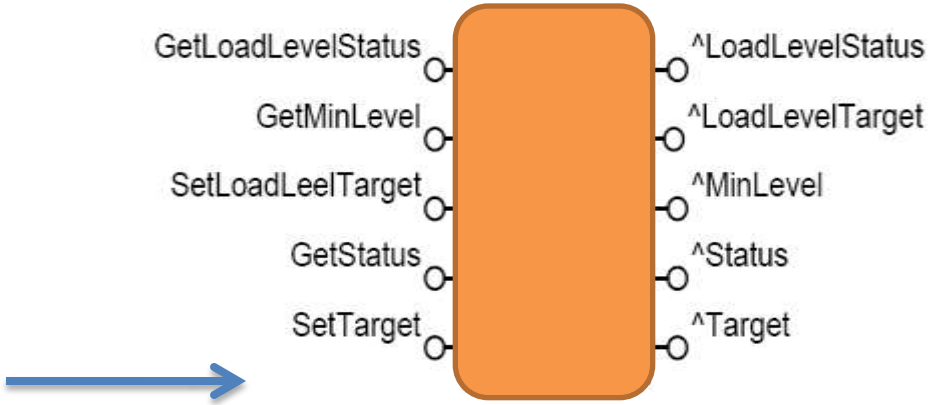
C1.Event (param) → C2.Method (param)

Complex Event based Connector

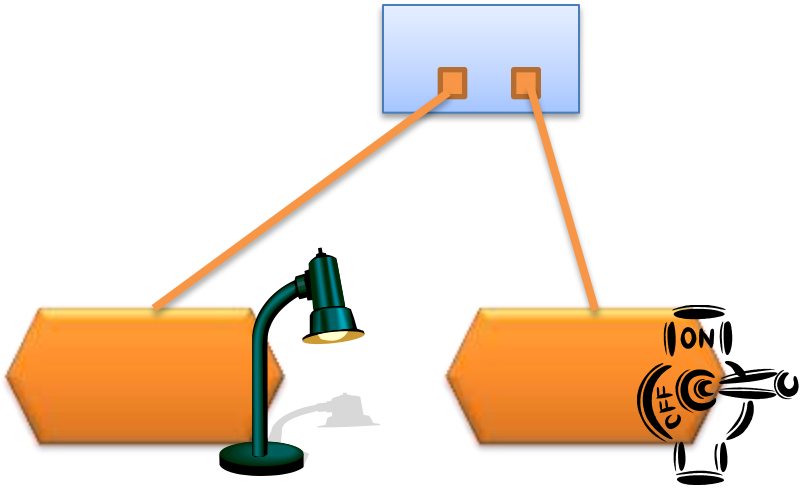
C1.Event (param) → C2.Method (C1.GetAProperty())

II.2 LCA Proxy components to access to Services for Devices





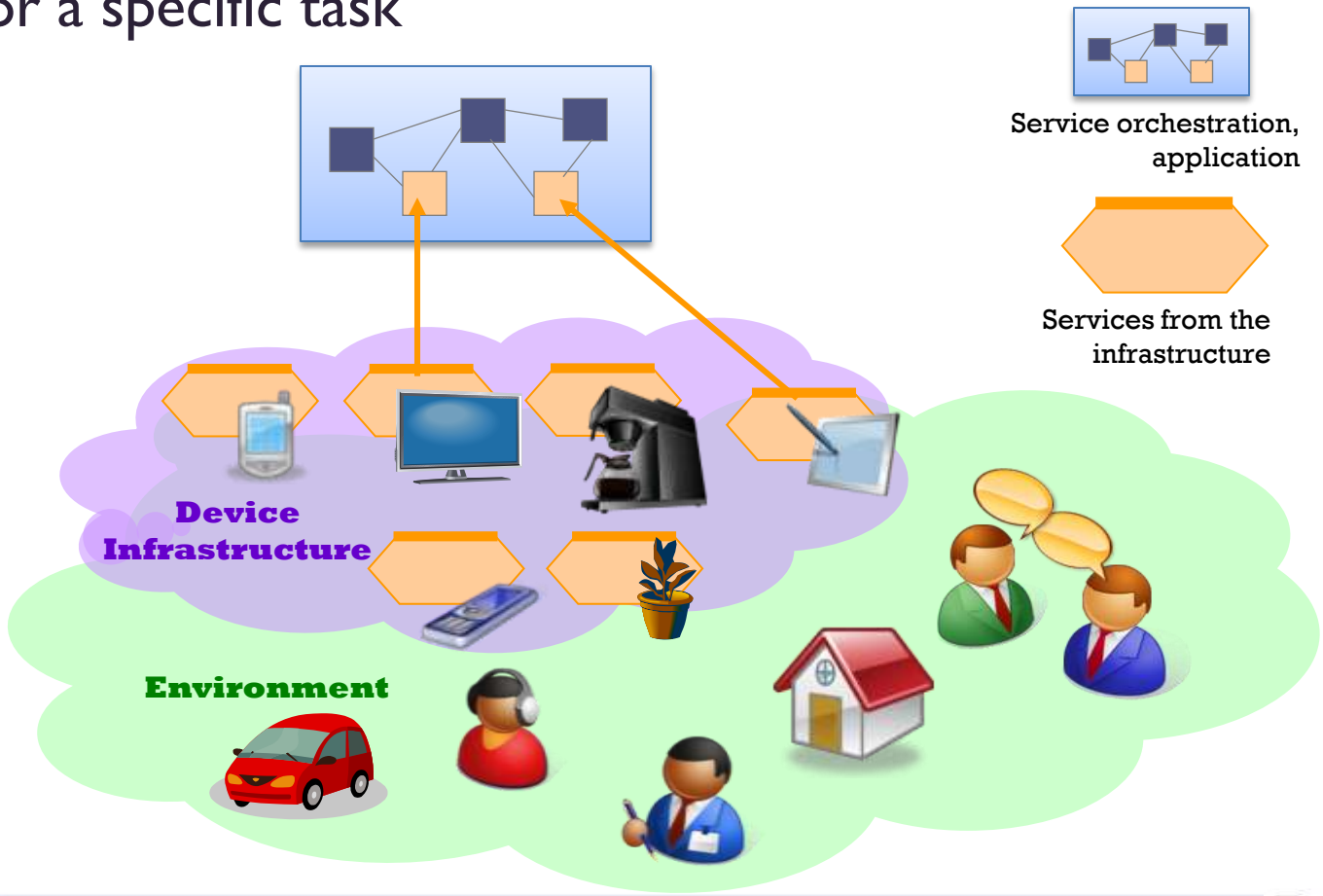
Proxy Component



Service for Device

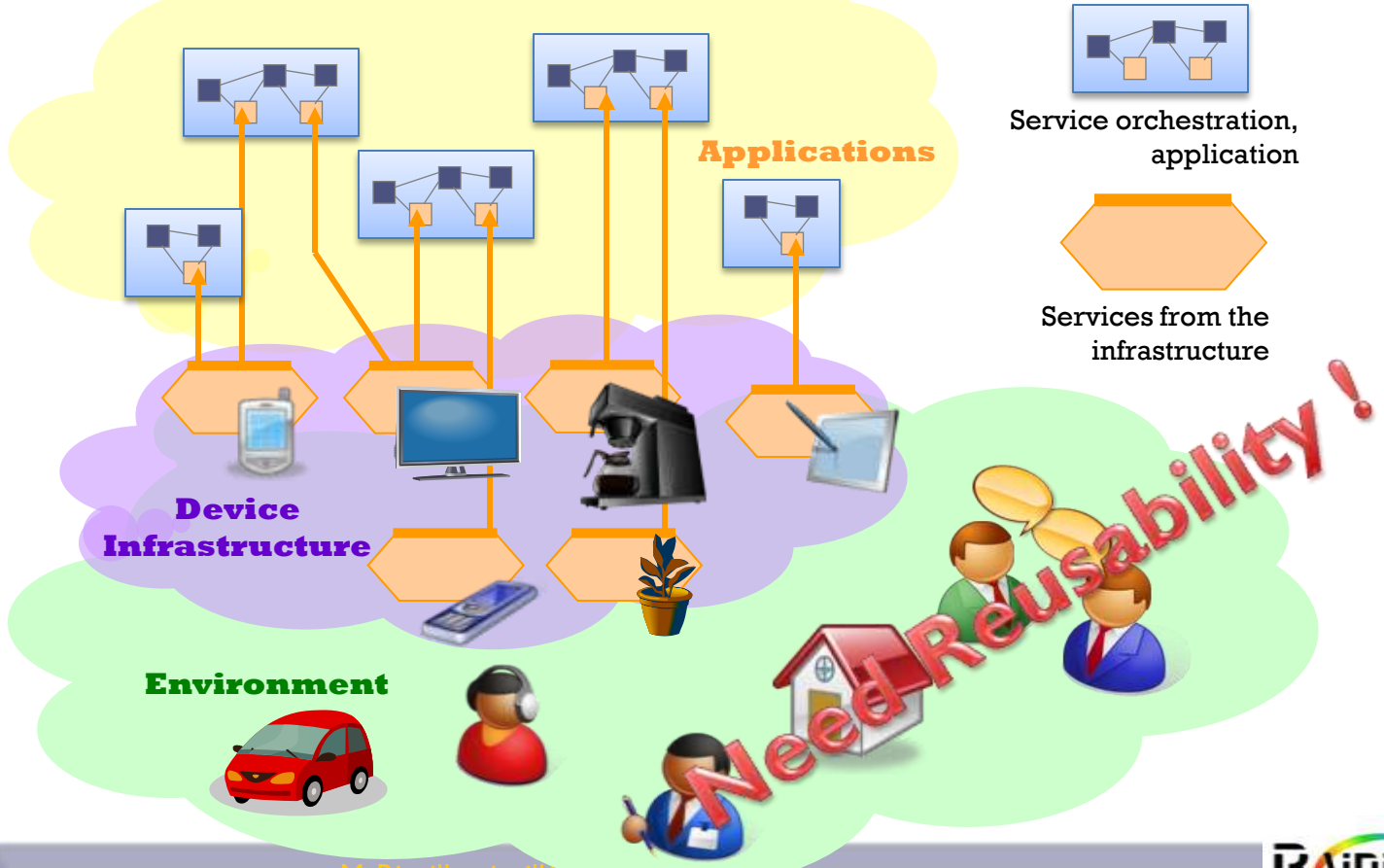
II.2 Application using Web Services for Devices

- Service orchestrations create service-based applications for a specific task



II.2 Applications using Web Services for Devices

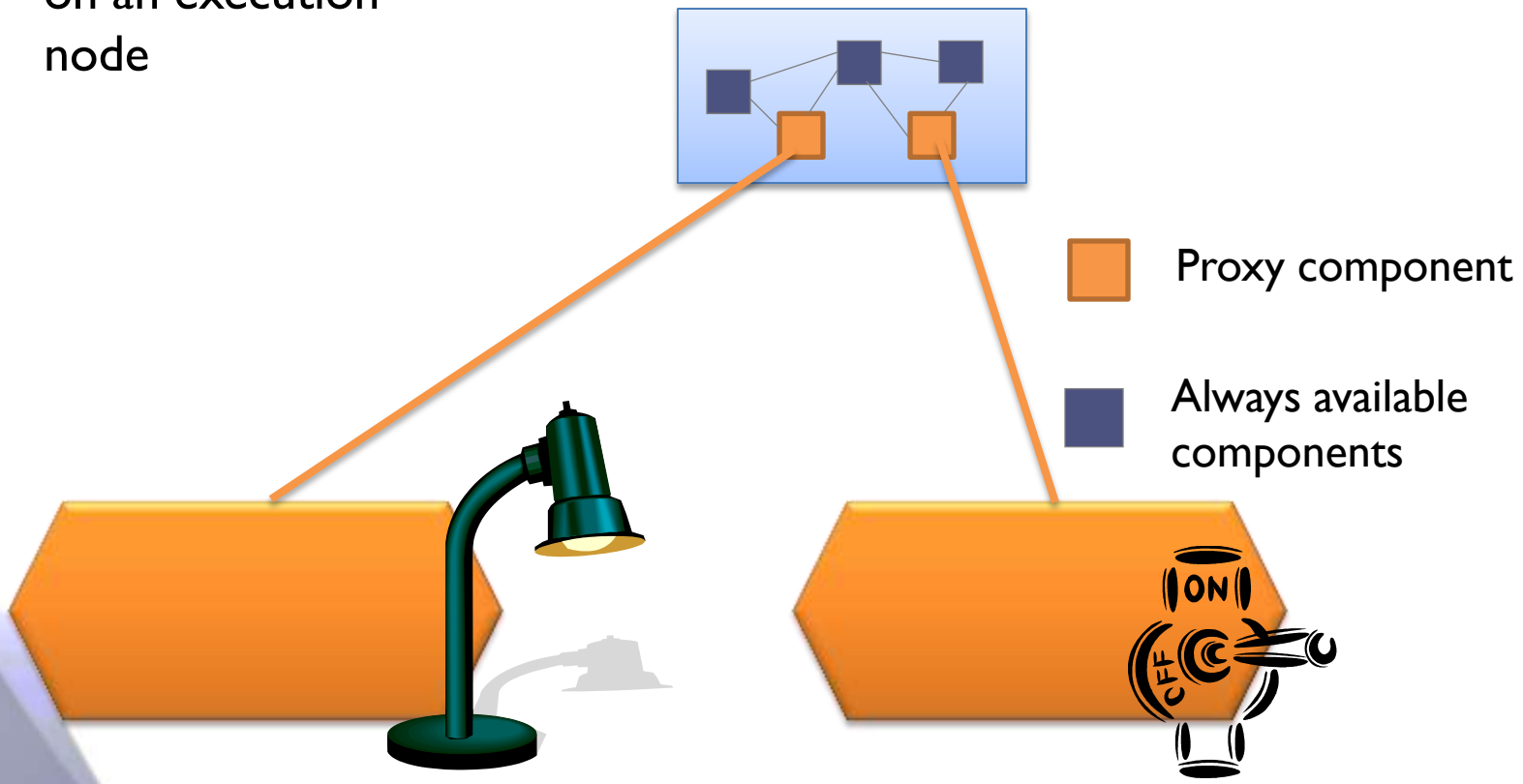
- Applications are specific and not reusable
  - Lots of applications need to be created



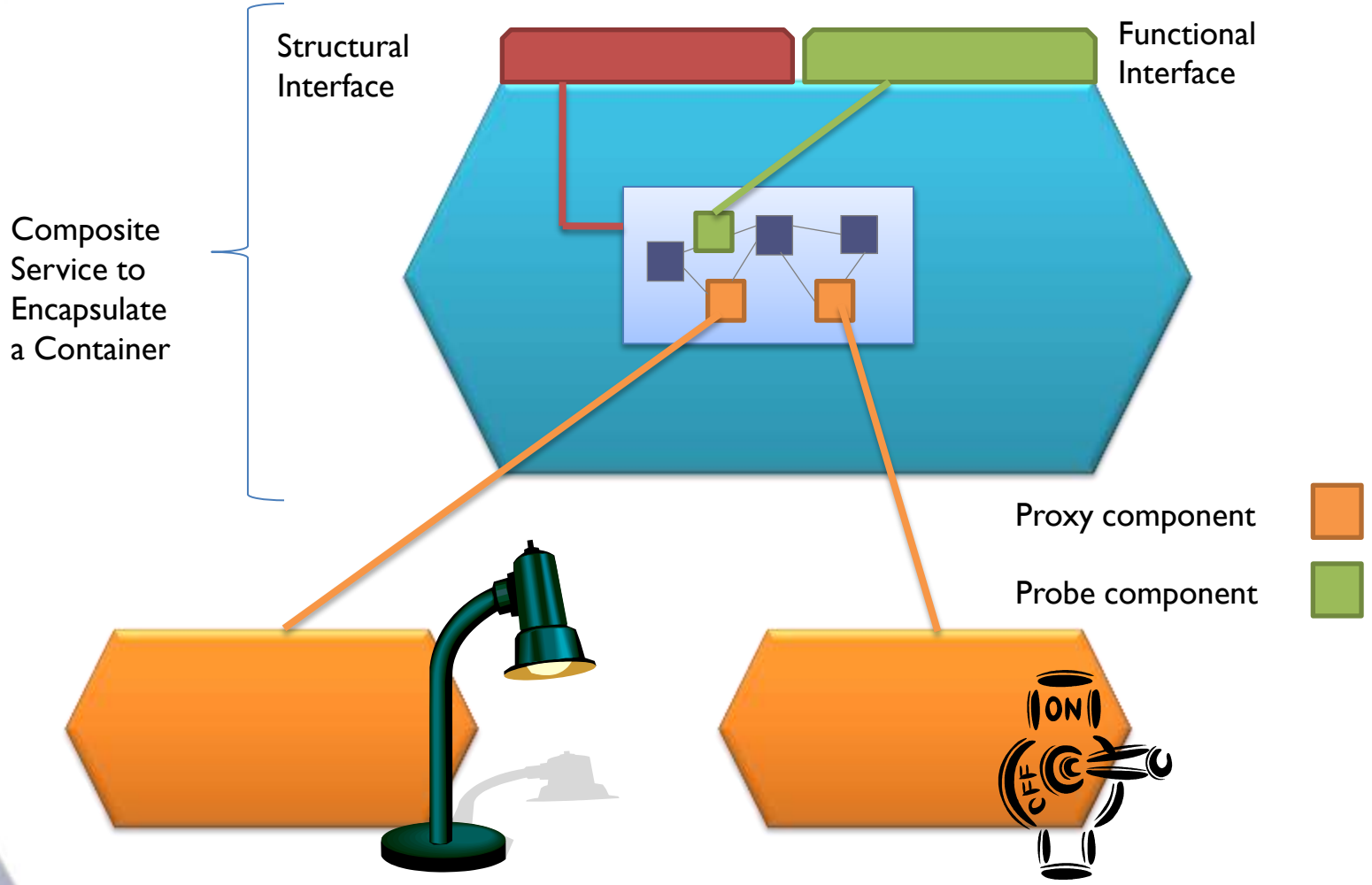
II.2 Distributed Composition: From LCA to SLCA

Application on an execution node

Assembly of Components

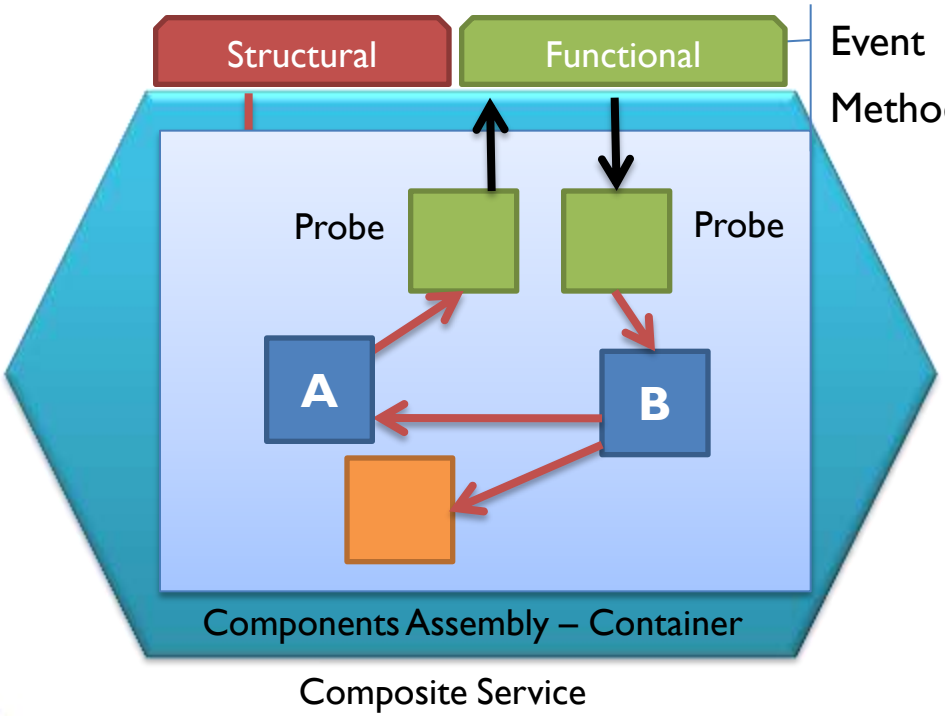


II.2 Distributed Composition: From LCA to SLCA



II.2 Composite Service : SLCA

- Probe Components

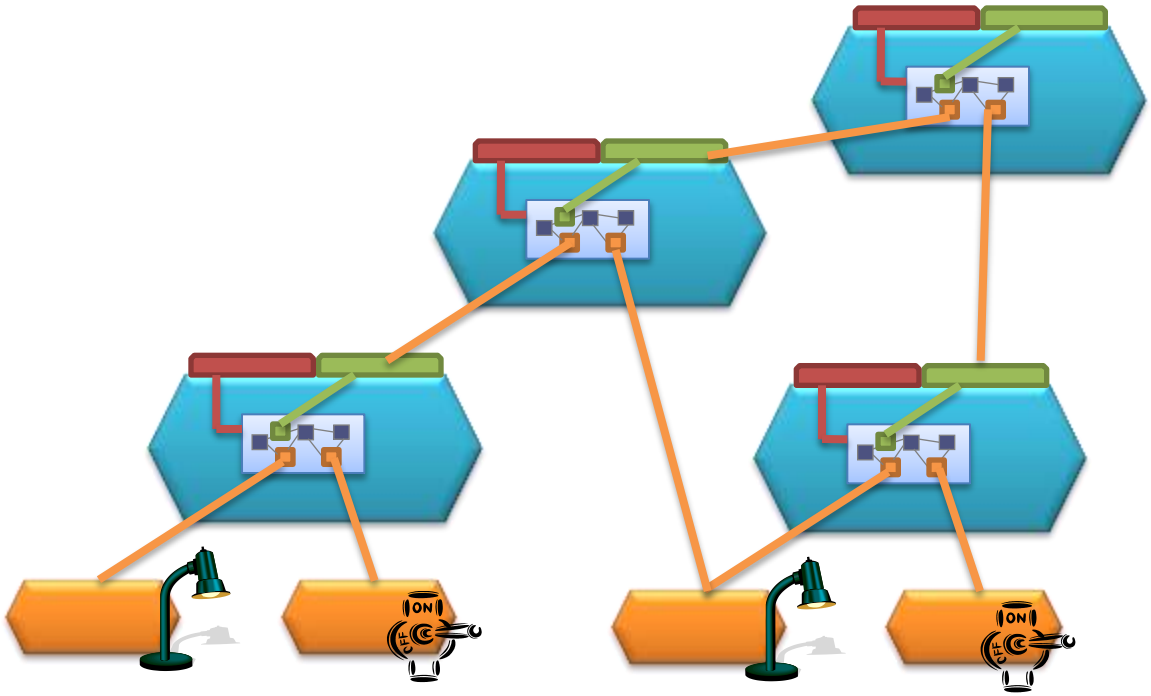


Probe emitter :
Local event → Service Event

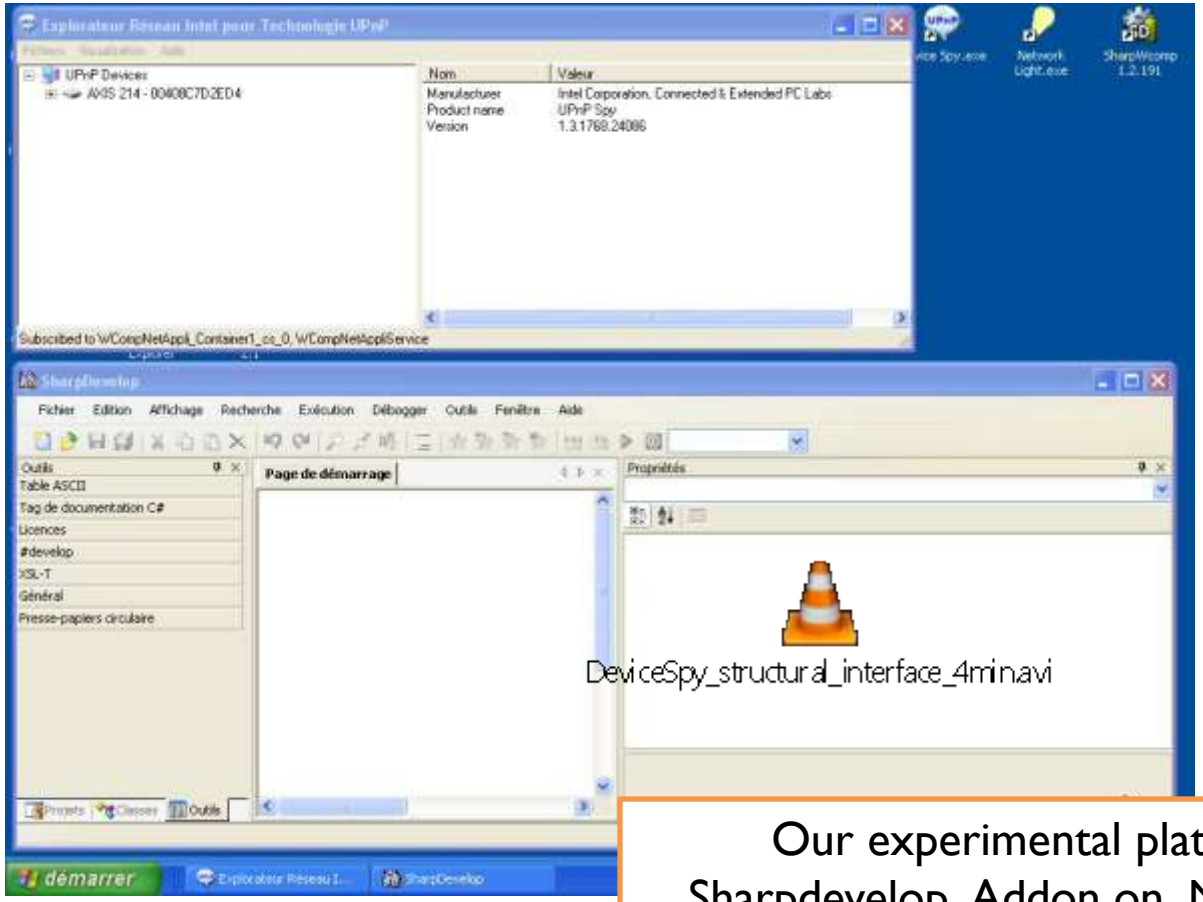
Probe receiver :
Service Method Invocation → Local Event (Request)

II.2 Distributed and Dynamic composition with SLCA, Demo in WComp

- The Ubiquitous Applications are spreading in a graph of Composite Services for Device

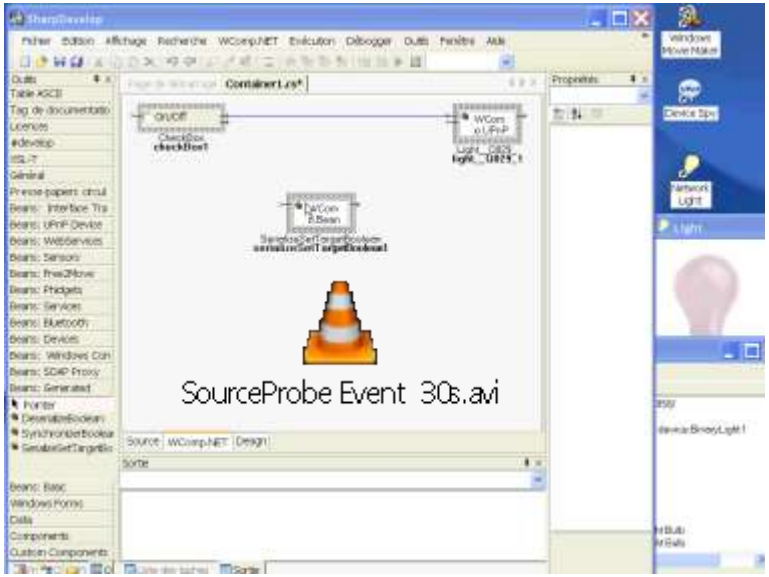


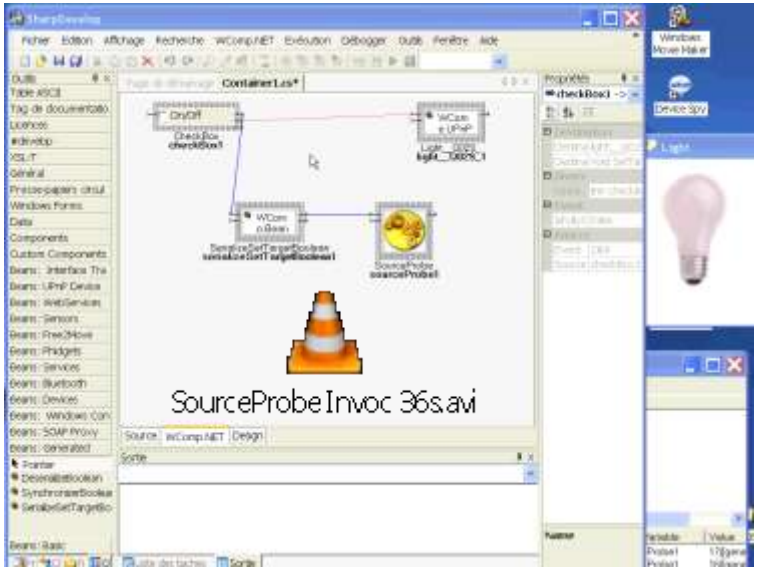
II.2 SLCA Demo in Wcomp: Control interface (4 min)



Our experimental platform is as a Sharpdevelop Addon on .Net Framework.
http://rainbow.i3s.unice.fr/wikiwcomp/

II.2 SLCA Demo in WComp: functionnal interface (1min)



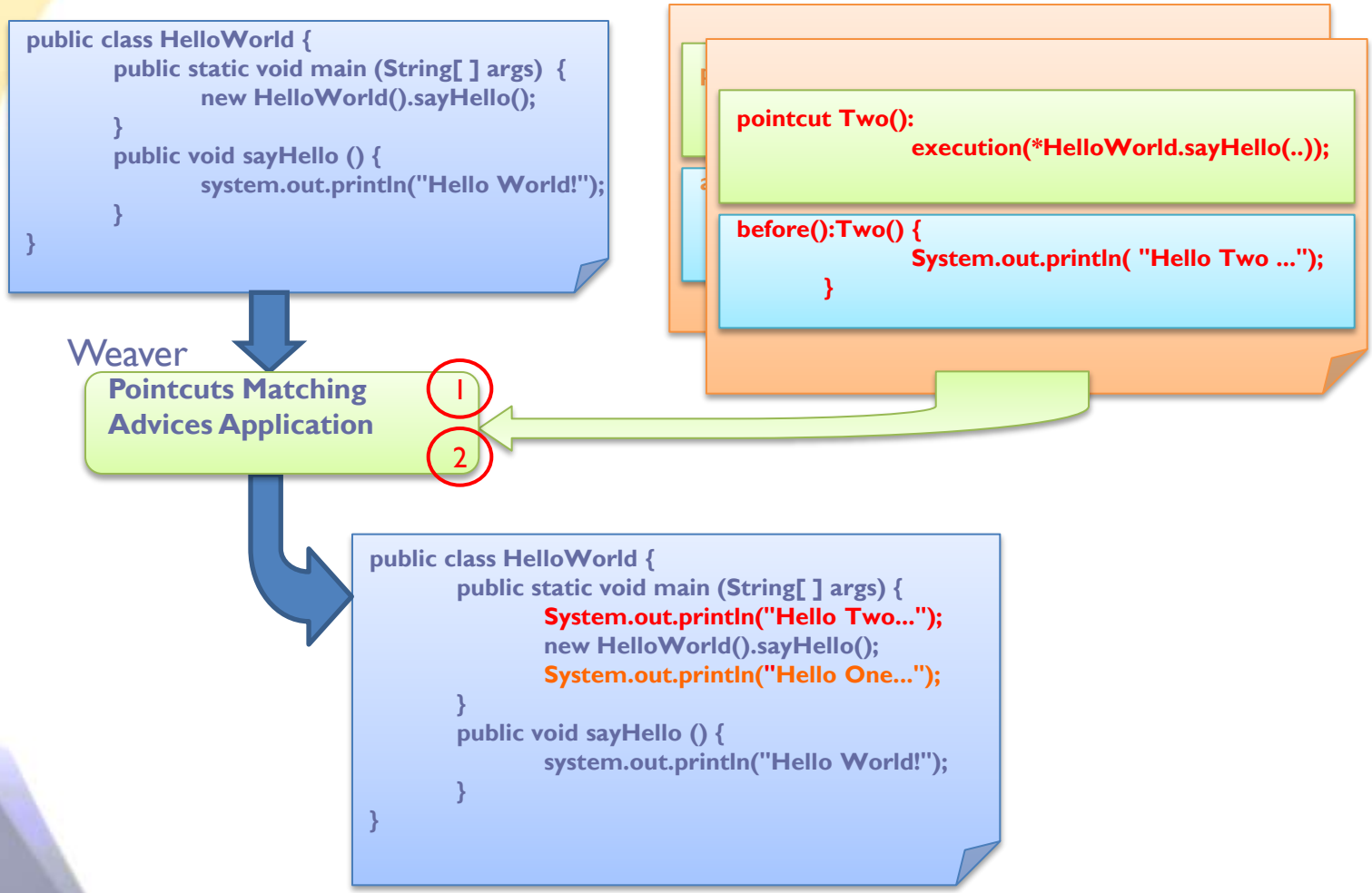


Our experimental platform is as a Sharpdevelop Addon on .Net Framework.
http://rainbow.i3s.unice.fr/wikiwcomp/

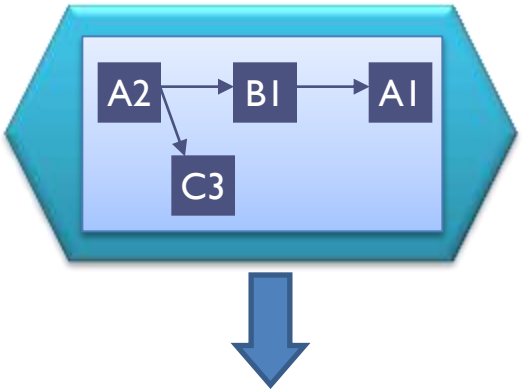
II.3 REACTIVE ADAPTATION: ASPECT OF ASSEMBLY

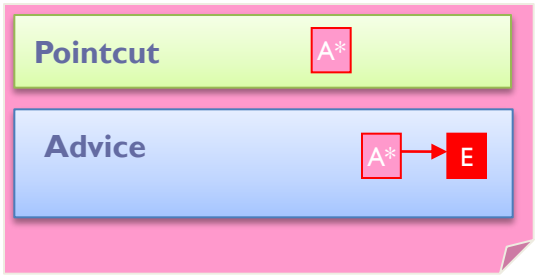
- Main requirements :
  - Reactive adaptation
  - Multi-domain adaptation
  - Semantic adaptation
  - At runtime ….
- Solution : *Aspect of Assembly*
  - *Principles of AA*
  - *Demo : AA in WComp*
  - *Reactivity and response time : Experiments and Results*

II.3 REMINDER: AOP PRINCIPLES

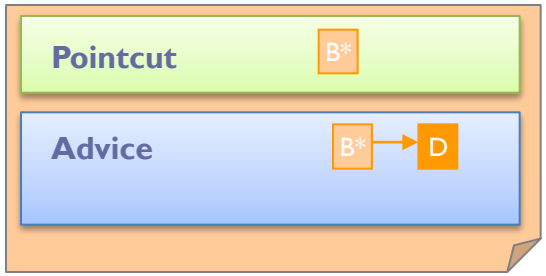


II.3 Aspect of Assembly Principles



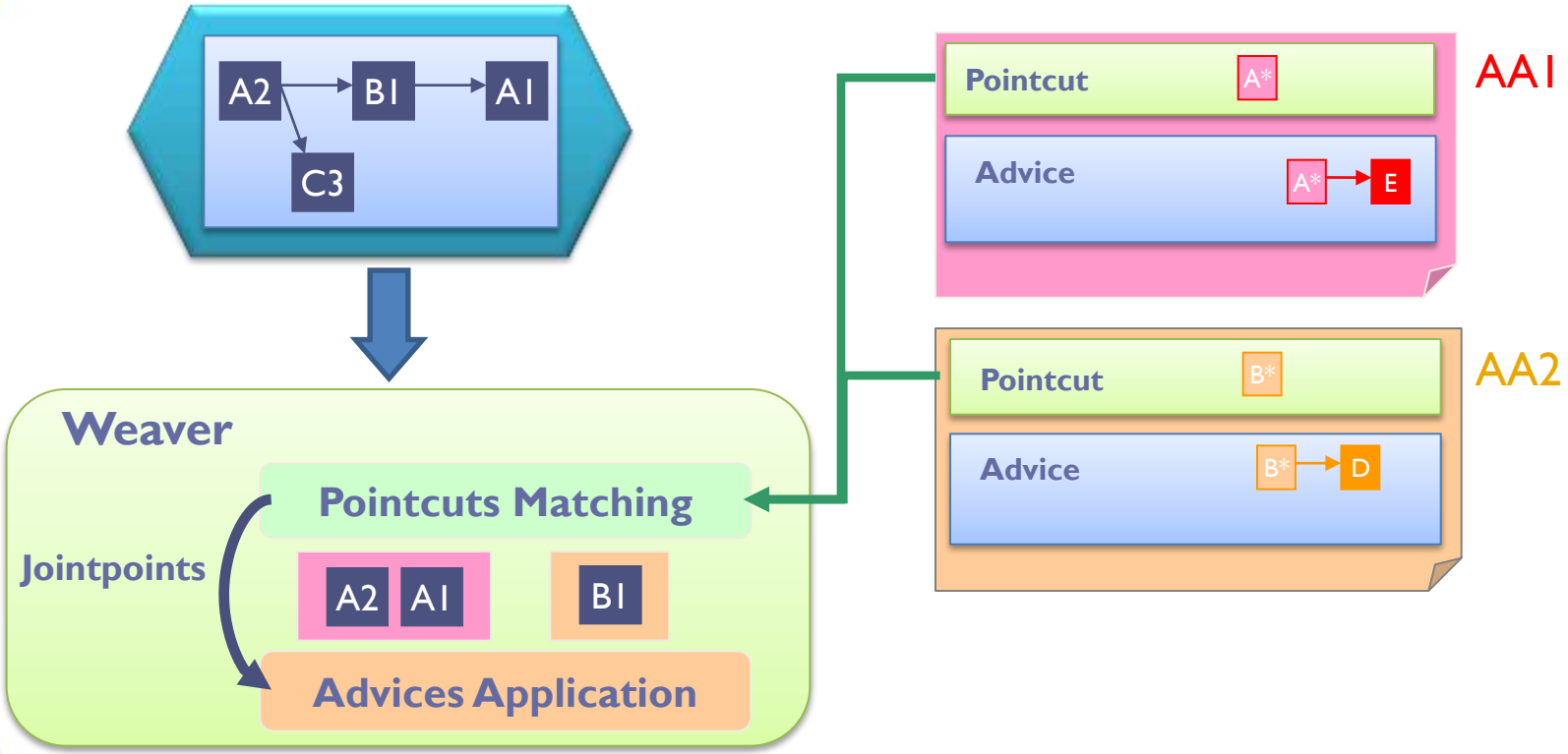


AA1

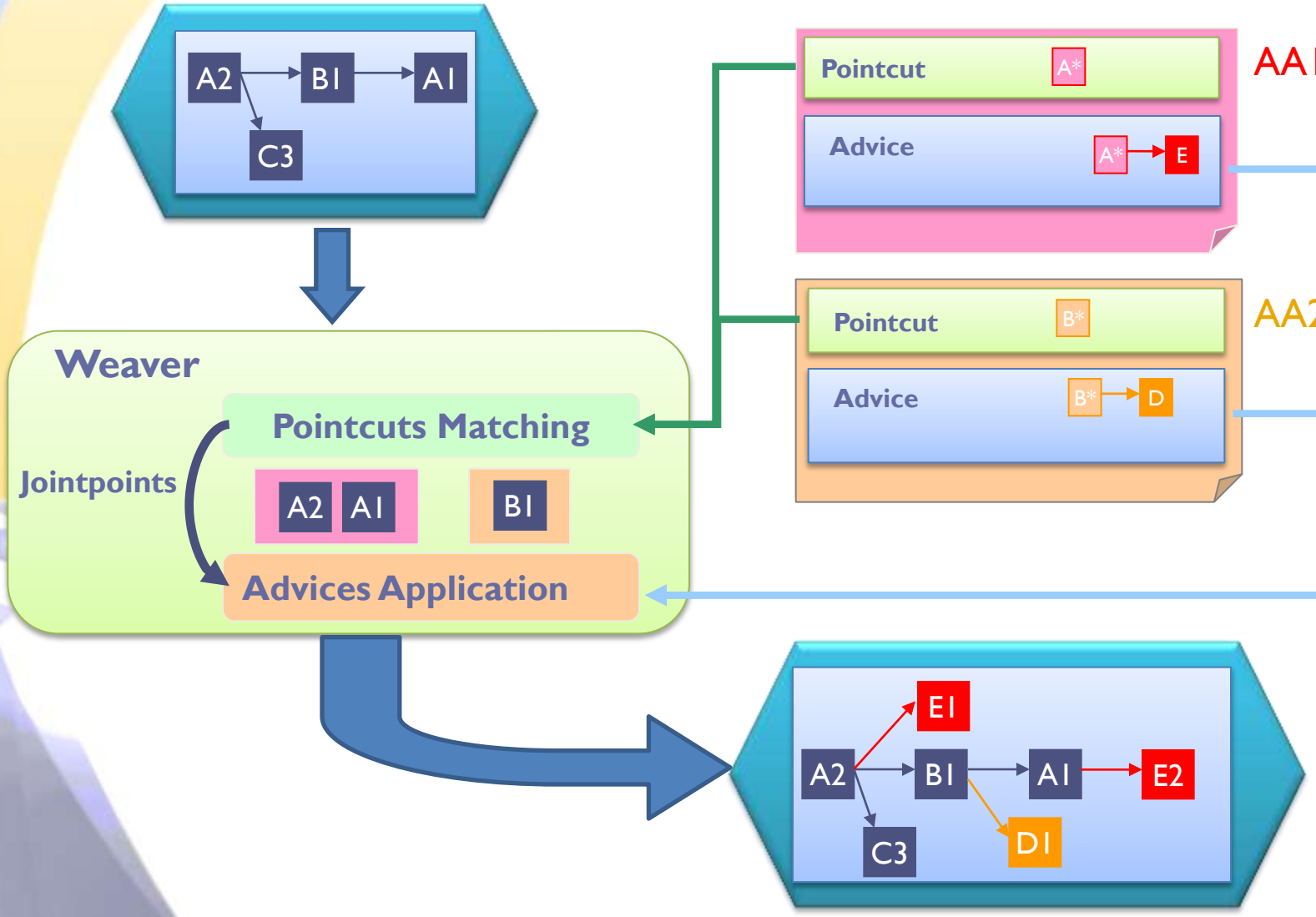


AA2

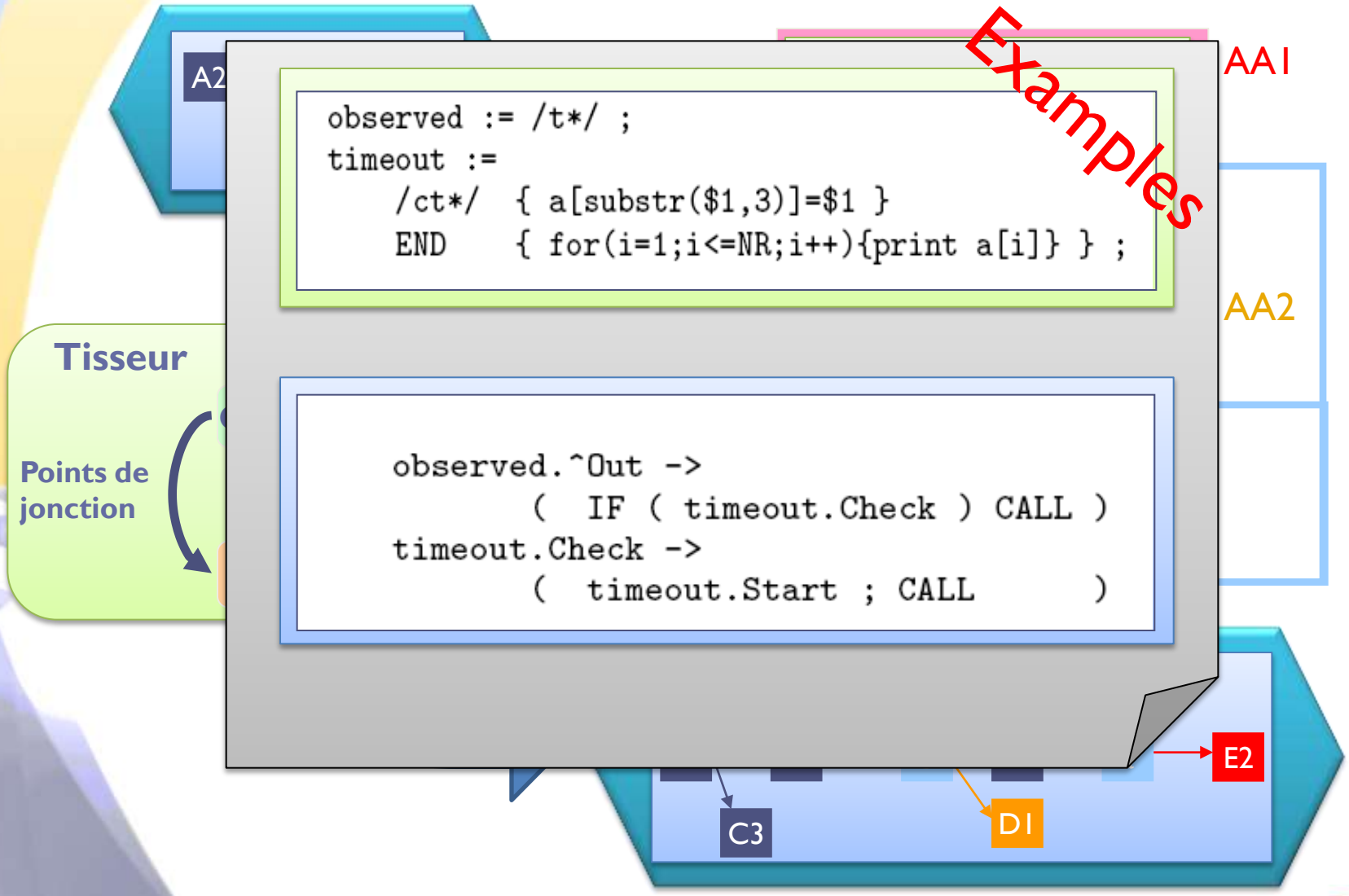
II.3 Aspect of Assembly Principles



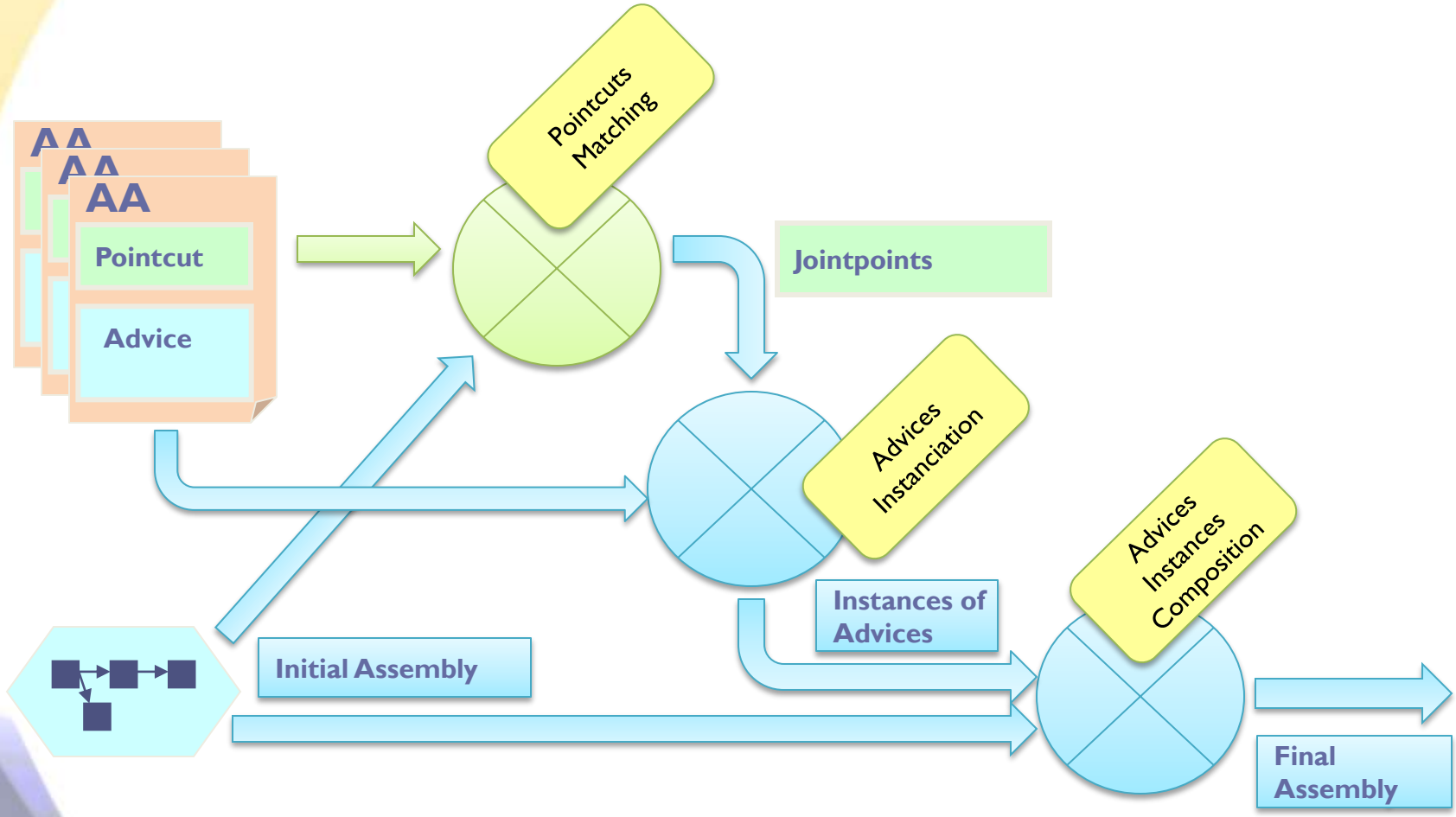
II.3 Aspect of Assembly Principles



II.3 Aspect of Assembly Principles

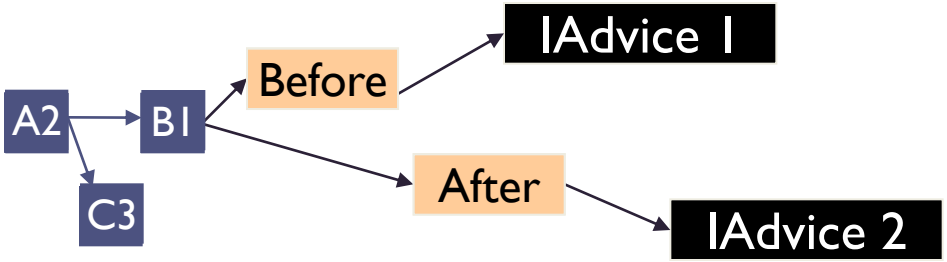


II.3 Internal Architecture of the AA Weaver

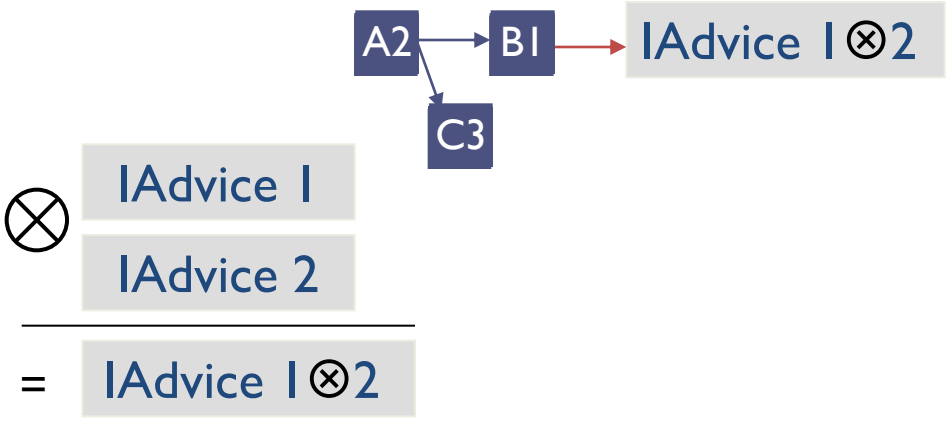


II.3 AA: I-Advices, Composition, and Conflicts

- External Composition :
  - I-Advices are « blackbox »
  - I-Advices are scheduled
  - Before, After, Around …



- Internal Composition with Merge :
  - I-Advice are « whitebox »
  - Conflicted I-Advices can be merged according to a specific logic and its properties
  - ex. ISL, ISL4WComp, BSL, …



II.3 AA: EXAMPLE OF SPECIFIC MERGING LOGIC AND ITS PROPERTIES

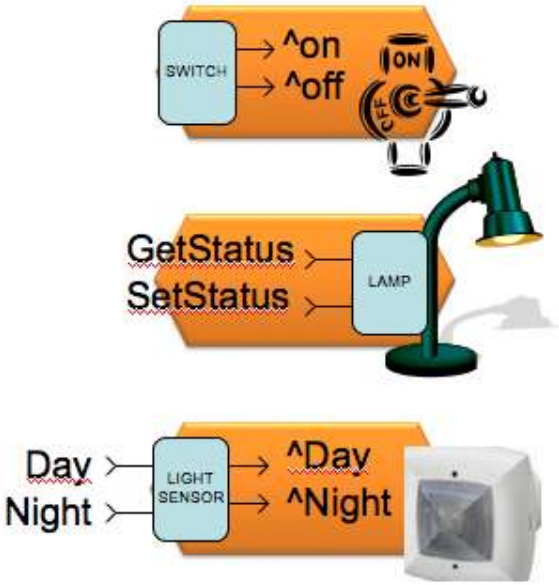
- Merging logic is based on rules modified according to the Advice language
- Example of prooved properties in the composition / merging logic :

Commutativity : $AA0 \otimes AA1 = AA0 \otimes AA1$
Associativity : $(AA0 \otimes AA1) \otimes AA2 = AA0 \otimes (AA1 \otimes AA2)$
Idempotence : $AA0 \otimes AA0 = AA0$

- Weaving mecanism became « Symmetric »
- Designer can apply a set of AA without caring of the their order.

An example

- When my lamp comes to my house, I will connect it to the switch 14
- // Description of needed components
  l := /lamp/
  sw := /switch14/
- // AA description
  // point cup -> advice
  sw.^on -> l.SetStatus (true)
  sw.^false -> l.SetStatus (false)



- Once the light sensor says it is day, I can not light the lamp
- // Description of needed components
  l := /lamp/
  ls := /light sensor/

- // AA description
  // point cup -> advice
  l.SetStatus (b) -> if (ls.days) then _call (false)
  else _call (b)

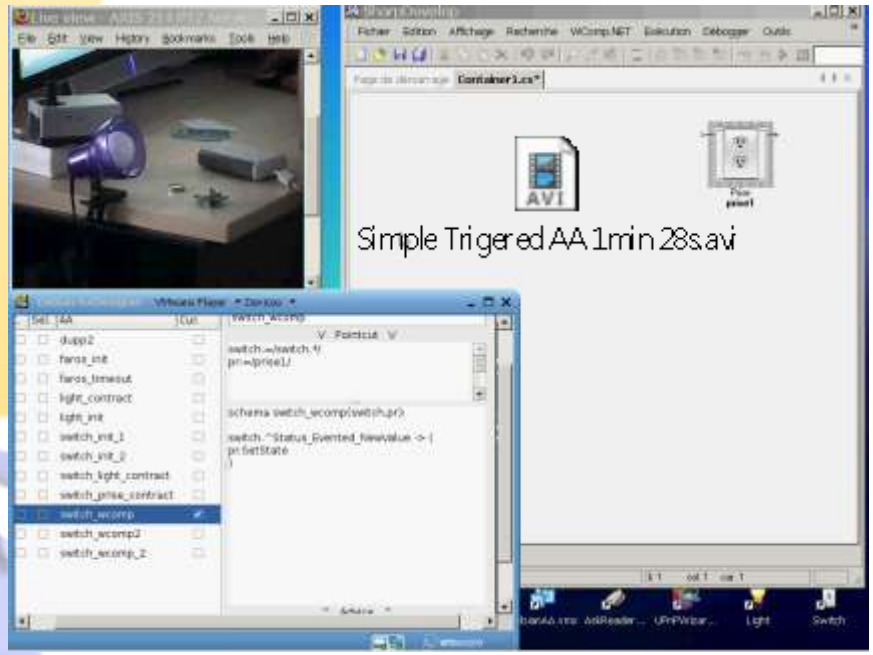
- Once the light sensor says it is daytime, the light goes off and when it is dark, it lights
  - All is doing automatically
- // Description of needed components
  l := /lamp/
  ls := /light sensor/

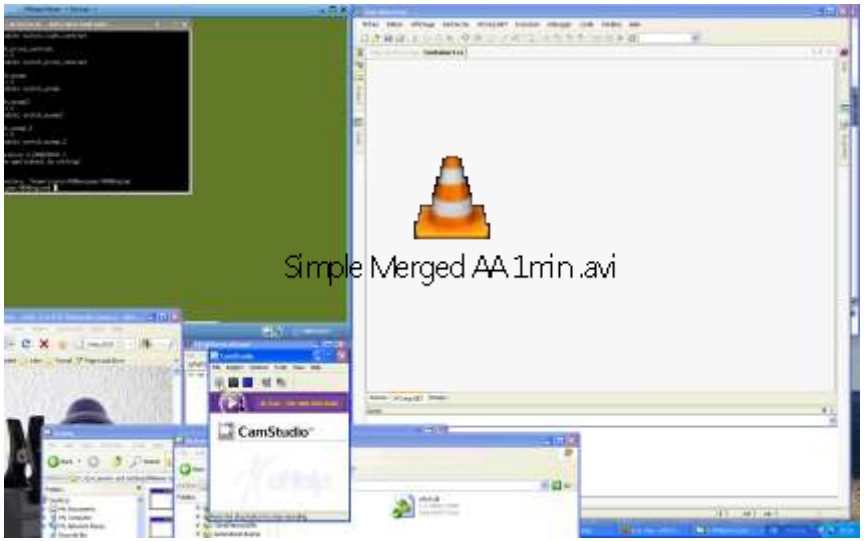
- // AA description
  // point cup -> advice
  ls.^day -> l.SetStatus (off)
  ls.^night -> l.SetStatus (on)

AA, IS TOO STRONG

- Now I have 3 AAs in my cart
  - My first connects the switch to my lamp
  - My second, keeps me turning on the light when it is day (but do not extinguished if it is on)
  - My third, off my lamp when the sun goes up (but does not prevent me from turning it on)
- What is the result?
  - I've home environmentally sustainable "compliant"
    - I can turning on my light only the night
    - My light is automatically turn of when the day arise

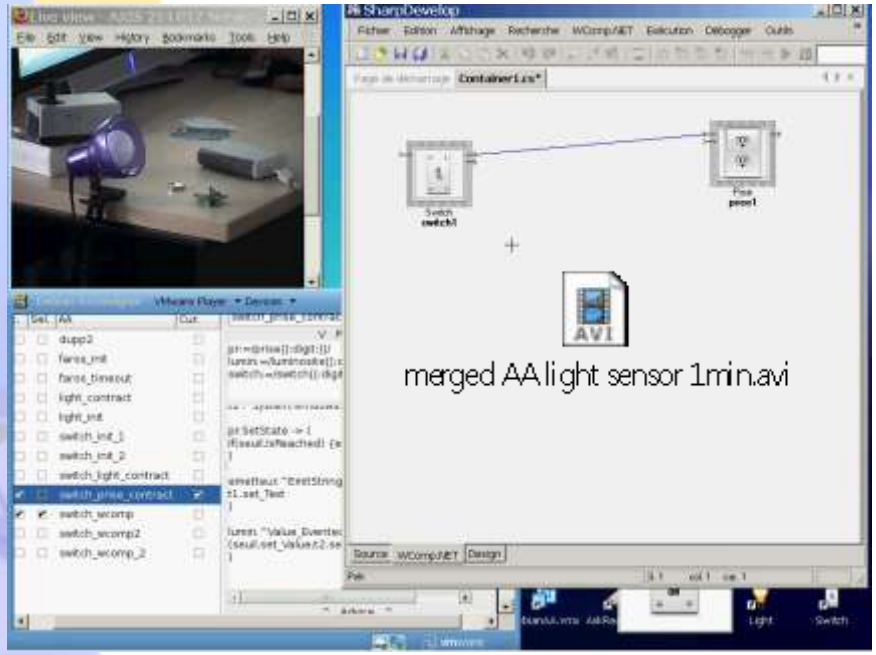
II.3 Simple Demo: AA in WComp (3 min)

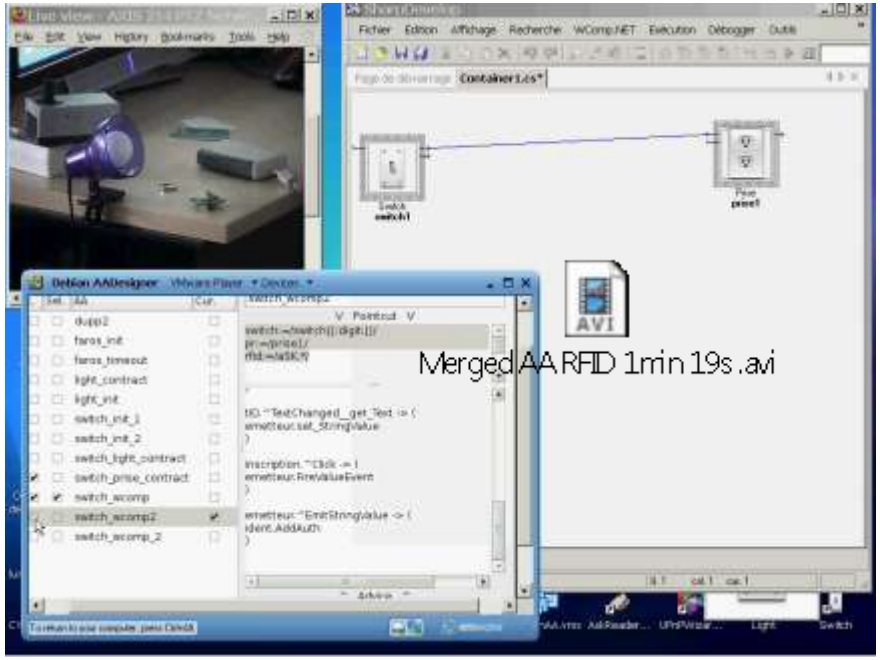




Our experimental platform is as a Sharpdevelop Addon on .Net Framework.
http://rainbow.i3s.unice.fr/wikiwcomp/

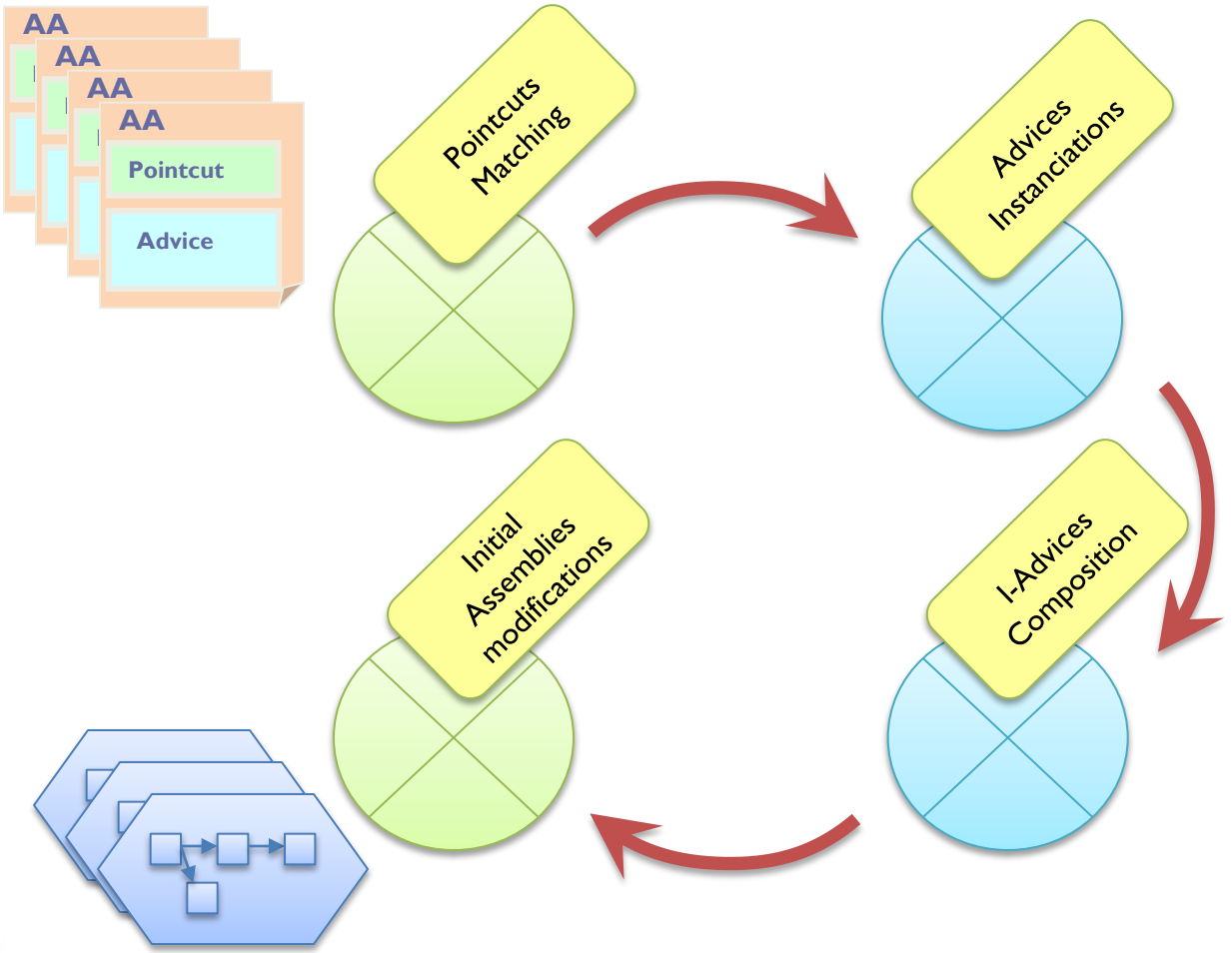
II.3 Other Demo: AA in WComp (2 min)





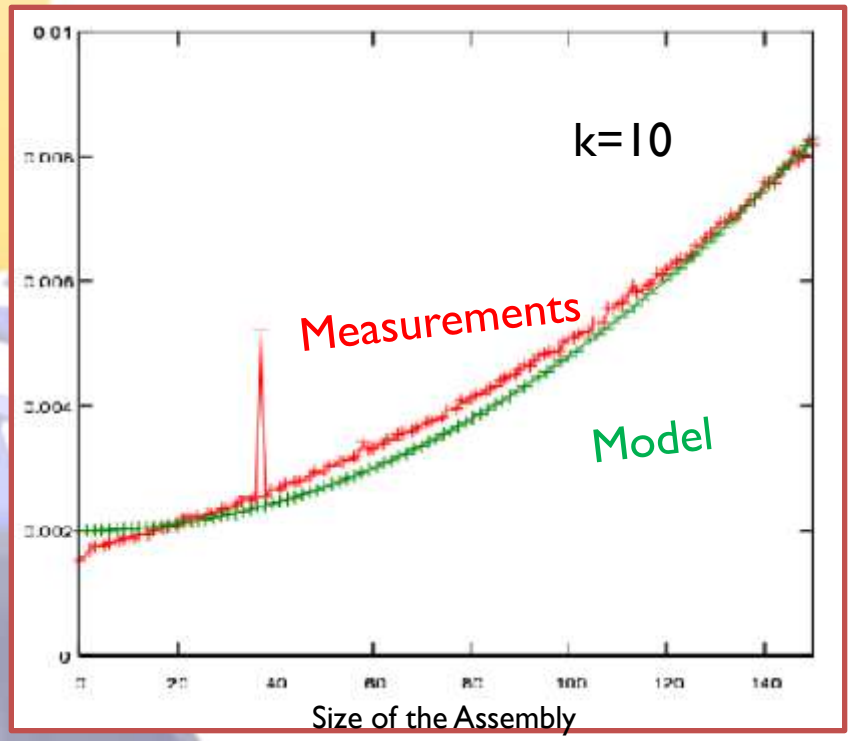
Our experimental platform is as a Sharpdevelop Addon on .Net Framework.
http://rainbow.i3s.unice.fr/wikiwcomp/

II.4 Reactivity, response time and Weaving Cycle



II.4 Cost of the weaving cycle: Pointcut Matching

$$D = a_1 . \sum_{i=1}^{k} (\delta_i + 1) . c^2 + a_2$$



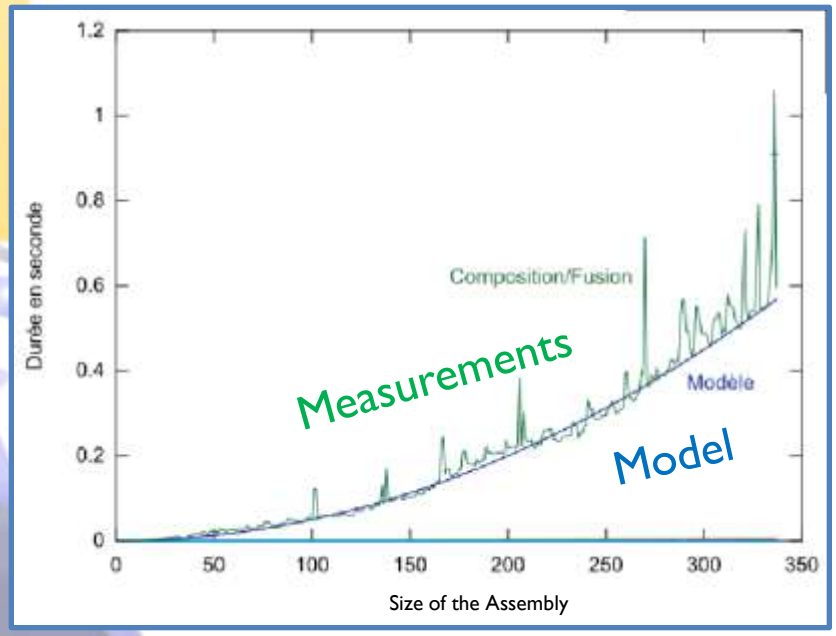
- Parameter indentification
  - δ_i : nb of applications of the advice *i*
  - *c* : nb of components

$$a_1 = 280 .10^{-9}$$
$$a_2 = 2.10^{-3}$$

- **Depend on the size of the initial assembly and the number of AA**

II.4 Cost of the weaving cycle: Weaving and modification

$$K = b.n^0.\sum_{i=1}^{N} n^i \left(1 + p_i.C\left(g_0, g_i\right)\right)$$



- Parameter Identification

  p_i : proba fusion

  C : merging cost

  n : number of rules

  N : number of I-advices

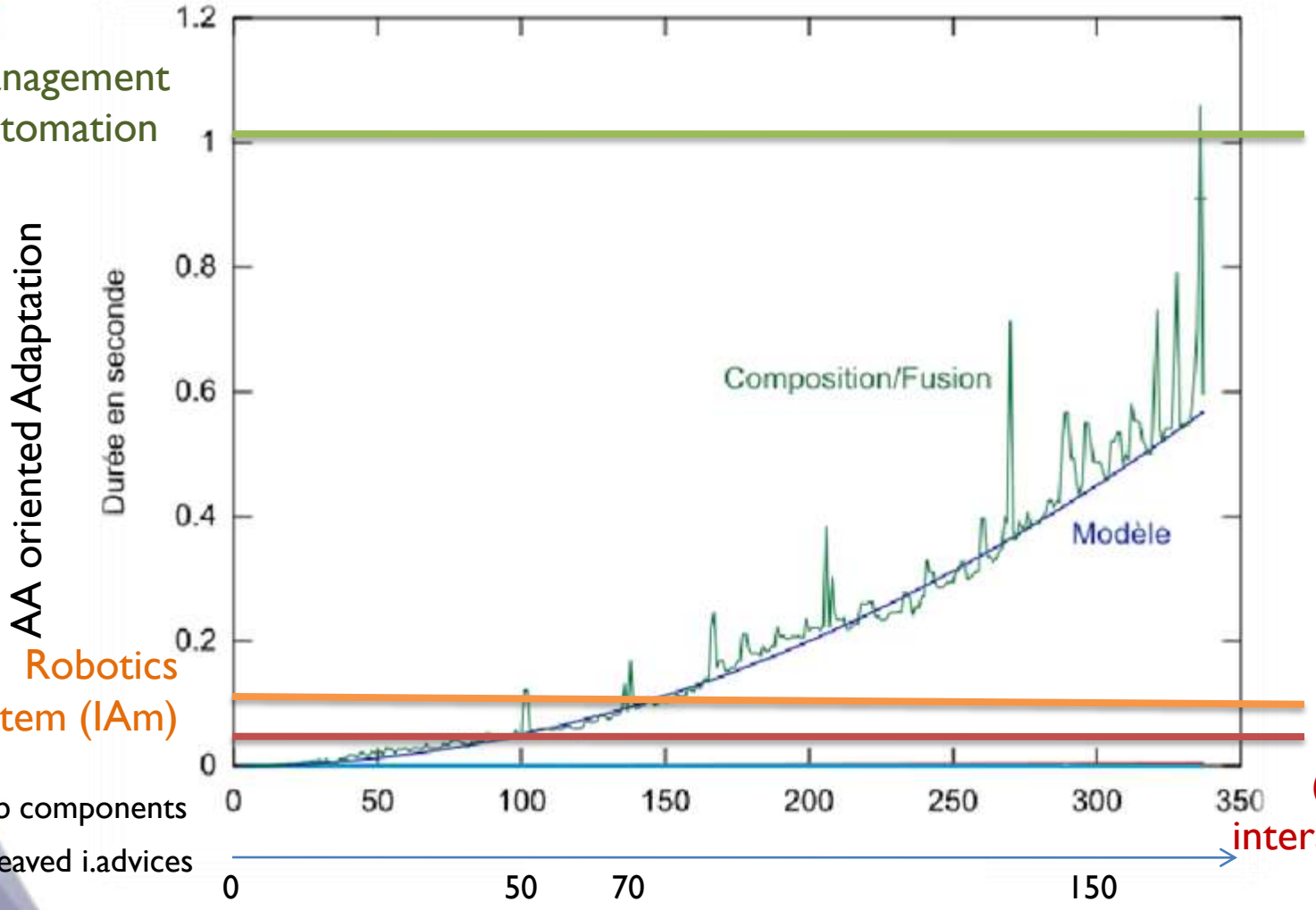
$$b = 2{,}6.10^{-6}$$

- **Only depend on the number of weaved AA**

↗ Nb components => ↗ Nb weaved AA

II.4 Reactivity and Adaptation: Experiments and Results

Energy management (Home Automation System)

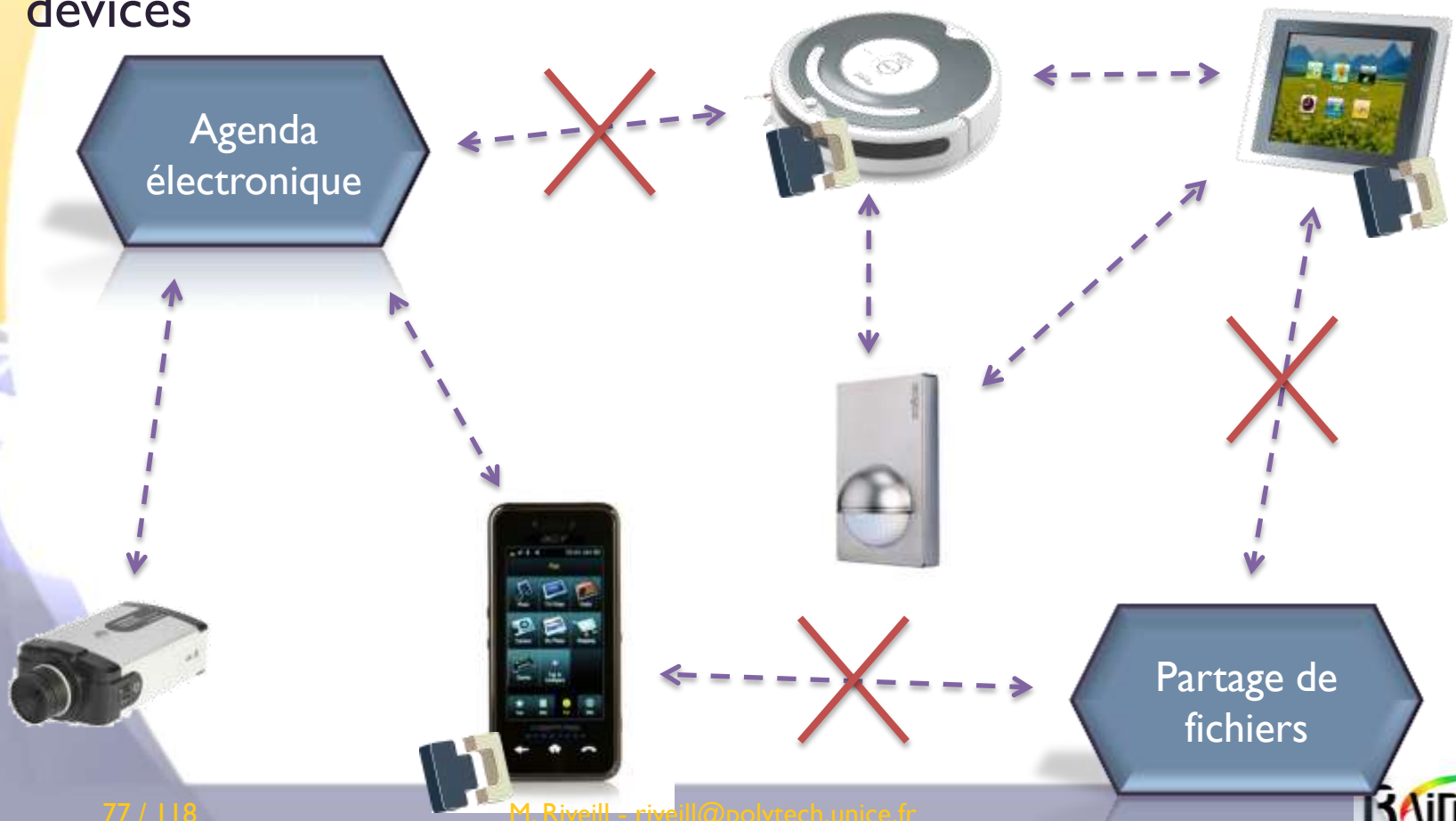


III. Control of interaction from the context

- III.1 Contexte aware computing
- III.2 A model
- III.3 Wcomp extended
- III.4 Illustration

CONTROL OF INTERACTION FROM THE CONTEXT

- Based on hardware caracteristics to identify devices
- Based on context, to allow or not interaction between devices

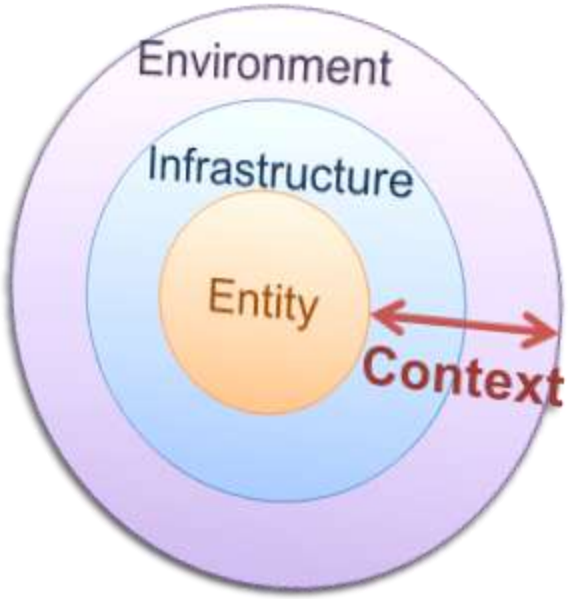


Context
General approaches
How to control interaction with context

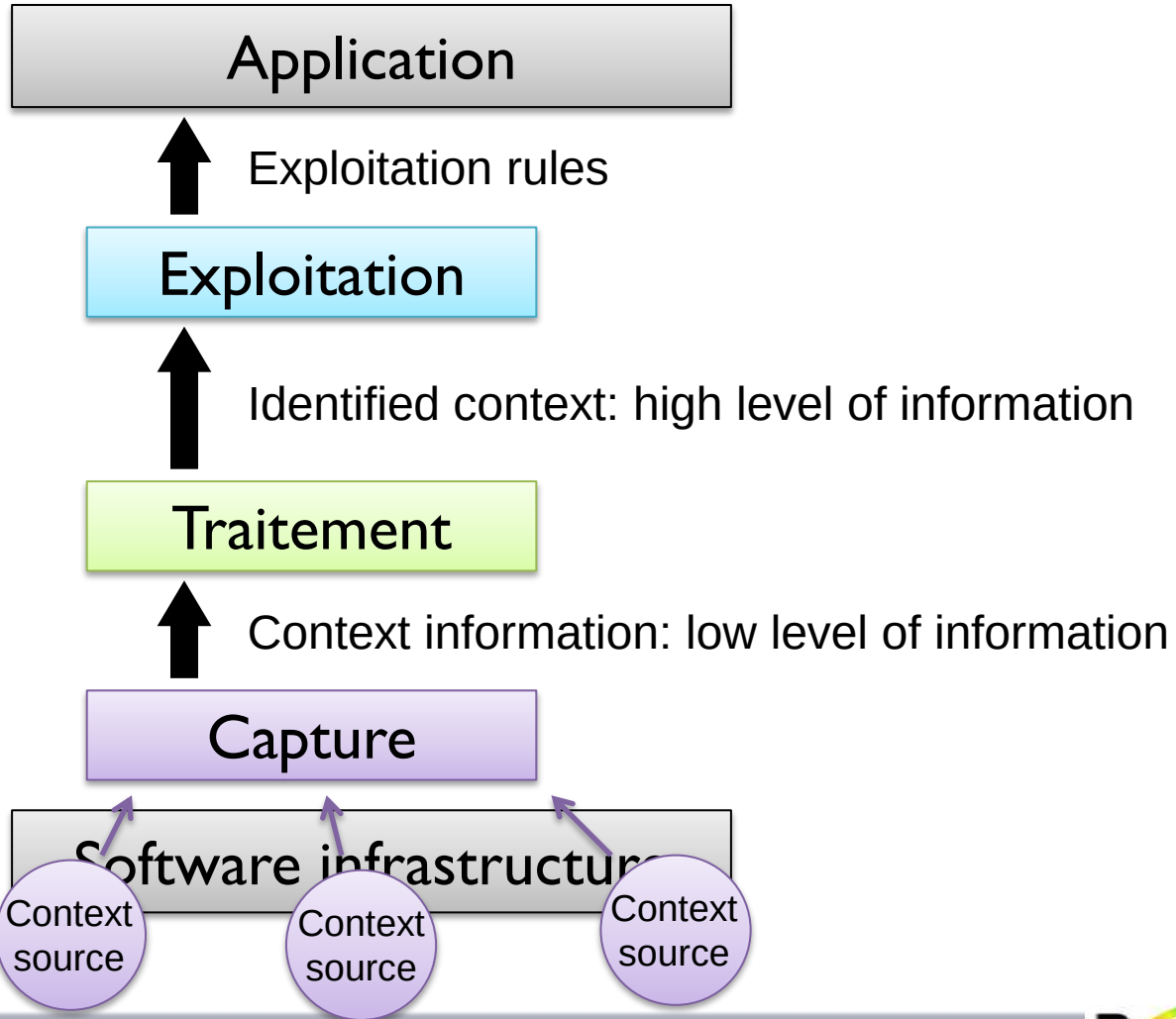
III.1 CONTEXT AWARE COMPUTING

Context

- A lot of definitions
- Context for us, are informations
  - Relevant for the entity
  - Concerning this entity
  - From the environment or the infrastructure



How to collect context
Functionnal decomposition

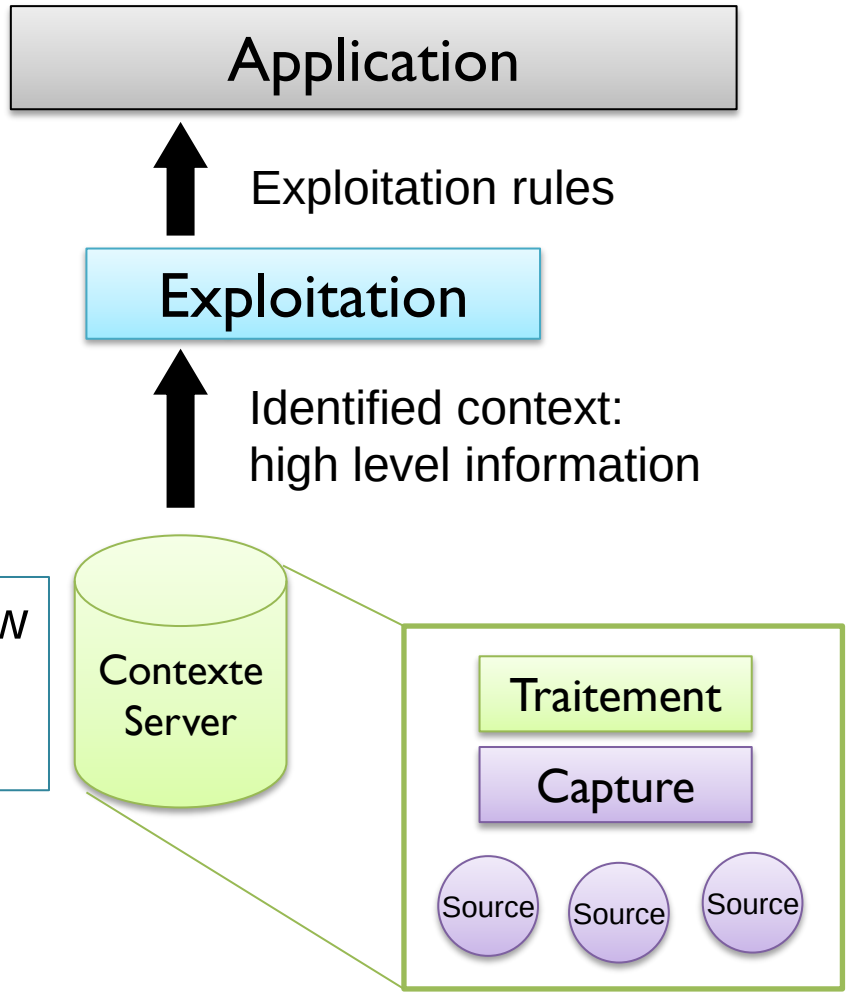


Context observation

First approach « Logic »

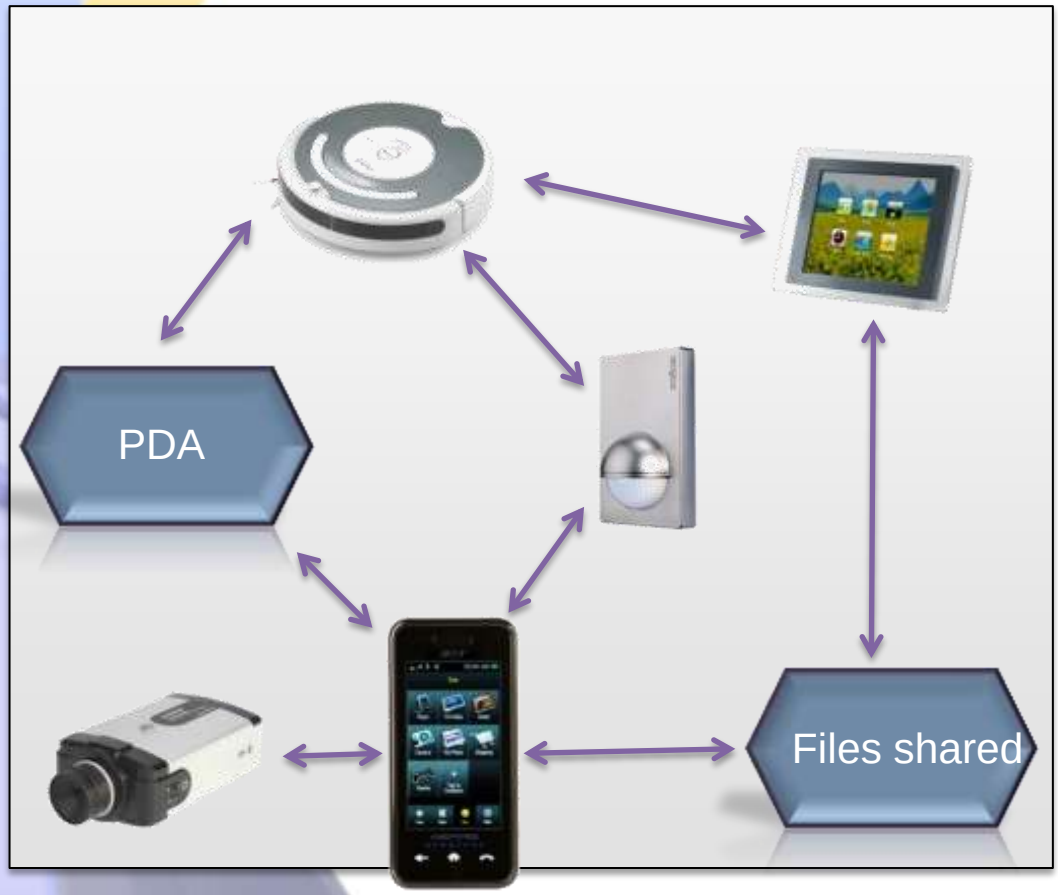
- Modelise the World for all applications
  - Dealing with all the contextual information
  - Server context / platform contextual services
- Exploitation rules simple

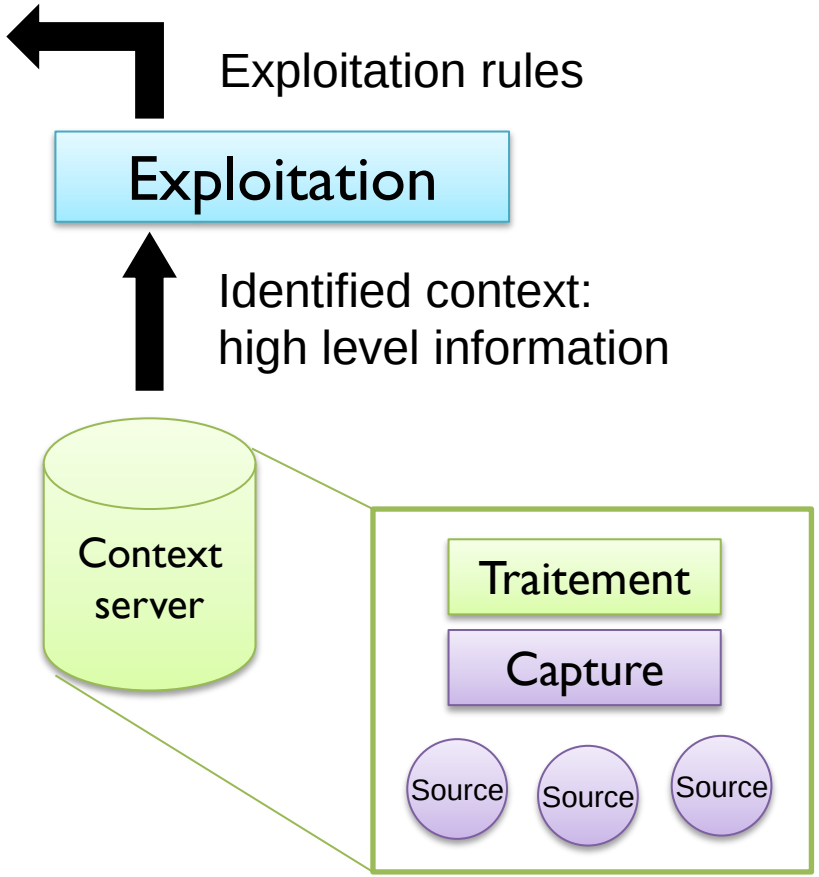
« When it is dark in the room, allow interaction with components of living"



GENERALLY USED FOR …

- Application architecture modification
  - Interaction are more created than controled

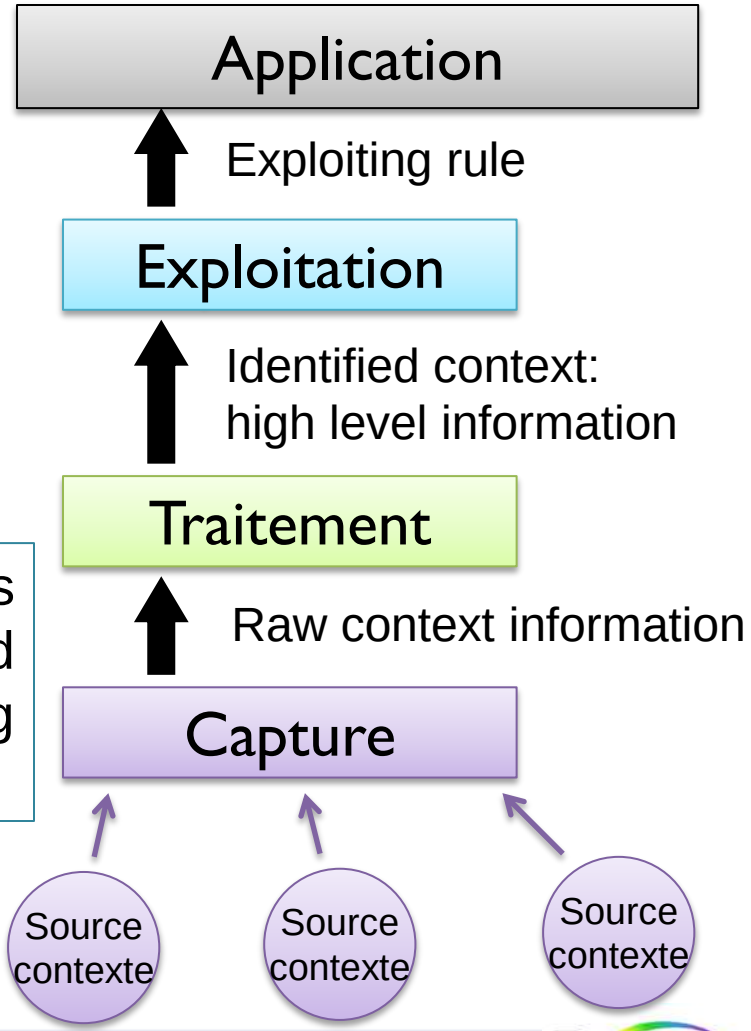




SECOND APPROACH « PHYSICAL »

- Observer / Capture are linked to the application
  - Reduces treatment of context
  - The treatment is application-specific
  - less information, but relevant
- Complex exploiting rules

« If a brightness information is available and less to 300 lux and flaps are discovered, allowing interactions with them»



USED FOR 'CONSTRAINING' AN APPLICATION

Exploitation
Traitement
Capture

Exploitation
Traitement
Capture

Exploitation
Traitement
Capture

Exploitation
Traitement
Capture

Exploitation
Traitement
Capture

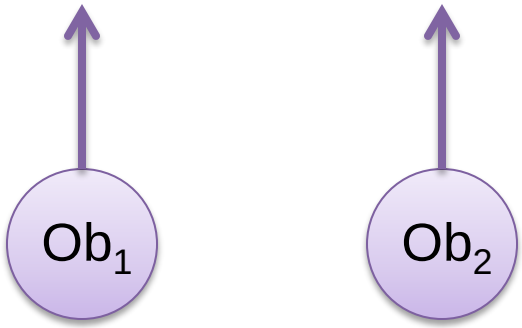
Exploitation
Traitement
Capture

Context observation and representation
Context traitement
Context aware exploitation rules
Access control list
Rules description

III.2 A MODEL FOR CONTEXT AWARE INTERACTION

CONTEXT OBSERVATION

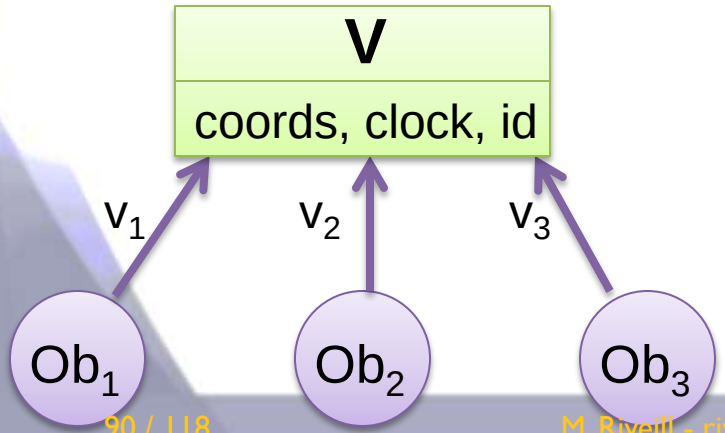
- Observers of context are software entities providing contextual information
- Each one defines:
  - Data type (real, string, boolean, …)
  - Semantic (temperature, pressure, id, …)
  - Attribut (localisation, authentification, …)
  - Value (26.8, "alice », …)



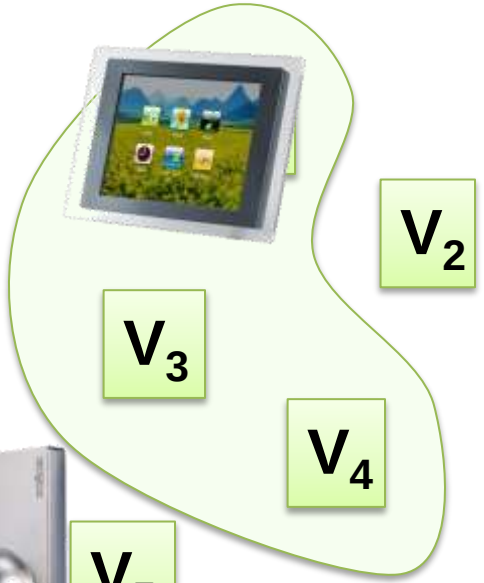
CONTEXT REPRESENTATION

- The context of an entity is defined by all observers associated with it
- Representation
  - Contextual N-uplet: $V = (v_0, v_1, \ldots, v_n)$

Exemple : $V \in$ (localisation, clock, id)
V = ({44.1, 7.02}, {12, 59, 23}, "alice")

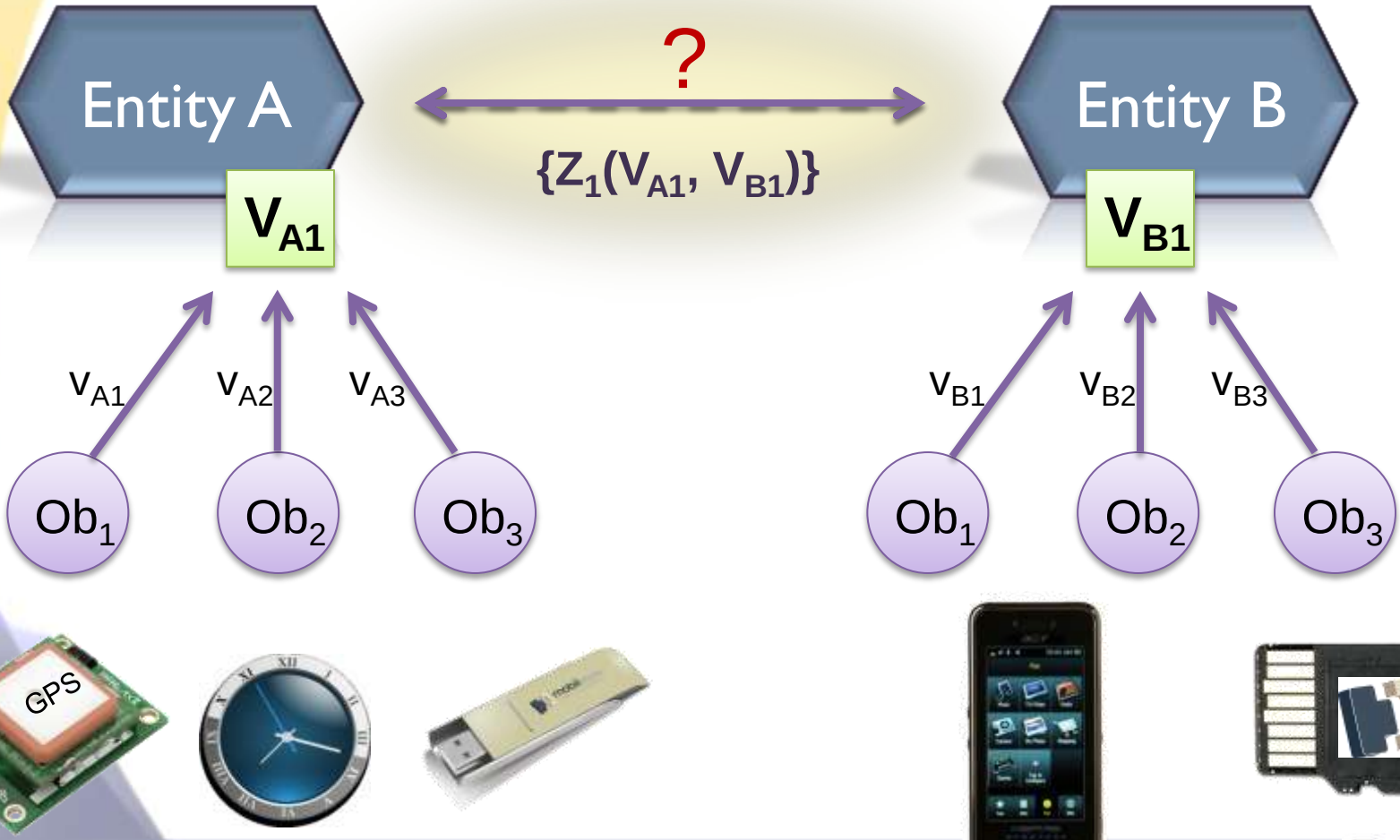


CONTEXT AND INTERACTION



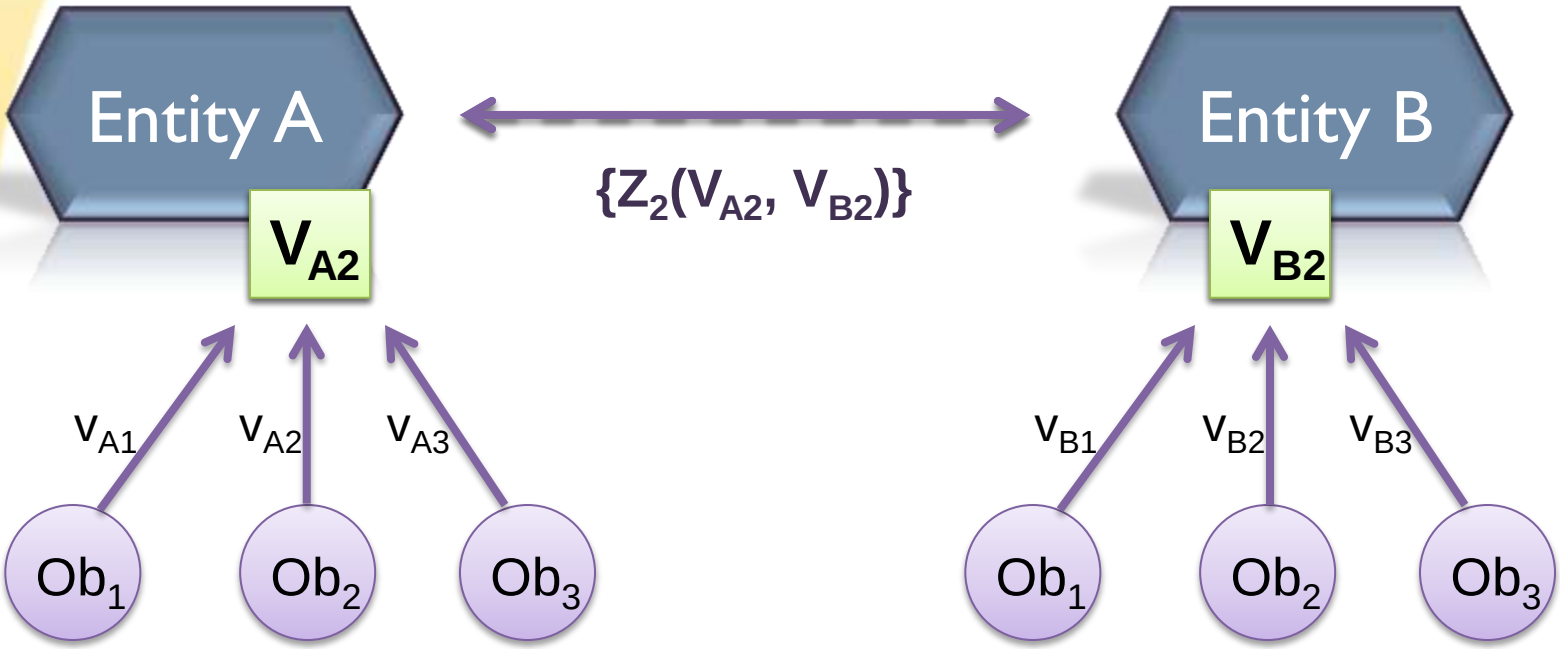
INTEACTION'S CONTROL RULES

<predicat Z_1, interaction$_{A \rightarrow B}$, effect (ALLOW)>



INTERACTION'S CONTROL RULES

<predicat Z_1, interaction$_{A \rightarrow B}$, effect (ALLOW)>

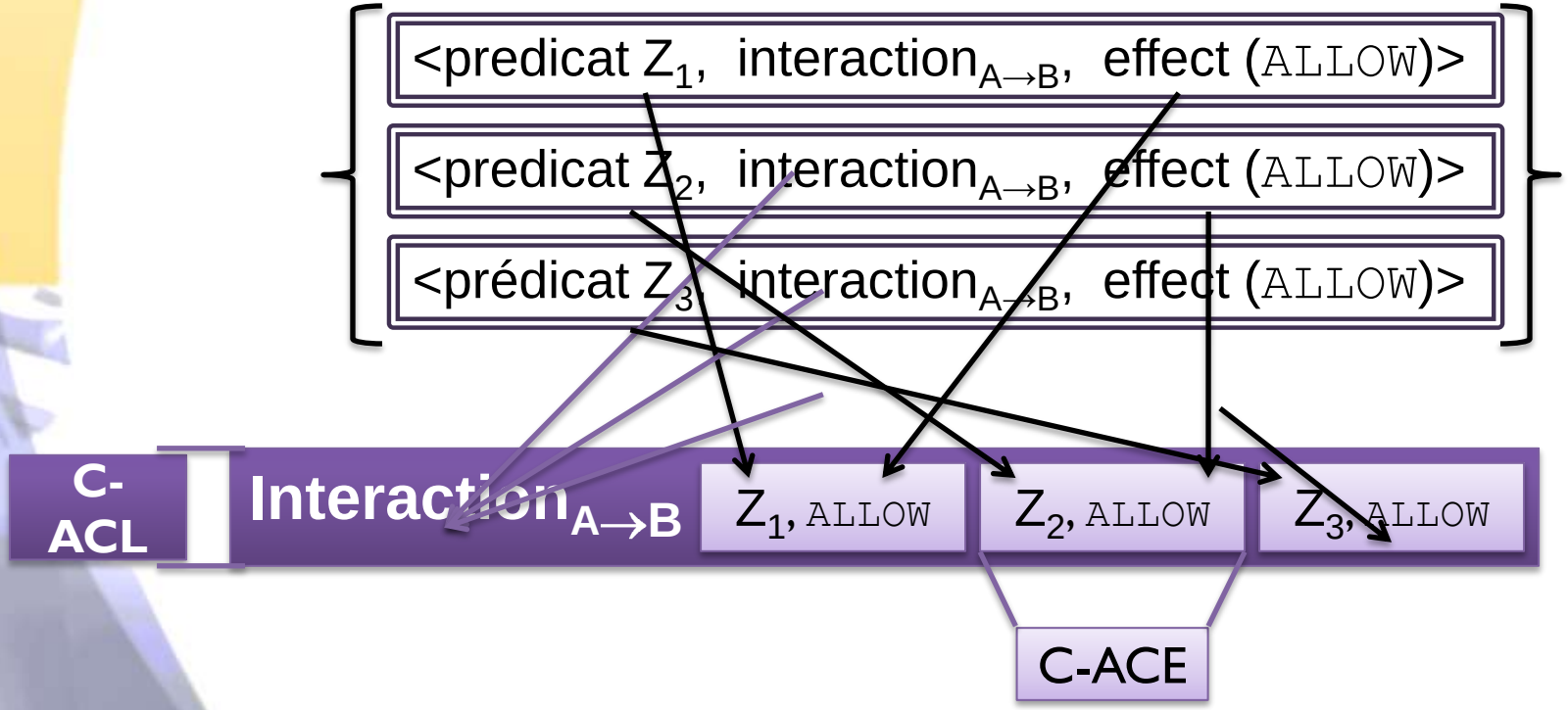






RULES ORGANIZATION: C-ACL

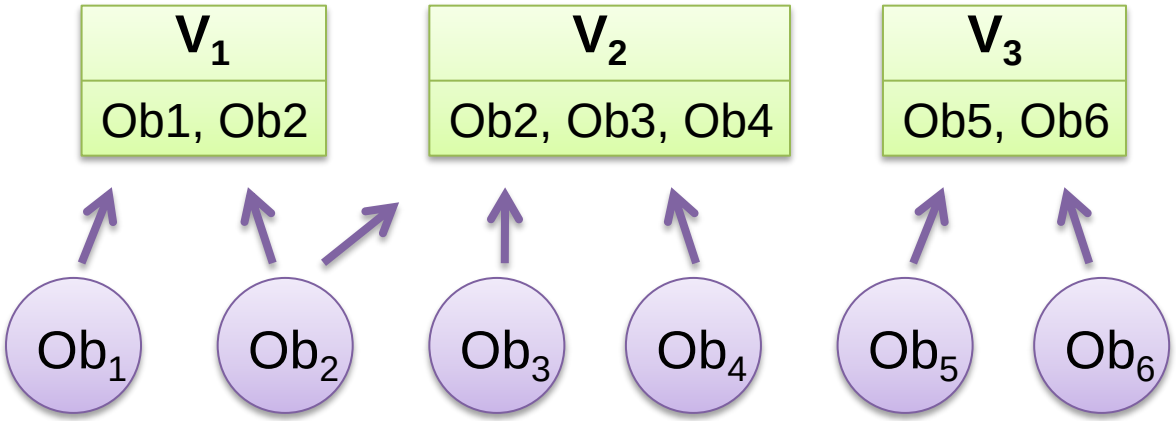
- For an interaction, several rules are defined
  - Each one with a *predicat* Z_i and an *effect*



DYNAMIC ACTIVATION OF THE C-ACE

- C-ACE are reactivly activated when each context are observed

- Context
- Observators

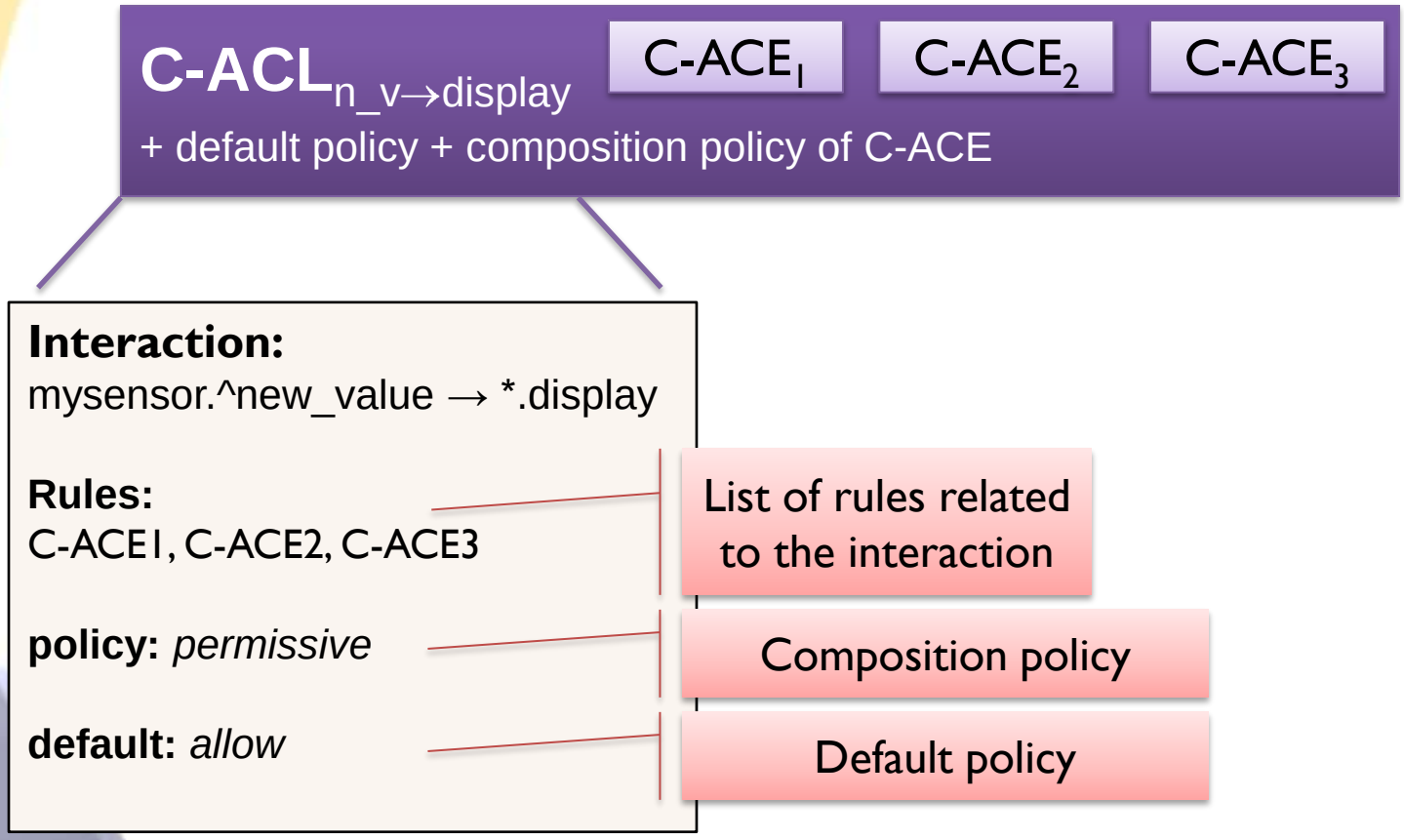


Evaluation of list of rules

- No rule can be activated: default policy
  - ALLOW
  - DENY
- Several rules activated: composition rules
  - Restrictive policy: ET ($\wedge$)
    - Decision rules in effect `ALLOW` : $Z_1 \wedge Z_2 \wedge \ldots \wedge Z_3$
    - Decision rules in effect `DENY` : $\neg(Z_1 \wedge Z_2 \wedge \ldots \wedge Z_n)$
  - Permissive policy: OU ($\vee$)
    - Decision rules in effect `ALLOW` : $Z_1 \vee Z_2 \vee \ldots \vee Z_n$
    - Decision rules in effect `DENY` : $\neg(Z_1 \vee Z_2 \vee \ldots \vee Z_n)$

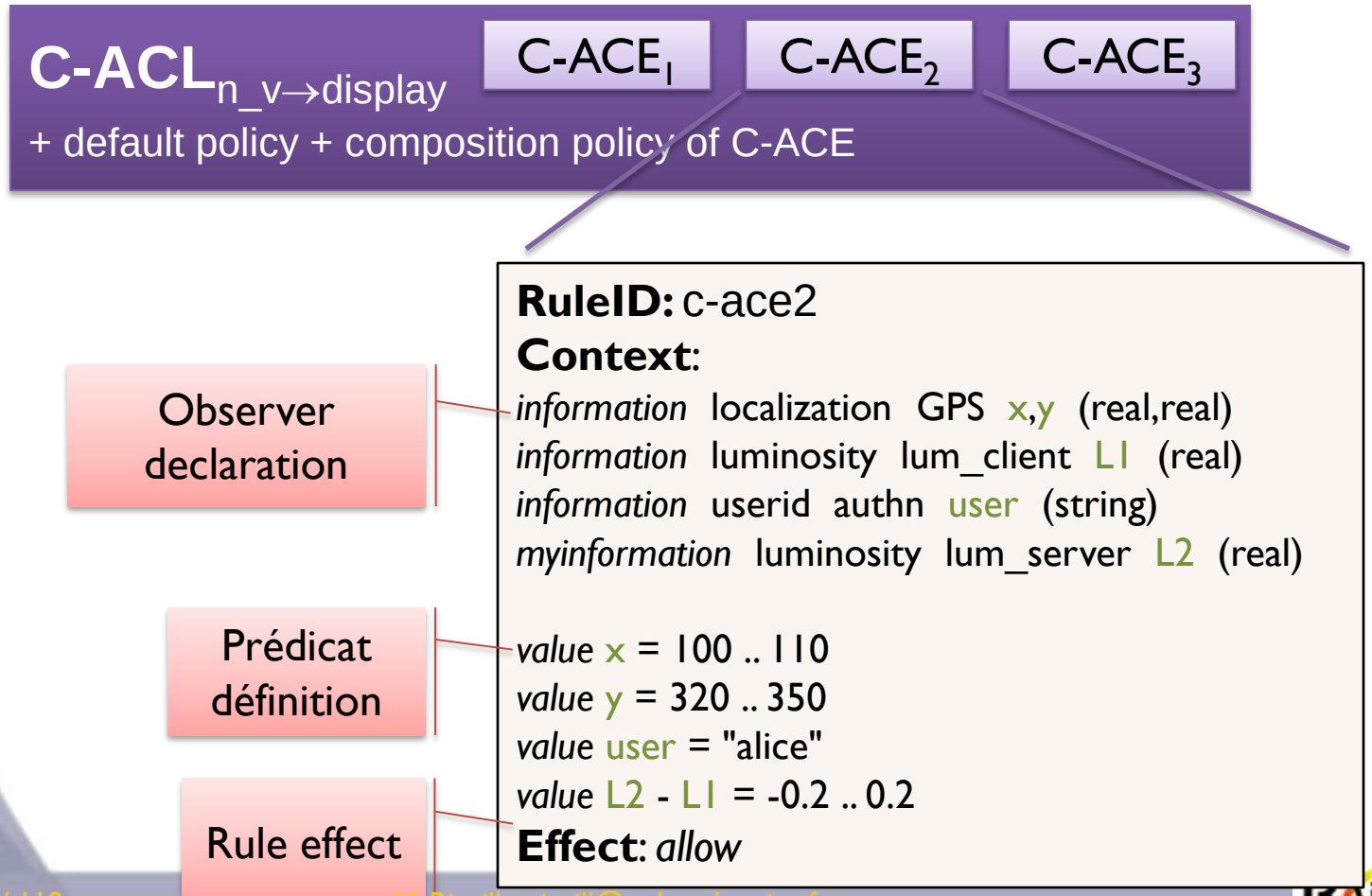
RULES EXPRESSION

- Expression of the access control list



RULES EXPRESSION

- Expression of a list entry



From control rules to aspect assemblage
Operator for composition politics

III.3 IMPLEMENTATION ON TOP OF WCOMP

TRANSLATION IN TO AA

- Automatic conversion of C-ACE in AA

C-ACE2

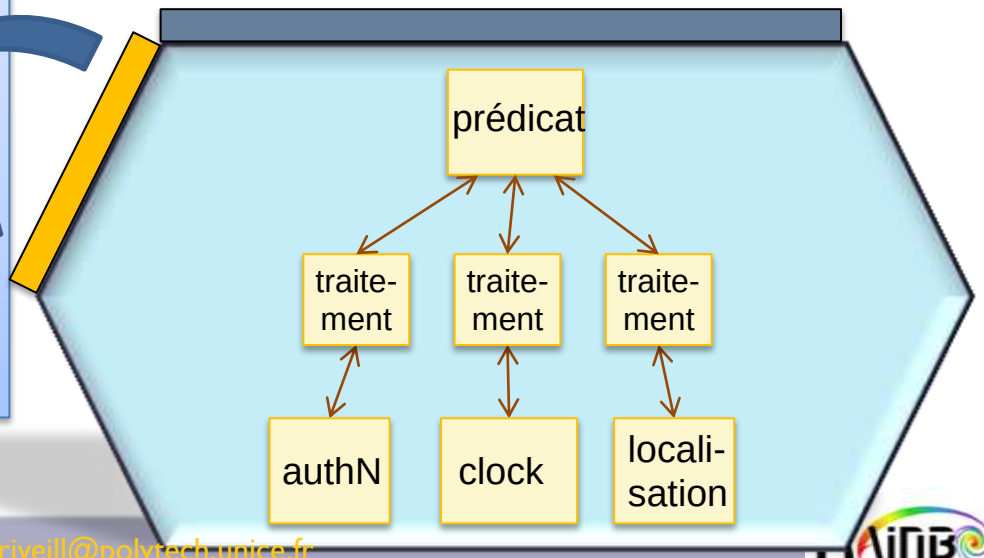
RuleID: c-ace2
Context:
information is_auth clientA_auth authn (string)
Information hour clock clock (int)
information inRoom clientA_localization room (string)
value authn = clientA
value hour = 8 .. 21
value room = 314
Effect: *allow*

```
// Pointcut
obs_authn = clientA_auth.^is_auth
obs_hour = clock.^minutes
obs_room = clientA_localization.^inRoom

// Advices
advice c-ace2(obs_authn, obs_hour, obs_room) :
ipred_authn : Context.Eval.String (ref="clientA")
ipred_hour : Context.Eval.IntInt (between="8,21")
ipred_loc : Context.Eval.String (ref="314")
pred_authn_hour_room : Logic.AND (nb_inp=3)

obs_authn → ( ipred_authn.inputString )
obs_hour → ( ipred_hour.inputIntInt )
obs_room → ( ipred_room.inputString )

ipred_authn.^Out → ( pred_authn_hour_room.In1 )
ipred_hour.^Out → ( pred_authn_hour_room.In2 )
ipred_room.^Out → ( pred_authn_hour_room.In3 )
```



AA COMPOSITION

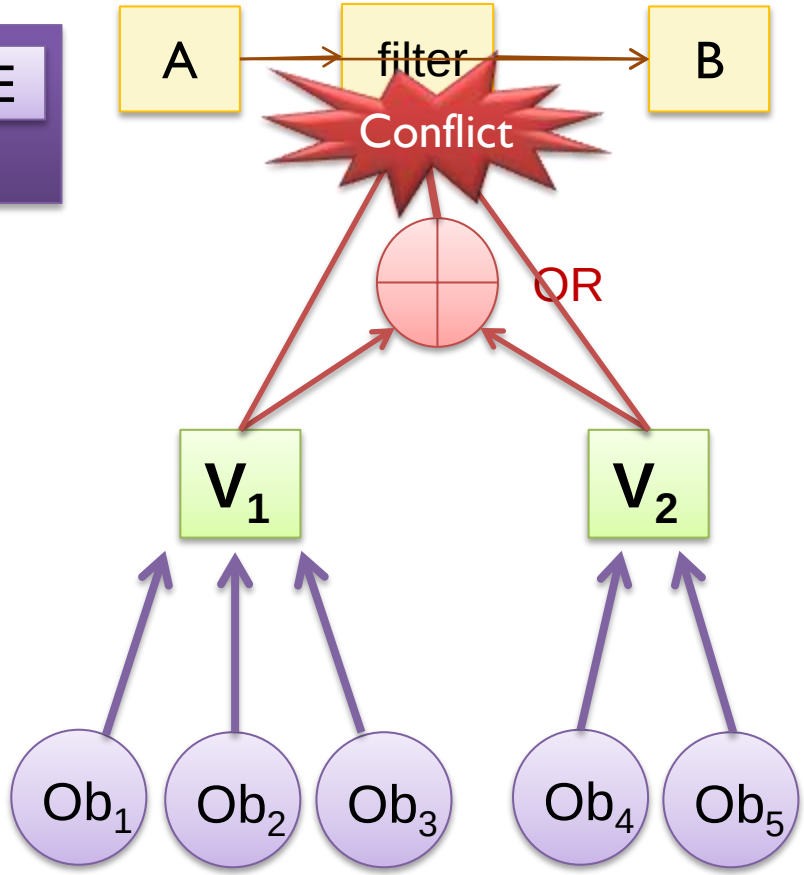
C-ACL
Interaction$_{A \rightarrow B}$
+Default politic + Composition politics of C-ACE

C-ACE | C-ACE | C-ACE

Transformation

Pointcut
clock, localisation,
authN, interactionAB
Advice
new predicat
clock → predicat
localisation → predicat
authN → predicat
filtrer_OR(interactionAB)

Pointcut
authN, presence,
interactionAB
Advice
new predicat
authN → predicat
presence → predicat
filter_OR(interactionAB)



THREE COMPOSITION OPERATORS

- Operators of semantics known to the weaver
  - *filter_OR* *filter_AND* *default_filter*
- Advices fusion rules

$$default\_filter(l) \oplus e \rightarrow filter\_OR(l) \implies e \rightarrow filter\_OR(l)$$
$$default\_filter(l) \oplus e \rightarrow filter\_AND(l) \implies e \rightarrow filter\_AND(l)$$

$$\oplus \begin{cases} e_1 \rightarrow filter\_OR(l) \\ e_2 \rightarrow filter\_OR(l) \end{cases} \implies e_1 \vee e_2 \rightarrow filter\_OR(l)$$

$$\oplus \begin{cases} e_1 \rightarrow filter\_AND(l) \\ e_2 \rightarrow filter\_AND(l) \end{cases} \implies e_1 \wedge e_2 \rightarrow filter\_AND(l)$$

$$\oplus \begin{cases} e_1 \rightarrow filter\_OR(l) \\ e_2 \rightarrow filter\_AND(l) \end{cases} \implies (e_1 \rightarrow filter\_OR(l)) \wedge (e_2 \rightarrow filter\_AND(l))$$

Scenario
Video

III.4 ILLUSTRATION

SCENARIO

- Interaction targeted
  - Ultra-Mobile Computer
    - The client user mail
  - Access to Information System
    - Messaging internal company
- Context included

C-ACL
Interaction : info → mobile

Default Policy: `DENY`
Rules composition policy: `PERMISSIVE`

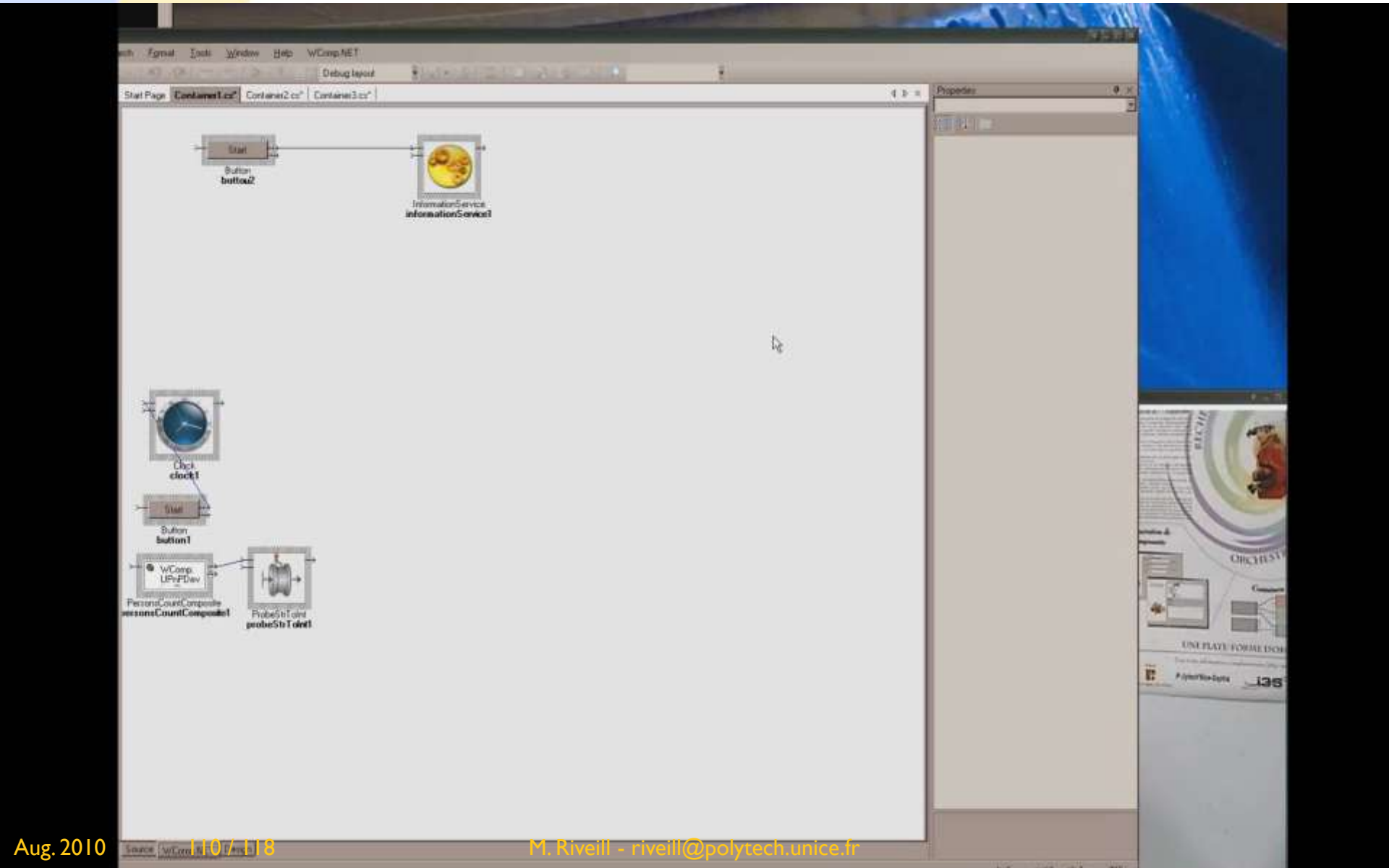
C-ACE 1

Clock = 8 .. 18
RFID = "0x7f21a533"
Pressure > 10

C-ACE 2

Luminosity > 200lux
RFID ≠ ""
NbPeople = 1

VIDEO


Start Page
Container1.cs*
Container2.cs*
Container3.cs*
Properties
Start
Button
button2
InformationService
informationService1
Clock
clock1
Start
Button
button1
WComp UPnPDev
PersonsCountComposite
personsCountComposite1
ProbeStrToInt
probeStrToInt1

IV CONCLUSION AND FUTURE WORKS

IV.1 Conclusion

	Reactive adaptation	Semantic adaptation	Multi-Domain adaptation
AA composition	✓	low	to improve

WComp	Event based interactions	Dynamic composition (at runtime)	Dynamic publication (at runtime)
LCA composition	✓	✓	
SLCA composition	✓	✓	✓

WComp	Event based interactions	Discovery of devices at runtime	Deal with appareance and dispareance of devices at runtime
Software Infrastructure	✓	✓	✓

IV.2 Future Works in WComp

- Multi-Domain weaving for AA to adapt Mobile Workers applications (Cf. CONTINUUM project of the French National Research Agency towards « Continuity of Service »)
- From AA to AOM (Aspect oriented Modeling) : a way to generalize Aspect to Adapt target architectures according to their model
- Semantic adaptation : Improving of Pointcut Matching algorithms from Ontology-Based Metadata and mapping between ontologies (Cf. CONTINUUM project of the French National Research Agency towards « Continuity of Service »)

THANK YOU FOR LISTENING ...

...Question ?

M. Riveill - riveill@polytech.unice.fr

QUESTIONS ?

Prof. Michel RIVEILL
http://rangiroa.polytech.unice.fr
riveill@unice.fr

IV.4 Historical References

- Mark Weiser. "The Computer for the 21th Century." Scientific American, September 1991.
- Mark Weiser. "Some computer science issues in ubiquitous computing." Communications of the ACM, 36(7):75-85, July 1993.
- Mark Weiser, John S. Brown. "The Coming Age of Calm Technology." 1996.
- M. Satyanarayanan. "Fundamental Challenges in Mobile Computing." Fifteenth ACM Symposium on Principles of Distributed Computing, May 1996.
- M. Satyanarayanan. "Pervasive Computing: Vision and Challenges." IEEE Personal Communications, August, 2001.

IV.5 WComp References

- J.-Y. Tigli, S. Lavirotte, G. Rey, V. Hourdin, D. Cheung, E. Callegari, M. Riveill "WComp middleware for ubiquitous computing: Aspects and composite event-based Web services" in the journal *Annals of Telecommunications*, Springer Paris editor, ISSN 0003-4347 (Print) 1958-9395 (Online), Vol. 64, No 3-4, March-April 2009
- Vincent Hourdin, Jean-Yves Tigli, Stéphane Lavirotte, Gaëtan Rey, Michel Riveill, "SLCA, Composite Services for Ubiquitous Computing", in International Conference on Mobile Technology, Applications and Systems, Sep 2008.
- Daniel Cheung-Foo-Wo, Jean-Yves Tigli, Stéphane Lavirotte et Michel Riveill. « Self-adaptation of event-driven component-oriented Middleware using Aspects of Assembly ». Dans 5th International Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC), California, USA, novembre 2007.