A Monitoring Approach for Dynamic Service-Oriented Architecture Systems

Y. Dan[1,3], **N. Stouls**[1], S. Frénot[1], and C. Colombo[2]

[1] *Université de Lyon, INRIA, INSA-Lyon, CITI, F-69621, France – Email: first.second@insa-lyon.fr*
[2] *Department of Computer Science, University of Malta – Email: first.second@um.edu.mt*
[3] *College of Computer Science Chongqing University, Chongqing, China*

UNIVERSITÉ DE LYON C I T I INRIA INSA LYON

Monitoring

What kind of monitoring

- Observation of an execution
- Check observed behavior against a property
- Observation granularity : external I/O of services

Our objectives

- Considering dynamics in SOA
- Focus observation technique

Future work

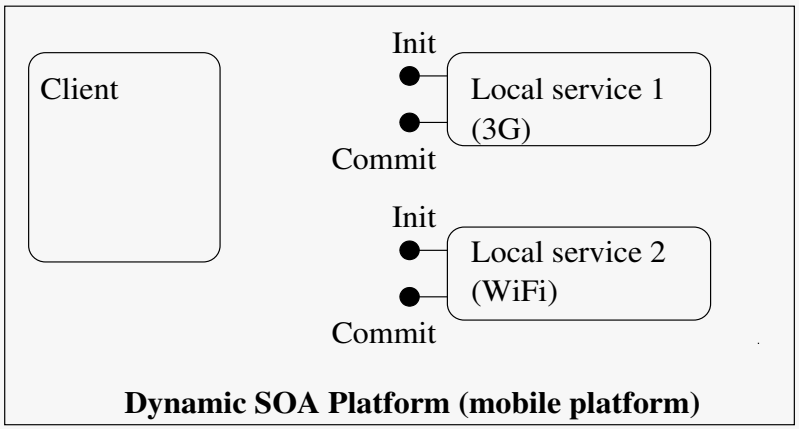
- Property description language expressiveness

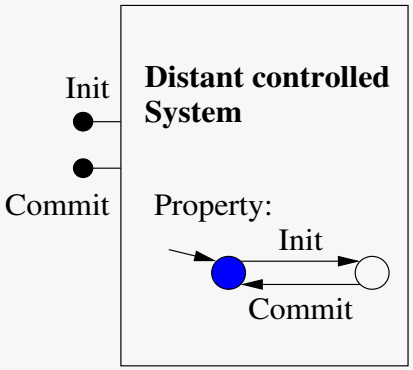
Dynamic SOA

Service Oriented Architectures

- Loosely coupled client-server through interfaces
  - A client request a service
  - System fulfils the interface with an implementation
  - Client uses the service
- But :
  - Used service can disappear
  - Used service can be substituted
  - Each invocation can be provided with a different service
- Our case study:
  - developed under OSGi.

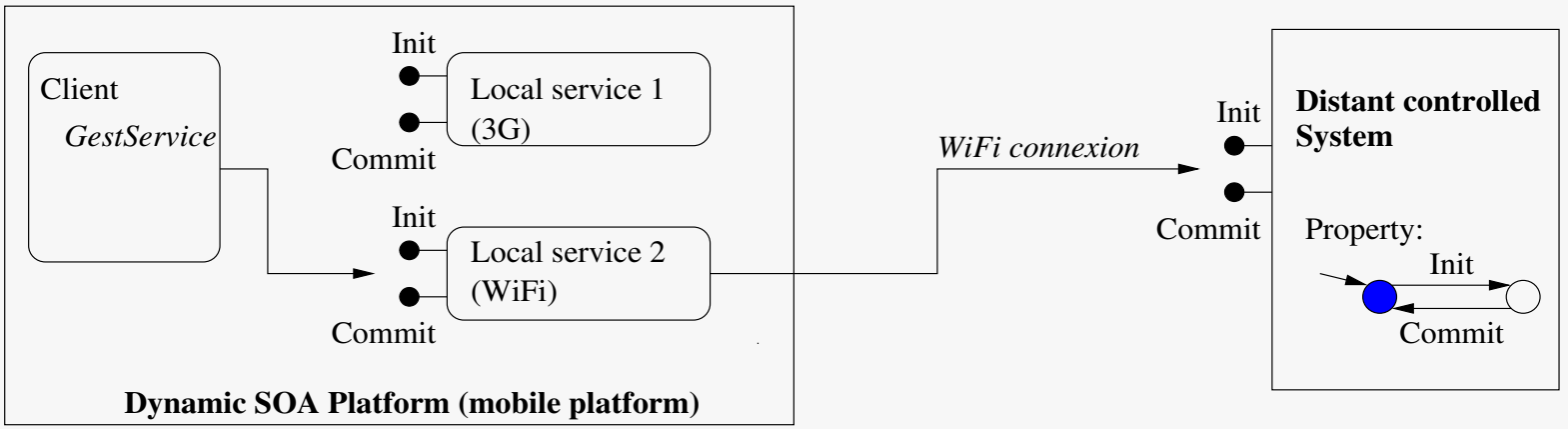
Our case study





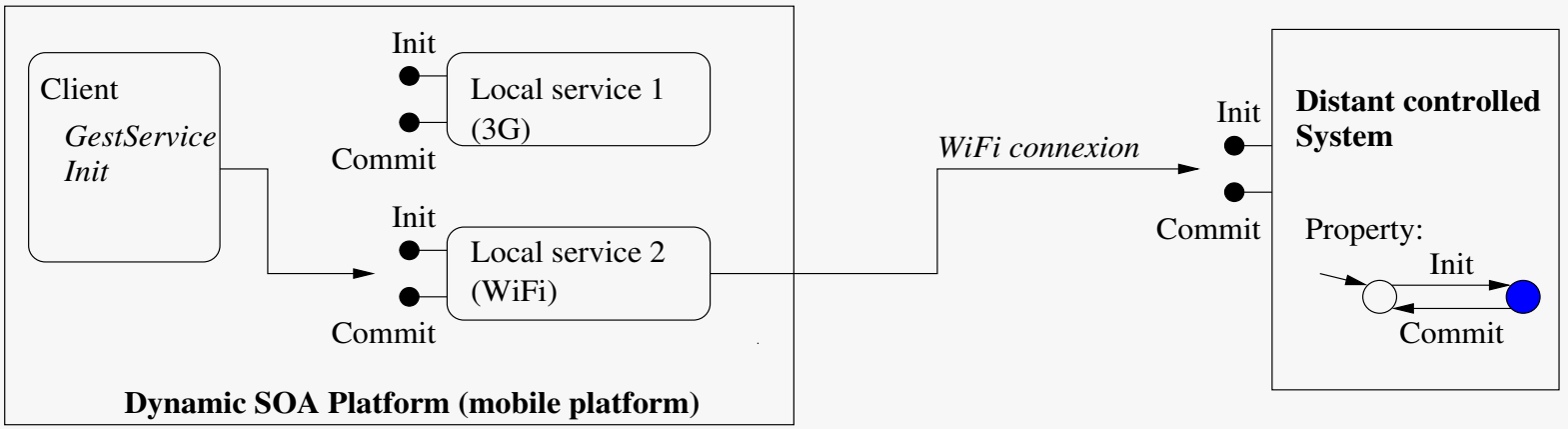
- Distant system controlled through local services
- Local services can dynamically (dis)appear
- Objective: *to check the respect of a property through the substitution of services*

Our case study



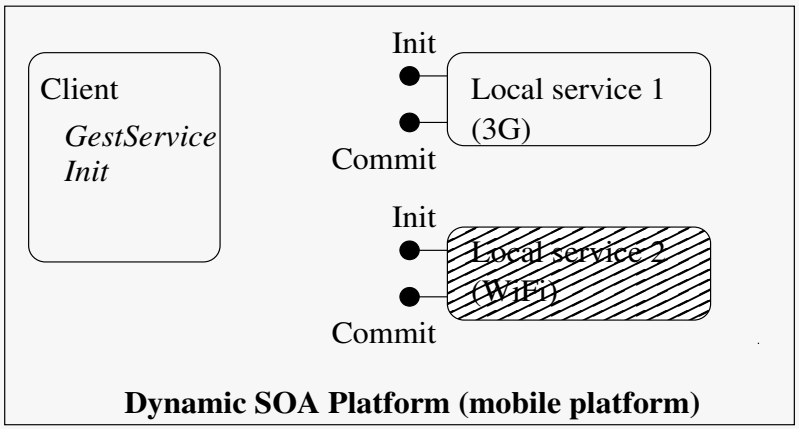
- Distant system controlled through local services
- Local services can dynamically (dis)appear
- Objective: *to check the respect of a property through the substitution of services*

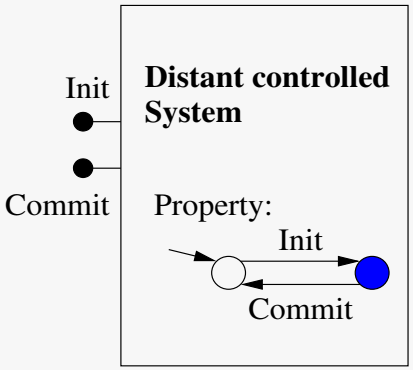
Our case study



- Distant system controlled through local services
- Local services can dynamically (dis)appear
- Objective: *to check the respect of a property through the substitution of services*

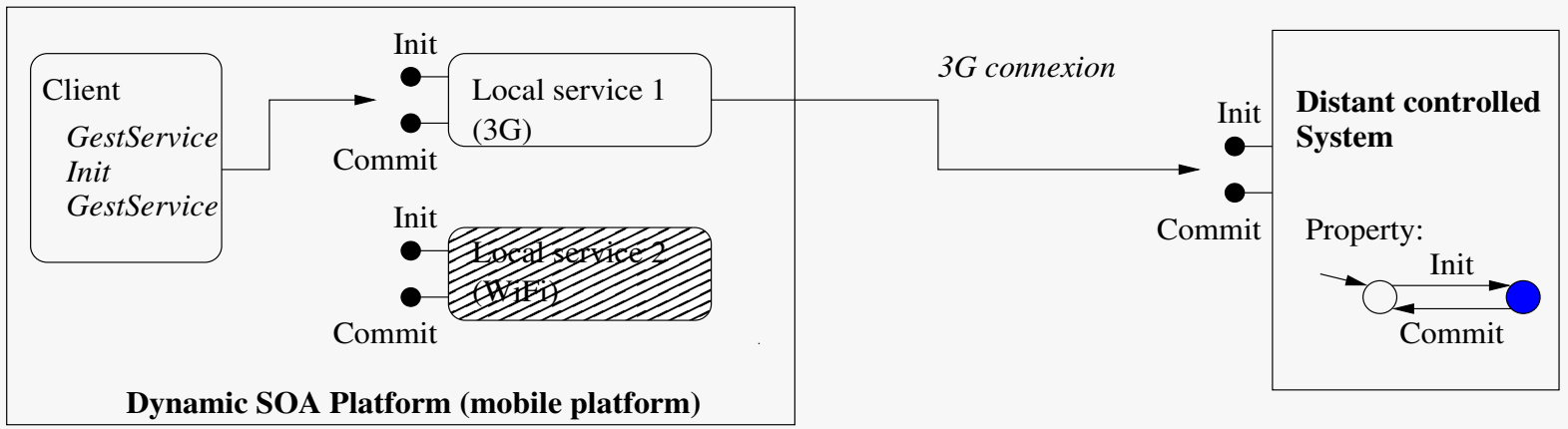
Our case study





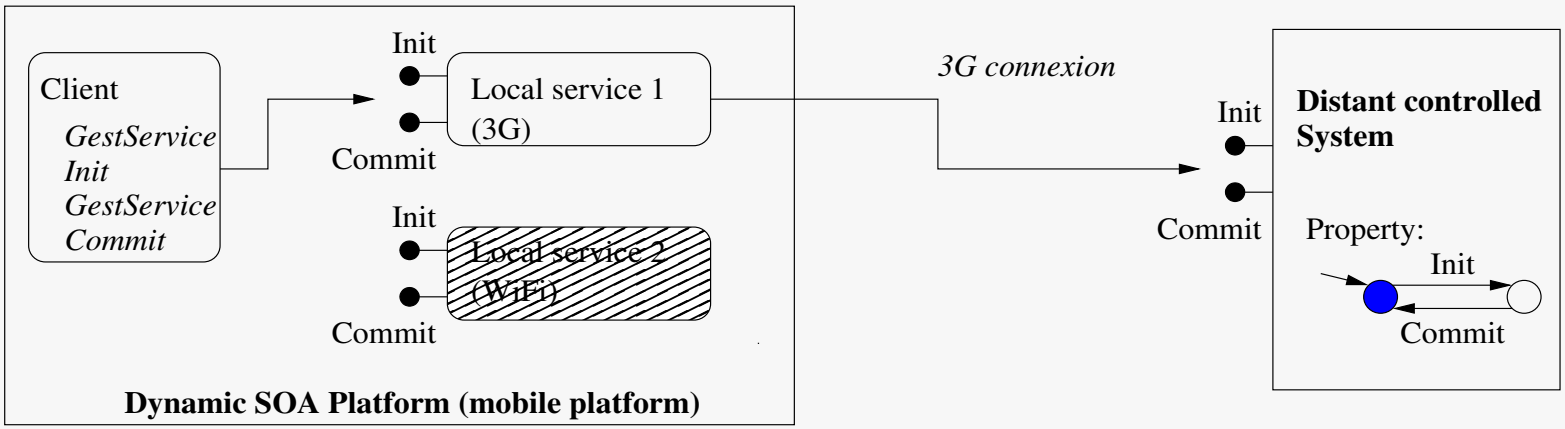
- Distant system controlled through local services
- Local services can dynamically (dis)appear
- Objective: *to check the respect of a property through the substitution of services*

Our case study



- Distant system controlled through local services
- Local services can dynamically (dis)appear
- Objective: *to check the respect of a property through the substitution of services*

Our case study



- Distant system controlled through local services
- Local services can dynamically (dis)appear
- Objective: *to check the respect of a property through the substitution of services*

Guidelines

Our objectives

- What's the meaning of monitoring a dynamic SOA?
- How to define/interface such monitor?

Our contributions

- Definition of two main properties:
  - dynamicity resilient
  - comprehensiveness
- Proposition of an architecture:
  - A non-intrusive proxy
  - Declared neither in client, nor in service
  - OSGiLarva : proof of concept

Related works – Classified by monitor configuration

Hard-coding

- *Properties manually written inside the code*
- Ex: **JML** or **Spec#**, annotation languages
- not *resilient to dynamic* code loading

Soft-Coding

- *Properties automatically injected inside the code*
- Ex: **Larva** or **JavaMOP**, monitors by aspects
- not *resilient to dynamic* code changing

Agnostic-Coding

- *Properties kept out of the code*
- Ex: **EventLog**, IDS trace analyzer
- not *comprehensive*, depends on what is logged

First contribution: monitoring properties

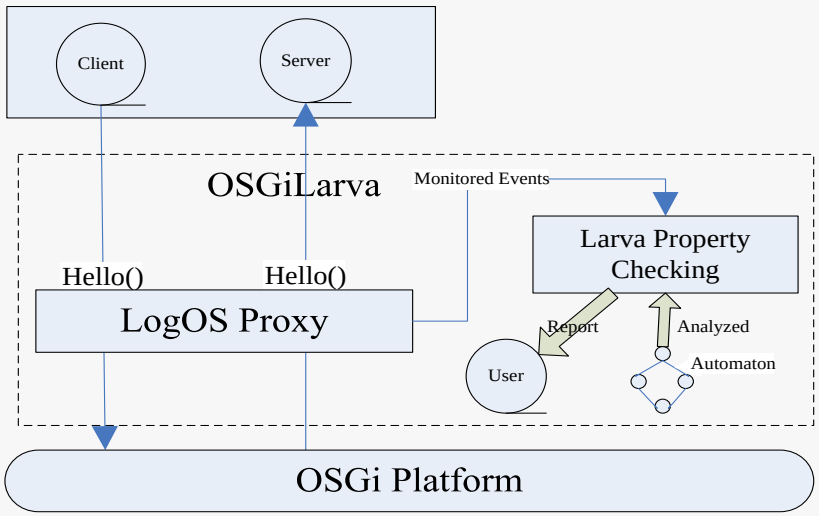
Dynamicity resilient

- monitor kept in memory even if a service is unloaded
- property is not hardly linked to the code

Comprehensiveness

- monitor can not be bypassed
- the monitor is not provided as a part of the code

Second contribution: proof of concept under OSGi



OSGi-Larva

- Checks the use of services VS a property
- Based on OSGi framework, LogOS proxy and Larva Monitor

OSGi-Larva

LogOS

- Logging proxy based on OSGi
- Patched to generate events description to send to Larva

Larva

- Java Monitor based on aspects
- Property description language based on parametrized timed automaton
- Patched to replace aspects by events description

Life Cycle

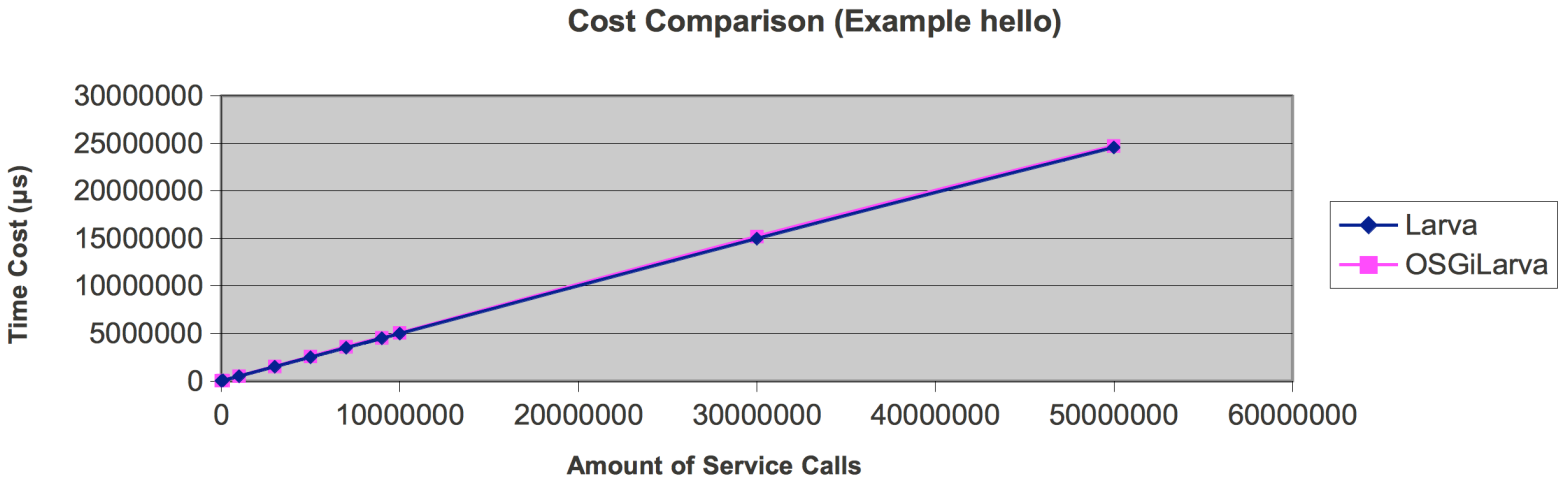
OSGi bundle: Deployment Unit

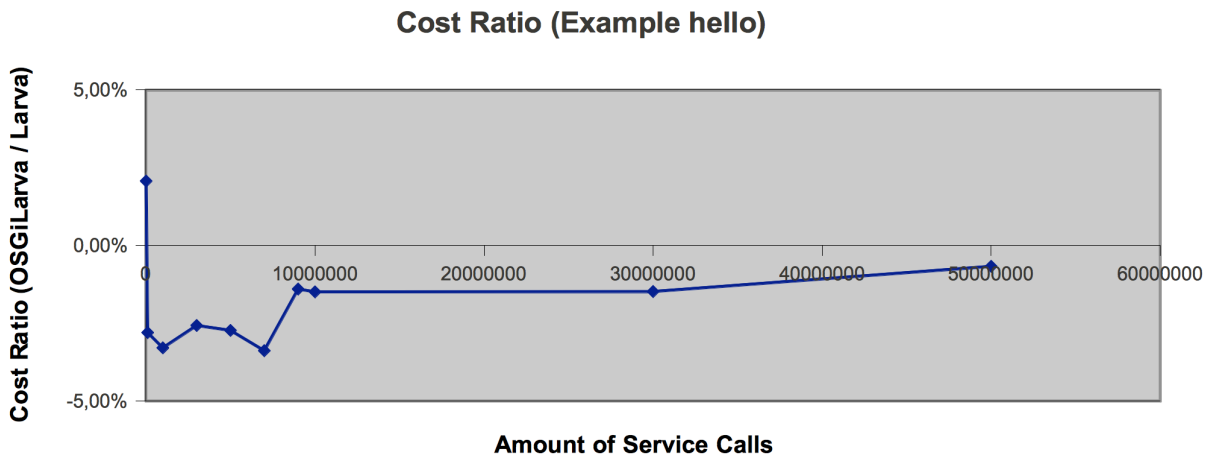
- Interfaces
- Service implementations
- Deployment code

Property life cycle

- Associated to Interfaces
- Can be deployed without any implementation
- Kept in memory at least while the client needs it

Comparison with Aspects: Some Results





Conclusion

Contributions

- Two main properties to monitor D-SOA systems
  - *Resilience to dynamicity*
  - *Comprehensiveness*
- Proof of Concept: OSGiLarva
  - Monitor for OGSi services
  - Based on Larva monitor and LogOS logger
  - Larva property description language
  - Time cost really close to aspect approach
  - Non intrusive approach *(Binary unchanged)*

Future Work

- To increase the property language with dynamic primitives *(Loading / unloading / substituting services)*
- To complete the tool
  - to consider value of I/O parameters
  - to make the deployment to be automatic
  - to externalize the monitor in another thread
- To merge this approach with our substitution API to make some autonomous and intelligent substitutions

Questions ?