OSGiLarva: A Monitoring System Supporting OSGi's Dynamicity

Yufang Dan[1,4], Nicolas Stouls[1], Christian Colombo[2], and Stéphane Frénot[3]

1.Université de Lyon, INSA-Lyon, CITI- INRIA F-69621, Villeurbanne, France-Email: first.second@insa-lyon.fr

2. Department of Computer Science, University of Malta-Email:first.second@um.edu.mt

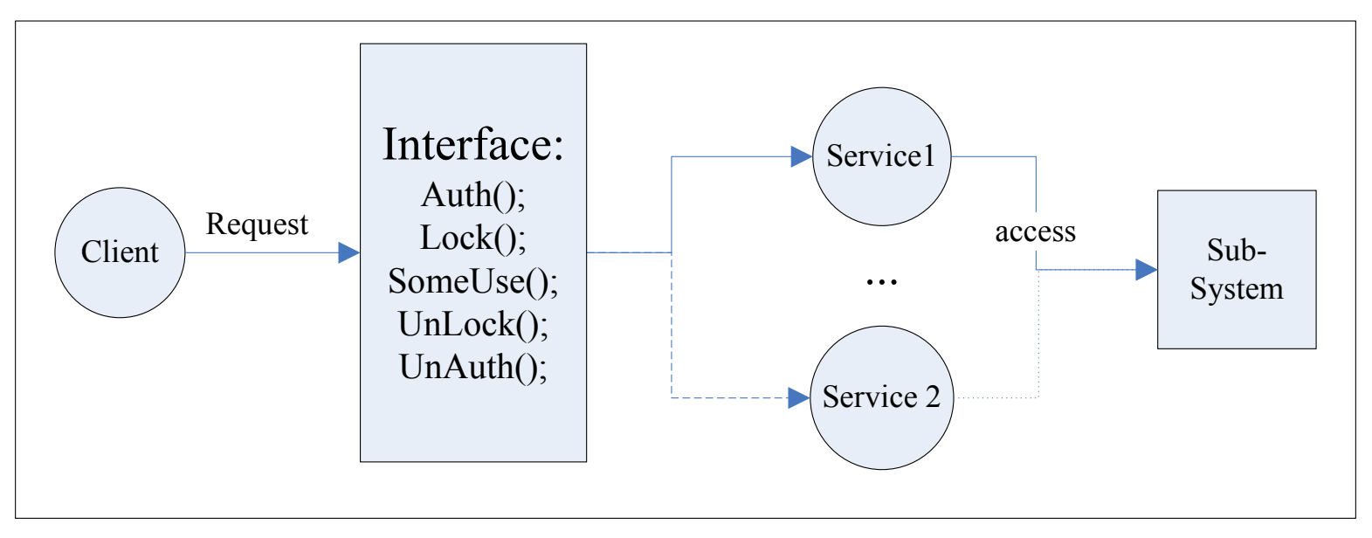
3. Université de Lyon, INRIA, INSA-Lyon, CITI- INRIA F-69621, Villeurbanne, France-Email: first.second@insa-lyon.fr

4. College of Computer Science, Chongqing University, Chongqing, China

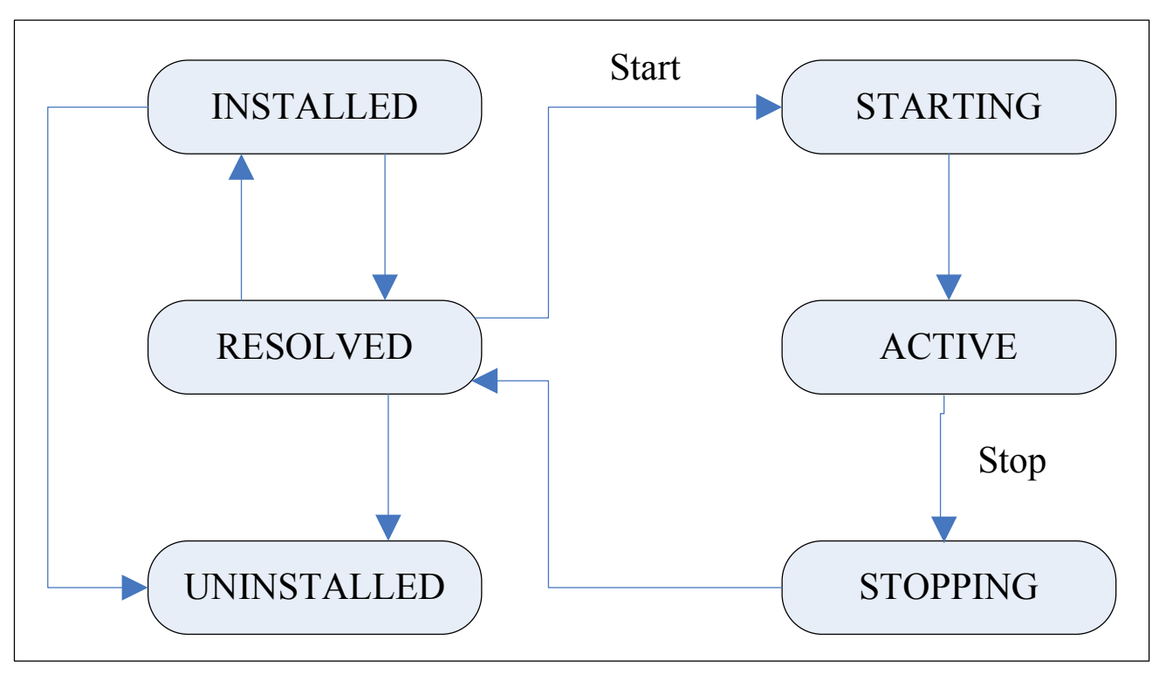
Context

Dynamic Service-Oriented Architecture System(DSOAS)

Service-Oriented Architecture(SOA):
- ➢ Loosely coupled client-server through interfaces
- ➢ Service
- ➢Client dynamically requests for a service
- ➢Client use the service



Characteristics of OSGi:
- ➢ Dynamic module system
- ➢ Self-contained unit: Bundle
- ➢ Life-cycle management of bundles:
  Install, start, stop,update and uninstall
- ➢ Without requiring a reboot system when bundle state changed.



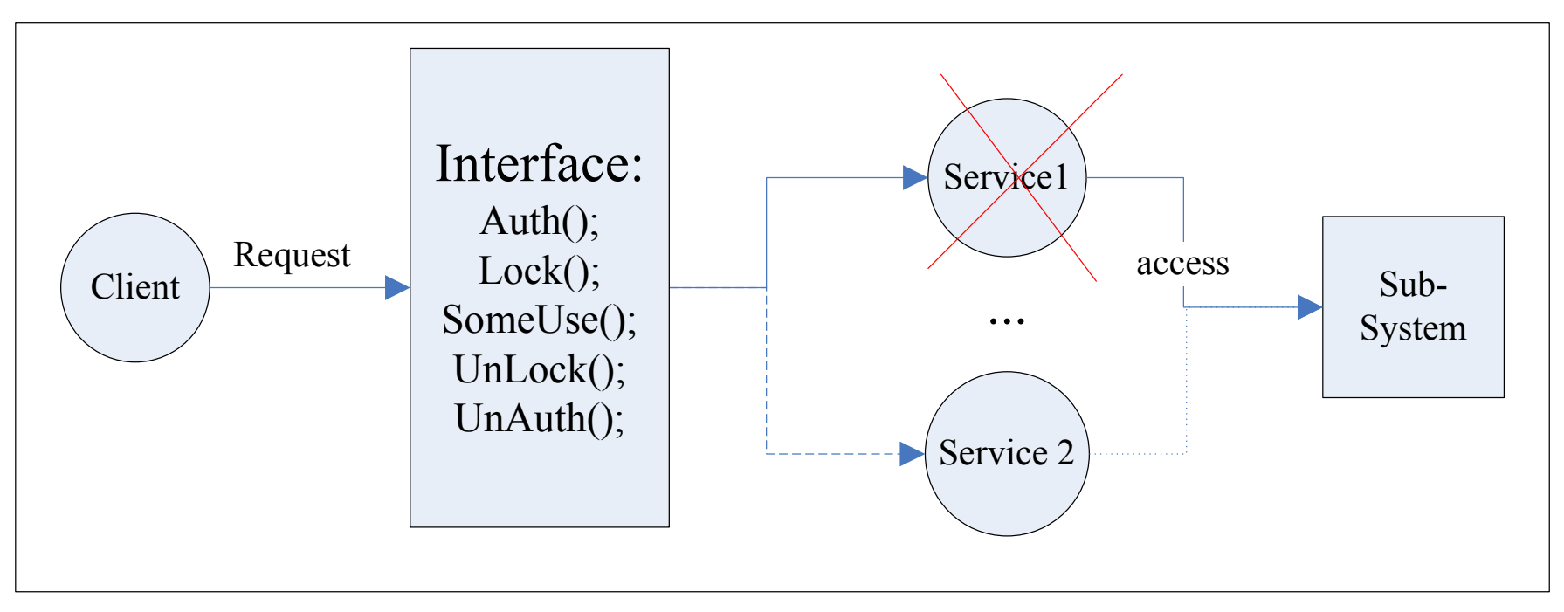
Life-cycle of Bundles

Objective of this work: To design a monitor supporting dynamic service-oriented architectures, which can check whether the client's behavior is authorized to perform or not.

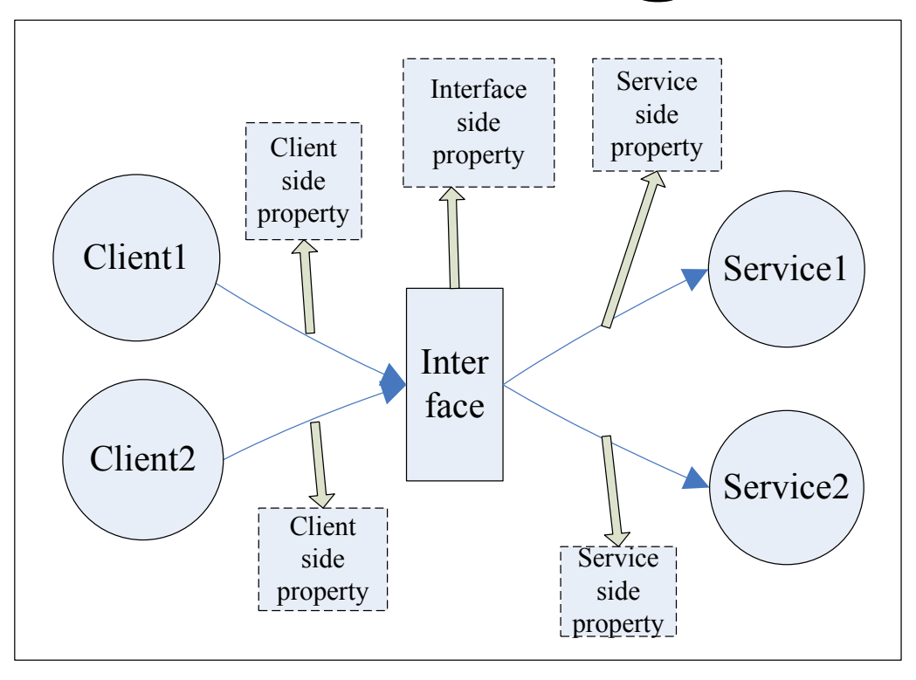
Motivation

A Dynamic behaviral monitoring bindings in DSOAS

A DSOAS with a traditional monitor:



The scope of property description in a monitor:
- ➢Client side
- ➢Interface side
- ➢Service implementation side



In the recent developed monitoring tools:

➢JavaMOP and Larva tools
- Java monitoring tools
- Relies on AspectJ for monitor injection
- Property described by the monitored system

➢Logging System
- Kept out of the logged system
- Made a loosely links
- Property described the written logs on logging system

Proposition: A monitor for dynamic systems needs to be resilient to dynamicity and comprehensive and has to give the possibility to express properties with an adapted scope as well as in terms of framework events.

Proposition

A monitoring system supporting OSGi's Dynamicity

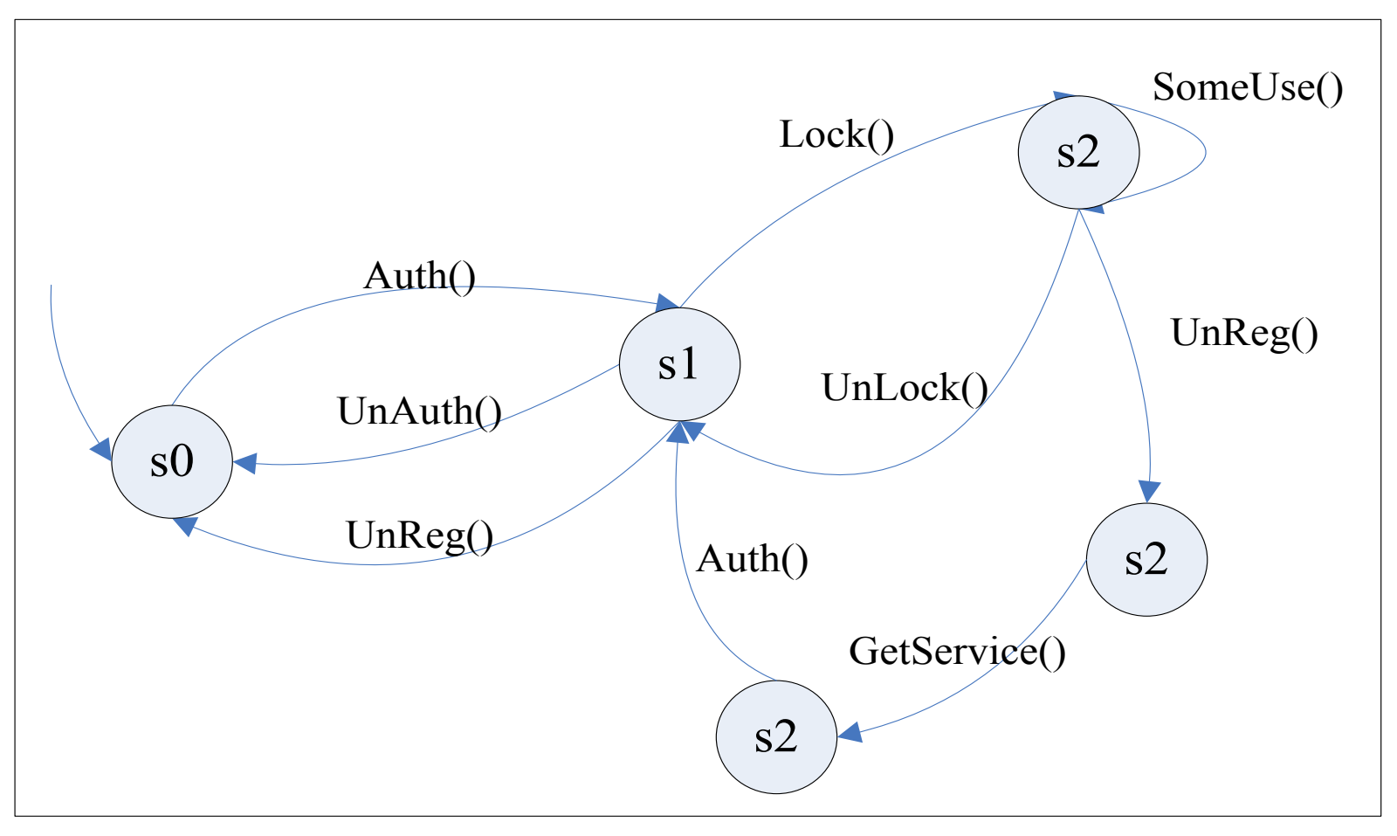
OSGiLarva tool:

➢Based on LogOs and Larva tool
- LogOs: OSGi Logging tool
- Larva: Java monitoring tool

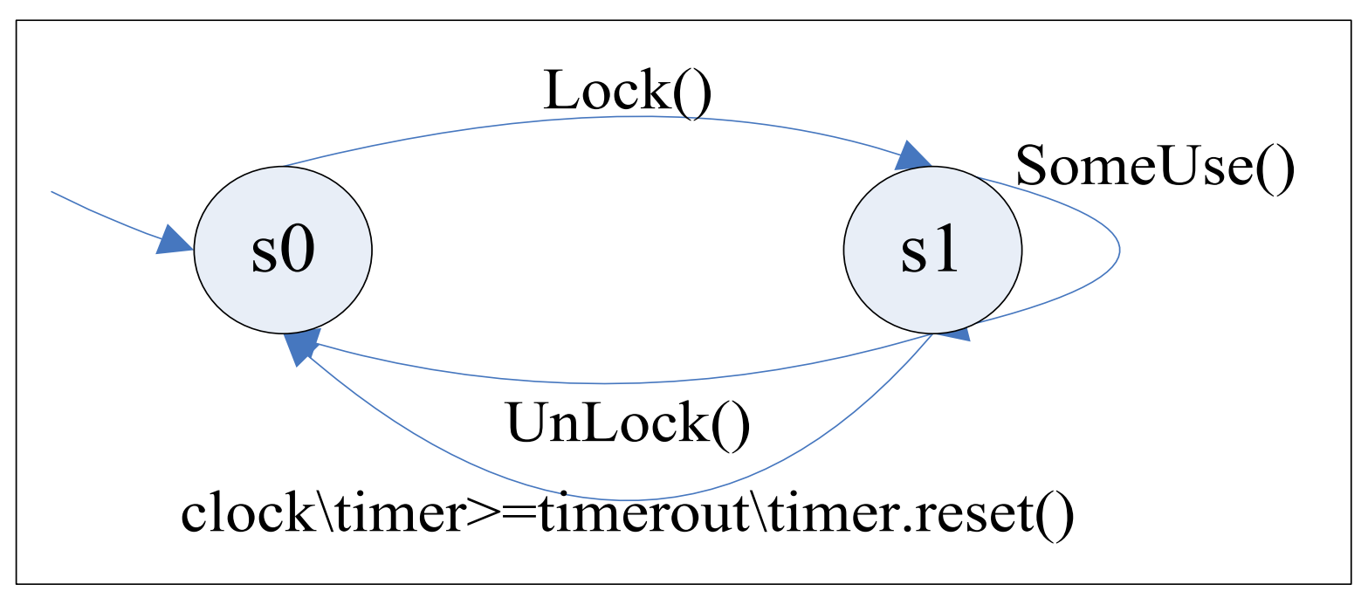
➢A dynamic runtime monitoring
- Dynamicity resilience
- Comprehensiveness

➢Dynamicity in property description

Dynamicity in property description:
- **Instance property**
- **Framework event primitives**



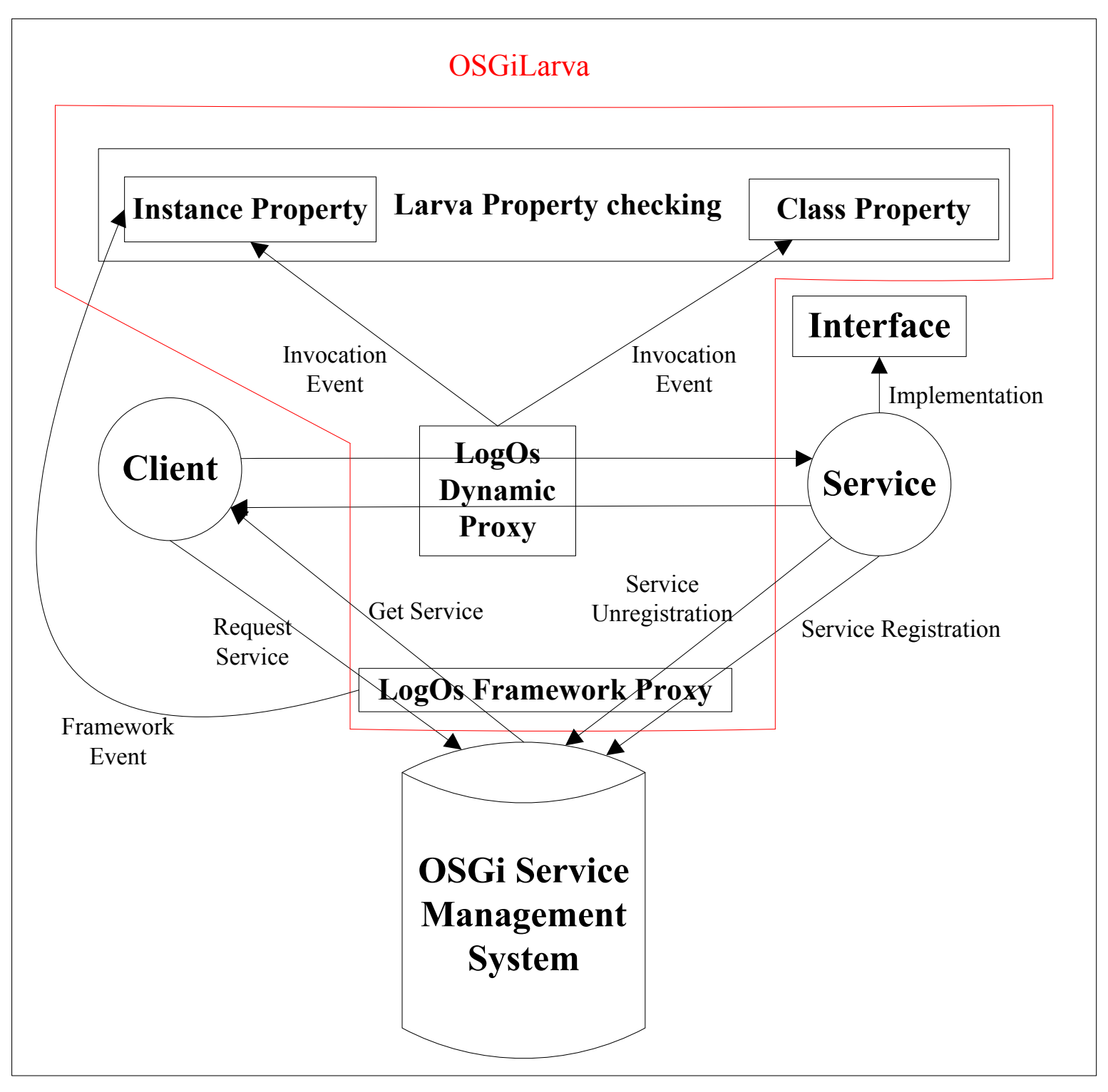
- **Class property**



Property description language:

```
GLOBAL{
    VARIABLES{ … }  EVENTS{ … }
    PROPERTY P1{
        %% Class property (same as Larva)
        STATES { … }
        TRANSITIONS { … }}
        %% Introduction of this new keyword
        FOREACHCLIENT(Long pid, String s){
           %%Instance property { … }
           VARIABLES { … }
           EVENTS{
              %% Just an event desription
              someEvent()=frameworkEvent();
              anotherEvent()=someMethod();
               … }
            PROPERTY P2{
                STATES{ … }
                TRANSITIONS{ … }}}}
```

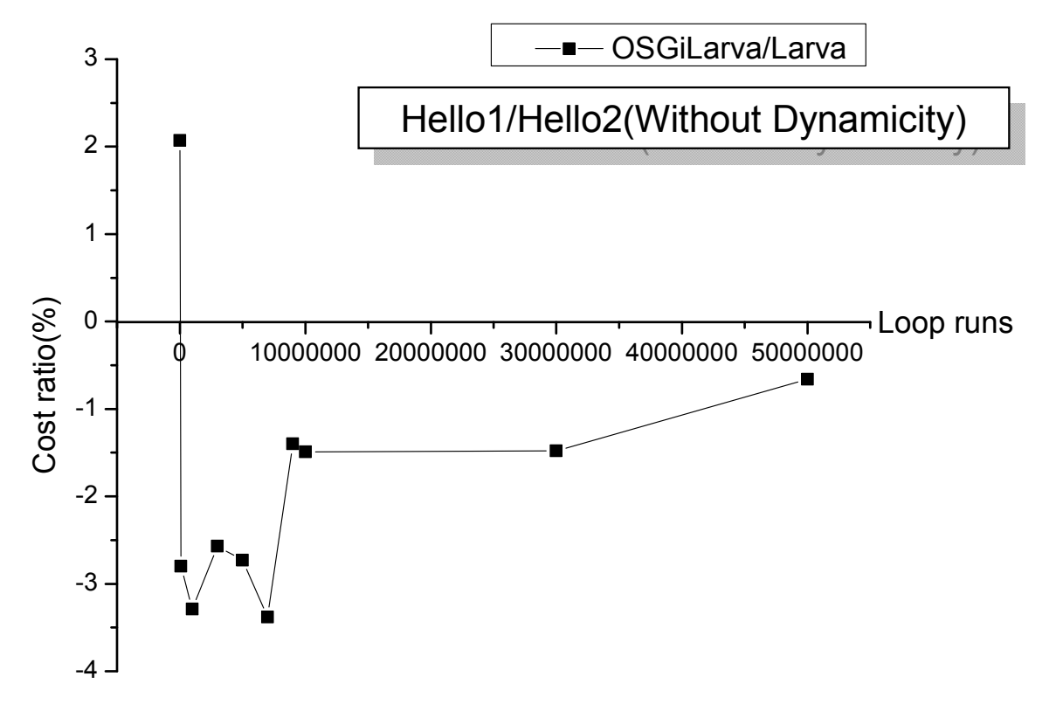
OSGiLarva Implementation:

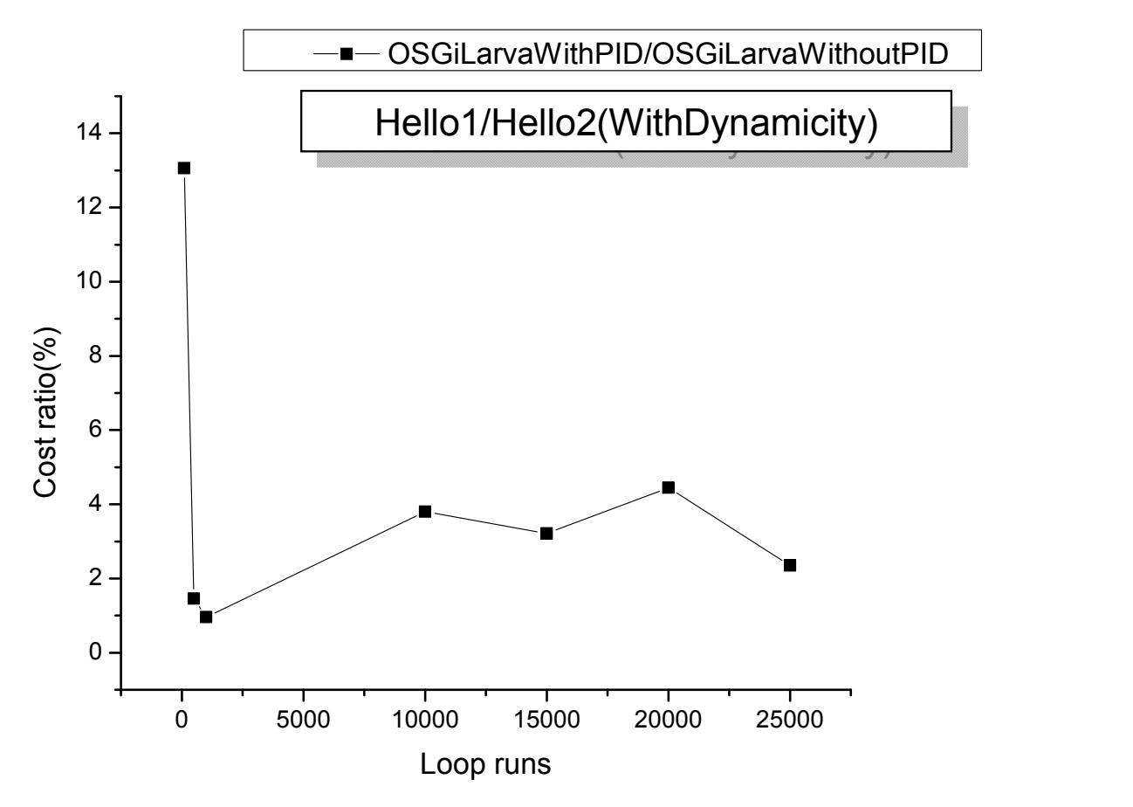


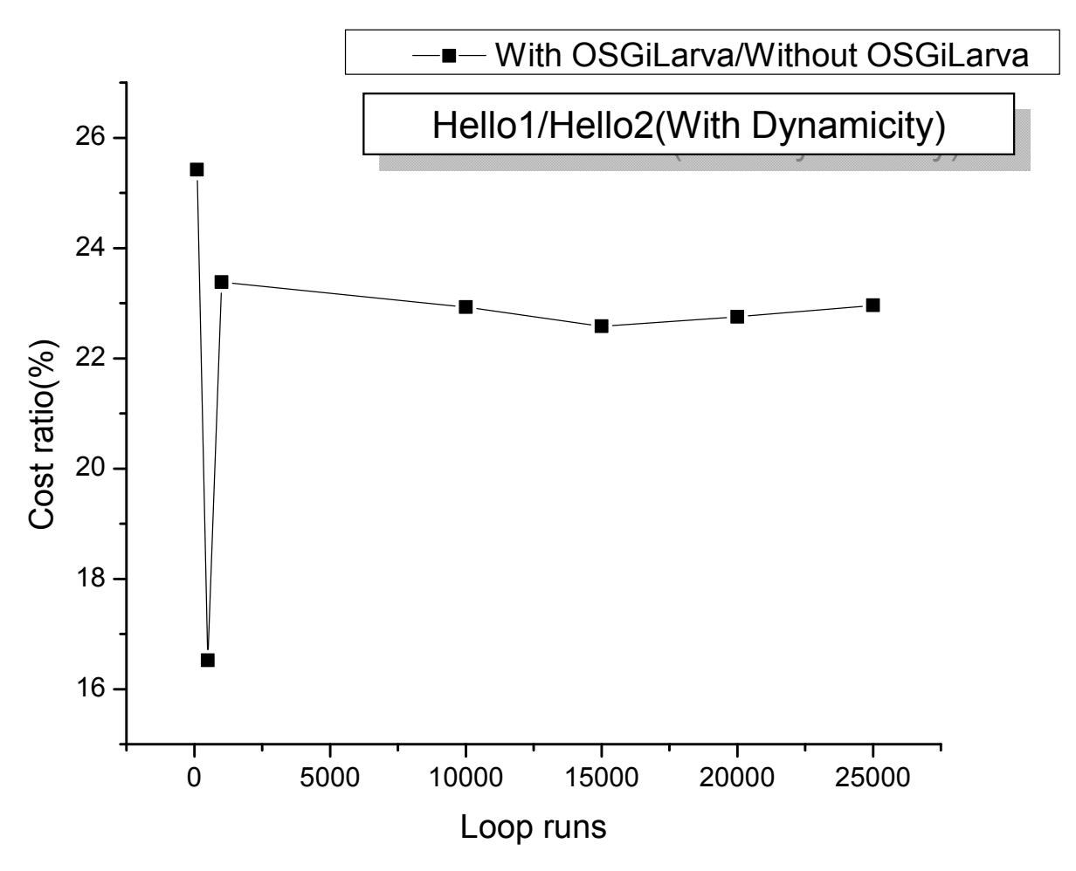
Results

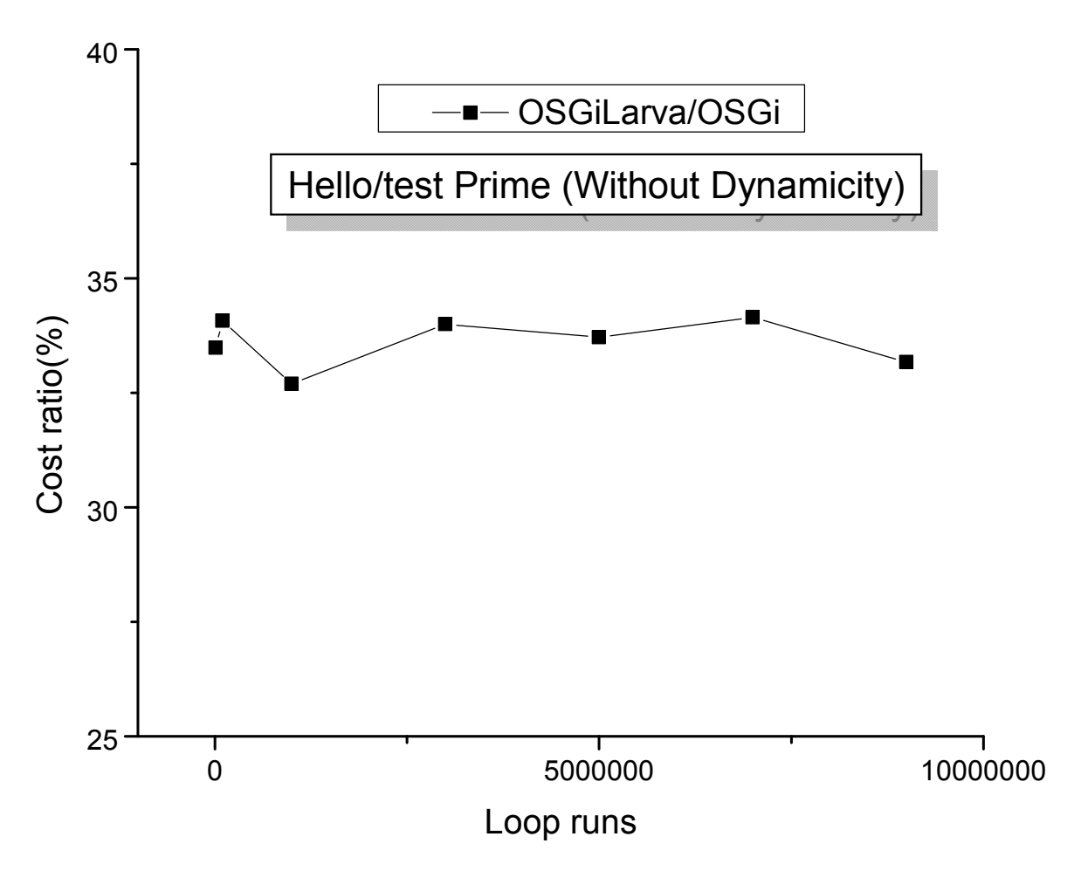
Three kinds of tests to analyze performance of OSGiLarva:
- ➢ Monitoring cost by using a proxy(OSGiLarva VS Larva)
- ➢ OSGiLarva efficiency(OSGi VS OSGiLarva)
- ➢ Overhead associated to get the caller Id

Simulation settings
- ➢ Two examples: one without dynamicity and another with dynamicity
- ➢ A loop on the client side:
- The client gets service and has the first call to the requested service
- Next, the client unregisters this service and registers the second service.
- And then, the client gets the second service and has the second call to this serivce.
- Finally, the client unregisters the second service and registers the first service.









Discussions

Conclusion

The proposed monitoring system can be binded in DSOAS with dynamicity resilience and comprehensivenes. When service substitutes during runtime, the monitor and the state of the substituted service can be kept in memory. It doesn't like the Aspect-oriented programming technique monitoring tool. Our monitoring tool can keep the original byte-code unchanged. We have also made our property description having more dynamicity. Property file is composed by instance property and class property, and authorized the use of framework events in it.

Perspectives:

1. Reduce our monitor time cost., 2. Do request time compliant with AspectJ technology. 3. To make multiple interfaces in property description. 4. To add call parameters in property description

References

➢C, Colombo, G. J. Pace, and G. Schneider, «Larva-safer monitoring of real-time java programs», in SEFM, 2009

➢Stephane Frenot and Julien Ponge, «LogOs: an Automatic Logging Framework for Service-Oriented Architectures», in SEAA'12, Izmir, Turquie, Sep. 2012.[Online]. Available: http://hal.inria.fr/hal-00709534

➢Yufang Dan, Nicolas Stouls, Stephane Frenot, Christian Colombo, «A Monitoring approach for Dynamic Service-Oriented Architecture Systems», in SERVICE COMPUTATION 2012, pp. 20-23

COMMUNAUTÉS DE RECHERCHE ACADÉMIQUE RhôneAlpes
INNOVATIONS, MOBILITÉS, TERRITOIRES ET DYNAMIQUES URBAINES
COMMUNAUTÉS DE RECHERCHE ACADÉMIQUE RhôneAlpes
T.I.C. ET USAGES INFORMATIQUES INNOVANTS