*WSInject*

# A Fault Injection Tool for Testing Web Services Composition

Fayçal Bessayah, Ana Cavalli - IT/Telecom SudParis, FRANCE
Willian Maja, Eliane Martins, Andre W. Valenti, IC/UNICAMP, BRAZIL

faycal.bessayah@it-sudparis.eu

# Topics

- Web services - Fault injection
- Fault injection tools for Web services
- WSInject
- Case study : The Travel Reservation Service

# Web services

- Application components communicating over the Internet
- Features
  - Communication based on standard Web protocols :HTTP,XML,SOAP…
  - Platforms and languages independent
  - Reusable and modular applications

# Fault injection

- Definition

  *Deliberate introduction of faults into a running system to observe its behavior*

- Applicability

  - To verify whether the error detection and recovery mechanisms behave as expected.

  - To evaluate dependability measures such as reliability for a giving mission time, availability, performance degradation due to fault handling.

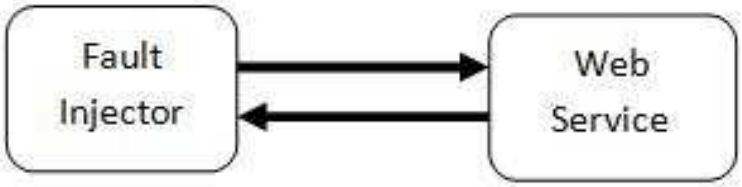
  - To understand the effects of real faults.

# Fault injection tools for Web services

- Faults can be injected both at interface and communication levels
  - Interface faults
    Corrupting input/output messages & parameter values
  - Communication faults
    Delay, reorder, replicate messages

# Fault injection tools for Web services

- Network level fault injectors
  - DOCTOR[1], Orchestra [2], DEFINE [3]…
  - Not able to decode SOAP messages➔ No interface faults !
- Web services fault injectors
  - WSBang [4], PUPPET[5], GENESIS [6]…

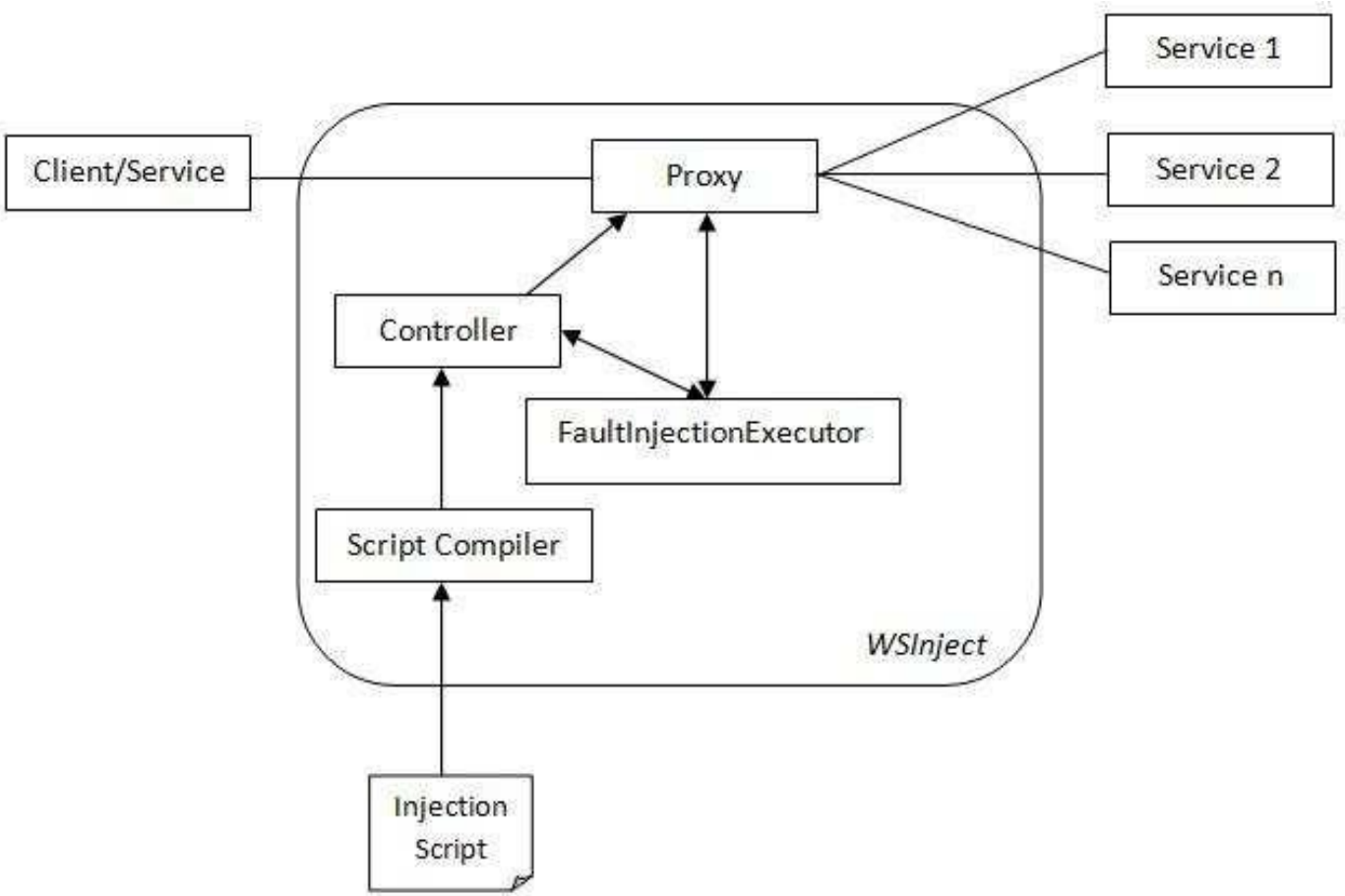



  - Consume the tested service
    - ➔ No communication faults !
    - ➔ Cannot be used to test composed services

# WSInject

- A Web service fault injector able to inject both interface and communication faults
- Can be used to test both real and simulated services
- It can inject faults on both single and composed services

# WSInject : Architecture



# WSInject : Injection script

- A script-driven fault injector
- An injection script is a set of conditions-actions statements

```
CampaignDescriptor -> FaultInjectionStatement [CampaignDescriptor]

FaultInjectionStatement -> ConditionSet : FaultList ; [FaultInjectionStatement]

ConditionSet -> Condition [&& ConditionSet]

Condition -> operation(String)| contains(String) | uri(String) |
isRequest()|isResponse()

FaultList -> Fault [, FaultList]

Fault -> delay(Integer) | multiply(String,Integer) | stringCorrupt(String,String) |
        xPathCorrupt(String, String) | empty() |CloseConnection ()
```

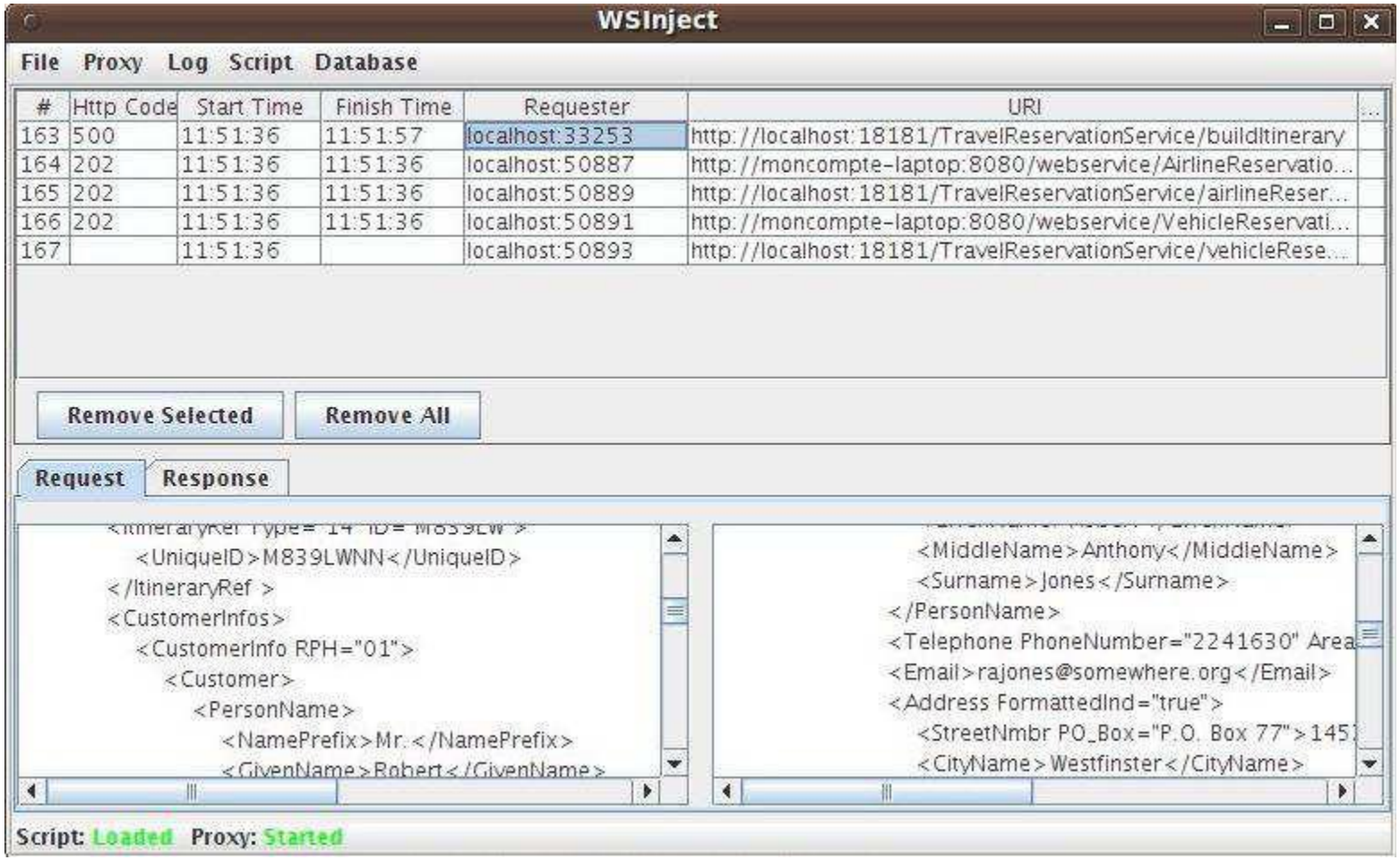
# WSInject : Injection script

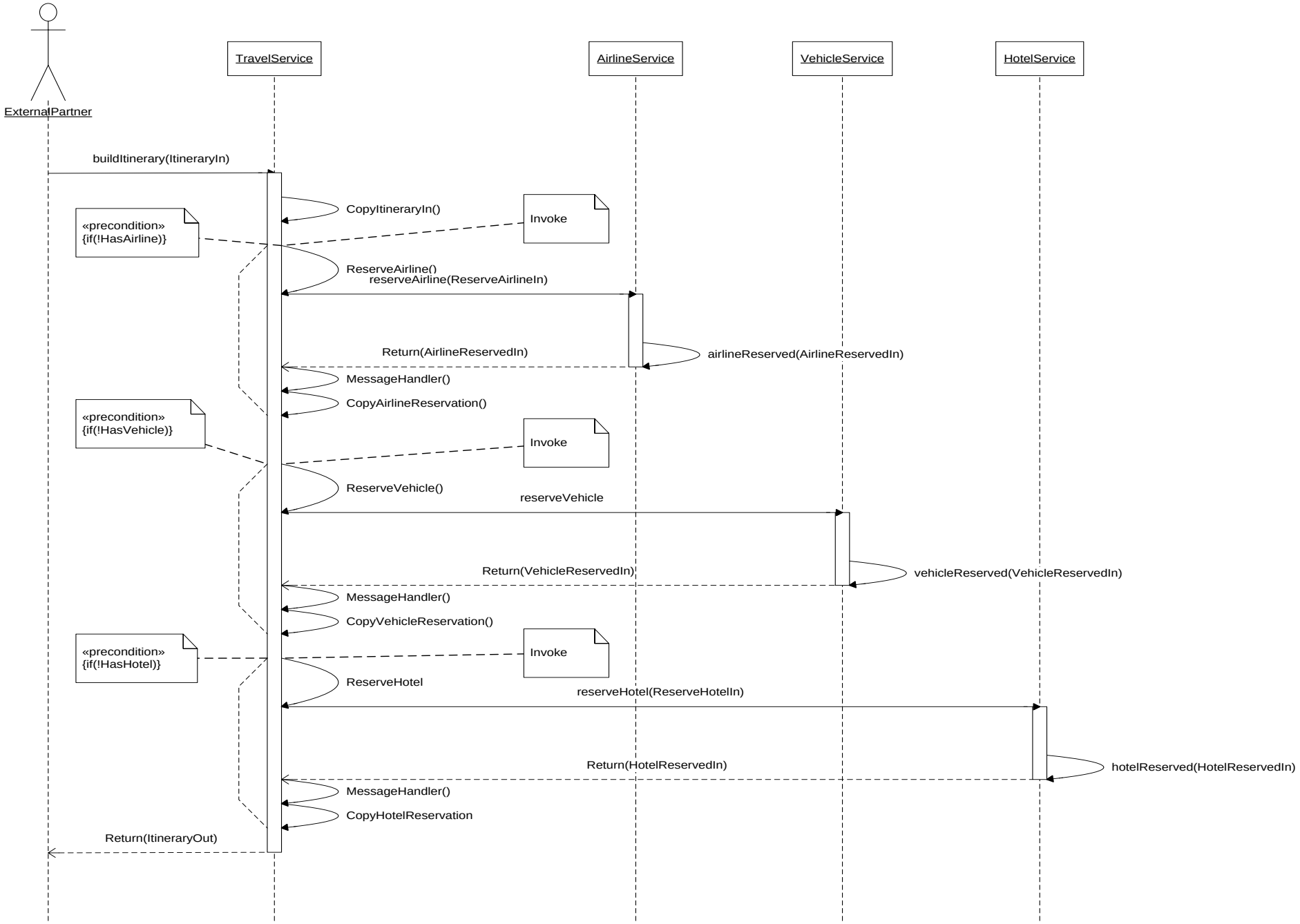
- Examples

```
Operation(«LoginRequest »): delay(5000);
```

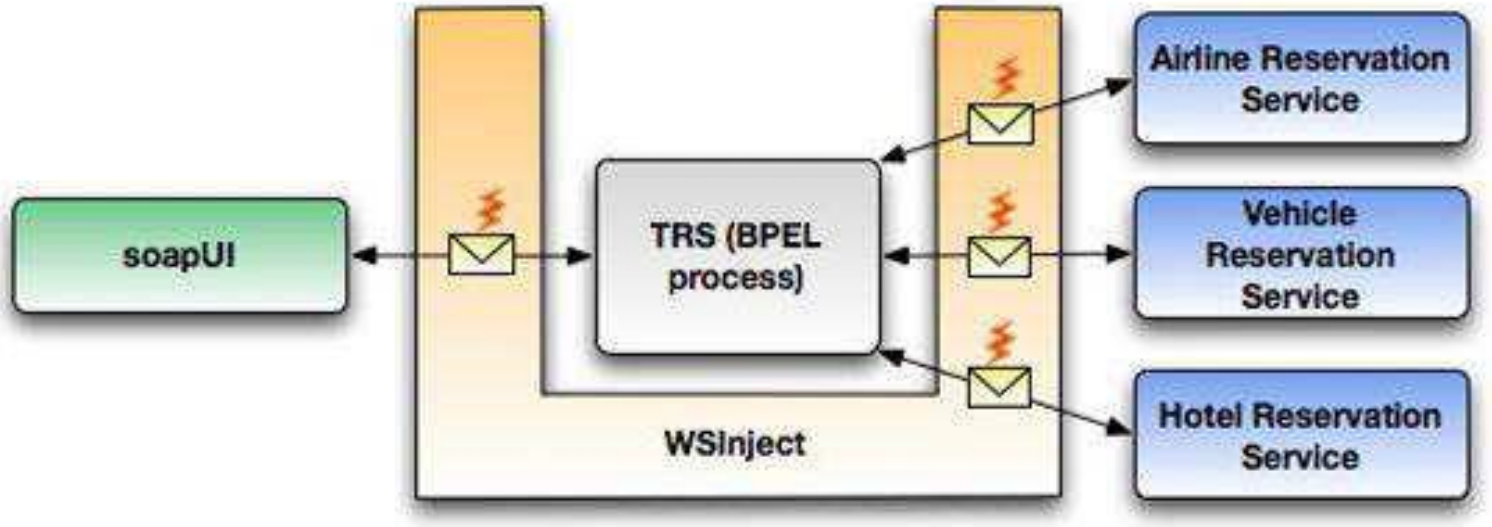
```
Operation(«LoginRequest») && contains («<username> Bob </username>»):
        xPathCorrupt(«//username/text()», «XXXX»);
```

# WSInject : GUI




ExternalPartner
TravelService
AirlineService
VehicleService
HotelService
buildItinerary(ItineraryIn)
CopyItineraryIn()
Invoke
«precondition»
{if(!HasAirline)}
ReserveAirline()
reserveAirline(ReserveAirlineIn)
Return(AirlineReservedIn)
airlineReserved(AirlineReservedIn)
MessageHandler()
CopyAirlineReservation()
«precondition»
{if(!HasVehicle)}
Invoke
ReserveVehicle()
reserveVehicle
Return(VehicleReservedIn)
vehicleReserved(VehicleReservedIn)
MessageHandler()
CopyVehicleReservation()
«precondition»
{if(!HasHotel)}
Invoke
ReserveHotel
reserveHotel(ReserveHotelIn)
Return(HotelReservedIn)
hotelReserved(HotelReservedIn)
MessageHandler()
CopyHotelReservation
Return(ItineraryOut)

# Experimentations & Results



# Experimentations & Results

- Injection process
  - Interface faults
    - Corrupting string and integer values of SOAP messages
    - Structure corruption:
      Replicate/ Delete XML elements, inverse openning and closing tags
  - Communication faults
    - Message delaying and simulation of connection loss

# Experimentations & Results

- wsAS crash scale [8] defines two failure modes : ***Abort*** and ***Silent***

| Injected faults | wsAS scale | |
|---|---|---|
| | Silent failure | Abort failure |
| Corruption of parameter values | X | |
| Corruption of message structure | | |
| Delaying requests | X | |
| Delaying responses | | X |
| Empty messages | | X |
| Message replication | | |

# References


[1] K. G. Shin S. Han and H. A. Rosenberg. Doctor: An integrated software fault injection environment for distributed realtime systems. *In Proceedings of IEEE International computer performance and dependability symposium. Erlangen, Germany,*1995.

[2] F. Jahanian S. Dawson and T. Mitton. Orchestra: A probing and fault injection environment for testing protocol implementations,. *In Proceedings of IEEE International computer performance and dependability symposium. Urbana- Champaign,IL,USA, 1996.*

[3] W.L. Kao and R.K. Iyer. Define: A distributed fault injection and monitoring environment. *In Proceedings of IEEE Fault-Tolerant Parallel and Distributed Systems (IEEE-FTPDS'94)., pages 252–259, 1994.*

[4] WSBang at https://www.isecpartners.com/wsbang.html

[5] G. D. Angelis A. Bertolino and A. Polini. A qos test-bed generator for web services. *In ICWE, ser. Lecture Notes in Computer Science, L. Baresi, P. Fraternali, and G.-J. Houben, Eds., 4607:17–31, 2007.*

[6] H.L. Truong L. Juszczyk and S. Dustdar. Genesis - a framework for automatic generation and steering of testbeds of complexweb services,. *In Proceedings of the 13th IEEE International Conference on Engineering of Complex Computer Systems(ICECCS'08), pages 131–140, 2008.*

[7] NetbBeans IDE at http://netbeans.org/

[8] N. Laranjeiro M. Vieira and H. Madeira. Benchmarking the robustness of web services,. In *Proceedings of the IEEE 13th Pacific Rim international Symposium on Dependable Computing (PRDC), 2007.*

Thank you