RobustWeb

Construction & Validation of SOA Applications with Web Services

58th. IFIP 10.4 WG Meeting – Chicago – June/2010

Participants

UNICAMP

TELECOM & Management SudParis

LAAS-CNRS

INPE
MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

.inf
INSTITUTO DE INFORMÁTICA
UFRGS

Universidade Federal de Alagoas
Instituto de Computação

departamento de INFORMÁTICA
UFES

Centro de Informática
U·F·P·E

Web Services (WS)

- According to World Wide Web Consortium (W3C), a WS:
  - Is a software system designed to support interoperable machine-to-machine interaction over a network
  - Has an interface described in a machine-processable format – WSDL (Web Service Definition Language)
  - Communicates with other WS using SOAP-messages, typically conveyed using HTTP
  - Can be discovered and connected to an application during runtime

About robustness

- What is robustness:
  - IEEE Std. Glossary:

  « *The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions* »

  - Can be interpreted as system ability to:
    - Tolerate external faults
    - Handling exceptions
    - Tolerate attacks

Robustness and Web Services

- Why is robustness important for Web services?
  - WS-based architectures are promising for the development of omnivalent systems, which are systems which must present characteristics such as ubiquity, dependability and security

(SBC - *Grand Research Challenges in Computer Science in Brazil* from 2006 to 2016)

Robustness testing issues

- Workload
  - How to generate inputs to exercise WS operations?
- Faultload
  - What faults to inject?
- Fault injector
  - How to inject the faults?
- Robustness failures identification
  - How to characterize service mal-functioning?

Robustness testing issues

- Workload
  - How to generate inputs to exercise WS operations?
- Faultload
  - What faults to inject?
- Fault injector
  - How to inject the faults?
- Robustness failures identification
  - How to characterize service mal-functioning?

Robustness testing architecture

- WSInject – fault injector
  - Injects faults to test a service or a composition of services

Robustness testing issues

- Workload
  - How to generate inputs to exercise WS operations?
- Faultload
  - What faults to inject?
- Fault injector
  - How to inject the faults?
- Robustness failures identification
  - How to characterize service mal-functioning?

How to identify robustness failures?

- There are several ways to classify robustness failures [Vieira 2009]:
  - CRASH scale of the Ballista approach, adapted to the WS context – wsCRASH :
    - Catastrophic: application server crashes or reboots
    - Restart: WS execution hangs
    - Abort: abnormal termination of the WS
    - Silent: after a timeout, no error is indicated
    - Hindering: incorrect error code or delayed response

However …

- From the point of view of WS consumer, only Abort and Silent are observable
  - A system can fail without aborting or delaying responses or sending error messages
  - For example:
    - in an Elevator service, if there is a requestUP (floorID), then the Elevator should moveUp( ), stopAt(floorID) and opendoor( ) or else it stays in the same floor
    - In case of a request to a valid floor, if the Elevator opendoor( ) before stopAt(floorID), a failure occurs

Our proposal

- Use of a passive testing approach:
  - Similar to monitoring:
    - Observes the exchange of messages (inputs and outputs) of an implementation under test during runtime → trace of messages
  - Analyze trace to detect anomalous behaviors
    - Compare with properties :
      - derived from standards
      - derived from formal specifications
      - obtained from hazard analysis
      - proposed by experts

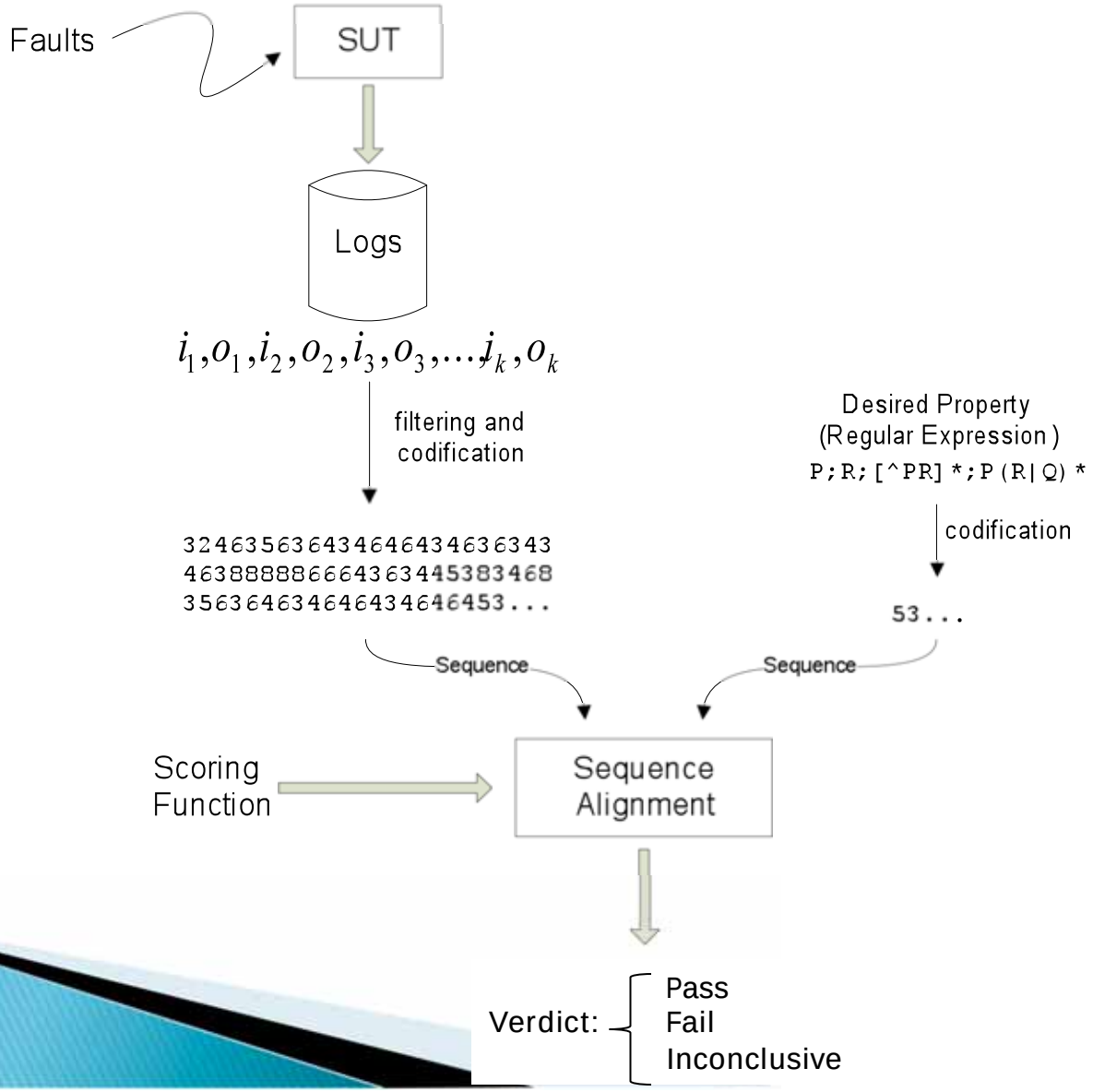
Aspects to consider (1)

- What properties to analyze:
  - For the moment, only safety properties are being considered: something bad never happens during execution
  - If something bad happens → a robustness failure is identified
- How to express the properties:
  - Regular expressions, as they are good to represent allowable sequences of interactions
    - Detection of incorrect sequencing is useful:
      - It can be the failure itself or the cause of a failure

Aspects to consider (2)

- Which algorithm to use to analyze the trace?
  - Traditionally, in passive testing based on properties analysis → pattern matching
  - We get inspiration on Bioinformatics: algorithms used for the alignment of two DNA sequences
    - It is possible to take into account semantic aspects, not allowable in pattern matching algorithms
      - Use of a scoring system
    - It is possible to detect insertions, deletions or replacements of one or more inputs or outputs in the sequence
    - It is possible to obtain some statistics, as for example, number of matches and mismatches

Schematic view of the approach



Characteristics of the algorithm used

- Based on dynamic programming
  - interesting for testing purposes, as it guarantees to find the optimal alignment between sequences
- Local alignment
  - more useful for sequences of different sizes. The focus is to find regions of high similarity in the longer sequence (the trace)

```
GTGTATACC-AGAG
  |||  ||  ||
--GTAC-CCAAG--
```

Next steps

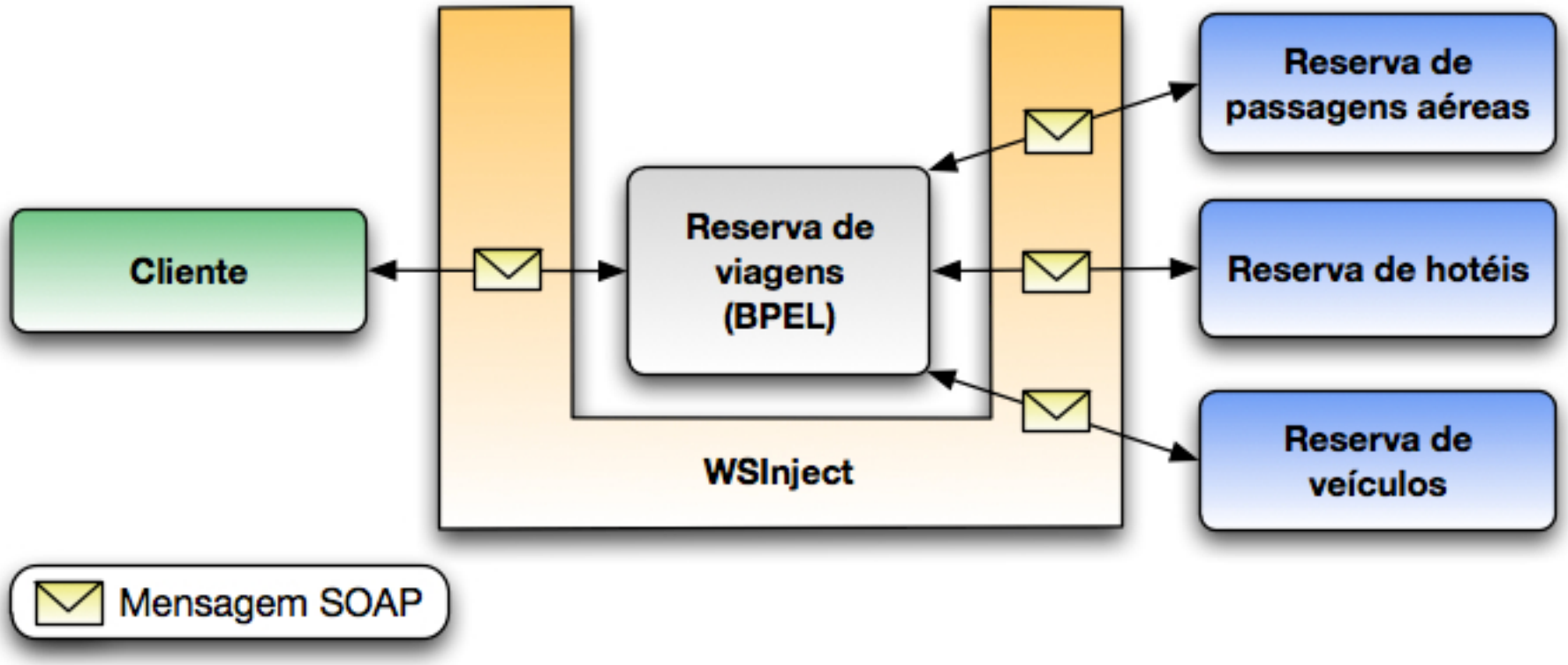
- Analysis of different scoring systems:
  - What scores to give to matches, mismatches (good and bad), gaps?
- Analysis of false positives, false negatives
- Are quasi-optimal alignments useful or not?
- Applications:
  - Benchmark programs for the analysis
  - Real-world WS

Some open issues

- Expressing more “powerful” properties:
  - How to take into account timing constraints
    - Ex.: event A should occur within 30u.t after event B
- Considering the traces of different service interactions in a service composition

Thanks!

http://robustweb.ic.unicamp.br/


Cliente
Reserva de viagens (BPEL)
Reserva de passagens aéreas
Reserva de hotéis
Reserva de veículos
WSInject
Mensagem SOAP