Web Service Monitoring - Part I
An Introduction (informal)

Fabio Barbon

Astro project
ITC-IRST

21.09.2005
Motivating questions

- External versus Internal monitoring
- Passive versus Active monitoring
- Protocol versus Property monitoring
- Language for Properties
External Monitoring

- We\(^1\) don’t trust completely our partners\(^2\)
- Observational behaviour
  - We sense communication I/O
  - From identity relation on states to distinguishability equivalence relation on states

**current status**

We automatically synthesize protocol and \textit{KPLTL} property monitors via a power-set construction over an \textit{STS} set representing a web service choreography.

\(^1\)as service composers and/or composed service providers
\(^2\)component service providers
Internal Monitoring

- We don’t trust completely ourselves
- Actual (activity-level) behaviour
  - Observational reasoning still possible (via localhost monitoring)
  - *Full Knowledge* (we are inside execution engine)
  - We can mix observational and actual behaviour information.

**current status**

Just started investigation of internal monitoring possibilities and technical issues.
Passive vs Active Monitoring

Further extend the idea of internal monitoring: monitor components could play an active role in the web service implementation.

**current status**

- We emit pieces of java code (suited for deployment in ActiveBPEL execution engine) that continuously output monitors validity status in engine’s administrative interface.
- Not yet started investigation of active monitoring possibilities.
Informal briefing

Motivating questions
External vs Internal Monitoring
Passive vs Active monitoring
Protocol versus Property monitoring
Language for Properties

Protocol Monitoring

- A protocol monitor verifies that a service behaves consistently with its published interaction flow (*protocol*)
- Automatically generated using only *Abstract BPEL* specification (no need to mark-up anything)
- Is generated as a deterministic *STS* with a set of final states
- It monitors:
  - Message lexicon
  - Correct life (message order preserved (by parallel exec))
  - Correct death (service’s instance won’t die prematurely)

**current status**

**done. (complexity bounds for protocol monitor?)**
KPLTL Property Monitoring

Knowledge-level Past-only LTL

\[ K(\text{User.state=CANC}) \rightarrow (0 K(\text{Hotel.state=CANC}) \& 0 K(\text{Flight.state=CANC})) \]

- Automatically generated using Abstract BPEL specifications and a formula like the one above
- No restriction to single services: formulas can traverse (asynchronous) choreographies
- Shifting to belief-level needs a (reasonable) interpretation for equality between states: know and maybe?

Current status:

Done (past-time only, know/maybe modes!).
Language for Choreography Properties

- State-based or Message-based?
- Need to speak about future?
- LTL, regular expressions, ...?
- Knowledge-level monitoring?

current status

- Just started investigation
- Strong need for realistic scenarios of property monitoring.