Modeling and Analyzing Timed Web Services Protocols

Julien Ponge\textsuperscript{1,2} – ponge@isima.fr
http://www.isima.fr/ponge/

\textsuperscript{1}Laboratoire LIMOS, ISIMA, Clermont-Ferrand, France

\textsuperscript{2}Computer School of Engineering, The University of New South Wales, Sydney, Australia

ICSOC’05 PhD Symposium
Agenda

Introduction
Web services today
Outline of approach

Framework
Timed business protocols
Temporal compatibility and replace-ability analysis
ServiceMozaic

Conclusion
Conclusion
Perspectives and future work
Agenda

Introduction
  - Web services today
  - Outline of approach

Framework
  - Timed business protocols
  - Temporal compatibility and replace-ability analysis
  - ServiceMozaic

Conclusion
  - Conclusion
  - Perspectives and future work
Web services?

- Middlewares evolution (RPC / MOM) [Alonso, Casati, Kuno, Machiraju].
- Extensive use of standards (XML, SOAP, HTTP(S), SMTP, ...).
- Loose-coupling, easier integration.
On the developer’s side...

- SOAP / WSDL are well accepted.
- Static / Dynamic binding.
- Rich services (ex: Amazon AWS) provide many messages.
- ”Understanding” a service can be tedious.
- Low-level standards, lots of manual processes.
Agenda

Introduction

Web services today
Outline of approach

Framework

Timed business protocols
Temporal compatibility and replace-ability analysis
ServiceMozaic

Conclusion

Conclusion
Perspectives and future work
Capturing conversations

- Business protocols that describe the external behavior [Benatallah, Casati, Toumani].
- Based on deterministic automata.
- A conversation is a complete interaction.
- Easy to understand, well-suited and has formal semantics.
- Expressiveness / complexity trade-off.
- Extensible (ex: time, transactions, policies, ...).
A business protocol (subset of Amazon AWS)
A need for temporal abstractions

- Business protocols only specify the allowed messages orderings.
- There are countless examples (deadlines, soft-locks, ...).
- This abstraction is essential to better "understand" the external behavior of a service.
Research problem and applications

Summary
Take temporal abstractions into account and perform flexible compatibility and replace-ability analysis by using a protocol operators based algebra.

Why?

- Help in making a compliant implementation (ex: against specifications such as RosettaNet).
- Generate adapters in case of mismatches [Hamid Motahari].
- Support evolution.
- Enhanced discovery and dynamic binding.
Research problem and applications

Summary
Take temporal abstractions into account and perform flexible compatibility and replace-ability analysis by using a protocol operators based algebra.

Why?

• Help in making a compliant implementation (ex: against specifications such as RosettaNet).
• Generate adapters in case of mismatches [Hamid Motahari].
• Support evolution.
• Enhanced discovery and dynamic binding.
Introduction

Web services today
Outline of approach

Framework

Timed business protocols
Temporal compatibility and replace-ability analysis
ServiceMozaic

Conclusion

Conclusion
Perspectives and future work
Extended model

- A user-oriented model.
- Introduction of implicit transitions.
- Models temporal availability windows and deadlines.
Formalization

Web services business protocol
\( \mathcal{P} = (\mathcal{S}, s_0, \mathcal{F}, \mathcal{M}, \mathcal{R}) \)

Timed web services business protocol [BDA’05, CAiSE’05 Forum]

- \( \mathcal{M} = \mathcal{M}_e \cup \mathcal{M}_i \)
- For \( \mathcal{R}(s, s', m), m \in \mathcal{M}_i \), we define
  \( \text{Time}(s, m) \rightarrow t \in \mathbb{Q}_{\geq 0} \).
Formalization

Web services business protocol
\[ \mathcal{P} = (\mathcal{S}, s_0, \mathcal{F}, \mathcal{M}, \mathcal{R}) \]

Timed web services business protocol [BDA’05, CAiSE’05 Forum]

- \[ \mathcal{M} = \mathcal{M}_e \cup \mathcal{M}_i \]
- For \( \mathcal{R}(s, s', m), m \in \mathcal{M}_i \), we define
  \[ \text{Time}(s, m) \rightarrow t \in \mathbb{Q}^{\geq 0}. \]
Formalization – cont.

- Deterministic.
- At most 1 implicit outgoing transition per state.
- Deadlocks-free.
- Assumptions:
  - instantaneous transitions
  - time relative to the entrance in a state \( s \)
  - every state is \( \text{reachable} \)
  - no implicit circuits
  - messages semantics is another issue.
Semantics

2 kind of constraints:

- **conversations** – *Linear time*
  \[ a(+) \cdot b(-) \cdot c(+) \]

- **temporal** – *timed traces*
  \[ (a(+), 0) \cdot (b(-), 3) \cdot (c(+), 20) \]

*We focus on observable traces.*
A few lessons from timed automata [Alur, Dill]

Facts

• TA are more expressive and less user-friendly.
• Many decision problems are undecidable, unless you choose adequate subclasses and pay attention to the constraints grammar.
• $\varepsilon$-transitions are a problem, but can sometimes be removed, under certain conditions.

$\rightarrow$ We have mappings and use TA to identify properties on our model.
$\rightarrow$ Interestingly, implicit transitions are not a problem in our case.
A few lessons from timed automata [Alur, Dill]

Facts

- TA are more expressive and less user-friendly.
- Many decision problems are undecidable, unless you choose adequate subclasses and pay attention to the constraints grammar.
- $\varepsilon$-transitions are a problem, but can sometimes be removed, under certain conditions.

$\rightarrow$ We have mappings and use TA to identify properties on our model.
$\rightarrow$ Interestingly, implicit transitions are not a problem in our case.
Agenda

Introduction
Web services today
Outline of approach

Framework
Timed business protocols
Temporal compatibility and replace-ability analysis
ServiceMozaic

Conclusion
Conclusion
Perspectives and future work
Compatibility

2 services can talk to each other
Replaceability

1 service can replace another one
Classes

• Partial or full compatibility.
• Replace-ability:
  • equivalence, subsumption, partial replace-ability
  • w.r.t. client protocol
  • w.r.t. interaction role.

→ the flexibility introduced by these classes is original and needed by the versatility induced by the web.
Example: replace-ability w.r.t. a client protocol
Example: replace-ability w.r.t. a client protocol

start

logged

vehicle
Selection

payment
Estimation

creditApplication

credit
Accepted
 cancelled

login(+)
changeLoginInfo(+)
selectVehicle(+)
estimatePayment(+)
fullCredit(+)
reject(-) accept(-) 15 hours

application
Rejected

credit
Accepted

cancelled
Example: replace-ability w.r.t. a client protocol

1. Start
2. Login
3. Log in
4. Select vehicle
5. Select vehicle
6. Estimate payment
7. Estimate payment
8. Full credit
9. Full credit
10. Reject
11. Reject
12. Accept
13. Accept
14. Logged
15. Change login info
16. Change login info
17. Accepted
18. Cancelled
19. Login
20. Change login info
21. Select vehicle
22. Estimate payment
23. Full credit
24. Reject
25. Accept
26. 15 hours
Example: replace-ability w.r.t. a client protocol
Timed business protocols operators

- Timed compatible composition: $\parallel^{TC}$
- Timed intersection: $\parallel^{TI}$
- Timed difference: $\parallel^{TD}$
- Projection: $[\mathcal{P}_1 \parallel^{TC} \mathcal{P}_2]_{\mathcal{P}_1}$
Example: timed difference

\[ \mathcal{P}_3 = \mathcal{P}_2 \parallel^{TD} \mathcal{P}_1 \]
Characterization

- We can characterize the compatibility and replace-ability classes with these operators.
- Ex: $TR_{P_C}(P_1, P_2)$

$$P_C \parallel^{TC} (P_2 \parallel^{TD} P_1) = \emptyset$$

- Polynomial-time complexity algorithms.
Agenda

Introduction
   Web services today
   Outline of approach

Framework
   Timed business protocols
   Temporal compatibility and replace-ability analysis
   ServiceMozaic

Conclusion
   Conclusion
   Perspectives and future work
Goals

- A model-driven conceptual framework and a CASE toolset.
- Support for design, development and management of web services.
- Technologies: Eclipse + J2EE.
Specificities

• Techniques for analyzing and managing services interactions at the protocol and traces level.
• Re-engineering (ex: protocols mining).
• Scalable development.
• Protocols evolution.
• Execution monitoring support.
Agenda

Introduction
  Web services today
  Outline of approach

Framework
  Timed business protocols
  Temporal compatibility and replace-ability analysis
  ServiceMozaic

Conclusion
  Conclusion
  Perspectives and future work
Related work

- Timed automata [Alur, Dill].
- The (many) ”standardization” efforts.
- Temporal logics and their extensions.
- Work on components in software engineering.
What has been done

Flexible context-oriented model and analysis:

- Temporal extension of the business protocol model [BDA’05, CAiSE’05 Forum].
- Timed operators to characterize the compatibility / replace-ability classes.
- Polynomial-time timed operator algorithms.
- Protocols library (untimed) and editor for the ServiceMozaic platform.
Agenda

Introduction
  Web services today
  Outline of approach

Framework
  Timed business protocols
  Temporal compatibility and replace-ability analysis
  ServiceMozaic

Conclusion
  Perspectives and future work
Perspectives and future work

- A more expressive temporal constraints framework.
- Multi-protocols analysis (open issues).
- Protocol changes management.
- Timed implementations for the ServiceMozaic platform.
- Other abstractions investigations (transactions).
Thanks!