Flexible techniques for analyzing and manipulating web service protocols.

Farouk Toumani

27 June 2006

ISIMA, Université Blaise Pascal – Clermont-Ferrand II
LIMOS, Lab. d'Informatique, de Modélisation et d'Optimisation des Systèmes
Material based on work in ServiceMosaic project.

Collaboration with colleagues and students

(Mr. Julien Ponce (TIMOS, France)

Mr. Hamid Motahari (UNSW, Australia)

Dr. Fabio Casati (HP Labs, USA)

Prof. Boualem Benatallah (UNSW, Australia)

http://servicemosaic.isima.fr)
Agenda

• Web services vision and technologies
• Representing web service protocols
• Analysis and management of web service protocols
• Summary and outlook
Motivations

- Enterprise Application Integration
- Middleware (RPC, MOM, CORBA)
- Distributed information systems

Web services: evolution of current technologies

Automation of inter-organizational business processes

Integration of autonomous and heterogeneous systems
Beyond current technologies

- New integration context
  - Open environment: autonomous systems
  - Large and dynamic integration space
  - Semantic heterogeneity (both data and business processes)

- Limitations of current technologies
  - Inter-organizational interactions (trust, security, transactions)
  - One-to-one mappings between partner systems do not scale
  - Centralized and rigid infrastructures, costly development and maintenance of
    integrated systems
  - Close environment/tightly coupled systems (semantics known from the context)

- Beyond current technologies
  - Autonomous systems
  - Large and dynamic integration space
  - Semantic heterogeneity (both data and business processes)

- Inter-organizational interactions (trust, security, transactions)
Web services

"A software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols." [W3C]
Main characteristics

- Generic interface for service oriented architectures
- Intensive use of standards (SOAP, WSDL, etc)
- Ultimate goal: rapid low-cost development and easy composition
- Loosely coupled integration
Web service technologies

• Service composition
• Service interactions
• Service discovery
• Service description

Specifications and languages providing core functionality of web services
Interoperability Layers

Standard information transportation protocol

Requirements and properties related to a set of message exchanges among two or more partners

Functional properties of services (interfaces and protocols)

Policy specification (e.g., privacy policies) and non-functional properties (e.g., cost, response time, ..)

Interoperability Layers
Interoperability Layers

- **WSDL**: Interface & Protocol Layer
- **WS-***: Policy Layer
- **BPSS**: Core, Components CPA, BPSS
- **ebMS**: (Messaging Service)

**Messaging Layer**
- **SOAP**: Protocol & Interface: WSDL

**Basic Coordination Layer**
- **WS-Coordination, WS-CROR**
- **BTP, WS-Transaction, WS-CROR**

**Business Interface & Protocol Layer**
- **WS-SecurityPolicy, XACML**
- **WS-PolicyAssertions, WS-Policy**

**Interoperability Layers**
Some Observations

- Trade-off: expressivity vs. readability/usability
- Parameters, ...
- Meanings of the parameters, operators effects, negotiation
- Functional and non-functional properties (e.g., QoS, ...)'
- Business protocols'
- Interface'

explicit (essential in autonomous environments)

Making implicit information (as in closed environments)

Services are loosely-coupled and need to be fully specified
Agenda

• Summary and outlook
• Analysis and management of web service protocols
• Representing web service protocols
• Web services vision and technologies
ServiceMosaic project

Joint project with SOC group (UNSW, Sydney) and HP Laboratories (Palo Alto, USA)

F. Toumani - QSL - Nancy, 27 June 2006
Rightsizing is a key issue

• Transactional Implications and Effects

• Time-sensitive Conversations

• Message Choreography

An extended protocol model

Describe external behavior of services
Example: a credit online financing services
Example: a credit online financing service (cont.)
and replaceability between pairs of protocols

Provide an automated support for verification of compatibility

- tools that implement these operators
- key to achieve the benefits
- primitives to analyze and manage protocols

Need for a protocol algebra

Protocol analysis and management

Characterization of different levels of protocol compatibility and replaceability

- Characterization of different levels of protocol compatibility and replaceability
- same set of conversations
- Replaceability: verifying whether two protocols can support the
  correctly based on their protocol definitions
- Compatibility: checking whether two services can interact
  two dimensions of the analysis

⇒ Provide an automated support for verification of compatibility and replaceability between pairs of protocols
Applications and benefits

- Service discovery and composition (e.g., reduce the number of false positives during service discovery)
- Support for static and dynamic binding
- Change support and evolution
- Compliance verification (e.g., with B2B standards)
Formalization

Small is beautiful

A simple adaptation to compare protocols

Useful to deal with service composition

Semantics of interactions: set of complete interaction trees

Semantics of a protocol: set of complete execution trees

Protocol behavior

A protocol schema (i.e., intentional description of service)

Semantics: branching time view

Protocol model: based on a finite state machine

A basic protocol: message choreography + message polarity
Compatibility classes

- Partial compatibility
  
  $P_1$ is partially compatible with $P_2$ if there are some executions of $P_1$ that can interoperate with $P_2$, i.e., if there is at least one possible conversation that can take place.

- Full compatibility
  
  $P_1$ is fully compatible with $P_2$ if all executions of $P_1$ can be understood by $P_2$, i.e., if there is at least one possible conversation that can take place.

- Partial compatibility
  
  $P_1$ is partially compatible with $P_2$ if there are some executions of $P_1$ that can interoperate with $P_2$, i.e., if there is at least one possible conversation that can take place.
Example: a basic online financing services
Example: a simple client protocol
Example: compatibility
Replaceability classes

- Protocol equivalence
  Identifies when two protocols can be interchangeably used in any context and the change is transparent to clients.

- Protocol subsumption
  Identifies when a protocol \( P_1 \) can be transparently used instead of \( P_2 \) (the opposite is not necessarily true).

- Protocol subsumption with respect to a client protocol
  \( P_2 \) can replace \( P_1 \) with respect to a client protocol if \( P_2 \) behaves as \( P_1 \) when \( P_1 \) behaves as \( P_R \).

- Protocol equivalence and subsumption with respect to a client
  Identifies replaceability relations with respect to a certain client protocol.

- Protocol equivalence and subsumption with respect to an interaction role
  \( P_2 \) can replace \( P_1 \) with respect to a role if \( P_2 \) behaves as \( P_1 \) when \( P_1 \) behaves as \( P_R \).
Example: subsumption
Example: equivalence w.r.t. a client protocol
Example: equivalence w.r.t. a client protocol (cont.)
Toward a protocol algebra

A set of operators to manipulate and analyze protocols

Example

- Compatible composition
- Intersection
- Difference

First input protocol that cannot be supported by the second input protocol
Returns a protocol that describes the set of conversations of the common between the input protocols
Returns a protocol that describes the set of conversations that are common between the considered protocols
Returns a protocol that describes all the possible conversations
Example: compatible composition
Example: difference operation
Example: difference operation (cont.)
Characterization of the classes

• Partial compatibility

\[ \text{Partial-compat}(P_1, P_2) \equiv \text{if } P_1 \not\equiv P_2 \text{ then } P_1 \not\succ P_2 \] is not an empty protocol

• Full compatibility

\[ \text{Full-compat}(P_1, P_2) \equiv \text{if } P_1 \not\equiv P_2 \text{ then } P_1 \not\succ P_2 \] is not an empty protocol

• Subsumption

\[ \text{Subs}(P_2, P_1) \equiv \text{if } P_2 \not\prec P_1 \text{ then } P_2 \not\ll P_1 \] is not an empty protocol

• Replaceability with respect to an interaction role

\[ \text{Replace-role}(P_1, P_2) \equiv \text{if } P_1 \not\succ P_2 \text{ then } P_1 \not\ll P_2 \] is not an empty protocol

• Replaceability with respect to a client protocol

\[ \text{Replace-c}(P_1, P_2) \equiv \text{if } P_1 \not\succ P_2 \text{ then } P_1 \not\ll P_2 \] is not an empty protocol

• Equivalence

\[ \text{Equiv}(P_1, P_2) \equiv \text{if } P_1 \not\equiv P_2 \text{ then } P_1 \not\succ P_2 \] is not an empty protocol
Extension to timed protocols

Timed protocol = basic protocol + temporal abstractions

Two main temporal abstractions

C-invoke: temporal window

M-Invoke: expiration

Formal semantics

Interaction semantics: set of timed interaction paths (conversation)

Protocol semantics: set of timed execution paths
Example of a timed protocol

Timed conversation: (login(+), 0); (selectVehicle(+), 0); (T2: selectVehicle(+), 1);

EstimatePayment(+) (T3: EstimatePayment(+), 2); (T4: fullCredit(+), 3);

(T5: accept(-), 4); (C-Invoke(T3<36H), 5);

(T6: cancel(-), 6); (M-Invoke(T3=36H), 7);

(T7: reject(-), 8); (T8: ModifySelection(+), 9);

(T9: reapplication(+), 10); (C-Invoke(T3<24H), 11);

(T1: loggin(+), 12); (T2: selectVehicle(+), 13);

(T3: fullCredit(+), 14);
Definition of operators for analyzing and managing timed protocols

- Definition of new time-sensitive replaceability and compatibility classes
- Definition of new time-sensitive replaceability and compatibility

Analysis and management of timed protocols
Example: time compatible composition

\[S_0 \parallel TC P_3\]

\[S_0 \parallel TC P_2\]
Mapping: timed protocols $\mapsto$ timed automata

- Extensively studied formalism
- clock variables
- State-transition graphs with timing constraints using real-valued
- Formal notation to model behavior of real time systems

Strong link with the theory of timed automata

Timed protocols: decision problems
Mapping to timed automata

T1: a(-)

S0'

T2: b(+)

Timed automata

T3: c(+)

M-invoke(T2 = 10 hours)

XT2 := 0

XT2 < 10

40
Main results

A few lessons from Timed Automata

Closed under complementation

Timed protocols = New subclass of timed automata

Silent transitions strictly increase expressiveness when they reset

Closed under all boolean operations but complementation

Closed under all boolean operations but complementation
Agenda

• Summmary and outlook
  • Analysis and management of web service protocols
  • Representing web service protocols
  • Web services vision and technologies
Protocol discovery

Composite services analysis and synthesis

Analysis of multi-party protocols (consistency analysis)

Trust negotiation and security protocols in Web services

Abstractions

Extension to business protocols augmented with transactional development and interoperability

So far, focus on primitives for facilitating automation of services

Analysis and management of web service protocols

Summary and outlook
Thanks