Modeling and Analyzing Timed Web Services Protocols

Julien Ponge[1,2] – ponge@isima.fr
http://www.isima.fr/ponge/

[1]Laboratoire LIMOS, ISIMA, Clermont-Ferrand, France

[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

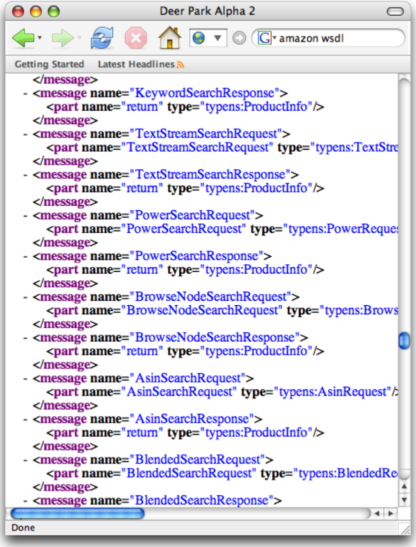
Conclusion
- Conclusion
- Perspectives and future work

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.



```
    </message>
  - <message name="KeywordSearchResponse">
      <part name="return" type="typens:ProductInfo"/>
    </message>
  - <message name="TextStreamSearchRequest">
      <part name="TextStreamSearchRequest" type="typens:TextStre
    </message>
  - <message name="TextStreamSearchResponse">
      <part name="return" type="typens:ProductInfo"/>
    </message>
  - <message name="PowerSearchRequest">
      <part name="PowerSearchRequest" type="typens:PowerReque:
    </message>
  - <message name="PowerSearchResponse">
      <part name="return" type="typens:ProductInfo"/>
    </message>
  - <message name="BrowseNodeSearchRequest">
      <part name="BrowseNodeSearchRequest" type="typens:Brows
    </message>
  - <message name="BrowseNodeSearchResponse">
      <part name="return" type="typens:ProductInfo"/>
    </message>
  - <message name="AsinSearchRequest">
      <part name="AsinSearchRequest" type="typens:AsinRequest"/:
    </message>
  - <message name="AsinSearchResponse">
      <part name="return" type="typens:ProductInfo"/>
    </message>
  - <message name="BlendedSearchRequest">
      <part name="BlendedSearchRequest" type="typens:BlendedRe
    </message>
  - <message name="BlendedSearchResponse">
```

Agenda

Introduction

- Web services today
- Outline of approach

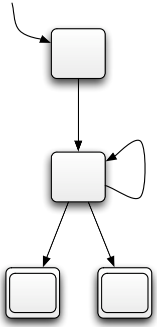
Framework

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

Conclusion

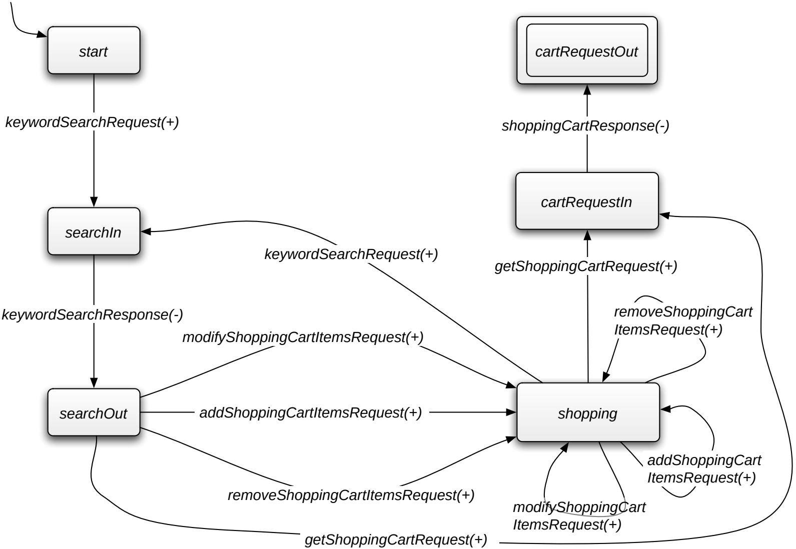
- Conclusion
- Perspectives and future work

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)


start
keywordSearchRequest(+)
searchIn
keywordSearchResponse(-)
searchOut
keywordSearchRequest(+)
modifyShoppingCartItemsRequest(+)
addShoppingCartItemsRequest(+)
removeShoppingCartItemsRequest(+)
getShoppingCartRequest(+)
cartRequestOut
shoppingCartResponse(-)
cartRequestIn
getShoppingCartRequest(+)
removeShoppingCart ItemsRequest(+)
shopping
addShoppingCart ItemsRequest(+)
modifyShoppingCart ItemsRequest(+)

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.

Agenda

Introduction

- Web services today
- Outline of approach

Framework

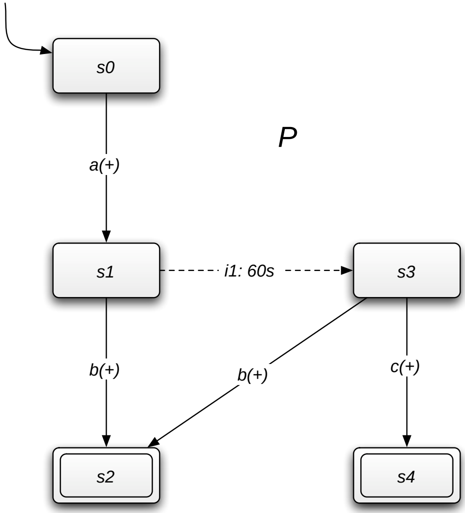
- Timed business protocols
- Temporal compatibility and replace-ability analysis
- ServiceMozaic

Conclusion

- Conclusion
- Perspectives and future work

Extended model

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



Formalization

Web services business protocol

$\mathcal{P} = (\mathcal{S}, s_0, \mathcal{F}, \mathtt{M}, \mathcal{R})$

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

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

Formalization

Web services business protocol

$\mathcal{P} = (\mathcal{S}, s_0, \mathcal{F}, \mathtt{M}, \mathcal{R})$

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

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

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 *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.
- ε-transitions are a problem, but can sometimes be removed, under certain conditions.

⟶ We have mappings and use TA to identify properties on our model.
⟶ 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.
- ε-transitions are a problem, but can sometimes be removed, under certain conditions.

$\longrightarrow$ We have mappings and use TA to identify properties on our model.
$\longrightarrow$ 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

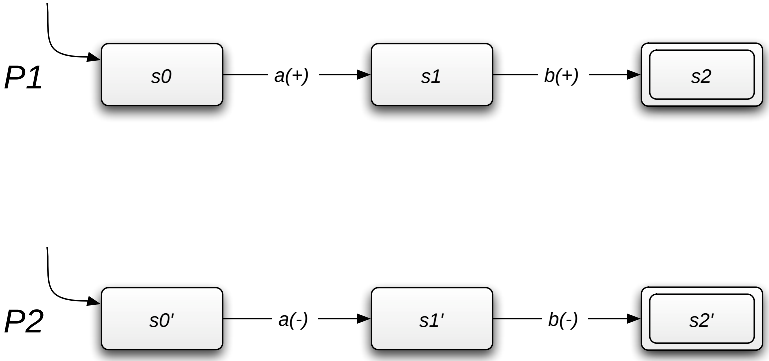
ServiceMozaic

Conclusion

Conclusion

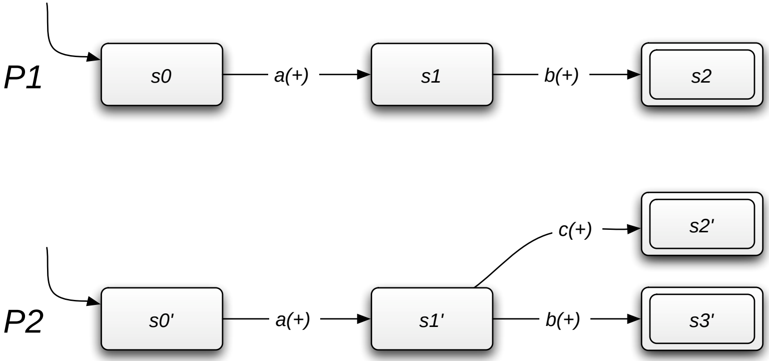
Perspectives and future work

Compatibility



2 services can talk to each other

Replaceability



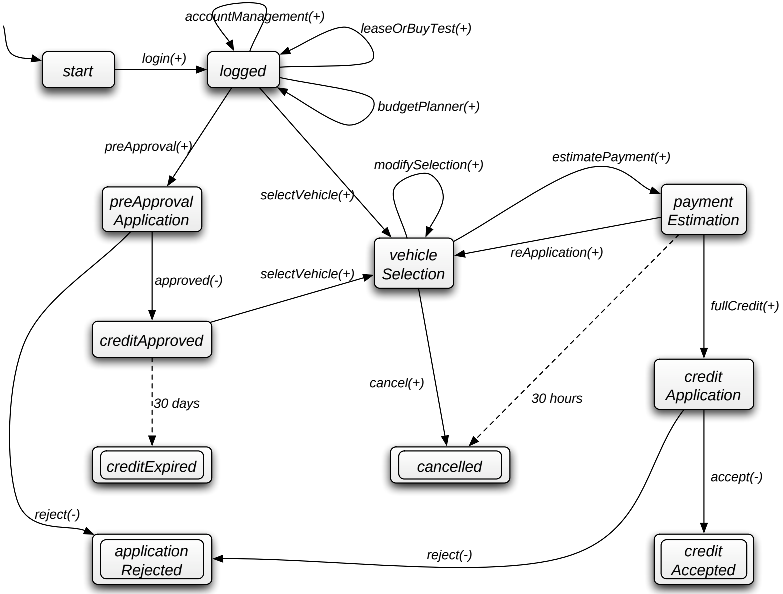
1 service can replace another one

Classes

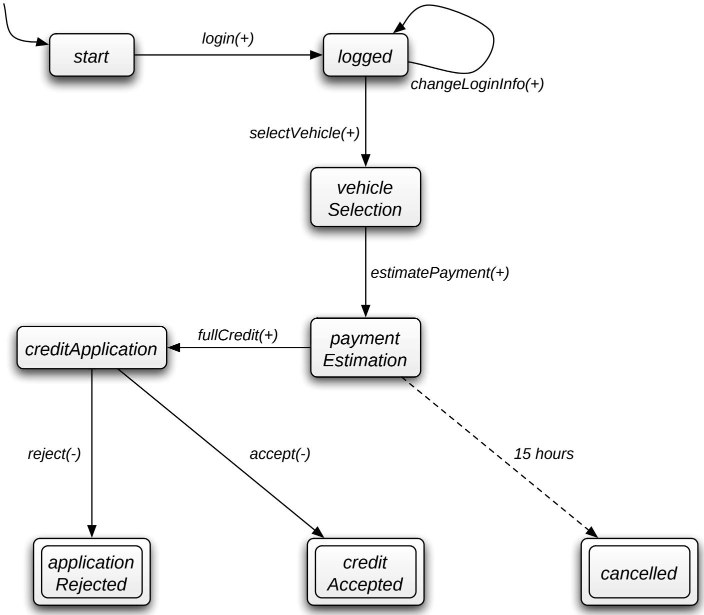
- Partial or full compatibility.
- Replace-ability:
    - equivalence, subsumption, partial replace-ability
    - w.r.t. client protocol
    - w.r.t. interaction role.

⟶ the flexibility introduced by these classes is original and needed by the versatility induced by the web.

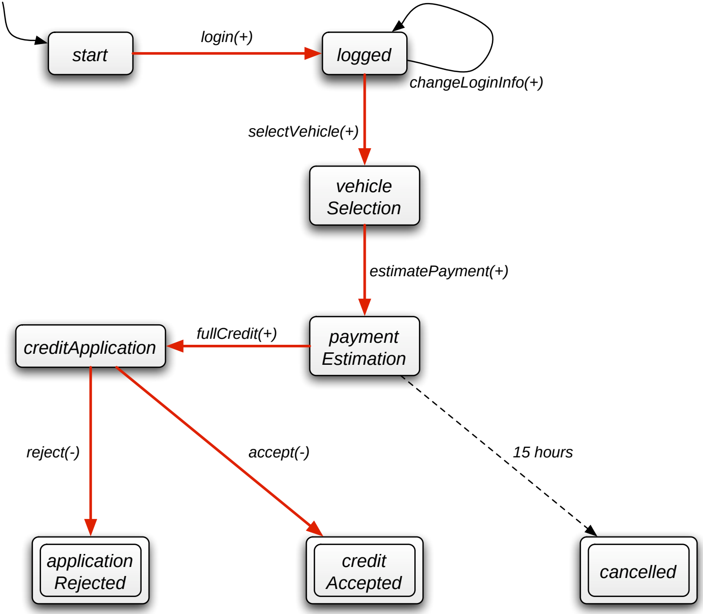
Example: replace-ability w.r.t. a client protocol



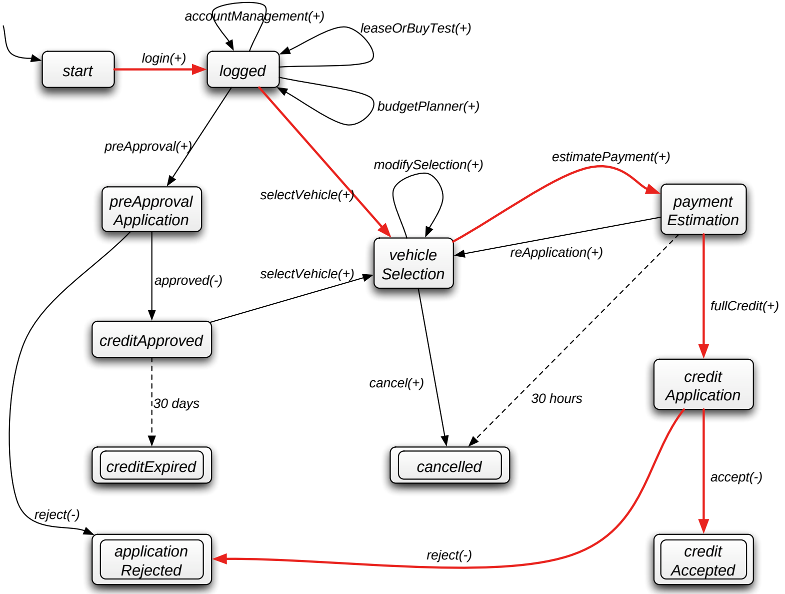
Example: replace-ability w.r.t. a client protocol



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



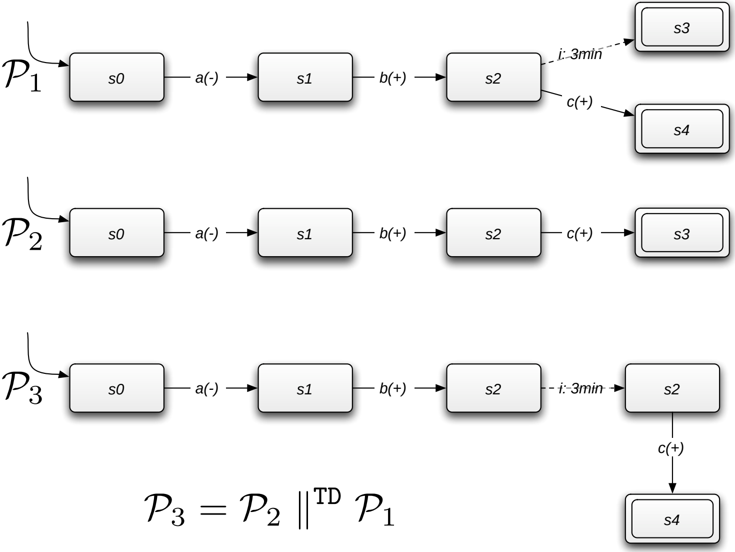
Example: replace-ability w.r.t. a client protocol



Timed business protocols operators

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

Example: timed difference



$$\mathcal{P}_3 = \mathcal{P}_2 \parallel^{\mathtt{TD}} \mathcal{P}_1$$

Characterization

- We can characterize the compatibility and replace-ability classes with these operators.
- Ex: $TRepl_{\mathcal{P}_C}(\mathcal{P}_1, \mathcal{P}_2)$

$$\mathcal{P}_C \parallel^{\texttt{TC}} (\mathcal{P}_2 \parallel^{\texttt{TD}} \mathcal{P}_1) = \emptyset$$

- Polynomial-time complexity algorithms.

Agenda

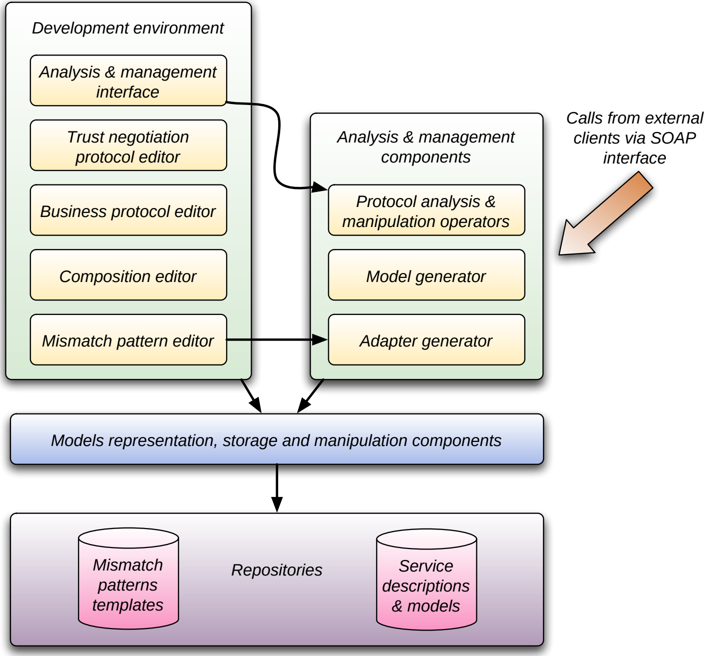
Introduction
- Web services today
- Outline of approach

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.


Development environment
Analysis & management interface
Trust negotiation protocol editor
Business protocol editor
Composition editor
Mismatch pattern editor
Analysis & management components
Protocol analysis & manipulation operators
Model generator
Adapter generator
Calls from external clients via SOAP interface
Models representation, storage and manipulation components
Mismatch patterns templates
Repositories
Service descriptions & models

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

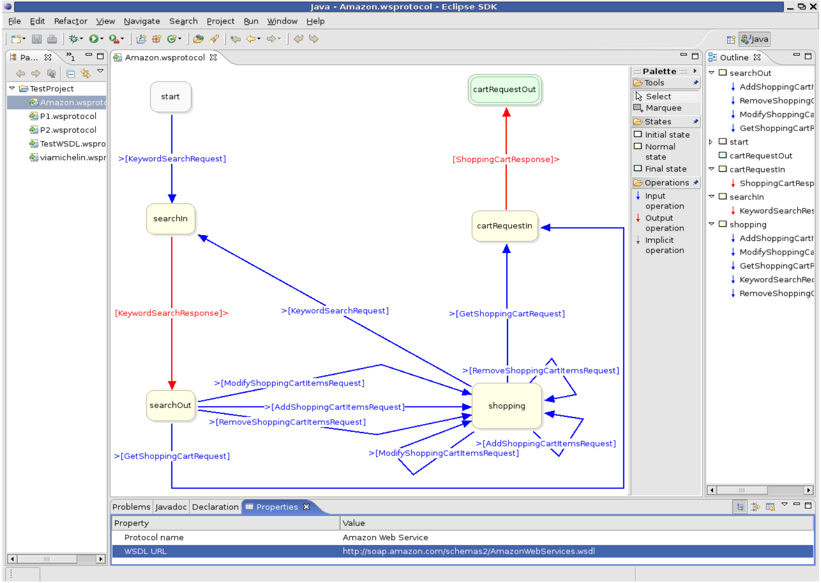
Related work

- 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.



Property	Value
Protocol name	Amazon Web Service
WSDL URL	http://soap.amazon.com/schemas2/AmazonWebServices.wsdl

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

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!