Synthesizing adapters for conversational web-services from their WSDL interface

Luca Cavallaro, Elisabetta Di Nitto, Patrizio Pelliccione, Matteo Pradella, Massimo Tivoli
SOA and dynamic binding

Service Invocation

Service 1

Service 2

Service 3
Our solution to the problem

Service Requester

Adaptation Script

Adaptation Script executor

Automatic Construction

Service Description

Service Description

Service 1

Service 2

S_{sex}

S_{act}
Adaptation scripts

- Adaptation scripts define operation sequences of finite length
Service description

• Services are described in terms of:
  – Operations names and parameters
    • i.e., encoded in terms of WSDL or equivalent for REST
  – Order of operations invocation (protocol)
    • Defined through a *Behavior Protocol Automaton* (BPA)

• Usually only operation names and parameters are provided
Problem: need for a *rich* description

- Usually only operation names and parameters are provided
  - E.g. No BPA in WSDL

- *StrawBerry* discovers BPA from provided description
  - Derive BPA from a partial ordering relation among the possible invocations to the WSDL operations
  - Discovery process is based on testing, black-box and extra-procedural
The StrawBerry method overview

- Saturated Dependencies Automaton Synthesis
- Input/Output Dependencies
- Dependencies Elicitation
  - WSDL
  - Test-cases Generation
  - SOAP envelope
- Dependencies Automaton Refinement Through Testing
- Validated Dependencies Automaton
- Behavior Protocol Automaton
- Behavior Protocol Synthesis
- Error found
- No Error found
- Refinement Through Testing
### Activity 1, dependency elicitation

<table>
<thead>
<tr>
<th>Operation</th>
<th>Input data</th>
<th>Output data</th>
</tr>
</thead>
<tbody>
<tr>
<td>SearchLiric</td>
<td>artist: string</td>
<td>ArrayOf:</td>
</tr>
<tr>
<td></td>
<td>song: string</td>
<td>LyricChecksum: string;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LyricId: int;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SongUrl: string;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ArtistUrl: string;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Artist: string;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Song: string;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SongRank: int;</td>
</tr>
<tr>
<td>SearchLiricText</td>
<td>lyricText: string</td>
<td>ArrayOf:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LyricChecksum: string;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LyricId: int;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SongUrl: string;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ArtistUrl: string;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Artist: string;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Song: string;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SongRank: int;</td>
</tr>
<tr>
<td>GetLiric</td>
<td>LyricId: int;</td>
<td>ArrayOf:</td>
</tr>
<tr>
<td></td>
<td>lyricCheckSum: string;</td>
<td>LyricSong: string;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LyricArtist: string;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LyricUrl: string;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LyricCovertArtUrl: string;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LyricRank: int;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lyric: string;</td>
</tr>
</tbody>
</table>
Activity 2, dependencies automaton synthesis

SearchLyric

SearchLyric.song → SearchLyric.song
SearchLyric.LyricId → GetLyric.LyricId
SearchLyric.SongRank → GetLyric.LyricId
SearchLyric.LyricChecksum → GetLyric.LyricChecksum
SearchLyric.LyricChecksum → SearchLyric.song

start

SearchLyric

…

SearchLyricText

…”

GetLyric

…”

SearchLyric

…”

GetLyric

…”

Checksum

…”
Activity 3, test-cases generation

```xml
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:api="http://api.chartlyrics.com/">
  <soapenv:Header/>
  <soapenv:Body xmlns="http://api.chartlyrics.com/">
    <api:SearchLyric>
      <api:artist>KOVjotMZBEfbeynkhtAviBIEs</api:artist>
      <api:song>HJDSFGhfklskjgHFSKJnee</api:song>
    </api:SearchLyric>
  </soapenv:Body>
</soapenv:Envelope>
```
Activity 4, automaton refinement through testing

- The objective is to either prune false deps. or promote uncertain deps. to certain deps.
  - step1: positive tests in order to prune false deps.
  - step2: negative tests in order to confirm uncertain deps. as certain
  - step3: (experimentally few) remaining uncertain deps. are directly pruned
Activity 2, dependencies automaton synthesis

SearchLyric

SearchLyric.song → SearchLyric.song

SearchLyric.LyricId → GetLyric.LyricId

SearchLyric.SongRank → GetLyric.LyricId

SearchLyric.LyricChecksum → GetLyric.LyricChecksum

SearchLyric.LyricChecksum → SearchLyric.song

GetLyric

GetLyric.Lyric → SearchLyric.song

start

...
BAP synthesis

start

SearchLyric

SearchLyricText

GetLyric

SearchLyric

SearchLyricText

GetLyric

Start

SearchLyric_start

End
Automatic generation of adaptation scripts

Adaptation Generator

Expected service -> Adaptation script Generator

Actual service -> Adaptation script Generator

Executed by Adaptation Script

Executed by SEAMS 2010
Analysis of Expected and candidate Actual services

• Service descriptions analysis is performed by searching in $S_{act}$ a sequence equivalent to that highlighted in $S_{exp}$

• A sequence in $S_{act}$ is equivalent to a sequence in $S_{exp}$ if:
  – Starts in a state $s^i_{act}$ compatible with $s^i_{exp}$
  – Ends in a state $s^f_{act}$ compatible with $s^f_{exp}$
  – Needs as input data a subset of those provided by $s^i_{act}$ … $s^f_{act}$
  – Returns as output parameters at least all the data needed by operations in $s^i_{act}$ … $s^f_{act}$
Service description analysis: Step 1

Start

checkSongExists

checkSongExist_start

searchSong

searchSong_start

checkSongExist__searchSong_start

getSong

checkSongExist__searchSong__searchArtists_start

SearchLyric_start

SearchLyric / SearchLyricText

GetLyric

End
Service description analysis: Step 2

- Two possible search semantics:
  - Synchronous: if an operation in $S_{exp}$ returns some data, no other operation in $S_{exp}$ should be called until those data have been returned.
  - Asynchronous: return data required by an operation in $S_{exp}$ can be received also after subsequent operations have been invoked.
Example

Start

checkSongExists (song; artist) \rightarrow (LyricId, LyricChecksum)

checkSongExist_start

Start

SearchLyric:   artist;song
LyricChecksum; 
SearchLyricText: searchString
LyricId;
SongUrl;
ArtistUrl;
artist;
song;
SongRank;

SearchLyric_start
Example

checkSongExist_start

searchSong (song; artist) \rightarrow (song, artist)

checkSongExist_searchSong_start

SearchLyric_start

SearchLyric

SearchLyricText

SongUrl;
ArtisUrl;
song;
SongRank;

LyricId;
LyricChecksum;
End

GetLyric
Example

checkSongExist
searchSong_start

getSong
(LyricChecksum; LyricId;)

checkSongExist_
searchSong
searchArtists_start

(song; artist
Lyric;LyricUrl)

SearchLyric_start

SearchLyric / SearchLyricText

GetLyric
(LyricChecksum; LyricId;)

song; artist
LyricUrl; Lyric;
LyricCovertArtUrl;LyricRank

End
Adaptation script generation in practice

• Mapping strategies are implemented by bounded model checking
• $S_{\text{exp}}$ and $S_{\text{act}}$ protocols are encoded in the input format of Zot, a SMT-solver based model checker
• The output is a finite length operations sequence which is used as an adaptation script
Conclusion

• Main contribution: A tool-supported methodology to enable service replacement
  – Input: WSDL of two services
  – Output: Adaptation script
• The methodology showed up to be feasible on the example

• Future work:
  – Validate Methodology
    • http://home.dei.polimi.it/cavallaro/sefm10-experiments.html
  – Improve protocol elicitation through heuristics