Generating Service Models by Trace Subsequence Substitution

Miao Du, Jean-Guy Schneider, Cameron Hine, John Grundy, Steven Versteeg (CA Technologies)
Overview

- Context
- Environment Emulation
  - Basic Idea
  - Problems of Existing Emulation Approach
- Novel Framework for Executable Endpoints
  - Similarities and Symmetric Fields
- Evaluation
- Conclusions and Future Work
Enterprise Software Environments

- Workstations
- Directory Server
- Application Server
- Email Server
- Terminal Server
- Terminal
- Laptop
- Smart Phone
- PDA
Clients and Services

Clients

Component under Test

CUT

Services
Thousands of Endpoint Services

CUT

Clients

Component under Test

Services
Environment Emulation

Endpoint Services Modeled "Approximation"

Component Under Test

Emulation Environment
Approach - Emulation Environment

Scalability:
Lightweight models to ensure that *thousands* of endpoints may be emulated *on a single physical machine*

Heterogeneity:
Emulate *as many endpoint types* as needed for testing

Multiple Environment Instances:
Supply *different combinations* of models/configurations to emulation environment

Facilitate Evaluation Activities:
Record exact interactions between CUT and emulation environment
Runtime/playback visualization of interactions [ASE2012]
But how can we best generate executable endpoint models?
Endpoint Model Specification

In previous work:

“programmatically” using a high-level programming language (e.g., Java, Haskell) [ASWEC 2009, ASE 2010]
Too much “low-level” details to consider

High-level model creation
Coloured Petri Nets (CPN) [QoSA 2012]
Non-trivial modeling of I/O structures

Model-driven
(semi-)automatic generation from high-level protocol specification
Endpoint behaviour still needs to be completed manually

Reliance on availability of protocol specifications
Significant human effort required
Observing and recording interactions:

- Observe and **record** the interaction between the Component Under Test and **a “real” endpoint system**.
  
  - If CUT not yet available, replace by system that uses the same protocol.

- Assumption: interaction protocol defined by sequences of **request/response pairs**.

- Problem: recording is only a snapshot, but not a full protocol.
Proposed Framework

Suitable “format” of Traces

Find the “closest” recorded request

Copy “symmetric” information across
Assumption

Having a suitable distance measure and a corresponding translator, “good enough” responses can be synthesized from pre-recorded interaction traces.

What level of abstraction to target?
What distance measure/translator to use?
How effective are they?
What Level of Abstraction?

- Text
- Message Structure
- "Binary"

Encode/Decode:

1. From Text to Message Structure
2. From Message Structure to "Binary"
3. From "Binary" to Message Structure
4. From Message Structure to Text

ToString/FromString:

1. From Text to String
2. From String to Text
3. From Text to "Binary"
4. From "Binary" to Text
Edit Distance as Distance Measure

Needleman-Wunsch algorithm *globally aligns* two sequences of elements. Commonly used in bioinformatics to align protein and/or nucleotide sequences.

Minimizes the “distance” between two sequences by inserting *gaps* at the right places.

“Normalized” Edit Distance as *dissimilarity* measure between two requests.
Sequence Alignment - Example

Two sequences

Where is my computer book?
Where is your computer magazine?

Alignment

Where is my ___ computer book ________?
Where is __your computer ___ magazine?

Result: distance = 16, dissimilarity = 0.28
“Symmetric Fields” – LDAP Example

LDAP request

Message ID: 37
ProtocolOp: searchRequest

ObjectName: cn=Michael SMITH, ou=Administration,
           ou=Corporate,o=DEMOCORP,c=AU

Scope: 0 ( baseObject )

Corresponding LDAP response

Message ID: 37
ProtocolOp: searchResEntry

ObjectName: cn=Michael SMITH, ou=Administration,
           ou=Corporate,o=DEMOCORP,c=AU

Scope: 0 ( baseObject )
Message ID: 37
ProtocolOp: searchResDone
resultCode: success
LDAP – A Working Example

**Incoming request**
Message ID: 18
ProtocolOp: searchRequest
ObjectName: cn=Mal BAIL,ou=Administration,
            ou=Corporate,o=DEMOCORP, c=AU
Scope: 0 ( baseObject )

**Generated response**
Message ID: 18
ProtocolOp: searchResEntry
ObjectName: cn=Mal BAIL,ou=Administration,
            ou=Corporate,o=DEMOCORP, c=AU
Scope: 0 ( baseObject )

Associated response
Message ID: 37
ProtocolOp: searchResEntry
ObjectName: cn=Michael SMITH,ou=Administration,
            ou=Corporate,o=DEMOCORP, c=AU
Scope: 0 ( baseObject )

**“Best” matching request**
Message ID: 37
ProtocolOp: searchRequest
ObjectName: cn=Michael SMITH,ou=Administration,
            ou=Corporate,o=DEMOCORP, c=AU
Scope: 0 ( baseObject )

ProtocolOp: searchResDone
resultCode: success
Evaluation

10 fold Cross-Validation using LDAP (498 request/response pairs) and SOAP (1000 request/response pairs)
Results – SOAP Evaluation

90.7%

9.3%

Identical
Conformant
Results – LDAP Evaluation

Identical: 93.6%
Well-Formed: 3.6%
Conformant: 1.0%
Ill-Formed: 1.8%
Results – SOAP Dissimilarities

![Box plot showing dissimilarity in SOAP](image)
Results – LDAP Dissimilarities
Discussion and Conclusions

- Interaction Traces a promising approach to generate responses to incoming requests
  - Basic approach works for the chosen protocols
  - 99% “good enough” responses

- Avoid human effort in specifying executable endpoint models
  - Shift towards framework configuration

- Lack of:
  - “temporal” properties of interaction protocols
  - Support for “write” operations
  - Consideration of “operation names”
Future Work

- Extend evaluation to more/different kinds of protocols (e.g., ReST, CAM/CAFT, BitTorrent)
- Differentiate structure and payload
- Partitioning of recorded interaction traces
  - Improve efficiency and accuracy
- Consider “temporal” protocol properties
- Different distance measures (e.g., tree distance) and translators
- Explore framework at binary level
- Hook into existing emulator
Acknowledgements

This work is supported by the Australian Research Council (ARC) Linkage project “Large-Scale Emulation for Enterprise Software Systems”.

Many thanks also to CA Technologies for their ongoing collaboration and support of this project.
Generating Service Models by Trace Subsequence Substitution

Jean-Guy Schneider
jschneider@swin.edu.au