Evolving Services Architectures

Petr Hnetynka and Jan Kofron
Charles University, Czech Republic

A joint presentation of the OW2 projects SOFA 2 and Q-ImPrESS.

www.ow2.org.
Overview

- SOFA 2 overview
- Q-ImPrESS overview
- Dependencies between SOFA 2 and Q-ImPrESS
- Short Q-ImPreSS tools demo
SOFA 2 Basic Info

- [http://sofa.ow2.org/](http://sofa.ow2.org/)
- Second generation of the SOFA project
- OW2 (formerly ObjectWeb) project since 2003

- SOFA 2 is a component system offering
  - hierarchical components
  - formal specification and verification of component behavior
  - transparently distributed run-time environment
  - many other features
SOFA 2 Facts (Ohloh)

Ohloh Analysis Summary

- Mostly written in Java
- Mature, well-established codebase
- Increasing year-over-year development activity
- Large, active development team
- Well-commented source code

Lines of Code

![Graph showing lines of code over time for SOFA at OW2 and SOFA 2, with categories for blanks, comments, and code. The graph shows an increase in code and comments over the years 2000 to 2010.]
SOFA 2 Features

- Component model defined via a meta-model
  - EMF
  - generated repository and tools
- Components have separated type and implementation
  - necessary for supporting product line development
- Explicit connectors between components
  - multiple communication styles
  - supporting extra-functional properties
SOFA 2 Features (cont.)

- Formal specification of component behavior
  - using behavior protocols
    - a type of a process algebra
  - verification of
    - component composition
    - specification vs. actual implementation
SOFA 2 Features (cont.)

```java
component CardReader {
  types {
    states = {CARD_READER_ENABLED, CARD_READER_DISABLED}
  }

  vars {
    states state = CARD_READER_ENABLED
  }

  provided {
    (?CardReader.enable() + ?CardReader.disable() ) |*
  }

  reactions {
    CardReader.enable() {
      state <- CARD_READER_ENABLED
    }
    CardReader.disable() {
      state <- CARD_READER_DISABLED
    }
  }

  threads {
    T1: 
      while (?) {
        if (state == CARD_READER_ENABLED) {
          !CardReaderCallback.CreditCardScanned();
          while (!CardReaderCallback.PINEntered() == FAIL) { NULL }
        }
      }
  }
}
```
SOFA 2 Development Tools

1. Set of command line tools
2. Eclipse-based IDE
3. Runtime-monitoring tool

SOFA IDE
MConsole

www.ow2.org.
SOFA 2 Implementation

- Implementation in Java
  - tools
  - run-time
  - demos
Q-ImPrESS

• “Quality Impact Prediction for Evolving Service-Oriented Software”
• Seventh Framework Programme
• Three-year (2008-2010) STREP project
• http://www.q-impress.eu
• 9 partners – industrial:
  • ABB AG, Germany
  • itemis GmbH, Germany
  • Softeco Sismat, Italy
  • Ericsson Nikola Tesla, Croatia
Project Goal

Define a new service engineering method to create and evolve service-oriented software with predictable end-to-end quality
Example Q-ImPrESS Application
Example Q-ImPrESS Application

SOA? QoS?

Legacy Application

Java C / C++
Example Q-ImPrESS Application

SOA? QoS?

Legacy Application

Java
C / C++
Example Q-ImPrESS Application
Example Q-ImPrESS Application

SOA? QoS?

10 ms

Legacy Application

Java C / C++
Example Q-ImPrESS Application
Example Q-ImPrESS Application

20,000 €

0.01%

10 ms

10 ms

SOA? QoS?

Legacy Application

Java

C / C++


www.ow2.org.
Example Q-ImPrESS Application

Service 2

0.01%
0.10%

SOA? QoS?

Legacy Application

Java
C / C++

20.000 €
25.000 €

10 ms
5 ms

Service 1

Service 2

20.000 €
25.000 €

Service 1

Service 2

Java
C / C++
Method Overview

Q-ImPrESS IDE

Legacy Code (Java / C++)

System Architecture

Service Architecture Model

Evolution Alternative #1

Evolution Alternative #n

Best Evolution Alternative

Legend:
- Tool-supported manual action
- Tool-supported automatic action
- Development iterations

Applied at
ABB
ERICSSON
Ericsson Nikola Tesla

Best Evolution Alternative Implementation

www.ow2.org.
Method Overview

www.ow2.org.
What Reverse Engineering Does
What Reverse Engineering Does

Reverse Engineering

C/C++, Java & Delphi Code → extendable
What Reverse Engineering Does

- C/C++, Java & Delphi Code \(\rightarrow\) extendable
- Component-Based Software Architectures
- Component Behaviour
- Trace Model
What is Recognised?

Components
Boundaries
What is Recognised?

Components
Boundaries
Their related classes
What is Recognised?

Components
Boundaries
Their related classes

Interfaces

www.ow2.org.
What is Recognised?

Components
Boundaries
Their related classes
Composite Components

Interfaces

Components Diagram:
- MyClass.java
- Another.java
- Converter.java

Interfaces Diagram:
- BClass.java
- Util.java
- Con.java
What is Recognised?

Components
Boundaries
Their related classes
Composite Components

Interfaces
Connectors

Components:
- MyClass.java
- Another.java
- Converter.java

Interfaces:
- Calc.java
- Util.java
- Con.java

Connectors:
What is Recognised?

Components
- Boundaries
- Their related classes
- Composite Components

Interfaces
- Connectors
- Additionally:
  - Trace & Behaviour Model

Components Boundaries Their related classes Composite Components Interfaces Connectors Additionally:

Target Model: Q-ImPrESS Software Architecture Model (SAM)
Specific Models

• Once the Service Architecture Model (SAM) is available
  • either via RE or designed as new system
• it is used for transformation to specific models (PCM, Klaper, SOFA) for specific analyses
Q-ImPrESS and SOFA

- SOFA is used for checking the consistency between model and implementation
  - transparent transformation into SOFA TBP
  - verification of primitive components
- Java PathFinder is used
- Correspondence of observable behavior
  - accepted and fired service requests
- at the model and implementation level is verified
Demo – Scenario

• Simple component (a Java class)
  • corresponding protocol – TBP
Demo – Scenario

• Simple component (a Java class)
  • corresponding protocol – TBP

1. Running the tool on corresponding pair
  • no error is found
Demo – Scenario

• Simple component (a Java class)
  • corresponding protocol – TBP

1. Running the tool on corresponding pair
  • no error is found

2. Modifying the sources
  • inconsistency is discovered and reported
Demo – Scenario

• Simple component (a Java class)
  • corresponding protocol – TBP

1. Running the tool on corresponding pair
  • no error is found

2. Modifying the sources
  • inconsistency is discovered and reported

3. The inconsistency is fixed
  • correspondence re-established
Conclusion

• [http://sofa.ow2.org/](http://sofa.ow2.org/)
  • [http://twitter.com/#!/sofaproject](http://twitter.com/#!/sofaproject)

• [http://www.q-impress.eu](http://www.q-impress.eu)
  • The method applied on case studies of ABB and ENT