A Generic Gateway for Testing Heterogeneous Components in Acceptance Testing Tools

Jessica Díaz
Agustín Yagüe
Pedro P. Alarcón
Juan Garbajosa

System and Software Technology Group
http://syst.eui.upm.es
Universidad Politécnica de Madrid
Technical University of Madrid
Contents

- Motivation
- Objectives
- Gateways in acceptance testing tools
- Approaching and defining a generic gateway
- Validation of the approach
- Conclusions
Motivation

Usual system testing structure

Acceptance Testing Tool

Gateway

Gateway

Gateway

SUT

C1

C2

C3

Cn
Motivation

A test case: a home network with heterogeneous components

Motivation

Objectives

Gateways

Approaching and defining

Validation

Conclusion

A Generic Gateway for Testing Heterogeneous Components in Acceptance Testing Tools
Motivation

Acceptance Testing Tools and Systems Under Test (SUT) require a gateway to set up the communication link

=> interoperability

SUTs: large systems, systems of systems, have heterogeneous components including heterogeneous platforms and networks

=> interoperability problem
Objectives

To show the interest of defining a generic gateway

To present the design of this gateway comprising

- A single access point to SUT components
- Easy adaptation to heterogeneous SUT interfaces
Objectives

Motivation

Objectives

Gateways

Approaching and defining

Validation

Conclusion

A Generic Gateway for Testing Heterogeneous Components in Acceptance Testing Tools
Gateways in acceptance testing tools

- Motivation: Execution of test suites using TTCN-3. A SUT Adapter must be implemented for each specific SUT.
Gateways in acceptance testing tools

- TOPEN: supports validation, operation and monitoring complex systems. SUT specific gateways must be implemented.
Gateways in acceptance testing tools

- **Motivation**
- **Objectives**
- **Gateways**
- **Approaching and defining**
- **Validation**
- **Conclusion**

- **Gateways**
  - **Fit/FitNesse**: mixtures code for supporting communication between testing tables and SUT.
  - **EasyAccept**: Java Façade for supporting communication between testing scripts and SUT.
  - **Selenium**: specific tool for Web applications
Approaching a generic gateway

Gateway as a central component
Approaching a generic gateway

A number of issues to go through

- Technology analysis
  - Interoperability and Middleware technologies
  - Open Service Gateway initiative (OSGi specifications)

- Design of a generic gateway
  - OSGI as the baseline for a generic (service oriented) gateway
  - Variation points for adapting to specific SUTs
Interoperability and Middleware Technologies

- Domain specific technologies
  - e.g. home network protocols such as HAVi, FireWire IEEE 1394, Konnex, LonWorks, X.10
- UPnP, Jini
- CORBA
- Service-Oriented Computing (SOC)
  - Web Services (WS)
  - Device Profile for Web Services (WS-DP or DPWS)
  - OSGi (Open Service Gateway initiative)
Open Service Gateway initiative

- OSGi as OSGi specifications
- The OSGi Service Platform specifies a layered architecture for dynamic components based systems
Module layer defines how the bundles interact, package sharing, native code, and how the dependencies are solved.

Life Cycle Layer defines an API for start, stop, update and uninstallation of bundles and resolves dependencies.

Service Layer contains a registry where services are published:
- Separate the contract from the implementation
- Dynamically discover and bind available implementations
- Components are reusable (not coupled to)

OSGI can be regarded as a service virtual machine.
Open Service Gateway initiative

Benefits:
- Heterogeneous networks integration
- Testing is simplified by the service model
- Services are much less coupled than objects
- Dynamism: explicit support for incorporating new services during execution
- Security, scalability

OSGi implementations:
- Commercial: IBM, ProSyst, Gatespace, Siemens, Hitachi, Samsung, Espial, Atinav, …
- Open Source: Apache Felix, Eclipse Equinox, Knopflerfish
OSGi middleware is a suitable technology to implement a generic gateway.

Communication interface between acceptance testing tools and the generic gateway is implemented via SOAP.

Communication interface between the generic gateway and the SUT is specific for each system component.
Gateway layered view

**Motivation**

**Objectives**

**Gateways**

**Approach and define**

**Validation**

**Conclusion**

**Gateway layered view**

- **Acceptance Testing Tool**
  - Java VM / .NET
  - Operating System
  - Driver
  - Driver
  - Driver
  - Hardware

- **Generic Gateway**
  - Bundle SOAP
  - Bundle SUT
  - OSGi Framework
  - Java VM
  - Operating System
  - Driver
  - Driver
  - Driver
  - Hardware

- **System Under Test**
  - Operating System
  - Driver
  - Hardware
Gateway design guidelines

- SOAP bundle implements the communication interface with the testing tool
- ControlService bundle listens to service requests and locates the implementation of the specific service.
- SUT-Driver bundles provide access to native devices, i.e., they implement particular communication interface (e.g. X.10)
Gateway architectural framework

Motivation

Objectives

Gateways

Approach and define

Validation

Conclusion
Architectural framework variation points

Bundles linked to specific drivers and device functionalities will be subject to modification
Validating the gateway

TOPEN test system applied to home domain

The communication interface between TOPEN and the home gateway is implemented via SOAP.
A simple rice cooker example

Specifically, TOPEN has been applied for testing a rice cooker.

The communication interface between the home gateway and the rice cooker simulator is implemented via IP based sockets technology.
Variation points

- The implementation guidelines are based on the generic gateway design.
- SUT-SocketsBundle supports the sockets communication
- RiceCookerService bundle implements some specific features of the ricecooker
- The rest of bundles do not require changes
Comparing some results

TTWorkbench (TTCN-3) – the SUT Adapter is the entity for adapting to specific SUTs

Motivation

Objectives

Gateways

Approaching and defining

Validation

Conclusion
Comparing some results

- TOPEN and the generic gateway
  - TOPEN core implements the SUT behaviour
  - The generic gateway implements the communication functionalities

**Motivation**

**Objectives**

**Gateways**

**Approaching and defining**

**Validation**

**Conclusion**
Experimental results

Comparing the use of the generic gateway with respect other approaches:

```java
Code 1 TTCN-3 SUT Adapter implementation.
+import com.testingtech.ttcn.tri.TestAdapter;
+import org.etsi.ttcn.tri.TriCommunicationSA;

class MyTestAdapter extends TestAdapter implements TriCommunicationSA{
    @Constructor for the MyTestAdapter object*/
    public MyTestAdapter() {
        super();
    }
    /*This operation is called by the TE immediately before the execution of any test case.*/
    public TriStatus triExecuteTestcase(TriTestCaseId testcase, TriPortIdList tsIList) {
        [...]}
    } /*Maps a test component port to a test system interface port.*/
    public TriStatus triMap(TriPortId cmpPortId, TriPortId tsIPortId) {
        [...]}
    } /*This operation is called by the TE when it executes a TTCN-3 send operation on a component port*/
    public TriStatus triSend(TriComponentId cmpId, TriPortId tsIPortId, TriAddress address, TriMessage sendMessage) {
        [...]}
    }
```

Motivation

Objectives

Gateways

Approaching and defining

Validation

Conclusion
Experimental results

Motivation

Objectives

Gateways

Approaching and defining

Validation

Conclusion

Code 2 Generic gateway approach implementation

```java
public interface RiceCookerService {
    public void start();
    public void finish();
    public Value get (Attribute att);
    public void set (Attribute att, Value val);
}
```

```java
public class RiceCookerServiceImpl implements RiceCookerService{
    public void start(){
        [...]
        sendCommand("start")
    }
    [...]
}
```

```
+ import java.net.Socket;
+ public class SUT-SocketBundle implements Driver{
+     public void connect() throws java.net.SocketTimeoutException {
+         [...]
+         socketServ = new ServerSocket(port);
+         socketCliente = socketServ.accept();
+         [...]
+     }
+     public void disconnect() throws java.net.SocketTimeoutException {
+         [...]
+     }
+     public void sendCommand(String command) {
+         [...]
+     }
+ }
```
Conclusions

- A generic gateway to communicate acceptance testing tools and SUTs has been designed
- The gateway is easily adaptable to specific SUTs
  - complex SUTs with heterogeneous components are supported
- OSGi has been chosen for the gateway design and implementation
- Variation points have been identified
- Variation points are associated to specific bundles that implement particular device interfaces

Future Work:
- Make this solution general for other other testing tools
- Produce components for several domains