Are your interfaces used as expected?

October 19th, 2016
Validate D-Bus messages with franca deployment models and EB solys

Torsten Mosis
Software Architect (Elektrobit Automotive GmbH), GENIVI Alliance
Are your interfaces used as expected?
Developing SW components in large projects

Service Perspective:
Provided interfaces are validated with unit tests, in a way the service expects its calls

Client Perspective:
The implementation of required interfaces are usually mocked until the concrete implementation is available
Composing the software system

What happens:
Although all components have been tested carefully (possibly test-driven) the software runs into trouble when constructing the single components into a larger system.

Typical errors:
- Calls in the wrong order/sequence
- Pre-conditions were not fulfilled
- Post-conditions were not fulfilled
- Calls with wrong range of values
- Call causes performance drawback
- Service called by not-authorized client
- …
Composing the software system

Root cause:
Obviously there are some rules or conventions beyond the static description (e.g. method signatures, data types) of interfaces. But these rules are usually not defined at all or they are documented alongside the interface definition, e.g. in requirement or design documents.

Proposal for solution:
Define **formalized rules (contracts)** of the interfaces and **check** the **compliance** of their implementation at runtime.
Classification of Contract-aware components
Classification of interface contracts

Classification of interface contracts

Franca IDL is used to define interfaces and data structures. Documentation and IPC binding (e.g. Common API) is generated from the model.

Franca Contracts are provided to define state machines and sequence logic.

Currently no tooling available for behavior and quality of service aspects. Neither for specification nor for validation.

Design-by-Contract extension for Franca Interfaces
Define Design-by-Contract methods with a Franca Deployment Model

```java
specification org.franca.depl.dbc.DesignByContract {

    for interfaces {
        // Define the clients which are allowed to invoke the methods of this interface
        AuthorizedClients : String[] ;
        // Allows to define global variables or simple expressions, that can be used in other rules
        GlobalDef : String[] ( optional ) ;
    }

    for methods {
        // max Response time to a given request in milliseconds
        MaxResponseTime : Integer ( default: 50 ) ;
    }

    for arguments {
        // Set a pre-condition, applicable for in-arguments only
        Pre : String ( optional ) ;
        // Set a post-condition, applicable for out-arguments only
        Post : String ( optional ) ;
        // Can be used to store a value, that is used later
        Set : String ( optional ) ;
        // Check the value of this argument against the value of a previous value call, applicable for out-arguments only
        Old : String ( optional ) ;
    }
}
```
Define Design-by-Contract methods with a Franca Deployment Model

```java
specification org.franca.depl.dbc.DesignByContract {
    for interfaces {
        // Define the clients which are allowed to invoke the methods of this interface
        AuthorizedClients : String[];
        // Allows to define global variables or simple expressions, that can be used in other rules
        GlobalDef : String[] (optional);
    }

    for methods {
        // Max Response time to a given request in milliseconds
        MaxResponseTime : Integer (default: 50);
    }

    for arguments {
        // Set a pre-condition, applicable for in-arguments only
        Pre : String (optional);
        // Set a post-condition, applicable for out-arguments only
        Post : String (optional);
        // Can be used to store a value, that is used later
        Set : String (optional);
        // Check the value of this argument against the value of a previous value call, applicable for out-arguments only
        Old : String (optional);
    }
}
```

**Behavior Rules:**
You specify pre-conditions (requires) and post-conditions (ensures) by the usage of your method parameters.

**Helper Rules:**
Define expressions, that can be used in other rules.

**Quality of Service Rules:**
You need to specify the clients, that are allowed to use the interface or you guarantee a maximum response time for the given request (default is 50ms).

**Copyright © GENIVI Alliance 2016 | Oct 2016 |**
Define Design-by-Contract methods with a Franca Deployment Model

```java
class specification org.franca.depl.dbc.DesignByContract {
    for interfaces {
        // Define the clients which are allowed to invoke the methods of this interface
        AuthorizedClients: String[];
        // Allows to define global variables or simple expressions, that can be used in other rules
        GlobalDef: String[] (optional);
    } // for interfaces

    for methods {
        // max Response time to a given request in milliseconds
        MaxResponseTime: Integer (default: 50);
    } // for methods

    for arguments {
        // Set a pre-condition, applicable for in-arguments only
        Pre: String (optional);
        // Set a post-condition, applicable for out-arguments only
        Post: String (optional);
        // Can be used to store a value, that is used later
        Set: String (optional);
        // Check the value of this argument against the value of a previous value call, applicable for out-arguments only
        Old: String (optional);
    } // for arguments
}
```

Rules of type String are Xtend (Java) expression and of return type boolean. The expression can refer to every method argument, all definitions in "GlobalDef", the "Old" value and the entire Java API.
Apply the contract to a concrete franca interface (fidl)

import "navigation/navigationcore/LocationInput.fidl"
import "classpath:/model/spec/DesignByContract.fdepl"

define org.franca.depl.dbc.DesignByContract for interface org.genivi.navigationcore.LocationInput {

    AuthorizedClients = { "hmi" }

    GlobalDef = {
        'var int globalLocationInputHandler'
    }

    method search {
        MaxResponseTime = 1
        in {
            locationInputHandle { Pre = "locationInputHandle == globalLocationInputHandler" }
            maxWindowSize { Pre = "maxWindowSize <= 10" }
            inputString { Pre = "inputString.length < 15" }
        }
    }

    method createLocationInput {
        in {
            sessionHandle { Pre = "sessionHandle > 0" }
        }
        out {
            locationInputHandle {
                Old = "locationInputHandle == Old + 1"
                Set = "globalLocationInputHandler = locationInputHandle"
            }
        }
    }
}
Apply the contract to a concrete franca interface (fidl)

import "navigation/navigationcore/LocationInput.fidl"
import "classpath:/model/spec/DesignByContract.fdepl"

define org.franca.depl.dbc.DesignByContract for interface org.genivi.navigationcore.LocationInput {

  AuthorizedClients = { "hmi" }
}

GlobalDef = {

  "var int globalLocationInputHandler"
}

method search {

  MaxResponseTime = 1
  in {
    locationInputHandle { Pre = "locationInputHandle == globalLocationInputHandler" }
    maxWindowSize { Pre = "maxWindowSize <= 10" }
    inputString { Pre = "inputString.length < 15" }
  }
}

method createLocationInput {

  in {
    sessionHandle { Pre = "sessionHandle > 0" }
  }

  out {
    locationInputHandle {
      Old = "locationInputHandle == Old + 1"
      Set = "globalLocationInputHandler = locationInputHandle"
    }
  }
}
Design-by-Contract violations visualized in EB solys
Demo
Benefits using Franca Deployment Models as specification tooling

- Franca is **widely-used** in Genivi Expert Groups
- Can be **applied** to existing interfaces **immediately**
- Franca is **technology agnostic** and thus independent regarding programming language and IPC
- Designed for intercepting interface elements
- Deployment models **enrich the origin interfaces**, but do not change them
- Can be **customized** and **extended** to project needs
Benefits using EB solys as validation tooling

- **D-Bus Monitor** (target agent plug-in) is already available
- Automatic D-Bus to **Franca mapping** available
- Can easily be **adapted** to the **projects** needs:
  - Target agent framework and its plug-ins (e.g. D-Bus, DLT, resource monitor) will be **open sourced shortly**
  - A **free** version of **EB solys** will be available Q1/2017
- Designed for **data correlation** (QoS rules)
- Built-in **script engine** allows to hook into validation procedure
- Can also be used in **batch mode** (e.g. in continuous integration tooling)
Download and try it on your own

Note: The Design-by-Contract Extension is a proof-of-concept.

- Franca (incl. Franca Deployment Models)
  - https://github.com/franca/franca
- Design-by-Contract Extension
  - https://github.com/Elektrobit/franca-deployment-model-dbc
  - https://github.com/Elektrobit/franca-deployment-model-dbc-validator

EB solys

Contact us to get an evaluation license of EB solys as long as the free version is not released.
Thank you!
Visit us at the GENIVI Showcase

Visit GENIVI at http://www.genivi.org or http://projects.genivi.org
Contact me: torsten.mosis@elektrobit.com