



# Callista Enterprise

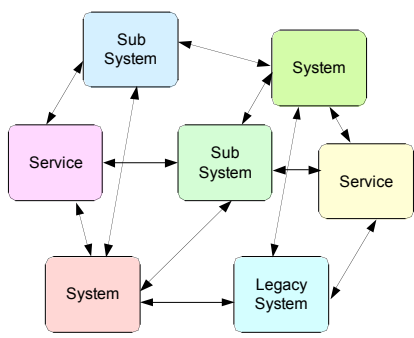
## Test Driven ESB Development

Sofia Jonsson  
sofia.jonsson@callistaenterprise.se

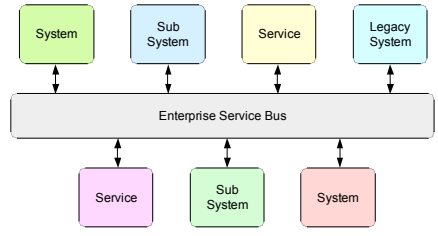


## The Enterprise Service Bus


Point to Point



ESB



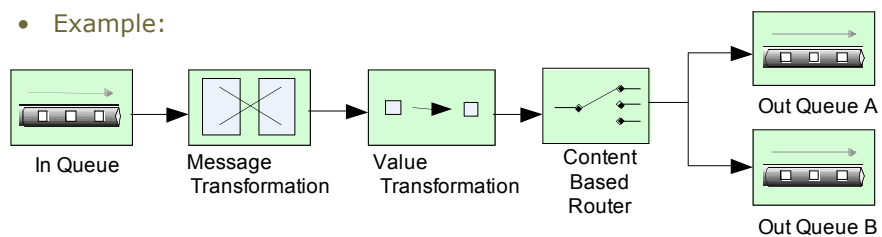
Cadec 2007, Test Driven ESB Development, Slide 2  
Copyright 2007, Callista Enterprise AB



## ESB Development

- Configuring/implementing ESB **message flows**.
- Each flow consists of multiple **nodes** with functionality for e.g.:
  - Message Routing
  - Message Transformation
  - Validation
  - Logging
  - Error handling

- Example:



Cadec 2007, Test Driven ESB Development, Slide 3  
Copyright 2007, Callista Enterprise AB



## Testing in an ESB Environment

- Services, sub systems and systems are often tested early
  - In isolation
- The message flows are often tested late
  - During integration or system test phase
  - Not tested in isolation

Cadec 2007, Test Driven ESB Development, Slide 4  
Copyright 2007, Callista Enterprise AB



## Why Test ESB Code?

- The flows often contain mission critical functionality that is difficult to change.
- Errors in the flows can be particularly hard to track down:
  - No GUI
  - Difficult to simulate all types of behaviour
  - Unclear responsibility

Cadec 2007, Test Driven ESB Development, Slide 5  
Copyright 2007, Callista Enterprise AB



## Test Driven Development

- TDD deals with:
  - "Change is the only constant".
  - Errors/changes are more expensive the later they're identified.
- TDD Key Concepts:
  - Tests are written up front.
  - Test suites are automated.
  - Units are tested in isolation.

Cadec 2007, Test Driven ESB Development, Slide 6  
Copyright 2007, Callista Enterprise AB



## Why Test Driven ESB Development - Summary

ESB flows contain a lot of functionality.

- ☑ TDD can help test that functionality in isolation.

ESB flows are often tested late.

- ☑ TDD means to write tests upfront.

ESB flows often contain mission critical functionality that is difficult to change.

- ☑ Having an automated unit test suite asserts quality when changes are applied.

Errors in the ESB flows can be particularly hard to track down.

- ☑ TDD assures that errors are detected as early as possible.

Cadec 2007, Test Driven ESB Development, Slide 7  
Copyright 2007, Callista Enterprise AB



## Test Driven ESB Development - the Challenge

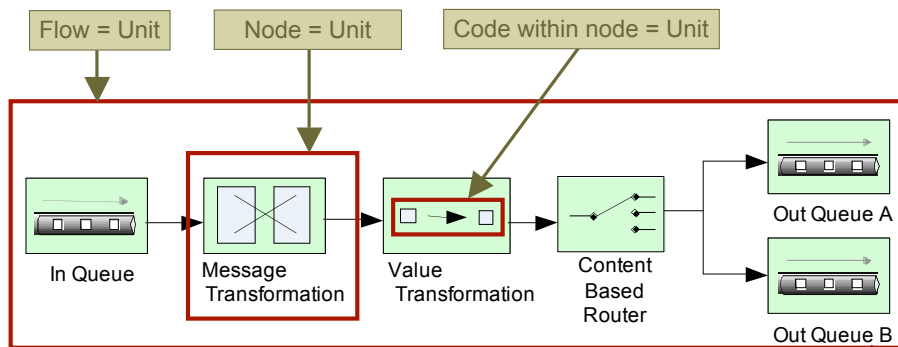
- Varied environment
  - Product level
  - Flow level
  - Node level
- Asynchronous communication
  - Tests typically handle request – reply
  - Timeout issues
- Lack of tool support
  - Environments without good support for testing
  - Few specialized tools available

Cadec 2007, Test Driven ESB Development, Slide 8  
Copyright 2007, Callista Enterprise AB



## Defining the Units

- What should the tests cover?
  - Code + configuration
  - Own code in isolation

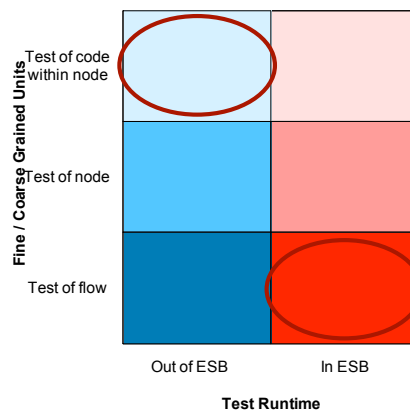


Cadec 2007, Test Driven ESB Development, Slide 9  
Copyright 2007, Callista Enterprise AB

CALLISTA

## Unit Granularity vs. Test Runtime

- Out of ESB "container"
  - Test of code within node
  - Test of node
  - Test of flow
- In ESB "container"
  - Test of code within node
  - Test of node
  - Test of flow



- Be pragmatic!
  - Find out what works in your environment

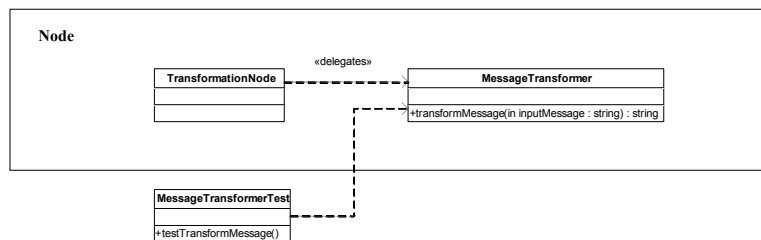
Cadec 2007, Test Driven ESB Development, Slide 10  
Copyright 2007, Callista Enterprise AB

CALLISTA

## Isolated Test of Code Within Node



- Characteristics:
  - Similar to “ordinary” unit testing
  - Typically covers code and not configuration
  - Tests can be run outside of the ESB
  - Delegation can be used within a node to improve testability



Cadec 2007, Test Driven ESB Development, Slide 11  
Copyright 2007, Callista Enterprise AB



## Isolated Test of Code Within Node



- Pros:
  - Fast runtime
  - Fine grained tests
  - Isolation of own code
  - No asynchronous issues
- Cons:
  - Variations at node level can cause problems
    - Need tests in different languages
    - Lack of unit test support
    - Graphical modelled nodes cannot be tested
  - Difficult to mock some products' APIs.
  - Expensive to cover all code

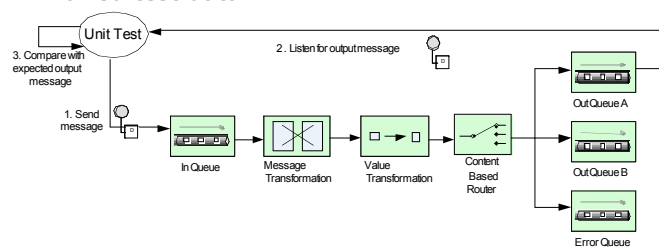
Cadec 2007, Test Driven ESB Development, Slide 12  
Copyright 2007, Callista Enterprise AB



## Test of flow in ESB



- Characteristics:
  - Covers both code and configuration
  - Typical "black box" test
    - Handles input and expected output
    - Reusable test cases
    - Varied test data



Cadec 2007, Test Driven ESB Development, Slide 13  
Copyright 2007, Callista Enterprise AB



## Test of flow in ESB - Continued



- Pros:
  - Clear Interface
  - Don't have to deal with variations at node level
  - Black box testing means test cases often can be reused
  - Tests can easily be based on existing unit test frameworks
- Cons:
  - Custom plumbing code often necessary
  - Can be too coarse grained
  - Slow runtime
    - Deployment is necessary
    - Asynchronous aspects has to be handled

Cadec 2007, Test Driven ESB Development, Slide 14  
Copyright 2007, Callista Enterprise AB



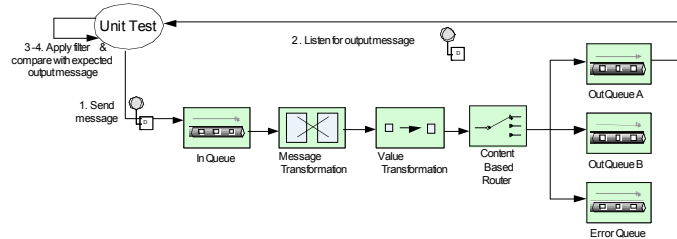
## A Real Life Example – Test of Flow in ESB

- Each test covered one (asynchronous) message flow
- Tests were run in ESB, using WebSphere MQ as communication protocol
- Only a few very basic test cases were used
  - input message -> expected output message
- Test framework based on JUnit was built
- Test data was externalized to an XML format

Cadec 2007, Test Driven ESB Development, Slide 15  
Copyright 2007, Callista Enterprise AB



## A Real Life Example - Continued



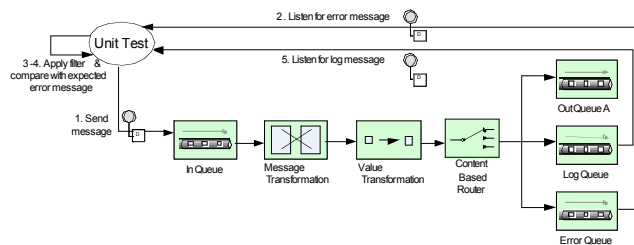
“Happy days”:

1. Send in one message
2. Listen for the outgoing message
3. Apply filter for irrelevant output, e.g. timestamps
4. Compare the outgoing message with an expected outcome

Cadec 2007, Test Driven ESB Development, Slide 16  
Copyright 2007, Callista Enterprise AB



## A Real Life Example - Continued



### Negative tests:

1. Send in one message
2. Check for error message
3. Apply filter for irrelevant output, e.g. timestamps
4. Compare with expected error message
5. Check log queue

Cadec 2007, Test Driven ESB Development, Slide 17  
Copyright 2007, Callista Enterprise AB



## Lessons Learned

- The automated test suite turned out to be a crucial success factor!
  - Made late changes possible
- Fine grained tests is a good complement
  - In our specific environment - quite expensive!
- Testing each flow as one unit made it simple to reuse test cases
  - Building a reusable test framework on top of JUnit
- Externalizing the test data means non programmers can manage the tests.
  - Integration specialists are not always programmers
  - The test framework can be reused for acceptance testing

Cadec 2007, Test Driven ESB Development, Slide 18  
Copyright 2007, Callista Enterprise AB



## Lessons Learned - Continued

- Simple string comparison is not always enough
  - Filter out parts of the response, e.g. timestamps
  - XML infoaset comparison
    - Ignore differences in whitespaces etc
- Address conflicts can cause problems
  - Use specific queues/urls for testing purposes

Cadec 2007, Test Driven ESB Development, Slide 19  
Copyright 2007, Callista Enterprise AB



## Lessons Learned - Choosing a Test Tool

- Specialized tools are available but often expensive.
- Certain ESB vendors offer specialized tools.
  - Often without support for automation.
- Our recommendation: Reuse an existing unit test framework, e.g. JUnit or NUnit.
  - IDE integration
  - Report functionality
- Necessary plumbing code can be written on top of the framework.
  - Often reusable between tests.

Cadec 2007, Test Driven ESB Development, Slide 20  
Copyright 2007, Callista Enterprise AB



## Summary

- Don't postpone testing the ESB flows!
  - TDD can be used in this environment as well.
- Our experience: An automated test suite can be a crucial success factor in a rapid changing world.
- Be pragmatic
  - Find out what test configuration works in your environment.
  - Remember: Rome wasn't built in a day...

Cadec 2007, Test Driven ESB Development, Slide 21  
Copyright 2007, Callista Enterprise AB



## Bottom Line: Start Testing the ESB Today!



Cadec 2007, Test Driven ESB Development, Slide 22  
Copyright 2007, Callista Enterprise AB

