Annual Retrospective *

Felipe M. Besson, Pedro M.B. Leal, Fabio Kon

Department of Computer Science
Institute of Mathematics and Statistics
University of São Paulo (USP)

{besson, pedrombl, fabio.kon}@ime.usp.br

*The research leading to these results has received funding from HP Brasil under the Baile Project and from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement number 257178 (project CHOReOS - Large Scale Choreographies for the Future Internet).

1. Introduction

In 2011, all milestones of the Research Line 2 (Verification & Validation of Choreography) were achieved. To improve our development process as well the Rehearsal features, the USP team in charge of this Research Line have conducted an annual retrospective in December/2011. This retrospective has been conducted following the agile method principles, and its primary goal is to identify the good and bad points belonging to the Rehearsal, focusing on how its features are related to the proposed TDD methodology. Thus, for the next year, the group will focus on keeping the good points identified and correcting the bad ones. The secondary goal consists on obtaining more feedback for writing more scientific papers on 2012.

We analyzed each Rehearsal feature. In the sections bellow we summed up the bad and good points of each feature. The tables presenting the bad points also present the solutions we can take to correct these bad points in the next year.

2. Dynamic creator of web service clients (WSClient)

During the retrospective we identified and discussed the following bad points:

Bad point: The developer must know all tag names for creating an Item object
Possible solution(s): Creating an *assertItem* command to compare the actual and the expected Item structure
Bad point: Verbosity in writing Item objects
Possible solution(s): Creating a new constructor and new methods to facilitate the Item development
Bad point: The developer must know all operation names and types
Possible solution(s): Unfortunately, this is a consequence of providing dynamic behavior. However, during a TDD session, this is not a problem. In this process, the developer defines the operation names and types in the test cases, and then, implements the service contract

The following good points have been highlighted:

- The client dynamic creation improves the test writing speedy since we do not need to create stub objects.
- With this feature, the developer does not need to manipulate XML directly.
- With Item objects we can represent Soap requests and responses. This property facilitates the use of WSClient. Besides, in the test cases, a service response object (Item object) can be used directly when requesting another service. This property can be useful for testing choreography workflows.
- WSClient provides to the developer the use of a test-driven approach in addition to the conventional contract-driven approach.

3. Scalability testing tool

Since this tool is not required under the 2011 work plan, its conception for itself can be considered a positive aspect of our work. Given the importance of applying scalability assessment on web service choreographies, we decide to develop this tool. During the retrospective we identified and discussed the following bad points:

Bad point: The developer must know all business process as well how to change the choreography resources (e.g., number of cloud nodes)
Possible solution(s): When the CHOReOS middleware components is developed, we may integrate this tool with these components. After this integration, tasks, such as the cloud nodes creation and configuration will be automated.
Bad point: The tool presents only the final result and if any error happened, the assessment is interrupted
Possible solution(s): Letting the assessment keeps running even if any error happened. Besides, the tool must provide a debug mode which stores logs from all execution steps

The following good points have been highlighted:

- Results are well presented through charts
- The tool is not restricted to choreographies
- The APIs tool are not verbose
- It is possible to define customized growing patterns

4. Service Mocking (WSMock)

During the retrospective we identified and discussed the following bad points:

Bad point: Each WSMock instance must be deployed in a different port
Possible solution(s): In exploratory study that will be applied next year, we plan to investigate how relevant is this requirement. If it is relevant, the tool must be improved to support this
Bad point: The MockResponse does not provide conditions with intervals
Possible solution(s): In exploratory study that will be applied next year, we plan to investigate how relevant is this requirement. If it is relevant, the tool must be improved to support this
Bad point: The WSMock has not been used in real situations
Possible solution(s): In exploratory study that will be applied next year, the Service Mocking will be used and analyzed in a real development situation

The following good points have been highlighted:

- With this feature we can simulate real dependencies which assists the integration testing
- The API is easy to learn and provide flexibility for manipulating the mock responses and operations

5. Message Interceptor

During the retrospective we identified and discussed the following bad points:

Bad point: The developer must communicate directly to the proxy
Possible solution(s): In exploratory study that will be applied next year, we plan to investigate how relevant is this requirement. If it is relevant, the tool must be improved to support this
Bad point: Each proxy must be deployed in a different port
Possible solution(s): In exploratory study that will be applied next year, we plan to investigate how relevant is this requirement. If it is relevant, the tool must be improved to support this

The following good points have been highlighted:

- The feature provides easy APIs for message interception
- The intercepted messages are stored in-memory, this way, we do not need a database as we nees in the prototype
- All intercepted message are stored as Item instances