Software Architectures Assignment 3: Service Oriented Architectures

Dehouck Samuel, Delhaye Quentin

April 24, 2014

1 Introduction

2 BPEL Processes

2.1 Parallelization

Before any modification of the process, the different actions done in the main sequence were:

- receiveInput: receives input from the requester.
- PrepareResponse: prepares the response variable.
- AssignSearchRequest: assigns the parameters.
- InvokeSearchBooks: invokes the search in the SoftLab library.
- AssignSearchRequest: assigns the parameters.
- InvokeSearchForBooks: invokes the search in the National library.
- AssignResultSoftLib: assigns the result to the response.
- replyOutput: sends the response to the requester.

As we can see, the first two actions need to be done sequentially: we need to receive the input first before preparing the response. The same goes for the last two: we need to prepare the response before sending it. The last two pairs of *Invoke-Assign* are unrelated and their execution can be separated. In order to parallelize these two couples, we need to use flows. A flow is declared using the tag <`bpel:flow`>, in which all the declared sequences (using <`bpel:sequence`>)will be executed at the same time.

A graph representing the execution can be found on figure 2.1

2.2 Data Structure Strategy

In this table, we describe how books are defined in each library.

	National Library	SoftLab Library
Author	String	String
Date	DateTime	Int
ISBN	String	Int
Language	String	Language
Publisher	String	String
Title	String	String

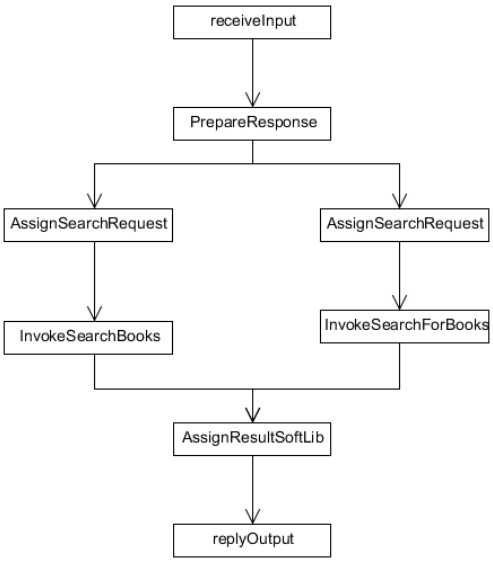

Figure 1: Execution of the BPEL process.

In order to ease the use of the *LibrarySearch* by additional services, we could normalize the data. There are two ways:

- We could use the types defined by the National Library. But in order to convert the data from the SoftLab Library, we would need to change the year in a full date (day, month and year). Some false informations would then be created. The transformation from a *String* to an *Int* would be made without any loss but from a *String* a type *Language*, it would depend on how *Language* is implemented.
- We could use the types defined by the SoftLab Library but we would then loose some informations about the date for instance.

As we can see, both choices are valid and the choice would depend on the case.

3 Integration with Legacy Software

In order to access the remote web services, we added a new package `softarch.portal.db.LibrarySearch` containing two classes: `DatabaseRemote` and `RegularDatabaseRemote`. Those are used in the constructor of `DatabaseFacade` by initializing a regular remote database, no matter what local database configuration has been chosen.

On top of that, the `findRecords()` method had to be modified. It now searches in both local and remote databases, and then concatenates the respective results into one single list of `Book` to return.

The data mapping from `librarysearch.soft.Book`[1] to `softarch.portal.data.Book` is done in `RegularDatabaseRemote.findRecords()` method. The results from the BPEL process are stored in a list of `Book` of the first type, which is then iterated over and from which each field is extracted and stored in a list of `Book` of the second type.

The advantage of doing so is that the translation is kept in one place, but adding a new field in the web service book type, or modifying any field would require a modification in this process as well.

4 Architecture

5 Conclusion

[1] From the client code generated from the WSDL specification.