CHAPTER 1

INTRODUCTION

1.1 Overview

Workflow, also known as business process choreography, is "an automation of a business process, in whole or part, during which documents, information, or tasks are passed from one participant to another for action, according to a set of procedural rules"[1]. Having the benefits of automations, and the growing ability to handle integrations of components on heterogeneous platforms, workflow has come to play an important part of e-business nowadays. More businesses are transforming themselves toward the direction of being a service provider over the network of e-business [2], and the use of workflow technology enable sharing of resources or services in a geologically separated location, with higher efficiency. As a result, integrating two or more business processes within an organization or cross organizationally has become a major application of workflow.

However, there are cases and situations where workflow behaves differently from its norm i.e. failing to be automatically executed, and thus requiring manual interaction. Although these exceptional situations have been categorized into different classes by groups of researchers, most agree on the two main classes: *expected* and *unexpected exception* [3]. Expected exception is a class of exceptions that can be foreseen, hence, by nature, the semantic used to handle this kind of exceptions is part of the semantic of the whole process. Workflow and exceptions are thus, two different aspects of the application semantics: normal behavior and occasional behavior. The definitions of normal behaviors are workflow specific. For example, in some system, any flow that does not require a supervisor's approval is defined as normal. Whereas in some system, any flow that is incapable of meeting a certain deadline, is considered an irregular behavior. Since the norm of each flow differs, unless its characteristics are defined in prior steps, no identification of a deviation could be made. In addition to specifying the irregular behaviors, compensation steps to be followed when such cases occurs also have to be defined. The compensation procedures could vary from the process being rerun, rolled back or compromised, depending on the business logic

of each flow. These compensation procedures are often referred to as *exception handlings*.

Among other workflow specification standards, Business Process Execution Language (BPEL) [4], by Oasis has been the most widely accepted these days. Having the advantage of being tailor made for web services, BPEL is capable of specifying behaviors of business processes both between web services, and as web services – any workflow written with BPEL is itself a web service. BPEL is a composition language designed to facilitate connection and collaborating among objects, business processes, components and web services – the wiring needed for Enterprise Application Integration (EAI).

However, a problem of synchronicity arises in an inter-workflow coordination, due to BPEL's definition of its connection through send-receive activity. A receive activity, as in BPEL specification, blocks and will not complete until a matching message is received by the process instance. This is based on the assumption that *normally* the matching message arrives only *after* the receive activity has been executed. In real practice, that is not always the case, the message could have arrived before the receiving flow is ready, due to various reasons, for example, the receiving flow is busy executing other activity or the message arrives before the receiving flow has been instantiated, or the system crashes. The BPEL specification does not clearly describe how this exceptional case should be handled. In [5], three possible choices of action are proposed:

• Buffer: The message can be stored in a buffer, so that the receive activity can fetch it later.

• Discard: The message can simply be discarded, when there is no receive activity waiting for it.

• Fault: A fault can be thrown since the Web service has received a message for which no process instance is waiting.

This thesis proposes an exception handling framework for the case of message send-receive method, which is the main part of any collaboration. The service will also serve as a match-up tool among different BPEL engines which correlate differently. Moreover, in some cases that there is a high volume of incoming designated to a particular instance of a Flow, this service will be able to handle

simultaneous requests and then forward them, one by one, to the designated flow. Moreover, the service could be used to buffer messages in case there is a commutation problem between some flows or when the enactment engine is not ready.

1.2 Objectives of the Study

To propose a framework in detecting and handling synchronicity exceptions in sending and receiving messages between workflows deployed with BPEL

1.3 Scope of the Study

1. The design will cover services implemented using BPEL version 1.1
2. The design will focus mainly on the service that has already been instantiated
3. Security measures will not be covered in this thesis.
4. The service will correlate mainly using WS-Addressing

1.4 Research Plan

1. Study the background of workflow concepts
2. Study the background of exception handling.
3. Study the Business Process Execution Language standard
4. Design the service interfaces
5. Choose the tools and languages for implementation
6. Implement the service
7. Design sample client services and test against the gateway service
8. Evaluate and review the study
9. Summarize and write the research paper

1.5 Contribution

Obtain a framework in detecting and handling synchronicity exceptions in sending and receiving messages between workflows deployed with BPEL

1.6 Thesis Outline

Chapter 2 introduces basic workflow and web service concepts including BPEL and WSDL including a work in BPEL (version 1.1) specification validation on the problem of asynchronicity based on the definition of the "receive" activity. Chapter 3 proposes the design of a service, named Gateway Service, to handle the problem. And Chapter 4 describes the implementation and introduces an e-auction service as a test case against the developed prototype. And in the last Chapter, a conclusion of the thesis has been made and discussed, also future work and suggestion proposed