CHAPTER 2

THEORETICAL BACKGROUND AND RELATED WORK

2.1 Theoretical Background

2.1.1 Workflow

Workflow is a computerized facilitation or automation of a business process in whole or part. The automation of a workflow is commonly provided thru a Workflow Management System which "completely defines, manages and executes 'workflows' through the execution of software whose order of execution is driven by a computer representation of the workflow logic." The definition of the business logic at build time is termed "process definition" which includes both manual definition and workflow definition. The Workflow Coalition has defined common characteristic of a a Workflow Management System (WFMS) as three functional areas (Figure 2-1) :

- The build time function which includes defining and modeling workflow process and activities.
- The run-time control concerned with managing the workflow process in the operational environment
- The run-time interaction with user and other application

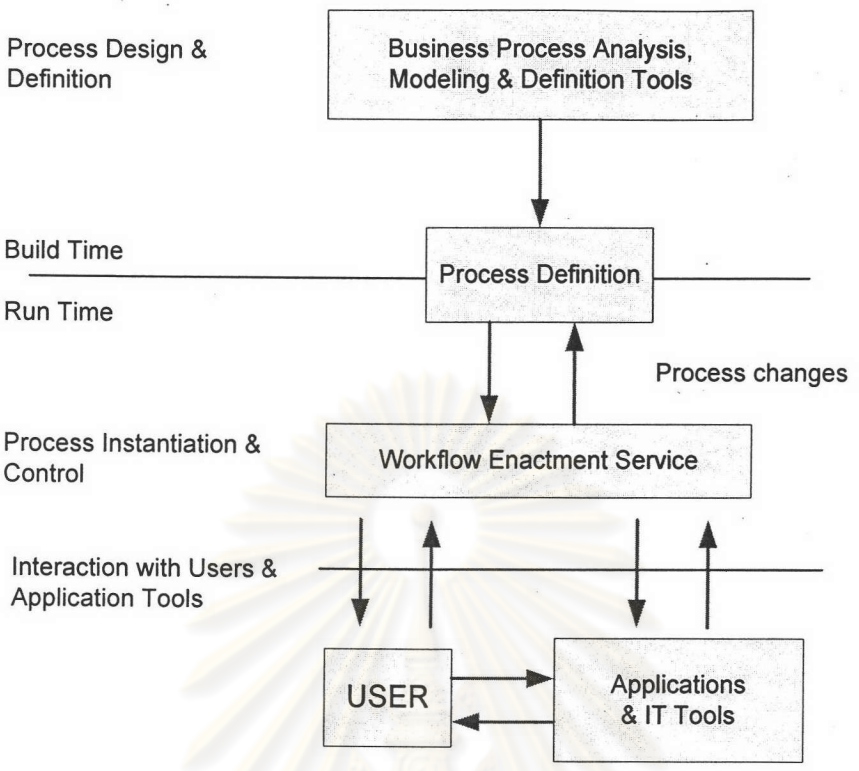

Figure 2-1 Workflow System Characteristic

1. *Build time function* – The process definition is specified with the help of process definition tools, which could be anything ranged from the pencil and paper, a text editor to a sophisticated visualization tool. Most workflow product provides its own process definition tools whose resulting process definition will be interpreted at runtime by workflow engine. In the past few years there has been a number of process definition standards that released and used in products for instance, Business Process Definition Language (BPEL) by Oasis, XML Process Definition Language (XPDL) by Workflow Management Coalition or Business Process Modeling Language (BPML) by Business Process Management Initiative (BPMI)

2. *Run-time Process Control Function* – A workflow Enactment service contains one or more workflow engines providing run-time environment where an instantiation and activation of a process occur. The workflow engine is responsible for interpretation of the process definition and control of process instances such as creation, activation, suspension, termination etc. This also includes the monitoring service of the workflow instances.

3. *Run-time Activity Interaction* – the workflow could interact with user through the workflow client application which is the software that interacts with the user which could be a part of the WFMS itself or could be written by the client. Also it has to interact with other application like process control software or a database. Also a workflow in invoke other applications as its constituent activities. The invoked applications could be local applications or web services. In some environments, in order to invoke these applications, application wrappers might be needed.

2.1.2 Business Process Execution Language (BPEL) [4]

The Business Process Execution Language (BPEL) was design mainly to describe a business process as an interaction among web services. This composition of services to obtain a new service is termed "Web Service Choreography". BPEL was originally called BPEL for Web Service (BPEL4WS) for this obvious reason. BPEL is a successor of two languages which are WSFL from IBM and XLANG from Microsoft and thus allows for the combinations of block structured and graph structured process model, passed on from each of its parent languages.

As it is designed to work mainly with web service, BPEL only support Simple Object Access Protocol (SOAP) and XML (eXtensible Markup Language) messages format. SOAP is a protocol for exchanging information using typed message exchange and remote invocation. BPEL also depends on Web Service Description Language (WSDL) XML schema, XPath and WS-Addressing. Among these standard, WSDL has the most influence on BPEL, as it is the standard describing the service model whose process is describe by BPEL.

2.1.3 Web Service

A web service [6] is a system which creates and requests for any service using network. The communication is based on XML Standard. Message transmission using XML standard makes web service platform and language independent. This provides interoperability between various software applications running on various platforms. Web services easily allow software and services from different companies and locations to be combined easily to provide an integrated service. In the development aspect, web services leverage open standards and protocols. Protocols and data formats are text based where possible, making it easy for developers to see and understand what is going on and it also allows the reuse of services and components within an infrastructure.

2.1.4 Web Services Description Language (WSDL)

WSDL[7] is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. In WSDL, the abstract definition of endpoints and messages is separated from the network deployment and data format binding, and thus allow the reuse of the abstract definition: 'messages' as an abstract definition of data exchanged and 'portType' as an abstract definition of a collection of operations regardless of what message formats or network protocols are used to communicate, such as, SOAP 1.1, HTTP GET/POST, and MIME.

The concrete protocol and data format specifications for a particular port type constitute a reusable binding. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service. Hence, a WSDL document uses the following elements in the definition of network services:

types, which provides data type definitions used to describe the messages exchanged.

message, which represents an abstract definition of the data being transmitted. A message consists of logical parts, each of which is associated with a definition within some type system.

portType, which is a set of abstract operations. Each operation refers to an input message and output messages.

binding, which specifies concrete protocol and data format specifications for the operations and messages defined by a particular portType.

port, which specifies an address for a binding, thus defining a single communication endpoint.

operation which is an abstract description of an action supported by the service.

service, which is used to aggregate a set of related ports.

The grammar is as followed in Figure 2-2:

```
<wsdl:definitions name="nmtoken"? targetNamespace="uri"?>
    <import namespace="uri" location="uri"/>*
    <wsdl:documentation .... /> ?
    <wsdl:types> ?
        <wsdl:documentation .... />?
        <xsd:schema .... />*
        <-- extensibility element --> *
    </wsdl:types>

    <wsdl:message name="nmtoken"> *
        <wsdl:documentation .... />?
        <part name="nmtoken" element="qname"? type="qname"?/> *
    </wsdl:message>

    <wsdl:portType name="nmtoken">*
        <wsdl:documentation .... />?
        <wsdl:operation name="nmtoken">*
           <wsdl:documentation .... /> ?
           <wsdl:input name="nmtoken"? message="qname">?
               <wsdl:documentation .... /> ?
           </wsdl:input>
           <wsdl:output name="nmtoken"? message="qname">?
               <wsdl:documentation .... /> ?
           </wsdl:output>
           <wsdl:fault name="nmtoken" message="qname"> *
               <wsdl:documentation .... /> ?
           </wsdl:fault>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:binding name="nmtoken" type="qname">*
        <wsdl:documentation .... />?
        <-- extensibility element --> *
        <wsdl:operation name="nmtoken">*
           <wsdl:documentation .... /> ?
           <-- extensibility element --> *
           <wsdl:input> ?
               <wsdl:documentation .... /> ?
               <-- extensibility element -->
           </wsdl:input>
           <wsdl:output> ?
               <wsdl:documentation .... /> ?
               <-- extensibility element --> *
           </wsdl:output>
           <wsdl:fault name="nmtoken"> *
               <wsdl:documentation .... /> ?
               <-- extensibility element --> *
           </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="nmtoken"> *
        <wsdl:documentation .... />?
        <wsdl:port name="nmtoken" binding="qname"> *
           <wsdl:documentation .... /> ?
           <-- extensibility element -->
        </wsdl:port>
        <-- extensibility element -->
    </wsdl:service>
    <-- extensibility element --> *

</wsdl:definitions>
```

Figure 2-2 The WSDL Grammar

2.1.5 BPEL Collaboration [4]

One of the important aspects of describing realistic business collaboration is specifying the relationship with a partner process. A partner in this sense refers to both the service customer and service provider having a two-way dependency at a service level. This is especially important as interactions are based on asynchronous messaging rather than remote procedure call. The BPEL uses the concepts of Partner link express partner relationships and add this to WSDL via WSDL's extensibility elements. The descriptions for each element are as followed:

1. Partner Link Types

   A partner link type characterizes the conversational relationship between two services by defining "roles" for each service, for example:

```
<plnk:partnerLinkType name="GatewayService">

  <plnk:role name="GatewayServiceProvider">
    <plnk:portType name="tns:GatewayService"/>
  </plnk:role>

  <plnk:role name="GatewayServiceRequester">
    <plnk:portType name="tns:GatewayServiceCallback"/>
  </plnk:role>

</plnk:partnerLinkType>
```

Figure 2-3 Partner Link Types example

Each role specifies exactly one WSDL portType. Notice that the portType of the role Seller and Buyer are from different name spaces.

2. Partner Link

   The services which a business process interacts are modeled as a partner link in BPEL. Each partner link is characterized by a partnerLinkType, as depicted in Figure 2-4.

```
<partnerLinks>

  <partnerLink name="client"
    partnerLinkType="tns:GA_CreditRatingRequester"
    myRole="GA_CreditRatingRequesterProvider"
    partnerRole="GA_CreditRatingRequesterRequester"/>

  <partnerLink name="GatewayService"
    partnerLinkType="nsxml0:GatewayService"
    partnerRole="GatewayServiceProvider"
    myRole="GatewayServiceRequester"/>

</partnerLinks>
```

Figure 2-4 Partner link example

Another issue in collaboration is how to deliver correct message to a correct instance. BPEL specification proposes the use of correlation sets.

3. Correlation sets

Correlation Sets are the data contained in a message that are meant to be regarded as an identity of conversation. These data often project what the real world business uses as identification of corresponding documents, e.g. a purchase order number or an invoice number. In order for an enactment engine to know which part data in a message is a correlation data, it must be explicitly declared in the flow description. However, there are some BPEL enactment environments that use other mean of specifying and identifying conversation id, such as a WS-Addressing header that are present in messages being sent in SOAP format. An example of such environment is Oracle BPEL Process Manager [3].

2.2 Related Work

2.2.1 Specification and Validation of the Business Process Execution Language for Web Service [5]

This work evaluates the BPEL4WS standards and pointes out the synchronization problem that may arise if the <receive> activity has not been executed before the arrival of the expected message. For example, consider the following code fragment:

```
<sequence>
        <activity1>
        <activity2>
        …
        <receive partnerLink="PL1" portType="PT1" operation="OP1">
</sequence
```

Figure 2-5 BPEL Code Snippet

If the message from partner link "PL1" arrives while the flow is still executing <activity2> the BPEL specification does not clearly describe how this situation should be handled. The author suggests the options of

- Buffering the message
- Discarding the message
- Throwing Fault

However the details in the design in each option have not been proposed. This thesis design and implement a buffer service to store the incoming message until the service is ready to receive its designated message.

2.2.2 Web Service Composition Languages: Old Wine in New Bottles?[8]

This work aims to evaluate and compare workflow management system and web service composition languages using a set of patterns as web service is a paradigm emerging to be used in architecting and implementing business collaboration, both within and cross organizational. When deployed, web services provide by various organizations can be connected to implement business collaboration. A current trend is to express logic of composite web service using business process modeling language tailored for web service. An analysis of some of web service composition languages: BPML, WSCI, and BPEL4WS have been conducted. The analysis is based on a framework composed of a set of patterns.

The study reveals that web service composition languages adopt most of the functionality present in workflow systems. At the same time, it is remarkable that web service composition languages are more expressive than traditional workflow product

and it can be developed to totally support workflow collaboration pattern within and cross organizational.