ON REALIZATION OF ARTIFACT-CENTRIC MODEL FOR
BUSINESS PROCESSES

BY

KAN NGAMAKEUR

Master of Technology (IT)

A thesis submitted to
The Faculty of Information and Communication Technologies
Swinburne University of Technology
For the degree of

MASTER BY RESEARCH

FEBRUARY, 2013
ABSTRACT

In fast-paced business market, business processes which are series of activities performed by various stakeholders are an essential element for an enterprise to ensure growth and most significantly, gain competitive advantages over its rival. To communicate business goals and perform business process reengineering, business process modeling is a significant activity for representing a business process in a comprehensive format. Currently, most existing standards for business process modeling and workflow are based on activity-centric approach. However, this approach cannot provide complete unified information of business processes since data is incorporated at a limited level.

To address this issue, a new paradigm so called artifact-centric approach has been emerging. It provides many benefits to business process modeling areas. The approach focuses on business artifacts as key entities for modeling business processes. One of the challenges is workflow realization of artifact-centric model. Based on our observation, there is an existing approach for realizing artifact-centric process models. This approach is based on the conversion of artifact-centric model into a procedural model. The advantage of this realization approach is ease of implementation. However, the flexibility of an artifact-centric model is reduced since tasks or services are locked up by control flows. Moreover, the conceptual flow diagram may not preserve business logic defined in the ECA rules correctly.

In this research, we aim to investigate approach for realizing the artifact-centric model without transforming the model. There are several challenges identified. To address challenges of this realization approach, we develop the executable model of artifact-centric business process. Then, we present the system so called artifact centric process system. Moreover, we present a technique to use XQuery to monitor and track a running process. A test case is used which shows that our system is able to execute and manage a business
process correctly. Also, running processes can be monitored and tracked using a set of queries based on XQuery.

Furthermore, we also extend the system to support the process execution across organizations. In order to support inter-organizational process, the notion of contract based on artifact-centric approach is introduced. Next, the extended system so called artifact-centric coordination system is developed based on the artifact centric process system. Then, a test case is used to test our system and it turns out that the system is able to execute and manage a business process across organization. Moreover, Monitoring and tracking information can be queried from a log file during the process execution.
THE AUTHOR’S PUBLICATION

Conference paper

ACKNOWLEDGEMENTS

The successful completion of this research has been made possible through the contributions of many people for the past two years. Their contributions come in different forms and are critical to this accomplishment of my study. Thus, I would like to appreciate their help and support, and give my gratitude.

First of all, I would like to give my gratitude to my principal coordinating supervisor, Professor Chengfei Liu who is truly inspirational and diligent. The opportunity to work under his supervision has been a great privilege for his constructive advice and knowledge. During these two years of my study, he has shown me his methodology and techniques to carry out research. Without his guidance, it is impossible to complete my study by myself.

I also would like to give my gratitude to my former associate supervisor, Dr. Jinjun Chen. Although it was a relatively short time under his supervision, he gave me several good advices for my research. I would like to appreciate Professor Jun Han and Dr. Alan Colman for their suggestions in the progress reviews.

Last but not least, I would like to express my thanks to my dear friend, Dr. Sira Yongchareon who has been awarded his PhD title recently. During these two years, he gave me his supports and suggestions for my research topic and the thesis writing as well as his suggestion about my research life. I would like to thank Dr. Malinda Karpuruge for giving his source code and example of his work so I applied some ideas to my work.

Finally, I am grateful to my parents and siblings who is supporting me continuously to my research and enduring the long time separation. Their supports gave me such a great confidence and determination to fulfill the tough work during several years of study in Australia. Then, I would like to thank all my friends who always cheer me up.
DECLARATION

This thesis contains no material which has been accepted for the award to the candidate of any other degree in any University. To the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is given in the text of the thesis.

Kan Ngamakeur

22 Feb 2013
# Table of Contents

**CHAPTER 1:**

**INTRODUCTION** .................................................................................................................. 1

1.1 Introduction to Artifact-centric business process.............................................. 1  
1.2 Motivation ........................................................................................................ 3  
1.3 Research contribution .................................................................................. 7  
1.4 Thesis organization .................................................................................... 9  

**CHAPTER 2:**

**LITERATURE REVIEW** ......................................................................................................... 10

2.1 Activity-centric Business Process Modeling ............................................. 10  
2.2 Artifact-centric Business Process Modeling .......................................... 14  
2.3 Comparison Between Activity-centric and Artifact-centric model........ 22  
2.4 Artifact-centric Business Process Realization ....................................... 24  
2.5 Related system architecture .................................................................... 28  
2.6 Summary ................................................................................................. 31  

**CHAPTER 3:**

**DEFINING AN ARTIFACT-CENTRIC BUSINESS PROCESSES** .................................. 32

3.1 Artifact-centric process model ............................................................... 32  
3.2 Designing an artifact-centric business process....................................... 35  
3.2.1 Overview of design methodology .................................................. 35  
3.2.2 Demonstration of the design methodology .................................... 36  

vi
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.2.1</td>
<td>Business artifact discovery</td>
<td>36</td>
</tr>
<tr>
<td>3.2.2.2</td>
<td>Design of business operation model</td>
<td>38</td>
</tr>
<tr>
<td>3.2.2.3</td>
<td>Workflow realization</td>
<td>41</td>
</tr>
<tr>
<td>3.3</td>
<td>Artifact-centric executable model</td>
<td>43</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Artifact definition</td>
<td>44</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Business rule definition</td>
<td>46</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Service definition</td>
<td>52</td>
</tr>
<tr>
<td>3.4</td>
<td>Summary</td>
<td>53</td>
</tr>
</tbody>
</table>

**CHAPTER 4:**

**EXECUTION AND MONITORING OF ARTIFACT-CENTRIC BUSINESS PROCESSES... 55**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>System requirement analysis</td>
<td>55</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Execution and monitoring problems</td>
<td>55</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Artifact-centric process System requirements</td>
<td>57</td>
</tr>
<tr>
<td>4.2</td>
<td>Design &amp; Implementation of Artifact-centric process system</td>
<td>58</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Overview of the system architecture</td>
<td>58</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Details on the prototype implementation</td>
<td>60</td>
</tr>
<tr>
<td>4.2.2.1</td>
<td>Creation of running instance</td>
<td>60</td>
</tr>
<tr>
<td>4.2.2.2</td>
<td>Rule engine integration</td>
<td>63</td>
</tr>
<tr>
<td>4.2.2.3</td>
<td>Coordination of running instance</td>
<td>65</td>
</tr>
<tr>
<td>4.3</td>
<td>Monitoring &amp; tracking support of Artifact-centric business process</td>
<td>70</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Business process monitoring approach</td>
<td>70</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Monitoring and tracking queries</td>
<td>72</td>
</tr>
</tbody>
</table>
List of Figures

Fig. 1-1 Four “dimensions” in artifact-centric process modeling ...................... 2
Fig. 1-2 Artifacts and their lifecycles in product ordering processes .............. 3
Fig. 1-3 Artifact-centric realization framework ........................................... 7
Fig. 2-1 Meta model of BPEL (from [31]) ...................................................... 12
Fig. 2-2 BPMN Elements (from [30]) ......................................................... 13
Fig. 2-3 Three layers framework for BPM (from [5]) ................................. 15
Fig. 2-4 Operation view vs Role-based view in the view framework (from [9]) 17
Fig. 2-5 Specialization of Artifact-centric process model (from [54]) ............ 18
Fig. 2-6 ALC enrichment with contextual details (from [12]) ...................... 19
Fig. 2-7 The concept of interoperation hub (from [6]) ............................... 21
Fig. 2-8 Location-aware artifact (from [7]) ................................................. 22
Fig. 2-9 Artiflow workflow model (from [11]) ............................................ 25
Fig. 2-10 Realization of Artiflow (from [26]) ............................................. 26
Fig. 2-11 Four layers of the event-driven architecture (from [61]) ............... 30
Fig. 3-1 Example of Business Rules ......................................................... 34
Fig. 3-2 Design methodology (from [5]) ..................................................... 35
Fig. 3-3 Key business artifact from the example business process ............... 36
Fig. 3-4 life cycle of business artifact in the example business process ....... 37
Fig. 3-5 A sample of the conceptual flow diagram ..................................... 42
Fig. 3-6 Data mapping between SOAP Message and Artifacts ................... 52
Fig. 4-1 Architecture of ACP System ....................................................... 58
Fig. 4-2 Class design of running instance of ACP System ......................... 60
Fig. 4-3 Illustration of ACP Factory Operation ......................................... 62
CHAPTER 1:
INTRODUCTION

We organize this chapter as follows. Section 1.1 provides an overview of artifact-centric approach for modeling business processes. Section 1.2 discusses motivation and challenges for realizing artifact-centric business process model. Section 1.3 summarizes our research contributions.

1.1 INTRODUCTION TO ARTIFACT-CENTRIC BUSINESS PROCESS

In fast-paced business market, business processes which are series of activities performed by various stakeholders are an essential element for an enterprise to ensure growth and most significantly, gain competitive advantages over its rival. Business processes usually stem from an enterprise’s business strategies. Since most modern enterprises changed their business strategies from a product-based to a customer-based, it is so significant for enterprises to organize their business operations to deliver value to their customers without involving their customers to cost and risks. Business process modeling is a task for representing a business process in a comprehensive format e.g. graphical notation. Business process model not only can be used to communicate on how business goals are achieved among business stakeholders but also can be used to as a driver for implementing a complete workflow to automate a particular business process. In the traditional business process modeling approach, there are several standards e.g. Business Process Modeling and Notation (BPMN) [29], Business Process Execution Language (BPEL) [27,28], Xml Process Definition Language (XPDL) [49] and Yet Another Workflow Language (YAWL) [50]. However, this approach focuses on defining the sequence of activities but they incorporate data aspect of business processes at limited level. Thus, it is difficult to comprehend the possible effects of the sequence of processing steps on key business entities [5]. Moreover, a lack of holistic view of information and process makes it difficult to communicate business intents in a unified way for each stakeholder [1, 2]. This approach often slows down consolidation of business operations. Control flows being used to tie activities together lead to highly-cohesive and tightly-
coupled process structures. Therefore, process componentization and extension are hard to be achieved in a natural way [51].

In the past few years, an artifact-centric approach to business process modeling has been introduced as a promising paradigm [1]. In essence, the approach has a central focus on defining key business entities, so called “business artifacts”, which are evolved and manipulated within a process. In artifact-centric approach [5], business processes are defined using four components including business artifacts, lifecycles of artifacts, association and services (a.k.a. tasks) as illustrated in Fig. 1-1.

![Fig. 1-1 Four “dimensions” in artifact-centric process modeling](image)

Business artifacts contain all relevant information required to complete business process execution as well as their life cycles which describe possible stages in the evolution of business artifacts and are represented by using the finite state machines. Services are used to make changes to both business information and lifecycles of one or more business artifacts in a manner that is controlled by a set of constraints or associations. These constraints can be expressed by using declarative specifications, e.g., a set of business rules or a set of ECA (Event-Condition-Action) rules. The benefits of this approach have been evidenced in both academic and industrial researches which can be summarized as follows [1, 2, 4, 5, 8, 22, 54, 55, 56, 60].
- It provides higher level of flexibility of workflow enactment and evolution.
- It facilitates the process of business transformation.
- It helps communicating the business intent for consolidating business operations across organizations.

1.2 Motivation

In this section, we will discuss research motivation and challenges. Artifact-centric business process which focuses on key entities so called “business artifacts” has emerged over the past several years. This approach provides many benefits as we described in the previous section; however, there are still many research challenges need to be explored. According to [4], implementation and optimization is another interesting research challenge in this approach since it is less clear how to implement artifact-centric business process in an efficient manner. From an example of business process regarding online ordering process, the process starts when a customer places an order including billing information through a web site. Then the order is sent to a manufacturing factory where the ordered product is assembled, tested and packaged. Finally, the product is shipped to the customer. After we examine this process, several business artifacts are identified. Fig. 1-2 shows data model and lifecycle model of key business artifacts involved with this business process. For each artifact, the data model represents its data attributes, while the lifecycle model represents its state transition of the artifact.

We can see that this process consists of three classes of artifacts: invoice, shipment and order. Apart from the artifacts, we can see other two components in the lifecycle model.
They are essential for constructing a complete business process – these are *services* (a.k.a. tasks) and *business rules*. A service is used to make change on artifacts. An association between services and business artifact(s) is specified by using business rules as to describe on what condition such service is performed on the artifact(s). From the above example, one possible and practical approach for realizing the artifact-centric business processes is by transforming an artifact-centric model to a conceptual flow model, which is a procedural model. The conceptual flow model is, then, mapped into an executable workflow, e.g., BPEL [5]. The advantages of using this approach is not only an ease of implementation as workflow technologies and standards based on the traditional model have been developed, e.g., in [11,12], but also the artifact-centric workflow can be distributed across organizations. In spite of such good points, we argue that this approach has several drawbacks as the transformation, which is unidirectional, poses loss of information. By converting the model, business rules are degraded into control flows; therefore, it is difficult to track and manage the rules based on the converted model. The flexibility of the process is also reduced as business rules are not available to be modified at run-time. Moreover, monitoring and tracking cannot be achieved directly since information of business artifacts and business rules are split and converted into control flows. As a result, this realization approach needs sophisticated technique involving data gathering and processing to support monitoring and tracking of business processes.

Another possible approach is to realize the artifact-centric process model directly without converting or transforming the model. In this approach, business rules are the main mechanism to drive the workflow execution since each step of the workflow execution would focus on identifying services eligible for invocation based on decision of a rule engine. We anticipate several benefits of this approach. One of these benefits is the logical information of artifact-centric process model is preserved since this is a direct interpretive approach. Without model conversion, this approach uses less time and reduces chance of making errors. Flexibility can also be achieved by using business rules which allow changes to be made at design time and runtime in this approach. The easy monitoring and tracking is another advantage since data model of business artifacts is not converted or transformed. Thus, we can perform monitoring and tracking by inspecting business artifact...
directly since attributes and life cycle of each business artifact can reflect progress of a particular process towards business goals.

Despite the advantages of this approach, there are still challenges needed to be further investigated. As opposed to the former approach which is well supported by the current workflow standards and technologies, it is still unclear on realizing the artifact-centric process model directly and efficiently. Artifact-centric process model provides logical specification of how business process is conducted; however, the model cannot be executed directly. The model has to be converted to an executable format or model. Importantly, the executable model has to preserve business logics defined in the conceptual model. This is going to be an issue on how to define an executable model that can capture essential information described in the higher level model.

Another issue is the execution of an artifact-centric executable model. Since all implementation details of a particular business process are defined in the executable model, we need to develop a system that can read and parse the model to put it into an action. There are several points regarding a system design and development to be concerned. First of all, running instances which are representation of the model during run-time not only need to be able to capture all run-time execution information but also have to reflect all aspects of the model, i.e., business artifact model. Thus, we need to design appropriate data models of running instances that provide correct and adequate information to support operation of a system. Next, a system needs to be able to create and coordinate running instances correctly during runtime execution. The mechanism for creating running instances can be as same as or similar to traditional workflow engines existed in the current software market. However, coordination of running instances in a system is different for both artifact-centric approach and traditional approach. In traditional model, coordination of running instances can be achieved by means of control flows which are used to explicitly define a sequence of tasks whereas rules are used to coordinate these instances in the artifact-centric approach. Thus, a rule engine must be implemented into a system. This is going to be another point that needs to be explored in order to use a rule engine as a central controller to facilitate the coordination.

Monitoring and tracking can also be a challenge since a business process defined by artifact-centric approach is implicit. Although we can inspect business artifacts directly to
observe their progress towards business goals; however, these information may not be sufficient to describe overall monitoring and tracking aspects of a business process for some stakeholders. Thus, a mechanism that provides complete and consistent information for monitoring and tracking purpose needs to be explored. This includes a task to collect and correlate the business data and the relationship among them. In artifact-centric approach, business artifacts can facilitate a task of collecting business data; however, they do not provide any information regarding tasks and rules that make changes to them since all components of the process model are separate units. Thus, we need to discover a method to correlate these components during run-time to create relationship between business artifacts, business rules and services. Stakeholders may want to use the execution data to perform business process analysis. Thus, we need to provide an access to the collected data for detailed investigation, cause and effect analysis and monitoring. This is going to be another issue to be addressed.

The nature of business rules in this approach enforces a workflow to become centralized; however, there have been increasing trends for collaborating across organizations to achieve their mutual business goals. This is a challenge if artifact-centric workflow needed to be distributed across organizations. The mechanism to provide consistent workflow execution across organizations needs to be investigated as well. In collaborative business process, we normally have a process model act as a contract between participating organizations, such as BPEL [27] and CDL [35] in the traditional approach whereas there are some preliminary works that try to address this issue [6,7,22] in the artifact-centric approach. However, these works only focus on the conceptual level. Therefore, we aim to develop the executable model to serve as a contract. This is going to be an issue needs to be solved since the model that will be used as a contract must contain valid and adequate information regarding interaction between partner organizations to support operations of an execution engine during run-time.

The execution of artifact-centric business process model in inter-organizational environment is an interesting issue need to be investigated since core components of the artifact model which consist of business artifacts, business rules and services are completely different from the traditional approach. To achieve the collaboration in the artifact-centric business process, an organization needs to share business artifacts with its
partners since business artifacts contain data that may be interesting to its partner. Thus, a mechanism for supporting artifact sharing needs to be explored. Moreover, during a process execution, interactions between participating organizations need to be coordinated. In the traditional workflow, the control flows can be used to address this requirement but this is going to be an issue for artifact-centric approach since business rules are used to coordinate a process execution. Thus, the mechanism to coordinate interactions between participating organizations using pure business rules needs to be investigated.

1.3 RESEARCH CONTRIBUTION

In this thesis, we study artifact-centric approach to modeling business processes and propose our realization framework as shown in Fig. 1-3 to address key issues for realizing artifact-centric business model discussed in Section 1.2. We summarize our contributions of this thesis as follows.

![Fig. 1-3 Artifact-centric realization framework](image)

- **An executable model for support execution of artifact-centric business process**

We propose an executable version of the artifact-centric process model. The executable model is defined using XML language and consists of three definitions which are Artifact definition, business rule definition and service definition. These components not only provide implementation details required by a system to execute a particular business process but also preserve business logics specified in the higher level conceptual model.

- **An execution engine and support for monitoring**

We propose a system so called Artifact-centric process system (a.k.a. ACP system) for realizing artifact-centric business process. Firstly, we propose data structures of running
instances which not only capture business information during the course of process
execution but also help defining relationships among artifacts, business rules, and services.
Next, we propose architecture of the ACP system that consists of six components. They are
Process Deployer, Business Rule Engine, Process Controller, Web service Controller,
Artifact Controller and Front-end UI Interface. These components are used to create and
coordinate running instances during process execution. Moreover, we propose a technique
to use a log file which represents running instances in the ACP system to support
monitoring and tracking. Based on the structure of a log file, we propose a mechanism to
access and query a log file to help answer stakeholders’ questions regarding business
process execution.

− Support for executing artifact-centric business process in the inter-organizational
  environment

We propose an extension of our realization approach to support execution of artifact-centric
business process in the inter-organizational environment. This includes an executable
model that serves as a contract between organizations. The model consists of shared
artifacts, shared rules, and roles. A shared artifact not only stores sharable business
information to facilitate data sharing between partner organizations but also has its life
cycle which allows stakeholders to determine status of a particular business process
towards its collaborative goal. A shard rule is used to coordinate interactions of
participating organizations or roles. We also propose a system to support process execution
in the distributed environment. Based on the proposed inter-organizational process model,
we propose data structures of running instances which are used to define relationships
among shared artifacts, share rules, and roles, and used to support operations of the system.
The system consists of several components which are used to create and manages running
instances of collaborative business processes.
1.4 Thesis Organization

The thesis is organized in six chapters. Chapter 1 provides a broad overview of the thesis including the preliminary knowledge of artifact-centric business process, challenges of our research and our contributions. Chapter 2 reviews related works in both activity-centric and artifact-centric areas. The comparison between activity-centric and artifact-centric is also illustrated. Chapter 3 focuses on the first step of our realization framework. Firstly, we introduce the conceptual level of artifact centric model. Then, we discuss the design methodology of the artifact-centric business process model and the approaches for realizing the model, and introduce the executable model of artifact-centric business processes. Chapter 4 proposes an engine that is used to execute the proposed executable model. First, we define the system requirements. Based on the requirements, the system architecture is designed and implemented. Then, we test our system using a test case. Finally, we discuss a result and evaluate the system. Chapter 5 discusses the extension of the proposed framework to support business process execution in inter-organizational environment. First, we extend our artifact model to support inter-organizational process. Then, we define the requirement and design system architecture for inter-organizational processes. Next, we test the system using a test case. Finally, we discuss a test result and evaluate the system. Chapter 6 summarizes our research and future works.
CHAPTER 2:
LITERATURE REVIEW

This chapter is organized as follows. Section 2.1 provides the basic concept of an activity-centric business process modeling, workflow and the existing standards. Section 2.2 introduces and discusses various works related to artifact-centric business process modeling. Section 2.3 provides a comparison between these two approaches. Section 2.4 discusses existing approaches for realizing artifact-centric business process.

2.1 Activity-centric Business Process Modeling

A business process is a collection of structured activities or tasks that are performed by an organization and aims to deliver values, services or products to customers. Moreover, it is an essential tool for every organization to gain competitive advantages over its rivals. The process of representing business processes using flowcharts or graphical notations is called business process modeling. This process is usually performed by business analysts or domain experts and results in a business process model which is a representation of a business process which consists of sequential activities or tasks that are carried out to accomplish the intended goal of an organization. The purpose of the process model is not only to communicate a particular procedure of a business process and its goals but also it can be analyzed to discover the areas that can be improved to increase efficiency and reduce cost of operations. Moreover, the model can be used to serve as an input to support the execution of a business process in business process management system or workflow management system.

Based on WFMC [17], workflow is the automation of procedures where documents, information or tasks are passed between participants to achieve an overall business goal. Several workflows are usually managed by a workflow management system that automates and manages the sequence of related activities and the invocation of human tasks or IT resources in a particular business process. The workflow system provides basic functionalities to support the workflow management. At design-time, the system provides various tools to help defining and modeling the process definition. The definition usually
consists of a network of activities and their relationships. At run-time, the system not only provides a functionality to manage the workflows and handle various activities that are part of each process, but also manages interaction with human users and IT application tools for processing the various activity steps.

As far as we observed, the current standards and concepts of business process modeling and workflow focus on sequencing various activities to model a business process model, e.g. BPEL, BPMN, XPDL etc. Thus, they are basically based on the activity-centric approach. We review some current business modeling standards to provide basic understanding of the activity-centric business process model.

**Business Process Execution Language (BPEL)**

Business Process Execution Language (BPEL or WS-BPEL) [27, 28] is a modeling and executable language for defining business process as a composition of Web services. WS-BPEL provides a language for the specification of executable and abstract business processes. As a result, it extends the Web service interaction model and enables to support business transaction. Moreover, it defines an interoperable integration model that supports the expansion of automated process integration in both intra-corporate and the business-to-business areas. The core constructs of BPEL are basic and structured activities, variables, partner links, and handlers as illustrated in Fig. 2-1. Basic activities which specify the operations of a BPEL process consist of receive, reply, wait, assign, throw, empty, and invoke. The structured activities include sequence, switch, while, pick and flow. Moreover, the structured activities can be nested to describe control flow and synchronization. Variables are used to store workflow data as well as input and output message that are exchanged in invocations of Web service. Handlers define response to unexpected behavior, e.g. time event, message event, fault, compensation, or termination. Partner links depict message exchange pattern between two parties and are referenced by a basic activities involved with Web service invocation.
In basic activities, the receive construct enables a business process to do a blocking wait for a matching message to arrive. The reply construct is to reply a response message to the one that sends a message to the receive construct. The invoke construct is to allow a business process to invoke operations offered by a partner or a Web service. The assign construct allows updating of variables in a business process. The empty construct allows inserting a “no-op” instruction into a business process. This is useful for synchronization of concurrent activities. In structured activities, the sequence construct is used to express a sequential execution of ordered activities. The switch construct allows a business process to have a set of choices that can be selected and executed. The pick construct allows a business process to be paused and wait for an appropriate message to be received or a time-out alarm goes off. The flow construct is to define a concurrency in a business process. The while construct defines a particular iterative activity. This construct keeps executing an iterative activity until the condition is no longer true.

**Business Process Modeling and Notation (BPMN)**

Business Process Modeling and Notation [29] is a graphical notation for modeling business process. This business modeling standard was developed by Business Process
Management Initiative (BPMI) and is currently maintained by the Object Management Group since these two organization merged in 2005. The current version of BPMN is 2.0. The purpose of BPMN is to create a simple and understandable mechanism for developing business process models, while at the same time being able to handle the complexity inherent to business processes. BPMN can cover many types of modeling and allows the development of end-to-end business processes. Currently, there are three types of sub-models including processes, choreographies and collaborations within the BPMN model. Thus, the viewers are able to differentiate between sections of a BPMN diagram. In the BPMN diagram, there are five basic categories of elements including Flow Object, Data, Connecting Objects, Swimlanes and Artifacts.

![Fig. 2-2 BPMN Elements (from [30])](image)

*Flow objects* are the key graphical elements to specify the behavior of a business process. In this category, there are three elements including *Event, Activities* and *gateways* as shown in Fig. 2-2. *Event* is basically something that occurs during the course of a process. *Activity* is a general term of work that is undertaken by a company. *Gateway* is to govern the divergence and convergence of sequence flows in a process. Gateway contains one type of internal markers that is used to indicate the behavior control. *Data* provides information regarding input and output of activities, a singular object and a collection of objects. Data are depicted with *Data Objects, Data Inputs, Data Outputs* and *Data stores*. *Connecting Objects* are used to link the flow objects to each other or other information.
There are four types of Connecting Objects. *Sequence Flow* is used to show the order of activities involved in a process. *Message Flow* represents flow of message between two participants. *Association* is used to link information and Artifacts. *Data Association* is used to show inputs and outputs. *Swimlane* which consists of *Pool* and *lane* element is used to group the primary modeling elements. *Pool* represents a participant in collaboration. *Lane* is a sub-partition in a process and can be used to organize and categorize Activities. Lastly, *Artifacts* are used to provide additional information about the Process. Currently, there are only *Group* and *Text Annotation*.

### 2.2 Artifact-centric Business Process Modeling

The notion of a business artifact was originated in [1]. An artifact is a concrete, identifiable, self-describing set of business information that can be used by a stakeholder to run a business process. The key properties of business artifacts are that a business artifact has unique identity that cannot be changed and self-describing information that can be represented as nested data structure or name-value pairs. The information of an artifact can be modified or made changes arbitrarily. The artifact life cycle is to capture the end-to-end processing of a specific artifact, from creation to completion. This paper also introduced the concept of how artifacts are processed including function [Fun] and flow [Flow]. The function defines specification regarding how artifacts are processed by tasks. Several Tasks may need to refer or modify more than one artifact. The life cycles of artifacts intersect and result in the interactions between artifacts. Flow describes how artifacts are transported across functional units of the business. Thus, business operational model can be constructed using artifacts, their life cycles and their interaction. The model describes how a business goal can be achieved by means of business artifacts and their life cycles. Moreover, the operational model based on business artifacts provides the benefits that are flexibility of the representation, ability for analyzing changes, and ability for managing application.

Rong et al [2] stated that traditional process modeling approaches focus on connecting several activities together to model a particular business process. However, these approaches often raise some issues in the process of consolidating processes across an organization since stakeholders use different process modeling languages which result in the problems in communicating business intents as well as problems in agreeing on a
uniform representation of a business process. However, these problems can be solved by using business artifacts since they can capture the contexture of a business and can serve as fundamental elements to develop operational models. They improved the idea of the business operational model by introducing nine operational patterns for constructing the model including Pipeline, Repository, Branch, Convergence, Project, Creation, Synchronization, Rework and Disposal pattern. Moreover, they developed the method for verification the model based on Petri Nets.

Bhattacharya et al [5] proposed that the artifact-centric business process model can be constructed using four core constructs that are business artifact informational models, artifact lifecycles, services, and association. The informational model of a business artifact is to store information needed in business process. The information model must capture business process goals and allow for reasoning how these goals are achieved. The artifact life cycle is to describe possible stages in the evolution of artifacts, from start to final stage. A business artifact is basically made up of these two pieces of information. Service is a mechanism to make changes to a set of artifacts in an artifact-centric business process. Moreover, these changes should reflect a measurable stop of progress towards the business goals and the division of the business process into some collection of services should be able to conform to administrative organization structures, IT infrastructures, etc. Association is a family of constraints to govern services that make changes in a business process. Association can be represented in either a procedural form (e.g. flowchart) or a declarative form (e.g. ECA Rule).

![Three layers framework for BPM](from [5])

---

**Fig. 2-3 Three layers framework for BPM (from [5])**
They also presented a three-layer framework for artifact-centric business process, as illustrated in Fig. 2-3, and the design methodology of artifact-centric process model. The artifact-centric business process model considered as a logical specification (a.k.a. BOM) sits on the top level. This layer describes how business goals can be achieved in a declarative manner. Then, it is converted to a conceptual flow that captures an essence of the top-level model in a procedural manner. This layer allows stakeholders to be able to optimize the conceptual flow and analyze the physical requirements for implementation to support legacy systems and workflow distribution across organization. Finally, it is mapped into an operational workflow for automation.

The artifact concept based on [5] has been extended in [53]. The preliminary concept of business entities (or business artifacts) with Guard-Stage-Milestone Lifecycles (a.k.a. BEL [GSM] or simply GSM) was introduced. A BEL Type comprises both an information model that captures all business-relevant data of entities of that type, and a lifecycle model that describes the possible stages an entity may evolve. The lifecycle model is defined using stages, where each stage consists in one or more milestones, a stage body, and one or more guards. GSM lifecycle is more declarative than the finite state machine variant proposed in [5], and supports hierarchy and parallelism within a single entity instance.

Kamal et al [25] focused on defining a formal modelling framework based on business artifacts. This paper also stated that an artifact-centric model is made up of artifacts, services and business rules. The basic idea of this paper is that services are less frequently changed. However, connecting the services together or adapting the current workflow of the services for new business poses a technical challenge. In this framework, business rules that are defined in declarative languages and are easy to modify are used to link services together. The formal artifact-based model and a declarative semantics based on the business rules which can be created and modified separately from business artifacts were developed. The process model was also investigated regarding three problems of immediate practical concerns. This study was focused on three key issues including the ability to complete and execution, existence of a dead end and redundancy. The result showed that the problems were undecidable in general, but under various restrict, they are decidable but complete in PSPACE, CO-NP, and NP. In some cases, they are decidable in linear time. Since verification of artifact-centric process models is usually performed business analysts
to ensure that the models satisfy certain artifact properties. These properties manifest the requirement to satisfy business goals. A business analyst usually tries to check and reason whether the process model satisfies the property. This process is not only complicated but it is also repetitive and can be automated. However, a formalism to describe artifact-centric model and a specification language to describe the properties the model are required in the verification process. Thus, Gerede and Su [3] aimed to develop a logic language based on the computational tree logic called Artifact Behavior Specification Language (ABSL) to specify artifact behaviors in artifact-centric process models. This paper also presented a technique for verifying the properties specified in ABSL as well as the results of the study.

There were other works that extend the artifact-centric approach. Yongchareon and Liu [9] introduced a process view framework for artifact-centric business processes. Since there are various stakeholder participating in business processes, generating views to participant according to their roles is a critical requirement to foster the effective business modeling and management. In the artifact-centric process model, constructing view is one key issue since the natures of artifact-centric model are so different compared to the conventional process model.

Fig. 2-4 Operation view vs Role-based view in the view framework (from [9])

To address this issue, an artifact-centric view framework was developed as shown in Fig. 2-4. The framework consists of artifact-centric process model, a process view model with the construction approach, and a set of view consistency rules to preserve the structural and
behavioral consistency between process views. As a result, this proposed framework provides different views for each role, and these views are able to satisfy requirement of each role and support different level of authority when accessing artifacts of collaborative business processes. Moreover, Yongchareon and Liu [54] tackled the challenges to reuse artifact-centric process model. Process specialization is an effective reuse method that can be used to customize and extend base process models to specialized models. However, the specialization of artifact centric process model has not been studied. They observed that inheriting interactions among artifacts for specialized processes and ensuring the consistency of the consistency of the processes are challenging.

To address this issue, the framework of process specialization for artifact-centric business processes has been proposed. This framework introduces the method for defining a specialized process model based on an existing process model, and the behavior consistency between the specialized model and its base model. Fig. 2-5 shows examples of specializations of Artifact-centric process model, where two specializations (Online Ordering and Offline Ordering) of the base business process model Ordering are given.

The view framework was extended and incorporated in the paper called An Artifact-centric View-based Approach to Modeling Inter-organizational Business Processes [22]. During the collaboration, it is observed that the changes in attributes of the artifacts in an internal process of an organization can be reflected in the message sent between organizations. Moreover, the global constraints are linked with some states of the artifacts of each individual party. The fundamental concept of this approach focuses on shared artifact and local artifact. A local artifact is used locally in one organization. A shared
artifact serves as a contract between organizations. The shared artifacts and collaboration model construct a public view of the inter-organizational business process. This public view only acts as the agreements of lifecycles which are sufficient for all organizations to customize their own artifacts including responsible parts of shared artifact and all local artifacts that required in the collaboration. This extension of an individual organization is named as private view. To address verification and conformance between public view and private view, this paper also presented the notion of behavior conformance to tackle the compliance, autonomy, and flexibility issues of business collaboration.

Narendra et al. [12] tried to address flexibility and monitoring issues of the service composition using business artifacts, and their lifecycles. Service-oriented architecture is a set of principles and methodologies for designing and developing software. It can be used to support modeling and enacting business processes by means of the composition of loosely-coupled Web services. This architecture helps improving flexibility and monitorability. However, the current practice of service composition is based on the process-centric approach and still does not provide the desired flexibility and monitorability. This paper stated that these issues can be addressed by means of a unified process- and data-centric approach. A business process is modeled from a set of business entities called as artifacts. An artifact is a collection of business-relevant data or a business record that used in the business process. Changes in an artifact are represented by a state transition system called Artifact Life-Cycle (ALC) as shown in Fig. 2-6. Transitions between states in an ALC are the outcome of executing tasks that corresponds to operations of Web service in a business process. Thus, a business process can be modeled as a set of interacting artifact life cycles.

Fig. 2-6 ALC enrichment with contextual details (from [12])
This paper proposed the contextual enrichment of artifacts (A-context and S-context) and Web services (W-context) and discusses how to combine artifacts, contexts and Web services to support coordination and tracking of changes that make on all artifacts in a business process. A prototype was also presented to proof the concept of this paper.

Liu et al [23] proposed an approach to performance monitoring based on Artifact-centric business. In the current competitive market, there are increasing demands for more effective method for real-time performance monitoring. To achieve this goal, the monitoring context is used to aggregate all data that are involved with real-time computation of business metrics. Then, a monitoring model can be derived from the contexts. However, defining and identifying monitoring contexts is a difficult and time consuming task with many challenges because of the fragmented nature of the activity-centric approach. To address the problems, an artifact-centric process modeling approach is used as a basis for defining the monitoring contexts which is the container for performance measures and key performance indicators since the artifact-centric model are able to describe both information and process in a unified framework. The first step is to create monitoring context skeletons from business-artifact definition and from user inputs. This paper presented a top-down model-driven methodology for definition of monitoring models and monitoring contexts that do not expose all the technical details to the designer. Therefore, the designer can focus on defining KPIs and Performance metrics. The second step is to derive the executable monitoring models from the monitoring context skeletons. An algorithm was proposed to automatically transform the monitoring contexts to the executable monitoring models. This approach resulted in increased process visibility and also agile and effective monitoring systems.

The artifact-centric hubs were proposed by Hull et al. [6]. The hubs provide a centralized, computerized access point, where stakeholders can access data of the common interest and check the current status of an aggregate process.
The interoperation hubs, shown in Fig. 2-7, are considerably different from conventional orchestrators because business artifacts, unlike BPEL, provide a holistic view of process and data. The purpose of the hub is to facilitate communication and business-level synchronization between stakeholders. The formal definitions of Artifact-centric interoperation hub are presented including artifact data schema, artifact life cycle schema, artifact schemas and hub schemas. The framework also incorporates access control mechanisms to cope security issues. These mechanisms consist of views of the artifact schemas and snapshots which restrict what participants from a given organization type can see, windows which restrict a set of artifact instances that a given participant can see and access right called CRUDAE that restricts the operations of a participant can perform based on the current state of an artifact such as read, update etc.

Instead of the centralized hubs, Lohmann and Wolf [7] proposed the use of artifacts in a choreography setting as shown in Fig. 2-8. In particular, business artifacts were enhanced with the explicit information of agents and locations.
This information can be modeled manually or derived systematically from high level description. It is suggested in this paper that the two-dimensional categorization of artifacts can be possibly used as a basis for defining the high level description. The first dimension focuses on the possible changes of ownership and remote visibility of the artifact. There are three types of artifacts which are mobile, persistent, and transient artifacts in this dimension. The second dimension tackles with remote accessibility to action. There are three types including none, synchronous, and asynchronous. After extending artifacts with the agent and location information, the enhanced-version artifact is so called Location-aware artifact. By defining location-aware artifacts and goal states for the artifacts, an interaction model that acts as a contract between the agents can be generated automatically and ensure that specified global goal states on the involved artifacts are reached.

2.3 Comparison Between Activity-Centric and Artifact-Centric Model

In this section, we discuss differences between Activity-centric and Artifact-centric business process model. In the activity-centric approach, the core components of the process model comprise activity and control flows. Activity is a work that forms one logical step within a process. An activity may be performed by human or automated
system. Control flows are used to connect activities together to form logical steps of a business process. The basic types of control flows include Sequential routing, Parallel routing, And-Split, And-Join, Or-Split, and Or-Join. A sequential routing is a part of business process where various activities are performed in sequence under a single thread of execution, whereas a parallel routing focuses on executing two or more activity in parallel. And-Split and And-join are used to indicate a point where a single thread of execution need to be split to execute two or more activities concurrently and a point where multiple threads of execution are converged to a single thread of execution. Or-split and Or-join are used to provide two or more alternatives for a single thread of execution to make a decision upon which branch to take.

In the artifact-centric model, a business process model is developed using business entities so called business artifacts. A business artifact has an identity. It consists of a set of attributes and its lifecycle. Each attribute of an artifact contains the data needed for the workflow execution. These attributes can be created, updated, or deleted by the service in the workflow. The attributes can be just a simple scalars or nested data structures. The life cycle of a business artifact describes various stages in the evolution of the artifact, from start to the final stage. The life cycle can be depicted using a state finite machine. Service is used to make changes to both data and life cycle of one or more artifacts. Business rules is used to govern invocation of services that induce a state transition in involved business artifacts.

The differences between artifact-centric and activity-centric models stem from key constructs of the model. First of all, the activity-centric approach focuses on modeling a business process based on activities, whereas the other approach focuses on business artifacts. The activity-based process model tends to be easier to be developed since we easily identify key activities involved in a business process. However, the data are incorporated at a limited level as input and output of activities. But, business artifacts are able to capture both information aspect and process aspect of a particular business process in a unified way. Thus, artifact-centric approach can provide the better understanding of the overall possible effects of processing steps on key business entities. Secondly, the process model based on activity-centric approach is more explicit than artifact-centric process model since business logics are defined explicitly using control flows, whereas artifact-
centric approach uses a set of business rules which is implicit to define business logics. Next, the life cycle of an artifact is much different than linked activity in the activity-based model. However, many people misunderstand this concept. In the activity-centric model, various activities are connected to form a business process, whereas the artifact life cycle only provides information regarding the possible stages that a particular artifact can evolve during the workflow enactment. We can represent the life cycle of the artifact using a finite state machine. Moreover, the life cycle of one artifact may interact with the life cycles of other artifacts in the business. Thus, this means that the process model requires several business artifacts and their life cycles to form a complete process. Lastly, term of service in the artifact-centric approach is similar to an activity in the activity-centric approach. However, activities are locked up by control flows, whereas services are loosely coupled by business rules. Thus, artifact-centric process model tends to be more flexible than the activity centric model.

2.4 Artifact-centric Business Process Realization

As it was indicated in [4] by Hull, the implementation of artifact-centric business process is considered as one area of research challenges needed. An artifact process model can be converted to a conceptual flow. Then, it is mapped into an executable workflow. This approach was adopted in [11]. G. Liu et al proposed a new artifact workflow model called Artifact Conceptual Flow (ArtiFlow) as illustrated in Fig. 2-9.
The proposed model consists of four types of basic construct including artifacts, services, repositories and events. An artifact stores the business information needed for completing the workflow execution. The life cycles of artifacts are depicted as a graph where nodes are either services or repositories and edges describe how the artifacts move between services and repositories. A repository stores one or more types of artifacts. A service performs on one or more artifacts. Operations of a service include creating artifacts, reading data in the artifacts, modifying artifacts, etc. An event represents a change which can be generated from either
an external or an internal source in the workflow execution. Then, this work described that translation from ArtiFlow to BPEL is feasible. The ArtiFlow model was further adopted in [26]. Since the area of workflow and business process management is taking up willingly to change from the conventional control flow-centric to artifact workflow design and specification. This work introduced a system that stored ArtiFlow specification as XML documents, and mapped the specification to executable workflow communicating existing Web services and human operators through the user interface.

Fig. 2-10 Realization of Artiflow (from [26])

Fig. 2-10 shows realization methods of Artiflow. Firstly, ArtiFlow specifications are defined using ArtiFlow Definition Language (ADL). Then, the definition language is translated into XML document for the purpose of realization and management. To realize the ArtiFlow model, there are two steps for mapping ArtiFlow models to executable models. The first step, ArtiFlow definitions in ADL are translated into XML documents. The next step, the XML documents that represent ADL are mapped to executable workflows. The latter step consists of two phases. The first phase is to produce artifact-aware workflows that are the skeleton for task assignment, data management, workflow execution and management. Then, the detailed implementations are tackled in the second phase such as mapping of the artifacts and other data in the abstract workflow to messages and data flow in executable systems. Finally, a prototype so called ArtiMT was designed and developed.
To support a dynamic runtime modification, W. Xu et al. [24] developed a hybrid model, which gains advantages of both declarative and procedural model, so called DEZ-Flow. Firstly, the model of EZ-Flow, previously called ArtiFlow, was introduced. The notion of EZ-Flow consists of artifacts and classes, tasks and workflow schemas. Each workflow schema has a primary artifact class called a core artifact. A core artifact stores both data and the enactment it is involved in. Auxiliary artifacts may be needed if a workflow requests for other source of information. Artifacts are manipulated by tasks. The execution of a task is completed through a sequence of action. The process of a task execution is represented a model that is similar to Petri nets. The model has three types of tokens which include event, enactment, and data. An event token contains information that used by a task as input or correlation. An enactment token contains the core artifact. A data token contains all necessary artifacts needed in the task. The task execution is triggered by an event. During the task execution, each token is created and transited through predefined places. Upon the completion of the task execution, an event is generated. A repository is used to store artifact during the workflow execution. The engine for executing EZ-Flow was also developed. Then, a technique for dynamic modification of execution for EZ-Flow was presented. Four types of execution modification constructs were introduced. The skip construct is to omit the execution of a task. The replace construct is to replace a current task with another task. The add construct is to execute an additional task immediately before executing another task. The retract constructs is to roll back the execution to the point of a previous completed task. Finally, an execution engine to support a runtime modification was developed as well as a simple rule-based language to define modifications based on four modification constructs. Cohn and Hull [8] illustrated that IBM has used BELA tool to map an artifact-centric process model into a workflow that runs on IBM’s WebSphere Process Server. In this research, we use a different approach compared to Artiflow.

Our realization approach is to generate the executable model from the logical specification of an artifact-centric model based on [9,10] without any transformation of the model. We also develop our prototype to execute our proposed executable model where the system uses business rules to control each state of process execution. In Siena [15], users can model business artifacts and process as an XML documents in order to create a
composite web application. Then, the application is deployed and executed on an execution engine. However, there is no use of business rules. Moreover, processes are still executed in a procedural manner. In [52] and [57], AXML Artifact model is built on AXML document, which is an XML document with embedded function/Web service calls. AXML artifact is an artifact tree with nodes and is identified by an artifact identifier. Its nodes include subartifact nodes, element nodes, content nodes, and function call nodes. The function calls are special nodes. When these function calls get activated, the corresponding Web services are called using XML data as parameters. The result of Web service invocations which is an XML data are integrated as siblings to the XML node that corresponds to the function call. Based on The AXML Artifact model, AXART system has been developed and is a peer to peer system. The main goal of the system is to manage the updates of artifacts. The system checks properties of stages of the artifacts and the higher-level artifacts to ensure that they can pass to a next stage. The properties of stages are maintained in the tree-patterns. Once the properties of stages are satisfied, the update is realized by calling a function of the AXML system that stores artifacts. In this work, function/Web service calls are carried within artifact as well as logics of these functions. In contrast, artifacts, rules, services are separate components in our model. Thus, the AXML Artifact model is suitable for execution in distributed environment, but Web services and logics embedded in artifacts can be difficult to manage. Moreover, Monitoring for AXML Artifact model may need some mechanisms to provide complete information to all peers.

2.5 Related System Architecture

In this section, we provide the fundamental concept of service-oriented architecture and event-driven architecture that are adopted in our research since the artifact-centric model is composed of artifacts, services and business rules. However, the term of service used in this research closely relate to the concept of service in SOA. Moreover, we can make use of SOA standards and technologies to facilitate our implementation, such as Simple Object Protocol (SOAP), Web Service Description Language (WSDL). We use ECA rule as our business rule. An ECA rule is triggered by an event happened in the artifact-centric business process. The event is validated against a condition. If a condition is satisfied, an action is undertaken. Thus, the artifact-centric business process usually involves with the
event generating and event processing to some extent. Based on this observation, we can see that event-driven architecture might be able to help supporting our implementation.

**Service-oriented architecture**

Service-oriented architecture [39,62] is a set of principles and methodologies for designing and developing distributed systems which are composed of interoperable service. The main purposes of SOA are to help managing the growth of large-scale enterprise system, facilitating internet-scale provisioning and use of service, and reducing cost in organizations. The architecture has a set of distinct characteristics. It is significant for developers to comprehend the concept of SOA. The first characteristic is *discoverable and dynamic bound* which indicates that interfaces of each service are registered and published. Thus, the service consumer can ask the registry for a service that meets its requirement. Secondly, each service must be *self-contained and modular*. A service must provide a set of interfaces. These interfaces should relate to each other in the context of a module so that a service can be easily combined with other services to create an application. Next, each service must have *interoperability*, meaning that they can be invoked by any potential service consumers. Moreover, *loose coupling* is also an important characteristic in SOA since each service must not be tightly coupled to other services and clients so changes can be made without affecting operations of an application. This can be achieved by using standard, dependency-reducing, decoupled message-based method. Furthermore, each service must be *location transparency* so that a location of a service is unknown to service consumers unless they discover it in the registry. Last but not least, *composability* means that services can be combined to create applications or systems since services are coarse-grained reusable components which exhibit their functionality through a well-specified interface. Finally, *self-healing* indicates that distributed systems or applications can recover from errors by themselves without human supports. Some common standard of service-oriented architecture are W3C’s Web Service Description language (WSDL) and ebXML’s Collaboration Protocol Profile.
Event-driven architecture

Event-driven architecture (EDA) [61] is a software architecture that focuses on the production, detection, consumption of, and reaction to events. In essence of this architecture, event is an important thing that occurs inside or outside our business. An event may indicate a problem, an opportunity, a threshold, or a deviation. Each event occurrence is composed of an event header and event body. The event header contains information for describing the event occurrence, e.g. event ID, event type, event name, timestamp etc. The event body explains in detail what happened so that any interest party can make use of the information to take some actions. The actions may result in the invocation of a service, the initiating of a business process.

![Simple Event Processing Flow](image)

**Fig. 2-11 Four layers of the event-driven architecture (from [61])**

Once events are occurred, they need to be processed. In event-driven architecture, each event must go through the event flow which consists of four logical layers as illustrated in Fig. 2-11. The first layer is *event generators*. In this layer, events are generated from a source such as an application, business process, service etc. Then, the events are sent between event generator, event processing and downstream subscribers through *event channel*. Events are processed in the *event processing* layer. In this layer, the events are
evaluated against event processing rules and actions are initiated. There are three common styles of event processing. Simple event processing that is generally used to drive real time flow of work. Stream event processing is used to drive the real-time flow of information in and around the enterprise. Complex event processing is usually used to detect and respond to business anomalies, threats, and opportunities. In downstream event-driven activity layer, the invocation of downstream activities may be issued by the event processing engine or requested by subscribers. Most implementations of the event-driven architecture are based on the four layers design discussed above.

2.6 SUMMARY

This chapter introduces and reviews related works from the past to the present in details. From these papers, it is obvious to see that business processes are important to organizations since enterprises can ensure growth and gain competitive advantages over their rivals. Thus, a business process modeling is so crucial for various enterprises to represent business processes using a flowchart or graphical notation and to discover areas that can be improved to increase efficiency and reduce cost of operations. Workflow is the automation of procedures of business processes to achieve overall business goals and managed by a workflow management system that automates and manages the sequence of related activities and the invocation of human tasks or IT resources in a particular business process. The current standards of business process modeling and workflow focuses on sequencing various activities to form a logical steps in a business process. However, in activity-centric approach, data is incorporated at a limited level. It is difficult to understand the overall aspects of a business process. To address this issue, artifact-centric business process modeling was proposed. In this approach, the process model consists of business artifacts and their life cycles, business rules, and services.

To realize an artifact-centric process model, there are two methods. The first method is to convert the model which defined in a declarative manner to a procedural model. Then, the converted model is mapped to an executable model. Currently, there are some existing works, such as ArtiFlow (changed to EZ-Flow later), which realized the artifact model using this approach. The second approach realizes the artifact model without converting or changing its natures. This is the approach we are going to discuss in this thesis.
CHAPTER 3:
DEFINING AN ARTIFACT-CENTRIC BUSINESS PROCESSES

This chapter gives the fundamental details of a conceptual and an executable model of artifact-centric business process using in our framework. Section 3.1 introduces artifact-centric business process model which contains logical specifications of a business process. Section 3.2 discusses on designing a business process model using artifact-centric executable model. Section 3.3 defines an executable model of the artifact-centric business process.

3.1 ARTIFACT-CENTRIC PROCESS MODEL

In this section, components and syntaxes of an artifact-centric business process model (ACP model) that has been proposed in the previous works [9,10] are introduced. The ACP model consists of sets of artifact classes, services, and business rules.

**Definition 1: (Artifact class).** An artifact, which is a key business entity involved in business processes, contains its relevant attributes and many finite processing states and is abstracted by artifact class. Artifact class $C$ is a tuple $(A, s^{init}, S, S^f)$ where,

- $A = \{a_1, a_2, ..., a_y\}$, and each $a_i \in A$ is a name-value pair attribute
- $S = \{s_1, s_2, ..., s_z\}$ contains the possible states of the instances of class $C$
- $s^{init}$ is the initial state
- $S^f \subseteq S$ is a set of its final states.

**Definition 2: (Artifact schema).** $Z$ is a finite set of artifact classes that are used in a particular process, i.e., $Z = \{C_1, C_2, ..., C_x\}$ where $C_i \in Z (1 \leq i \leq x)$ is an artifact class.
**Definition 3:** (Service). A task that is used to perform read/write operations on some artifact(s), and it is denoted as \( v(C_1, C_2, \ldots, C_y) \) where \( C_1, C_2, \ldots, C_y \) are artifacts that are read/updated by service \( v \).

**Definition 4:** (Business rule). It is used to associate service(s) with artifact(s). It is defined in a Condition-Action style to describe on what pre-condition a particular service is executed, and on what post-condition after performing such service must satisfy. A business rule, denoted as \( r \), is a tuple \( (\lambda, \beta, v) \) where,

- \( \lambda \) and \( \beta \) are a pre-condition and post-condition, respectively;
- \( v \) is a service that performs read/update operations on the attributes and the processing states of some artifacts in schema \( Z \).

We restrict both pre- and post-conditions to be expressed by a conjunctive normal form. This form can contain two types of proposition over schema \( Z \): (1) state proposition (by instate predicate) and (2) attribute proposition (by defined predicate and scalar comparison operators). We write \( \text{defined}(C, a) \) if attribute \( a \in C.A \) of artifact of class \( C \) has a value; and \( \text{instate}(C, s) \) if state \( s \in C.S \) of artifact of class \( C \) is active. Initially, \( \text{instate}(C, s^{\text{init}}) \) implies \( \forall x \in C.A, \neg \text{defined}(C, x) \). A complete set of business rules defined for a particular process model specifies the control logic (named ECA flow) of the whole process from the beginning to the termination of the process. Fig. 3-1 shows an example subset of business rules that are used in our product ordering process.

<table>
<thead>
<tr>
<th>r1: Customer requests to make an order O</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pre-condition</strong></td>
</tr>
<tr>
<td>( \text{instate}(O, \text{init}) \land \text{defined}(O, \text{OrderID}) \land \text{defined}(O, \text{CustomerName}) \land \text{defined}(O, \text{CustomerAddress}) )</td>
</tr>
<tr>
<td><strong>Service</strong></td>
</tr>
<tr>
<td>( \text{createOrder}(O) )</td>
</tr>
<tr>
<td><strong>Post-condition</strong></td>
</tr>
<tr>
<td>( \text{instate}(O, \text{Add_OrderItem}) \land \text{defined}(O, \text{OrderID}) \land \text{defined}(O, \text{CustomerName}) )</td>
</tr>
<tr>
<td>Rule</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>r2: Create Shipment $S$ for an order $O$</td>
</tr>
</tbody>
</table>
Pre-condition: \[ \text{instate}(O, \text{Add} \_ \text{Order} \_ \text{Item}) \land \text{instate}(S, \text{Init}) \land \text{defined}(O. \text{GrandTotal}) \land O.\text{GrandTotal}>0 \land \text{defined}(S.\text{ShipID}) \land \text{defined}(S.\text{OrderID}) \land \text{defined}(S.\text{ShippingAddress}) \]
Service: \[ \text{createShipping}(S,O) \]
Post-condition: \[ \text{instate}(O, \text{Create} \_ \text{Shipping}) \land \text{instate}(S, \text{waiting} \_ \text{for} \_ \text{Ship} \_ \text{Item}) \land \text{defined}(S.\text{CustomerName}) \land \text{defined}(S.\text{ShippingAddress}) \land \text{defined}(S.\text{ShipID}) \land \text{defined}(S.\text{OrderID}) \]

| r3: Create Invoice $I$ for an order $O$ | 
Pre-condition: \[ \text{instate}(I, \text{Init}) \land \text{instate}(O, \text{Creating} \_ \text{Shipping}) \land \text{defined}(I.\text{InvoiceID}) \land \text{defined}(I.\text{OrderID}) \land \text{defined}(I.\text{BillingAddress}) \land \text{defined}(I.\text{InvoiceDate}) \land \text{defined}(I.\text{Total}) \land I.\text{Total}=O.\text{GrandTotal} \]
Service: \[ \text{createInvoice}(I,O) \]
Post-condition: \[ \text{instate}(V, \text{Unpaid}) \land \text{instate}(O, \text{Billed}) \]

**Fig. 3-1 Example of Business Rules**

**Definition 5: (Artifact-Centric Process Model).** Let $\Pi$ denote for an ACP model, and it is a tuple $(Z, V, R)$ where,

- $Z = \{C_1, C_2, ..., C_x\}$ is an artifact schema,
- $V = \{v_1, v_2, ..., v_x\}$ is a set of tasks,
- $R = \{r_1, r_2, ..., r_x\}$ is a set of business rules.
3.2 **Designing an Artifact-Centric Business Process**

We introduced the artifact-centric process model as the business operational model in the previous section. Now, our discussion is turned to the design methodology regarding how to develop the operational model. This section provides the fundamental knowledge of the design methodology as well as the illustration of the design methodology.

### 3.2.1 Overview of design methodology

In this section, we describe the design methodology of artifact-centric business process modeling proposed in [5]. The design methodology consists of four steps: (1) business artifact discovery, (2) business operation modeling, (3) conceptual workflow design, and (4) workflow realization as shown in Fig. 3-2.

---

**Data-Centric Design Methodology**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1    | **Business Artifacts Discovery**  
  (a) Identify critical artifacts for the business process  
  (b) Discover key stages of artifacts’ life cycles from the scenario-based requirements |
| 2    | **Design of Business Operations Model (BOM)**  
  (a) Logical design of artifact schemas  
  (b) Specify services for artifacts needed for moving artifacts through the life-cycles  
  (c) Develop ECA rules that enable artifacts progress in their life cycles |
| 3    | **Design of Conceptual Flow Diagram** |
| 4    | **Workflow Realization** |

---

**Fig. 3-2 Design methodology (from [5])**

The purpose of step 1 is to (a) identify key business artifacts and (b) discover stages in their life cycles. Identifying business artifacts is not an easy task since it requires a deep understanding of an entire business process, especially how data being manipulated through the process and what data are critical to business process. This understanding can be achieved by using top-down analysis and by examining business scenarios. Possible stages of business artifacts can be discovered from the scenarios and combined to create artifact life cycle which can be represented using the finite state machines. In step 2, business artifact discovered in step 1 provides the basic foundation for constructing the business
operation model (a.k.a. BOM) which is a detailed logical specification of business process execution. In particular, step 2 (a) is to design artifact schemas. In step 2 (b), business activities which operate on the artifacts are examined in order to define abstract services for each business activity. Lastly, in step 2(c) the associations between the services and the artifacts are defined.

The last two steps focus on developing operational workflow system. Firstly, the conceptual flow diagram, which is specified in a procedural manner, is developed within step 3. The flow diagram describes how to coordinate artifacts and services to meet the business operation requirement specified in the BOM. Moreover, it can be used to determine the physical requirement for implementation. In step 4, individual components and the workflow are realized in the operational workflow system where services communicate with each other and manipulate business artifacts.

3.2.2 Demonstration of the design methodology

This section illustrates the key element of the artifact-centric design methodology using an example of online ordering process that introduced in the first chapter. The context of the business process is around a company that provides a portal for customers to browse products catalogue and order products through its web store. Then, the ordered products are produced and packaged in a manufacturing plant. The company also facilitates its customer by providing a delivery service to deliver purchased products to its customer. From this example, we discuss steps of the design methodology introduced in Section 3.2.1.

3.2.2.1 Business artifact discovery

<table>
<thead>
<tr>
<th>Order</th>
<th>Invoice</th>
<th>Shipment</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderID</td>
<td>InvoiceID</td>
<td>ShipmentID</td>
</tr>
<tr>
<td>CustomerName</td>
<td>OrderID</td>
<td>OrderID</td>
</tr>
<tr>
<td>OrderItem[]</td>
<td>InvoideDate</td>
<td>CustomerName</td>
</tr>
<tr>
<td>GrandTotal</td>
<td>Total</td>
<td>ShippingAddress</td>
</tr>
<tr>
<td>Customer</td>
<td>AmountPaid</td>
<td>ShipStartDate</td>
</tr>
<tr>
<td>Address</td>
<td></td>
<td>ShipEndDate</td>
</tr>
</tbody>
</table>

Fig. 3-3 Key business artifact from the example business process
In the artifact centric approach, Business artifacts are very critical components to manage the business operations since they can be used to keep track progress of the process towards its operational goal. Thus, the first step of the design methodology is to identify business artifact together with their life cycles. Note that business artifact discovery requires an intensive analysis of a whole business process as suggested in Section 3.2.1. The purpose of this sub section is to provide a preliminary idea on how to identify the artifacts. Fig. 3-3 shows key business artifacts identified from the example. Since the main goal of this process is to complete an order for a particular customer, we can identify a key business artifact called an order artifact which can be used to measure progress of an order towards the operational goal. It contains details of ordered items, pricing, etc. One important class of artifact could be invoice artifact which is used to record details regarding a payment of a particular order, e.g. bank account, credit card details. This artifact usually refers to a single order artifact. Since a company needs to delivery an order to a customer, shipment artifact is also needed. This artifact holds information regarding schedule time for delivery, delivery orders, etc.

![Fig. 3-4 life cycle of business artifact in the example business process](image)

After identifying business artifacts, there is another task to be done. A Life cycle of business artifact must be discovered. Business-relevant phases are represented as stages in the life cycle of a business artifact. These stages are formed to describe life cycle of the artifact. Then, the life cycle is visualized by using finite-state machine as shown in Fig. 3-4. Let’s briefly describe life cycle of an order artifact which has 10 stages in its life cycle. Once an order artifact is created, its state is moved from Init to Add_Order_item. At this state, a customer may add one or more products to his/her order. When a customer finishes adding products, the life cycle is transited from Add_order_Item to Creating_shipping. At the same time, shipment artifact is created. Then, a customer makes a payment and the state
of an order artifact is changed to billed. After that, an order is sent to a factory to be processed. This phase of the business process is depicted by Processing_order_item and Processing_complete state. The next three states which are Ready_for_shipping, In_shipping and Shipped represent delivery phase of this order process. Closed state is used to indicate the completion of the business process. Once we have these business artifacts and their cycles identified, we are ready to move to another step of this design methodology.

3.2.2.2 Design of business operation model

After we have key business artifacts defined in the previous step, we can use these business entities to develop a complete business operational model (BOM). In order to specify a business operational model, there are three steps including (1) design of artifact information model, (2) specifying services that move artifacts through life cycles, and (3) developing the ECA rules that govern services operate on artifacts. In the following, we illustrate how to develop each component that constitutes the operational model.

− Specification of Artifact information models and life cycles

In this step, we have to define information model and lifecycle for each key artifact discovered in the previous step. There are two primary strategies need to take into considerations during designing the artifact schemas. The first is business-relevant information stored in an artifact must be used to capture business process goals. The second one is artifact data must allow for monitoring and tracking progress towards business goals. The first task is to define the information models of the artifacts. The purpose of the information model of a business artifact is to hold all business-relevant information needed in business process execution. Based on the sample business process, a schema of an order artifact can include several scalar attributes. The ID is a unique identifier which can help tracking a particular order artifact during a process execution. The OrderItem provides details of chosen products for a particular order. This attribute is used to calculate a grand total price for an Invoice artifact. The Customer address holds information regarding location of a customer accommodation. The artifact can also contain ShipId and InvoiceId which are used to refer to given shipping and invoice involved in the same process. Thus,
this allows us to refer to specific order-relevant data such as the start date of shipping. An order artifact may hold start date and end date attributes. This allows us to calculate duration of a particular order which can be used in performance monitoring and business process analysis. Moreover, order artifact may include attributes that store information about the provenance of order artifact to help answer questions regarding a history of its evolution. As you can see that order artifact has no limit on attributes it can contain; however, these attributes must provide information that is relevant to context of order.

Regarding life cycles of artifacts, there is no need to discuss about them in this section since the life cycles can be depicted by using the finite-state machines as described in the previous section.

- Specification of Service

Now we turn our discussion to the task of identify services and their specifications in the business operational model. The purpose of service is to move artifacts through their life cycles. Based on the artifact schemas, we are able to identify services that correspond to both natural business activities and update coherent group of artifacts and attributes. In artifact-centric approach, the set of services for an application domain is normally defined as a library of services rather than specifying the sequencing of services using control flows. There could be a lot of services that can be identified from the sample business process, but let’s briefly describe some identified services, createOrder(O) service is used to create order artifact for O. AddItem(O,P) service is for adding product P to order O. CreateShipping(S,O) service has the effect of creating shipment artifact for S and O. CompleteItem(O) service is used to submit O to a manufacturing plant.

In this design methodology, a logical specification of the service is defined in Input-Output-Pre condition-Effect (IOPE) format. In an IOPE specification of a service, the input and output identify artifacts and their attributes that will be read and that may be updated by the service. The pre-conditions must be satisfied before the service can be invoked. Finally, the conditional effects provide information about the possible effects that applying the service will have.

Let’s describe the specification of createOrder(O) service using IOPE concept. The input of the service is details of an order O including customer name, customer address, payment details. An output of this service is that a new Order artifact will be created and its
attributes will be populated by details of an order $O$. The *Precondition* is that Details of an order $O$ must be completed. There must be no null value. Lastly, an *effect* of the service is that order artifact is in stage *Add_order_item*. As we can see that the IOPE concept can help describe behavior of a service. However, in our framework, precondition and effect of a service specification are defined into ECA rules which will be described in the next section since this allows service to be reuse across the application domains.

### Specification of ECA Rules

The last step in development of the operational model is to specify ECA rules that are used to associate artifacts with services. For several years, ECA rule has been popular in workflow and other areas. It provides a flexible mechanism for specifying behaviours of systems. The basic concept of ECA Rule is that if some event occurs, and if a condition is satisfied, then do a particular action. The basic build blocks of the ECA rules are *Event, Pre-Condition, and Action*. Event is an occurrence that triggers invocation of a rule. In artifact-centric approach, events occur during the business process execution can be (1) an incoming message, (2) an artifact has just changed its state and (3) attribute value of an artifact just assigned. Pre-condition is a condition that must be satisfied before the service can be invoked and usually written in first-order logic. Action defines a task need to be undertaken if a condition is true. Apart from the basic construct of ECA rules, we add Post-condition to describe effect of service invocation that is applied on one or more artifacts. Post-condition is written in the same manner as in the pre-condition. The specification of the business rule can be defined as follows:

<table>
<thead>
<tr>
<th>r1: Customer requests to make an order $O$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Event</strong></td>
</tr>
<tr>
<td><strong>Pre-condition</strong></td>
</tr>
<tr>
<td><strong>Service</strong></td>
</tr>
<tr>
<td><strong>Post-condition</strong></td>
</tr>
</tbody>
</table>
The above specification is an example taken from our online ordering process and describes the logical details of how new order artifact can be created. This rule is triggered when a customer submits the request through a website. The request includes order id, customer name, and customer address. When the condition of this rule is satisfied, the `createOrder` service is, then, invoked to create an `Order` artifact. After the artifact creation is completed, `Order` artifact should be in `Add_OrderItem` stage and its `OrderId`, `CustomerName` and `CustomerAddress` attributes must be assigned a value. Once all logical specifications of business operational model are completely designed, we can use this operational model to develop an operational workflow system.

### 3.2.2.3 Workflow realization

Artifact Schema, Services and ECA rules in a BOM provide a logical design of both the business process and data with a clear semantics. The purpose of realization is to develop an operational workflow system that conforms to the BOM specification. According to the realization approach presented in [5], there are 3-phases including (1) conceptual flow diagram, (2) operational optimization and (3) individual component implementation. The goal of the first phase is to develop a choreography that implements the control structure of the ECA rules. Thus, the result of this phase is a conceptual flow diagram. In this phase, the ECA rules are mapped into control flow components. Fig. 3-5 shows a flow diagram that represents the ECA rule for creating order artifact discussed in the previous section. The circle shape represents a single logical container for the class `order`. An event handler is shown as a rectangular shape with a swallowtail. The labelled rounded-corner box represents either service execution or change of stage. The edge linked the top of the box indicates an invocation. Moreover, it can represent flow of information. Edge labels indicate the information type which can be either an artifact or a message. A solid triangle attached to a node indicates the communication action a particular node will take including act of emitting and pulling information.
Once conceptual flow diagram is completely defined, the diagram needs to be verified to ensure that not only it is free of deadlocks and each node is reachable but also its behaviour must conform to the ECA rules of BOM. The second phase is operational optimization which focuses on finding the optimum flow diagram in term of behaviour constraints of the service and the cost metrics. In this phase, the conceptual flow may be refined or modified to help organizing services better and avoid implementation limitations. After the refined flow diagram is completed, the final step is to implement each node in the diagram. This step involves analysing software development techniques and making implementation decision. The decision may start from the higher level which focuses on communication between components. Based on current technology, SOA framework may be used to address communication requirements. Then, the implementation of individual components is carried out. For example, the Order container may be implemented using the relational database approach which involves developing ER diagram and creating tables using database design tool. The advantage of this realization approach is the ease of implementation since the declarative specifications of the business operational model are altered into a procedural model which can be easily mapped and implemented using the current workflow technologies. However, the flexibility of an artifact-centric model is reduced since tasks or services are locked up by control flows. Moreover, the conceptual flow diagram may not preserve business logic defined in the ECA rules correctly.

Thus, we consider realizing the BOM without transforming the model. In this way, we are able to maintain business logic and flexibility of the operational model. Since logic of the operational model can mapped faithfully into an executable version of BOM, we do not
need to perform conformance checking whereas the former approach must perform this activity since business logic can be mapped incorrect during the process of a model conversion. Moreover, monitoring and tracking of business process can be achieved directly since business artifacts are so called self-contained entities which can be used to describe both informational and behavioural aspects of business process. Thus, this is more appropriate approach for realizing an artifact-centric business process model. To support this approach, we develop the executable model of artifact-centric business process and the details of the executable model will be discussed in the next section.

3.3 Artifact-Centric Executable Model

Now, we propose to use an executable version of the ACP model based on the ACP Model definitions described in Section 3.1. Our artifact-centric executable process model is defined by using XML, and it consists of three definitions: artifact definition, business rule definition, and service definition, as shown below. It contains implementation details required by a system to execute a particular business process and they are used for creating running instances.

```xml
<artifacts>
  <artifact name = "Order">
    <attribute>
      <name>"orderID"</name> <structure>"pair"</structure> <type>"String"</type>
    </attribute>
    <state>
      <state name = "init" type = "init"/>
      <state name = "open_for_item"/>
      <state name = "end" type = "end"/>
    </state>
  </artifact>

  <artifact>
    <attribute>
      <name>"itemID"</name> <structure>"pair"</structure> <type>"String"</type>
    </attribute>
    <state>
      <state name = "init" type = "init"/>
      <state name = "open_for_item"/>
      <state name = "end" type = "end"/>
    </state>
  </artifact>

  <artifact>
    <attribute>
      <name>"item_quantity"</name> <structure>"pair"</structure> <type>"integer"</type>
    </attribute>
    <state>
      <state name = "init" type = "init"/>
      <state name = "open_for_item"/>
      <state name = "end" type = "end"/>
    </state>
  </artifact>
</artifacts>

<services>
  <service name = "createOrderService" inputMessage = "createOrderRequest" outputMessage = "createOrderResponse" operation = "createOrder">
    <port = "create_order"/>
    <location = "http://localhost:8080/gui/services/createOrder?wsdl"/>
    <namespace = "www.swin.edu.au"/>
  </services>
</services>

<businessrules>
  <rule name = "create_order">
    <event type = "inputMessage" type = "createOrder">
      <precon>
        <and>
          <atom property = "state" value = "init"/>
          <atom property = "id" value = "orderID"/>
          <atom property = "value" value = "0"/>
        </and>
      </precon>
      <do>
        <invoke type = "Internal" operation = "createOrder" service = "createOrderService"/>
        .......... 
        <mapping>
        .......... 
        <transition>
          transition artifact = "orderID" id = "orderID" fromState = "init" toState = "open_for_item"/>
        </transition>
        <invoke>
        </do>
    </businessrules>
```
3.3.1 Artifact definition

Based on the logical specification of a business artifact introduced in the previous section, Artifact definition is defined and contains the details of a business artifact involved in a particular business process. The syntax of a business artifact is illustrated below:

```xml
<artifact name="....">
  <attributes>
    <attribute .../>
  </attributes>
  <states>
    <state ...>
  </states>
</artifact>
```

A business artifact is defined using a `<artifact>` element. A name of a business artifact is represented by (name) attribute. The business artifact consists of a set of attributes is defined using a `<Attributes>` element as well as a set of states is depicted using a `<states>` element. The `<attributes>` element consists of one or more `<attribute>` elements. A `<attribute>` element provides details of business data that can be stored in each attribute of a particular artifact. The syntax and details of a `<attribute>` element are listed below:

```xml
<attribute name="..." structure="..." Type="..."/>
```

- `name`: defines the name of an attribute of a business artifact
- `structure`: defines data structure if an attribute of a particular artifact. Currently, there are two types of data structure which are a scalar-typed value represented using pair and a nested data structure represented using list.
- *type*: refers to a primitive type of data stored in an attribute of a particular artifact. The accepted types of data are String, Integer, Boolean, Float, and Double.

The `<states>` element also consists of several `<state>` elements. Each `<state>` element provides details of a possible state in a particular artifact life cycle, including name of state, initial state and final state. The syntax and details of a `<state>` state element are listed below:

```
<state name="..." type="..." />
```

- *name*: depicts a name of a possible state in life cycle of a business artifact.
- *type*: is used to specify whether a state is a starting point (init) or an ending point (end) of a life cycle of a business artifact.

Based on these elements and attribute described above, we can transform a logical specification of a business artifact into an executable form of a business artifact. Here is the example of executable model of business artifact:

```
<artifact name="order">
  <attributes>
    <attribute name="orderID" structure="pair" type="integer"/>
    <attribute name="customerName" structure="pair"
               type="String"/>
    <attribute name="orderItem" structure="list"
               type="String"/>
  </attributes>
  <states>
    <state name="init" type="init"/>
    <state name="add_orderItem"/>
    <state name="end" type="end"/>
  </states>
</artifact>
```
3.3.2 Business rule definition

In the previous section, we have introduced logical specification of business rules which defines how services are associated to artifacts. Business rule is defined in ECA-like rule format. This rule consists of event, precondition and action. Event describes a specific event occurrence that triggers a business rule. Precondition is a condition that must be satisfied before an action can be executed. Action must be undertaken if a precondition is satisfied. In order to execute the conceptual model of artifact-centric business process, the logical specification of business rules must be translated or transformed into an executable format. In this process, we considered adopting RuleML format to define business rules in our executable model. RuleML[16] is Rule markup language being used for sharing rule bases in XML and publishing them on the web. It is designed to be able to be extensible since a minimal set of core language construct which can be applied in every meaningful combination of the respective expressiveness class of the language. It also can be customized for supporting various platform-specific rule languages which make it interchangeable. The language family of RuleML covers the entire rule range, from derivation rules to reaction rules. The family of Reaction rule is involved with the execution of actions in response to events. It states the conditions under which actions must be taken and describes the effect of action executions. There are four rule types of Reaction rule in Reaction RuleML:

- Production Rules (condition-action rules) in the production RuleML subfamily
- Event-Condition-Action (ECA) rules in the ECA RuleML subfamily
- Rule-based Complex Event Processing (complex event processing reaction rules, (distributed) event messaging reaction rules, query reaction rules etc.) in the CEP RuleML Subfamily
- Knowledge Representation Event/Action/situation/Process logic and calculi in the KR Reaction Rule Subfamily

We considered using ECA RuleML to define business rules of our artifact-centric executable model since business rules in the artifact-centric model are Event-Condition-Action rule. The core syntax of ECA RuleML is illustrated as follows:
Event-Condition-Action (ECA) rules are defined by a <Rule> Element which consists of an explicit event part (<on>), a condition part (<if>) with one or more atomic conditions connected via <And> or <Or>, and an action part (<action>). Predicates are n-ary relations introduced via a <Atom> element in RuleML. Our Rule syntax will be in the similar manner as ECA RuleML as shown below:

Our rules are defined using <Rule> element which contains an event part, a precondition, and an action part. This <rule> element has (name) attribute to describe name of a business rule. The trigger event is defined using element <onEvent>. This element has an attribute
called “type” which used to define type of event that can trigger a particular rule. In our framework, there are two types of event that can trigger business rules.

- **Message received event**: this event is detected when an external message is received by a system
- **Artifact changed event**: this event is detected when business artifacts are updated/changed by a service

A `<preCon>` element defines condition that needs to be satisfied before an action is taken. The conditions are introduced by `<atom>` elements. The elements contain several attributes, which constitute atomic conditions of a particular business rule, are listed below:

- **type**: is used to define type of fact that will be evaluated by a rule engine. There are three types of fact in our model which are (state) and (attribute) of a business artifact and (part) of a request message.
- **artifact**: is name of an artifact involving in a rule evaluation.
- **attribute**: is name of an attribute in a particular artifact involving in a rule evaluation.
- **message**: is name of an input message involving in a rule evaluation.
- **part**: is name of a part in a particular input message involving in a rule evaluation.
- **op**: defines logical operators that used in comparison of a fact data and a rule data. The operators consist of `<,<=, >,>=, ==, !=`.
- **value**: defines a value of a particular rule that must be satisfied.

An action part in our business rule is defined by a `<do>` element. It contains a `<invoke>` element which provides details of a service invocation. There are two attributes which are `service` and `operation`. `Service` attribute defines name of a Web service that needed to be invoked if precondition is satisfied. `Operation` attribute defines name of an operation that must be performed if a Web service is invoked. Moreover, a `<invoke>` element has two child elements that are `<mapping>` and `<transitions>` elements. A `<mapping>` element provides details of data mapping between business artifacts and a message that used to invoke a Web service. The syntax of a `<mapping>` element is illustrated as follows:
As shown in the above syntax, there are two directions of mapping which are ArtifactToMessage and MessageToArtifact. They are defined in a type attribute of a <map> Element. The former is used to map attributes of several artifacts to an outgoing request message. The latter is used to map an incoming response message to attributes of artifacts. A <copy> element describes the details of each data mapping for a particular mapping direction. It consists of a <from> element which defines source of data mapping and a <to> element which destination of data mapping. There are four attributes being used by the elements:

- artifact: defines name of a source or destination artifact
- attribute: defines name of an attribute of a source or destination artifact
- message: defines name of an outgoing or incoming message
- part: defines name of a part of an outgoing or incoming message

The next element is an <transitions> element that describes the effect of a service invocation on business artifacts. The element consists of one or more <transition> elements as show below:
Each `<transition>` element provides details of the state transition of each artifact being updated by a particular service using the following attributes:

- `artifact`: defines name of a business artifact involving in a state transition
- `fromState`: defines a current or source state of a business artifact before a service invocation
- `toState`: defines a next or destination state of a business artifact after a service invocation

All these elements and attributes constitute the business rules that control interactions between services and artifacts in our executable model. The example of a complete business rule is illustrated as follows:

```
<transitions>
  <transition artifact="..." fromState="..." toState="..." />
  <transition artifact="..." fromState="..." toState="..." />
</transitions>
```
<rule name="r01-createOrder">
  <onEvent type="MessageReceived"/>
  <preCon>
    <and>
      <atom type="part" message = "createPurchaseRequest" part = "customerName"
          op = "not-equal" value = "null"/>
    </and>
  </preCon>
  <do>
    <invoke operation = "createOrder" service = "createOrderService">
      <mapping>
        <map type = "ArtifactToMessage">
          <copy>
            <from Artifact = "customer" attribute = "Name"/>
            <to Message = "createOrder" part = "customerName"/>
          </copy>
        </map>
        <map type = "MessageToArtifact">
          <copy>
            <from message = "createOrderResponse"
                  part = "orderID"/>
            <to artifact = "order" part = "orderID"/>
          </copy>
          <copy>
            <from message = "createOrderResponse"
                  part = "customerName"/>
            <to artifact = "order" part = "customerName"/>
          </copy>
        </map>
      </mapping>
      <transitions>
        <transition artifact = "order" fromState = "init"
                    toState = "add_orderItem"/>
      </transitions>
    </do>
  </rule>
3.3.3 Service definition

According to a concept of artifact-centric process model, one or more business artifacts are updated by services or tasks. The term of services being used here is basically as same as services in the SOA environment. Thus, this definition provides information that is necessary for a service invocation as follows:

− **name**: defines name of a Web service
− **inputMessage**: defines request message of a Web service
− **outputMessage**: defines response message of a Web service
− **operation**: defines a target operation of a Web service
− **port**: defines a port of a Web service defined in WSDL
− **location**: defines a location of a Web service WSDL
− **namespace**: defines a target namespace of Web service

![Fig. 3-6 Data mapping between SOAP Message and Artifacts](image)

However, the current Web service technology does not accept business artifacts as inputs to invoke a Web service. Thus, we need to extract data from business artifacts and map to request message of a Web service. This can be done by using mapping rules specified in business rules as shown in Fig. 3-6. The example of a complete service definition is shown below:

```xml
<map type="ArtifactMessage">
  <copy>
    <from message="callTotalMessage" part="customerID"/>
  </copy>
  <copy>
    <from artifact="order" attribute="orderID"/>
  </copy>
  <copy>
    <from message="callTotalMessage" part="numberOfItems"/>
    <from artifact="order" attribute="quantity"/>
  </copy>
</map>
```
After all these definitions are defined, we are now able to construct a complete artifact-centric business process model that can be executed by a workflow engine. The details of a design and implementation of a workflow engine will be discussed in the next chapter.

3.4 SUMMARY

In this chapter, we introduced the logical specification of artifact-centric business process (a.k.a. artifact-centric process model). The model consists of three core components including a set of business artifacts, business rules, and services. Furthermore, we discussed the design methodology of artifact-centric process model. The first is to discover the key business artifacts and their life cycles. Then, we can use these key business artifacts to develop the artifact-centric process model. In this step, we need to define the informational model and life-cycle model of business artifacts as well as the logical specifications of services and business rules. After the logical specification of artifact-centric business process is completed, the model must be realized to develop the operational workflow. One approach for realizing is to convert a model from artifact-centric to a procedural workflow model. The advantage of this approach is that it can help simplifying implementation tasks. However, we lose flexibility of artifact-centric model. Moreover, the activity-centric model may not contain business logics defined in artifact-centric model correctly. From these drawbacks, we considered realizing the artifact-centric process model without converting the model. Thus, we can still preserve business logic and flexibility of the artifact model. To support this realization approach, we developed and
proposed the executable version of artifact-centric process model. The executable model has the same components as the artifact-centric process model but it has concrete implementation details which can be used to support operations of a workflow system.
CHAPTER 4:
EXECUTION AND MONITORING OF ARTIFACT-CENTRIC BUSINESS PROCESSES

In this chapter, we discuss a prototype system for executing artifact-centric business processes. Section 4.1 focuses on analyzing realization problems and defining system requirements. Section 4.2 and Section 4.3 focus on implementation of the system and the support for monitoring & tracking of business processes, respectively. Section 4.4 provides a case study for evaluation.

4.1 SYSTEM REQUIREMENT ANALYSIS

Based on our realization approach, there are several problems need to be identified and analyzed in order to define the complete system requirements. However, the goal of our prototype system is to develop the execution engine with a monitoring support. Thus, we now define the requirements related to our goal.

4.1.1 Execution and monitoring problems

In order to define the requirements of a prototype system, we identified the problems related to our goal. We found that there are three main problems have to be addressed. The problems are listed below:

Problem 1: What are the data structures of running instances and how to create running instance?

During the process execution, a workflow engine needs to maintain workflow data to support several operations and coordination in the engine. In the traditional workflow engine [17], a process instance and activity instance are used to help storing workflow data. However, core components of the artifact-centric process model are far more different from the tradition process model because the artifact model consists of artifacts, business rules and services. Thus, only Process Instance and activity instance are not enough to support an execution of an artifact-centric business process on the regular workflow engine. We need
to design new running instances and their data structures. Moreover, the relationship between artifacts, business rules and services are not explicitly defined. It is really difficult to understand overall traces for each step of the process execution. Thus, we must incorporate this relationship aspect into a design as well. As a result, these running instances not only provide storage for workflow data but also describe a relationship between them in each step of the process execution so that they can be used to support executing and monitoring of artifact-centric business processes. After we have a complete design of the running instances, we also need to consider on how we can create these instances in a system. The creation of the running instances is involved in reading and parsing information from the definition file that is defined in XML language. This is going to be another problem to make sure that data from the definition file can be mapped correctly into the instances.

Problem 2: How to coordinate execution of business process?

In the traditional approach, coordination of the business process is explicitly defined using control flows. In contrast, an artifact-centric business process uses business rules to coordinate interaction between services and artifacts. Thus, business rules should be able to describe which service is invoked and which artifact is changed under what conditions. To integrate a rule engine into the system, this requires not only investigation on how rules can be defined in the most expressive and effective manner but also selection of an appropriate rule engine. This is going to be an issue in this implementation of the system. Moreover, there are three basic constructs of the artifact process model including artifact, business rule and service. These constructs will be used in a running instance to support several operations in a system. Thus, the mechanism for coordinating running instances is also needed to ensure a system can provide the consistent process execution. This is going to be another issue in our implementation.

Problem 3: How to collect and correlate the business data and how to access and search the collected business data?

Monitoring and tracking of run-time execution is one of desirable functionalities to allow stakeholders to observe running business processes and provide corrective action,
adjustments and optimizations to processes. In the traditional approach, monitoring and tracking requires technique for aggregating all business data that are relevant to real-time computation of business metrics. To achieve this goal, this usually requires several technical experts and business domains experts. In contrast, business artifact can provide the holistic view of both data and business process. Thus, business artifact is quite suitable to be used to support monitoring a business process. In this implementation, we focus on providing implementation to support our assumption regarding the use of business artifacts in monitoring business processes. There are several issues to be concerned in order to make use of the artifacts. Each execution step is involved with a set of artifacts, a business rule and a service in the artifact-centric business process. To gain a complete trace of the process execution, this requires a mechanism to collect and correlate data from these separate components. This task is important since the artifact model is implicit and it is really difficult for stakeholder to understand what happen recently in the process execution. By collecting and correlating data, we not only provide the execution trace of an artifact-centric business process in a comprehensive way but also facilitate the business process analysis. After we gather the process relevant information, we need a technique to access and search the collected data to enable stakeholder to make use of these data to analyze business processes.

### 4.1.2 Artifact-centric process System requirements

Based on the problems stated in the previous section, we define several requirements for a system for executing artifact-centric business process. These requirements are listed below:

- **R-1.** The system must be able to read and parse the definition file in order to create running instance to support operation of a workflow engine.
- **R-2.** A system needs to be able to use rule engine to coordinate sequence of process execution. It also needs to be able to coordinate running instances to ensure the consistent process execution.
- **R-3.** A system needs to have a mechanism to collect and correlate the process data as well as a technique to query the collected data.
4.2 DESIGN & IMPLEMENTATION OF ARTIFACT-CENTRIC PROCESS SYSTEM

In this section, we discuss how we design and implement a system for executing artifact-centric model. Firstly, the system architecture is introduced. Then, the discussion turns to how this architecture can address the requirements. Java programming language is used to implement the system since various tools, e.g. Drools, Axis2, used in our implementation are written in Java.

4.2.1 Overview of the system architecture

We show our proposed architecture of Artifact-Centric Process System (ACP System), as illustrated in Fig. 4-1. This system architecture ensures that we can address those requirements in the previous section. We adopted the concept of the event-driven architecture and service-oriented architecture. The system composes of six components. Here, we describe each ACP system’s component in more details.

Fig. 4-1 Architecture of ACP System
Process Deployer is used to deploy ACP Model definition file. In the process, a definition file will be validated against a schema of an ACP process. Then, a definition file will be registered and stored in the ACP Model repository. The definition file will be parsed to generate running instances of a particular business process. Business Rule Engine provides a rule evaluating functionality. For any change in a running process, a rule engine will evaluate an instance in order to determine the next possible action that will be undertaken. Process Controller is used to manage instances of a process based on rule engine’s command. Once the process controller receives a command from rule engine to start a new process execution, it will use a process factory to create running process instance. The factory will identify the corresponding instance and create it. The created process instance will be given an id for identification purpose. The process controller can issue a command to an artifact controller to update artifact instances or to web-service controller to invoke Web services. For any changes in a running process, the process controller will consult the rule engine to determine the next possible action. Web service Controller is used to invoke Web services to process artifact data. To invoke a Web service, the controller creates a soap message corresponding to message definition in WSDL. The artifact data is mapped to message data using a mapping rule. Finally, the request message is sent to a designated Web service. Once a response message is returned, the service controller processes the message, and returned message data is mapped to corresponding artifact attribute. Artifact Controller is used to manage and update artifact (which is stored in external repository). After the service controller receives a response message from a Web service, a data mapper will extract the message. Then the artifact controller uses such data to update corresponding artifacts. Front-end UI Interface, proposed in our previous work [10], is used to manage web-based interactions between ACP system and users, which includes automatic generation of web pages and receiving/responding via web form interfaces.
4.2.2 Details on the prototype implementation

Based on the proposed architecture, we have developed the prototype system. In this section, we discuss how the components of the proposed architecture can address the requirements and how they are implemented.

4.2.2.1 Creation of running instance

To address the requirement R-1, we, firstly, have to develop the instance classes that will be used to represent the core components of artifact-centric model. In our implementation, there are four classes to support operation of our system as illustrated in Fig. 4-2. These classes not only store workflow data and business data but also store relationships between artifact, business rule and service involved in each step of the process execution.

Fig. 4-2 Class design of running instance of ACP System
The first type of instance classes is \emph{Process instance}. Process instance acts as a container to store other running instances, which are artifact instance, rule instance and service instance. When the process is enacted then the system initially creates process instance. Once a process instance is created, it is given its name corresponding to the executed business process as well as an identifier key. The identifier key is assigned differently to each process instance created in the system. Thus, the process controller can use this key to distinguish the corresponding process instance and to support coordination in the system. The process instance contains the workflow data and business data involved in a particular process execution.

The second instance is \emph{Artifact instance}. In a process, an instance of particular artifact class can be created at the time that the process is initialized or after service invocation (that performs a creation of artifact). Newly created artifact instance is populated with artifact definition data and given an artifact identifier key. This instance serves a purpose of storing information including business data and lifecycle during each step of business process execution. Thus, it is a key to indicate or track progress of a running process. Next, \emph{Service instance} is instantiated and assigned the identifier key when designated service is invoked (as defined by the action in a business rule). It not only stores service invocation information defined in a service definition but also captures input/output message data. This input/output message data stored in a service instance can be used to track effect of service operate on artifacts in a particular step of process execution.

The last instance is \emph{Rule instance}. Each step of process execution is involved with a particular business rule. When pre-condition is satisfied, a business rule is fired. Then, rule instance is created and assigned a unique key. This instance contains relationship between service and artifacts involved in a business process. This information is be used by a system to associate corresponding artifacts with a designated service. Thus, we can describe what artifact and service needed in a current execution step and define what artifact attributes needed in a service invocation. After a service invocation is completed, the rule instance also provides information regarding the state transition for involved artifacts. This allows us to synchronize state of the artifacts during the process execution. Moreover, timestamp is assigned during the time of rule instance creation. This can be used to keep track a sequence of the business process execution.
After we have a complete set of running instance class, our discussion turns to how we can create these running instance objects in a system. In our implementation, we use a concept of factory pattern to address this requirement as illustrated in Fig. 4-3.

Fig. 4-3 Illustration of ACP Factory Operation
The ACP factory class has been developed and has method regarding creation of process instance, artifact instance and service instance. While a running instance object is being created, the factory needs to map information from a definition file to the object. We adopted Java Architecture for XML Binding (JAXB) to address this requirement. There are several advantages of JAXB. JAXB can help simplifying access to an XML document form a java program. It uses memory much efficient and is flexible. Firstly, we use JAXB compiler to generate java classes from the ACP schema. When the ACP factory’s method is invoked, JAXB API is, then, invoked to un-marshal an ACP definition file to create a tree of content objects that represent the content and organization of the designated ACP definition. Finally, the ACP factory creates running instance objects and map data from the content object to the instance objects. Thus, we are able to create running instances to support operation of our system.

4.2.2.2 Rule engine integration

In artifact-centric business process, business rules are main mechanism to control interaction between artifacts and services. To address the requirement R-2, we decided to use a rule engine for our system because there are several advantages on using a rule engine. A key advantage is that rules are easily used to express solutions to difficult problems and consequently have those solution verified since rule languages are very close to natural language. Rules are much easier to read and understand than code. Rule engines allow us to achieve data and logic separation. Data is stored in domain objects and the logic is stored in the rules. Thus, the logic can be much easier to maintain and make changes in the future since the logic can be organized in one or more rule files. A centralized repository of knowledge or rule can be developed to provide a single access point to rules that are stored and serve as documentation. The rule algorithm such as Rete algorithm provides very quick ways of matching rule pattern to object data. Moreover, there are several tools such as Eclipse that provide easy way to edit, manage, validate, audit rules during the process of rule design. Rule engine also provide logging functionality to make sure the decision can be logged. The log is able to describe what decisions are made and why the decisions are made.
We have chosen and integrated Drools rule engine [21] since the engine provides very efficient ways of evaluating business rules. The rule format is also easy to comprehend and can be written in XML format and drool format. Drools engine conforms to JSR94 standard and provides a set of APIs that allows us to integrate it to our system. The rule engine works in two steps as shown in Fig. 4-4. Firstly, once a system is started, all rule files are loaded into the knowledge base of a rule engine. The knowledge base serves as storage of rule engine during a system run-time. The rule engine is activated once it receives an internal and external event generated by the system. There are two types of events involved in artifact-centric business process. Thus, we developed two custom java event objects to support both event types.

![Fig. 4-4 Rule Engine and Its operation](image-url)
The custom event objects provide several methods to help accessing domain data objects such as artifact instance, soap message. This reduces complexity in the process of rule design and a rule engine can access to data much efficient. During the activation, a rule engine creates a new rule session from the knowledge base and an event object is inserted as an input of a rule engine. An event data is validated against the constraints of the rules by the process of pattern matching. If conditions are satisfied, an action will be undertaken to make changes to artifacts. A rule instance that keeps track of rule execution is also created in the process as well. Finally, the rule session is terminated.

4.2.2.3 Coordination of running instance

In order to coordinate running instances stated in the requirement R-3, we address this issue using process controller, artifact controller and service controller. Here, we will use our online ordering example to describe how these controllers work together to coordinate all instances that are created during a process execution. Once the ACP system receives a request from a user to start an ordering process, the rule engine evaluates the request. The rule \( r1 \) is triggered by the request. This rule defines how order artifact can be created. If preconditions of rule \( r1 \) are satisfied, the rule engine issues a command to the process controller to start or run a process execution of an ordering process. The rule instance of rule \( r1 \) is also created at this point as well. Then, the process controller invokes the process factory. The factory identifies the deployed process and instantiate a process instance for an ordering process as shown in Fig. 4-5. After receiving a corresponding process instance, the process controller initializes a unique process id and registers the process instance to a list of running process instance. Once a process instance is registered, an instance of rule \( r1 \) is added to a list of rule instances.
The process controller checks whether a new artifact instance is needed to be created or not as illustrated in Fig. 4-6. If yes, the factory is invoked. If not, the process controller continues the rest of its operation. Based on rule \( r1 \) in our online ordering process, the artifact instance of an order artifact is needed to be created by the factory. Then, the artifact instance is added to a list of artifact instances. The artifact instance is given a unique id that is used for identification purpose and its state is set to \textit{start}.

Fig. 4-5 Creation of a process instance during process execution

Fig. 4-6 Creation of an artifact Instance during process execution
Next, the process controller orders a web-service controller to invoke the *createOrder* service. The service instance of *createOrder* service is instantiated and added to a list of service instances. These steps are illustrated in Fig. 4-7.

![Fig. 4-7 Service Invocation during process execution](image)

The service controller will communicate with the artifact controller to retrieve data from corresponding artifacts. The *createOrder* service is then invoked. Once a response is returned, a service controller will send message data to an artifact controller to update the artifacts. The unique identifier key ensures that correct artifact instance is being updated. A new artifact instance will be generated if necessary during this step as well. Mapping rule defined in rule *r1* controls the data mapping between service input-output and artifact instance. The following figure represents the steps described above.
Once finishing data updating, the artifact controller will update a state of an order artifact from start to open_for_item. After completion of artifact updating, the artifact controller sends a signal to the process controller. The process controller will generate Artifact_change event to trigger rule engine to continue a process until ordering process is completed.

As we can see that each step of a process execution, the ACP system creates artifact instances, service instances and rule instances. These instances can be used for monitoring a process since they contain overall information. To prove our concepts, we developed a prototype of ACP system and generated a test case based on the motivating example. After a test case is executed, we are able to process data from running instances and generate a log file. In a log file, we can see detailed information for each step of a particular business process execution. To generate this log file, the system needs to capture data from rule instances, artifact instances and service instance at run-time. A rule instance contains identifier keys that belong to involved service instance and artifact instance. These keys help defining a relationship between rule instance and the other instance in each step of a process execution. This enables our system to be able to generate a record for each step during run-time as shown in Fig. 4-9. Each record is stored in a log file. Pre and Post-artifact are recorded in a log file to show progress of each artifact from initial state to final
state. The system records these data before and after service invocation. We will discuss how we make use of this log file in the next section.

Fig. 4-9 Sample of Log record
4.3 Monitoring & tracking support of artifact-centric business process

In the previous section, we discussed how the artifact model can be realized in the prototype system. Now, we tackle the monitoring & tracking challenges to show that business artifacts can help supporting in the area of business process management.

4.3.1 Business process monitoring approach

Monitoring and tracking of process execution is critical for enterprises to ensure that business policies, efficiency and reliability goals are satisfied. The purpose of monitoring and tracking \[40,41,42,43\] is to understand what actually happened during the life cycle of a process by examining how data are produced, what resources are involved and which tasks are invoked. A common practice of monitoring and tracking business process \[59\] is that the business data need to be collected during the execution of the processes controlled by a process engine or it may be accessed from an existing data warehouse and then combined with process data. Next, the data are analysed, consolidated, aggregated based on predefined rules or patterns. Finally, the data are displayed on a dashboard. Thus, we can summarize the requirements for monitoring and tracking as follows:

\[ \]

R-1. The technique for collecting business data must be developed.

R-2. The collected data must be correlated and aggregated into an appropriate format or pattern for querying or displaying information to users.

R-3. Once the data are prepared in a suitable format, the technique for querying or displaying monitoring information must be developed.

To support this functionality, business data must be collected during the process execution as stated in the requirement R-1. In the activity-centric approach, we need to have a data collecting mechanism to gather all business-relevant data. However, since we use business artifacts as basis of business process model, we are able to capture all business data without the need of sophisticated technique to collect business data. In order to use the collected data, the data must be correlated and aggregated into certain patterns as stated in the requirement R-2. We address this requirement by developing a mechanism to generate a
log file for supporting monitoring and tracking. The log file is used to correlate and aggregate all business data and process data. The log file is structured using XML language. It composes of several records sorted in chronological order. Each record represents a step of the process execution. One record contains five types of information including timestamp, rule, service, pre-artifact, and post-artifact. Timestamp stores information regarding when a particular step of execution started. Rule and service contain information regarding which rule is fired and which service operates on artifacts as a result of fired rule. Pre-artifact represents condition of artifacts before being updated by a service and post-artifact represents condition of artifacts after being updated by a service.

Since a log file not only stores details of rule, service, artifacts involving during a particular step of process execution but also describes relationship between them, it enables us to analyze completed process and monitor running processes. Thus, we address the requirement R-3 by defining a set of queries to extract information from the log file. In order to define the queries, we must observe and understand what kind of questions and queries the stakeholders ask during the process execution or after the process execution. Based on our observation, there are several tracking and monitoring queries that can be asked and we categorize them into four categories. The first type of queries is to track current status of running process. This allows stakeholders to track current status of running processes. The second type is to support calculation of process or task duration. This query can calculate duration of each task and stakeholders can use it in performance analysis. The next query type is cause-effect analysis. This allows stakeholders to perform deep analysis of business processes. The final type is aggregation. This type of query can be used by stakeholders to produce report and summary. In order to use these query on a log file which is structured using XML, we adopted XQuery [36] since it is a query and functional programming language that is designed to query collections of XML data and is capable of extracting and manipulating data from XML documents similar to SQL operates on relational databases. It utilizes XPath expression [37] to access specific parts of an XML document and FLWOR expression for computing joins between two or more XML documents and for restructuring data. Moreover, it has been used in several papers [44,45,46,47] to query process data.
4.3.2 Monitoring and tracking queries

Let’s us demonstrate on how we can use XQuery to support these types of query in more details. To demonstrate the use of XQuery, Firstly, we generate some sample questions that are usually asked by stakeholders based on the online ordering process. Then, we organize these questions into 4 categories as follows:

**Status tracking**

During the process execution, it is quite common that status of processes need to be tracked or monitored by stakeholders. So, stakeholder can analyse and ensure the correctness of running processes. Some questions that can be asked are listed below:

**Q 1:** What is the current status of the process with identifier \(x\)?

Stakeholders want to know the current status of a running process. Thus, we need to query an `order` artifact that is a key business artifact and can be used to represent the current status of the running process. We can structure a query in this pattern.

```
for each record in a log[@process_id= x] in a collection of logs
let $max_no. = max(record/@number)
where record/@number = $max_no.
return
  record/post_artifact/order/@state
```

This query iterates through each `record` in a log of process with id \(x\) and returns state of `order` artifact that belongs to the current `record`.

**Q 2:** What is the current status of each order item in the process with identifier \(x\)?

Stakeholders want to know current status of each order item in the running process. To answer this question, we need to query a latest record to be able access all `orderItem` artifacts.
for each record in log[@process_id= x] in a collection of logs
let $max_no. = max(record/@number)
where record/@number = $max_no.
return
{
    for each orderItem in record/post_artifact
    return
        orderItem/@id
        orderItem/@state
}

This query first iterates through each record in a log of process with id x and returns the current record. Then, we loops through all orderItem artifact stored in post_artifact and return identifier key and status of each orderItem artifact.

**Duration**

To evaluate the performance of business processes, stakeholders need to calculate the duration of each process. Then, they use this information to generate a performance report. Some basic queries that can be asked are listed below:

**Q1:** How long does it take to complete the process with id x?

Stakeholders may want to know duration of a particular process. Thus, we need to calculate duration using timestamp of starting date and end date.

let $records = records in log[@process_id = x] in a collection of logs
let $startDate = $records[@number = min(record/@number)]/timestamp
let $endDate = $records[@number = max(record/@number)]/timestamp
let $duration = $endDate - $startDate
return
    $duration
This query basically retrieves starting date and end date. Then, duration is calculated from these two points of time. Finally, a result is returned.

**Q 2:** How long does it take to complete an order item with id y in the process with id x?

This question is asked when stakeholders are interested to find out Total time is needed to complete a particular item in the process. This type of question will be used when stakeholder would like to analyse a performance aspect of a business process.

```plaintext
let $records = records in log[@process_id = x] in a collection of logs
let $startDate = $records/timestamp[record/pre_artifact/orderItem
  [@id = y and @state = init]]
let $endDate = $records/timestamp[record/post_artifact/orderItem
  [@id = y and @state = shiped]]
let $duration = $endDate - $startDate
return
  $duration
```

This query basically retrieves starting date and end date of orderItem artifact with id y. Then, duration is calculated from these two points of time. Finally, a result is returned.

**Cause-effect analysis**

Sometimes, issues are occurred during or after the process execution. Stakeholders usually want to understand the causes of the issues. So, they can prevent the issues that will happen in the future. Some questions that can be asked by stakeholders are listed below:

**Q 1:** What is a service that updates state of order artifact with id y from state s1 to state s2 in the process with id x?

Stakeholder may want to understand how a particular artifact is changed from one state to the next state. This kind of question may be raised when some issues occurred during the process execution and stake holder want to track causes of the issues.
for each record in a log[@process_id= x] in a collection of logs
where record/pre_artifact/order[@id=y and @state=s_1]
    and record/post_artifact/order[@id=y and @state=s_2]
return
    record/serviceId

This query iterates through a set of records to search for a record that has order artifact with id = y and its state move from state s_1 to state s_2. Then, a responsible service stored in the record is returned.

Q2: How is information of each artifact before being updated by service y in the process with id x?

This question may be raised when stakeholder want to know regarding information of business artifacts before being updated by a service. Stakeholders can use an answer of this question to support root cause analysis of a particular issue.

for each record in a log[@process_id= x] in a collection of logs
where record/serviceId = y
return
    record/pre_artifact/*

This query loops through each record and retrieve a record that has a service y. A result is returned as a set of artifacts in a state prior the service invocation.

Report/Summary

Normally, a report or summary of the operation must be documented on a monthly or yearly basis. Stakeholders need to summarize all information regarding the business process and its operation. Thus, stakeholder may ask some questions that are listed below:
Q 1: How many processes have been completed so far?

This question may be asked when stakeholders want to know how many processes are completed and use this information to generate a report or to monitor a system at run-time.

```xml
let $logs = a collection of logs
let $completeProcess = count($log/log/record[@number =
    max(log/record/@number)]/
    post_artifact/order[@state = complete])
return $completeProcess
```

This query basically counts number of processes that marked as completed. Then, it returns a result.

According the demonstration, we can see that a log file generated from ACP system can be queried by XQuery and can help facilitating monitoring and tracking if it is necessary. Therefore, this is a proof to illustrate a use of business artifacts to support monitoring and reporting.

4.4 CASE STUDY AND EVALUATION

In this section, we have developed a case study to test our prototype system that realizes an artifact-centric process model using our direct approach. Moreover, we evaluated and compared our realization approach to the model conversion approach.

4.4.1 Case study

In this section, a case study is based on an online ordering process. This business process is modeled using Artifact-centric Executable model that we proposed in Chapter 3. This process starts when a customer lodges an order through a web site. Then, the order is processed and delivered to the customer. This process is constructed using three key artifacts. These artifacts include Order, Invoice, and Shipment. Moreover, a set of business rules and services are also defined. After a process is completely constructed, it is deployed
to the prototype system. The deployed process is exposed to a public via a Web service interface. Thus, each client can interact with a running process through this interface as shown in Fig. 4-10.

```
<?xml version="1.0" encoding="UTF-8"?>
<registeredProcess>
  <process>
    <name>order_process</name>
    <location>C:/Users/KanPhd/DrPhd</location>
  </process>
</registeredProcess>
```

**OnlineOrderService**

Available operations
- addItem
- placeOrder
- confirmPayment

Fig. 4-10 Process registration and a Web service interface of the process

A simple web application provides the web-based interface to a client. When a user wants to place an order, he/she can simply fill a form and submit it. Then, a soap message is generated and sent to the system via the interface. The system processes the request and calls an appropriate service to update business artifact. Then, the system records a trace of process execution in a log file.

![Fig. 4-11 Web User Interface for a client](image)

Fig. 4-11 Web User Interface for a client

Fig. 4-11 shows the web user interface that provides a starting point of the process. After a client submits the information, a message is evaluated by a rule engine. In this case, the first rule is triggered when the message is received by the rule engine. The pre-condition of the first rule is that each part of the message including *name* and *address* of a customer must not be null or empty. The message is validated against this pre-condition. When the
pre-condition is satisfied, the rule is fired. The result of rule firing will cause \textit{CreateOrderService} to be invoked. An order artifact is created and contains no information. Then, it is updated by the service. The system records the states of artifacts before a service invocation as well as the states of artifact after a service invocation as illustrated in Fig. 4-12.

\begin{verbatim}
<log process_id="order_process-P1">
  <record number="1">
    <timestamp>2011-11-03, 13:49:36</timestamp>
    <ruleId>r01-createOrder:R1</ruleId>
    <serviceId>createOrderService:S1</serviceId>
    <pre_artifact>
      <order id="" state="init">
        <orderId>null</orderId>
        <orderItems>null</orderItems>
        <customerAddress>null</customerAddress>
        <customerName>null</customerName>
        <grand_total>null</grand_total>
        <amount_paid>null</amount_paid>
      </order>
    </pre_artifact>
    <post_artifact>
      <order id="order:123" state="open_for_item">
        <orderId>123</orderId>
        <orderItems>null</orderItems>
        <customerAddress>1/24</customerAddress>
        <customerName>Ken Ngamakur</customerName>
        <grand_total>null</grand_total>
        <amount_paid>null</amount_paid>
      </order>
    </post_artifact>
  </record>
</log>
\end{verbatim}

Fig. 4-12 Pre and Post Condition of Order Artifact in the first step of the process

As a result of the service invocation, attributes of \textit{order} artifact are assigned and its stage changes to \textit{open_for_item} state. This state indicates that \textit{order} artifact is opened for a customer to add their desired products. After completing this step of the process execution, the system may either wait for an incoming or keep continuing the process execution. The system keeps recording all execution traces from the start to the end.
Fig. 4-13 illustrates the last step of the execution where an order artifact is updated by `closeOrderService` and its state is moved from `all_shipped` to `completed`. You can also see from the figure that once the process execution is completed, all business artifacts involved in this process are in their final state.

During the process execution or after the process execution is completed, we are also able to perform monitoring and tracking of the current running by using the proposed queries. These queries are able to extract the required information from the log file. For example, a stakeholder wants to know about the current status of the process as illustrated in Fig. 4-14. By providing the process id as input for this query, a stakeholder is informed with the current status. Since the process has been completed, the current status of process
is shown as complete. In Fig. 4-15, a stakeholder wants to know the total duration of the process execution. The process id is provided as input of this query. The result of the query is that it takes 4 days 23 hours and 15 minute to complete the process.
From the case study, the prototype system can execute a business process and provides monitoring and tracking functionality. Thus, this is a proof of the concept that business artifacts can be used to support modeling, execution and monitoring of a business process.

4.4.2 Evaluation

In this section, based on the result of the case study, we discuss the technical evaluation of our ACP system as well as a detailed comparison between two realizations approaches. Note that we do not discuss the advantages and disadvantages between traditional activity-centric approach and the artifact-centric approach in this research. After a prototype of ACP system was completed, we have simulated the case study based on our online ordering process. The result showed that our framework can address our requirements. The developed system is able to manage running instances created during process execution. Each running instance, e.g., service instance, stores process execution data and can be used for purpose of monitoring and reporting. A business rule engine is proved to be able to work solely to provide a decision making that affects running processes. In our current prototype, all decision making process are centralized into a single rule engine. Thus, it simplifies rule management. However, this may raise performance issue of process execution if there are thousands of business rules to be evaluated by a rule engine. Non-deterministic is also an issue since a rule engine fires rules simultaneously. However, their ordering is non-determinism. Thus, sequence of process execution may be different even with the same business process. A task for evaluating reachability of running processes is needed to address this issue. During the case study was being conducted, we assumed that there is no issue regarding to non-determinism. Since business process models are defined implicitly in artifact-centric approach, this is going be another issue for a process modeler. An artifact model does not have any explicit control flows as in the traditional process model so this is not an easy task for the process modeler. Thus, an intuitive process designing tool needs to be further developed. As we know that there is another way to implement artifact-centric processes, we compared our system to this existing artifact system implemented using the model transformation approach described in papers [11,26,58].
– **Realization approach**

In the case study, we used our direct approach to realize an ACP model whereas the opposite approach proposed in [11, 26, 58] attempts the conversion from artifact-centric process model to a procedural model, e.g., BPEL. We are concerned about logical information of artifact-centric process model may be lost during the model conversion process. This is because some logical information which defined in a declarative manner, e.g., business rules, is spilt and mapped into several control flows in a procedural model. Moreover, this conversion task is quite cumbersome, error-prone and time consuming. Our approach ensures that there is no loss of data during transformation of the conceptual model since logical information of the conceptual model can be mapped faithfully to the proposed executable model. Without any model conversion, this approach uses less time and reduces chance of making mistake. Thus, direct approach that was used in our case study is considered to be more appropriate way to realize the artifact model compared to model conversion approach.

– **Flexibility and changes management**

Flexibility is the strength of a process model in declarative style. A conceptual model of artifact-centric business process owns this advantage as well since it is specified in the same style. Direct approach that we used to realize the conceptual model guarantees that the executable model inherits flexibility from its conceptual model, whereas the other approach does not since tasks are locked up by control flows. As a result, flexibility is well supported for design-time and run-time for our approach, whereas the other approach partially supports flexibility at run-time as it depends on the functionality offered by a particular workflow system. Thus, Changes can be made directly on the implementation level in our direct realization approach. In contrast, changes have to be made at design-time and then converted to the implementation if an artifact model is realized in a procedural workflow.

– **Monitoring and Reporting**

As opposed to traditional approach for process modeling, an artifact-centric process model focuses on business artifacts as its first class citizen to model a particular business
process. Each business artifact contains business-relevant data and its life cycle. Artifact data and life cycle of each artifact reflect the progress of a particular process toward a business goal. Thus, business process monitoring and tracking can be done by inspecting artifacts. Our approach provides a feature of direct and consistent monitoring and reporting at both model and instance level since both data and life cycle are combined at model level and instance level. The case study illustrated that a particular process can be monitored by directly inspecting running instances at run-time without any complicated technique involving data gathering and processing. Whereas, the other approach needs a sophisticated mechanism which may include retransformation from the implementation specification back to its model specification and backward mapping for some data to its model to gather and process all process information to provide monitoring and reporting functionality.

– **Verification and conformance checking**

Verification and conformance checking is an essential task for both traditional approach and artifact-centric approach for modeling business process to ensure validity of developed model so that it can be realized on an automated system to support decision making for a particular business process. Since we used direct approach to realize artifact-centric business process model, single model verification for both design-time and run-time can be achieved because the implementation level reflects its conceptual level. Thus, conformance checking can be achieved directly, whereas the other realization approach needs to have separate verification on both Artifact model and procedural model. Run-time verification does not reflect the base artifact model because of the conversion. Therefore, conformance checking needs some additional procedures.

– **Standards and technologies support**

Although our approach has several advantages, there are some drawbacks regarding to standards and technologies supports. Artifact-centric model realized on traditional workflow benefits from current industry-wide standards and technologies, e.g., OASIS, OMG, W3C and etc. Thus, an implementation of this realization approach is much easier and faster than our approach. Moreover, interoperability and execution in distributed environment are well supported when the artifact model is realized on traditional task-
based workflow system. Currently, the developed prototype system only supports execution of an artifact-centric business process model in local environment and need further extension to handle distributed executions.

4.5 SUMMARY

In this chapter, according to the artifact-centric executable model, we identified the problems regarding realization of artifact-centric process model and analyzed the system requirements. The first requirement is that the system must be able to read and parse the definition file in order to create running instance to support operation of a workflow engine. Secondly, a system needs to be able to use rule engine to coordinate sequence of process execution. It also needs to be able to coordinate running instances to ensure the consistent process execution. Lastly, a system needs to have a mechanism to collect and correlate the process data as well as a technique to query the collected data. Then, we developed a prototype system so called Artifact-Centric Process System (a.k.a. ACP system). The system consists of five core components. Business Rule Engine provides a rule evaluating functionality. Process Controller is used to manage instances of a process based on rule engine’s command. Web service Controller is used to invoke Web services to process artifact data. Artifact Controller is used to manage and update artifact (which is stored in external repository). Front-end UI Interface is used to manage web-based interactions between ACP system and users.

We also introduced a set of running instances that stores workflow data and business data. Process instance acts as a container for other running instances. Artifact instance stores information including business data and lifecycle during each step of business process execution. Thus, it is a key to indicate or track progress of a running process. Next, Service instance not only stores service invocation information defined in a service definition but it also captures input/output message data. This input/output message data stored in a service instance can be used to track effects of services operates on artifacts in a particular step of process execution. Rule instance contains relationship between services and artifacts involved in a business process. This information is used by a system to associate corresponding artifacts with a designated service.
Since traces of the process execution are stored in XML format, we introduced a set of XML queries to allow a user to extract information from a log file to support monitoring and tracking.

Lastly, we used a test case to test our prototype and the result was that the system was able to execute and manage a business process correctly. Moreover, we were able to query the required information from a process log file to answer the questions regarding monitoring and tracking of a running process.
CHAPTER 5:
EXTENDING ACP SYSTEM FOR SUPPORTING INTER-ORGANIZATION PROCESS

5.1 OVERVIEW

Over past several years, service-oriented architecture (SOA) has been serving an increasing trend of IT for businesses to meet the challenges of the ever-changing market [33]. Service-oriented architecture particularly enables the business collaboration across organizations by the means of Web service composition to achieve the mutual goal of collaboration as well as the individual goal of each participant.

Currently, there are two conventional standards to support the collaboration. Service Orchestration is the process of associating several existing services in a non-hierarchical arrangement using a software tool to develop a new application with a centralized set of goals to be achieved. The result application is an executable business process that contains business logic and task execution order. The process can interact with both internal and external Web services at the message level. Moreover, it can span applications and organizations to define a long-lived, transactional, multistep process model [32]. Orchestration always represents control from one organization’s perspective. There are several languages and standards to define service orchestration such as BPEL [27, 28]. Service choreography is used to specify the interaction between business parties whereby providing the agreed behavioral contract between collaboration organizations. Service choreography technology intends to provide a global view of the collaboration and allow each party to design and implement its own portion [34]. One example of the languages that can be used to define the choreography is CDL [35]. However, both existing modeling approaches and languages mainly describe the collaboration from the procedural perspective, and focus on control-flow.

In recent years, an artifact-centric approach for modeling business processes has emerged. It delivers a higher level of robustness and flexibility for describing process specification than traditional process-centric approaches [1,4,5,25]. Artifact-centric process modeling focuses on the changes of business artifacts involved in the processes. These
changes on the artifacts are made by services. The specification of how artifacts are involved and how services operate on them can be described in declarative rules, e.g., ECA rules. To support the business execution across organization, we observed that we are able to exchange attributes of the artifact used in an organization’s internal process between participating organizations using a message. Moreover, we are able to specify when we can exchange/share these attributes since business artifacts have their life cycles and we use states in the artifact life cycles to specify the point that we have to exchange information. From this observation, we considered extending our previous work to support execution of business processes across organizations. In this chapter, Section 5.2 describes the research problems and our approach. We investigate a model that serves as a contract between organizations in Section 5.3. We explain design and development of an extended framework in Section 5.4. Section 5.5 introduces a case study and the evaluation will be made based on the case study.

5.2 Issues & Approach for Inter-Organizational Process

5.2.1 Inter-organization process problems

In the previous chapters, we described how to realize artifact-centric business process model. Although the tasks that contribute to an artifact-centric process is distributed, the process execution still focuses on a single enterprise. However, there have been increasing trends for collaborating across organizations to achieve their mutual business goals. For the traditional approach, this collaboration can be realized by means of service orchestration and service choreography. When it comes to artifact-centric business process approach, there are some challenges need to be investigated.

Problem 1: What is structure of executable model for inter-organizational process?

A business process in the inter-organizational environment involves interactions among multiple parties. In the traditional workflow, these interactions are defined explicitly in the collaborative process model such as BPEL and CDL. Basically, the model serves as a contract between participating organizations. This method allows a process execution to be
consistent across multiple enterprises. In artifact-centric approach, core components of the process model are separate units. Therefore, the process model becomes more implicit compared to the traditional process model. To execute the implicit model across multiple parties, we can certainly adopt the concept of a contract from the traditional approach; however, there is an issue regarding how to describe these interactions in the implicit model of artifact-centric approach.

**Problem 2:** How to coordinate artifact-centric process systems in different organizations?

During the process execution, the workflow engines distributed across organizations need to be coordinated in order to achieve their mutual business operational goals. In the traditional approach, control flows are used to support coordination between roles or participating organizations. Artifact-centric approach uses business rules to control service/task invocations that operate on business artifacts. Basically, we can implement a rule engine to support the business rule execution. The rule engine that serve as a central controller usually works well if a business process is executed by a workflow engine within a single enterprise. However, there will be an issue if business rules are used to provide consistent coordination of workflow execution across multiple enterprises since the nature of business rules is centralized which makes them difficult to be used to provide workflow coordination in the distributed environment.

**Problem 3:** What are the data structures of the running instances of the inter-organizational process?

Similar to workflow execution in a local organization, we need to have running instances that capture workflow information. However, a workflow is executed in the distributed environment which makes it hard to capture this distributed information. Thus, this is going to be challenges. First of all, we need to design data structures of running instances. These data structures must be able to reflect all aspects of a workflow execution taking place in different regions and companies so that they can be used to support coordination in a workflow system. Secondly, we need to share and retrieve workflow information which is distributed across geographic regions or organizations during the
workflow execution. The traditional workflow only focuses on passing parameters between roles or organization. Thus, it is difficult to understand overall aspects of a business process. On the other hand, business artifacts are able to describe these aspects since they contain all business relevant information. One company may need to share its artifacts with its partners in order to provide an insight of a current running business process. However, this is not a simple task since a company does not want to share some sensitive information with other companies. Thus, a mechanism for selectively sharing business artifact data needs to be investigated.

5.2.2 The approach for executing inter-organizational process

In this section, we discuss our approach to address the challenges defined in the previous section. To the best of our knowledge, the common practice to address this challenge is to develop a contract that serve as an agreement between roles.
The contract usually contains information regarding roles, their interactions, and data exchange between roles. Thus, the first step in our approach is that we must define a business artifact-based contract for inter-organizational business process as shown in Fig. 5-1. The contract in our approach should contain a set of business artifacts that captures data exchanged between roles and status of the inter-organizational process. The rules are also included into the contract to define interactions between roles as well as details of data exchange. Lastly, roles are needed to provide details of participating organization. Once a contract is completed, we aim to develop a system for executing a contract. The system should coordinate the process execution between different systems. Moreover, it should create and maintain data and status of the process during the process using a set of running instances. Also, the instances can be used to provide monitoring and tracking of the running process.

5.2.3 Requirements for inter-organizational process

Based on the approach stated in the previous section, we summarize our requirements for an extended framework to support inter-organizational business processes. The requirements are listed below:

R-1. The executable model that will serve as a contract between participating organizations need to be developed. This model will be used by a system to support business process execution in inter-organizational environment.

R-2. A system needs to be able to coordinate operations of workflow engines in different organizations.

R-3. A system needs to be able to maintain workflow execution data during run-time. Moreover, it is able to capture workflow data distributed among organizations.
5.3 Artifact-Centric Executable Model for Inter-Organization

According to the first requirement R-1, the executable model that serves as contract between participants needs to be developed. This contract is a long term agreement of all organizations involved in the inter-organizational business processes to ensure each organization performing their part of business process correctly. In this section, we focus on defining a contract for supporting artifact-centric business process in the inter-organizational environment. To define the contract, we adopt the concept of shared artifact and local artifact proposed in [22]. Local artifact is used and managed internally within its own organization. The local artifacts can be used to support the coordination of business process execution in the inter-organization environment. Shared artifact is associated with various participating organizations. The shared artifact serves as a contract between organizations and can be used to indicate progress towards the completion of the collaborative process. In essence of our framework, local artifacts are created and managed by a local ACP system. Shared artifacts serve as a contract for ACP systems involved in a collaborative business process. The shared artifact contains two types of information. The first type is shared business data that are in interests of business partners. They are stored in corresponding attributes of the artifact and can be passed between stakeholder organizations during the process execution. The second type is its life cycle which indicates progress of a shared artifact towards its collaborative business goals. It also reflects progress of a local artifact managed within an organization. To define a complete contract, shared artifacts must be associated with a set of business rules and roles. Business rules are used to regulate state transition of shared artifacts. Pre-condition of each business rule is used to verify a current status of shared artifacts. If the status of the artifacts satisfies the condition of a business rule, a participating role is notified to take over a control of shared artifacts. Roles are used to define stakeholders that are responsible for controlling shared artifact for a period of time in a business process execution. Each role provides a set of service that makes changes on both data and life cycles of artifacts. Once a session of the process execution is passed to a particular role, an appropriate service is selected to manipulate artifacts.
5.3.1 Shared artifact definition

Shared artifact definition contains the details of shared business artifact involved in a particular inter-organizational business process. The business information in this artifact is shared from a local artifact. The syntax of a shared artifact is illustrated as follows:

```xml
<shared_artifact name="...">
  <attributes>
    <attribute .../>
  </attributes>
  <shared_states>
    <state .../>
  </shared_states>
</shared_artifact>
```

A shared business artifact is defined using a `<shared_artifact>` element. A name of a shared artifact is represented by `(name)` Attribute. The shared artifact consists of a set of shared attributes is defined using a `<attributes>` element as well as a set of shared states is depicted using a `<shared_states>` element. The `<attributes>` element consists of several `<attribute>` elements. A `<attribute>` element provides details of business data which are shared by participating organization and can be stored in each attribute of a particular shared artifact. The details of a `<attribute>` element are omitted since they are already explained in Section 3.3.1. The `<shared_states>` element also consists of several `<state>` elements. Each `<state>` element provides details of a possible state, including name of state, initial state and final state, in a particular shared artifact life cycle. Explanation of a `<state>` element is already described in Section 3.3.1.

Based on these elements and attribute, we are able to describe shared business artifacts which are used to store business information support execution of inter-organizational business process. Here is the example of executable model of a shared business artifact:
5.3.2 Shared business rule definition

A shared Business rule is used to direct interactions between participating organizations or roles and is defined in ECA-like rule format. This rule consists of event, precondition and action as same as in the business rules defined in the executable model in Chapter 3. However, some details of the business rule are different and will be described later in this section. Event describes a specific event occurrence that triggers a business rule. Precondition is a condition that must be satisfied before an action can be executed. Action is an action or task that must be undertaken by a specific role if a precondition is satisfied. Our shared business rule syntax is shown in the following paragraph:
Our shared rules are defined using a `<shared_rule>` element which contains an event part, a precondition, and an action part. This `<shared_rule>` element has (name) attribute to describe name of a business rule. The trigger event is defined using element `<onEvent>`. The children elements and attributes of a `<onEvent>` element are as same as in Section 3.3.2. Thus, their details are omitted here. An action part in our business rule is defined by a `<do>` element. It contains a `<invoke>` element which provides details regarding interaction between the business partners. There are three attributes which are role, service and operation. Role attribute defines a designated role responsible for the current session of business process. Service attribute defines a service that is provided by a particular role. Operation attribute defines name of an operation that have to be performed by a designated role. Moreover, a `<invoke>` element has 2 child elements that are `<mapping>` and `<transitions>` elements. A `<mapping>` element provides details of data mapping between shared business artifacts and messages that used to coordinate a business process execution. The syntax of a `<mapping>` element is illustrated below:

```
<shared_rule>
  <onEvent />
  <preCon>
    <and>
      <atom ... />
    </and>
  </preCon>
  <do>
    <invoke>...$/invoke>
  </do>
</shared_rule>
```
<coordination>
  <map>
    <from type=""..."" name=""..."" part=""..."" />
    <to type=""..."" name=""..."" part=""..."" />
  </map>
</coordination>

<notification>
  <map>
    <from type=""..."" name=""order"" attribute = "order_id" />
    <to type=""..."" name = "purchase_order" attribute = "Order_id" />
  </map>
</notification>

As shown in the above syntax, there are two types of mapping which are Coordination and Notification. They are defined as <coordination> and <notification> element respectively. The former is used to map attributes of several shared artifacts to an outgoing coordination message. The latter is used to map an incoming notification message to attributes of shared artifacts. A <map> element describes the details of each data mapping for a particular mapping direction. It consists of a <from> element which defines source of data mapping and a <to> element which destination of data mapping. There are four attributes being used by the elements:

- type: defines type of data that involved in the current mapping which can be local artifact, shared artifact and a payload message contained in a notification message.
- name: defines name of an artifact or a payload message
- attribute: defines name of an attribute of a source or destination artifact
- part: defines name of a part of a payload message

The next element is a <transitions> element that describes the effect of a service invocation on business artifacts. The element consists of one or more <transition> elements as show below:
Each `<transition>` element provides details of the state transition of each artifact being updated by a particular service using the following attributes:

- `shared_artifact`: defines name of a shared artifact involving in a state transition
- `fromState`: defines a current or source state of a shared artifact before a service invocation
- `toState`: defines a next or destination state of a shared artifact after a service invocation

All these elements and attributes constitute shared business rules that control interactions between roles or organization in our executable model. The following is an example of the completed business rule.

```xml
<transitions>
    <transition shared_artifact = "..." fromState = "..." toState = "..." />
    <transition shared_artifact = "..." fromState = "..." toState = "..." />
</transitions>
```
<shared_rule name="rule1">
    <onevent type = "MessageEvent" />
    <pre_con>
        <and>
            <atom type = "attribute" shared_artifact = "purchase_order" attribute_name = "id" operator = "!=" value = "null" />
            <atom type = "attribute" shared_artifact = "purchase_order" attribute_name = "customer" operator = "!=" value = "null" />
            <atom type = "state" shared_artifact = "purchase_order" value = "init"/>
        </and>
    </pre_con>
    <do>
        <invoke role = "Buyer" service = "Order" operation = "createorder">
            <mapping>
                <coordination>
                    <map>
                        <from type = "message" name = "OrderRequest" part = "customernname" />
                        <to type = "message" name = "createOrderRequest" part = "customernname" />
                    </map>
                    <notification>
                        <map>
                            <from type = "local_artifact" name = "order" attribute = "order_id" />
                            <to type = "shared_artifact" name = "purchase_order" attribute = "Order_id" />
                        </map>
                        <notification>
                            <mapping>
                                <transitions>
                                    <transition shared_artifact = "purchase_order" fromState = "init" toState = "pre_created" />
                                </transitions>
                            </mapping>
                        </notification>
                    </notification>
                </coordination>
            </mapping>
        </invoke>
    </do>
</shared_rule>
5.3.3 Role definition

In collaborative business environments, various organizations must be responsible for making change to shared artifacts for certain periods. Each role is defined using the role definition. This definition provides information regarding a role or organization participated in a particular business process. A role is depicted by a <role> element that has a name attribute and a namespace attribute. Moreover, a particular role can provide one or more services to its business partner or customers. Each service provided by a role is represented by a <service> element. The details of a <service> element are defined in the same manner as in Section 3.3.3. We use the Web service details to define each service delivered by a role because a local business process is exposed using a Web service interface in our ACP system. The example of a role definition is shown below:

```
<role name="buyer" namespace="www.swin.edu.au">
   <service name = "createOrder" location = "www.swinburne.com/createOrder"
            operation = "createorder" port="xxx" inputmessage = "createOrderRequest" />
</role>
```

Thus, these components can be used to define a complete contract that will be used by several organizations in a particular collaborative business. The design and implementation of an extension of previously proposed ACP system for supporting the usage of this contract will be described in the next section.

5.4 Design & Implementation of ACP System for Inter-organization

In this section, we discuss how we implement an extension of an ACP system. Firstly, we provide an overview of this extension. Then, we discuss implementation in more details.
5.4.1 Overview of an extension of ACP system

As the model that serves as a contract between each organization previously discussed and defined in Section 5.3, we propose the architecture of an extension of ACP System, as illustrated in Fig. 5-2, to address issues for executing artifact-centric business process in inter-organizational environment. To address the inter-organizational requirements, we consider using centralized controller rather than peer to peer systems since we define business logics using rules and the nature of rules exhibits centralized-control behavior. Moreover, rules can be managed efficiently using a single repository. The concept of the event-driven architecture and service-oriented architecture is adopted in this centralized controller as well as in the previously proposed system. The controller so called Artifact-Centric Coordination System (ACC system) is able to support the process execution across organizations using a set of shared artifacts which serve as a contract and use shared business rules to govern interaction between organizations. The system composes of four components including shared artifact manager, rule engine, role manager and shared artifact data manager.

Fig. 5-2 Architecture of An extended ACP system
Shared Artifact Manager provides management functionality to ensure each contract running in the system is created, managed and updated correctly. The artifact manager basically issues a set of commands to other system’s components to perform operations on a contract or a set of shard artifacts. Rule Engine is to deliver functionality of rule evaluation. The rule engine serves as a central controller to coordinate internal and external operations of each component in the system. Role Manager is to handle a task that is allocated to each role involved in a particular business process. A task or a session allocated to each role/organization is determined by a rule engine that evaluates business data and state stored in a set of shared artifacts. Shared Artifact Data Manager performs a task of updating these shared artifacts. The information or shared business data that is used to make changes shared artifact comes from each participating organization. Thus, Shared business data and state basically reflect the current status of a collaborative business process. These components aim to manage operations on contracts in the centralized controller. However, the information on shared artifacts must be shared from local artifacts managed by a certain role or organization. Thus, we have extended our proposed ACP system by adding shared artifact client manager. Shared Artifact Client Manager is designed to address communication between the central controller and local ACP systems. Its operations are to receive and pass messages issued by the central controller to a local ACP system and also detect status of process execution of the local system and notify the central controller regarding completion of a task or a session of the local system.

Both controller and client work in a synchronized manner to provide and to ensure consistency of the process execution across different organizations and regions. The detail of implementation of the system will be discussed in the next section.

5.4.2 Coordination between local ACP systems and ACC system

In this section, we focus on how to coordinate each participating role involved in an inter-organizational process. To achieve the consistent coordination, we make use of a contract located in artifact-centric coordination system. Moreover, we consider addressing communication between a global process and a local process to support the consistent process execution across organizations.
The inter-organization process gets started when a user submits a request to the artifact-centric coordination system. Once a request message is received, it is directed to a rule engine. The rule engine basically contains a set of shared rules which is a mechanism to govern interaction between roles or organization. Then, the message is evaluated. If precondition of a particular rule is satisfied, a rule is fired to start an inter-organizational process. The purpose of rule evaluation is to determine a corresponding role to undertake the next session of collaborative business process. Firstly, a coordination instance is created by the object factory of ACC system. The class diagram of a coordination instance and other running instances is illustrated in Fig. 5-3.

The Java Architecture for XML Binding (JAXB) is adopted in the implementation of this system as well as in our previously proposed ACP system. Once coordination instance is created, a shared artifact manager assigns an identifier key to the instance and adds the instance to a list of coordination instances. After the coordination instance is completed, the
rule instance is added to a list of rule instances. Next, a shared artifact manager determines whether any shared artifact instances needs to be instantiated or not. If yes, the factory is invoked to produce a corresponding shared artifact instance. The shared artifact instance is, then, assigned a unique key and added to a list of shared artifact instance inside the coordination instance. The next step is that a shared artifact manager sends a command to a role manager that is responsible for calling a participating role to undertake a session of the process execution. A role instance is instantiated and contains information regarding an interface of a service provided by a particular role.

A role manager uses this interface information to generate a service client as well as a payload that is generated from WSDL. The payload is populated with a business data. The process of mapping is governed by a set of mapping rules. These rules are used by a role manager to determine sources of information that is needed to be mapped to a payload. The source of business data can be either shared artifacts or a message. After the payload is completely mapped, a coordination message is created. The message contains details for supporting correlation between a local process and a global process including coordination id, rule id, and role as well as a payload message. The structure of this coordination message is described below:

```
<coordination>
  <coordinationId>...</coordinationId>
  <ruleId>...</ruleId>
  <role>...</role>
    <payload>
      ....
    </payload>
  </role>
</coordination>
```

The coordination message is assigned with correlation data and a payload message. Next, the coordination message is sent to a corresponding role. Once the message arrives at the destination, it passes through the service interface of shared artifact client manager. The
client manager accesses the message and records the correlation information in a list of
correlation key if correlation information does not exist in the list. Otherwise, it retrieves an
associated process id and uses it to get the corresponding process instance. The correlation
information includes coordination id and role. Then, a payload message is extracted and
directed to a rule engine along with a corresponding process instance if correlation key
does exist. The rule engine evaluates inserted data and fires a rule if the condition is
satisfied. Next, the process controller of an ACP system starts a new process execution or
resumes an existing process execution. The operation of ACP system is omitted here since
it already described in the previous chapter.

Once local ACP system completes a particular task, it must notify the artifact-centric
coordination system regarding this completion and sends shared business data. To indicate
the completion of a task, we use states in life cycles of business artifact since state
transition is an outcome of a service or a task performed on a set of artifact. Thus, we use a
sharing rule to support evaluation of state transition of business artifact instances in our
implementation. The structure of sharing rule is shown below. The sharing rule consists of
two main sections including <when> and <share>. The first section indicates a state of an
artifact that need to be satisfied before the client manager can notify the global system. The
latter section is used by the client manager to retrieve shared business data from
corresponding artifacts if state of artifacts in the first section is satisfied. These sections are
contained by a rule element. Finally, all sharing rule are wrapped by a
<client_sharing_rule> element that has process_type attribute. This attribute is used to
indicate a corresponding process that these rules belong to.
After a task/service invocation, artifact instances are checked against a set of sharing rule. If states of the artifact instances match a particular sharing rule, a process controller notifies the shared artifact client manager. Then, the client manager creates a notification message. The structure of notification message is illustrated below:

```xml
<clientsharing_rule process_type=".....">
    <rule no="">
        <when>
            <artifact>
                <name>...</name>
                <state>...</state>
            </artifact>
        </when>
        <share>
            <artifact>
                <name>...<name>
                <attributes>
                    <attribute>...</attribute>
                </attributes>
            </artifact>
        </share>
    </rule>
</clientsharing_rule>
```

```xml
<notification>
    <coordinationId></coordinationId>
    <ruleId></ruleId>
    <artifacts>
        <local_artifact name="...">
            <attribute name="...">
                <value>...</value>
            </attribute>
        </local_artifact>
    </artifacts>
</notification>
```
The correlation information is retrieved from a list of correlation key and a coordination message and assigned to the notification message. Then, the client manager uses \(<share>\) part in a sharing rule to determine which artifact attributes can be shard to the global process. The payload artifacts are generated and populated with attributes of artifacts defined in a particular sharing rule. The notification message is returned to artifact-centric coordination system. Once the message arrives, it is received by a role manager. The role manager determines a corresponding coordination instance using correlation information stored in the message. After the corresponding coordination is located, a shared artifact data manager is responsible for updating information on shared artifacts for both attributes and state. In the process of mapping, the artifact data manager uses mapping rule to ensure that shared artifact attributes are updated correctly. Finally, the shared artifact data manager notifies the completion of updating to the shared artifact manager. The shared artifact manager generates an artifact changed event and passes the event object to a rule engine. At this point, a rule engine may fire a particular rule if pre-condition of a rule is satisfied or it may wait for a request message from a user to start a rule evaluation section. The system repeats this cycle until all shared artifact are in the final state.

5.5 Case study and Evaluation

In this section, we developed a case study to test our extension for executing Artifact-centric business process in the inter-organizational environment. Moreover, we evaluated our approach to support the execution of the artifact-centric process across organizations.

5.5.1 Case study

In this section, the case study is still based on an online ordering process. However, there are three roles or organizations involved in this process. These roles include Seller, Banker, and Shipper. Once a Seller organization receives an order request from a customer, the order is created and recorded. Then, a bill is sent to the customer. The customer makes a payment for his/her order through a Banker organization. After the payment is completed, a Banker informs a Seller. Next, a Seller processes the order and schedule the delivery with a Shipper. Once the order is completed, a shipper delivers the order to the customer’s
doorstep. In this process, each role manages its own local process and local artifact. *Seller* role owns *Order* artifact. *Banker* role owns *Invoice* artifact. *Shipper* role owns *Shipment* artifact. The local processes are deployed in a local system as illustrated in Fig. 5-5. As we can see from the figure, there are three local processes including order process, payment process, and shipping process. To conduct a collaborative business process, they must agree on a contract. Then, the contract is defined using the model proposed in Section 5.3. This contract consists of 3 shared artifacts including *Purchase Order*, *Purchase Payment*, and *Purchase Delivery*. Moreover, a set of shared rules and roles are also defined. After a contract is completely constructed, it is deployed to the coordination system as shown in Fig. 5-4. The deployed process is exposed to a public via a Web service interface. Thus, each client can interact with a running process through this interface. This process starts when user lodge an order through a web site. Then, the order is processed and delivery to a customer.

```
<?xml version="1.0" encoding="UTF-8"?>
<registeredProcess>
  <process><name>purchase</name><location>C:/Users/</location></process>
</registeredProcess>
```

**Fig. 5-4 The Deployed Contract and its Web service interface**

```
<?xml version="1.0" encoding="UTF-8"?>
<registeredProcess>
  <process><name>order_process</name><location>C:/Users/</location></process>
  <process><name>payment_process</name><location>C:/Users/</location></process>
  <process><name>shipping_process</name><location>C:/Users/</location></process>
</registeredProcess>
```

**Fig. 5-5 The Deployed local process that is involved in the collaboration**

A web user interface is provided and is as same as the interface in the previous chapter. After a client submits the information, a message is evaluated by a rule engine. In this case, the first rule is triggered when the message is received by the rule engine. The pre-condition of the first rule is that each part of the message including *name* and *address* of a customer must not be null or empty. The message is validated against this pre-condition. When the pre-condition is satisfied, the rule is fired. The result of rule firing is that the
collaborative process is started and Seller role to be invoked. Purchase order artifact is created. The coordination message is constructed and sent from the coordination system to a local system to invoke a seller. Then, the local system processes a coordination message. The shared artifact client manager verifies the correlation data of the coordination message to check whether there is any existing running process to be continued. Since this is a new inter-organizational process, the new local order process is started and an order artifact is created and updated by a CreateOrderService.

Fig. 5-6 The sharing rules for an order process

As a result of CreateOrderService, the order artifact moves its state from start to open_for_item as illustrated in Fig. 5-8. Then, state transition of order artifact is evaluated using the sharing rule illustrated in Fig. 5-6. Once a state of Order artifact matches Open_For_Item state defined in the sharing rule, the local system generates a notification message and sends it back to the coordination system. The sharing rule also describes which part of Order artifact can be shared. Based on the sharing rule, the orderId and customername attributes of Order artifact are shared to the coordination system through the notification message. Then, the coordination system updates information and state of Purchase order artifact. Both the coordination system and the local system record the states of artifacts before updating and after updating as shown in Fig. 5-7 and Fig. 5-8.
Fig. 5-7 Purchase order artifact before and after invoking a seller

Fig. 5-8 Order artifact at the local system before and after invoke a service
After completing this step of the process execution, the coordination system may either wait for an incoming request or keep continuing the process execution. The both systems keep recording all execution traces from the start to the end. The queries that introduced in Section 4.3 can also be applied to use to extract information from a log file for the monitoring and tracking purposes. From Fig. 5-9, we illustrate the use of Xquery to monitor the process execution. A stakeholder wants to know the current state of the running process. The coordination is provided as input of the query. Thus, the result of the query is that the process is in \textit{ready\_to\_process\_order\_item} state.

![XQuery Example](image)

\textbf{Fig. 5-9 Use of XQuery in the inter-organizational process}

From the test case, our extended system is not only able to execute business process across organizations but also ensures that the process execution is executed consistently using coordination message and notification message.
5.5.2 Evaluation

Based on an outcome of the case study, we are able to execute the artifact-centric business process in inter-organizational environment. In this section, we provide the evaluation of our prototype system in some interesting points including interoperability, monitoring and tracking, and flexibility.

− Interoperability

Interoperation of workflows is a fundamental requirement for each organization since each company may need to conduct business processes beyond its boundary. Currently, two conventional approaches to address this requirement are orchestration and choreography [31]. Orchestration address interoperation by developing an application or process that serves as a centralized controller to govern various workflows and services, whereas choreography aims to address interoperation requirement while preserving the autonomy of the stakeholders. This choreography approach controls how message can be passed between the stakeholder workflows or services. In our realization approach, we considered orchestration to be more appropriate approach to address the use of business artifacts in the multi-organizational process. The main reason is that our artifact-centric process model uses a set of business rules to govern operations of services over business artifacts. Business rules exhibit centralized behaviors since it is often used as a central controller to govern business logics of the processes and the management of business rules is more efficient in a single repository. Thus, the controller system named artifact-centric coordination system (a.k.a ACC system) is design and developed to govern operations of participating local ACP systems using a set of business rules. The benefit of orchestration is that a single rendezvous point where stakeholders can find current status and information of the process is provided. However, it reduces autonomy of the stakeholders.

− Monitoring and tracking

In the artifact-centric approach, the process models are constructed using the key business entities called business artifacts. Each business artifact contains business-relevant data and its life cycle. Artifact data and life cycle of each artifact reflect progress of a particular process toward a business goal. In the inter-organizational business, each
organization manages its own local artifacts and the progress of a local process can be tracked using local artifacts. However, the progress of a local process is only visible to a particular organization. Thus, there is no way for the other participants in this collaboration to understand the current status or progress of the collaborative process. In this implementation, shared artifact is used to address monitoring and tracking requirements across organizations. Shared artifacts have a set of shared information that is shared from local artifacts. They also have their own life cycles which are used to represent the current stage of corresponding local artifacts in a restrictive manner. From the case study, we clearly illustrated that we are able to share information and state from local artifacts to shared artifact. Moreover, we are able to extract monitoring and tracking information from shared artifacts. Thus, not only shared artifacts can be used as an agreement between organizations but also they can be used to support monitoring and tracking across organizations since they can capture sharable data that passed across organization and their life cycles can describe the overall progress of a business process.

- **Flexibility and changes management**

  Flexibility is one of advantages of the artifact-centric process model since the model is defined in the declarative model. In this artifact-centric approach, we are able to easily adapt business processes in response to changes in business environment. This advantage of the artifact model has been proved in many researches [1,3,5,24]. In our realization approach, the artifact-centric process model for inter-organizational business is realized using the direct approach. Therefore, flexibility in the conceptual model is inherited to the executable model. Flexibility is well supported for design-time and run-time for the artifact model for inter-organizational business as well. Moreover, the SOA concept is adopted in our systems. Thus, we gain the benefit of loose coupling. This allows local ACP systems to be able to change implementation of their local processes if the changes do not affect the contract managed by artifact-centric coordination system.
5.6 Summary

In this chapter, we investigated the challenges of executing the artifact-centric process across organizations. To address these challenges, we adopted the concept of the artifact model for inter-organization business process that was proposed in [21]. The key components of this model include shared artifact, business rules, services and roles. Based on these components from the conceptual model, we defined the artifact-centric executable model for inter-organizational business processes. Based on this executable model, we analyzed the requirements of a prototype for supporting the execution of business processes across organization. The requirements are that a system must coordinate operations of workflow engines existing in different organizations and must be able to maintain workflow execution data during run-time. Moreover, it needs to be able to capture workflow data distributed among organizations. Then, we developed another prototype system so called Artifact Centric Coordination System (a.k.a. ACC system).

The system consists of four components. Shared Artifact Manager provides management functionality to ensure each contract running in the system is created, managed and updated correctly. Rule Engine is to deliver functionality of rule evaluation. Role Manager is to handle a task or a session of a business process is allocated and undertaken by each role. Shared Artifact Data Manager performs a task of updating these shared artifacts. Moreover, we observed that the information on shared artifacts must be shared from local artifacts managed by a certain role or organization. Thus, we have extended our proposed ACP system by adding shared artifact client manager. Shared Artifact Client Manager is designed to address communication between the central controller and the local ACP systems. Then, we discussed how the system coordinates operations of local ACP systems located in the different organizations.

Lastly, we used a test case to test our extended system and the result was that the system was able to execute and manage a business process across organizations correctly. Moreover, information monitoring and tracking can be queried from a log file during the process execution.
CHAPTER 6:
CONCLUSION AND FUTURE WORKS

6.1 SUMMARY OF THIS THESIS

Business processes are series of activities performed by various stakeholders and is important for enterprises to ensure growth and competitive advantages against their rivals. Business processes originate from enterprises’ business strategies which mainly focus on delivering values to their customer without involving their customers to cost and risks. To communicate business goals and perform business process reengineering, business process modeling is a significant activity for representing a business process in a comprehensive format. Moreover, business process model can also be used as a driver for implementing complete workflows. Currently, most existing standards for business process modeling and workflow are based on activity-centric approach. However, this approach cannot provide complete unified information of business processes since data is incorporated at a limited level.

To address this issue, a new paradigm so called artifact-centric approach has been emerging. It provides many benefits to business process modeling areas. The approach focuses on business artifacts as key entities for modeling business processes. The components of the artifact-centric model comprise business artifact (information model and life cycle), business rules and services.

The purpose of this thesis is to investigate the challenges to develop a new framework for realizing artifact-centric process model. The framework not only supports the process execution in one organization but also can be extended to support the process execution across organizations. From our investigation, one of the remaining challenges in the artifact-centric area is workflow realization of artifact-centric model. Currently, there is an existing approach for realizing artifact-centric process models. This approach is based on the conversion of artifact-centric model into a procedural model. The advantage of this realization approach is ease of implementation since the procedural model of artifact-centric business processes can be mapped and implemented using the current workflow technologies. However, there are some drawbacks such as flexibility, data loss.
Thus, we considered realizing the artifact-centric model without transforming the model. In this way, we are able to maintain business logic and flexibility of the operational model. Since logic of the operational model can mapped faithfully into an executable version of artifact-centric model, we do not need to perform conformance checking whereas the former approach must perform this activity since business logic can be mapped incorrect during the process of a model conversion. Moreover, monitoring and tracking of business process can be achieved directly since business artifacts can describe both informational and behavioural aspects of business process. Thus, this is more appropriate approach for realizing an artifact-centric business process model.

To support the direct realization approach, we developed the executable model of artifact-centric business process. Firstly, the logical details of the artifact-centric process model which based on [8,9] were introduced. The core components of the model include artifacts, business rules and services. Then, the design methodology for the artifact-centric process model was discussed. The methodology consists of four main steps: (1) business artifact discovery, (2) business operation modeling, (3) conceptual workflow design, and (4) workflow realization. Finally, we introduced an executable version of the ACP model. The executable model is defined using XML and consists of three definitions: artifact definition, business rule definition, and service definition. It contains implementation details required by a system to execute a particular business process and they are used for creating running instances.

Then, we presented the system so called Artifact-Centric Process System. First, we identified the problems and analyzed the system requirement. Then, the system architecture was designed. The concept of the event driven architecture and service oriented architecture is adopted in this architecture. The system consists of five core components including Business Rule Engine, Process Controller, Web service Controller, Artifact Controller and Front-end UI Interface. The components can provide various operations for supporting execution of artifact-centric business process such as rule evaluation, artifact updating, service invocation etc. Moreover, we discussed a technique for querying the process information from a log file using XQuery to support tracking and monitoring. Based on this design, we developed a prototype system and tested it with a case study of online ordering process. As a result, the system was able to execute the artifact-centric
model. Moreover, the monitoring and tracking of the running process is achieved using XQuery. We were able to query the required information from a process log file to answer the questions regarding monitoring and tracking of a running process.

Furthermore, we extended the system to support realization of artifact centric process model across organizations as we observed that we are able to exchange attributes of the artifact used in an organization’s internal process between participating organizations using a message. Moreover, we are able to specify when we can exchange/share these attributes since Business artifacts have their life cycles and we use states in the artifact life cycles to specify the point that we have to exchange information. First of all, we analyzed the requirements for inter-organization process. According to the requirements, we defined an artifact-centric executable model for inter-organizational business. The model consists of shared artifact definition, shared rule definition, and role definition.

Based on the executable model, we proposed a system so called Artifact-Centric Coordination System (a.k.a. ACC system). The system is composed of four components including Shared Artifact Manage, Rule Engine, Role Manager, and Shared Artifact Data Manager. These components work together to ensure the consistency of the process execution across organizations. Moreover, we extended our proposed ACP system by adding Shared Artifact Client Manager. The client manager acts as a controller at the client side. The communication between the local artifact-centric process systems and the artifact-centric coordination system is addressed by coordination message and notification message. These messages are sent between the role manager and the shared artifact client manager for supporting operations on local ACP systems and the artifact-centric coordination system. During the process execution, the correlation information is maintained in local systems and the global system to ensure consistency of the process execution. Based on this design, a prototype system was developed and tested using a case study of online ordering process. The outcome was that the system was able to execute the artifact-centric model across organizations. Furthermore, the monitoring and tracking of the running process was achieved by using the same XQuery technique to inspect a set of shared artifacts.
6.2 **Future works**

In the future, further investigation will be carried out to refine the framework for realizing artifact-centric business process model. Future research includes an extension to support modeling of artifact-centric business process, an extension to run-time dynamic modification, and so on.

- **Extension to support modeling of artifact-centric business process**
  Many research papers have claimed that business artifacts can be easily identified in the first step of the design process. However, the process designer may face some difficulties during the design process since the artifact-centric process model is not an explicit model. The implicit model is difficult for the designer and stakeholder to comprehend the logics of business processes. Thus, our future work aims to include a tool to support the artifact-centric process modeling for both local processes and inter-organizational processes. This tool will help visualizing the implicit model in the comprehensive way. Moreover, this tool will provide verification and conformance checking functionality to ensure that processes are free from errors, e.g. deadlocks, and able to fulfill business goals.

- **Extension to support run-time modification of business rules**
  Flexibility is one of the advantages of the artifact-centric business process model. Our system utilizes a rule engine to support decision making of business logics in a business process. The rule engine provides a feature for adding or removing business rules at any time. However, the changes of business rules affect sequence of a process execution. In our future work, we aim to provide mechanism to verify that a running process can reach its business goals after the rules changed.
BIBLIOGRAPHY


