การออกแบบและพัฒนาแบบจำลองและเครื่องมือประสานบริการเชิงปริภูมิ

พันตรี สรวิศ สุภเวชย์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต
สาขาวิชาวิศวกรรมสำรวจ   ภาควิชาวิศวกรรมสำรวจ
คณะวิศวกรรมศาสตร์   จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2554
ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

THE DESIGN AND DEVELOPMENT OF GEO-SPATIAL SERVICES ORCHESTRATION MODEL AND ENGINE

MAJOR SORAVIS SUPAVETCH

A Dissertation Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy Program in Geomatic Engineering

Department of Survey Engineering

Faculty of Engineering

Chulalongkorn University

Academic year 2011

Thesis Title	THE DESIGN AND DEVELOPMENT OF GEO-SPATIAL SERVICES ORCHESTRATION MODEL AND ENGINE
By	Major Soravis Supavetch
Field of Study	Geomatic Engineering
Thesis Advisor	Assistant Professor Sanphet Chunithipaisan, Ph.D.

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctoral Degree

พันตรี สรวิศ สุภเวชย์ : การออกแบบและพัฒนาแบบจำลองและเครื่องมือประสานบริการเชิงปริภูมิ. (THE DESIGN AND DEVELOPMENT OF GEO-SPATIAL SERVICES ORCHESTRATION MODEL AND ENGINE)
อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร.สรรเพชญ ชื้อนิธิไพศาล, 101 หน้า.

ปัญหาสำคัญของการรวบรวมบริการเชิงปริภูมิข้ามแพลตฟอร์ม ก็คือ ความหลากหลายของแพลตฟอร์มที่ถูกนำมาใช้จัดทำบริการ ซึ่งเป็นได้ทั้ง W3C SOAP เว็บเซอร์วิส OGC เว็บเซอร์วิส และ REST เว็บเซอร์วิส บริการเหล่านี้มีข้อแตกต่างกันตรงที่ส่วนติดต่อใช้งานของแต่ละแพลตฟอร์มไม่เหมือนกัน ซึ่งทำให้เกิดข้อจำกัดในการประยุกต์ใช้เทคโนโลยีที่นิยมใช้กันอย่างแพร่หลาย ดังเช่น BPEL ในการประสานบริการเว็บเซอร์วิสข้ามแพลตฟอร์มเหล่านั้น

วิทยานิพนธ์นี้ได้กำหนดแบบจำลองที่สำคัญขึ้น 3 แบบจำลอง คือ Heterogeneous geo-spatial services coordination model, Heterogeneous-payloads manipulation model และ Heterogeneous-exception handling model สำหรับใช้เป็นแนวทางในการสร้างภาษาการประสานบริการขึ้นมาใหม่ ให้สามารถนิยามการประสานบริการข้ามแพลตฟอร์มได้ นอกจากนี้เครื่องแปลภาษา หรือเครื่องสั่งทำงาน ได้ถูกพัฒนาขึ้นเพื่อใช้ทดสอบภาษาที่พัฒนาขึ้นอีกด้วย

แบบทดสอบ 3 แบบได้ถูกนิยามขึ้นเพื่อใช้ทดสอบภาษา คือ Site selection Coordinate transformation และ Network analysis แบบทดสอบทั้งสามแบบนี้ใช้ทดสอบภาษาการประสานบริการที่พัฒนาขึ้นใหม่ (สำหรับทดสอบแบบจำลองที่กำหนดขึ้นใหม่ทั้ง 3 แบบ) ผลการทดสอบแสดงให้เห็นความสำเร็จในการประสานบริการข้ามแพลตฟอร์ม และได้พิสูจน์แนวคิดของแบบจำลองทั้งสามว่าเป็นองค์ประกอบที่สำคัญจริงที่จะทำให้เกิดความสามารถในการประสานบริการเว็บเซอร์วิสเชิงปริภูมิข้ามแพลตฟอร์มได้

ภาควิชา วิศวกรรมสำรวจ ลายมือชื่อนิสิต...
สาขาวิชา วิศวกรรมสำรวจ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก....................................
ปีการศึกษา 2554 ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์ร่วม....................................

4971877721 : MAJOR GEOMATICS ENGINEERING
KEYWORDS : GEO-SPATIAL WEB SERVICE, WEB PROCESSING SERVICE, WEB SERVICES ORCHESTRATION, ORCHESTRATION LANGUAGE, WORKFLOW ENACMENT SERVICE

SORAVIS SUPAVETCH : THE DESIGN AND DEVELOPMENT OF GEO-SPATIAL SERVICES ORCHESTRATION MODEL AND ENGINE. ADVISOR : SANPHET CHUNITHIPAISAN, Ph.D., 101 pp.

The important problem of the integration of cross-organizational geo-spatial services is the various implementations of a service platform. The platforms can be W3C SOAP web service, OGC web services, and REST web service. Services which have different interfaces are the constraint that limits the use of the most popular orchestration technology such as BPEL for assembling those heterogeneous services.

In this thesis, the three mandatory models (i.e., the heterogeneous geo-spatial services coordination model, the heterogeneous-payloads manipulation model, and the heterogeneous-exception handling model) for a new language implementation which provide a cross-platform orchestration are identified and addressed. The language interpreter/executor so called orchestration engine is implemented for the testing and demonstration of a language.

The three workflow scenarios (i.e., site selection, coordinate transformation, and network analysis) are defined for a language testing. The heterogeneous on geo-spatial services coordination, payloads manipulation, and exception handling are tested. The result of each scenario demonstrated the successful of a cross-platforms orchestration, and proofed the three proposed models are the essential elements of a cross-platforms ability.

Department : Survey Engineering	Student's Signature
Field of Study : Geomatic Engineering	Advisor's Signature
Academic Year : 2011	

ACKNOWLEDGEMENTS

This research would not have been possible without the support of many people. I am deeply grateful to all of them and I want to thank some of them in particular.

I thank Assistant Professor Sanphet Chunithipaisan for being my supervisor. He gave valuable insight into research techniques and supported my research at anytime. I also express my sincere thanks to every lecturer in Department of Survey Engineering. They supported my research in various ways from the very beginning.

CONTENTS

page

LIST OF FIGURES

LIST OF ABBREVIATIONS

BPEL	Business Process Execution Language
GIS	Geographic Information System
GML	Geographic Markup Language
HTTP	Hyper Text Transfer Protocol
ISO	International Organization for Standardization
KVP	Key-Value Pair
OGC	Open Geospatial Consortium
REST	Representational State Transfer
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
UTM	Universal Transverse Mercator
W3C	World Wide Web Consortium
WFS	Web Feature Service
WGS	World Geodetic Coordinate System
WPS	Web Processing Service
WPS-T	Transactional Web Processing Service
WSDL	Web Service Description Language
XML	Extensible Markup Language
XPath	XML Path Language
XSLT	Extensible Stylesheet Language Transformations

Chapter 1
Introduction

The research covers the topic of Geo-spatial Services Orchestration. The services orchestration involves the integration approach of standardized geo-spatial web services for solving and finding answers of spatial-related problems by implementing web services technologies.

This chapter is structured as the background (Section 1.1), topic overview (Section 1.2), motivation (Section 1.3), objectives (Section 1.4), scope of the thesis (Section 1.5), research questions (Section 1.6), methodology (Section 1.7), and chapter overview (Section 1.8) respectively.

1.1 Background

Web service is designed to enable users to access one or more capabilities, whose access is provided by using a prescribed interface, constraints and policies as specified in a service description. Many organizations today try to develop their own web services responding to their authorities, and to cooperate with another web service which is provided by another agency. The access and cooperation knowledge between web services are required. For this reason, a number of researches on web service cooperation are established not only in academic but also in commercial software industry. Many efforts are trying to assembling web services by using workflow definition. Those web services assembling are called Web Services Composition, Web Services Chaining, Web Services Integration, and Web Services Orchestration (WSO). The latest technology such as web services orchestration is the abstract of services cooperation in message level including business logic and cooperation orders (Peltz, 2003).

In 2001, IBM introduced WSFL (Web Services Flow Language) which was XML-based language for describing web services integration which is a modeling business process. Microsoft also introduced its own language called XLANG which was XML-based language for describing web services integration. In 2003, IBM and Microsoft jointly developed BPEL (Business Process Execution Language) and used common technologies such as WSDL to describe service communication and SOAP to envelop exchanging message.

In geo-spatial web service community, geo-spatial web services integration currently does not have a standardized language to define a process sequence. Many efforts were recorded through a number of researches such as (Alameh, 2003) introduced "Chaining Geographic Information Web Service" with three components (i.e., Data Service, Processing Service, and Registry / Catalog Service). The chaining involves an assembling of geo-spatial web services which the client is a coordinator. Chen et al., (2005) extended "Web Service Chaining" to automatic concept by applying BPEL as the backend orchestration engine and called that a "Services Composition". The coordination between the services participant in the research was performed in a peer-to-peer architecture. In 2007, OGC released Web Processing Service (WPS) and claimed that WPS process can be designed to call a sequence of web services including another WPS process (acting as the service chaining engine) (OGC, 2008). In addition, Stollberg and Zipf (2008) proposed Composite-WPS, whose WPS was used without an orchestration language implementation for controlling a process flow between sub-services in a sequence for achieving a processing result.

A number of researches mentioned above are varying in scientific domain but use the same technology such as workflow-managed orchestration. The usefulness of using workflow-managed orchestration is to reduce time and cost of system development. But the widely accepted technology for web services orchestration such as BPEL was invented without including GI related concepts, then the use of BPEL for geo-spatial services orchestration is limited due to the incompatibilities between standard W3C and OGC web services such as described in Ioup et al., (2008). Some of those limitations can be described as follows:

- The standard specifications of OGC web services lack support in WSDL and SOAP which are used in BPEL.
- The web services which comply with W3C use certain XML structure for encoding request and response. Therefore, all services are compatible in

term of protocol or XML encoding structure. Whereas, each type of OGC web services has its own protocol in both request and response form.

- The WSDL document defines the list of function definitions, input parameters, and output types; but does not provide a geo-spatial metadata which is necessary in a client application.

Therefore, constrain on the integration of heterogeneous geo-spatial services is a protocol (or interface). Not only on W3C/OASIS and OGC web service protocol but also protocol adaptation such as WPS 0.4.0 adapted to WPS 1.0.0. Then the interface independent orchestration language should be designed and implemented.

1.2 Topic Overview

The Geo-spatial Services Orchestration topic is about an integration approach. The approach on each service-coordination is abstracted into the definition language with containing the logical-control of services flow. A number of services are involved despite of having a single process flow. Each service can be different to the others in operation names, function types or interface implementation. The integration of those different services can be realized in a number of approaches depending on technologies it's used.

The geo-spatial services orchestration in this research involve three sub-topics including orchestration language, orchestration engine, and orchestration service. Each topic can be described as follows:

Orchestration Language is a specialized language which is used to define how the orchestration engine should coordinate the entities (sub-services) participating in geo-processing workflow.

Orchestration Engine is an execution mechanism in which orchestration script is executed. The service participants are coordinated by the engine in a sequence that defined in orchestration script. The orchestration engine acts as interpreter and deals only with controls and data flow.

Orchestration Service is a mechanism which is allowing accessibility to the backend orchestration engine. In this method users can perform web services orchestration in the two ways; first, generic users can execute a pre-defined orchestration script as a single operation. Secondly, expert users which are able to translate the scientific workflow into an orchestration script are allowed to insert their scripts into the service repository for execution later or for use by other users.

The methodology using above three components to orchestrate geo-spatial services is called a semi-automatic orchestration. In a semi-automatic orchestration, an orchestration script is written manually by users. Users who wrote a script need to determine the availability of the services by their own. For example, users have to inspect (i.e., policy, contract, and availability) each web service which is defined by understanding its WSDL document then documenting SOAP for communication and checking the service availability.

1.3 Motivation

A geo-spatial services orchestration is a key topic that challenging the concrete implementation on several geo-spatial services specifications. The services orchestration is essential in a geo-spatial world because it can be used as a tool to extract a new aspect of geo-spatial knowledge which is not pre-defined in multidisciplinary domains from existing online geo-spatial resources. For this reason, numbers of researches have recently implemented the integration, composition, chaining, and the orchestration of geo-spatial services.

The problem of incompatible between different services interface (a message encoding format) obstructs the integration of an open geo-spatial web services due to a number of organizations and platforms are involved. This problem is a long standing challenge in geo-spatial web services and is still unsolved for providing a robust solution for an interface independent orchestration over multiple platforms.

The research identifies three essential aspects (i.e., service coordination, message manipulation, and exception handling) for interface-independent orchestration. The motivation of the research can be described in more details as follows:

1) *Service coordination* is one of the key constrains among a rapid development in web service technologies. In a business world, the web service always a particular purpose associated with interacting with a service as in "real world effect" (trying to get the service to do something). A W3C web service is a popular framework for developing such the service. In a geo-spatial world, geo-science users use a standard web service from OGC instead of W3C web service for a development. The differences between these two standards are about describing a service and interacting with that service. Ongoing demands of consumers in a mass market are pushing the development of the web technology into a new approach every day. In addition, a selection on technologies for service implementation is based on various conditions that are not a technical term such as organization's objectives and demands. The glue and robust solution for combining those platforms dependent is needed.

2) *Message Manipulation* is another important key of a message exchange especially in multiple message platforms. The geo-spatial message such as GML has a spatial related meaning that requires special technique to determine and manipulate the message (i.e., insert, delete, copy, filter, and transform).

3) *Exception handling* is another important mechanism for unsuccessful processes interruption. The exception handling can also be used for producing a service compensation mechanism. In web services, a different services platform produces different exceptions i.e., HTTP exceptions, SOAP exceptions, and OGC web service exceptions. To manage and handle these various forms of exceptions in a multiple services orchestration, the mechanism which is able to access and catch the multiple exception-formats is required.

1.4 Objectives

The objective of this research is to

Design and develop an orchestration language and engine facilitating geo-spatial services orchestration over distributed heterogeneous geo-information and geo-services that implemented in a different platform such as W3C web service, OGC web service and REST web service.

More specifically, the research aims at:

1) Developing a platform-independent geo-spatial services orchestration.
2) Designing a new orchestration language allowing user to define an orchestration script across the different services interface and platform.
3) Developing an orchestration engine by implementing a geo-spatial enactment service specification.
4) Developing a prototype of a geo-spatial enactment service.

1.5 Scope of the Thesis

Based on the research objectives this thesis focuses specifically on the following issues:

1) OGC web services tested in this thesis are Web Feature Service and Web Processing Service.
2) W3C web service technologies that involve an interaction, coordination, and communication patterns are investigated in a technical level of SOAP and WSDL application for providing geo-spatial web services.
3) WPS-T (Transactional Web Processing Service) framework which is extended from OGC WPS is used to produce a workflow enactment service prototype in order to testing a geo-spatial services orchestration language.
4) Focuses only on a synchronous communication pattern.

1.6 Research Questions

Based-on the research objectives (Section 1.4), the research question can be extracted as following question:

“How a geo-spatial services orchestration language and orchestration engine support geo-processing workflows which enable loosely coupled to a number of service platforms, interface versions, and various context types of payloads such as GML and so on?”

The above question can be divided into three sub-specific questions as follows:

1) What is necessity for an interface-independent of geo-spatial services orchestration aspect?

2) What are mandatory mechanisms of a geo-spatial services orchestration that realize a process definition (orchestration language) and facilitate a geo-processing flow?

3) How a designed language supports a binary data orchestration such as GML files?

1.7 Methodology

The thesis addresses two aspects which concentrate on design and implementation of a geo-spatial services orchestration. According to research motivation (Section 1.3) and for answering research questions (Section 1.6), the works of the thesis can be divided into three phases as follows:

1) Designing phase: the OGC web services interfaces are examined for inspecting what is the mandatory elements that are necessary for facilitating a services orchestration. BPEL's conceptual model is inherited for designing a new orchestration language which supports an interface-independent geo-spatial services orchestration. The standard UML model is used to draw language components, and a structure of the language is described in an XML-schema.

The relevant researches in an area of geo-spatial web services are reviewed for designing scenarios for testing the developed language and execution engine.

2) Implementation phase: the engine is written in a JAVA programming language that deployed in the Apache Tomcat. The relevant tools and libraries for the implementation such as GeoTools, JTS suite, and Geo API are selected and used to develop an orchestration engine. The OGC WPS-T platform is used as a standard service interface in a development for providine an orchestration service to users.

3) Testing phase: sub-services such as WFS, WPS, REST and SOAP are developed in the research to fulfill the research scenarios. The research scenarios are selected from related geo-spatial processing researches.

1.8 Chapter Overview

Chapter 2 reviews an existing technology of the W3C web services orchestration such as BPEL (Business Process Execution Language). Aims and benefits of BPEL on performing orchestration in a business world are explained. Then, the application of using BPEL technology on OGC web services is described. Finally, relevant researches on using BPEL with OGC web services are introduced.

Chapter 3, more specific constraints on using BPEL to orchestrate heterogeneous geo-spatial services are addressed for designing a new orchestration model in two sections; 1) existing constraints on applying BPEL with OGC web services and REST web service are explained, and 2) suggested models for solving BPEL constraints on multi-platforms orchestration are demonstrated.

Chapter 4, the new orchestration language designed from suggested models in Chapter 3 is displayed in term of an XML schema. Examples of an orchestration sub-process definition are demonstrated in each part of the language component.

Chapter 5 is a language testing. The designed scenarios are explained and executed through the orchestration service. The result of the processing is also shown the successful of a services orchestration.

Chapter 6, the research issues are addressed and summarized. The answers of research questions are explained. The contributions of the research are identified, and a future work is also described.

Finally, the XML-schema of the geo-spatial services orchestration language is shown in appendix A. All testing scripts are shown in appendix B - D.

Chapter 2
Literature Reviews

2.1 Technologies Overview

Geo-spatial information plays an important role at all levels of scientific disciplines. (OGC, 2008) noted that “Geography is a foundation property for modeling the world in a coherent, intuitive way that location and time can be exploited as a unifying theme to better understand the context of most real and abstract phenomena”. The geo-spatial information which is extracted from the knowledge of the Earth can be used in many purposes e.g., hydrological analysis, wildfire prediction, air pollution monitoring, water and land resource management, agricultural production estimation, environmental management, disaster management, and so on. Scientists use those geo-spatial-data to understand what is needed to utilize the data and associated information in the on-going scientific research.

Nowadays, numerous geo-spatial data are produced and provided through the Web by public organizations and authorities at various levels of detail (local, regional, national, and global) (Kiehle, 2006). The examples of those publication services are, JPL (Jet Propulsion Laboratory) Global Imagery Service, Microsoft TerraServer Map Server, Pacific and Yukon Region WFS for water quality monitoring stations, Geo-spatial Web Services from GeoBrain (Di, 2004), INSPIRE Geo-portal for environment information, OK-GIS for disaster management, and so on. Facility of the Internet is a key to increase and chain the way of business, then companies are moving their main operations to the Web for more automation, efficient business processes and global visibility (Tsalgatidou and Pilioura, 2002). The effect of this also gives scientific knowledge to be worldwide interdisciplinary as well.

Beyond geo-spatial services, the key success of these technologies is the interoperability because “no single organization produces all the data (so it’s inconsistent) and no single vendor provides all the systems” (OGC, 2008). The essential of this interoperability is usually appeared though a number of researches today in term of “services composition”, “services chaining”, “services orchestration”, and “services integration". Ability of compositing more complex processing tasks is the greatest value of geo-spatial web services (Einspanier et al., 2003).

The different types of geo-spatial services are published using different interfaces. To integrate those services, it requires the mediate component to manage those heterogeneous services for achieving a geo-processing solution. For examples, WFS (Web Feature Service) itself cannot coordinate with WMS (Web Map Service), WCS (Web Coverage Service) itself also cannot coordinate to WFS; therefore, the third entity (mediator) such as client-coordinated application/service, aggregate service, or workflow-managed service (Alameh, 2003), is necessary.

Several protocols, profiles, specifications, and standards are developed to service geo-spatial information through Internet technologies. Open Geo-spatial Consortium (OGC) and ISO/TC 211, the international organization for standardization, have jointly released a number of specifications for interoperable geo-spaital information services and message encodings. Those standard specifications allow developer to build reliable and sustainable geo-spatial services supporting geo-spatial users in multiple disciplines (Granell, Diaz and Gould, 2010). The standardized geo-spatial web services (so called OGC web services) are defined using open non-proprietary Internet standards in particular the WWW standards of HTTP, Uniform Resource Locators (URLs), Multipurpose Internet Mail Extensions (MIME) types and the Extensible Markup Language (XML) (OGC, 2001).

OGC WPS and WCPS, a standardized geo-processing service for vector based and raster based, are designed to define an interface that facilitates the publishing of geo-spatial processes to clients. WPS or WCPS processes can be incorporated into service chains and those services are designed to call a sequence of web services, thus acting as the service chaining engine (OGC, 2008).

Moreover, OGC have acknowledged on using BPEL (Business Process Execution Language) which is an OASIS standard executable language for generating recommendations and guidelines for geo-spatial services orchestration (OGC, 2009).

BPEL aim is to enable “programming in the large” which programmed by larger or smaller groups of people over the long time periods and coding managers place emphasis on partitioning work into modules, possibly written by different people with precisely-specified interactions (DeRemer and Kron, 1975). BPEL describes the automated arrangement, coordination, and management of web services and focuses on the collection of related, structured activities or tasks that produce a specific service (OASIS, 2007). BPEL mainly implements on XML technology e.g., WSDL, XML Schema, XPath, XSLT and XML Information Set. Using BPEL for orchestrating OGC web services is proprietary to those W3C web service standards (Gone and Schade, 2008). For example, OWS services which are orchestrated by BPEL engine must describe service interface through WSDL and provide message exchange through SOAP protocol.

BPEL rely on SOAP and WSDL, while some public and open geo-spatial web services do not. The differences between OGC and W3C web services are addressed and described in (Ioup et al., 2008) For this reason, geo-spatial services orchestration by using BPEL invalid in case of particular services in a chain do not support SOAP and WSDL. In addition, some geo-spatial services such as WMS and WCS respond a geo-spatial message in a byte-stream via HTTP transport protocol.

Many researches such as (Weiser and Zipf, 2007), (Fleuren and Muller, 2008), and (Sancho-Jiménez et al., 2008) proposed the mediator (e.g., Proxy OGC web service) as a message exchanging service to associate those different interfaces into homogenous workflow. That proxy service provides a way to integrate OGC web services into the BPEL process.

2.2 Interoperability Reference Model

Geographic interoperability is defined as the ability of information systems that run software cooperatively over the network for manipulating such spatial information about the objects and phenomena on, above, and below the Earth’s surface (ISO19119, 2001). The ability of interoperability is defined as about to access, integrate, analyze, and present geo-spatial data across a distributed computing environment by machines and personal devices such as cell phones, PDAs, and smart phones (Lee and Percivall, 2008).

The interoperability concept is typically implemented using web service. A loosely-coupled in distributed interaction (interoperability) between web services is designed to easier to develop interoperability among independent web services development. Benatallah et al., (2005) explains that even though web services using SOAP, web services may also provide in the different of operations, parameters, and business protocols (different constrains).

Interoperability among geo-spatial web services is defined in (ISO19119, 2001) as the production of services chaining. The geo-spatial services chaining can be classified into three types as follows:

User defined (transparent) chaining. The user acts as mediator with manually composes and executes the geo-spatial services.

Workflow-Managed (translucent) chaining. Sequence geo-spatial services are executed and controlled by a workflow service that acts as a mediator.

Aggregate Service (opaque-chaining). The sequence geo-spatial services appear to the user as a single service. The aggregate service acts as the mediator that services integration performs in the back-end.

The essential of above three models become more necessary in current geo-spatial web service technologies because:

The large heterogeneity of open geo-spatial services needs integration technique to produce science solution.

More and more complex geo-spatial problems can be solved by an interoperability of a number of geo-spatial services.

A growing number of users in geo-sciences disciplines drive heterogeneous geo-spatial services into a capability of integration.

2.3 Interface Standards

"OGC has been dedicated to the standardization of the interfaces of geo-spatial web services to enable interoperability" (Zhao, Yu and Di, 2007). In W3C web service, structure of information which sending between web services depend on business domain and its implementation. SOAP which is infrastructure of W3C web service provides only the way for defining what and how information get sent, but geo-spatial data also requires more specific structure of encoding message to enable interoperability between independent geo-spatial web services and client applications. Thus, OGC decided to design geo-spatial web service interface to common model and share some unified characteristics to every types of geo-spatial services. The same type of OWS service is implemented and published via the same interface; for examples, WMS provide GetCapabilities, GetFeatureInfo, and GetMap operations with unified parameters for servicing map image, WFS provides GetCapabilities, GetFeature, DescribeFeatureType operations for destibuted geo-spatial features, WCS provides GetCapabilities, GetCoverage, DescribeCoverage operations for publishing geo-referenced image.

Additionally, Amirian, (2006) notes that “Geo-spatial Web services and Web Services differ in a way that Web services are composed of particular set of technologies and protocols but Geo-spatial Web services are comprised of defined set of interface implementation specifications which can be implemented with diverse technologies” (cited in Amirian and Alesheikh, 2008, p. 731)

2.4 Standard Technologies for Geo-spatial Services Orchestration

Due to the growing of orchestration software tools and technologies convinces geo-spatial web service interface to SOAP and WSDL implementation, geo-spatial web services are re-produced for supporting that orchestration tools. The constraints and ability for enabling OGC web services orchestration are focused and elaborated for a long time by OGC since OWS-2 testbed (OGC, 2004). After that, OGC proposed to use BPEL hidden behind the new release specification such vendor specific WfCS (Workflow Chaining Service) interface as WPS (OGC, 2009). OGC web services that appear in

BPEL script must be adapted to service through SOAP and WSDL ((OGC, 2003), (OGC, 2005), and (OGC, 2008)). Thus, a more complicated geo-spatial model and process flow can be defined into a service chain such as BPEL for complex geo-spatial applications and knowledge discovery (Zhao, Yu and Di, 2007). For more discussion and detail about orchestration software tools and technologies are founded in the research of Peltz, (2003).

In addition, WPS is extended to support the deployment/undeployment for any algorithm/workflow called Transactional WPS (WPS-T). Then BPEL profile can be deployed for execution as a single WPS process. This is no standardized way to package the workflow model to be exchanged in GI community, but WPS-T which is the research work of Schaeffer, (2008) (Schaeffer, 2008)is a starting point for standardizing way to deploy geo-processing model to WPS. Even more, an executable JAR files or other XML-based workflow descriptions can be deployed to WPS-T as well.

2.5 Wrapper or Mediator Service

In open OGC web services which is platform-neutral require a mediator service for wrapping service interface to facilitate the OGC web services orchestration by BPEL. The deep description of that implementation is found in the research of (Sancho-Jiménez et al., 2008). The mediator (or wrapper) service should acts as a proxy from SOAP interface to OGC web service interface and vice versa. The aim of that mediator is a bridging gap between OGC web service services and W3C/OASIS services for using SOAP and WSDL without code modification. In additional point of view, the mediator services or communication services which described in Technology viewpoint (a basis for cross platform interoperability) are hardware and software components in distributed system which is responsible for routing service request from client object to server object (ISO19119, 2001). The inter-operations between those components are also needed to achieve interoperability.

2.6 Orchestration Service Use Cases

In a large enterprise application which involves a number of processes and actors, application developer may pay attention to coding with rules of business combination over existing communication technologies. For example in online banking, the bank customers are allowed to conduct financial transactions (i.e., bill payments, funds transfers, investment purchase, and loan transactions) on a secure web service in which operated by their retail. The business process of this online banking involves several actors in which invoked at appropriate time and appropriate data. If this process logic is implemented in glue code (such as execution script) in which executable in appropriate service, the changing of this business process is straightforward by changing only glue code.

The vital component of these services assembling is an orchestration language. Orchestration language is a set of rules for user to define a sequence of tasking order responding to a pre-designed workflow in which a number of services are coordinated with a control-flow. To reaching benefits of SOA principal, the request and execution of orchestration script should be carried out through a service as well.

For clarify a relation between participants and services orchestration patterns, Figure 1 is used to demonstrate the relationship of user-provider components in physical world. For examples; (1) application developer create end-user application which uses multiple services (client controls orchestration), (2) user uses application which multiple services (including orchestration service) are integrated behind single service (opaque chaining/orchestration), (3) service/data provider can creates each standard service and also be able to create an orchestration script in case of assembling the multiple existing services is required.

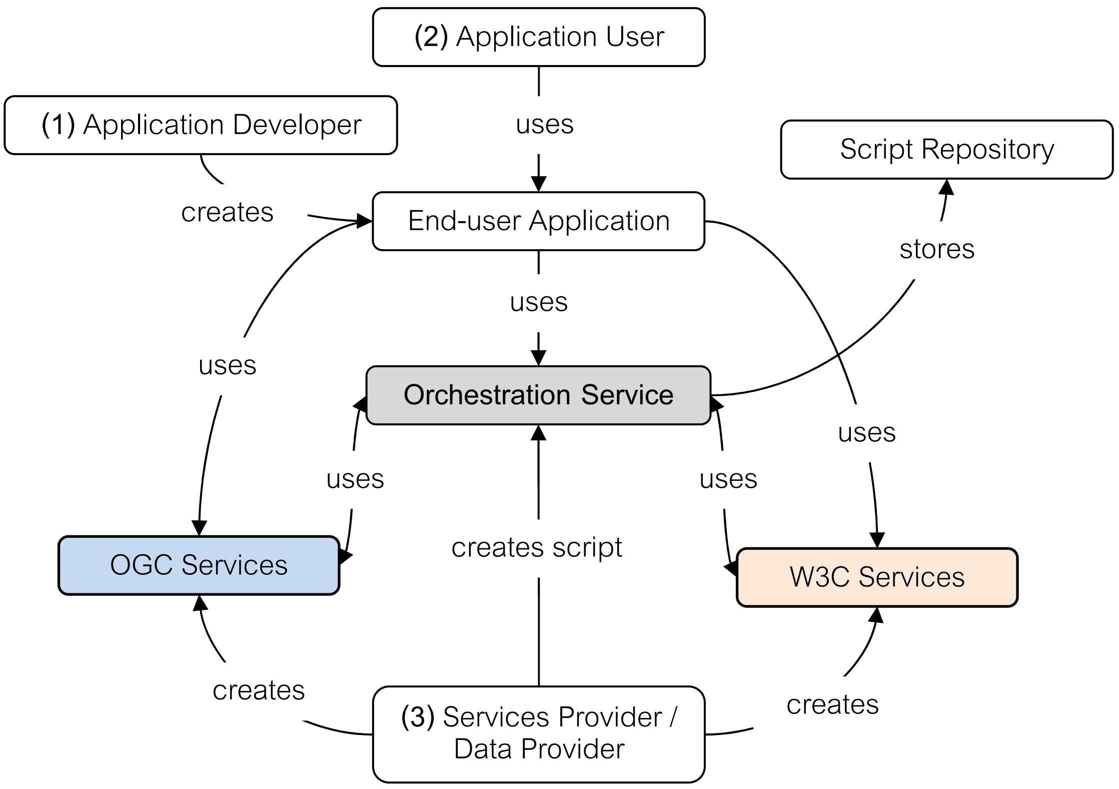


Figure 1. Use case model of Orchestration Service.

2.7 Business Process Execution Language (BPEL)

BPEL is a definition language and a standard specification of web service orchestration from IBM, Microsoft, and BEA (Peltz, 2003); it has an XML-based grammar for defining logical control on web services orchestration. Logical control so called business protocol is a model that involves the behavior of participants in a specific business interaction. BPEL defines message exchange between business parties through standard web service specifications such as SOAP and WSDL. Additionally, BPEL language can be interpreted and executed by orchestration engine software. Thus, BPEL script can be re-executed many times.

Business interactions in BPEL called activities that activities can be abstracted into <receive>, <reply>, and <invoke> messages. The example of activities that involves in BPEL process flow is demonstrated in Figure 2. In contrast, business flow is a structure of activities which are specified in order of what should be run in an appropriate time.

Each activity communicates with web service via SOAP protocol and understands communication contact by interpreting WSDL document.

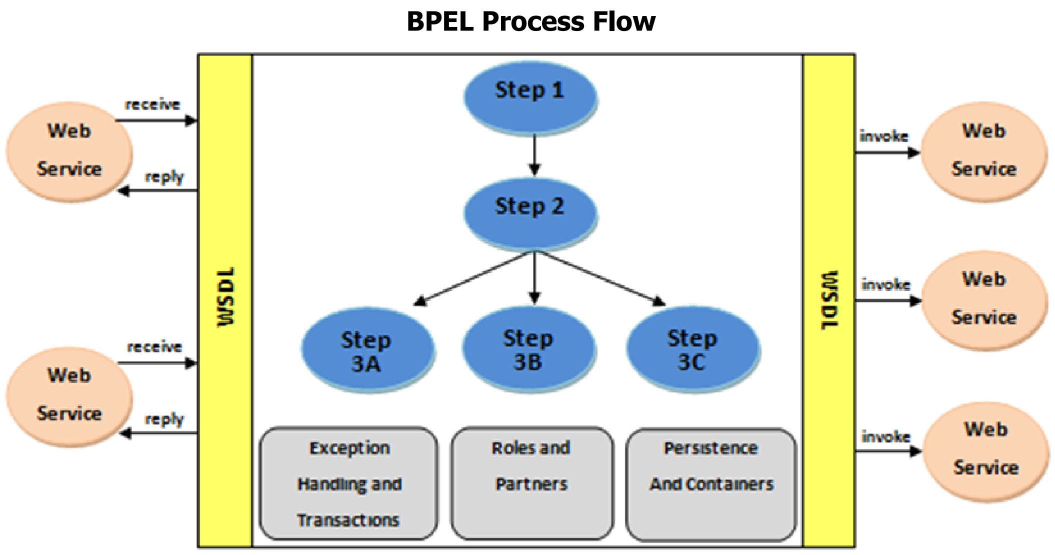


Figure 2. BPEL Process Flow and its business activities (Peltz, 2003).

2.8 BPEL Constraints on OGC Web Services.

BPEL is popularly used in business world that allowable to model the behavior of web services in business process interaction. BPEL is best fit in business world, but using BPEL with native OGC web services is invalid due to lack support of SOAP and WSDL. WSDL describes the programmatic interface of W3C web service which client will know what it does, where it is located and how it is invoked for performing service interaction. Business domains are varies, thus consistent XML of those do not exist. While geo-spatial web service such as OGC web service, the interaction interface is standardized, and client has already acknowledged of what it does and how it is invoked.

Additional assumption, SOAP is a standard protocol that only used as an envelope of business information, that means business encoding message does not have a formalized structure. Whereas, geo-spatial message is standardized and formalized but varying in format such as GML, GeoJSON, Shapefile, and GeoTIFF due to reserving the essential of services interoperability.

In conclusion, BPEL convinces service provider to provide duplicate endpoints of SOAP and WSDL to OGC web service. The details of the differences of OGC and W3C standard specification interface was described in (Ioup et al., 2008), and still mentioned in a number of implementation researches (e.g., (Stollberg and Zipf, 2007), (Fleuren and Muller, 2008), (Stollberg and Zipf, 2008), (Sancho-Jiménez et al., 2008), and (Meng, Bian and Xie, 4-5 July 2009)).

Chapter 3
Design of Conceptual Model

This chapter presents the conceptual model of an interface-independent orchestration language for implementing the geo-spatial services orchestration. The model is used as a foundation for the language implementation and testing in Chapter 4. In this research, the conceptual model is used to demonstrate an idea of orchestration especially on the heterogeneous geo-spatial services.

The geo-spatial services (W3C, OGC, and REST web services) are the web services which are different in communication patterns, messages, contracts, and also protocols its used (e.g., SOAP, WFS, WMS, WCS, WPS, etc.). BPEL, the standard web service orchestration language, is insufficient to use as a definition language to orchestrate those different services. The constraints of BPEL for defining services orchestration on the other standard services such as REST and OGC web services are explained in Section 3.1. Therefore, the suggested models for defining a process of a services orchestration on the heterogeneous geo-spatial services are introduced and also demonstrated in Section 3.2.

3.1 BPEL Constraints on Multi-platform Geo-spatial Services Orchestration

3.1.1 The Constraint on the Heterogeneous Geo-spatial Services Coordination

Service coordination is an important key for achieving services interoperability, due to it defines information such as what is a protocol used to a communication and what is a message used to exchange.

In the OGC vision about a service discovery and interoperability, the knowledge about how the service is implemented is less important than what kind of interfaces it supports (Bai, Di and Wei, 2009). The interface in this meaning is a name set of operations which support identifying the service behaviors in an information-processing of data, information, or knowledge such as OGC Web Feature Service, OGC Web Map Service, OGC Web Processing Service and so on.

In addition, the two levels of protocol such as transportation protocol and application protocol are used to make service coordination. In the transport protocol, HTTP is used only as a transportation of information and/or functionality to other applications in a context of HTTP GET/POST operation. In application protocol, the operations, functions and service contract are encoded in an XML document for interpreting and understanding by clients.

Both W3C and OGC web services use HTTP protocol to transport a communication message. Information about client-service interaction is encoded in the message context with an XML form. This communication pattern is called "document style". But OGC web services provide more flexible communication pattern to end-users by providing a mechanism to embed application information via HTTP URL such as HTTP GET/KVPs (Ioup et al., 2008).

Moreover, recently proposed architecture such as ROA (Resource Oriented Architecture) has introduced: HTTP operations are used as an application protocol (contain special meaning to application) for providing Get, Read, Remove and Update activities. Even though all OGC web services define a communication pattern based on the Remote Procedure Call (RPC) by using HTTP GET and POST with KVP as same as a weak REST compliant level, this OGC communication pattern is not equivalent to service operations as used in REST/RESTful protocol (Lucchi, Millot and Elfers, 2008).

Figure 3 and Figure 4 demonstrate the examples of using HTTP by OGC, W3C, and RESTful web service respectively.

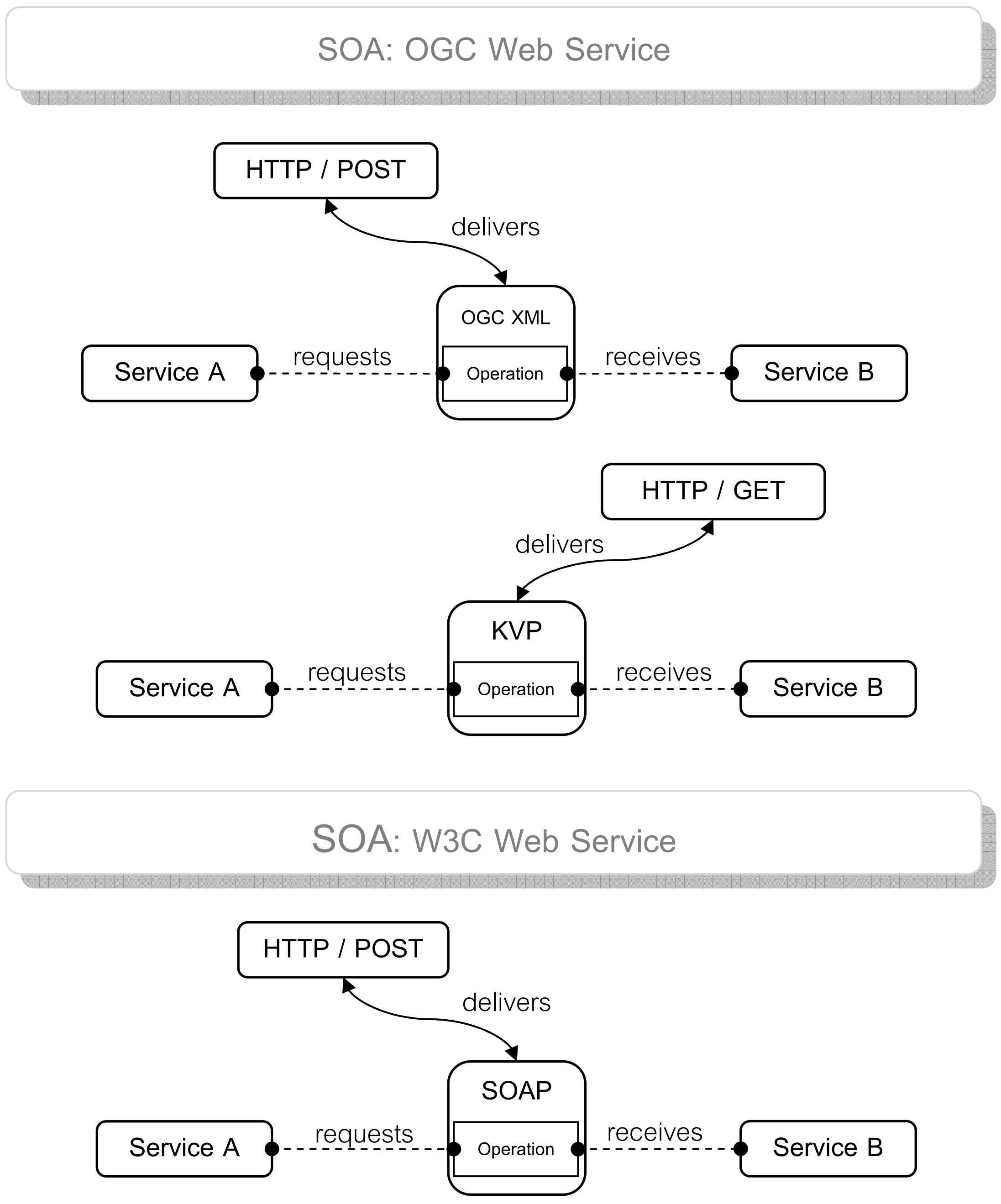


Figure 3 Examples of using HTTP operation as transport potocol for exchanging a message between OGC and W3C web service (Tsalgatidou and Pilioura, 2002).

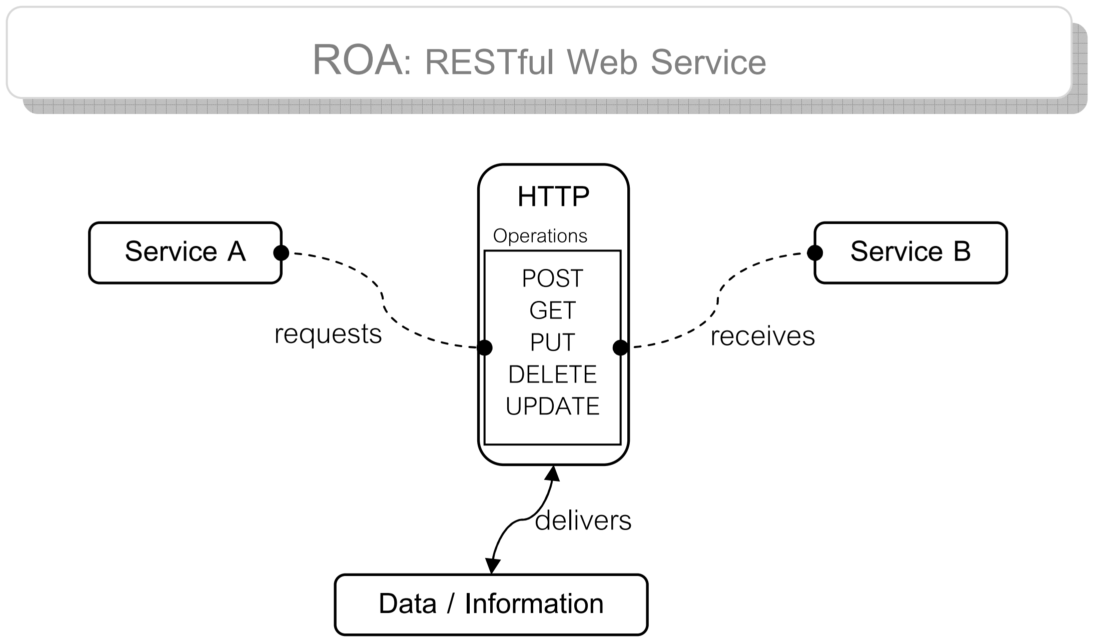


Figure 4 Example of using HTTP operations as a service's application protocol in RESTful web services (Lucchi, Millot and Elfers, 2008).

BPEL (version 2.0) process is layered on the top of the service model defined by WSDL 1.1 (Pautasso, 2009). From this reason, BPEL is tight coupling to WSDL 1.1 language that does not support REST and also OGC web services as well.

3.1.2 The Constraint on The Heterogeneous-Payload Manipulation

The orchestration engine should deals only with a control-flow and a data-flow for which it used to orchestrate the business processes from various application domains (Manolescu, 2004). That means an orchestration engine focuses on assumption of how to flow various data types through web services without understanding how the data encoded. But on the platform such as OGC web services how data encoded is importance for service clients. Understanding on message (or context) encoding will provides ability to transform information from one service to another in an orchestration engine.

In services interoperability, a service context (or payload) is about information to be transferred among two or more parties that compatible with each other by both agreed services contract in synchronous communication. Service context can be encoded to describe a service binding/capabilities (e.g., WSDL, wfs:Capabilities, wms:Capabilities), to send executable message, and also to exchange information/data between communicated services. For example, in Message Oriented Model: entity that interact with others using messaging (Booth et al., 2004), geo-spatial information services can originate/process various messages that depend on what is activity type and what is a participation role which is interacting to another (Figure 5).

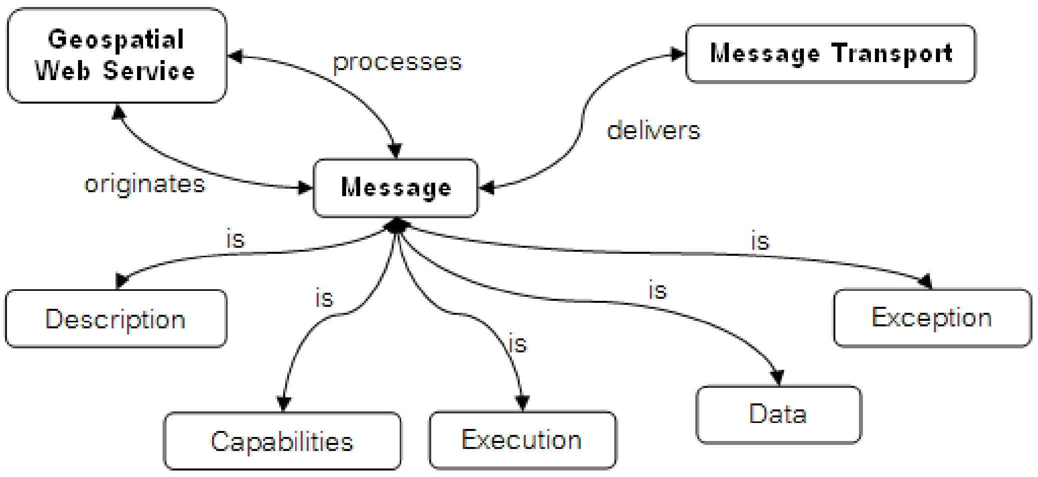


Figure 5 Geo-spatial messages in Message Oriented Model.

From various types of geo-spatial messages as demonstrated in Figure 5, orchestration engine needs programming capability for accessing a responding message to extract, copy, transform the responding message into a request message as input parameter. Especially for XML-based message, technologies such as XPath, XSLT, DOM Parser, SAX Parser, StAX, and XQuery can be used to access XML nodes for manipulating information.

The capability of XML queries in orchestration engine is important for performing business communication between each pair of services. The success of each communication pair affects the success of geo-spatial services orchestration. Geo-spatial message which is sending for processing contains location and spatial relationship (e.g., near, far, adjacent, inside, and so on). XML technologies such as XPath, XSLT, or XQuery are insufficient for accessing to those kinds of relationship. In this case, Filter Encoding (FE) technology should be applied to the orchestration language.

Thus, constraint on message (payload) manipulation in BPEL is lack support on various types of geo-spatial messages such as GML or others.

3.1.3 The Constraint on the Heterogeneous-Exception Handling

The aims of fault handling in BPEL are to undo partial or unsuccessful work of a scope in which a fault has occurred. The occurring of fault due to some service actor throws an exception or error as XML encoded message when service provider found that process was unsuccessfully task.

When orchestration was performed among different service platform, the challenge of interpretation is found. For example, in W3C web service, SOAP 1.1 provides <faultcode> to define classified faults according to one of several local names, and <faultstring> to define a human-readable explanation of the fault. SOAP 1.2 provides different syntax using <Code> and <Reason> respectively. Vice versa, OGC web service, provides <Exception exceptionCode="...." location="..."> and <ExceptionText> as child element in <Exception> element. The difference of syntax among W3C and OGC web service standard are demonstrated in Figure 6 (a, b, c):

```
<xs:element name="Fault" type="tns:Fault"/>
<xs:complexType name="Fault" final="extension">
   <xs:annotation>
      <xs:documentation>Fault reporting structure
      </xs:documentation>
   </xs:annotation>
   <xs:sequence>
      <xs:element name="faultcode" type="xs:QName"/>
      <xs:element name="faultstring" type="xs:string"/>
      <xs:element name="faultactor" type="xs:anyURI" minOccurs="0"/>
      <xs:element name="detail" type="tns:detail" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
```

(a)

```
<xs:element name="Fault" type="tns:Fault"/>
<xs:complexType name="Fault" final="extension">
```

```
   <xs:annotation>
      <xs:documentation>Fault reporting structure
      </xs:documentation>
   </xs:annotation>
   <xs:sequence>
      <xs:element name="Code" type="tns:faultcode"/>
      <xs:element name="Reason" type="tns:faultreason"/>
      <xs:element name="Node" type="xs:anyURI" minOccurs="0"/>
      <xs:element name="Role" type="xs:anyURI" minOccurs="0"/>
      <xs:element name="Detail" type="tns:detail" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
```

(b)

```
<element name="ExceptionReport">
<complexType>
   <sequence>
      <element ref="ows:Exception" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="version" use="required">
      <simpleType>
         <restriction base="string">
            <pattern value="\d+\.\d?\d\.\d?\d"/>
         </restriction>
      </simpleType>
   </attribute>
   <attribute ref="xml:lang" use="optional"/>
</complextype>
</element>

<element name="Exception" type="ows:ExceptionType"/>
<complexType name="ExceptionType">
   ...
   <sequence>
      <element name="ExceptionText" type="string"
               minOccurs="0" maxOccurs="unbounded" />
   </sequence>
   <attribute name="exceptionCode" type="string" use="required" />
   <attribute name="locator" type="string" use="optional" />
</complexType>
```

(c)

Figure 6 (a) SOAP Fault 1.1, (b) SOAP Fault 1.2, and (c) OWS ExceptionReport.

In ROA such as REST and RESTful provides fault in different way. That REST-based web services do not have a well-defined convention for returning error messages. The faults of REST-based web service can be responded as: HTTP Error status (such as 4xx and 5xx codes), HTTP Header message, return nothing (return null string), or XML-encoded document (error was encoded in XML pattern).

In conclusion, constraint on heterogeneous-exception handling in BPEL is lacking activity which can be used to define an access to exception code with various types of geo-spatial messages.

3.2 Suggested Model for Interface-independent Geo-spatial Services Orchestration

3.2.1 Heterogeneous Geo-spatial Services Coordination Model

From limitation on service coordination in BPEL language which depend on platform specific, the suggested model for eliminating that constraint is designed. For enabling orchestration engine to handle both XML-based message and File-based formats, HTTP structure must be accessed by expression in pre-defined orchestration script for supporting file-based data handling.

The proposed mechanism for service coordination, which supports a communication on OGC, W3C and REST web service platforms, is demonstrated in Figure 7.

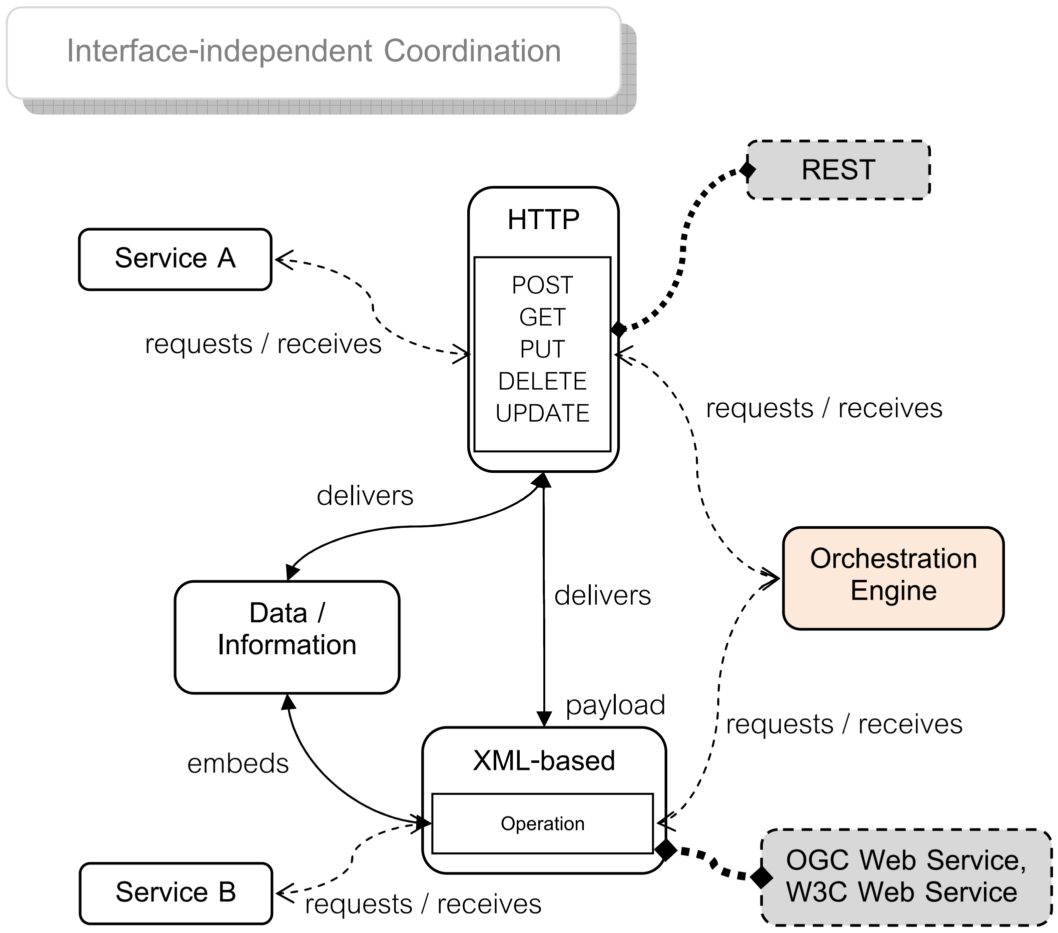


Figure 7 Suggested model which is demonstrated a robust mechanism to provide interface-independent coordination capability via web-based services.

In Figure 7, HTTP transportation control can be used as coordination control on all W3C, OGC and REST web service communications. The transportation control can handles all payloads. Payload such as coverage (GeoTIFF), image file (image/png), and proprietary file-based (Shapefile) can be handled by engine as temporary file then they can sending to next process in order of orchestration. Payload such as XML-based document can be handled and also interpreted by engine in order of semantic and syntactic data flow. Thus, HTTP transportation control (mechanism) is a solution which is suggested to use as coordination control over standard web service platforms such as W3C, OGC and REST web service.

3.2.2 Heterogeneous-Payloads Manipulation Model

In mass-market payloads, geo-spatial information exchange among web services can be classified into three types: SOAP (and generic XML-based), GML (i.e., GML2, GML3, and others), and File-based (i.e., Shapefile, GeoTIFF, and others). In case of XML-based message, XML technologies such as XPath and XSLT can be used to access and transform document into another form for being parameters for next process. In addition, standard spatial technology such as Filter Encoding can be applied and used to filter features from spatial data such as GML as well. The activities such as <Assign/> should be used to select, copy, insert, update, and delete specific XML node that for transferring information to next service invocation parameter.

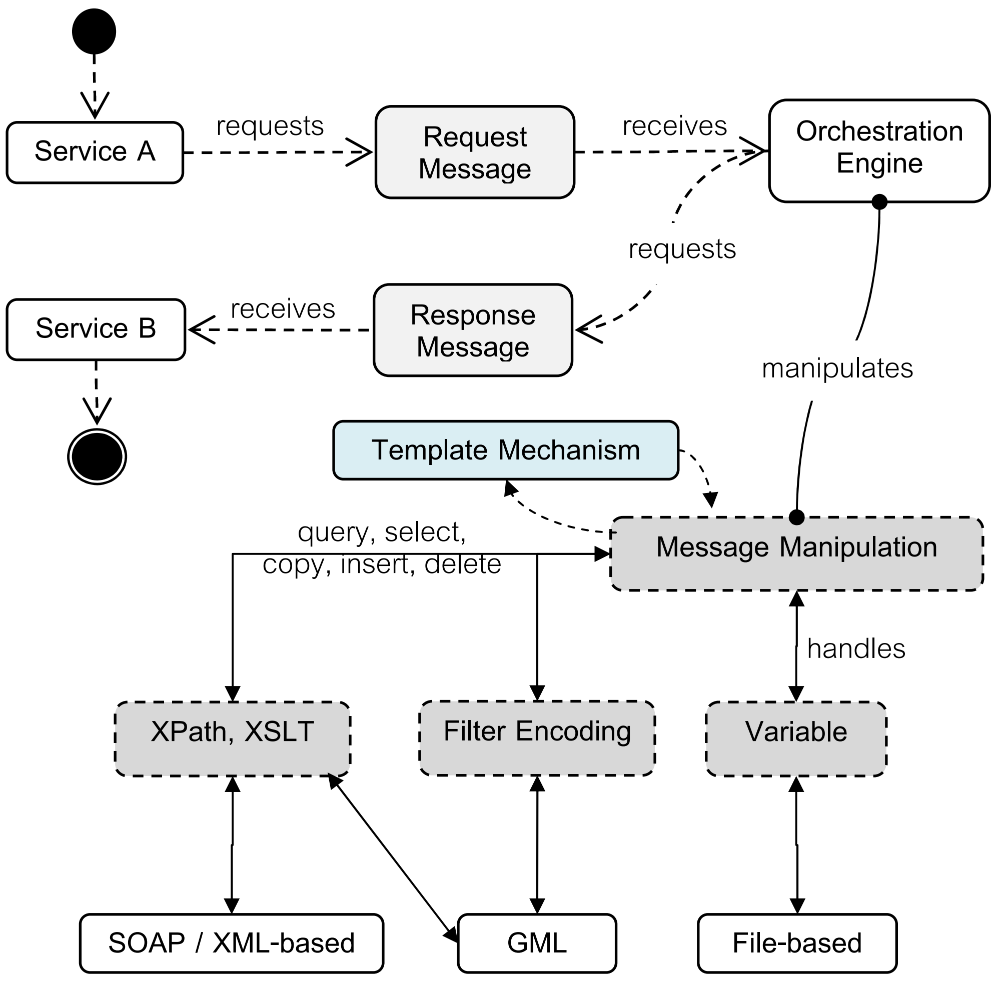


Figure 8 Message Manipulation Model

The suggested model demonstrated in Figure 8 provides additional mechanism for assigning XML document to each interface standard such as SOAP or wfs:GetFeature. The designed orchestration language did not based on some specific interface-standard, and then the methodology to define specific interface-standard is required. In this case, "user-defined template" mechanism is suggested to provide ability for composing an interface-specific document. Thus, user who writing a pre-defined script or third party designing tool should be aware of the validity of that embedded.

Additionally, in case of supporting adaptive of the standard interface (e.g., WFS 1.0.0, WFS 1.1.0, etc), the template mechanism is also used to solve the difference of interface versioning.

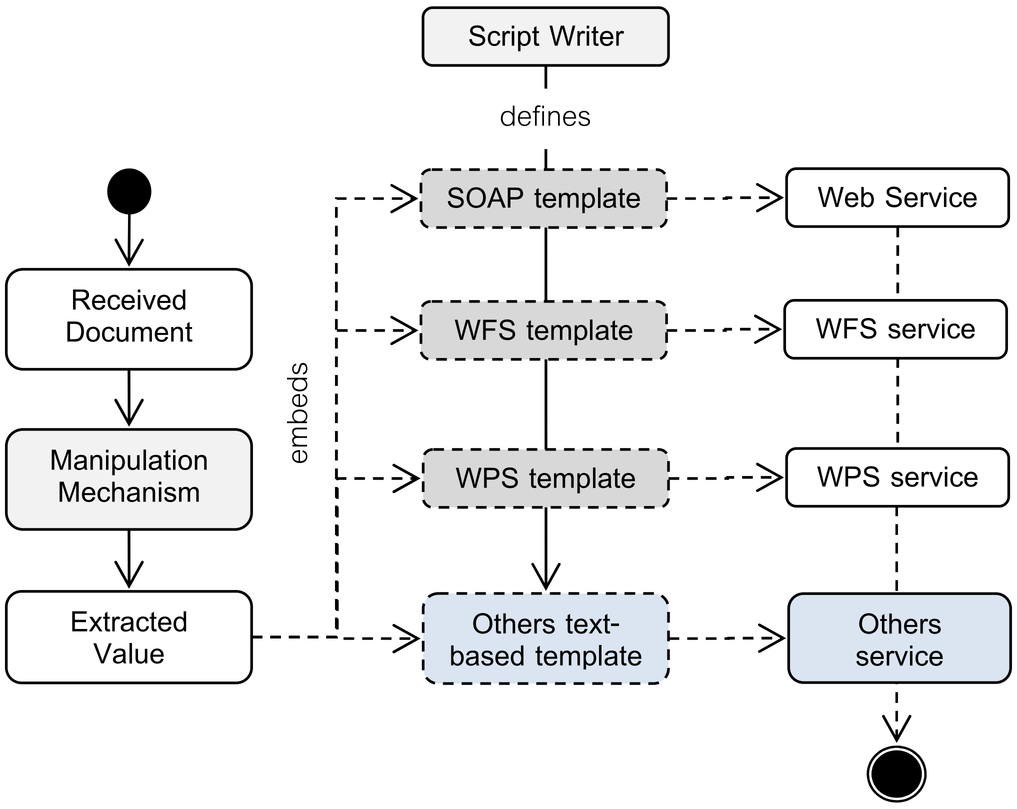


Figure 9 Template mechanism for composing interface-specific coordination.

In Figure 9, runtime feature document such as wfs:FeatureCollection or GML can be embedded to template message by string replacement. For example in Figure 10, template document such as <wps:Execute/> remarks feature value as parameter name ({$hospitalFeatures} and {$hospitalDistance}). Features value that responding from previous process can be simply embedded to template document by replacing a parameter name.

```
<wps:Execute>
…
<wps:DataInputs>
   <wps:Input>
      <ows:Identifier>features</ows:Identifier>
      <wps:Data>
         <wps:ComplexData mimeType="text/xml; subtype=wfs-collection/1.0">
            <![CDATA[{$hospitalFeatures}]]>
         </wps:ComplexData>
      </wps:Data>
   </wps:Input>
   <wps:Input>
      <ows:Identifier>buffer</ows:Identifier>
      <wps:Data>
         <wps:LiteralData>{$hospitalDistance}</wps:LiteralData>
      </wps:Data>
   </wps:Input>
</wps:DataInputs>
…
</wps:Execute>
```

Figure 10 Example of WPS interface document template

In conclusion, payloads which are features of previous processing result can be transforming to next request document (e.g., SOAP or such OGC standard documents as WPS and so on) by using template mechanism. Thus, geo-spatial data can flow from one standard to another standard by this proposed mechanism.

3.2.3 Heterogeneous-Exception Handling Model

There are two common ways to express exception message. Firstly, the exception is expressed through HTTP protocol. Secondly, XML is used for reporting the exception.

Suggested model for managing those exceptions is separated into two components. First is the exception which occurred in transportation protocol which uses HTTP status to define the type of error. Second is the exception which is encoded in XML as HTTP POST body (payload). Orchestration language should provide both accessible capabilities to catch exception. In XML-based exception encoding, XPath can be used as expression to access the hierarchy of document node that exception such as node value can be captured. By using XPath technology, exception caching can be realized on message-based platform independent.

Chapter 4
Design of Geo-spatial Services Orchestration Language

This chapter describes an orchestration language regarding suggested model described in Chapter 3. Firstly in Section 4.1,the language is explained in abstract model. The essential components of geo-spatial services orchestration language are shown. Secondly in Section 4.2, XML-schemas are presented for explaining an ability of language which can be used to define geo-processing workflow. Finally, the implementation of orchestration engine is shown in Section 4.3.

4.1 The Design of Abstract Model

This abstract model is developed to address the research problems (section 1.6). This thesis focuses on orchestration language which platforms-independent can be assembled and encoded into an orchestration script. Structure of language consists of three based components: Variables, Sequence, and FaultHandlers as demonstrated in Figure 11. The Variables is used to define all process variables which occurred during the execution process. The Sequence is used to define activities that will be executed in an ordered sequence. The FaultHandlers is used to define try-catch block for determining an activities when exception occurring during calculation process.

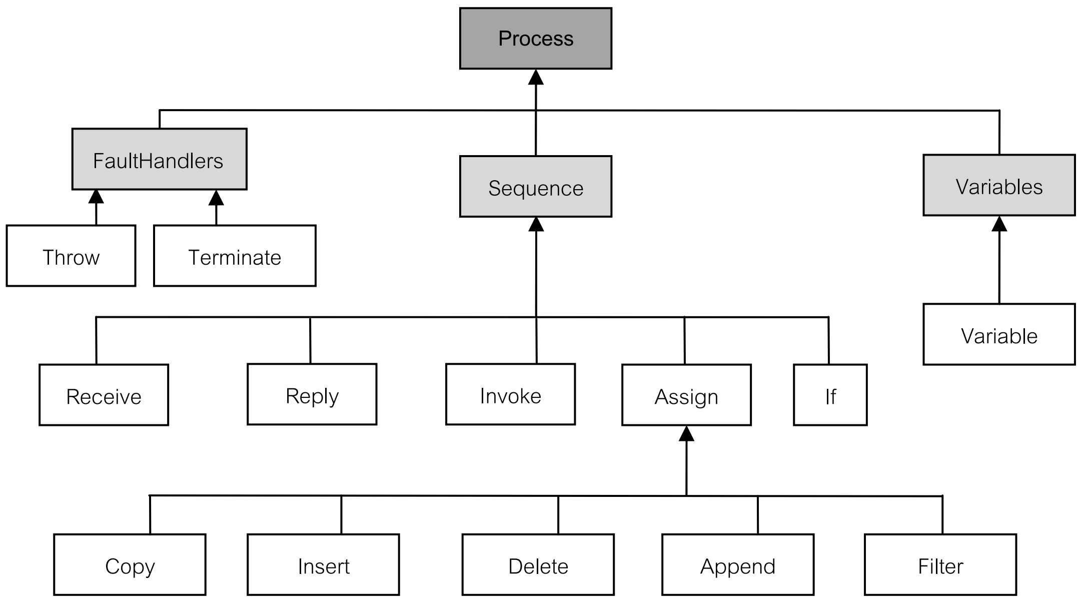


Figure 11 Abstract model of process definition

The remains of this section are demonstrating example models responding to topics which are used to explain language capability. First, file-based data handling and manipulation is shown to demonstrate how language defines a control on payload such as binary file. Second, XML-based data handling and manipulation is displayed for explaining how to define a control on message exchange procedure. Finally, example of fault-caching is demonstrated in order to describe an independent-exceptions structure caching.

4.1.1 File-based Data Handling and Manipulation

Variable is used to declare the handling method to be used to handle geo-spatial messages (i.e., Figure 5 Geo-spatial messages in Message Oriented Model.) for exchanging information between participants. Geo-spatial message types are ranging from XML document to proprietary file-based such as Shape file or Image file. Example in Figure 12 demonstrates the sample of definition that describes the work of binary file handling. The stereotypes (the standardized and simplified conceptions of groups based on some prior assumptions) which are placed above a class such as <<Service Coordination>>, <<HTTP Access>>, or <<Data Handling>> are the name of conceptual models which described in Chapter 3.

In Figure 12, GeoTIFF file which transport via HTTP body will be captured by <<HTTP Access>> function then <<Data Handling>> is expressed to be used to store GeoTIFF file to a service variable for next communication.

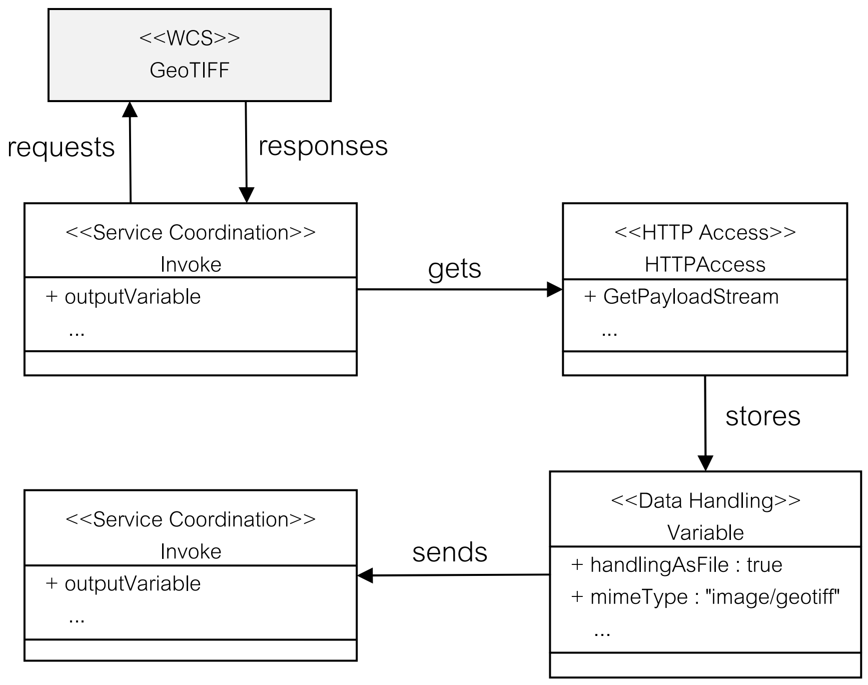


Figure 12 Abstract model demonstrates the related activities with work-flow of geo-spatial image handling.

4.1.2 XML-based Data Handling and Manipulation

Any XML-based message published via web services can be transformed to another format by message manipulation mechanism which allows user to define copy, insert, delete, append, and filer activities to transform specific information to the next request. The query language such as XPath is required for being used as information extraction tool from XML structure. The template message mechanism is designed as an extension of orchestration language on which document such as WFS:GetFeature, WPS:Execute, and so on can be embedded and used in orchestration definition. The example of class and associated classes which proposed to support XML-based interface-independent coordination is demonstrated in Figure 13.

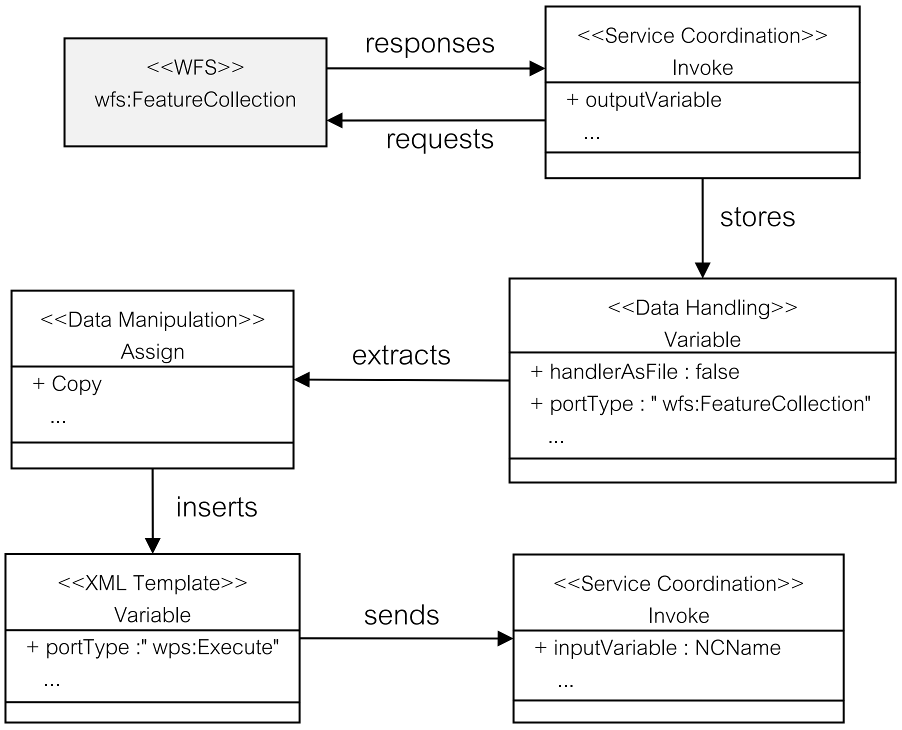


Figure 13 Abstract model which is defined for demonstrating the example of geo-spatial message transformation between two different services.

4.1.3 Caching of Faults

Faults can occurs from three sources via services orchestration: service fault, transportation fault, and orchestration fault. Service such as W3C and OGC web service issues faults by encoding it into XML document defining number and string for fault details. As assumption which is described in Section, the adaptation of interface version can be caused to deliver a coordination fault. The impact of versioning also effects to semantic fault which can be resolved by providing query expression to catching mechanism. For caching transport exceptions, HTTP error status (such as 400, 404, and 500) should be also accessed by an expression. Finally, unexpected exceptions which are occurred by engine itself can be catch by "catchAll". The following Figure 14 demonstrated the model of faults catching definition.

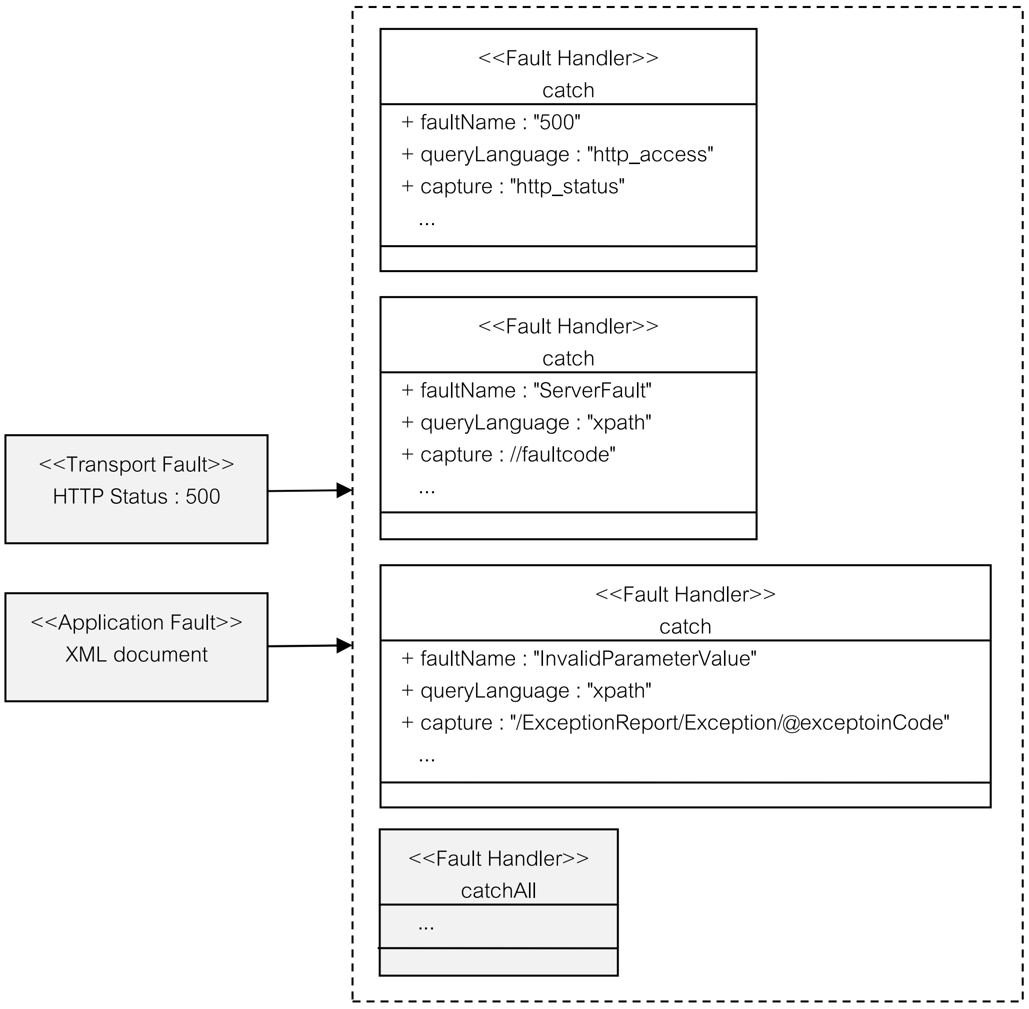

<<Fault Handler>>
catch
+ faultName : "500"
+ queryLanguage : "http_access"
+ capture : "http_status"
...
<<Fault Handler>>
catch
+ faultName : "ServerFault"
+ queryLanguage : "xpath"
+ capture : //faultcode"
...
<<Transport Fault>>
HTTP Status : 500
<<Application Fault>>
XML document
<<Fault Handler>>
catch
+ faultName : "InvalidParameterValue"
+ queryLanguage : "xpath"
+ capture : "/ExceptionReport/Exception/@exceptoinCode"
...
<<Fault Handler>>
catchAll
...

Figure 14 Example model demonstrates platform independent for catching faults.

4.2 Orchestration Language Schema

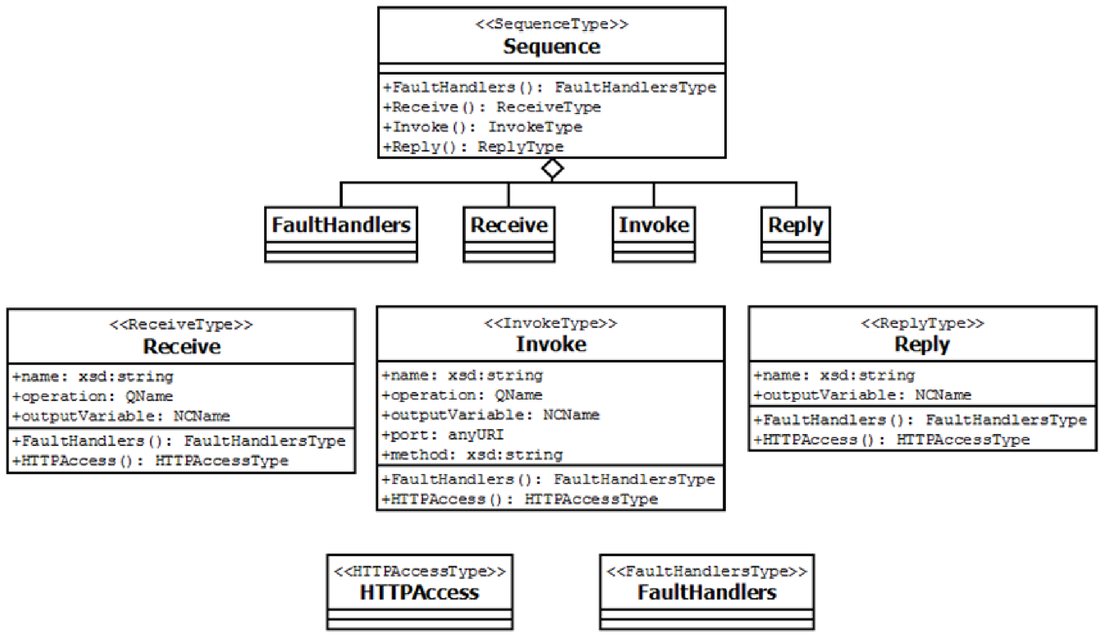
From abstract model and examples as defined in Chapter 3, this section is going to explain more specifics on conditions and constraints of orchestration language. The XML-schemas are grouped and described language in four parts regarding to the conceptual model; Service Coordination Control, Geo-spatial Data Handling, Geo-spatial Message Control, and Exception Handler as follows.

4.2.1 The Service Coordination Control

A number of types of service coordination can be expressed as activities element such as <Receive>, <Invoke> and <Reply>. The BPEL activities are adapted to specify the interaction type with geo-spatial information service. <Receive> activity is used to specify a variable to store a message when service receives a client request message at the beginning of services orchestration. <Reply> activity is used to reply message, data, or streaming object to client. <Invoke> activity is used to specify service execution that contains fifth attributes including:

- operation: is used to specify a similar operation to OGC service operation such as wfs:GetFeature, wfs:GetCapabilities, and wps:Execute.
- inputVariable: is used to define a variable name of input payload which pre-defined in <Variable> tag.
- outputVariable: is used to define a variable name of output payload which pre-defined in <Variable> tag.
- port: is an attribute which used to define web service URL that engine will send request document to.
- method: is an attribute specifying HTTP GET / POST / PUT / DELETE which used to send a message.

The basic activities which are described above are expressed as follows:



```
Example:

<Receive operation=”wps:Execute” outputVariable=”wpsExecuteRequest” />
…
<Invoke  operation="wfs:GetFeature" inputVariable="getHospital"
 outputVariable="responseHospital"
 port="http://161.200.86.131/geoserver/wfs"
 method=”POST” />
…
<Reply outputVariable=”intersectedPolygon” />
```

Invoking some public geo-spatial service requires additional value from HTTP request header such as SOAPAction for server managing incoming connections. To provide mechanism supporting this additional value to be placed in HTTP request header, “invoke” activity is adapted to this requirement by defining an array of header values through <Invoke><HTTPAccess>….</HTTPAccess></Invoke> tags as following schema:

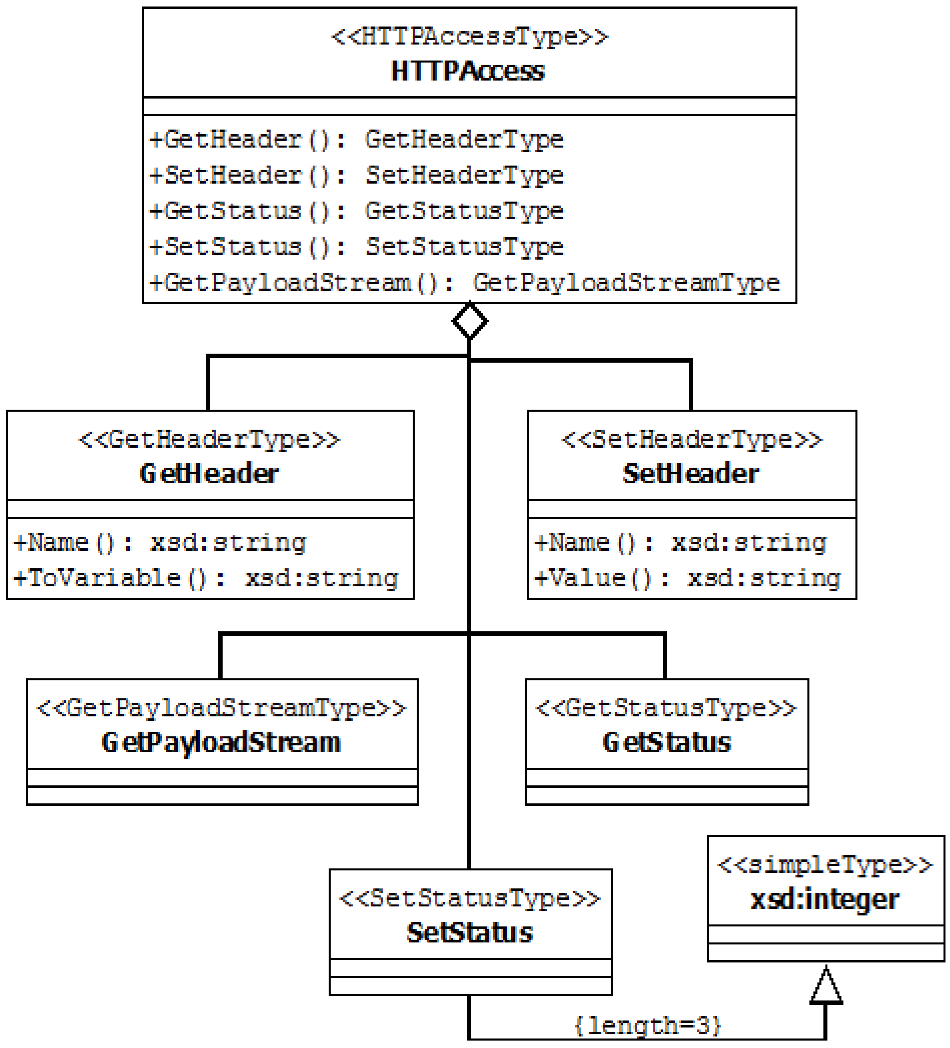


Figure 16 <HTTPAccessType> schema for defining transportation control statement.

The following example demonstrates the invocation to SOAP-based web service that requires SOAPAction value in HTTP request header. This activity sends request to service provider for converting Shapefile to GML.

```
<Invoke operation="wfs:GetFeature" inputVariable="getGmlFromShape"
 outputVariable="responseGml"
 port="http://geobrain.laits.gmu.edu/axis/services/Vector_SHP2GML"
 method="POST">
   <HTTPAccess>
      <SetHeader>
         <Name>SOAPAction</Name>
         <Value>…..</Value>
      </SetHeader>
   </HTTPAccess>
</Invoke>
```

<GetHeader/> is used to capture additional information from service provider which value are stored in <Variable/> by specifying value name in "ToVariable" attribute. <SetHeader/> is used to define additional value to HTTP header. <GetStatus/> and <SetStatus/> are used with "FaultHandler" to manage transportation errors. <GetPayloadStream/> is used to manage file-based geo-spatial data that uses file system to store the payload.

4.2.2 Geo-spatial Data Handling

A number of geo-spatial information messages are ranging from plain text and XML-based document e.g. wms:GetFeatureInfo, GML, CityGML, or wfs:FeatureCollection to byte stream object such as GeoTiff, Shapefile, or Image/png. These geo-spatial messages issued from OGC web services contain two patterns; XML-based document and streaming object.

Orchestration engine is decided to hold geo-spatial messages which is XML-based format by properties that pre-defined in <Variable/> definition.

Attribute name "handlingAsFile" is defined for managing message as storing into temporal file or storing in memory as pain text. Storing into temporal file is used for such data as satellite imagery due to it is usually large. Schema of such <Variable/> activity is demonstrated in Figure 17.

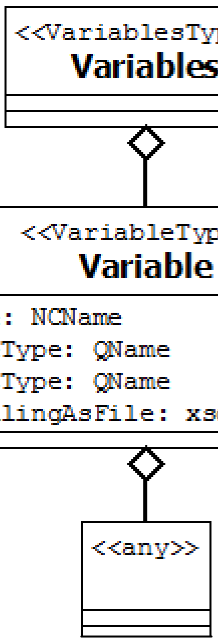


Figure 17 <VariablesType> schema for defining process variables.

The example below shows variable type of XML document and Shapefile respectively:

```
<Variable name=”wfsRequestDoc” mimeType=”application/xml”
   portType=”wfs:GetFeatureType” handlingAsFile="false">...</Variable>
<Variable name=”shapeOfBuilding” mimeType=”application/shp”
   handlingAsFile="true">...</Variable>
```

When orchestration engine participating with sub-services, message such defined in <Variable> tag is sent to that sub-service. Script writer can appends any types of XML-based document to this <Variable/> tag. That kind of XML called "template document". Attribute of <Variable/> such as "portType" is used to the purpose of XML validation. QCName value of "portType" attribute is used to identify an appended document in case of XML validation is required. Attribute "mimeType" is used to verify a data type of handle value that existing at runtime process. The following example demonstrates WFS request with Filter Encoding for variable "getHospital".

```
<Variable name="getHospital" mimeType=”text/xml”
portType="wfs:GetFeatureType">
   <wfs:GetFeature service="WFS" version="1.1.0"
      xmlns:cuwps="http://www.sv.eng.chula.ac.th"
      xmlns:wfs="http://www.opengis.net/wfs"
      xmlns:ogc="http://www.opengis.net/ogc"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.opengis.net/wfs
      http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">
      <wfs:Query typeName="cuwps:bma_admin_poly">
         <ogc:Filter>
            <ogc:PropertyIsEqualsTo>
               <ogc:PropertyName>LOCATION_TYPE</ogc:PropertyName>
               <ogc:Literal>HOSPITAL</ogc:Literal>
            </ogc:PropertyIsEqualsTo>
         </ogc:Filter>
      </wfs:Query>
   </wfs:GetFeature>
</variable>
```

4.2.3 Geo-spatial Message Control

Geo-processing workflow which specifies state of interactions involves the exchange of geo-spatial messages between services requires request messages in each step for performing orchestration. In BPEL, <Assign/> activity can be used to define such requirement using <Copy/> to copy message from one variable to another, or using <Insert/> to add specific value to defined variable. XPath and XSLT are applied to support this message manipulation mechanism. The example of using <Assign/> activity is shown as follows:

```
<Assign name="assignHospitalToBuffering">
   <Copy>
      <From inputValue="responseHospital"/>
      <To outputValue="getBufferingHospital">{$hospitalToBeBuffered}</To>
   </Copy>
</Assign>
```

The example demonstrates the use of element <Copy> to copy a message from one variable to another by replacing value at {$buildingToBeBuffered} which is declared in output variable context. Some OGC web services publish a sizeable geo-spatial message such as wfs:FeatureCollection or wcs:Coverage. The engine must uses that value by reference instead of value directly. Attribute "byReference" is an option (default is "false") used to assign orchestration engine to store data into file system, and uses that data by reference value such as URL.

Assign activity also supports XPath and XSLT similar to BPEL for accessing value in XML DOM as shown in example below.

```
<Assign name="assignUserHospitalDistance">
   <Copy>
      <From inputValue="wpsExecuteRequest">//Input[1]/LiteralValue</From>
      <To outputValue="getBufferingHospital">{$hospitalDistance}</To>
   </Copy>
</Assign>
```

Geo-spatial data, such as wfs:FeatureCollection responded from WPS service, may require spatial filter as input parameter for next request. The following example show how to attach Filter Encoding (<ogc:Filter/>) in <Assign/> activity.

```
<Assign name="assignRequiredIntersectArea">
   <Filter>
      <From inputValue="responseIntersectPolygon">
         <ogc:Filter>
            <ogc: PropertyIsGreaterThan>
               <ogc:PropertyName>AREA</ogc:PropertyName>
                  <ogc:Literal>2000</ogc:Literal>
            </ogc: PropertyIsGreaterThan>
         </ogc:Filter>
      </From>
      <To outputValue="suitableArea">{$suitableAreaFeature}</To>
   </Filter>
</Assign>
```

XML schema of assign activity can be described as follows:

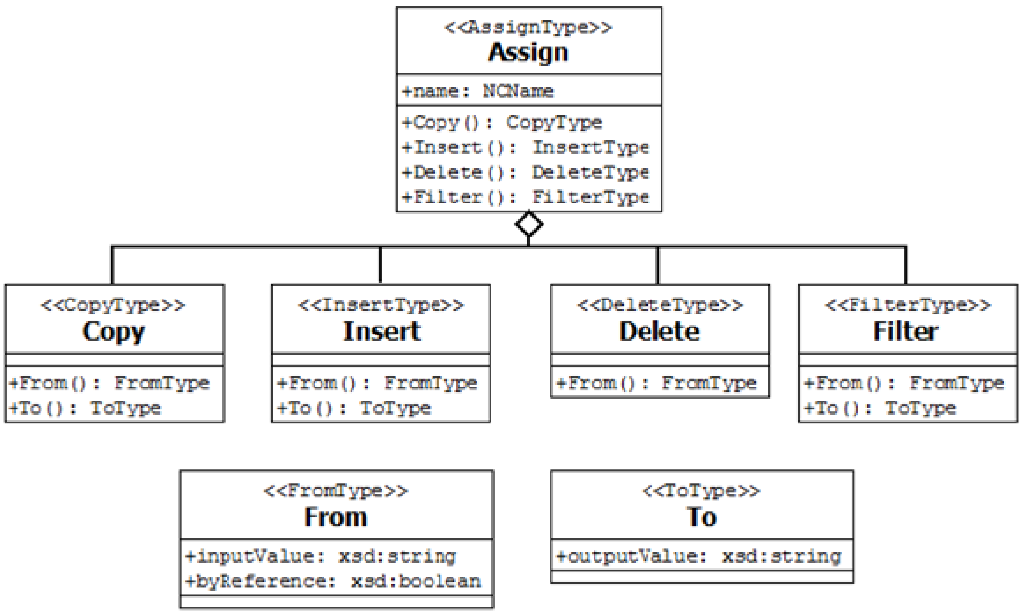


Figure 18 <AssignType> schema for transforming information between messages.

4.2.4 Exception Handler

In complex geo-spatial services chaining with involving number of services, the demand for managing exceptions are required that not only for debugging purpose, but also for the compensation mechanism for geo-spatial web services composition (Yanfeng et al., 2005).

A number of ways can be used to manage an exception responded from geo-spatial information service when the operation in service chain is invalid. Fault message such as ows:MissingParameterValue, soapenv:Service.userException and so on can be distributed from web service. Each service shall respond an Exception Report message that describes what request is invalid. Similar to typical programming language, the exception gives the glue code easier to maintain. Using fault handlers is simple, and it is similar to the way to catching exception in Java using Try-Catch block. Fault handlers can be declared globally across an entire process or directly on an <Invoke> activity locally. The fault handler that used to declare a catching of invalid work is demonstrated as follows:

```
<ProcessDefinition …>
<Variables>…</Variables>
<Activities>
   <!--Global Fault Handlers-->
   <faultHandlers>
      <catch faultName=””>…</catch>
      <catchAll>…</catchAll>
   </faultHandlers>
   <Sequence>
      <Receive operation="wps:Execute" />
      <Invoke operation="wfs:GetFeature" >
         <!--Local Fault Handlers-->
         <faultHandlers>
            <catch faultName=”ows:MissingParameterValue”>…</catch>
            <catch faultName=”ows:InvalidParameterValue”>…</catch>
            <catch faultName=”soapenv:Server.userException”>…</catch>
         </faultHandlers>
      </Invoke>
      <Invoke operation="wfs:GetFeature" />
      …
      <Assign name="assignSchoolToBuffering” />
      …
      <Reply outputVariable="intersectedPolygon" />
   </Sequence>
</Activities>
</ProcessDefinition>
```

XML schema of fault handler activity can be described as follows:

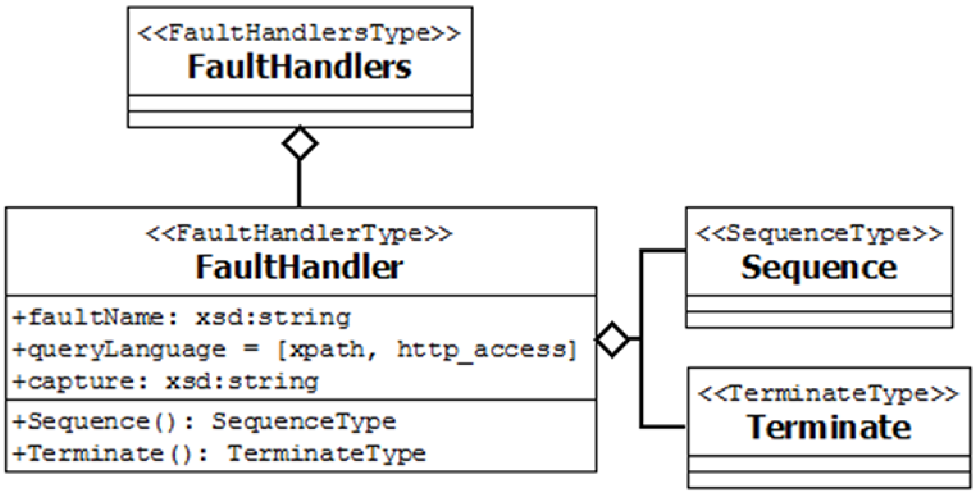


Figure 19 <FaultHandlersType> schama for catching exception during process.

4.3 Orchestration Engine Implementation

The implementation mainly demonstrated the components of orchestration engine regarding to orchestration model described in Section 4.2. For publishing orchestration engine as web service, a standard web service framework must be implemented first. Selected framework especially for implementing geo-spatial web service is WPS-T (Transactional Web Processing Service) due to interoperability purpose of other OGC web services which be able to communicate with this service.

The orchestration service framework was inherited from WPS-T (Schaeffer, 2008). In WPS-T, WPS is extended to support on-the-fly deployment processes. WPS-T offers <DeployProcessRequest/> and <UndeployProcess/> interface for clients expose orchestration script to WPS and then each sub-process can be executed through execution engine that running in backend.

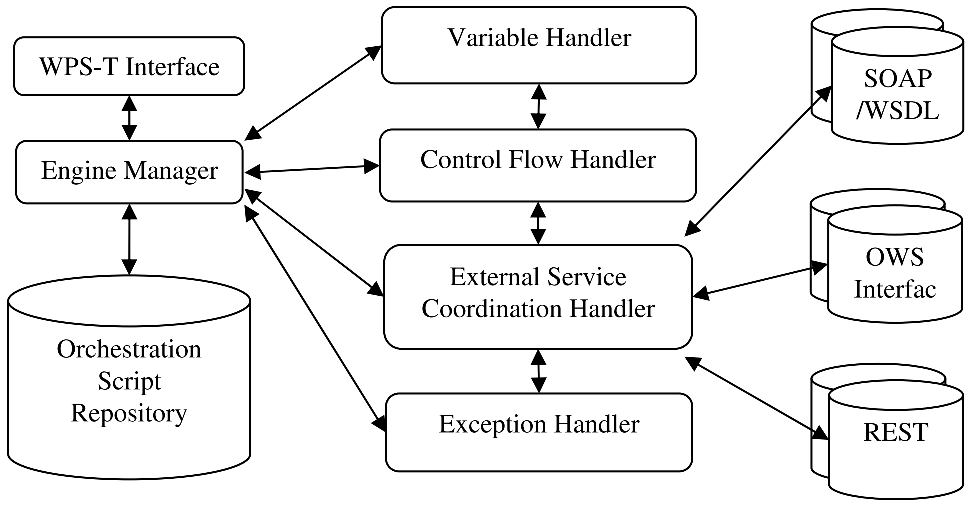


Figure 20 Proposal to Geo-spatial Orchestration Service framework.

In Figure 20, orchestration engine is located at the backend of orchestration service which controlled by Engine Manager. The Engine Manager is composed from components i.e., Variable Handler, Control Flow Handler, External Service Coordination Handler, and Exception Handler. Client pre-defined script is stored in Orchestration Script Repository. When client requests an execution through WPS operation (<wps:Execute/>), Engine Manager will read orchestration script from repository for interpretation and execution. The result of execution will be returned to user via WPS interface.

Chapter 5
Orchestration Testing

In this chapter, First section describes geo-spatial services implementation. Each geo-spatial service acts as sub-service which appeared in orchestration activity (defined in <Invoke/> tag). Sub-services consist of services platforms; SOAP web service (W3C web service), REST web service, OGC WPS, and OGC WFS. These geo-spatial services are prepared for the testing of services orchestration in each scenario.

Section 0 - 5.4 explain the testing scenarios. All geo-spatial services which are prepared in the first section (5.1) are used to compose the orchestration workflow in each scenario. The scenario is selected and derived from generic geo-processing tasks. Each scenario is setup to a test responding to the purposes as follows:

- Site selection scenario: the standard interfaces such as REST web service, OGC WFS, and OGC WPS are orchestrated. Additionally, Filter Encoding support in new orchestration language is also testing by filter features which retrieved from REST web service.
- Coordinate transformation scenario: the different standard interfaces such as SOAP web service, OGC WFS, and OGC WPS are orchestrated.
- Road network analysis scenario: the standard interfaces such as OGC WFS and OGC WPS are orchestrated. Additionally, caching services exception is tested during the instance of services orchestration.

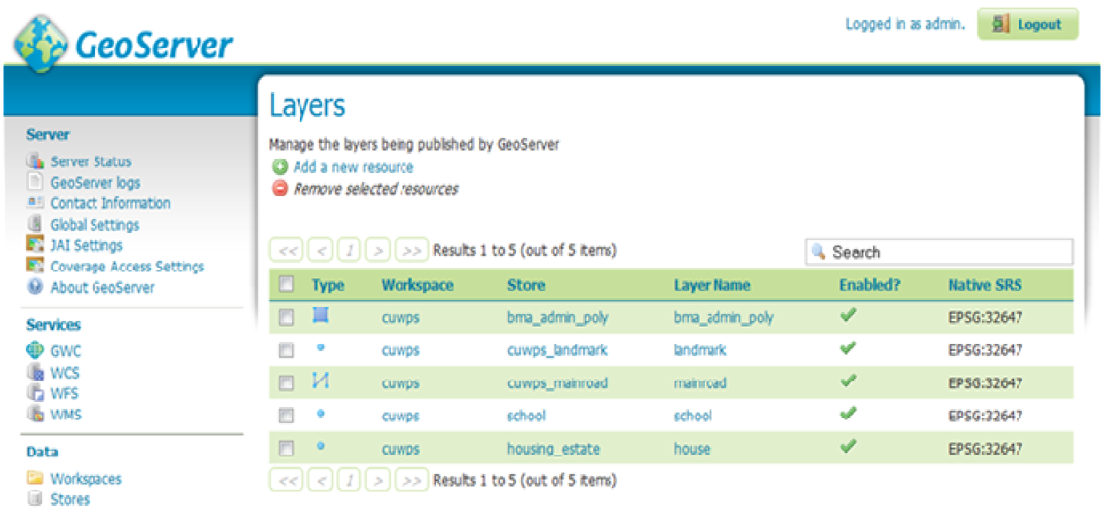
5.1 Sub-services Development

Sub-services are the process actors that appear in an orchestration workflow. Each process actor is a similar to the one of the geo-spatial service standard. Thus, for the testing of services orchestration, each sub-service must be firstly implemented. The following sub-sections demonstrate those implementations i.e., Web Feature Service, Web Processing Service, REST Web Service, and SOAP Web Service. The multiple technologies are involved and described as follows:

5.1.1 Web Feature Services

Open source software, GeoServer, is used to setup the OGC WFS services (Figure 21). Three feature services are published through this standard as follows:

- Landmark feature service: is publishing a landmark feature such as government building, temple, police station, and market.
- School feature service: is publishing the location of the schools in Bangkok district.
- House feature service: is publishing a house feature which for a person who wants to find a house for renting or buying.



Layers

Manage the layers being published by GeoServer

Add a new resource

Remove selected resources

Results 1 to 5 (out of 5 items)

Type	Workspace	Store	Layer Name	Enabled?	Native SRS
	cuwps	bma_admin_poly	bma_admin_poly	✓	EPSG:32647
	cuwps	cuwps_landmark	landmark	✓	EPSG:32647
	cuwps	cuwps_mainroad	mainroad	✓	EPSG:32647
	cuwps	school	school	✓	EPSG:32647
	cuwps	housing_estate	house	✓	EPSG:32647

Results 1 to 5 (out of 5 items)

Figure 21 Setting up WFSs (i.e., landmark, school, and house feature service) through open source software "GeoServer" for publishing features via OGC Web Feature Service interface.

5.1.2 REST Web Service for W3C Feature Service

Concept of REST is simply such as client initiates request to a service then server (service) processes a request and returns the appropriate response. In this chapter, REST is implemented for representing the feature resource. Hospital features are setup in GML encoded document then the features can be accessed by HTTP GET protocol. The entire features shell respond to the client without a support on feature filter. The following example of HTTP GET protocol is used to demonstrate the request of hospital features

```
GET /hosptials.gml HTTP/1.1
Host: localhost
```

Through this REST resource the design of Filter Encoding supporting in an orchestration language can be tested by assigning a filter statement to an orchestration script. The first scenario, "site selection", is prepared to demonstrate the Filter Encoding support in an orchestration language and the coordination to the REST web service.

5.1.3 Web Processing Services for Buffer and Intersection Operation

The framework of using power of spatial database to develop WPS is described in Chunithipaisan and Supavetch, (2009) and Supavetch and Chunithipaisan, (2011). The framework is a based architecture for OGC WPS implementation. Spatial functions such as ST_Buffer and ST_Intersection of PostGIS is used to provide Buffer and Intersection operation for WPS. The framework of which using spatial database functions implementing WPS operation is demonstrated as belowing Figure:

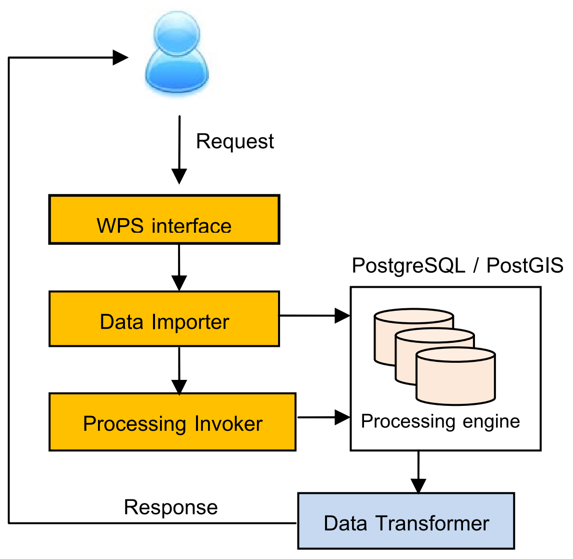


Figure 22 WPS framework of using power of spatial database to process the request features.

In above Figure, the Data Importer is responsible to get a feature type from a feature resource, and then interpret its feature type for creating a temporary table in

PostgreSQL. After table was created, Data Importer inserts features into the temporary table. Then Processing Invoker executes a processing function in a pre-defined SQL statement. After features are processed, Data Transformation will transform the features into GML format then responds to user as WPS response document.

The Buffer operation requires two input parameters; *features resource* and *buffer distance*. The following example is a WPS execution document which is used to request a buffer operation within distance 200 meters around each feature:

```
<wps:Execute version="1.0.0" service="WPS" >
   <ows:Identifier>BufferFeatureCollection</ows:Identifier>
   <wps:DataInputs>
      <wps:Input>
         <ows:Identifier>features</ows:Identifier>
            <wps:Data>
            <wps:ComplexData mimeType="text/xml; subtype=wfs-
collection/1.0"><![CDATA....]]></wps:ComplexData>
         </wps:Data>
      </wps:Input>
      <wps:Input>
        <ows:Identifier>buffer</ows:Identifier>
        <wps:Data>
         <wps:LiteralData>200</wps:LiteralData>
        </wps:Data>
      </wps:Input>
   </wps:DataInputs>
   <wps:ResponseForm>
      <wps:RawDataOutput mimeType="text/xml; subtype=wfs-collection/1.0">
        <ows:Identifier>result</ows:Identifier>
      </wps:RawDataOutput>
   </wps:ResponseForm>
</wps:Execute>
```

The Intersection operation requires two features layers; *InputFirstFeatures* and *InputSecondFeatures*. Below XML is an example of a WPS request document for the intersection operation.

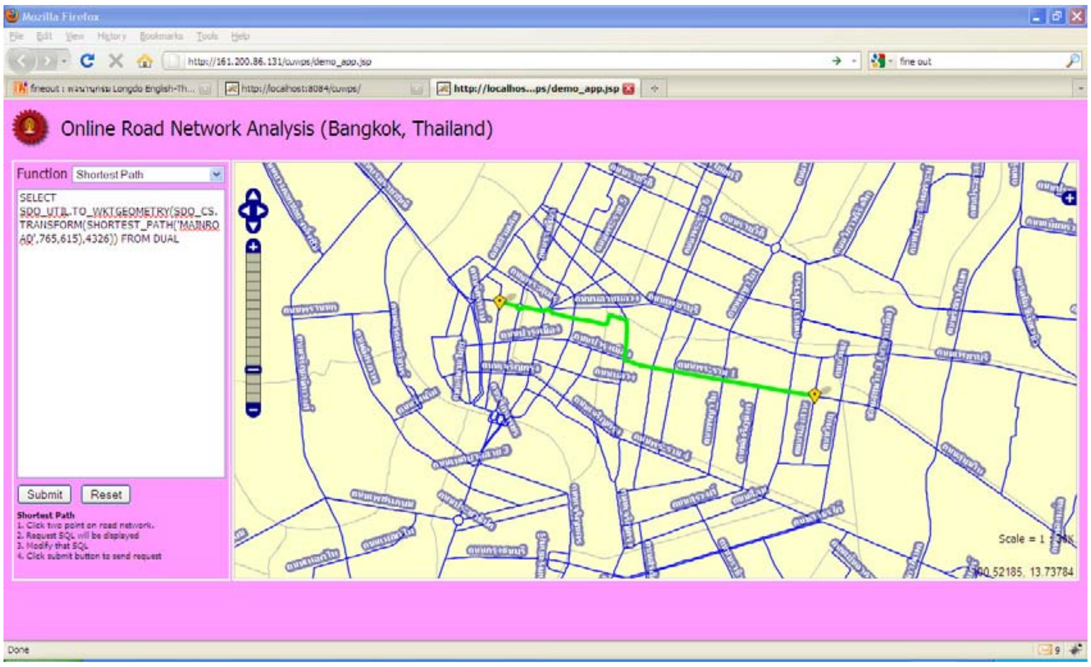
```
<wps:Execute version="1.0.0" service="WPS" >
   <ows:Identifier>Intersect</ows:Identifier>
   <wps:DataInputs>
      <wps:Input>
         <ows:Identifier>InputFirstFeatures</ows:Identifier>
         <wps:Data>
            <wps:ComplexData mimeType="text/xml; subtype=wfs-
collection/1.0"><![CDATA....]]></wps:ComplexData>
         </wps:Data>
      </wps:Input>
      <wps:Input>
         <ows:Identifier>InputSecondFeatures</ows:Identifier>
         <wps:Data>
            <wps:ComplexData mimeType="text/xml; subtype=wfs-
collection/1.0"><![CDATA....]]></wps:ComplexData>
         </wps:Data>
      </wps:Input>
   </wps:DataInputs>
   <wps:OutputDefinitions>
      <wps:Output mimeType="text/xml; subtype=wfs-collection/1.0">
        <ows:Identifier>IntesectedPolygon</ows:Identifier>
      </wps:Output>
   </wps:OutputDefinitions>
</wps:Execute>
```

5.1.4 Web Processing Service for Road Network Analysis

The network analysis functions are commonly implemented in a transportation industry. Road network model is huge, complex and divers implementations. WPS framework described in Section 5.1.3 is also implemented again for a road network analysis. Oracle spatial is used to create and analyze network model. Network model is stored in three tables; MAINROAD_NODES$, MAINROAD_LINK$ and MAINROAD_PATH$. Features of a network are SDO_GEOMETRY type. Custom PL/SQL function is defined and prepared for a *shortest_path* operation. The shortest path function requires two input features as the start node and the end node for finding shortest route between these two features. The return of the function is a line feature which represented as a shortest path between the two input nodes.

Additionally, *tsp_path (Travelling Salesman Problem)* operation is another WPS operation. TSP function provides client to find the way to reach a place more than one place. For example, sale man has a plan to meet three customers before going home. The way (route) to meet all customers is the answer of the *tsp_path* operation. The operation requires multiple features for representing the places to be met. The return of the function is a line feature which is the way (route) that meets all places.

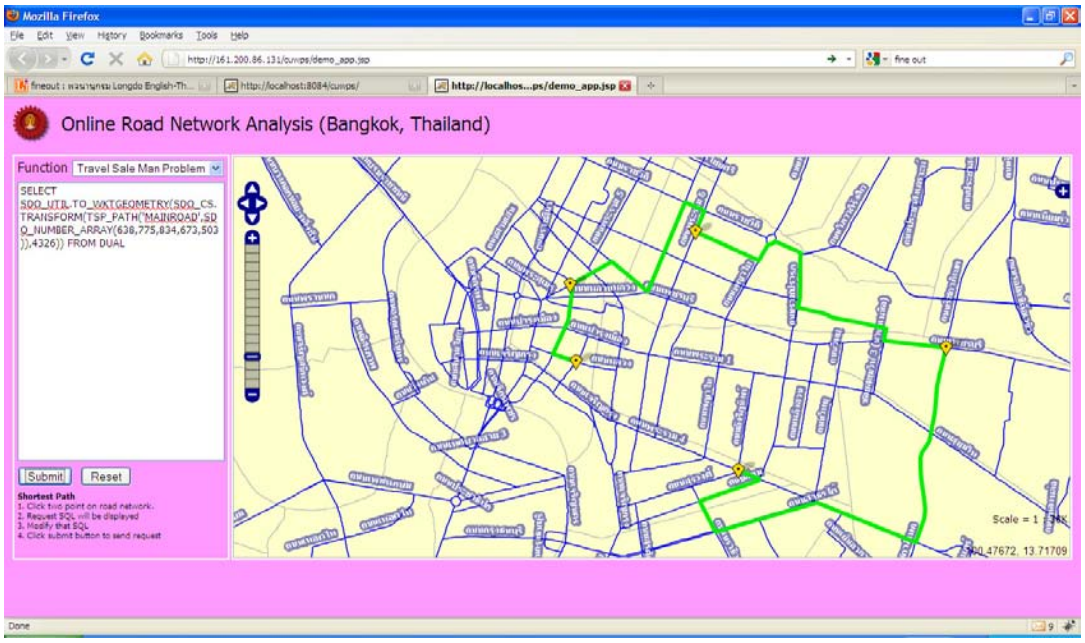
The example of Road Network Analysis Service is displayed in Figure 23 and Figure 24.


Mozilla Firefox
http://161.200.86.131/cuwps/demo_app.jsp
Online Road Network Analysis (Bangkok, Thailand)
Function
Shortest Path
Submit
Reset
Scale = 1 : 74K
100.52185, 13.73784
Done

```
SELECT
SDO_UTIL.TO_WKTGEOMETRY(SDO_CS.
TRANSFORM(SHORTEST_PATH('MAINRO
AD',765,615),4326)) FROM DUAL
```

Shortest Path
1. Click two point on road network.
2. Request SQL will be displayed
3. Modify that SQL
4. Click submit button to send request

Figure 23 The example of *shortest_path* operation for finding shortest route between two nodes.



```
SELECT
SDO_UTIL.TO_WKTGEOMETRY(SDO_CS.
TRANSFORM(TSP_PATH('MAINROAD',SD
O_NUMBER_ARRAY(638,775,834,673,503
)),4326)) FROM DUAL
```

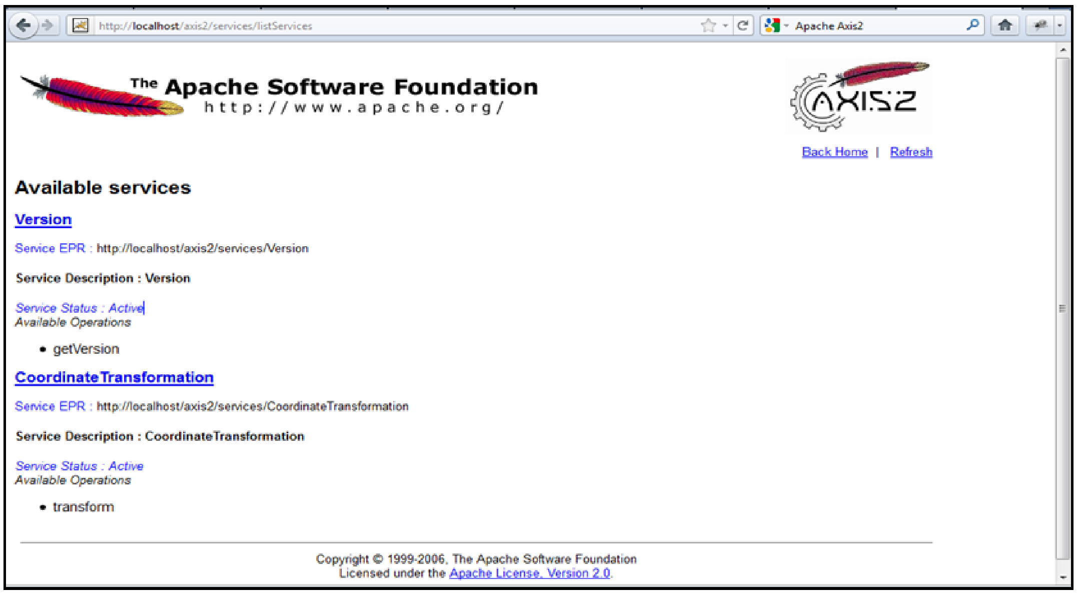
Shortest Path
1. Click two point on road network.
2. Request SQL will be displayed
3. Modify that SQL
4. Click submit button to send request

Figure 24 The example of *tsp_path* operation which the result is the shortest route that meets all nodes.

5.1.5 SOAP Web Service for Coordinate Transformation Service

The Coordinate Transformation Service provides a datum transformation operation to clients. Features which needs datum transformation between Indian 1975 and WGS84 can be transformed though this service. The service complies with W3C web service standards such as SOAP and WSDL. Apache Axis2, the W3C web service framework, is used to develop the coordinate transformation engine. The service uses open source such as GeoTools library for programming the backend transformation. This service provides a *transform* operation (Figure 25) which clients can send their features via SOAP in a request form as following XML.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instnace"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<soap:Body>
<cuwps:Transform>
   <cuwps:Features ...><![CDATA[...]]></cuwps:Features>
   <cuwps:Function>Indian1975toWGS84</cuwps:Function>
</cuwps:Transform>
</soap:Body>
</soap:Envelope>
```


http://localhost/axis2/services/listServices

Apache Axis2

The Apache Software Foundation
http://www.apache.org/

Back Home | Refresh

Available services

Version

Service EPR : http://localhost/axis2/services/Version

Service Description : Version

Service Status : Active
Available Operations

- getVersion

CoordinateTransformation

Service EPR : http://localhost/axis2/services/CoordinateTransformation

Service Description : CoordinateTransformation

Service Status : Active
Available Operations

- transform

Copyright © 1999-2006, The Apache Software Foundation
Licensed under the Apache License, Version 2.0.

Figure 25 Coordinate transformation service which is deployed in Apache Axis2.

5.2 Site Selection Scenario

5.2.1 Scenario Overview

The scenario for testing an orchestration engine is derived from common problem of GIS projects such as “site selection” algorithm (Stollberg and Zipf, 2008). The site selection is about finding the suitable area from various factors involving a different specific location. For example, a family may look for a house near a school and hospital in a specific distance between those things. The question for this algorithm could be “In which area do I have to rent a house from housing estate which is within distance of 0.5 km of a hospital, and within a distance of 0.4 km of a school. In practical the services which is orchestrated by engine, requires WFS service to provide a location of hospital, school, and housing estate. WPS is used for invoking “buffer” and “intersect” operation for buffering location features to polygon. The intersection of hospital polygon, school polygon and housing estate polygon is made for achieving an intersected area that user will look for renting a house.

In site selection scenario, an orchestration script consists of five geo-spatial services containing the three different services interface (i.e., REST, WFS, and WPS). Five geo-spatial services which displayed in Figure 26 are the actors (sub-services) of "site selection" scenario. Each service acts as sub-process in which provided by different providers. Algorithm for finding an intersection area is translated into a pre-defined orchestration script that will be explained in the next section.

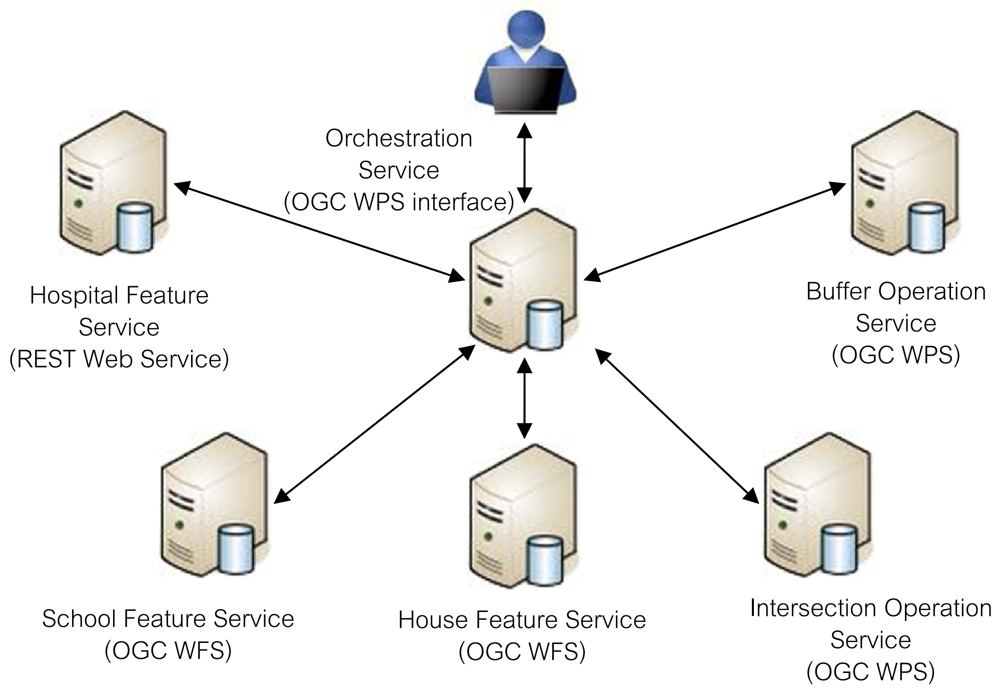

Orchestration Service (OGC WPS interface)
Hospital Feature Service (REST Web Service)
Buffer Operation Service (OGC WPS)
School Feature Service (OGC WFS)
House Feature Service (OGC WFS)
Intersection Operation Service (OGC WPS)

Figure 26 Geo-spatial services which are sub-process actors and are used to evaluate geo-spatial services orchestration.

5.2.2 Defining an Orchestration Script

We propose procedure to define an orchestration script from any geo-processing workflows by following the way that defined in a sequence diagram. Many GI systems that have a dynamic behavior can be expressed in terms of specific sequence of messages. Firstly, defining sequence diagram for describing site selection workflow that showing how process operate which others in a sequence order. WFS web service standard is used to distribute the location and information of schools and housing estate in Bangkok. Hospital location and detail is distributed via REST web service. WPS services are used to provide GIS functions such as Buffer and Intersect for decision making. The orchestration engine itself is also provided through WPS service interface.

The following sequence diagram demonstrates the processes in the order from top to bottom. The engine receives three buffering distances of hospital, school and housing estate from client request. Then the engine starts to retrieve feature that represents the location of hospital, school and housing estate from REST and WFS services. Each features layer is buffered at WPS service with user specified distance. After finish buffering three location features, the results of three polygons layers will sending to another WPS for intersection in a last step.

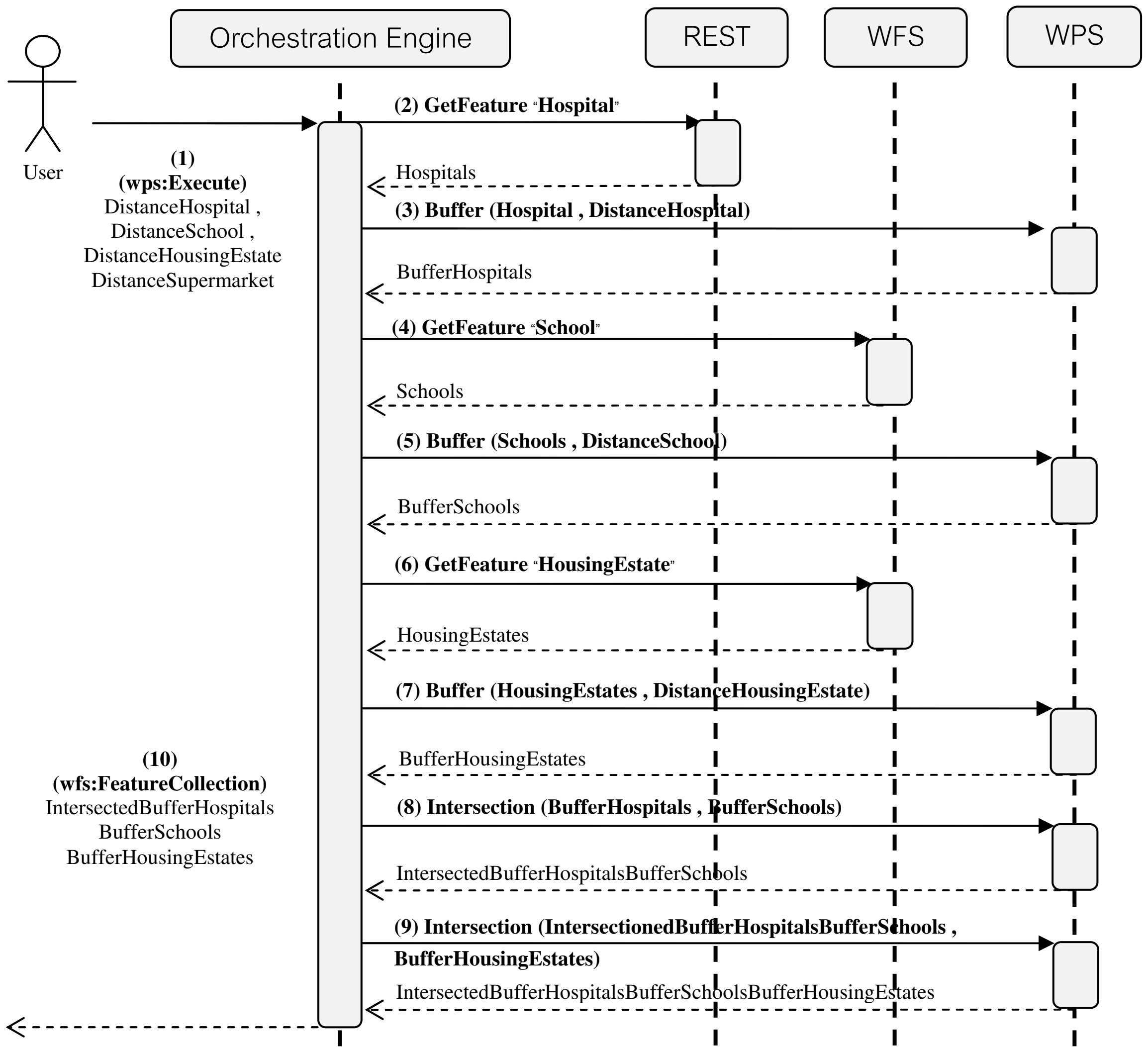

Orchestration Engine
REST
WFS
WPS
User
(1)
(wps:Execute)
DistanceHospital ,
DistanceSchool ,
DistanceHousingEstate
DistanceSupermarket
(2) GetFeature "Hospital"
Hospitals
(3) Buffer (Hospital , DistanceHospital)
BufferHospitals
(4) GetFeature "School"
Schools
(5) Buffer (Schools , DistanceSchool)
BufferSchools
(6) GetFeature "HousingEstate"
HousingEstates
(7) Buffer (HousingEstates , DistanceHousingEstate)
BufferHousingEstates
(8) Intersection (BufferHospitals , BufferSchools)
IntersectedBufferHospitalsBufferSchools
(9) Intersection (IntersectionedBufferHospitalsBufferSchools ,
BufferHousingEstates)
IntersectedBufferHospitalsBufferSchoolsBufferHousingEstates
(10)
(wfs:FeatureCollection)
IntersectedBufferHospitals
BufferSchools
BufferHousingEstates

Figure 27 Site Selection sequence diagram.

Translating above sequence diagram (Figure 27) into an orchestration language as following steps:

Step 1: Get familiar with partner geo-spatial services (e.g., REST, WFS, and WPS). Writer has to aware what's version of service interface, acknowledges the capabilities among that services, and be able to write the request document.

Step 2: Defining the request document for executing each geo-spatial web service e.g., wfs:GetFeature, wps:Execute as follows Figure 3 and Figure 4.

WFS GetFeature:

```
<wfs:GetFeature service="WFS" version="1.1.0" outputFormat="GML2”>
      <wfs:Query typeName="cuwps:landmark">
         <ogc:Filter>
            <ogc:PropertyIsEqualsTo>
               <ogc:PropertyName>LOCATION_TYPE</ogc:PropertyName>
               <ogc:Literal>SCHOOL</ogc:Literal>
            </ogc:PropertyIsEqualsTo>
         </ogc:Filter>
      </wfs:Query>
</wfs:GetFeature>
```

WPS Execute:

```
<wps:Execute version="1.0.0" service="WPS">
   <ows:Identifier>gt:BufferFeatureCollection</ows:Identifier>
   <wps:DataInputs>
      <wps:Input>
         <ows:Identifier>features</ows:Identifier>
         <wps:Data>
            <wps:ComplexData mimeType="text/xml; subtype=wfs-collection/1.0">
               <![CDATA[{$hospitalFeatures}]]>
            </wps:ComplexData>
         </wps:Data>
      </wps:Input>
      <wps:Input>
         <ows:Identifier>buffer</ows:Identifier>
         <wps:Data>
            <wps:LiteralData>{$hospitalDistance}</wps:LiteralData>
         </wps:Data>
      </wps:Input>
   </wps:DataInputs>
   <wps:ResponseForm>
      <wps:RawDataOutput mimeType="text/xml; subtype=wfs-collection/1.0">
         <ows:Identifier>result</ows:Identifier>
      </wps:RawDataOutput>
   </wps:ResponseForm>
</wps:Execute>
```

Step 3: Develop the orchestration script.

Declare all variables involving geo-spatial service coordination that defined in step 2 into <Variable> tag, and then writing the process logic definition into <Activities/>.

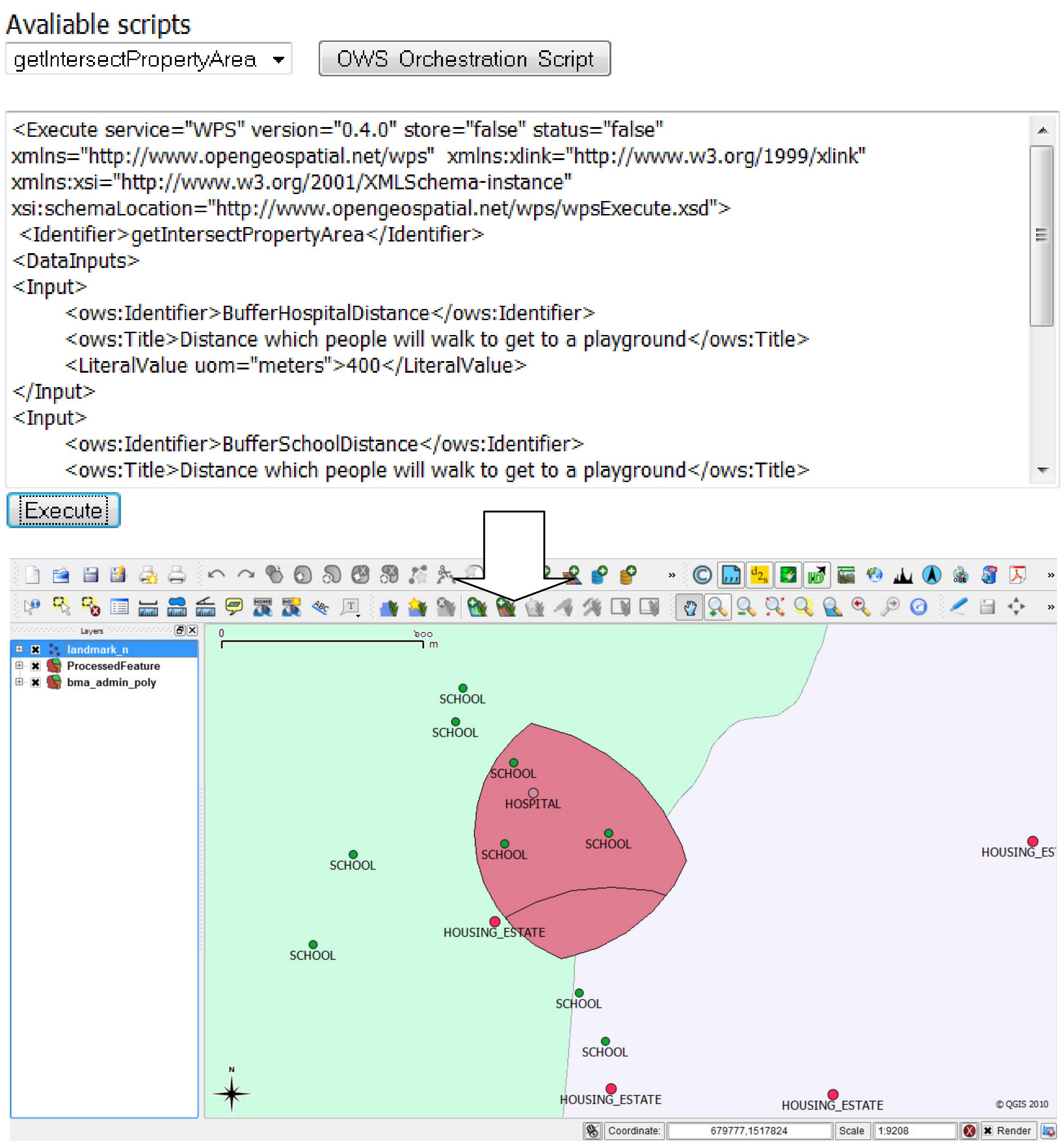
The following script is a shorthand script of the whole site selection workflow that is derived from sequence diagram in Figure 27.

```
<cuwps:ProcessDefinition name=”SiteSelection”>
   <Variables>
      …
   </Variables>
   <Activities>
      <Sequence>
         <Receive operation="wps:Execute" outputVariable="wpsExecuteRequest"
/>
         <!—Assign User Specific Distance into WPS:Execute Document -->
         <Assign name=”assignUserHospitalDistance”>…</Assign>
         <Assign name=”assignUserSchoolDistance”>…</Assign>
         <Assign name=”assignUserHousingEstateDistance”>…</Assign>
         <!-- GetFeature and Buffer -->
         <!-- Invoke REST web service -->
         <Invoke operation="getHospital"
            outputVariable="responseHospital"
            port="http://localhost:8080/georest/hospital" />
         <Assign name=”assignHospitalToBuffering”>…</Assign>
         <!-- Invoke OGC Web Services -->
         <Invoke  operation="wps:Execute" inputVariable="getBufferingHospital"
            outputVariable="bufferedHospital"
            port="http://localhost:8080/geoserver/wps"/>
         <Invoke  operation="wfs:GetFeature" inputVariable="getSchool"
            outputVariable="responseSchool"
            port="http://localhost:8080/geoserver/wfs"/>
         <Assign name="assignSchoolToBuffering">…</Assign>
         <Invoke  operation="wps:Execute" inputVariable="getBufferingSchool"
            outputVariable="bufferedSchool"
            port="http://localhost:8080/geoserver/wps"/>
         <Invoke  operation="wfs:GetFeature" inputVariable="getHousingEstate"
            outputVariable="responseHousingEstate"
            port="http://localhost:8080/geoserver/wfs"/>
         <Assign name="assignHousingEstateToBuffering">…</Assign>
         <Invoke  operation="wps:Execute"
inputVariable="getBufferingHousingEstate"
            outputVariable="bufferedHousingEstate"
            port="http://localhost:8080/geoserver/wps"/>
         <!-- Intersect Features -->
         <Assign name="assignHospitalToIntersect">…</Assign>
         <Assign name="assignSchoolToIntersect">…</Assign>
         <Invoke  operation="wps:Execute"
inputVariable="getIntersectHospitalSchool"
            outputVariable="intersectedSchoolHospitalPolygon"
            port="http://localhost:8080/cuwps/wps_intersect.jsp"/>
         <Assign name="assignHospitalSchoolToIntersect">…</Assign>
         <Assign name="assignHousingEstateToIntersect">…</Assign>
         <Invoke  operation="wps:Execute"
```

```
            inputVariable="getIntersectHospitalSchoolHousingEstate"
            outputVariable="intersectedSchoolHospitalHousingEstatePolygon"
            port="http://localhost:8080/cuwps/wps_intersect.jsp"/>
         <!—Response Result to User -->
         <Reply outputVariable="intersectedSchoolHospitalHousingEstatePolygon"
/>
      </Sequence>
   </Activities>
</cuwps:ProcessDefinition>
```

5.2.3 Script Execution

Which this first scenario, WPS-T is examined that user can read, execute, deploy, and undeploy orchestration script to service repository. The script is exposed to user/client as such a simple process. The following figure shows an execution with the result is displayed in GIS software named QGIS.



```
<Execute service="WPS" version="0.4.0" store="false" status="false"
xmlns="http://www.opengeospatial.net/wps"  xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengeospatial.net/wps/wpsExecute.xsd">
 <Identifier>getIntersectPropertyArea</Identifier>
<DataInputs>
<Input>
        <ows:Identifier>BufferHospitalDistance</ows:Identifier>
        <ows:Title>Distance which people will walk to get to a playground</ows:Title>
        <LiteralValue uom="meters">400</LiteralValue>
</Input>
<Input>
        <ows:Identifier>BufferSchoolDistance</ows:Identifier>
        <ows:Title>Distance which people will walk to get to a playground</ows:Title>
```

Figure 28 Showing WPS request and response of site selection scenario testing.

5.2.4 Summary

In this site selection scenario the three heterogeneous geo-spatial services such as REST, WFS, and WPS are orchestrated. The solution of intersected area which near hospital, school and housing estate proofed that orchestration script and engine can performs geo-spatial services orchestration.

5.3 Coordinate Transformation Scenario

5.3.1 Scenario Overview

Various resources, which have different providers, usually have a different coordinate reference system. For example, Department of Lands develops WFS for publishing land parcel which referenced to local spatial coordinate system (UTM Zone 47 N, Indian 1975). While another provider such as Department of City Planning develops WFS for publishing landmarks which referenced to WGS84 (UTM Zone 47 N, WGS84).

A variety of coordinate reference systems from different layers are frequently found in any geographical analysis. Coordinate transformation is always required for making those different layers into the same coordinate system. Thus, this scenario is set up to demonstrate another example of orchestration. The problem which is used to define the workflow diagram is "To find out the detail of land parcel by landmark name".

This coordinate transformation scenario consists of three sub-services; landmark location service (WFS), parcel location service (WFS), feature transformation service (SOAP Web Service), and intersection operation service (WPS). Each feature service is reference to different coordinate system as follows.

- Landmark location service is referenced on Geographic Coordinate System.
- Parcel location service is referenced on Projected Coordinate System such UTM Zone 47 N (Indian 1975 datum).

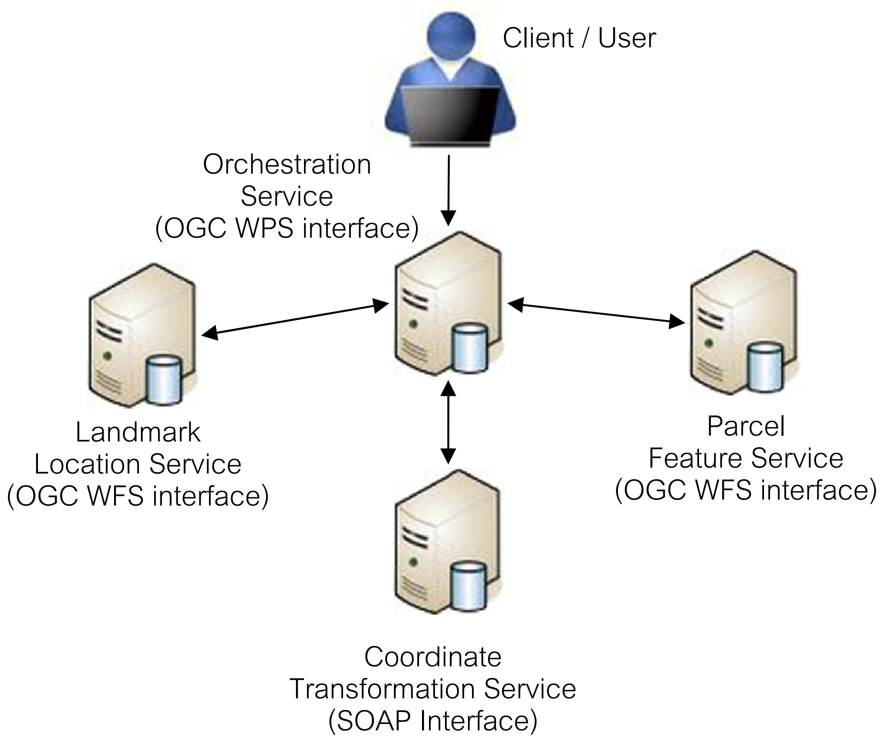

Client / User
Orchestration Service (OGC WPS interface)
Landmark Location Service (OGC WFS interface)
Parcel Feature Service (OGC WFS interface)
Coordinate Transformation Service (SOAP Interface)

Figure 29 Sub-service actors of Coordinate Transformation Scenario.

5.3.2 Sequence Diagram of Coordinate Transformation Scenario

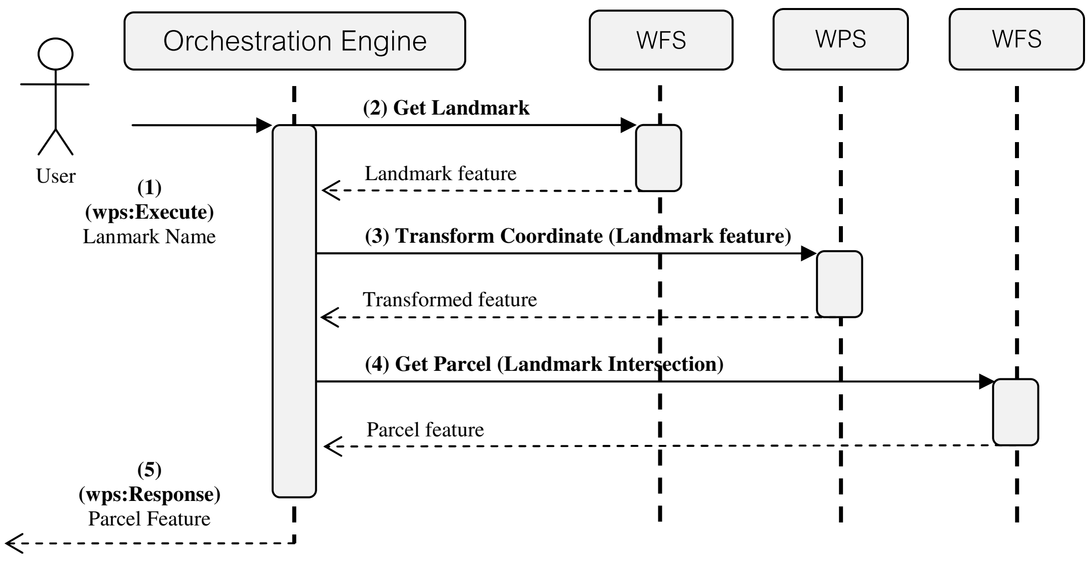

Orchestration Engine
WFS
WPS
WFS
User
(1) (wps:Execute) Lanmark Name
(2) Get Landmark
Landmark feature
(3) Transform Coordinate (Landmark feature)
Transformed feature
(4) Get Parcel (Landmark Intersection)
Parcel feature
(5) (wps:Response) Parcel Feature

Figure 30 Sequence diagram of Coordinate Transformation Scenario

Figure 30 shows a sequence diagram of finding land parcel in which land mark is located. First, user specifies the name of landmark then sending as input parameter to orchestration service. Orchestration service sends request to the landmark location service wfs:GetFeature with filter statement for finding landmark after received the landmark name. Orchestration service retrieve landmark feature then sending to coordinate transformation service. Landmark coordinate sends to parcel location service for parcel intersection. Then, parcel shall return to user via wps:Response.

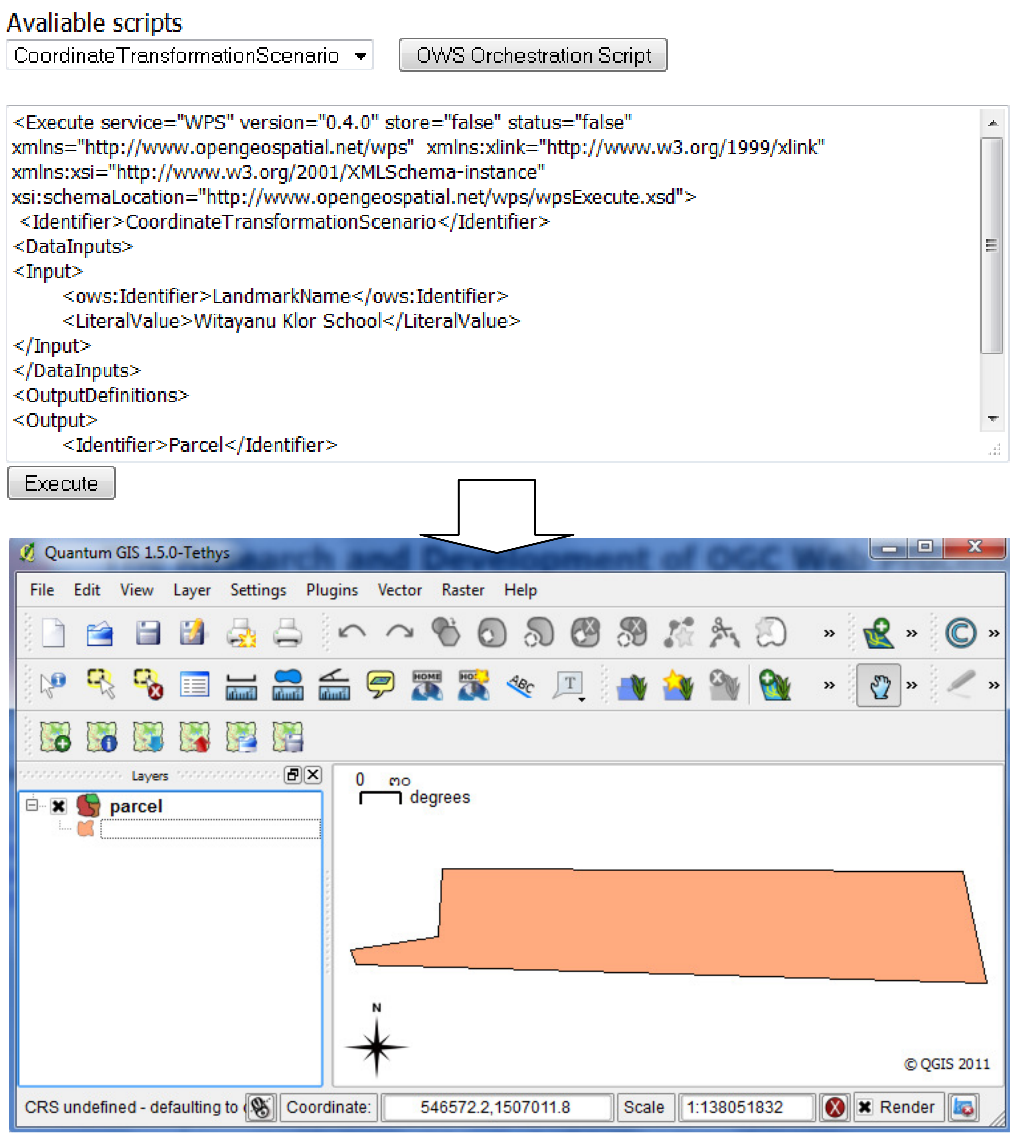
5.3.3 Orchestration Script of Coordinate Transformation Service

Orchestration script which derived from sequence diagram of coordinate transformation scenario can be demonstrated as following short hand script:

```
<cuwps:ProcessDefinition name=”CoordinateTransformationScenario”>
  <Variables>
    <Variable name="wpsExecuteRequest" portType="wps:Execute"/>
    <Variable name="getLandmarkByName" portType="wfs:GetFeature"/>
    <Variable name="responseLandmark" portType="wfs:FeatureCollection"/>
    <Variable name="transformLandmark" portType="soap:Request"/>
    <Variable name="transformedLandmark" portType="soap:Response"/>
    <Variable name="getParcel" portType="wfs:GetFeature"/>
    <Variable name="parcel" portType="wfs:FeatureCollection"/>
  </Variables>
  <Activities>
    <Sequence>
      <Receive operation="wps:Execute" outputVariable="wpsExecuteRequest"
/>
      <!—Assign landmark name to wfs:GetFeature-->
      <Assign name=”assignLandmarkName”>…</Assign>
      <!-- Invoke landmark location service -->
      <Invoke operation="wfs:GetFeature" inputVariable="getLandmarkByName"
        outputVariable="responseLandmark"
        port="http://localhost:8080/geoserver/wfs" />
      <Assign name=”assignLandmarkToTransformationRequest”>…</Assign>
      <!-- Invoke coordinate transformation service -->
      <Invoke operation="wps:Execute" inputVariable="transformLandmark"
        outputVariable="transformedLandmark"
        port="http://localhost:8080/axis2"/>
      <Assign name="assignLandmarkToParcelIntersection">…</Assign>
      <!-- Invoke parcel intersection -->
      <Invoke operation="wfs:GetFeature" inputVariable="getParcel"
        outputVariable="parcel"
        port="http://localhost:8080/geoserver/wfs"/>
      <!-- Response Result to User -->
      <Reply outputVariable="parcel" />
    </Sequence>
  </Activities>
</cuwps:ProcessDefinition>
```

5.3.4 Script Execution

In this scenario user sends one parameter as a name of landmark (Witayanu Klor School) which is used to find a land parcel. Below Figure is showing orchestration script and the result of an execution in QGIS software.


Avaliable scripts: CoordinateTransformationScenario

```
<Execute service="WPS" version="0.4.0" store="false" status="false"
xmlns="http://www.opengeospatial.net/wps"  xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengeospatial.net/wps/wpsExecute.xsd">
 <Identifier>CoordinateTransformationScenario</Identifier>
<DataInputs>
<Input>
       <ows:Identifier>LandmarkName</ows:Identifier>
       <LiteralValue>Witayanu Klor School</LiteralValue>
</Input>
</DataInputs>
<OutputDefinitions>
<Output>
       <Identifier>Parcel</Identifier>
```

CRS undefined - defaulting to | Coordinate: 546572.2,1507011.8 | Scale 1:138051832

Figure 31 Showing the result of Coordinate Transformation Scenario in QGIS.

5.3.5 Summary

In this scenario the heterogeneous geo-spatial services such as SOAP, WFS, and WPS web services are orchestrated. The result showing the services orchestration between W3C and OGC web service standard can be realized, and that proofed the orchestration language and engine can be used to orchestrate W3C and OGC web services.

5.4 Network Analysis Scenario

5.4.1 Scenario Overview

Route service facilitates travelers for determining travel route and navigation information. Route service demand is more increasing today due to rapid development of network information and mobile device. At the backend of route service the network analysis concepts such as shortest path, and travel sale man problem are implemented. Shortest path problem is applied to find a path between two vertices (places) in a road network. The sum of the weights (i.e., distance, time, and real-time traffic) of its constituent edges (such as intersection, and junction) is minimized.

Network analysis scenario, the third scenario, is set up for testing the research orchestration language and the development of orchestration engine. This scenario consists of two sub-service actors; Landmark Location Service and Network Analysis Service. Landmark location service provides location features which user shall define specific place by the landmark name. After two landmarks was specified, two landmark features will send to Network Analysis Service for finding a shortest path (route) between these two landmarks. Below Figures is showing the relevant actors in this scenario.

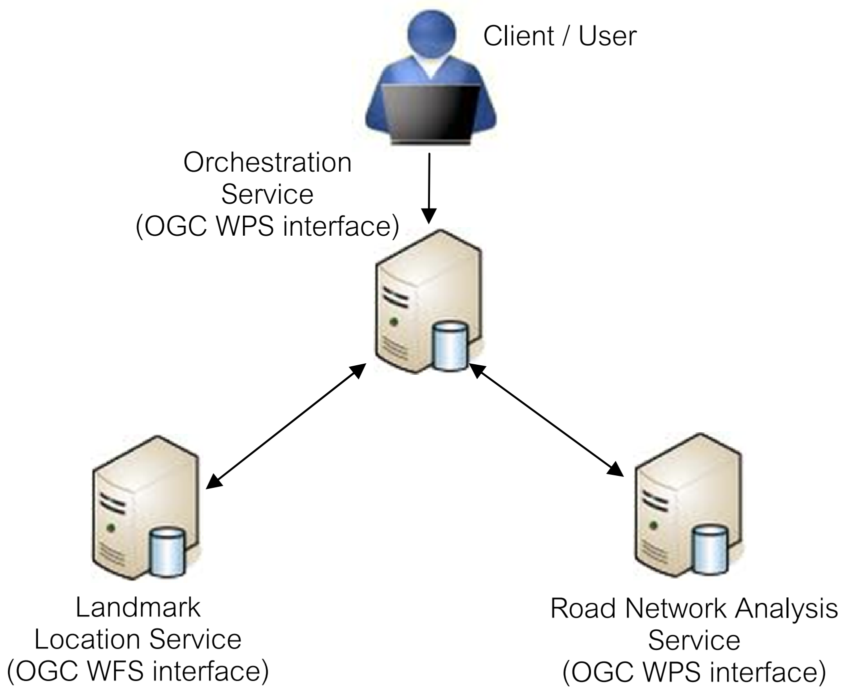

Client / User
Orchestration Service (OGC WPS interface)
Landmark Location Service (OGC WFS interface)
Road Network Analysis Service (OGC WPS interface)

Figure 32 Sub-service actors of Network Analysis Scenario.

5.4.2 Sequence Diagram of Network Analysis Scenario

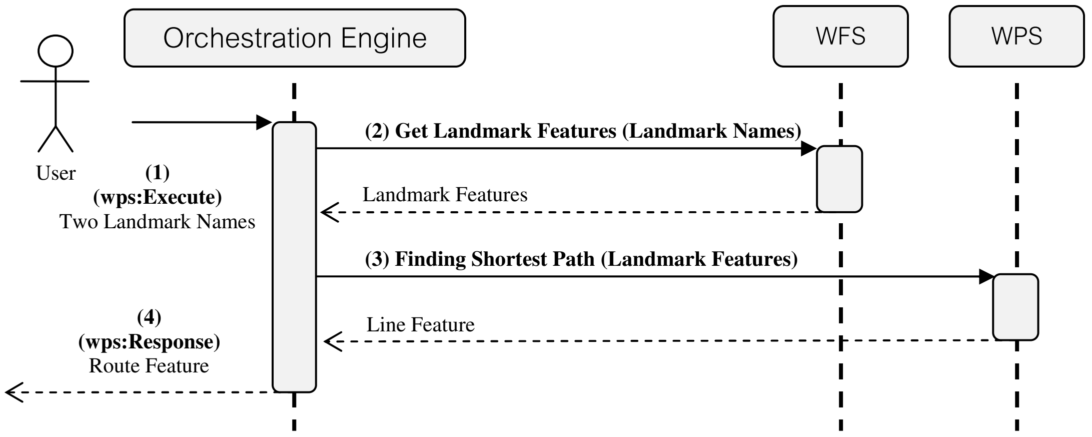

Orchestration Engine
WFS
WPS
User
(1) (wps:Execute) Two Landmark Names
(2) Get Landmark Features (Landmark Names)
Landmark Features
(3) Finding Shortest Path (Landmark Features)
Line Feature
(4) (wps:Response) Route Feature

Figure 33 Sequence diagram of Network Analysis Scenario.

From above Figure user have sends two landmark names in OGC WFS request form to WFS service. After engine retrieved landmark features from WFS service, the features will send to WPS service for processing shortest path. Then the result such as GML LineString is returned to client in the final.

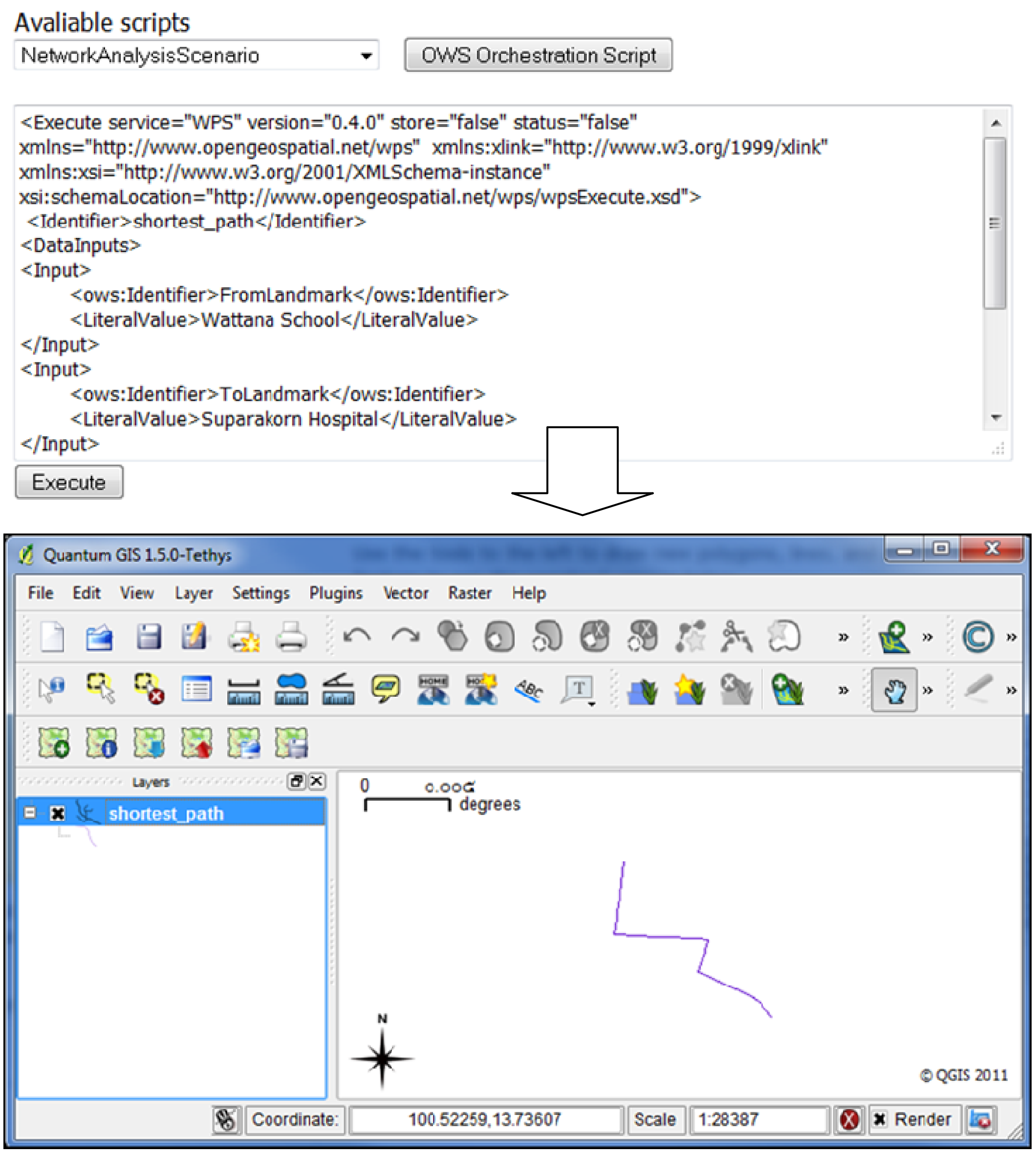
5.4.3 Orchestration Script of Network Analysis Scenario

Orchestration script which derived from sequence diagram of network analysis scenario can be demonstrated in below short hand script (full script displayed in Appendix D). In this script, additional tag <faultHandlers/> is added for testing an exception caching. This exception is located in a global scope of sequence activities. All invoke activities in <Sequence/> will be monitored the exceptions which is returned from service actor.

```
<cuwps:ProcessDefinition name=”NetworkAnalysisScenario”>
   <Variables>
      <Variable name="wpsExecuteRequest" portType="wps:Execute"/>
      <Variable name="getLandmarkByNames" portType="wfs:GetFeature"/>
      <Variable name="responseLandmarks" portType="wfs:FeatureCollection"/>
      <Variable name="getShortestPath" portType="wps:Execute"/>
      <Variable name="routeBetweenLandmarks"
                portType="wfs:FeatureCollection"/>
   </Variables>
   <Activities>
      <!-- Catch Service Exception -->
      <faultHandlers>
         <catch faultName=”ows:InvalidParameterValue”>…</catch>
         <catchAll>
            <exceptionText>Do not found any landmark name</exceptionText>
         </catchAll>
      </faultHandlers>
      <Sequence>
         <Receive operation="wps:Execute" outputVariable="wpsExecuteRequest"
/>
         <!—Assign landmark names to wfs:GetFeature-->
         <Assign name=”assignLandmarkNames”>…</Assign>
         <!-- Invoke landmark location service -->
         <Invoke operation="wfs:GetFeature" inputVariable="getLandmarkByNames"
            outputVariable="responseLandmarks"
            port="http://localhost:8080/geoserver/wfs" />
         <Assign name=”assignLandmarkToShortestPathRequest”>…</Assign>
         <!-- Invoke Road Network Analysis service -->
         <Invoke operation="wps:Execute" inputVariable="getShortestPath"
            outputVariable="routeBetweenLandmarks"
            port="http://localhost:8080/cuwps/network_analysis_service.jsp"/>
         <!-- Response Result to User -->
         <Reply outputVariable="routeBetweenLandmarks" />
      </Sequence>
   </Activities>
</cuwps:ProcessDefinition>
```

5.4.4 Script Execution

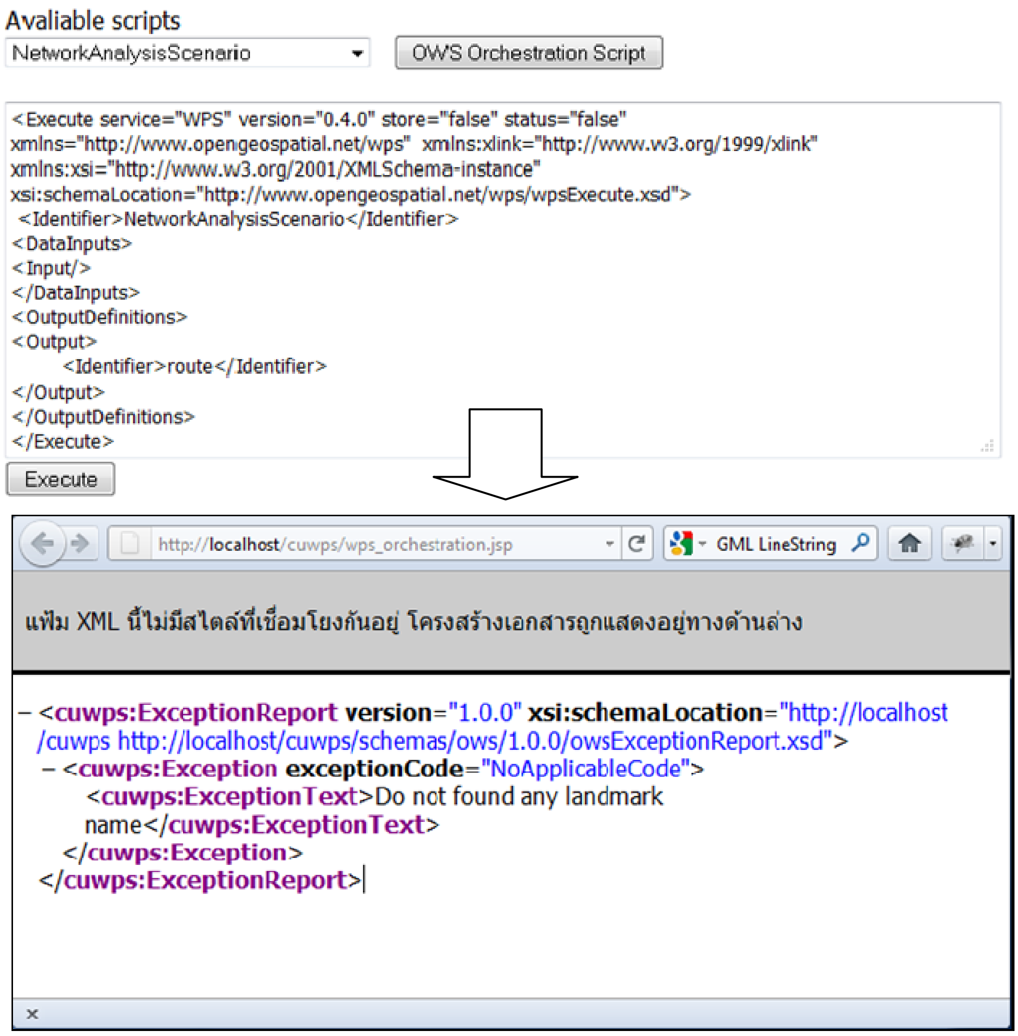
This scenario is testing on orchestration and exception caching. In Figure 34, the request message contains two landmark names which are from-to location names. Then orchestration service sends query statement (WFS request document in which contains Filter Encoding) to WFS service. After features were retrieved from WFS, orchestration engine sends these two features to Network Analysis service for processing a shortest path. The return feature (Route feature) of *shortest_path* operation is shown in Figure 34.



```
<Execute service="WPS" version="0.4.0" store="false" status="false"
xmlns="http://www.opengeospatial.net/wps"  xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengeospatial.net/wps/wpsExecute.xsd">
 <Identifier>shortest_path</Identifier>
<DataInputs>
<Input>
        <ows:Identifier>FromLandmark</ows:Identifier>
        <LiteralValue>Wattana School</LiteralValue>
</Input>
<Input>
        <ows:Identifier>ToLandmark</ows:Identifier>
        <LiteralValue>Suparakorn Hospital</LiteralValue>
</Input>
```

Figure 34 Execution request which sending two landmark names and the response feature (Route feature) of orchestration service.

Second execution message is used for testing exception caching mechanism. In Figure 35, orchestration service returns an exception which defines "NoApplicableCode" to client in order that request message have no landmark name specified.



```
<Execute service="WPS" version="0.4.0" store="false" status="false"
xmlns="http://www.opengeospatial.net/wps"  xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengeospatial.net/wps/wpsExecute.xsd">
 <Identifier>NetworkAnalysisScenario</Identifier>
<DataInputs>
<Input/>
</DataInputs>
<OutputDefinitions>
<Output>
      <Identifier>route</Identifier>
</Output>
</OutputDefinitions>
</Execute>
```

แฟ้ม XML นี้ไม่มีสไตล์ที่เชื่อมโยงกันอยู่ โครงสร้างเอกสารถูกแสดงอยู่ทางด้านล่าง

```
- <cuwps:ExceptionReport version="1.0.0" xsi:schemaLocation="http://localhost
  /cuwps http://localhost/cuwps/schemas/ows/1.0.0/owsExceptionReport.xsd">
  - <cuwps:Exception exceptionCode="NoApplicableCode">
      <cuwps:ExceptionText>Do not found any landmark
      name</cuwps:ExceptionText>
    </cuwps:Exception>
  </cuwps:ExceptionReport>
```

Figure 35 Execution request which unspecified landmark name and the exception which is responded from orchestration service.

5.4.5 Summary

In this scenario the exception mechanism was tested. The exception text is defined in <exceptionText/>. Caching definition which is defined before <Sequence/> tag is means that caching will monitor all activities which defined in <Sequence/>. The message which is returned from each sub-service will be checked in an occurrence of exception.

Chapter 6
Discussion and Outlook

This chapter concludes the research by summarizing research issues, answering the research questions (Section 1.6), identifying scientific contributions and exploring some possible direction for the future work.

The chapter is structured as follows: Section 6.1 gives research issues and a summary. Section 6.2 outlines the answers of the research questions. Section 6.3 explains the scientific contribution of the thesis. Section 6.4 discusses the limitations of the research. Finally in Section 6.5, the recommendations for future works are identified and addressed.

6.1 Issues Addressed in This Research

The research has presented methodology addressing a number of significant issues of a geo-spatial services orchestration. The significant issues can be described and summarized as follows:

- The constraints of using currently standard technology such as BPEL for orchestrating heterogeneous geo-spatial web services are addressed and identified (see; Section 3.1).
- The orchestration on different service interfaces standard (i.e., WFS, WPS, WMS, SOAP, REST, etc.) can be realized by defining those XML-based interface documents in a template mechanism. The data can flow from one service to another service by using this template mechanism (see; 4.2.3 Geo-spatial Message Control).
- REST web services which are using the interaction primitives found in the HTTP standard protocol for exchanging information can be orchestrated with standard W3C and also OGC web services by using HTTP access mechanism. The designed mechanism provides control definitions to manage HTTP protocol such as get or set HTTP header, handle HTTP payload as proprietary files (e.g., image and coverage) (see; 4.2.1 The Service Coordination Control)

– XML-based exception documents from different service platforms usually has a different structure can be handled by the exception handler. The exception handler provides users a mechanism to define a hierarchy of XML nodes to catch an exception code (using XPath). Thus, the variety of exception message structures can be accessed and managed through this mechanism. (see; 4.1.3 Caching of Faults)

6.2 Answer of Research Questions

6.2.1 What is necessity for an interface-independent of geo-spatial services orchestration aspect?

Web services published in the Internet provide a communication interface to clients in two common ways; HTTP operations and XML encoded document. In the REST web service such as RESTful, HTTP UPDATE / DELETE / PUT have more meaning to service itself than a transportation protocol. To support REST and OGC web services communication, an orchestration language must provides HTTP-protocol accessing mechanism for getting such file-based information and understanding REST web services. Thus, the interface-independent service coordination can be realized by providing the HTTP-protocol accessing mechanism (see, Section 3.1.1). Additionally, the message manipulation mechanism (see, Section 3.1.2) is helpful for a platform of an exchanging message between different message formats.

6.2.2 What are mandatory mechanisms of a geo-spatial services orchestration that realize a process definition (orchestration language) and facilitate a geo-processing flow?

An important ability for a platform-independent coordination is a capability of supporting interface-independent communication. Interfaces of the web services are ranging from a transport protocol (HTTP) to an application protocol (such as SOAP, WFS, WMS, WPS document). Service coordination, Message transformation, and Exception handling are three mandatory mechanisms that important for an orchestration language which allows users to define sub-processes on various service platforms. The different formats of geo-spatial data such as open formats (e.g., GML, KML, GeoJSON, etc.) and proprietary formats (e.g., Shapefile, AutoCAD DXF, MapInfo, ESRI grid etc.) can pass from one service to another service by these three mandatory mechanisms as demonstrated in Section 5.

6.2.3 How a designed language supports a binary data orchestration such as GML files?

Binary data which is distributed from such a service as OGC WMS or WCS is challenging assumption of standard web service architecture due to its isn't an XML document. For the binary data, an orchestration engine must not manipulate the data because binary data needs a proprietary application to manage and manipulate the content. From this reason, an orchestration engine can focuses only on passing that data from one service to another service. Thus, the proposed HTTP-protocol accessing mechanism as described in Section 3.1.1 is a solution which the testing in Section 5 proofed that the mechanism can be used to orchestrate a payload such as binary data.

6.3 Contributions

There are many efforts from researches to adapt such orchestration language as BPEL to more support on parallel developed technologies such as OGC web services, REST web service, and also on such protocol adaptation as WSDL 1.0 to WSDL 2.0. The challenging of those efforts is the rapid and parallel development on relevant web service technologies. In addition, a service description and contract in which described by XML-based document are usually changed responding to their development.

The low level orchestration language such as supporting all XML interface-document formats is the initial idea and also the key success of interface-independent services orchestration. Due to rapid development on web service technologies, a service description and a service contract which described in XML-based document frequently changes. To support those changes, the client which is a service must be re-built. This research proposes the geo-spatial services orchestration language in which the language design supports those changing. Thus, users have a change only an orchestration script for the change of a service description and a service contract.

In conclusion, this research contributes three important models for the interface-independent geo-spatial services orchestration as follows:

(1) Heterogeneous geo-spatial service coordination model (Section 3.2.1).

(2) Heterogeneous-payloads manipulation model (Section 3.2.2).

(3) Heterogeneous-exceptions handling model (Section 3.2.3).

These three models describe key mechanisms that can be used to develop a geo-spatial services orchestration language with support an interface-independent orchestration. Through the implementation and the testing of the designed orchestration language in this research, the three models are proofed that the heterogeneous geo-spatial services (such as W3C, OGC and REST web services) can be orchestrated by the orchestration script execution.

6.4 Discussion

This section discusses the limitations of the research regarding to the orchestration testing in the designed scenarios.

Due to a number of relevant services (sub-services), the testing scenarios are limited to a features processing. Nevertheless, some testing during the research found that a large data processing needs an asynchronous orchestration mechanism that does not implement in this thesis. A client timeout will occur when an orchestration process (such as coverage processing) is a long running in a synchronous orchestration pattern.

In addition, a type of feature which is processed by WPS has changed. For example, point features have change to polygon features after the buffer operation was executed. Thus, the processed features cannot validate with a tool such as GML validation in GeoTools library because the feature type had changed. The processing of the features at the next WPS in orchestration script will be failed. In the research test case we found that most WPSs usually need a features validation.

6.5 Recommendations for Future Work

A number of requirements that need to be identified and addressed were found during the research period. These requirements are out of the research scope but importance for the future work. Some notional of those requirements can be mentioned and open research questions as follows:

6.5.1 Improving the Workflow Management

For orchestration over huge and bulk information such as coverage file, asynchronous communication pattern such as pull model and push model (OGC, 2009) should be included to improve orchestration capability. Proprietary spatial data file such as satellite imagery requires time for downloading and processing bulk imagery data. Requester must uses a checking process status in a period of time than waiting for process successful (pull model). In addition, orchestration engine must also provides such mechanism as supporting sub-service callback when process successful (push model) if their processing time too long.

Transactional of geo-spatial services orchestration is another huge topic for including in workflow management research especially in transactional over distributed architecture. Invalid sub-process which is occurred during orchestration instance needs rollback mechanism for returning feature state back to the previous which before its execution. This feature is necessary for such system as risk management system, warning system, or alarming system.

6.5.2 Geo-spatial Orchestration Service

Metadata is important for any geo-spatial service clients. The data such as bounding box of service features provides clients to know which feature is useful for their application. Spatial reference system is another of important information for client use with its own data. Thus, several resources and operations which appeared in orchestration workflow will need explanation mechanism for explaining significant information about how to check the quality and capability of each resource and operation. This mechanism is necessary for used as an orchestration service description with providing client ability to understanding a pre-defined orchestration script. A suitable conceptual model of describing process activities which are appeared in orchestration workflow is not defined then the research is required.

REFERENCES

Alameh, N. (2003) 'Chaining Geographic Information Web Services', in *Internet Computing*, Newjurcy, USA: IEEE Computer Society.

Amirian, P. and Alesheikh, A. (2008) A Hybrid Architecture for Implementing Efficient Geospatial Web Services: Integrating.Net Remoting and Web Services Technologies, *Journal of Applied Sciences* 8: 730-742.

Bai, Y., Di, L. and Wei, Y. (2009) 'A taxonomy of geospatial services for global service discovery and interoperability', *Computer & Geoscience*, vol. 35, pp. 783-790.

Benatallah, B., Casati, F., Grigori, D., Nezhad, H.R.M. and Toumani, F. (2005) 'Developing Adapters for Web Services Integration', International Conference on Advanced Information Systems Engineering (CAiSE), Porto, Portugal.

Best, B.D., Halpin, P.N., Fujioka, E., Read, A.J. and Qian, S.S. (2007) 'Geospatial web services within a scientific workflow: Predicting marine mammal habitats in a dynamic environment', *Ecological Informatics*, pp. 210-223.

Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. and Orchard, D. (2004) *Web Services Architecture*, W3C.

Chang, J.F. (2006) *Business Process Management Systems: Strategy and Implementation, Chapter 7 Workflow Technology*, Taylor & Francis Group, LLC.

Chen, L., Xiujun, M., Guanhua, C., Yanfeng, S. and Xuebing, F. (2005) 'A Peer-to-Peer Architecture for Dynamic Executing GIS Web Service Composition', Geoscience and Remote Sensing Symbosium, 979-982.

Chunithipaisan, S. and Supavetch, S. , The Development of Web Processing Service Using Power of Spatial Database, Proceeding of International Conference on Emerging Trends in Engineering and Technology, 2009, Nagpur, India.

Di, L. (2004) 'Distributed Geospatial Information Services-Architectures, Standards, and Reserach Issues', in *International Archives of Photogrammetry, Remote Sensing, and Spatial Information Sciences*, Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.121.7836 [19 December 2010].

Einspanier, U., Lutz, M., Senkler, K., Simonis, I. and Sliwinski, A. (2003) 'Toward a Process Model for GI Service Composition', In GI-Tage (GI Days), Munster.

Fielding, R.T. (2002) *Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST)* [Online]:, Available from: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm [2011, May].

Fleuren, T. and Muller, P. (2008) 'BPEL Workflows Combining Standard OGC Web Services and Grid-enabled OGC Web Services', 34th Euromicro Conference on Software Engineering and Advanced Applications, Parma, Italy, 337-344.

Granell, C., Diaz, L. and Gould, M. (2010) 'Service-oriented applications for environmental models: Reusable geospatial services', *Environmental Modelling & Software*, pp. 182-198.

Ioup, E., Lin, B., Sample, J. and Shaw, K. (2008) 'Chapter 4: Geospatial Web Services: Bridging the Gap between OGC and Web Services', in Sample, J.T., Shaw, K., Tu, S. and Abdelguerfi, M. *Geospatial Services and Applications for the Internet*, New York: Springer Science+Business Media.

ISO19119 (2001) *Geographic Information - Services*, Open GIS Consortium.

Kadry, S. and Smaili, K. (2007) 'A Solutions for Authentication of Web Service Users', *Information Technology Journal*, vol. 6, no. 7, pp. 987-995.

Kiehle, C. (2006) 'Business logic for geoprocessing of distributed geodata', *Computers & Geosciences*, pp. 1746-1757.

Lake, R. and Farley, J. (2007) 'Infrastructure for the Geospatial Web', in Scharl, A. and Tochtermann, K. *The Geospatial Web*, London: Springer-Verlay London Limited.

Lee, C. and Percivall, G. (2008) *Standards-Based Computing Capabilities for Distributed Geospatial Applications*, IEEE Computer Society.

Lucchi, R., Millot, M. and Elfers, C. (2008) *Resource Oriented Architecture and REST*, European Communities.

Manolescu, D.A. (2004) *Patterns for Orchestration Environments*, Available: http://hillside.net/plop/2004/papers/dmanolescu0/PLoP2004_dmanolescu0_0.pdf.

Meng, X., Bian, F. and Xie, Y. (4-5 July 2009) 'Research and Realization of Geospatial Information Service Orchestration Based on BPEL', Internatianal Conference on

Environmental Science and Information Application Technology, Wuhan, China, 642-645.

OASIS (2006) *Reference Model for Service Oriented Architecture 1.0*.

OGC (2001) *The OpenGIS Abstract Specification Topic 12: OpenGIS Service Architecture Version 4.3*.

OGC (2004) *OWS-2 IH4DS Service Chaining IPR. OGC #06-178r1*.

OGC (2005) *The OGC Abstract Specification, Topic 0: Abstract Specification Overview*, 5th edition.

OGC (2008) *OGC Reference Model (OGC 08-062r4)*, Open Geospatial Consortium.

OGC (2008) *OGC Reference Model version 2.0 (OGC 08-062r4)*.

OGC (2009) *OGC OWS-6 Geoprocessing Workflow Architecture Engineering Report (OGC 09-053r5)*, Open Geospatial Consortium Inc.

Pautasso, C. (2009) 'RESTful Web service composition with BPEL for REST', *Data & Knowledge Engineering*, vol. 68, pp. 851-866.

Peltz, C. (2003) *Web Services Orchestration: A Review Of Emerging Technologies, Tools, and Standards*, Technical report, Hewlett Packard, Available: http://devresource.hp.com/drc/technical_white_papers/WSOrch/WSOrchestration.pdf [11 Sep 2010].

PERCIVALL, G. (2002) *ISO 19119 and OGC Service Architecture*.

Sancho-Jiménez, G., Béjar, R., Latre, M. and Muro-Medrano, P. (2008) 'A Method to Derivate SOAP Interfaces and WSDL Metadata from the OGC Web Processing Service Mandatory Interfaces', in *Advances in Conceptual Modeling – Challenges and Opportunities*, Springer Berlin / Heidelberg}.

Schaeffer, B. (2008) 'Towards a Transactional Web Processing Service (WPS-T)', GIDays, Muenster.

Stollberg, B. and Zipf, A. (2007) 'OGC Web Processing Service Interface for Web Service Orchestration Aggreating Geo-processing Services in a Bomb Threat Scenario', 7th International Symposium on Web and Wireless GIS, Cardiff, UK , 239-251.

Stollberg, B. and Zipf, A. (2008) 'Geoprocessing Services for Spatial Decision Support in the Domain of Housing Market Analyses Experiences from Applying the OGC Web

Processing Service Interface in Practice', AGILE International Conference on Geographic Information Science, University of Girona, Spain, 10.

Supavetch, S. and Chunithipaisan, S. (2011) 'The SQL-Based Geospatial Web Processing Service', *International Journal of Computer Information Systems and Industrial management Applicatoins*, vol. 3, pp. 119-126.

Tsalgatidou, A. and Pilioura, T. (2002) 'An Overview of Standards and Related Technology in Web Services', *Distributed and Parallel Databases*, pp. 135-162.

W3C (2004) *Web Services Architecture*.

Yue, P., Gong, J., Di, L., Yuan, J., Sun, L. and Wang, Q. (2009) 'GeoPW: Towards the Geospatial Processing Web', in *Web and Wireless Geographical Information Systems*, Springer Berlin / Heidelberg.

Zhao, P., Di, L., Yu, G., Yue, P., Wei, Y. and Yang, W. (2009) 'Semantic Web-based geospatial knowledge transformation', *Computers & Geosciences*, pp. 798-808.

Zhao, P., Yu, G. and Di, L. (2007) 'Geospatial Web Services', in Hilton, B.N. *Emerging Spatial Information Systems and Applications*, IDEA GROUP PUBLISHING.

APPENDICES

APPENDIX A
Geospatial Services Orchestration Language Schema

This appendix shows XML-schema of Geo-spatial Services Orchestration Language which is developed from the research proposed models (Section 3.2). The language is used to define processing steps for composing geo-processing workflow (described in Section 5.2.2).

```
<!-- Orchestration activities --.>
<xsd:element name="Sequence">
<xsd:complexType>
   <xsd:sequence>
      <xsd:element ref="FaultHandlersType" minOccurs="0" maxOccurs="1">
         <xsd:annotation>
            <xsd:document>
            This is globally catching exception which occurs during
orchestration
            </xsd:document>
         </xsd:annotation>
      </xsd:element>
      <xsd:element ref="Receive" minOccurs="0" maxOccurs="1" />
      <xsd:element ref="Invoke" minOccurs="0" maxOccurs="unbound" />
      <xsd:element ref="Reply" minOccurs="0" maxOccurs="1" />
   </xsd:sequence>
</xsd:complexType>
</xsd:element>

<xsd:element name="Receive">
<xsd:complexType>
   <xsd:attribute name="name" type="xsd:string" use="required"/>
   <xsd:attribute name="operation" type="xsd:string" use="required"/>
   <xsd:attribute name="outputVariable" type="xsd:string" use="required"/>
   <xsd:sequence>
      <xsd:element ref="HTTPAccessType" minOccurs="0" maxOccurs="1" />
      <xsd:element ref="FaultHandlersType" minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
</xsd:complexType>
</xsd:element>

<xsd:element name="Invoke">
<xsd:complexType>
   <xsd:attribute name="name" type="xsd:string" use="required"/>
   <xsd:attribute name="operation" type="xsd:string" use="required"/>
   <xsd:attribute name="inputVariable" type="xsd:string" use="option"/>
   <xsd:attribute name="outputVariable" type="xsd:string" use="required"/>
   <xsd:attribute name="port" type="anyURI" use="required">
      <xsd:simpleType>
         <xsd:restriction base="anyURI"/>
      </xsd:simpleType>
   </xsd:attribute>
   <xsd:attribute name="method" use="optional" default="GET">
      <xsd:simpleType>
```

```
         <xsd:restriction base="xsd:string">
            <xsd:enumeration value="GET" />
            <xsd:enumeration value="POST"/>
            <xsd:enumeration value="PUT"/>
            <xsd:enumeration value="DELETE"/>
            <xsd:enumeration value="UPDATE"/>
         </xsd:restriction>
      </xsd:simpleType>
   </xsd:attribute>
   <xsd:sequence>
      <xsd:element ref="HTTPAccessType" minOccurs="0" maxOccurs="1" />
      <xsd:element ref="FaultHandlersType" minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
</xsd:complexType>
</xsd:element>

<xsd:element name="Reply">
<xsd:complexType>
   <xsd:attribute name="name" type="xsd:string" use="option"/>
   <xsd:attribute name="outputVariable" type="xsd:string" use="required"/>
   <xsd:sequence>
      <xsd:element ref="HTTPAccessType" minOccurs="0" maxOccurs="1" />
      <xsd:element ref="FaultHandlersType" minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
</xsd:complexType>
</xsd:element>

<!-- HTTP access mechanism -->

<xsd:element name="HTTPAccess" type="HTTPAccessType"/>

<xsd:complexType name="HTTPAccessType">
   <xsd:choice>
      <xsd:element name="GetHeader">
         <xsd:annotation>
            <xs:documentation xml:lang="en">
            Use this function with Receive activity for getting header
attached
            information
            </xs:documentation>
         </xsd:annotation>
         <xsd:complexType>
            <xsd:sequence>
            <xsd:element name="Name" type="xsd:string" minOccurs="1"
               maxOccurs="1" />
            <xsd:element name="ToVariable" type="NCName" minOccurs="0"
               maxOccurs="1"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
      <xsd:element name="SetHeader">
         <xsd:annotation>
            <xs:documentation xml:lang="en">
            Use this function with Invoke activities for setting additional
information
            </xs:documentation>
         </xsd:annotation>
```

```
            <xsd:complexType>
               <xsd:element name="Name" type="xsd:string"/>
               <xsd:element name="Value" type="xsd:string"/>
            </xsd:complexType>
         <xsd:element>
         <xsd:element name="GetStatus">
            <xsd:annotation>
               <xs:documentation xml:lang="en">
               Use this function with Fault for caching transportation error

               </xs:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element name="SetStatus">
            <xsd:annotation>
               <xs:documentation xml:lang="en">
               Use this function with Reply activities
               </xs:documentation>
            </xsd:annotation>
            <xsd:simpleType>
               <xsd:restriction base="xsd:integer">
                  <xsd:length value="3" />
               </xsd:restriction>
            </xsd:simpleType>
         </xsd:element>
         <xsd:element name="GetPayloadStream">
            <xsd:annotation>
               <xs:documentation xml:lang="en">
               Use this function to get HTTP post body in a bytes stream
               </xs:documentation>
            </xsd:annotation>
         <xsd:element>

      </xsd:choice>
   </xsd:complexType>

   <!-- Variables -->
   <xsd:element name="Variable" type="VariableType/>
   <xsd:complexType name="VariableType">
      <xsd:attribute name="name" type="NCName" use="required"/>
      <xsd:attribute name="mimeType" type="QName" use="optional"/>
      <xsd:attribute name="portType" type="QName" use="required"/>
      <xsd:attribute name="handlingAsFile" type="xsd:boolean" default="false" >
         <xsd:annotation>
            <xsd:document>
            Handling the payload as XML or File
            </xsd:document>
         </xsd:annotation>
      </xsd:attribute>
      <xsd:any minOccurs="0">
         <xsd:annotation>
            <xsd:document>
            User defined XML document, which is a service API, as template
   document
            </xsd:document>
         </xsd:annotation>
      </xsd:any>
```

```
</xsd:complexType>

<xsd:element name="Assign" type="AssignType/>
<xsd:complexType name="AssignType">
   <xsd:attribute name="name" type="NCName" use="required"/>
   <xsd:choice>
      <xsd:element name="Copy">
         <xsd:sequence>
            <xsd:element ref="FromType"/>
            <xsd:element ref="ToType"/>
         </xsd:sequence>
      </xsd:element>
      <xsd:element name="Insert">
         <xsd:sequence>
            <xsd:element ref="FromType"/>
            <xsd:element ref="ToType"/>
         </xsd:sequence>
      </xsd:element>
      <xsd:element name="Delete">
         <xsd:element ref="FromType"  />
      </xsd:element>
      <xsd:element name="Filter">
         <xsd:sequence>
            <xsd:element ref="FromType"/>
            <xsd:element ref="ToType"/>
         </xsd:sequence>
      </xsd:element>
   </xsd:choice>
</xsd:complexType>

<xsd:element name="From" type="FormType">
<xsd:complexType>
   <xsd:attribute name="inputValue" type="xsd:string" use="required"/>
   <xsd:attribute name="byReference" type="xsd:boolean" use="optional"
      default="false"/>
   <xsd:choice minOccurs="1">
      <xsd:anyType/>
      <xsd:simpleType>
         <xsd:restriction base="xsd:string">
            <xsd:annotation>
               <xsd:document xml:lang="en">
               XPath expression statement for filter information form variable
               document value
               </xsd:document>
            </xsd:annotation>
         </xsd:restriction>
      </xsd:simpleType>
   </xsd:choice>
</xsd:complexType>
</xsd:element>

<xsd:element name="To" type="ToType">
<xsd:complexType>
   <xsd:attribute name="outputValue" type="xsd:string"/>
   <xsd:simpleType>
      <xsd:restriction base="xsd:string">
         <xsd:annotation>
```

```
          <xsd:document>
          Variable replacement marker
          </xsd:document>
        </xsd:annotation>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:complexType>
</xsd:element>
<xsd:element name="FaultHandlers" type="FaultHandlersType"/>
<xsd:element name="FaultHandler" type="FaultHandlerType"/>

<xsd:complexType name="FaultHandlersType">
   <xsd:element name="FaultHandler" ref="FaultHandlerType"
      minOccurs="1" maxOccurs="unbound"/>
</xsd:complexType>

<xsd:complexType name="FaultHandlerType">
<xsd:attribute name="faultName">
   <xsd:annotation>
      <xsd:documentation xml:lang="en">
      HTTP error number, string, code that represent the fault
      </xsd:documentation>
   </xsd:annotation>
</xsd:attribute>
<xsd:attribute name="queryLanguage">
   <xsd:simpleType>
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="xpath"/>
         <xsd:enumeration value="http_access"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="capture">
   <xsd:annotation>
      <xsd:documentation xml:lang="en">
      Query expression matches queryLanguage
      </xsd:documentation>
   </xsd:annotation>
</xsd:attribute>
<xsd:element ref="SequenceType" minOccurs="1"/>
<xsd:element name="Terminate" minOccurs="0" maxOccurs="1">
   <xsd:annotation>
      <xsd:documentation xml:leng="en">
      Terminating the whole process flow
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
</xsd:complexType>
```

APPENDIX B

Orchestration Script of Site Selection Scenario

This appendix displays a script called orchestration script of site selection scenario which is used to testing a designed language (Geo-spatial Services Orchestration Language). The details of site selection scenario are described in Section 0

```
<?xml version="1.0" encoding="UTF-8"?>
<cuwps:ProcessDefinition name="SiteSelection"
   xmlns="http://www.eng.chula.ac.th/cuwps"
   xmlns:cuwps="http://www.eng.chula.ac.th/cuwps"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wps="http://www.opengis.net/wps"
   xmlns:wfs="http://www.opengis.net/wfs">
   <Variables>
      <Variable name="wpsExecuteRequest" portType="wps:ExecuteType"/>
      <!-- GET HOSPITALS -->
      <Variable name="getHospital" portType="wfs:GetFeatureType">
         <wfs:GetFeature service="WFS" version="1.1.0" outputFormat="GML2"
            xmlns:cuwps="http://www.eng.chula.ac.th/cuwps"
            xmlns:wfs="http://www.opengis.net/wfs"
            xmlns:ogc="http://www.opengis.net/ogc"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.opengis.net/wfs
            http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">
               <wfs:Query typeName="cuwps:landmark">
                  <ogc:Filter>
                     <ogc:PropertyIsEqualTo>
                        <ogc:PropertyName>TYPE</ogc:PropertyName>
                        <ogc:Literal>5010</ogc:Literal>
                     </ogc:PropertyIsEqualTo>
                  </ogc:Filter>
               </wfs:Query>
         </wfs:GetFeature>
      </Variable>
      <Variable name="responseHospital"
portType="wfs:FeatureCollectionType"/>
      <Variable name="getBufferingHospital" portType="wps:ExecuteType">
         <wps:Execute version="1.0.0" service="WPS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.opengis.net/wps/1.0.0"
xmlns:wfs="http://www.opengis.net/wfs"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:gml="http://www.opengis.net/gml"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:wcs="http://www.opengis.net/wcs/1.1.1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://schemas.opengis.net/wps/1.0.0/wpsAll.xsd">
           <ows:Identifier>gt:BufferFeatureCollection</ows:Identifier>
           <wps:DataInputs>
            <wps:Input>
              <ows:Identifier>features</ows:Identifier>
              <wps:Data>
                  <wps:ComplexData mimeType="text/xml; subtype=wfs-
collection/1.0"><![CDATA[{$hospitalFeatures}]]></wps:ComplexData>
              </wps:Data>
            </wps:Input>
            <wps:Input>
              <ows:Identifier>buffer</ows:Identifier>
              <wps:Data>
               <wps:LiteralData>{$hospitalDistance}</wps:LiteralData>
              </wps:Data>
            </wps:Input>
           </wps:DataInputs>
           <wps:ResponseForm>
            <wps:RawDataOutput mimeType="text/xml; subtype=wfs-
collection/1.0">
              <ows:Identifier>result</ows:Identifier>
            </wps:RawDataOutput>
           </wps:ResponseForm>
         </wps:Execute>
      </Variable>
      <Variable name="bufferedHospital"
portType="wfs:FeatureCollectionType"/>
      <!-- GET SCHOOLS -->
      <Variable name="getSchool" portType="wfs:GetFeatureType">
         <wfs:GetFeature service="WFS" version="1.1.0" outputFormat="GML2"
            xmlns:cuwps="http://www.eng.chula.ac.th/cuwps"
            xmlns:wfs="http://www.opengis.net/wfs"
            xmlns:ogc="http://www.opengis.net/ogc"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.opengis.net/wfs
            http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">
               <wfs:Query typeName="cuwps:landmark">
                  <ogc:Filter>
                     <ogc:PropertyIsEqualTo>
                        <ogc:PropertyName>TYPE</ogc:PropertyName>
                        <ogc:Literal>4700</ogc:Literal>
                     </ogc:PropertyIsEqualTo>
                  </ogc:Filter>
               </wfs:Query>
         </wfs:GetFeature>
      </Variable>
      <Variable name="responseSchool" portType="wfs:FeatureCollectionType"/>
      <Variable name="getBufferingSchool" portType="wps:ExecuteType">
         <wps:Execute version="1.0.0" service="WPS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.opengis.net/wps/1.0.0"
xmlns:wfs="http://www.opengis.net/wfs"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:gml="http://www.opengis.net/gml"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:wcs="http://www.opengis.net/wcs/1.1.1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://schemas.opengis.net/wps/1.0.0/wpsAll.xsd">
           <ows:Identifier>gt:BufferFeatureCollection</ows:Identifier>
```

```
          <wps:DataInputs>
           <wps:Input>
             <ows:Identifier>features</ows:Identifier>
             <wps:Data>
                <wps:ComplexData mimeType="text/xml; subtype=wfs-
collection/1.0"><![CDATA[{$schoolFeatures}]]></wps:ComplexData>
             </wps:Data>
           </wps:Input>
           <wps:Input>
             <ows:Identifier>buffer</ows:Identifier>
             <wps:Data>
              <wps:LiteralData>{$schoolDistance}</wps:LiteralData>
             </wps:Data>
           </wps:Input>
          </wps:DataInputs>
          <wps:ResponseForm>
           <wps:RawDataOutput mimeType="text/xml; subtype=wfs-
collection/1.0">
             <ows:Identifier>result</ows:Identifier>
           </wps:RawDataOutput>
          </wps:ResponseForm>
        </wps:Execute>
     </Variable>
     <Variable name="bufferedSchool" portType="wfs:FeatureCollectionType"/>
     <!-- GET HOUSING_ESTATE -->
     <Variable name="getHousingEstate" portType="wfs:GetFeatureType">
        <wfs:GetFeature service="WFS" version="1.1.0" outputFormat="GML2"
           xmlns:cuwps="http://www.eng.chula.ac.th/cuwps"
           xmlns:wfs="http://www.opengis.net/wfs"
           xmlns:ogc="http://www.opengis.net/ogc"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://www.opengis.net/wfs
           http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">
              <wfs:Query typeName="cuwps:landmark">
                 <ogc:Filter>
                    <ogc:PropertyIsEqualTo>
                       <ogc:PropertyName>LOC_TYPE</ogc:PropertyName>
                       <ogc:Literal>HOUSING_ESTATE</ogc:Literal>
                    </ogc:PropertyIsEqualTo>
                 </ogc:Filter>
              </wfs:Query>
        </wfs:GetFeature>
     </Variable>
     <Variable name="responseHousingEstate"
portType="wfs:FeatureCollectionType"/>
     <Variable name="getBufferingHousingEstate" portType="wps:ExecuteType">
        <wps:Execute version="1.0.0" service="WPS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.opengis.net/wps/1.0.0"
xmlns:wfs="http://www.opengis.net/wfs"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:gml="http://www.opengis.net/gml"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:wcs="http://www.opengis.net/wcs/1.1.1"
xmlns:xlink="http://www.w3.org/1999/xlink"
```

```
xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://schemas.opengis.net/wps/1.0.0/wpsAll.xsd">
          <ows:Identifier>gt:BufferFeatureCollection</ows:Identifier>
          <wps:DataInputs>
           <wps:Input>
             <ows:Identifier>features</ows:Identifier>
             <wps:Data>
                <wps:ComplexData mimeType="text/xml; subtype=wfs-
collection/1.0"><![CDATA[{$housingestateFeatures}]]></wps:ComplexData>
             </wps:Data>
           </wps:Input>
           <wps:Input>
             <ows:Identifier>buffer</ows:Identifier>
             <wps:Data>
              <wps:LiteralData>{$housingestateDistance}</wps:LiteralData>
             </wps:Data>
           </wps:Input>
          </wps:DataInputs>
          <wps:ResponseForm>
           <wps:RawDataOutput mimeType="text/xml; subtype=wfs-
collection/1.0">
             <ows:Identifier>result</ows:Identifier>
           </wps:RawDataOutput>
          </wps:ResponseForm>
        </wps:Execute>
     </Variable>
     <Variable name="bufferedHousingEstate"
portType="wfs:FeatureCollectionType"/>
     <!-- HOSPITAL AND SCHOOL INTERSECT -->
     <Variable name="getIntersectHospitalSchool" portType="wps:ExecuteType">
        <Execute service="WPS" version="0.4.0" store="true" status="false"
xmlns="http://www.opengeospatial.net/wps"
            xmlns:ows="http://www.opengeospatial.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengeospatial.net/wps/wpsExecute.xsd">
           <ows:Identifier>Intersect</ows:Identifier>
           <DataInputs>
             <Input>
                <ows:Identifier>InputFirstFeatures</ows:Identifier>
                <ows:Title>First features</ows:Title>
                <wps:ComplexData mimeType="text/xml; subtype=wfs-
collection/1.0"><![CDATA[{$hospitalToBeIntersect}]]></wps:ComplexData>
             </Input>
             <Input>
                <ows:Identifier>InputSecondFeatures</ows:Identifier>
                <ows:Title>Second features</ows:Title>
                <wps:ComplexData mimeType="text/xml; subtype=wfs-
collection/1.0"><![CDATA[{$schoolToBeIntersect}]]></wps:ComplexData>
             </Input>
           </DataInputs>
           <OutputDefinitions>
             <Output>
                <ows:Identifier>IntersectedPolygon</ows:Identifier>
                <ows:Title>Area of school and hospital
intersection</ows:Title>
                <ows:Abstract></ows:Abstract>
            </Output>
          </OutputDefinitions>
        </Execute>
      </Variable>
      <Variable name="intersectedSchoolHospitalPolygon"
portType="wfs:FeatureCollectionType"/>
      <!-- HOUSING_ESTATE INTERSECT -->
      <Variable name="getIntersectHospitalSchoolHousingEstate"
portType="wps:ExecuteType">
        <Execute service="WPS" version="0.4.0" store="true" status="false"
xmlns="http://www.opengeospatial.net/wps"
            xmlns:ows="http://www.opengeospatial.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengeospatial.net/wps/wpsExecute.xsd">
           <ows:Identifier>Intersect</ows:Identifier>
          <DataInputs>
            <Input>
              <ows:Identifier>InputFirstFeatures</ows:Identifier>
              <ows:Title>First features</ows:Title>
              <wps:ComplexData mimeType="text/xml; subtype=wfs-
collection/1.0"><![CDATA[{$hospitalSchoolToBeIntersect}]]></wps:ComplexData>
            </Input>
            <Input>
              <ows:Identifier>InputSecondFeatures</ows:Identifier>
              <ows:Title>Second features</ows:Title>
              <wps:ComplexData mimeType="text/xml; subtype=wfs-
collection/1.0"><![CDATA[{$housingestateToBeIntersect}]]></wps:ComplexData>
            </Input>
          </DataInputs>
          <OutputDefinitions>
            <Output>
              <ows:Identifier>IntersectedPolygon</ows:Identifier>
              <ows:Title>Area of school and hospital
intersection</ows:Title>
              <ows:Abstract></ows:Abstract>
            </Output>
          </OutputDefinitions>
        </Execute>
      </Variable>
      <Variable name="intersectedSchoolHospitalHousingEstatePolygon"
portType="wfs:FeatureCollectionType"/>
      <Variable name="NullFeatures">Not found any features</Variable>
   </Variables>
   <!-- PROCESSING ACTIVITIES -->
   <Activities>
      <Sequence>
          <Receive operation="wps:Execute"
outputVariable="wpsExecuteRequest"/>
        <!-- ASSIGN PARAMETERS -->
        <Assign name="assignUserHospitalDistance">
          <Copy>
            <From
inputValue="wpsExecuteRequest">//Input[1]/LiteralValue</From>
            <To outputValue="getBufferingHospital">{$hospitalDistance}</To>
          </Copy>
        </Assign>
        <Assign name="assignUserSchoolDistance">
           <Copy>
              <From
inputValue="wpsExecuteRequest">//Input[2]/LiteralValue</From>
              <To outputValue="getBufferingSchool">{$schoolDistance}</To>
           </Copy>
        </Assign>
        <Assign name="assignUserHousingEstateDistance">
           <Copy>
              <From
inputValue="wpsExecuteRequest">//Input[3]/LiteralValue</From>
              <To
outputValue="getBufferingHousingEstate">{$housingestateDistance}</To>
           </Copy>
        </Assign>
        <!-- GET HOSPITAL FEATURES -->
        <Invoke  operation="wfs:GetFeature" inputVariable="getHospital"
outputVariable="responseHospital" port="http://localhost/geoserver/wfs"/>
        <Assign name="assignHospitalToBuffering">
           <Copy>
              <From inputValue="responseHospital"/>
              <To outputValue="getBufferingHospital">{$hospitalFeatures}</To>
           </Copy>
        </Assign>
        <Invoke  operation="wps:Execute" inputVariable="getBufferingHospital"
outputVariable="bufferedHospital" port="http://localhost/geoserver/wps"/>
        <!-- GET SCHOOL FEATURES -->
        <Invoke  operation="wfs:GetFeature" inputVariable="getSchool"
outputVariable="responseSchool" port="http://localhost/geoserver/wfs"/>
        <Assign name="assignSchoolToBuffering">
           <Copy>
              <From inputValue="responseSchool"/>
              <To outputValue="getBufferingSchool">{$schoolFeatures}</To>
           </Copy>
        </Assign>
        <Invoke  operation="wps:Execute" inputVariable="getBufferingSchool"
outputVariable="bufferedSchool" port="http://localhost/geoserver/wps"/>
        <!-- GET HOUSING_ESTATE FEATURES -->
        <Invoke  operation="wfs:GetFeature" inputVariable="getHousingEstate"
outputVariable="responseHousingEstate"
port="http://localhost/geoserver/wfs"/>
        <Assign name="assignHousingEstateToBuffering">
           <Copy>
              <From inputValue="responseHousingEstate"/>
              <To
outputValue="getBufferingHousingEstate">{$housingestateFeatures}</To>
           </Copy>
        </Assign>
        <Invoke  operation="wps:Execute"
inputVariable="getBufferingHousingEstate"
outputVariable="bufferedHousingEstate"
port="http://localhost/geoserver/wps"/>
        <!--ASSIGN HOSPITAL AND SCHOOL TO INTERSECT-->
        <Assign name="assignHospitalToIntersect">
           <Copy>
              <From inputValue="bufferedHospital"/>
```

```
            <To
outputValue="getIntersectHospitalSchool">{$hospitalToBeIntersect}</To>
          </Copy>
        </Assign>
        <Assign name="assignSchoolToIntersect">
          <Copy>
            <From inputValue="bufferedSchool"/>
            <To
outputValue="getIntersectHospitalSchool">{$schoolToBeIntersect}</To>
          </Copy>
        </Assign>
        <Invoke  operation="wps:Execute"
inputVariable="getIntersectHospitalSchool"
outputVariable="intersectedSchoolHospitalPolygon"
port="http://localhost/cuwps/wps_intersect.jsp"/>
        <!--ASSIGN HOSPITAL_SCHOOL AND HOUSING_ESTATE TO INTERSECT-->
        <Assign name="assignHospitalSchoolToIntersect">
          <Copy>
            <From inputValue="intersectedSchoolHospitalPolygon"/>
            <To
outputValue="getIntersectHospitalSchoolHousingEstate">{$hospitalSchoolToBeIn
tersect}</To>
          </Copy>
        </Assign>
        <Assign name="assignHousingEstateToIntersect">
          <Copy>
            <From inputValue="bufferedHousingEstate"/>
            <To
outputValue="getIntersectHospitalSchoolHousingEstate">{$housingestateToBeInt
ersect}</To>
          </Copy>
        </Assign>
        <Invoke  operation="wps:Execute"
inputVariable="getIntersectHospitalSchoolHousingEstate"
outputVariable="intersectedSchoolHospitalHousingEstatePolygon"
port="http://localhost/cuwps/wps_intersect.jsp"/>
        <!-- REPLY TO USER -->
        <Reply outputVariable="intersectedSchoolHospitalHousingEstatePolygon"
/>
      </Sequence>
   </Activities>
</cuwps:ProcessDefinition>
```

APPENDIX C

Orchestration Script of Coordinate Transformation Scenario

This appendix displays a script called orchestration script of coordinate transformation scenario which is used to testing a designed language (Geo-spatial Services Orchestration Language). The details of coordinate transformation scenario are described in Section 5.3

```
<?xml version="1.0" encoding="UTF-8"?><cuwps:ProcessDefinition
   xmlns="http://www.eng.chula.ac.th/cuwps"
   xmlns:cuwps="http://www.eng.chula.ac.th/cuwps"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wps="http://www.opengis.net/wps"
   xmlns:wfs="http://www.opengis.net/wfs">
   <Variables>
      <Variable name="wpsExecuteRequest" portType="wps:ExecuteType"/>
      <Variable name="getLand" portType="wfs:GetFeatureType">
         <wfs:GetFeature service="WFS" version="1.1.0"  outputFormat="GML2"
            xmlns:topp="http://www.openplans.org/topp"
            xmlns:wfs="http://www.opengis.net/wfs"
            xmlns:ogc="http://www.opengis.net/ogc"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.opengis.net/wfs
            http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">
               <wfs:Query typeName="cuwps:bma_admin_poly">
                  <ogc:Filter>
                     <ogc:PropertyIsEqualTo>
                        <ogc:PropertyName>ADMIN_ID</ogc:PropertyName>
                        <ogc:Literal>1</ogc:Literal>
                     </ogc:PropertyIsEqualTo>
                  </ogc:Filter>
               </wfs:Query>
         </wfs:GetFeature>
      </Variable>
      <Variable name="responseLand" portType="wfs:FeatureCollectionType" />
      <Variable name="transformFeature" portType="soap:Envelope">
         <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
            soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:xsi="http://www.w3.org/1999/XMLSchema-instnace"
            xmlns:xsd="http://www.w3.org/1999/XMLSchema">
            <soap:Body>
               <ns:transform
xmlns:ns="http://ws.apache.org/axis2"><![CDATA[http://localhost/geoserver/ow
s?service=WFS&version=1.0.0&request=GetFeature&typeName=cuwps:point_coordina
te_transform&maxFeatures=50&outputFormat=GML2]]></ns:transform>
            </soap:Body>
         </soap:Envelope>
      </Variable>
      <Variable name="responseSoap" portType="soap:Envelope" />
      <Variable name="getIntersectOperation" portType="wps:ExecuteType">
         <Execute service="WPS" version="0.4.0" store="true" status="false"
xmlns="http://www.opengeospatial.net/wps"
```

```
            xmlns:ows="http://www.opengeospatial.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengeospatial.net/wps/wpsExecute.xsd">
           <ows:Identifier>Intersect</ows:Identifier>
           <DataInputs>
              <Input>
                 <ows:Identifier>InputFirstFeatures</ows:Identifier>
                 <ows:Title>First features</ows:Title>
                 <ComplexValueReference reference="{$feature01}" />
              </Input>
              <Input>
                 <ows:Identifier>InputSecondFeatures</ows:Identifier>
                 <ows:Title>Second features</ows:Title>
                  <wps:Data>
                    <wps:ComplexData mimeType="text/xml; subtype=wfs-
collection/1.0"><![CDATA[{$feature02}]]></wps:ComplexData>
                 </wps:Data>
              </Input>
           </DataInputs>
           <OutputDefinitions>
              <Output>
                 <ows:Identifier>IntersectedPolygon</ows:Identifier>
                 <ows:Title>Intersect polygon</ows:Title>
                 <ows:Abstract></ows:Abstract>
              </Output>
           </OutputDefinitions>
        </Execute>
     </Variable>
     <Variable name="outputFeature" portType="wfs:FeatureCollection" />
  </Variables>
  <Activities>
     <Sequence>
        <Receive operation="wps:Execute" outputVariable="wpsExecuteRequest"/>
        <Invoke  operation="wfs:GetFeature" inputVariable="getLand"
outputVariable="responseLand" port="http://localhost/geoserver/wfs"/>
        <Invoke  operation="SOAPAction" inputVariable="transformFeature"
outputVariable="responseSoap"
port="http://localhost/axis2/services/CoordinateTransformation">

  <HTTPAccess><SetHeader><Name>SOAPAction</Name><Value>transform</Value></Se
tHeader></HTTPAccess>
        </Invoke>
        <Assign name="assignTransformedPoint">
           <Copy>
              <From
inputValue="responseSoap">//transformResponse[1]/return</From>
              <To outputValue="getIntersectOperation">{$feature01}</To>
           </Copy>
        </Assign>
        <Assign name="assignLand">
           <Copy>
              <From inputValue="responseLand"/>
              <To outputValue="getIntersectOperation">{$feature02}</To>
           </Copy>
        </Assign>
        <Invoke  operation="wps:Execute"
inputVariable="getIntersectOperation" outputVariable="outputFeature"
port="http://localhost/cuwps/wps_intersect.jsp"/>
        <Reply outputVariable="outputFeature"/>

     </Sequence>
   </Activities>
</cuwps:ProcessDefinition>
```

APPENDIX D
Orchestration Script of Network Analysis Scenario

This appendix displays a script called orchestration script of network analysis scenario which is used to testing a designed language (Geo-spatial Services Orchestration Language). The details of network analysis scenario are described in Section 5.4

```
<?xml version="1.0" encoding="UTF-8"?><cuwps:ProcessDefinition
   xmlns="http://www.eng.chula.ac.th/cuwps"
   xmlns:cuwps="http://www.eng.chula.ac.th/cuwps"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:wps="http://www.opengis.net/wps"
   xmlns:wfs="http://www.opengis.net/wfs">
   <Variables>
      <Variable name="wpsExecuteRequest" portType="wps:ExecuteType"/>
      <Variable name="getFromPoint" portType="wfs:GetFeatureType">
         <wfs:GetFeature service="WFS" version="1.1.0"  outputFormat="GML2"
            xmlns:topp="http://www.openplans.org/topp"
            xmlns:wfs="http://www.opengis.net/wfs"
            xmlns:ogc="http://www.opengis.net/ogc"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.opengis.net/wfs
            http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">
               <wfs:Query typeName="cuwps:landmark">
                  <ogc:Filter>
                     <ogc:PropertyIsEqualTo>
                        <ogc:PropertyName>LANDMARK_I</ogc:PropertyName>
                        <ogc:Literal>1220</ogc:Literal>
                     </ogc:PropertyIsEqualTo>
                  </ogc:Filter>
               </wfs:Query>
         </wfs:GetFeature>
      </Variable>
      <Variable name="responseFromPoint"
portType="wfs:FeatureCollectionType"/>
      <Variable name="getToPoint" portType="wfs:GetFeatureType">
         <wfs:GetFeature service="WFS" version="1.1.0"  outputFormat="GML2"
            xmlns:topp="http://www.openplans.org/topp"
            xmlns:wfs="http://www.opengis.net/wfs"
            xmlns:ogc="http://www.opengis.net/ogc"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.opengis.net/wfs
            http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">
               <wfs:Query typeName="cuwps:landmark">
                  <ogc:Filter>
                     <ogc:PropertyIsEqualTo>
                        <ogc:PropertyName>LANDMARK_I</ogc:PropertyName>
                        <ogc:Literal>1321</ogc:Literal>
                     </ogc:PropertyIsEqualTo>
                  </ogc:Filter>
               </wfs:Query>
         </wfs:GetFeature>
      </Variable>
      <Variable name="responseToPoint" portType="wfs:FeatureCollectionType"/>
      <!--WPS NETWORK ANALYSIS-->
      <Variable name="getRoute" portType="wps:ExecuteType">
         <Execute service="WPS" version="0.4.0" store="true" status="false"
xmlns="http://www.opengeospatial.net/wps"
            xmlns:ows="http://www.opengeospatial.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengeospatial.net/wps/wpsExecute.xsd">
           <ows:Identifier>NetworkAnalysisScenario</ows:Identifier>
           <DataInputs>
              <Input>
                 <ows:Identifier>FromNode</ows:Identifier>
                 <ows:Title>First feature</ows:Title>
                 <wps:Data>
                    <wps:ComplexData mimeType="text/xml; subtype=wfs-
collection/1.0"><![CDATA[{$feature01}]]></wps:ComplexData>
                 </wps:Data>
              </Input>
              <Input>
                 <ows:Identifier>ToNode</ows:Identifier>
                 <ows:Title>Second feature</ows:Title>
                 <wps:Data>
                    <wps:ComplexData mimeType="text/xml; subtype=wfs-
collection/1.0"><![CDATA[{$feature02}]]></wps:ComplexData>
                 </wps:Data>
              </Input>
           </DataInputs>
           <OutputDefinitions>
              <Output>
                 <ows:Identifier>Route</ows:Identifier>
                 <ows:Title>LineString</ows:Title>
                 <ows:Abstract></ows:Abstract>
              </Output>
           </OutputDefinitions>
         </Execute>
      </Variable>

      <Variable name="responseRoute" portType="wfs:FeatureCollectionType"/>
   </Variables>
   <Activities>
      <Sequence>
         <Receive operation="wps:Execute" outputVariable="wpsExecuteRequest"/>
         <Invoke  operation="wfs:GetFeature" inputVariable="getFromPoint"
outputVariable="responseFromPoint" port="http://localhost/geoserver/wfs"/>
         <Invoke  operation="wfs:GetFeature" inputVariable="getToPoint"
outputVariable="responseToPoint" port="http://localhost/geoserver/wfs"/>
         <Assign name="assignFromFeature">
           <Copy>
              <From inputValue="responseFromPoint"/>
              <To outputValue="getRoute">{$feature01}</To>
           </Copy>
         </Assign>
         <Assign name="assignToFeature">
           <Copy>
```

```
             <From inputValue="responseToPoint"/>
             <To outputValue="getRoute">{$feature02}</To>
           </Copy>
        </Assign>
        <Invoke  operation="wps:Execute" inputVariable="getRoute"
outputVariable="responseRoute"
port="http://localhost/cuwps/wps_network_analysis.jsp"/>
        <Reply outputVariable="responseRoute"/>
     </Sequence>
  </Activities>
</cuwps:ProcessDefinition>
```

BIOGRAPHY

Major Soravis Supavetch is currently attached to Geodesy and Geophysics Division in Royal Thai Survey Department since 1998. He was born in Bangkok in 1975. He successfully completed the five-year course of academic and military science studies and was awarded the degree of Bachelor of Science in Survey Engineering in 1998 from Chulachomklao Royal Military Academy. He also completed the Degree Master of Science (Forestry) from Kasetsart University in 2004.