Web Service Implementation Methodology

Public Review Draft 1.0, 05 September 2005

Document identifier:

Location:

Editors:
Eng Wah LEE, Singapore Institute of Manufacturing Technology
<ewlee@SIMTech.a-star.edu.sg>
Marc HAINES, individual <mhaines@uwm.edu>

Contributors:
Lai Peng CHAN, Singapore Institute of Manufacturing Technology
<lchan@SIMTech.a-star.edu.sg>
Chai Hong ANG, Singapore Institute of Manufacturing Technology
<chang@SIMTech.a-star.edu.sg>
Puay Siew TAN, Singapore Institute of Manufacturing Technology
<pstan@SIMTech.a-star.edu.sg>
Han Boon LEE, Singapore Institute of Manufacturing Technology
<hblee@SIMTech.a-star.edu.sg>
Yushi CHENG, Singapore Institute of Manufacturing Technology
<ycheng@SIMTech.a-star.edu.sg>
Xingjian XU, Singapore Institute of Manufacturing Technology
<xju@SIMTech.a-star.edu.sg>
Zunliang YIN, Singapore Institute of Manufacturing Technology
<zyin@SIMTech.a-star.edu.sg>

Abstract:
This document specifies Web Service specific activities in a Web Service Implementation Methodology and illustrates the approach to incorporate these activities into an existing agile software development methodology.

Status:
Committee members should send comments on this specification to the fwsi-imsc@lists.oasis-open.org list. Others should subscribe to and send comments to the fwsi-comment@lists.oasis-open.org list. To subscribe, send an email message to fwsi-comment-request@lists.oasis-open.org with the word “subscribe” as the body of the message.
For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the FWSI TC web page (http://www.oasis-open.org/committees/fwsi/).
# Table of Contents

1. Introduction .......................................................................................................................... 8
  1.1 Purpose ............................................................................................................................... 8
  1.2 Target Audience ................................................................................................................ 8
  1.3 Scope ................................................................................................................................ 8
    1.3.1 Not in Scope .................................................................................................................. 8
2. Implementation Methodology Overview .............................................................................. 10
  2.1 Objective ............................................................................................................................ 10
  2.2 Web Service Implementation Lifecycle ............................................................................ 10
  2.3 Phase .................................................................................................................................. 10
    2.3.1 Requirements Phase ..................................................................................................... 11
    2.3.2 Analysis Phase .............................................................................................................. 11
    2.3.3 Design Phase ............................................................................................................... 12
    2.3.4 Coding Phase .............................................................................................................. 12
    2.3.5 Test Phase .................................................................................................................... 12
    2.3.6 Deployment Phase ....................................................................................................... 12
  2.4 Role .................................................................................................................................... 13
  2.5 Glossary .............................................................................................................................. 13
3. Web Service Implementation Methodology .......................................................................... 14
  3.1 Overview ............................................................................................................................ 15
  3.2 Requirements Phase .......................................................................................................... 16
    3.2.1 Activity: Determine the need for Web Service ............................................................... 16
    3.2.1.1 Tasks .......................................................................................................................... 16
    3.2.1.2 Roles .......................................................................................................................... 16
    3.2.1.3 Artifacts ..................................................................................................................... 16
    3.2.2 Activity: Elicit Web Service requirements ................................................................... 16
    3.2.2.1 Tasks .......................................................................................................................... 16
    3.2.2.2 Roles .......................................................................................................................... 16
    3.2.2.3 Artifacts ..................................................................................................................... 16
    3.2.3 Activity: Manage the Web Service requirements ......................................................... 17
    3.2.3.1 Tasks .......................................................................................................................... 17
    3.2.3.2 Roles .......................................................................................................................... 17
    3.2.3.3 Artifacts ..................................................................................................................... 17
    3.2.4 Activity: Model the usage scenarios ............................................................................. 17
    3.2.4.1 Tasks .......................................................................................................................... 17
    3.2.4.2 Roles .......................................................................................................................... 17
    3.2.4.3 Artifacts ..................................................................................................................... 17
    3.2.5 Activity: Prepare Test Cases for User Acceptance Test (UAT) and System Test ........ 17
    3.2.5.1 Tasks .......................................................................................................................... 17
    3.2.5.2 Roles .......................................................................................................................... 17
    3.2.5.3 Artifacts ..................................................................................................................... 17
  3.3 Analysis Phase .................................................................................................................... 18
    3.3.1 Activity: Select a technology platform as implementation framework ...................... 18
    3.3.1.1 Tasks .......................................................................................................................... 18
    3.3.1.2 Roles .......................................................................................................................... 18
    3.3.1.3 Artifacts ..................................................................................................................... 19
  3.3.2 Activity: Define a candidate architecture for the Web Service ................................... 19
3.3.3 Activity: Decide on the granularity of the Web Service ........................................... 19
94 3.3.3.1 Tasks .................................................................................................................... 19
95 3.3.3.2 Roles .................................................................................................................... 19
96 3.3.3.3 Artifacts................................................................................................................ 19
97 3.3.3.4 Activity: Identify reusable Web Services ............................................................ 20
98 3.3.3.4.1 Tasks ................................................................................................................ 20
99 3.3.3.4.2 Roles ................................................................................................................ 20
100 3.3.3.4.3 Artifacts......................................................................................................... 20
101 3.3.4 Activity: Identify service interface for new Web Services ..................................... 20
102 3.3.4.1 Tasks ................................................................................................................ 20
103 3.3.4.2 Roles ................................................................................................................ 20
104 3.3.4.3 Artifacts.......................................................................................................... 20
105 3.3.5 Activity: Prepare Test Cases for Performance Test .............................................. 20
106 3.3.5.1 Task .................................................................................................................. 20
107 3.3.5.2 Roles ................................................................................................................ 20
108 3.3.5.3 Artifacts......................................................................................................... 20
109 3.3.6 Activity: Prepare Test Cases for Integration / Interoperability Test ..................... 20
110 3.3.6.1 Task .................................................................................................................. 20
111 3.3.6.2 Roles ................................................................................................................ 20
112 3.3.6.3 Artifacts......................................................................................................... 20
113 3.3.7 Activity: Prepare Test Cases for Functional Test ............................................... 20
114 3.3.7.1 Task .................................................................................................................. 20
115 3.3.7.2 Roles ................................................................................................................ 20
116 3.3.7.3 Artifacts......................................................................................................... 20
117 3.3.8 Activity: Testbed preparation .............................................................................. 20
118 3.3.8.1 Task .................................................................................................................. 20
119 3.3.8.2 Roles ................................................................................................................ 20
120 3.3.8.3 Artifacts......................................................................................................... 20
121 3.4 Design Phase ........................................................................................................... 23
122 3.4.1 Activity: Transform signatures of reusable Web Services .................................... 23
123 3.4.1.1 Tasks ................................................................................................................ 23
124 3.4.1.2 Roles ................................................................................................................ 23
125 3.4.1.3 Artifacts......................................................................................................... 23
126 3.4.2 Activity: Refine service interface of the new Web Service ................................ 23
127 3.4.2.1 Tasks ................................................................................................................ 23
128 3.4.2.2 Roles ................................................................................................................ 23
129 3.4.2.3 Artifacts......................................................................................................... 23
130 3.4.3 Activity: Design Web Service ............................................................................ 23
131 3.4.3.1 Tasks ................................................................................................................ 23
132 3.4.3.2 Roles ................................................................................................................ 23
133 3.4.3.3 Artifacts......................................................................................................... 23
134 3.4.4 Activity: Refine Test Cases for Functional Test ................................................... 24
135 3.4.4.1 Task .................................................................................................................. 24
136 3.4.4.2 Roles ................................................................................................................ 24
137 3.4.4.3 Artifacts......................................................................................................... 24
138 3.5 Coding Phase .......................................................................................................... 24
139 3.5.1 Activity: Construct Web Service code ................................................................. 24
140 3.5.1.1 Tasks ................................................................................................................ 24
List of Figures

196  Figure 1: Web Service Implementation Lifecycle ................................................................. 11
197  Figure 2: The “V” Model incorporates the Web Services specific Interoperability test......... 15
198  Figure 3: Relationship between phase, activities, tasks, roles and artifacts ...................... 15
### List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Table 1: Mapping between phases and roles assigned</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>Table 2: Overview of activities, tasks, roles and artifacts in the Requirements Phase</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>Table 3: Overview of activities, tasks, roles and artifacts in the Analysis Phase</td>
<td>22</td>
</tr>
<tr>
<td>4</td>
<td>Table 4: Overview of activities, tasks, roles and artifacts in the Design Phase</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>Table 5: Overview of activities, tasks, roles and artifacts in the Coding Phase</td>
<td>26</td>
</tr>
<tr>
<td>6</td>
<td>Table 6: Overview of activities, tasks, roles and artifacts in the Test Phase</td>
<td>29</td>
</tr>
<tr>
<td>7</td>
<td>Table 7: Overview of activities, tasks, roles and artifacts in the Deployment Phase</td>
<td>31</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Purpose

The purpose of this document is to define a practical and extensible Web Service Implementation Methodology that can be used as a reference for Web Services development and deployment. This document is a consolidation of the best practices by Web Services practitioners and aims to improve the Web Services implementation process through the formalization of a Web Service implementation lifecycle and defining Web Service specific activities and artifacts.

This document should be used in conjunction with the Functional Elements specifications to govern the approach by which the Functional Elements are implemented.

1.2 Target Audience

The target audiences are likely to be:

- Project Managers
  This document provides a formal methodology for Web Services implementation, which can be used for management and control.
- Software Architects/Designers/Developers/Testers
  This document identifies activities that are repeatable and which can be abide by, so as to ensure the quality of the software produced.

1.3 Scope

This document focuses Web Service specific activities, artifacts, roles and responsibilities that can be incorporated into an existing agile software development methodology (e.g. RUP, Extreme Programming, Feature Driven Development etc). For a few common agile methodologies the technical committee is preparing examples that show in detail how the generic activities, artifacts, roles, and responsibilities described in this document can be incorporated and used in a given methodology. These case examples are provided in separate documents that will be published along with this document when they become available. Currently the technical committee is preparing cases for RUP and Extreme Programming (XP).

1.3.1 Not in Scope

This document does not define yet another novel software development methodology. Instead, the Web Service implementation methodology highlights important features in the context of Web Services. The elements of the Web Service implementation methodology are based on existing agile software methodology and extend it by incorporating Web Service specific activities.

Also, it is not in the scope of this document to specifically address how each of these software development methodologies should be tailored to incorporate Web Service specific parts. Examples are provided only to illustrate just one possible way of tailoring a specific agile development methodology for Web Service implementation.

---

1 The Functional Elements are to be specified as components, which are to be exposed as Web Services where appropriate.
This document does not intend to define a new software development methodology. Instead, the Web Service Implementation Methodology leverages on an existing agile software methodology and extend it by incorporating the Web Services specific activities.

This document also does not cover the detailed description or explanation of any of the existing agile software development methodology nor does it recommend one particular agile software development methodology over another.
2 Implementation Methodology Overview

2.1 Objective

The Web Service Implementation Methodology defines a systematic approach to Web Service development by leveraging on an agile software development methodology and extending that methodology by specifying the Web Services specific activities and the corresponding roles and work-products that are produced in the process.

This methodology will define a set of common practices that create a method-independent framework, which can be applied by most software teams for developing Web Service applications.

2.2 Web Service Implementation Lifecycle

A Web Service Implementation Lifecycle refers to the phases for developing Web Services from requirement to deployment.

The Web Service implementation lifecycle typically includes the following phases:

1. Requirements Phase [see 2.3.1]
2. Analysis Phase [see 2.3.2]
3. Design Phase [see 2.3.3]
4. Coding Phase [see 2.3.4]
5. Test Phase [see 2.3.5]
6. Deployment Phase [see 2.3.6]

The transitions through these phases need not be a single-pass sequential process. On the contrary, the process tends to be iterative and incremental in nature and should be agile enough to accommodate revisions in situations where the scope cannot be completely defined up front.

2.3 Phase

A Phase when used in the context of a Web Service implementation lifecycle refers to the period of time a set of related software implementation activities are carried out.

In general, the phases detailed in the sub-sections are identified to be pertinent in a Web Service implementation lifecycle. These phases may overlap with each other in the course of the implementation process as shown in Figure 1.
2.3.1 Requirements Phase

The objective in the requirements phase is to understand the business requirements and translating them to Web Service requirements in terms of the features, the functional and non-functional requirements, and the constraints within which the Web Service has to abide.

Requirements elicitation should be done by the requirements analyst and should involve the project stakeholders such as the project champion, customers, end users, etc. Following which, the analyst should interpret, consolidate and communicate these requirements to the development team.

If possible, Requirements should be aggregated in a centralized repository where they can be viewed, prioritized, and “mined” for iterative features. In all cases, enabling the team to easily capture requirements, search, prioritize and elaborate as necessary is the primary function of the repository.

2.3.2 Analysis Phase

In the analysis phase, the requirements of the Web Service are further refined and translated into conceptual models by which the technical development team can understand. It is also in this phase that an architecture analysis is done to define the high-level structure and identify the Web Service interface contracts. This process should be performed by both the requirements analyst and the architect and communicated to the design and development teams.
2.3.3 Design Phase

The detailed design of Web Services is done in this phase. In this phase, the designers should define the Web Service interface contract that has been identified in the analysis phase. The defined Web Service interface contract should identify the elements and the corresponding data types (possibly using a XML schema) as well as mode of interaction between the Web Service and the client, for example, whether it should be synchronous/asynchronous or RPC/Document style etc.

2.3.4 Coding Phase

The coding and debugging phase for Web Service implementation is essentially quite similar to other software component-based coding and debugging phase. The main difference lies in the creation of additional Web Service interface wrappers (to expose the components’ public APIs), generation of WSDLs and client stubs. Web Services in addition have to be deployed to a Web Server/Application Server before the test clients can consume them.

The component developer and/or the tester should perform these activities.

2.3.5 Test Phase

For testing of Web Services, besides testing for functional correctness and completeness, testers should also perform interoperability testing between different platforms and clients' programs. Furthermore, performance testing has to be conducted to ensure that the Web Services are able to withstand the maximum load and stress as specified in the non-functional requirements specification. Other tasks like profiling of the Web Service application and inspection of SOAP messages should also be done in this phase.

2.3.6 Deployment Phase

The purpose of the deployment phase is to ensure that the Web Service is properly deployed. The phase will be executed after the service has been tested. The deployment of the Web Service is platform specific. The service end points of the Web Service specifies where the service is deployed and it needs to be identified and configured accordingly. The deployer primary tasks are to ensure that the Web Service has been properly configured and managed (e.g. version controlled, presetting of configuration files, packaged and loaded in the correct location etc.) and running post-deployment tests to ensure that the Web Service is indeed ready for use. Other optional tasks like specifying and registering the Web Service with an UDDI registry may also be performed in this phase.

Table 1 summaries the overview of each phase against its respective assigned roles.

<table>
<thead>
<tr>
<th>Phases</th>
<th>Primary Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>Requirements Analysts</td>
</tr>
<tr>
<td>Analysis</td>
<td>Requirements Analysts</td>
</tr>
<tr>
<td></td>
<td>Architects</td>
</tr>
<tr>
<td>Design</td>
<td>Designers</td>
</tr>
<tr>
<td>Coding</td>
<td>Developers</td>
</tr>
<tr>
<td></td>
<td>Testers</td>
</tr>
<tr>
<td>Test</td>
<td>Testers</td>
</tr>
<tr>
<td>Deployment</td>
<td>Deployers</td>
</tr>
</tbody>
</table>

Table 1: Mapping between phases and roles assigned
2.4 Role

Commonly defined roles in software development methodology include the following:

<table>
<thead>
<tr>
<th>Roles</th>
<th>Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Analyst</td>
<td>Responsible for eliciting and interpreting the stakeholders’ needs, and communicating those needs to the entire team.</td>
</tr>
<tr>
<td>Architect</td>
<td>Responsible for the software architecture, which includes the key technical decisions that constrain the overall design and implementation for the project.</td>
</tr>
<tr>
<td>Designer</td>
<td>Responsible for designing a part of the system, within the constraints of the requirements, architecture, and development process for the project.</td>
</tr>
<tr>
<td>Developer</td>
<td>Responsible for developing and unit testing the components, in accordance with the project’s adopted standards.</td>
</tr>
<tr>
<td>Deployer</td>
<td>Responsible for planning the product’s transition to the user community, ensuring those plans are enacted appropriately, managing issues and monitoring progress.</td>
</tr>
<tr>
<td>Stakeholder</td>
<td>Responsible for providing the domain expertise and specifying the system requirements. Stakeholder usually includes the project champion and the end users.</td>
</tr>
<tr>
<td>Project Manager</td>
<td>Responsible for managing and monitoring the project including the project scope, schedule and staffing of the project team.</td>
</tr>
<tr>
<td>Test Manager</td>
<td>Responsible for the total test efforts including the quality and test advocacy, resource planning and management of the testing schedule, and resolution of issues that impede the test effort.</td>
</tr>
<tr>
<td>Test Designer</td>
<td>Responsible for defining the test approach and ensuring its successful implementation. The role involves identifying the appropriate techniques, tools and guidelines to implement the required tests, and to give guidance on the corresponding resources requirements for the test effort. The role also involves monitoring detailed testing progress and results in each test cycle and evaluating the overall quality as a result of testing activities.</td>
</tr>
<tr>
<td>Tester</td>
<td>Responsible for the core activities of the test effort, which involves conducting the necessary tests and logging the outcomes of that testing.</td>
</tr>
<tr>
<td>System Administrator</td>
<td>Responsible for planning, installing and maintaining the hardware and software of the different environments e.g. development, test, live environment</td>
</tr>
</tbody>
</table>

2.5 Glossary

Activity An Activity refers to a unit of work a role may be assigned to perform. Activities are performed within each of the phases in the Web Service implementation lifecycle.

Artifact An Artifact refers to the work-product that is used or produced as a result of performing an activity. Examples of Artifacts include models, source files, scripts, and binary executable files.

Role A Role refers to the responsibilities that a person or a team has been assigned with.
3 Web Service Implementation Methodology

The term Web Service describes a specialized type of software, which is designed to support a standardized way for provision and consumption of services over the Web, through the compliance with open standards such as eXtensible Markup Language (XML), SOAP, Web Services Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI).

Web Services, unlike traditional client/server systems, such as browser/Web server systems, are not meant for direct end-user consumption. Rather, Web Services are pieces of business logic, which have programmatic interfaces and it is through these interfaces that developers can create new application systems.

The motivation behind Web Services is to facilitate businesses to interact and integrate with other businesses and clients, without having to go through lengthy integration design and/or to expose its confidential internal application details unnecessarily. This is made possible by leveraging on the non-platform dependent and non-programming language dependent XML to describe the data to be exchanged between businesses or between the business and its clients, using a WSDL to specify what the service is providing; using a UDDI to publish and locate who is providing the service; and typically using SOAP over HTTP to transfer the message across the internet.

A Web Service, naturally, is a software element, but because of its specialized interface and mechanism to interoperate with others, all the prevalent generic software development methodology would need to be tailored to handle the unique features of Web Service. This could translate to identification of Web Service specific requirements (e.g. conformance to Web Services standards), analysis of the specific implications of Web Service on the overall system, design of the Web Service interface and XML message structure, coding, testing, deployment and execution of the Web Service.

The Web Service Implementation Methodology that we define is to promote a systematic approach to Web Service development. Rather than defining a new software development methodology and expecting software practitioners to forget their own familiar and established methodology to re-learn another, the better alternative is to leverage on what is already available and customize that methodology to incorporate the specifics of Web Services.

The candidate software development methodology should, ideally, be agile and able to accommodate refinement throughout the development cycle in an iterative and incremental approach. The methodology should consist of phases that cover from the conception of the need of the Web Service, to the construction of the Web Service and finally to be deployed for use by the eventual client application. In this document, these phases are identified as requirements, analysis, design, code, test and deployment.

The Web Service Implementation Methodology would leverage on any of the candidate agile software development methodology and extend the said methodology by specifying additional and/or customized Web Service specific activities and its corresponding roles and work-products.

The Web Service Implementation Methodology is iterative and incremental. In each iteration, the Web Service would go through all the phases (i.e. requirements, analysis, design, code, testing and finally deployment), thereby developing and refining the Web Services throughout the project lifecycle.

---

2 SOAP is transport agnostic. Therefore, other Internet (e.g. SMTP) or non-Internet (IBM MQ Series) may be used. From a practical perspective, however, SOAP over HTTP appears to be the typical scenario.
In addition, for Web Service testing, a multitude of tests have to be conducted to ensure that the Web Service is developed according to its functional as well as non-functional requirements. Figure 2 illustrates using the “V” Model to perform these tests.

Figure 2: The “V” Model incorporates the Web Services specific Interoperability test

The specifications produced in each of the development phases are sources of input to derive the test scenarios and test cases. From these test cases, test scripts and test data are compiled, which will be used in unit testing, functional testing, integration/interoperability testing, system/performance testing and the final user acceptance testing.

3.1 Overview

The Web Service Implementation Lifecycle describes the phases a typical Web Service would undergo, from the identification of the need of the Web Service to the final deployment and usage by the end-users. The phases identified to be relevant in the Web Service Implementation Lifecycle are: requirements, analysis, design, code, test and deployment. In each of these phases, Web Service specific activities are carried out. These activities, as well as the roles and responsibilities, and the artifacts will be elaborated in the subsequent subsections. Figure 3 illustrates the above-mentioned relationship between phase, activities and their respective tasks, roles and artifacts.

Figure 3: Relationship between phase, activities, tasks, roles and artifacts
3.2 Requirements Phase

3.2.1 Activity: Determine the need for Web Service

3.2.1.1 Tasks

- Identify the stakeholders
  Stakeholders would usually include the end users, project champion, project manager, etc.
- Understand the inadequacies/problems to address
  Understand the stakeholders’ need for Web Services.
- Identify the need for Web Service technology
  Based on the current technology available, identify needs specially for Web Services.
- Determine the positioning of the Web Service within the boundaries of the problem identified
- Define the features of the Web Service based on the needs list
- Identify the limitations to be imposed on the Web Service

3.2.1.2 Roles

Architect, Requirements Analyst, Stakeholders, Project Manager

3.2.1.3 Artifacts

The results should be recorded in Business Requirement Specifications.

3.2.2 Activity: Elicit Web Service requirements

3.2.2.1 Tasks

- Identify the sources for requirements gathering based on the features list
  Identify the departments, end users, domain experts, etc. who would be impacted by the introduction of Web Services.
- Gather information from these sources and elicit the requirements for the Web Service
- Identify functional requirements for the Web Service and categorise them
- Identify non-functional requirements for the Web Service
  Non-functional requirements are requirements pertaining to Usability, Reliability, Performance, Scalability, Supportability and other design considerations.

3.2.2.2 Roles

Requirements Analyst, Architect, Test Manager

3.2.2.3 Artifacts

The results should be recorded in Requirement Specifications.
3.2.3 Activity: Manage the Web Service requirements

3.2.3.1 Tasks

- Based on the functional requirements categories, identify the Web Services and establish the dependencies and priorities
- Create traceability matrices from the requirements to the identified Web Services
  Traceability matrices help to track the requirements that have been taken care of by the Web Services identified.
- Manage changes to the requirements

3.2.3.2 Roles

Requirements Analyst, Architect, Test Manager

3.2.3.3 Artifacts

The results should be recorded in Requirement Specifications.

3.2.4 Activity: Model the usage scenarios

3.2.4.1 Tasks

- Translate the functional requirements into conceptual usage models using some form of analysis modeling techniques
- Specify the major interaction scenarios with the Web Service clients
  This is to highlight the usage of Web Services involved. Especially, the message exchange scenarios should be captured.

3.2.4.2 Roles

Requirements Analyst, Architect, Test Manager

3.2.4.3 Artifacts

The results should be recorded in Requirement Specifications.

3.2.5 Activity: Prepare Test Cases for User Acceptance Test (UAT) and System Test

3.2.5.1 Tasks

- Write business scenario test case(s) based on the requirements gathered to be used for UAT and System Test
  Test case(s) can be derived from requirements. This is also a way to verify the requirements when they are implemented.
- Build requirement validation matrix
  The requirement validation matrix will include the requirements and a reference to the test case(s) that will validate the requirement.
- Manage changes to the test cases when requirements changed
### 3.2.5.2 Roles

Requirements Analyst, Test Manager, Test Designer

### 3.2.5.3 Artifacts

The results should be recorded in Test Plan – UAT and System Test.

Table 2 summaries the overview of each activities and the corresponding tasks, roles and artifacts under the activities.

<table>
<thead>
<tr>
<th>Activities</th>
<th>Tasks</th>
<th>Roles</th>
<th>Artifacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine needs</td>
<td></td>
<td>Architect</td>
<td>Business Requirements Specifications</td>
</tr>
<tr>
<td></td>
<td>Identify stakeholders</td>
<td>Requirements Analyst</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Understand the inadequacies/problems to address</td>
<td>Stakeholders</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Identify need for WS technology</td>
<td>Project Manager</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Determine positioning of WS within the boundaries of the problem identified</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Define features of WS based on needs</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Identify limitations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elicit requirements</td>
<td>Identify sources for requirements gathering</td>
<td>Architect</td>
<td>Requirement Specifications</td>
</tr>
<tr>
<td></td>
<td>Gather information</td>
<td>Requirements Analyst</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Identify functional requirements</td>
<td>Test Manager</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Identify non-functional requirements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manage requirements</td>
<td>Identify WS and establish dependencies and priorities</td>
<td>Architect</td>
<td>Requirement Specifications</td>
</tr>
<tr>
<td></td>
<td>Create traceability matrices</td>
<td>Requirements Analyst</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Manage changes to requirements</td>
<td>Test Manager</td>
<td></td>
</tr>
<tr>
<td>Model usage scenarios</td>
<td>Translate functional requirements into conceptual usage models</td>
<td>Architect</td>
<td>Requirement Specifications</td>
</tr>
<tr>
<td></td>
<td>Specify major interaction scenarios with WS clients</td>
<td>Requirements Analyst</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Test Manager</td>
<td></td>
</tr>
<tr>
<td>Prepare test cases for UAT and System Test</td>
<td>Write business scenarios test cases</td>
<td>Requirements Analyst</td>
<td>Test Plan – UAT and System Test</td>
</tr>
<tr>
<td></td>
<td>Build requirement validation matrix</td>
<td>Test Manager</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Manage changes to test cases</td>
<td>Test Designer</td>
<td></td>
</tr>
</tbody>
</table>

Notes: WS stands for Web Services

Table 2: Overview of activities, tasks, roles and artifacts in the Requirements Phase

### 3.3 Analysis Phase

#### 3.3.1 Activity: Select a technology platform as implementation framework

##### 3.3.1.1 Tasks

- Specify the Web Services standards that the implementation must adhere
- Identify the Web Service standards based on the requirements and implementation constraints. Consider issues like the standards compatibility, version of standards, standards adoption in industry sector, and the organization approving the standards.
- Decide the technology platform for implementing Web Services
- Choose a technology platform that is suitable for implementation. E.g. dotNet or Java platform.
- Decide the technology platform for hosting Web Services
Based on implementation constraints and considerations for standards support and interoperability requirements, choose the appropriate hosting platform for Web Services.

- Decide the IDE tools used to develop Web Services
  Available options include commercial vendor’s IDE tools, open source IDE tools.
  Normally, the selection of IDE is tied together with the implementation platforms.

### 3.3.1.2 Roles

#### Architect

### 3.3.1.3 Artifacts

The results should be recorded in Software Architecture Specifications.

### 3.3.2 Activity: Define a candidate architecture for the Web Service

#### 3.3.2.1 Tasks

- Define a high-level architecture

- Identify the architectural component that expose functionality as Web Services
  It is necessary to identify the architectural components that implement the wrapping of functionality as Web Services and implement the message exchanges in the high level architecture.

- Specify the major information exchange with Web Service clients
  Identify and specify the first cut definition of message that is exchanged with Web Services clients. The definition includes the element of data, data type and format.

#### 3.3.2.2 Roles

#### Architect

#### 3.3.2.3 Artifacts

The results should be recorded in Software Architecture Specifications.

### 3.3.3 Activity: Decide on the granularity of the Web Service

#### 3.3.3.1 Tasks

- Decide on the coarseness of the Web Service operations to be exposed
  Set up criteria on the coarseness of Web Services operations. Its definition depends on the usage scenarios and requirements.

- Identify and group functionality into the Web Service
  Based on the requirements and criteria mentioned above, identify the functions that are needed to group into the Web Services.

- Decide on the mechanisms to compose or aggregate functionality
  In case there is a need to compose individual Web Services, choose and decide the mechanism to implement the compositions.

#### 3.3.3.2 Roles

#### Architect
3.3.3 Artifacts
The results should be recorded in Software Architecture Specifications.

3.3.4 Activity: Identify reusable Web Services

3.3.4.1 Tasks
- Identify the architectural components that can be realized by existing Web Services
  If the functionality of the architecture component can be fulfilled with existing Web
  Services (internal or third party Web Services), the architectural components should
  be identified to make use of these existing Web Services.
- Identify the Web Service providers for the reusable Web Services
  Identify and gather the information about provider of existing Web Services.
- Define the major invocation scenarios of re-use
  Identify the functions that are going to be used. Define the interface of invocation.

3.3.4.2 Roles
Architect

3.3.4.3 Artifacts
The results should be recorded in Software Architecture Specifications.

3.3.5 Activity: Identify service interface for new Web Services

3.3.5.1 Tasks
- Define the new Web Service operation signatures
  Based on the usage models and analysis models, identify the operations and its
  signatures.
- Define XML schema for the message exchange
  If message exchanges are involved, the XML schema that guides the structure of the
  message should be defined.

3.3.5.2 Roles
Architect, Designer

3.3.5.3 Artifacts
Web Service Signature Specifications, XML schema.

3.3.6 Activity: Prepare Test Cases for Performance Test

3.3.6.1 Task
- Write performance test case(s) to be used for Performance Test
  Test case(s) can be derived from Architectural Design Specifications.
- These test cases should cover load testing scenarios to see how the system will
  perform under various loads (in terms of concurrent users/requests and/or
  transactions).
3.3.6.2 Roles
Test System Administrator, Test Designer

3.3.6.3 Artifacts
The results should be recorded in Test Plan – Performance Test.

3.3.7 Activity: Prepare Test Cases for Integration / Interoperability Test

3.3.7.1 Task

- Write integration / interoperability test case(s) to be used for Integration / Interoperability Test
  Test case(s) can be derived from Architectural Design Specifications.

3.3.7.2 Roles
Test Designer, Tester

3.3.7.3 Artifacts
The results should be recorded in Test Plan – Integration / Interoperability Test.

3.3.8 Activity: Prepare Test Cases for Functional Test

3.3.8.1 Task

- Write functional test case(s) to be used for Functional Test
  Test case(s) can be derived from Architectural Design Specifications.

3.3.8.2 Roles
Test Designer, Tester

3.3.8.3 Artifacts
The results should be recorded in Test Plan - Functional Test.

3.3.9 Activity: Testbed preparation

3.3.9.1 Task

- Set up testing environment that include hardware and software
- This environment may be similar to the production/live environment in terms of hardware, OS, Web Server/Application Server, etc.

3.3.9.2 Roles
Test System Administrator, Test Designer

3.3.9.3 Artifacts
The results should be recorded in Test Plan - Testbed.
Table 3 summaries the overview of each activities and the corresponding tasks, roles and artifacts under the activities.

<table>
<thead>
<tr>
<th>Activities</th>
<th>Tasks</th>
<th>Roles</th>
<th>Artifacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select a technology platform as implementation</td>
<td>Specify implementation standards</td>
<td>Architect</td>
<td>Software Architecture Specifications</td>
</tr>
<tr>
<td>framework</td>
<td>Decide technology platform for implementation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Decide technology platform for hosting</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Decide IDE tools used for development</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Define candidate architecture</td>
<td>Define high-level architecture</td>
<td>Architect</td>
<td>Software Architecture Specifications</td>
</tr>
<tr>
<td></td>
<td>Identify architectural component that expose functionality as WS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specify major information exchange with WS clients</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decide granularity</td>
<td>Decide on coarseness of the operations to be exposed</td>
<td>Architect</td>
<td>Software Architecture Specifications</td>
</tr>
<tr>
<td></td>
<td>Identify and group functionality</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Decide on mechanisms to compose or aggregate functionality</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identify reusable WS</td>
<td>Identify architectural components that can be realized by existing WS</td>
<td>Architect</td>
<td>Software Architecture Specifications</td>
</tr>
<tr>
<td></td>
<td>Identify WS providers for reusable WS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Define major invocation scenarios of re-use</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identify service interface</td>
<td>Define new WS operation signatures</td>
<td>Architect, Designer</td>
<td>WS Signature Specifications</td>
</tr>
<tr>
<td></td>
<td>Define XML schema for message exchange</td>
<td></td>
<td>XML Schema</td>
</tr>
<tr>
<td>Prepare test cases for Performance Test</td>
<td>Write Performance test cases</td>
<td>Test System Administrator, Test Designer</td>
<td>Test Plan – Performance Test</td>
</tr>
<tr>
<td>Prepare test cases for Integration / Interoperability Test</td>
<td>Write Integration / Interoperability test cases</td>
<td>Test Designer, Tester</td>
<td>Test Plan – Integration / Interoperability Test</td>
</tr>
<tr>
<td>Prepare test cases for Functional Test</td>
<td>Write functional test cases</td>
<td>Test Designer, Tester</td>
<td>Test Plan – Functional Test</td>
</tr>
<tr>
<td>Testbed preparation</td>
<td>Set up testing environment</td>
<td>Test System Administrator, Test Designer</td>
<td>Test Plan - Testbed</td>
</tr>
</tbody>
</table>

Notes: WS stands for Web Services

*Table 3: Overview of activities, tasks, roles and artifacts in the Analysis Phase*
3.4 Design Phase

3.4.1 Activity: Transform signatures of reusable Web Services

3.4.1.1 Tasks

- Identify the data type mapping if required
  If the type of a parameter of the reusable service is not directly supported by the identified platform, data type mapping should be performed.

- Identify the design patterns for mapping the re-used Web Service interface to the identified (desired) one
  Certain design patterns could be used to reuse existing Web Service(s), such as adapter pattern, façade pattern etc. Adapter pattern could be used to expose a new interface of an existing Web Service. The façade pattern could be used to encapsulate the complexity of existing Web Services and provide a coarse-grained Web Service.

3.4.1.2 Roles

Designer

3.4.1.3 Artifacts

The results should be recorded in Design Specifications.

3.4.2 Activity: Refine service interface of the new Web Service

3.4.2.1 Tasks

- Refine Web Service interfaces signature
  In the detailed design stage, the signature may be refined further. Care must be taken to ensure that the design decision should not affect the interoperability of the service.

- Refine XML schema for message exchange
  The XML schema may be refined to further expand on the data structure, data types, namespaces etc.

3.4.2.2 Roles

Designer

3.4.2.3 Artifacts

The results should be recorded in Design Specifications.

3.4.3 Activity: Design Web Service

3.4.3.1 Tasks

- Use some form of modeling techniques to describe the internal structure of the Web Service
  The design of the internal structure needs to consider the receiving and pre-processing of request, delegating of the request, processing of the request and sending of the response. Existing modeling techniques such as UML, design patterns could be applied to the design.
• Consider non-functional requirements (e.g. usability, reliability, performance, scalability etc.) and design constraints (e.g. interoperability etc.)

3.4.3.2 Roles
Designer

3.4.3.3 Artifacts

The results should be recorded in Design Specifications.

3.4.4 Activity: Refine Test Cases for Functional Test

3.4.4.1 Task
• Refine functional test case(s) to be used for functional Test
  Test case(s) can be refined by Design Specifications.

3.4.4.2 Roles
Test Designer, Tester

3.4.4.3 Artifacts
The results should be recorded in Test Plan – Functional Test.

Table 4 summaries the overview of each activities and the corresponding tasks, roles and artifacts under the activities.

<table>
<thead>
<tr>
<th>Activities</th>
<th>Tasks</th>
<th>Roles</th>
<th>Artifacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transform signatures of reusable WS</td>
<td>• Identify data type mapping if required</td>
<td>Designer</td>
<td>Design Specifications</td>
</tr>
<tr>
<td></td>
<td>• Identify design patterns for mapping the re-used WS interface to the desired one</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refine service interface of new WS</td>
<td>• Refine WS interface signature</td>
<td>Designer</td>
<td>Design Specifications</td>
</tr>
<tr>
<td></td>
<td>• Refine XML schema for message exchange</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design WS</td>
<td>• Use modeling techniques to describe internal structure of WS</td>
<td>Designer</td>
<td>Design Specifications</td>
</tr>
<tr>
<td></td>
<td>• Consider non-functional requirements and design constraints</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refine test cases for Functional Test</td>
<td>• Refine functional test cases</td>
<td>Test Designer</td>
<td>Test Plan – Functional</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tester</td>
<td>Test</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes: WS stands for Web Services

Table 4: Overview of activities, tasks, roles and artifacts in the Design Phase

3.5 Coding Phase

3.5.1 Activity: Construct Web Service code

3.5.1.1 Tasks
• Based on the implementation language choice, code the Web Service according to the design
Consider other constraints that are imposed by the specific implementation language itself. For example, consider the language dependent data types and the need to map these data types to the ones specified by the Web Service interface.

- Expose public APIs as Web Service interface
  For example, in Java, to create the interface class to expose the class method as a Web Service operation or in dotNet, to annotate the class API as a [WebMethod].

- Generate WSDL for client to consume
  Most IDEs can auto-generate the WSDL from the interface code.

3.5.1.2 Roles
Developer

3.5.1.3 Artifacts
Web Service Implementation Codes.

3.5.2 Activity: Construct Web Service client code

3.5.2.1 Tasks
- Decide on the Web Service Client programming model
  Among the three available are:
  a) Static Stub
  The client invokes the Web Service operation through a stub. Any IDE can generate this stub at compile time.
  b) Dynamic Proxy
  As the name implies, dynamic proxy is dynamically generated when the client application is executed. Because dynamic proxy is generated during runtime, Web Service invocation using this method takes the longest time amongst the three approaches.
  c) DII (Dynamic Invocation Interface)
  It is the most flexible approach among the three programming models. The client does not even need to know the signature of the Web Service operation until runtime. The Web Service invocation can be dynamically constructed.
  Hence, identify and decide on a suitable client programming model based on the weightage of flexibility against performance requirements.

- Write client code to consume the Web Service
  Use the WSDL to generate client stubs, which can be used in the client code to invoke the methods provided by the Web Service.

3.5.2.2 Roles
Developer

3.5.2.3 Artifacts
Web Service Client Codes.
3.5.3 Activity: Unit Test Web Service

3.5.3.1 Tasks

- Deploy Web Service in local test environment and perform functional unit testing
  The emphasis is on the correctness of the functionality and the exceptions handling.

3.5.3.2 Roles

Developer

3.5.3.3 Artifacts

Unit Test Scripts.

Table 5 summaries the overview of each activities and the corresponding tasks, roles and artifacts under the activities.

<table>
<thead>
<tr>
<th>Activities</th>
<th>Tasks</th>
<th>Roles</th>
<th>Artifacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct WS code</td>
<td>Code based on implementation language chosen</td>
<td>Developer</td>
<td>Implementation codes</td>
</tr>
<tr>
<td></td>
<td>Expose public APIs as interface</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Generate WSDL for client</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Construct WS client code</td>
<td>Decide client programming model</td>
<td>Developer</td>
<td>Client codes</td>
</tr>
<tr>
<td></td>
<td>Write client codes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unit test</td>
<td>Deploy in local test environment and perform functional unit testing</td>
<td>Developer</td>
<td>Unit test scripts</td>
</tr>
</tbody>
</table>

Notes: WS stands for Web Services

Table 5: Overview of activities, tasks, roles and artifacts in the Coding Phase

3.6 Test Phase

For Web Services, additional tests may be conducted to ensure that the Web Services are interoperable, secured and scalable.

Interoperability is an issue in Web Services because the standards governing Web Services are still evolving. Furthermore, different vendors that implement these specifications may interpret and comply with these specifications differently. Currently there is an effort by Web Services Interoperability Organization (WS-I) to recommend basic profiles to minimise these incompatibilities. The aim of conducting interoperability tests is to ensure that these recommendations are followed and the Web Service developed will interoperate with other Web Services and products without problems.

Network congestion created by Web Services is the major contributor to Web Services' slow performance. Not only is the messaging between requesters and Web Services impacted by network latency, but also the service discovery and description protocols that precede those message exchanges. The cumulative effect of these delays can seriously degrade the performance of Web Services. Therefore it is necessary to do a performance test on the Web Services before they are deployed for operation, and then to monitor the Web Services to determine if they can meet the service level agreements.

Web Services introduce special security issues e.g. in privacy, message integrity, authentication and authorization. Tests have to be conducted to ensure that these security requirements have been fulfilled. However, security schemes could complicate the process of testing and debugging Web Service basic functionality. For example, non-intrusive monitors
are often used in functional testing but encrypted traffic presents an obvious complication to this approach to testing.

### 3.6.1 Activity: Test functionality of Web Service

#### 3.6.1.1 Tasks

- Testing basic Web Service functionality
  - The Web Service should respond correctly to requests from their clients. The format of the SOAP message should be in compliance with the specifications. WSDL files, which contain metadata about Web Services’ interfaces, should be in compliance with the WSDL specifications published by W3C. Perform fault checking to see how it handles unexpected input. The test scripts and data prepared in the earlier phases are executed in this activity. The test results should be recorded, and bugs found should be reported to the code owners and fixed by them.

- Test for security
  - If a service requires a certain level of privacy, or if it requires that messages be authenticated in a certain way, then specific tests are needed to ensure that these security requirements are met. The test scripts and test data prepared in the earlier phases should be executed in this activity. Any inadequacies that may lead to possible security breaches should be reported and resolved by the code owner, designer or architect.

- Test the UDDI functionality
  - If a service is registered to a registry server, perform registering of Web Service, then write test clients to perform finding and binding of Web Service on the registry, and then use the registry data to actually invoke the service. Test results from the test scripts and data should be recorded and bugs should be fixed by the code owners.

- Test for SOAP intermediary capability
  - If particular SOAP message has one or more intermediaries along the message route that take actions based upon the instructions provided to them in the header of the SOAP message. Web Service SOAP intermediary testing must verify the proper functionality of these intermediaries. Test results from the test scripts and data should be recorded and bugs should be fixed by the code owners.

#### 3.6.1.2 Roles

**Tester, Test Designer**

#### 3.6.1.3 Artifacts

The results should be recorded in Client Test Code, Test Scripts and Test Results.

### 3.6.2 Activity: Integration Test on the Web Service

#### 3.6.2.1 Tasks

- Test for conformance to Web Services Interoperability Organization (WS-I) recommendations. Execute test scripts and data according to the test cases based on the WS-I recommendations.

- Perform interoperability testing based on various scenarios
  - This is to highlight the interoperability issues of Web Services implementation. Refer to Interoperability Guideline for the interoperability testing scenarios.
Perform integration testing based on various scenarios.
Based on the test cases prepared in the Analysis Phase, test scripts and test data, which are prepared are executed and analyzed in this activity.

3.6.2.2 Roles
Tester, Test Designer, Test System Administrator

3.6.2.3 Artifacts
The results should be recorded in Client Test Code, Test Scripts and Test Results.

3.6.3 Activity: System Test on the Web Service

3.6.3.1 Tasks
- Check system functionality and response time under different degrees of load increases.
The test cases that are prepared in the earlier phases are executed in this activity. The load increases can be sudden surges or gradual ramp-ups. The test results should be analyzed to determine potential bottlenecks and if the system is scalable.

- Check functionality and response time under different combinations of valid and invalid requests.
The results from the test execution should be analyzed to determine if the system can still render the expected quality of service as specified in the non-functional requirement specifications.

3.6.3.2 Roles
Tester, Test Designer, Test System Administrator

3.6.3.3 Artifacts
The results should be recorded in Client Test Code, Test Scripts and Test Results.

3.6.4 Activity: User Acceptance Test on the Web Service

3.6.4.1 Tasks
- Run the user acceptance test cases(s) for the Web Services system.
The test cases prepared in the Requirement Phase are used in this activity to validate the correctness and completeness of the Web Service system. Any bugs found should be reported and fixed by the code owners.

3.6.4.2 Roles
User, Test Manager, Test System Administrator

3.6.4.3 Artifacts
The results should be recorded in Client Test Code, Test Scripts and Test Results.
Table 6 summaries the overview of each activities and the corresponding tasks, roles and artifacts under the activities.

<table>
<thead>
<tr>
<th>Activities</th>
<th>Tasks</th>
<th>Roles</th>
<th>Artifacts</th>
</tr>
</thead>
</table>
| Test functionality | Test basic WS functionality  
|                  | Test for security  
|                  | Test UDDI functionality  
|                  | Test for SOAP intermediary capability                                  | Tester  
|                  | Test Designer  
| Integration test | Test for conformance to WS-I  
|                  | Perform interoperability test based on various scenarios  
|                  | Perform integration test based on various scenarios                                  | Tester  
|                  | Test Designer  
|                  | Test System Administrator                                               | Tester  
|                  | Test Designer  
|                  | Test System Administrator                                               | Test System Administrator |
| System test      | Check system functionality and response time under different degrees of load increases  
|                  | Check functionality and response time under different combinations of valid and invalid requests | Tester  
|                  | Test Designer  
|                  | Test System Administrator                                               | Test System Administrator |
| User acceptance test | Run UAT test cases                                                       | User  
|                  | Test Manager  
|                  | Test System Administrator                                               | Client test code  
|                  | Test scripts  
|                  | Test results                                                             | Test scripts  
|                  | Test results                                                             |

Notes: WS stands for Web Services

Table 6: Overview of activities, tasks, roles and artifacts in the Test Phase

3.7 Deployment Phase

3.7.1 Activity: Prepare deployment environment

3.7.1.1 Tasks

- Set up and configure the hardware for Web Service deployment
- Set up and configure the software for Web Service deployment

The software may include application server, database, etc. The application server should have a SOAP listener to support Web Services. Some Web Services may need the SOAP handler to be configured.

3.7.1.2 Roles

System Engineer

3.7.1.3 Artifacts

Release Notes.

3.7.2 Activity: Deploy Web Service

3.7.2.1 Tasks

- Determine service URL

Web Service URL is unique and used to identify the Web Service and where it is located.
• Prepare the deployment script

Deployment script is used to determine the steps of deployment. Although it is
different for different application server, most of them will include creation of directory,
copying files, shutting down and restarting the server.

• Deploy the Web Service

Execute the prepared deployment script.

• Generate WSDL file

After successfully deploying the Web Service, a WSDL file is needed to describe the
functions provided by the Web Service. WSDL can be created manually or by most
application servers, which will automatically generate the WSDL file after deployment.

3.7.2.2 Roles

Developer

3.7.2.3 Artifacts

WSDL File, Deployment Script.

3.7.3 Activity: Test deployment

3.7.3.1 Tasks

• Create (reuse) Web Service client code

The Web Service client code should be created by the developer during code and
debug phase.

• Consume Web Service with the client code

Because the functionality of the Web Service is properly tested, there is no need to
test all the operations. To make sure the Web Service is properly deployed and
configured, the best candidates of operations for invocation are the ones needed for
database connection, configuration of SOAP handler or any other special features of
the application server.

3.7.3.2 Roles

Tester

3.7.3.3 Artifacts

Web Service Client Codes.

3.7.4 Activity: Create end user support material

3.7.4.1 Tasks

• Create end user support material

The support material is needed to help the users to understand and use the Web
Service. For example, an interoperability guide of the Web Service.

3.7.4.2 Roles

Developer

3.7.4.3 Artifacts

### 3.7.5 Activity: Publish Web Service

#### 3.7.5.1 Tasks

- Identify the UDDI registry for publishing the Web Service
  - Based on the requirements, decide whether a private or public UDDI registry is needed and the version of the UDDI Business Registry specifications to follow.

- Prepare the information needed for publishing
  - The information may include key words for searching, description of Web Service, URL of WSDL file, etc.

- Publish the Web Service in the UDDI registry
  - Normally, the UDDI registry will support the publishing via browser.

- Search the Web Service by key words after publishing
  - Search the Web Service through browser provided by UDDI registry or tools provided by other vendors.

#### 3.7.5.2 Roles

- Developer

#### 3.7.5.3 Artifacts

- None.

Table 7 summaries the overview of each activities and the corresponding tasks, roles and artifacts under the activities.

<table>
<thead>
<tr>
<th>Activities</th>
<th>Tasks</th>
<th>Roles</th>
<th>Artifacts</th>
</tr>
</thead>
</table>
| Prepare deployment environment | • Set up and configure hardware  
• Set up and configure software | System Engineer | Release Notes |
| Deploy WS | • Determine service URL  
• Prepare deployment script  
• Deploy WS  
• Generate WSDL | Developer | WSDL file  
Deployment script |
| Test deployment | • Create (reuse) client code  
• Consume WS with client code | Tester | Client codes |
| Create end user support material | • Create support material | Developer | Interoperability guide  
User guide  
On-line help  
Tutorials  
Training material |
| Publish WS | • Identify UDDI registry for publishing  
• Prepare information for publishing  
• Publish in UDDI registry  
• Search by key words after publishing | Developer | -- |

Notes: WS stands for Web Services

Table 7: Overview of activities, tasks, roles and artifacts in the Deployment Phase
4 References

Appendix A. Acknowledgments

The following individuals were members of the committee during the development of this documentation:

- Ravi Shankar, CrimsonLogic Pte. Ltd.
- Jagdip Talla, CrimsonLogic Pte. Ltd.
- Andy Tan, Individual
- Roberto Pascual, The Infocomm Development Authority of Singapore
### Appendix B. Revision History

<table>
<thead>
<tr>
<th>Rev</th>
<th>Date</th>
<th>By Whom</th>
<th>What</th>
</tr>
</thead>
<tbody>
<tr>
<td>wd-01</td>
<td>2004-09-30</td>
<td>Lai Peng CHAN Chai Hong ANG</td>
<td>Initial version</td>
</tr>
<tr>
<td>-</td>
<td>2004-12-23</td>
<td>Chai Hong ANG</td>
<td>Split the document into two</td>
</tr>
<tr>
<td>wd-01a</td>
<td>2005-05-24</td>
<td>Chai Hong ANG Puay Siew TAN Han Boon LEE</td>
<td>▪ Remover Section 2.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▪ Terminology, Section 2.2 Concepts and Section 2.2.1 Web Service and combined them as Section 2.1 Objective</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▪ Renumber Section 2.2.2 to Section 2.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▪ Renumber Section 2.2.3 to 2.3. The rest of the sub-sections are renumbered accordingly</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▪ Added Table 1 as summary for phase and its assigned roles</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▪ Section 2.2.4 Activity and Section 2.2.6 Artifact are moved into Section 2.5 Glossary</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▪ Renumber Section 2.2.5 to 2.5 and put them into table format</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▪ Section 3 is renamed as Web Service Implementation Methodology</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▪ Removed Section 3.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▪ Renumber Section 3.1.1 to 3.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▪ Renumber Section 3.1.2 to 3.2. The rest of the sub-sections are renumbered accordingly</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▪ Tables are added into each phase for summary</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▪ Removed Normative and Non-Normative from Section 4</td>
</tr>
<tr>
<td>wd-01b</td>
<td>2005-06-02</td>
<td>Chai Hong ANG Prof. Marc Haines</td>
<td>Edited based on Prof. Haines' comments</td>
</tr>
<tr>
<td>wd-</td>
<td>2005-08-29</td>
<td>Eng Wah LEE</td>
<td>Added Prof. Marc Haines as Editor and renamed the document to Public Review Draft 1.0</td>
</tr>
<tr>
<td>PublicReviewDraft1.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wd-</td>
<td>2005-09-05</td>
<td>Eng Wah LEE</td>
<td>Removed the phrase “This document is updated periodically on no particular schedule. Send comments to the editor” under Status as this document is now in a stable version ready for public review.</td>
</tr>
<tr>
<td>PublicReviewDraft1.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix C. Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.

Copyright © OASIS Open 2005. All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself does not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an “AS IS” basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.