การพัฒนาเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องด้วยโครงสร้างของบีเพล

นางสาวธันยธร ลีลาวัชรมาศ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2554

A Development of a Supporting Tool for Constructing Fault Tolerant Web Services with BPEL Structure

Miss Tunyathorn Leelawatcharamas

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Computer Science

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2011

หัวข้อวิทยานิพนธ์	การพัฒนาเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องด้วยโครงสร้างของบีเพล
โดย	นางสาวธันยธร ลีลาวัชรมาศ
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(อาจารย์ ดร.ยรรยง เต็งอำนวย)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา)

.....กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร.เบญจพร ลิ้มธรรมาภรณ์)

ธันยธร ลีลาวัชรมาศ : การพัฒนาเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องด้วยโครงสร้างของบีเพล. (A Development of a Supporting Tool for Constructing Fault Tolerant Web Services with BPEL Structure) อ. ที่ปรึกษาวิทยานิพนธ์หลัก : รศ. ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา, 104 หน้า.

เทคโนโลยีที่เกี่ยวกับเซอร์วิซ อาทิเช่น เว็บเซอร์วิซเป็นเทคโนโลยีหนึ่งที่สำคัญของการพัฒนาซอฟต์แวร์ในปัจจุบัน การให้บริการเซอร์วิซอาจจะมีปัญหาเรื่องการติดต่อสื่อสารหรือการเกิดข้อผิดพร่องขึ้น ดังนั้นผู้ใช้บริการอาจจะประสบกับการขัดข้องของเซอร์วิซ ซึ่งการแก้ปัญหาคือการสร้างเซอร์วิซที่ทนต่อความผิดพร่อง เพื่อทำให้ผู้ใช้บริการยังคงใช้บริการต่อไปได้แม้จะเกิดความขัดข้องขึ้น เนื่องจากมีแบบรูปการทนต่อความผิดพร่องหลายแบบรูปที่สามารถใช้งานได้ จึงเกิดคำถามว่า จะใช้แบบรูปใดกับเซอร์วิซหนึ่งๆ งานวิจัยนี้ทำการแนะนำแบบรูปการทนต่อความผิดพร่องให้กับผู้พัฒนาเซอร์วิซ โดยเสนอแบบจำลองการแนะนำแบบรูปจากลักษณะของเซอร์วิซและลักษณะของสภาพแวดล้อมการทำงานของเซอร์วิซ เมื่อผู้พัฒนาเลือกแบบรูปแล้ว จะสามารถสร้างเว็บเซอร์วิซนี้ให้ทนต่อความผิดพร่องโดยใช้โครงสร้างของบีเพล ผู้วิจัยได้ทำการพัฒนาเครื่องมือสนับสนุนการแนะนำแบบรูปและการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่อง จากการประเมินแบบจำลองการแนะนำแบบรูปโดยผู้พัฒนาเว็บเซอร์วิซจำนวนหนึ่ง พบว่าได้ผลเป็นที่น่าพอใจ โดยส่วนใหญ่แบบจำลองสามารถแนะนำแบบรูปได้ตรงกับแบบรูปที่ผู้พัฒนาเว็บเซอร์วิซใช้

ภาควิชา……วิศวกรรมคอมพิวเตอร์……ลายมือชื่อนิสิต……
สาขาวิชา……วิทยาศาสตร์คอมพิวเตอร์….ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก……
ปีการศึกษา…2554……..

5271427621: MAJOR COMPUTER SCIENCE
KEYWORDS : FAULT TOLERANCE PATTERNS / WEB SERVICES / WS-BPEL

TUNYATHORN LEELAWATCHARAMAS :A DEVELOPMENT OF A SUPPORTING TOOL FOR CONSTRUCTING FAULT TOLERANT WEB SERVICES WITH BPEL STRUCTURE. ADVISOR :ASSOC.PROF. TWITTIE SENIVONGSE, 104 pp.

Service technology such as Web services has been one of the mainstream technologies in today's software development. Distributed services may suffer from communication problems or contain faults themselves, and hence service consumers may experience service interruption. A solution is to create services which can tolerate faults so that failures can be made transparent to the consumers. Since there are many patterns of software fault tolerance available, we end up with a question of which pattern should be applied to a particular service. This research recommends to service developers the patterns for fault tolerant services. A recommendation model is proposed based on characteristics of the service itself and of the service provision environment. Once a fault tolerance pattern is chosen, a fault tolerant version of the service can be created as a WS-BPEL service. A software tool is developed to assist in pattern recommendation and generation of the fault tolerant service version. The recommendation model is evaluated by a number of Web service developers and the result is satisfactory, showing that mostly the model can recommend fault tolerance patterns similar to what the developers design for their services.

Department : Computer Engineering Student's Signature .....
Field of Study : Computer Science Advisor's Signature .....
Academic Year : 2011

กิตติกรรมประกาศ

ข้าพเจ้าขอกราบขอบพระคุณรองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่คอยให้คำแนะนำ ให้ความรู้ และคำปรึกษา ตลอดจนแนวทางต่างๆ ในการจัดทำวิทยานิพนธ์ให้สำเร็จลุล่วงไปด้วยดี

ขอกราบขอบพระคุณอาจารย์ ดร.ยรรยง เต็งอำนวย ประธานกรรมการสอบวิทยานิพนธ์ และผู้ช่วยศาสตราจารย์ ดร.เบญจพร ลิ้มธรรมาภรณ์ กรรมการสอบวิทยานิพนธ์ ที่ได้ให้ข้อเสนอแนะต่างๆ ในการจัดทำวิทยานิพนธ์

ขอขอบพระคุณภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ที่อำนวยความสะดวก และเพื่อนๆ พี่ๆ ที่ให้คำแนะนำในเรื่องต่างๆ ตลอดมา

สุดท้ายขอกราบขอบพระคุณบิดา มารดา ผู้มีพระคุณที่ให้กำลังใจ และสนับสนุนเกี่ยวกับการเรียนตลอดมา

สารบัญ

หน้า

หน้า

สารบัญตาราง

สารบัญภาพ

ภาพที่ หน้า

บทที่ 1

บทนำ

1.1. ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันในองค์กรต่างๆ ประกอบด้วยระบบงานหลายชนิด การนำระบบงานต่างๆ มาเชื่อมต่อกัน เพื่อแบ่งปันทรัพยากรและเทคโนโลยีระหว่างกันจึงเป็นสิ่งสำคัญที่จะช่วยลดการใช้ทรัพยากร เพิ่มประสิทธิภาพในการติดต่อสื่อสารกัน และรองรับรูปแบบของธุรกิจที่เปลี่ยนแปลงไปอย่างรวดเร็ว จึงทำให้เกิดสถาปัตยกรรมเชิงบริการ (Service-Oriented Architecture: SOA) ที่เป็นสถาปัตยกรรมของซอฟต์แวร์ที่นิยามถึงวิธีการใช้บริการที่เป็นอิสระต่อกัน เพื่อรองรับการทำงานร่วมกันของระบบต่างๆ ที่มีการใช้แพลตฟอร์มและเทคโนโลยีที่แตกต่างกัน สำหรับเทคโนโลยีที่นิยมนำมาใช้กับเอสโอเอ คือ เว็บเซอร์วิซ (Web Service) ซึ่งจะเป็นระบบซอฟต์แวร์ที่พัฒนาขึ้น เพื่อให้สามารถแลกเปลี่ยนข้อมูลซึ่งกันและกันระหว่างเครื่องคอมพิวเตอร์ โดยภาษาที่ใช้ในการติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์ คือเอ็กซ์เอ็มแอล เว็บเซอร์วิซมีส่วนต่อประสานที่ใช้อธิบายรายละเอียดในการติดต่อสื่อสารที่เรียกว่า วิสเดิล (Web Service Definition Language: WSDL) เพื่อให้แอพพลิเคชันทราบถึงบริการ และวิธีติดต่อสื่อสารของแต่ละเซอร์วิซ และใช้ภาษาบีเพล (Business Process Execution Language: BPEL) ในการพัฒนากระบวนการทางธุรกิจ

การพัฒนาเว็บเซอร์วิซให้มีคุณภาพ ถือเป็นสิ่งสำคัญที่จะทำให้ผู้ใช้บริการนิยมเข้ามาใช้บริการ และแอพพลิเคชันที่เรียกใช้บริการเว็บเซอร์วิซสามารถทำงานได้อย่างมีประสิทธิภาพ ซึ่งการทำให้กระบวนการทางธุรกิจทนต่อความผิดพร่อง (Fault Tolerant) เป็นสิ่งหนึ่งที่ทำให้การเรียกใช้เว็บเซอร์วิซสามารถดำเนินงานต่อไปได้ ถึงแม้ว่าจะเกิดข้อผิดพร่องต่างๆ ระหว่างที่มีการเรียกใช้เซอร์วิซ ทั้งข้อผิดพร่องที่เกิดจากเซอร์วิซที่เรียกใช้งานไม่ได้และผลลัพธ์ที่ได้จากเซอร์วิซไม่ถูกต้อง ในกระบวนการทางธุรกิจมีการเรียกใช้เซอร์วิซต่างๆ การทำให้เซอร์วิซมีการทนต่อความผิดพร่องในการพัฒนากระบวนการทางธุรกิจด้วยภาษาบีเพลจึงเป็นสิ่งสำคัญสิ่งหนึ่งที่ช่วยป้องกันและแก้ไขข้อผิดพร่องที่เกิดขึ้นกับเว็บเซอร์วิซ เนื่องจากแบบรูป (Pattern) ของการทนต่อความผิดพร่องมีหลายแบบ โดยแต่ละแบบมีลักษณะที่แตกต่างกัน ดังนั้นการเลือกแบบรูปการทนต่อความผิดพร่องให้เหมาะกับลักษณะของแต่ละเซอร์วิซ ถือเป็นสิ่งจำเป็นที่จะทำให้ผู้ใช้บริการได้เซอร์วิซที่ทนต่อความผิดพร่องและเหมาะกับลักษณะของเซอร์วิซ เพื่อให้สามารถทำงานร่วมกันได้อย่างมีคุณภาพ

งานวิจัยหลายงานได้เสนอวิธีการที่ทำให้เซอร์วิซทนต่อความผิดพร่องผ่านกระบวนการทางธุรกิจของบีเพล วิธีการแบบหนึ่ง คือการใช้ความสามารถของโครงสร้างบีเพลอย่างเดียวในการ

ทำให้ทนต่อความผิดพร่อง [1, 2, 3, 4, 5] แต่บีเพลมีชุดคำสั่งที่จำกัด ทำให้ความสามารถในการออกแบบโครงสร้างบีเพลที่ทนต่อความผิดพร่องทำได้ค่อนข้างจำกัดตามไปด้วย โดยแต่ละงานวิจัยมีแบบรูปการทนต่อความผิดพร่องทั้งที่เหมือนกันและแตกต่างกัน วิธีการอีกแบบหนึ่งของการทำให้เซอร์วิซทนต่อความผิดพร่อง คือ แบบที่มีการสร้างส่วนต่อขยายให้กับเครื่องประมวลผลบีเพล (BPEL Engine) เพื่อพัฒนาให้บีเพลมีความสามารถในการทนต่อความผิดพร่องได้มากขึ้น ซึ่งมีการพัฒนาในหลายรูปแบบ อาทิเช่น การเพิ่มกลไกให้เครื่องประมวลผลสามารถกู้ระบบโดยอัตโนมัติ เมื่อมีการตรวจพบข้อผิดพร่องเกิดขึ้น [6] การใช้ตัวบริการแทนในการค้นหาและเลือกเซอร์วิซที่ทำงานแบบเดียวกัน เพื่อทำงานแทนเซอร์วิซจริงที่เกิดข้อผิดพร่อง [7, 8] และการใช้สำเนาในการทนต่อความผิดพร่อง [9, 10] เป็นต้น ถึงแม้ว่าแบบที่ใช้ความสามารถของโครงสร้างบีเพลอย่างเดียวจะมีข้อจำกัด แต่มีข้อดีคือ บีเพลที่ถูกทำให้ทนต่อความผิดพร่องจะยังเป็นบีเพลที่เป็นไปตามมาตรฐานอยู่และทำงานได้บนเครื่องประมวลผลที่ทำงานตามมาตรฐาน ผู้วิจัยจึงสนใจในวิธีการนี้ โดยไม่ต้องเพิ่มส่วนต่อขยายให้กับเครื่องประมวลผล ซึ่งจำเป็นต้องใช้ทรัพยากรและการลงทุนในการใช้งานเพิ่มจากเดิม

งานวิจัยส่วนใหญ่ที่เกี่ยวข้องกับการทำให้เว็บเซอร์วิซและบีเพลทนต่อความผิดพร่อง จะเสนอวิธีการหรือกลไกในการพัฒนาแบบรูปต่างๆ ของการทนต่อความผิดพร่อง ตัวอย่างเช่น หากต้องการให้เกิดการเรียกเซอร์วิซซ้ำ (แบบรูป Retry) จะเรียกอย่างไร ซ้ำกี่ครั้ง หรือหากต้องการให้เรียกเซอร์วิซอื่นมาทำงานแทน (แบบรูป Recovery Block) จะเรียกอย่างไร เซอร์วิซใดจะเป็นตัวแทน เป็นต้น ซึ่งในลักษณะนี้หมายความว่า ผู้พัฒนาเว็บเซอร์วิซหรือบีเพล จะต้องกำหนดไว้ก่อนแล้วว่า เซอร์วิซใดจะให้ทนต่อความผิดพร่องโดยใช้แบบรูปใด มีงานวิจัยเป็นส่วนน้อยที่จะกล่าวถึงการแนะนำแบบรูปที่เหมาะสมสำหรับแต่ละเซอร์วิซให้กับผู้พัฒนาเซอร์วิซ ในงานวิจัยนี้ผู้วิจัยขอเสนอการพัฒนาเครื่องมือเพื่อช่วยสนับสนุนผู้พัฒนาเว็บเซอร์วิซทางฝั่งผู้ให้บริการ ในการพัฒนาเว็บเซอร์วิซให้สามารถทนต่อความผิดพร่องได้ โดยฟังก์ชันงานหลักของเครื่องมือจะประกอบด้วย (1) การแนะนำผู้พัฒนาเว็บเซอร์วิซเกี่ยวกับการทนต่อความผิดพร่องซึ่งเหมาะสมกับเว็บเซอร์วิซทางฝั่งผู้ให้บริการ โดยใช้แบบจำลองการแนะนำแบบรูปการทนต่อความผิดพร่อง ซึ่งมีการพิจารณาลักษณะของตัวเว็บเซอร์วิซและลักษณะการให้บริการเว็บเซอร์วิซนั้น (2) การสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องตามแบบรูปที่นักพัฒนาเว็บเซอร์วิซเลือกจากข้อ (1) โดยใช้ภาษาบีเพล เครื่องมือนี้จะช่วยสนับสนุนการพัฒนาเว็บเซอร์วิซที่ทนต่อความผิดพร่องได้อย่างเหมาะสม ผ่านการประเมินลักษณะต่างๆ ของเว็บเซอร์วิซโดยผู้พัฒนาเอง

1.2. วัตถุประสงค์ของการวิจัย

1.2.1. เพื่อสร้างแบบจำลองการแนะนำแบบรูปการทนต่อความผิดพร่องสำหรับเว็บเซอร์วิซ

1.2.2. เพื่อพัฒนาเครื่องมือสนับสนุนแบบจำลอง โดยสามารถแนะนำแบบรูปและสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องได้

1.3. ขอบเขตของการวิจัย

1.3.1. วิเคราะห์แบบรูปการทนต่อความผิดพร่องแบบการกู้ระบบจากความผิดพร่องตามงานวิจัย [1, 11, 12]

1.3.2. สร้างแบบจำลองการแนะนำแบบรูปการทนต่อความผิดพร่อง

1.3.3. พัฒนาเว็บเซอร์วิซที่ทนต่อความผิดพร่องโดยใช้มาตรฐานดับเบิลยูเอส-บีเพล 2.0 และเว็บเซอร์วิซที่พิจารณาจะเป็นเซอร์วิซเดี่ยว ไม่ใช่เซอร์วิซประกอบ

1.3.4. ประเมินคุณภาพของเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องจากการใช้งานของผู้พัฒนาเซอร์วิซอย่างน้อย 10 ราย

1.3.5. เครื่องมือที่ใช้ในการพัฒนาและประมวลผลบีเพลเป็นโอเพนซอร์ซ คือ GlassFish ESB v2.2

1.4. ขั้นตอนการวิจัย

1.4.1. ศึกษาโครงสร้างของดับเบิลยูเอส-บีเพล 2.0 และวิธีการพัฒนาบีเพลด้วยเครื่องมือต่างๆ

1.4.2. ศึกษาและวิเคราะห์แบบรูปการทนต่อความผิดพร่องแบบการกู้คืนจากความผิดพร่องที่รองรับโครงสร้างของดับเบิลยูเอส-บีเพล 2.0

1.4.3. ศึกษาและวิเคราะห์ลักษณะของเซอร์วิซ

1.4.4. วิเคราะห์และออกแบบวิธีที่ใช้แนะนำแบบรูปการทนต่อความผิดพร่องสำหรับเซอร์วิซ เพื่อสร้างแบบจำลองการแนะนำแบบรูปการทนต่อความผิดพร่อง

1.4.5. สร้างบีเพลที่ทนต่อความผิดพร่องตามกรอบงานที่วางแผนไว้

1.4.6. สร้างแบบประเมินที่ใช้ทดสอบแบบจำลองและเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่อง

1.4.7. ทดสอบและวิเคราะห์ผลการประเมินแบบจำลองและเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่อง

1.4.8. สรุปผลการทดสอบ

1.4.9. จัดทำเอกสารและวิทยานิพนธ์

1.5. ประโยชน์ที่คาดว่าจะได้รับ

สามารถนำเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องไปใช้ในการอำนวยความสะดวกให้กับผู้ให้บริการที่ต้องการพัฒนาเว็บเซอร์วิซให้มีการทนต่อความผิดพร่องด้วยโครงสร้างของดับเบิลยูเอส-บีเพล 2.0 และช่วยในการแนะนำแบบรูปการทนต่อความผิดพร่องให้เหมาะสมกับลักษณะของเซอร์วิซ ซึ่งประเมินตามความเห็นของผู้พัฒนาเซอร์วิซ ข้อมูลที่ได้จากการประเมินคุณภาพของแบบจำลองและเครื่องมือสนับสนุนจะเป็นแนวทางในการพัฒนาการแนะนำแบบรูปและเครื่องมือสนับสนุนที่มีความสามารถเพิ่มขึ้นต่อไป

1.6. ผลงานตีพิมพ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้ตีพิมพ์และจะนำเสนอในงานประชุมวิชาการ The 8th International Conference on Computing and Information Technology (IC^2IT) ซึ่งจะจัดในวันที่ 9-10 พฤษภาคม 2555 ณ จังหวัดชลบุรี ประเทศไทย

บทความชื่อ “Recommendation and Application of Fault Tolerance Patterns to Services”

ชื่อผู้แต่ง Tunyathorn Leelawatcharamas และ Twittie Senivongse

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

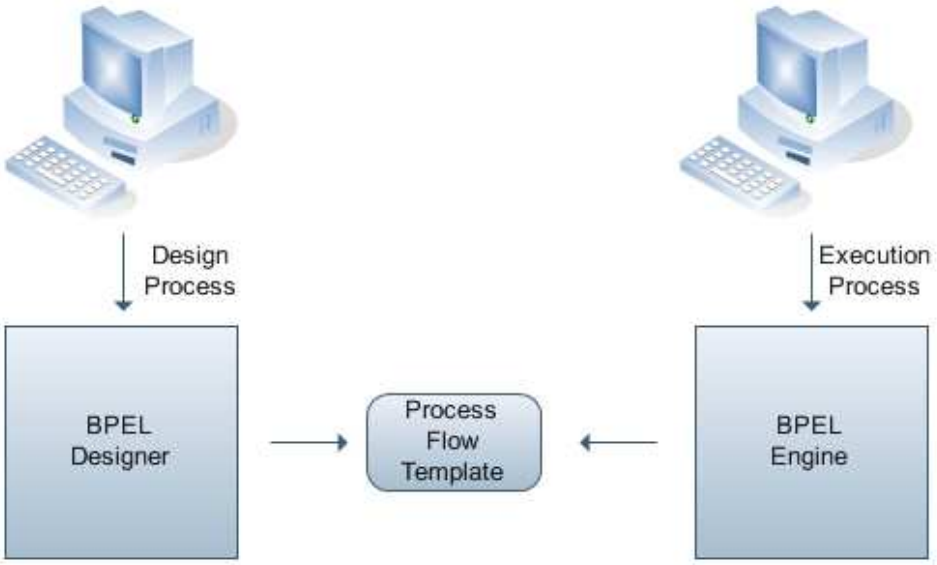
2.1. แนวคิดและทฤษฎี

2.1.1. ดับเบิลยูเอส-บีเพล

ดับเบิลยูเอส-บีเพล (WS-BPEL) ย่อมาจาก Web Services Business Process Execution Language เป็นมาตรฐานของโอเอซิส (OASIS) ที่ใช้ภาษาเอกซ์เอ็มแอลในการพัฒนากระบวนการทางธุรกิจ (Business Process) จากเว็บเซอร์วิซอื่น โดยใช้ภาษาวิสเดิลในการอธิบายการติดต่อกับกระบวนงานของเว็บเซอร์วิซที่เกี่ยวข้องเหล่านั้น ซึ่งก่อนที่จะมีภาษาบีเพล แต่ละบริษัทผู้ผลิตต่างมีรูปแบบของการเขียนกระบวนการทางธุรกิจที่แตกต่างกัน ดังนั้นจุดประสงค์ของการกำหนดมาตรฐานบีเพล เพื่อนิยามมาตรฐานกลางสำหรับการเขียนกระบวนการทางธุรกิจ โดยใช้แพลตฟอร์มที่เป็นเว็บเซอร์วิซ เวอร์ชันล่าสุดของบีเพลคือ 2.0

บีเพลจะมีองค์ประกอบหลักที่เกี่ยวข้องสามส่วนดังภาพที่ 2.1 คือ

- ตัวออกแบบบีเพล (BPEL Designer) เป็นเครื่องมือที่ให้ผู้เชี่ยวชาญด้านกระบวนการทางธุรกิจสามารถจำลองกระบวนการทางธุรกิจ โดยใช้แผนภาพกราฟฟิกส์ เพื่อสร้างไฟล์แผ่นแบบกระแสกระบวนการ (Process Flow Template) โดยทั่วไปเครื่องมือเหล่านี้จะอิงตามมาตรฐานบีพีเอ็มเอ็น (Business Process Modeling Notation: BPMN) ในการเขียนแผนภาพ
- เครื่องประมวลผลบีเพล (BPEL Engine) เป็นตัวประมวลผลไฟล์แผ่นแบบกระแสกระบวนการตามมาตรฐานบีเพล โดยจะทำงานต่างๆ เช่น เรียกใช้เว็บเซอร์วิซ กำหนดเนื้อหาของข้อมูล จัดการข้อผิดพร่อง หรือควบคุมลำดับการทำงาน โดยทั่วไปเครื่องประมวลผลบีเพลจะทำงานร่วมกับแอพพลิเคชันเซิร์ฟเวอร์ (Application Server)
- แผ่นแบบกระแสกระบวนการ (Process Flow Template) เป็นไฟล์ที่ระบุกระบวนการทางธุรกิจตามข้อกำหนดของบีเพล โดยจะเป็นไฟล์ที่ถูกสร้างมาจากตัวออกแบบบีเพล และจะใช้เครื่องประมวลผลบีเพลในการประมวลผล



ภาพที่ 2.1 องค์ประกอบของบีเพล

2.1.2. คำสั่งของภาษาบีเพล

โปรแกรมบีเพล จะใช้แท็กเอกซ์เอ็มแอลในการประกอบเว็บเซอร์วิซเพื่อสร้างกระบวนการทางธุรกิจ โดยบีเพลที่สร้างขึ้นมาสามารถจะแสดงออกมาเป็นเว็บเซอร์วิซที่นิยามโดยวิสเดิล และสามารถเรียกใช้บีเพลอื่นได้เหมือนการเรียกเว็บเซอร์วิซโดยทั่วไป บีเพลจะมีชุดของคำสั่งที่ระบุภารกิจพื้นฐานที่ใช้ในการประกอบเว็บเซอร์วิซดังนี้

- <invoke> - คำสั่งเพื่อให้กระบวนการทางธุรกิจเรียกใช้โอเปอเรชัน (Operation) ภายในแท็ก Porttype ที่นิยามอยู่ในวิสเดิลของเว็บเซอร์วิซอื่น
- <receive> - คำสั่งเพื่อให้กระบวนการทางธุรกิจหยุดรอข่าวสารที่จะมาถึง
- <reply> - คำสั่งเพื่อให้กระบวนการทางธุรกิจส่งข่าวสารเพื่อตอบกลับข่าวสารที่ได้รับมา
- <assign> - คำสั่งเพื่อคัดลอกข้อมูลจากตำแหน่งหนึ่งไปยังตำแหน่งอื่น
- <throw> - คำสั่งเพื่อระบุข้อผิดพร่องที่เกิดขึ้น
- <wait> - คำสั่งเพื่อให้กระบวนการทางธุรกิจหยุดรอตามระยะเวลาหนึ่ง
- <terminate> - คำสั่งเพื่อยกเลิกกระบวนการทางธุรกิจทั้งหมด

บีเพลมีชุดคำสั่งที่เป็นภารกิจแบบโครงสร้าง (Structured Task) ที่ใช้การผนวกภารกิจพื้นฐานเข้าด้วยกัน เพื่อใช้ควบคุมลำดับการทำงานและสร้างกระบวนการทางธุรกิจที่ซับซ้อนขึ้น โดยมีคำสั่งต่างๆ ดังนี้

- <sequence> - คำสั่งเพื่อนิยามการทำงานของภารกิจต่างๆ แบบต่อเนื่อง
- <flow> - คำสั่งเพื่อระบุให้ชุดภารกิจทำงานแบบขนาน
- <switch> - คำสั่งเพื่อกำหนดให้ชุดภารกิจทำงานแบบเลือกทำ (case-switch) ตามเงื่อนไขตรรกะที่ระบุ
- <while> - คำสั่งเพื่อกำหนดให้มีการทำงานของกลุ่มภารกิจซ้ำจนกว่าเป็นไปตามเงื่อนไขที่ระบุ
- <pick> - คำสั่งเพื่อบล็อกและรอจนกระทั่งมีข่าวสารที่เหมาะสมมาถึงหรือหมดเวลาที่รอ เมื่อคำสั่งประเภทนี้ถูกกระตุ้น กิจกรรมที่เกี่ยวพันกันจะถูกกระทำและจะสิ้นสุดการเลือก (Pick)

นอกจากนี้บีเพลยังมีคำสั่งในการนิยามข้อมูลดังนี้

- <partnerLink> - คำสั่งเพื่อกำหนด porttype ของเว็บเซอร์วิซ (จะเรียกว่า คู่ค้าหรือ Partner) ที่จะเข้ามาร่วมในกระบวนการทางธุรกิจ
- <variable> - คำสั่งเพื่อกำหนดค่าตัวแปรในกระบวนการทางธุรกิจ

2.1.3. **ความผิดพร่องที่เกิดจากเซอร์วิซของคู่ค้า**

เนื่องจากความผิดพร่องที่เกิดจากเซอร์วิซของคู่ค้า (Partner) เกี่ยวข้องกับเซอร์วิซภายนอก ซึ่งโดยทั่วไปอาจมีการติดต่อกันผ่านทางอินเทอร์เน็ตที่ไม่น่าเชื่อถือ และกระบวนการทางธุรกิจของเซอร์วิซได้มีการเรียกบริการจากเซอร์วิซภายนอก ดังนั้นความผิดพร่องที่เกิดจากเซอร์วิซคู่ค้าจึงมีผลกระทบกับแอพพลิเคชันของเซอร์วิซ ความผิดพร่องที่เกิดจากเซอร์วิซของคู่ค้าแบ่งออกเป็น 4 ประเภท [1, 3] ดังนี้

- ความผิดพร่องเชิงตรรกะ (Logical Faults) เกิดจากเซอร์วิซภายนอกที่ไม่สามารถทำงานได้อย่างสมบูรณ์ เนื่องจากเหตุผลต่างๆ เช่น เซอร์วิซเที่ยวบินที่ถูกเรียกเพื่อจองตั๋วเครื่องบิน 4 ใบ แต่สามารถจองได้จริง 2 ใบ ดังนั้นเซอร์วิซจึงแจ้งข้อผิดพร่องให้ผู้ร้องขอบริการทราบ ซึ่งจะถูกตรวจหาความผิดพร่องได้จากตรรกะทางธุรกิจของเว็บเซอร์วิซ และในเอกสารวิสเดิลที่อยู่ในรูปแบบของข้อความแจ้งข้อผิดพร่อง
- ความผิดพร่องที่ระบบ (System Faults) เกิดจากการประมวลผลและสภาพแวดล้อม เช่น กำลังไฟของเครื่องคอมพิวเตอร์ดับอย่างกระทันหัน หรือเครื่องประมวลผลบีเพลปิดด้วยเหตุผลบางอย่าง เป็นต้น

- ความผิดพร่องที่เนื้อหา (Content Faults) เกิดจากผลลัพธ์ของเซอร์วิซไม่ถูกต้อง เช่น เซอร์วิซประมวลผลหาระยะทางระหว่างโรงแรมกับสถานที่ที่สนใจ ได้ผลลัพธ์เป็น 12 กิโลเมตร แต่ระยะทางจริงคือ 9 กิโลเมตร ซึ่งตรวจหาความผิดพร่องได้จากกลไกของการพิสูจน์ยืนยัน เพื่อทดสอบความถูกต้องของผลลัพธ์ที่ได้จากเซอร์วิซ
- ความผิดพร่องที่เอสแอลเอ (SLA Faults) เกิดเมื่อเซอร์วิซทำงานตามความต้องการได้อย่างสมบูรณ์ แต่ไม่เป็นไปตามเอสแอลเอ (Service Level Agreement : SLA) หรือข้อตกลงในการให้บริการ เช่น เอสแอลเอระบุว่า ระยะเวลาในการประมวลผลที่ต้องการต้องน้อยกว่า 5 วินาที แต่ระยะเวลาจริง คือ 8 วินาที ซึ่งตรวจหาความผิดพร่องได้จากเอกสารเอสแอลเอที่ใช้สำหรับประเมินว่า เงื่อนไขของการให้บริการเป็นไปตามที่ตกลงไว้หรือไม่

2.1.4. แบบรูปการทนต่อความผิดพร่องของดับเบิลยูเอส-บีเพล

การทำให้เซอร์วิซทนต่อความผิดพร่องสามารถทำได้โดยการจัดการกับสิ่งผิดพร่องต่างๆ ดับเบิลยูเอส-บีเพลสามารถนำมาใช้ดำเนินการให้กระบวนการทางธุรกิจสามารถทนต่อความผิดพร่องได้ด้วยไวยากรณ์ระดับพื้นฐานของตัวภาษาเอง ซึ่งแบบรูปการทนต่อความผิดพร่องของดับเบิลยูเอส-บีเพล [1, 11, 12] มีดังนี้

- Retry – การประมวลผลเซอร์วิซซ้ำจนกระทั่งเงื่อนไขเป็นจริง ในเงื่อนไขอาจเป็นจำนวนรอบที่ทำซ้ำ แบบรูปนี้อาจใช้ได้กับข้อผิดพร่องที่เกิดขึ้นชั่วคราว เพื่อลดการติดต่อสื่อสารที่มากเกินไป สามารถกำหนดช่วงเวลาในการรอระหว่างทำซ้ำได้
- Wait – การรอเรียกใช้งานของเซอร์วิซโดยการระบุเวลาที่คงที่ได้ เนื่องจากบางเซอร์วิซสามารถใช้งานได้หรือมีคุณภาพของเซอร์วิซดีในเวลาใดเวลาหนึ่ง ดังนั้นควรรอการเรียกใช้งานไว้จนกว่าจะถึงเวลาที่ระบุ เพื่อลดความน่าจะเป็นในการเกิดความผิดพร่อง เช่น สมมติว่า เซอร์วิซเที่ยวบินจะทำงานหลัง 8:00 น. ถ้าเรียกเซอร์วิซนี้ตอนก่อน 8:00 น. จะทำให้มีความผิดพร่องเกิดขึ้น ดังนั้นจึงควรใช้แบบรูป Wait
- Recovery Block หรือ Alternate – เมื่อเซอร์วิซล้มเหลว จะเลือกเซอร์วิซอื่นที่มีฟังก์ชันการทำงานเหมือนกันมาทำงานแทน พื้นฐานของแบบรูปนี้คือ จำนวนของเซอร์วิซที่มีฟังก์ชันการทำงานเหมือนกัน โดยจะเลือกเซอร์วิซตัวแทนไปทีละตัวจนกว่าจะทำงานสำเร็จหรือทุกเซอร์วิซตัวแทนถูกเรียกจนหมด เซอร์วิซตัวแทนอาจเป็นเซอร์วิซที่เหมือนกันเลย แต่อยู่ในสภาพแวดล้อมที่แตกต่างกัน (Replica) หรือเป็นเซอร์วิซที่มีฟังก์ชันการทำงาน

เหมือนกัน แต่พัฒนาแตกต่างกัน โดยอาจจะพัฒนาด้วยทีมพัฒนาคนละทีมหรือทีมพัฒนาเดียวกัน แต่ใช้อัลกอริทึมหรือภาษาโปรแกรมที่แตกต่างกัน (N-Version Programming: NVP)

- Active – ใช้เมื่อมีการเรียกเซอร์วิซที่ทำงานเหมือนกันหลายตัวพร้อมกันแบบขนาน เพื่อลดโอกาสที่จะเกิดความขัดข้องเมื่อเทียบกับกรณีที่เรียกเซอร์วิซเพียงตัวเดียว ดังนั้นจึงทำให้มีหลายคำตอบจากหลายเซอร์วิซ ในการเลือกคำตอบ จะเลือกคำตอบของเซอร์วิซตัวแรกสุดที่ให้คำตอบก่อน โดยเซอร์วิซที่ถูกเรียกอาจเป็นเซอร์วิซที่เหมือนกันเลย หรือเป็นเซอร์วิซที่ถูกพัฒนาแตกต่างกัน
- Voting – ใช้เมื่อมีการเรียกเซอร์วิซที่ทำงานเหมือนกันหลายตัวพร้อมกันแบบขนาน เพื่อลดโอกาสที่จะเกิดความขัดข้องเมื่อเทียบกับกรณีที่เรียกเซอร์วิซเพียงตัวเดียว ดังนั้นจึงทำให้มีหลายคำตอบจากหลายเซอร์วิซ ในการเลือกคำตอบ จึงต้องมีอัลกอริทึมในการเลือกคำตอบที่เหมาะสม ซึ่งอาจเลือกคำตอบที่เซอร์วิซที่ให้คำตอบแบบเดียวกันมากที่สุด หรือเลือกจากประสบการณ์ที่เคยเรียกใช้เซอร์วิซ หรือเลือกค่ากลางของคำตอบทั้งหมด หรือเลือกคำตอบแบบสุ่ม เป็นต้น ทั้งนี้ขึ้นกับลักษณะของเซอร์วิซด้วย โดยเซอร์วิซที่ถูกเรียกอาจเป็นเซอร์วิซที่เหมือนกันเลย หรือเป็นเซอร์วิซที่ถูกพัฒนาแตกต่างกัน

2.2. เอกสารและงานวิจัยที่เกี่ยวข้อง

2.2.1. งานวิจัยที่เกี่ยวข้องกับแบบรูปการทนต่อความผิดพร่องของบีเพล

Avizienis และคณะ [13] ได้ให้แนวคิดและจัดประเภทของการทำให้การประมวลผลมีความน่าเชื่อถือและปลอดภัย ซึ่งได้ให้ความหมายของการประมวลผลที่มีความน่าเชื่อถือและปลอดภัย โดยจะต้องป้องกันการเกิดความผิดพร่อง ทนต่อความผิดพร่อง กำจัดความผิดพร่องและประเมินความผิดพร่องที่อาจจะเกิดขึ้นได้ ซึ่งการทนต่อความผิดพร่องแบ่งออกเป็น 2 วิธี คือ การตรวจจับหาข้อผิดพร่อง และการกู้ระบบจากความผิดพร่อง งานวิจัยหลายงานได้เสนอแบบรูปการทนต่อความผิดพร่องของเว็บเซอร์วิซในส่วนของบีเพล เช่น Liu และคณะ [1] ได้เสนอกรอบงานสำหรับการทนต่อความผิดพร่องของส่วนประกอบของเว็บเซอร์วิซ โดยใช้ Fault-Handling Logic ในดับเบิลยูเอส-บีเพลตามรูปแบบของ Event-Condition-Action Rules แบบรูปการทนต่อความผิดพร่องมีกระบวนงานที่จะพัฒนา ECA Rules ทั้งหมด 8 แบบ ตัวอย่างเช่น

1) การเรียกใช้เซอร์วิซซ้ำ (Retry)

Begin
01: add an integer variable count whose initial value is the allowed repeat times;
02: find the scope, say S, which raises the fault;
03: add an assign activity immediately after the last activity in the scope to let the value of count to -1;
04: add a fault handler for the scope;
05: add a catch construct into the fault handler;
06: add an if activity whose if condition evaluates whether the value of count is 0 into the catch construct;
07: add a rethrow activity into the if branch;
08: add a wait activity into the else branch. The wait is a duration type, and its waitExpression is the parameter duration in the retry strategy;
09: add an assign activity (still in the else branch) immediately after the wait activity to subtract one from the variable count;
10: add a while activity whose loop condition is that the value of the count variable is greater than -1;
11: move the links related to S onto the while activity;
12: move S into the while activity;
End

2) การเรียกใช้เซอร์วิซตัวแทน (Recovery Block หรือ Alternate)

Begin
01: add a boolean variable needExecute whose initial value is false;
02: find the scope that raises the fault;
03: add a fault handler for the scope;

04: add a catch construct into the fault handler;
05: add an assign activity that lets the value of the variable needExecute be true into the catch construct;
06: add an if activity whose if condition is the value of the variable needExecute is true immediately after the scope;
07: move the links started from the scope onto the if activity;
08: add the service specified in the alternate strategy to the if branch of the if activity;
End

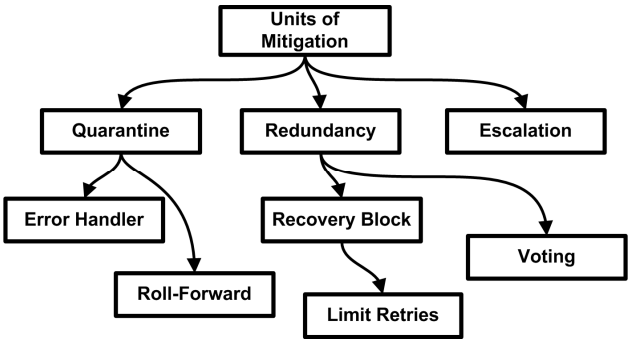
3) การรอการเรียกใช้เซอร์วิซ (Wait)

Begin
01: add a variable needWait whose initial value is false;
02: find the scope that raises the fault;
03: add a fault handler for the scope;
04: add a catch construct into the fault handler;
05: add an assign activity that lets the value of the variable needWait be true into the catch construct;
06: add an if activity whose if condition is the value of the variable needWait is true immediately before the scope specified in the wait strategy;
07: add a wait activity to the if branch of the if activity;
08: move the links ended at the scope onto the if activity;
End

โดยแบ่งแบบรูปออกเป็น 2 กลุ่ม คือ กลุ่มที่ 1 ได้แก่ Ignore, Notify, Skip, Wait กลุ่มที่ 2 คือ Retry, RetryUntil, Alternate, Replicate เนื่องจากการใช้แบบรูปเดียวอาจไม่สามารถแก้ไขข้อผิดพร่องที่เกิดขึ้นได้ จึงได้มีการรวมแบบรูปต่างๆ เข้าด้วยกัน โดยรวมระหว่างแบบรูปของกลุ่มที่ 1 กับกลุ่มที่ 2 เพื่อใช้ในการสร้างบีเพลตามมาตรฐานดับเบิลยูเอส-บีเพล 2.0 โดยใช้ ActiveBPEL ตัวอย่างเช่น เมื่อแบบรูปการเรียกใช้เซอร์วิซซ้ำ (Retry) ไม่สามารถแก้ไขข้อผิดพร่องที่เซอร์วิซ

เรียกใช้งานไม่ได้ จะทำการใช้แบบรูปการเพิกเฉยกับความผิดพร่อง (Ignore) หลังจากที่ครบจำนวนรอบมากที่สุดที่เรียกซ้ำ เป็นต้น เหตุผลที่ไม่รวมแบบรูปในกลุ่มเดียวกัน เนื่องจากว่าการรวมแบบรูปในกลุ่มเดียวกันจะทำให้กระบวนงานของการทนต่อความผิดพร่องมีความซับซ้อน ซึ่งอาจเป็นสาเหตุของการเกิดข้อผิดพร่องแบบอื่นเพิ่มขึ้นได้

Thaisongsuwan และ Senivongse [2] ได้ประยุกต์แบบรูปสำหรับซอฟต์แวร์ที่ทนต่อความผิดพร่องกับกระบวนการดับเบิลยูเอส-บีเพล 2.0 โดยใช้เครื่องประมวลผลบีเพลแบบโอเพนซอร์ซ และได้ออกแบบโครงสร้างของบีเพลที่ครอบคลุมแบบรูปที่ทนต่อความผิดพร่องต่างๆ ดังภาพที่ 2.2



ภาพที่ 2.2 แผนภาพแบบรูปที่ทนต่อความผิดพร่อง

จากรูปข้างต้น ลูกศรที่โยงจากแบบรูปหนึ่งไปยังอีกแบบรูปหนึ่งแสดงถึงความสัมพันธ์ระหว่างแบบรูป คือ ถ้าแบบรูป A โยงไปที่แบบรูป B หมายความว่า การประยุกต์แบบรูป A ทำให้เกิดเป็นแบบรูป B หรือแบบรูป A ถูกประยุกต์เพื่อให้แบบรูป B ดำเนินต่อ หรือแบบรูป B ถูกใช้เพื่อแก้ปัญหาต่อจากแบบรูป A ในบางกรณี อาจใช้แบบรูป B แบบเดียว โดยไม่ต้องใช้แบบรูป A แต่ละแบบรูปได้ถูกนำมาเชื่อมโยงกับคำสั่งในดับเบิลยูเอส-บีเพล ดังนี้

1) Units of Mitigation คือ การแบ่งส่วนของระบบในการจัดการกับข้อผิดพร่อง โดยคำสั่ง <scope> สามารถใช้แบ่งกระบวนการในดับเบิลยูเอส-บีเพลให้เป็นหน่วยที่เล็กลงมาได้ แต่ละหน่วยสามารถเป็นกิจกรรมหรือชุดลำดับของกิจกรรม
2) Quarantine คือ เมื่อมีข้อผิดพร่องเกิดขึ้น ควรมีการจัดการและป้องกันข้อผิดพร่องที่เกิดขึ้น เพื่อไม่ให้กระจายความผิดพร่องให้ไปกระทบกับส่วนอื่นของระบบ

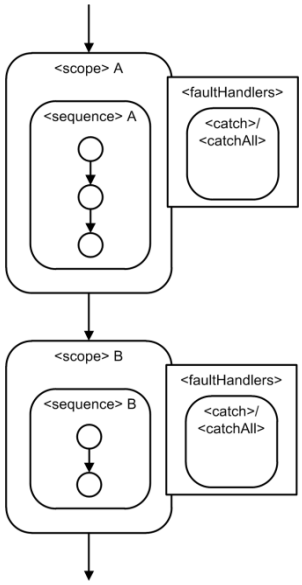
ในแบบรูป Units of Mitigation ควรถูกออกแบบแยกจากระบบหลัก เพื่อให้หน่วยไม่กระทบกับส่วนอื่นของระบบ เมื่อมีข้อผิดพร่องเกิดขึ้น

3) Error Handler คือ เมื่อระบบตรวจพบข้อผิดพร่องในการประมวลผล การจัดการกับข้อผิดพร่องควรแยกออกจากกระบวนการปกติ ดับเบิลยูเอส-บีเพลสนับสนุนการจัดการกับข้อยกเว้นด้วยคำสั่ง <faultHandlers> ติดกับคำสั่ง <scope> หรือ <invoke> โดยผู้ออกแบบสามารถใช้คำสั่ง <catch> จัดการกับข้อผิดพร่องที่รู้จักและสามารถใช้คำสั่ง <catchAll> จัดการกับข้อผิดพร่องที่ไม่รู้จัก
4) Redundancy คือ การมีเซอร์วิซที่เหมือนกันแต่ทำงานอยู่ในสถานที่ที่แตกต่างกัน หรือเซอร์วิซที่พัฒนาต่างกัน แต่มีฟังก์ชันการทำงานเหมือนกัน ซึ่งอาจจะเรียกเซอร์วิซพร้อมกัน แล้วเลือกเซอร์วิซที่ให้ผลลัพธ์ดีที่สุด (Voting) หรืออาจจะเรียกเซอร์วิซตามลำดับ ถ้าเซอร์วิซหลักเกิดข้อผิดพร่อง จะทำการเรียกเซอร์วิซที่มีลักษณะเหมือนกันแทน (Recovery Block)
5) Recovery Block คือ เมื่อเซอร์วิซหลักเกิดความผิดพร่อง จะทำการเรียกเซอร์วิซสำรองแทน โดยเซอร์วิซสำรองจะมีเซอร์วิซที่พัฒนาแตกต่างกันหนึ่งหรือมากกว่าหนึ่งชุด ถ้าเซอร์วิซสำรองทุกเซอร์วิซไม่สามารถผ่านการทดสอบที่ยอมรับได้ จะทำการใช้แบบรูป Escalation เพื่อหาวิธีอื่นที่สามารถจัดการกับข้อผิดพร่องได้ต่อไป
6) Limit Retries คือ การจำกัดจำนวนรอบที่ทำซ้ำ เมื่อเซอร์วิซหลักไม่สามารถเรียกใช้งานได้ เพื่อหลีกเลี่ยงการใช้ทรัพยากรที่มากจนเกินไป และการผูกกับการประมวลผลเฉพาะส่วนของกระบวนการ
7) Escalation คือ เมื่อการจัดการกับข้อผิดพร่องไม่ประสบผลสำเร็จ ควรจะหาวิธีการจัดการแบบอื่นที่จะมาจัดการแทนวิธีเดิม ดับเบิลยูเอส-บีเพล จะสนับสนุนแบบรูปนี้ด้วยการใช้คำสั่ง <throw> หลังจากลำดับของกระบวนการจัดการกับข้อผิดพร่อง ถ้ากระบวนการจัดการกับข้อผิดพร่องอยู่ใน <faultHandlers> จะใช้คำสั่ง <rethrow> ในการจัดการกับข้อผิดพร่องเดียวกัน
8) Roll Forward คือ เมื่อกระบวนการจัดการกับข้อผิดพร่องเสร็จสมบูรณ์ ควรจะดำเนินไปสู่การประมวลผลปกติ โดยใช้แบบรูป Roll-Forward เพื่อเพิกเฉยกับจุดที่ทำให้เกิดข้อผิดพร่อง และข้ามไปยังจุดข้างหน้าของแผนงาน Error Handler จะสามารถจัดการกับการข้ามไปจุดข้างหน้าได้

9) Voting คือ เมื่อเซอร์วิซซ้ำกันหลายเซอร์วิซถูกเรียกแบบขนาน จะทำให้ได้ผลลัพธ์หลายผลลัพธ์ กระบวนการควรจะมีกลไกในการลงคะแนนเพื่อตัดสินใจเลือกคำตอบจากผลลัพธ์ ถ้าผู้ลงคะแนนไม่สามารถตกลงกันได้ อาจจะทำให้เกิดข้อผิดพร่องในการเรียกเซอร์วิซที่ซ้ำกันได้

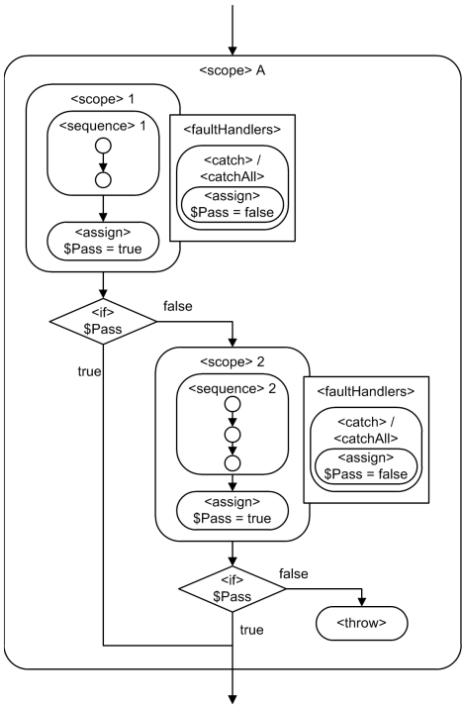
จากลักษณะของแบบรูปการทนต่อความผิดพร่องแบบต่างๆ ทำให้สามารถประยุกต์แบบรูปแบบต่างๆ เข้าด้วยกัน และนำมาออกแบบโครงสร้างของบีเพลได้ดังนี้

การประยุกต์แบบรูป Units of Mitigation, Quarantine และ Error Handler เข้าด้วยกัน มีแม่แบบของดับเบิลยูเอส-บีเพล แสดงดังภาพที่ 2.3 โดยแต่ละส่วนของกระบวนการถูกหุ้มด้วย <scope> และแต่ละ <scope> จะถูกเพิ่มด้วย <faultHandlers> เพื่อจัดการกับข้อผิดพร่องที่จะเกิดขึ้นด้านใน โดยแม่แบบนี้จะเป็นพื้นฐานของแบบรูปอื่นๆ



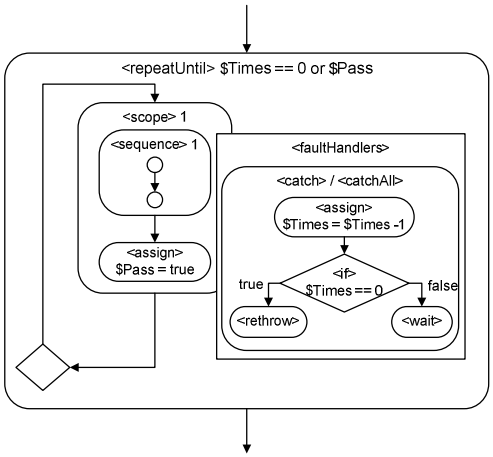
ภาพที่ 2.3 Scope and Fault Handler

การใช้แบบรูป Recovery Block และ Limit Retries ในการสร้างแม่แบบ ดังภาพที่ 2.4 จะมีตัวแปรชนิดตรรกะ คือ $Pass ใช้เป็นตัวบ่งชี้ว่า ส่วนหลักสามารถประมวลผลโดยปราศจากข้อผิดพร่องได้หรือไม่ ถ้ามีข้อผิดพร่องเกิดขึ้นกับส่วนหลัก จะกำหนดค่า False ให้กับ $Pass และประมวลผลส่วนสำรอง ถ้าส่วนสำรองล้มเหลว แม่แบบนี้จะโยนข้อผิดพร่องไปที่ด้านนอกของ <scope> และในด้านนอกของ scope อาจจะประยุกต์ใช้แบบรูป Escalation เพื่อใช้วิธีการกู้แบบอื่นที่จะจัดการกับข้อผิดพร่อง หรือจะใช้แบบรูป Roll Forward ข้ามส่วนที่เป็นปัญหาไป เพื่อให้ไม่เกิดข้อผิดพร่อง



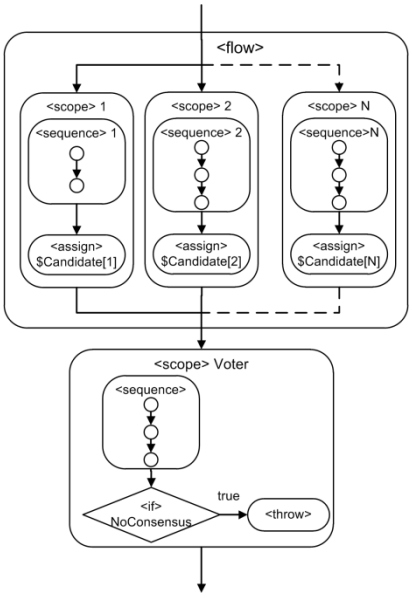
ภาพที่ 2.4 Recovery Block with Limit Retries

แบบรูป Recovery Block สามารถถูกใช้แบบ Retry ได้ ถ้าส่วนหลักแบบเดียวกันถูกเลือกเป็นส่วนสำรอง ในภาพแม่แบบดังภาพที่ 2.5 จะใช้วิธี Retry แม่แบบนี้จะถูกใช้เมื่อเซอร์วิซที่ถูกเลือกไม่สามารถใช้งานได้ชั่วคราวและจะกลับสู่สถานะปกติในเวลาไม่นาน ตัวแปรชนิดจำนวนเต็มคือ $Times จะถูกใช้เป็นตัวนับจำนวนรอบที่ทำซ้ำ <scope> จะถูกห่อหุ้มด้วยคำสั่ง <repeatUntil> ด้วยเงื่อนไขที่จะหยุดการทำซ้ำ เมื่อภารกิจประสบความสำเร็จหรือค่าของตัวแปร $Times ครบตามจำนวนที่กำหนด ในแต่ละรอบของการประมวลผลสามารถใช้คำสั่ง <wait> ในการหน่วงเวลาของกระบวนการ ก่อนที่จะเริ่มภารกิจอีกครั้ง



ภาพที่ 2.5 Recovery Block with the same scope

สำหรับแบบรูป Voting ดังภาพที่ 2.6 ในแต่ละลำดับที่เซอร์วิซถูกร้องขอจะถูกใส่ไว้ในคำสั่ง <flow> เพื่อให้เซอร์วิซประมวลผลพร้อมกัน หลังจากที่ได้รับการตอบกลับจากเซอร์วิซผลลัพธ์จะถูกใส่ไว้ในรายการของตัวเลือกโดย <scope> Voter จะใส่อัลกอริทึมในการลงคะแนนไว้ เพื่อหาคำตอบ ถ้าผู้ลงคะแนนไม่สามารถหาคำตอบได้ จะทำการโยนข้อผิดพร่องออกไปข้างนอก <scope>



ภาพที่ 2.6 Voting

Dobson [4] ได้เสนอแบบรูปที่ทำให้บีเพลทนต่อความผิดพร่องตามมาตรฐานดับเบิลยูเอส-บีเพล 2.0 อยู่ 3 วิธี ได้แก่ การเรียกใช้เซอร์วิซเดิมซ้ำหรือเซอร์วิซสำรอง (Retry) การเรียกใช้

เซอร์วิซแบบขนานและเลือกเซอร์วิซแรกที่ตอบกลับหรือเลือกเซอร์วิซจากการโหวต (Parallel Execution) และการตรวจหาข้อผิดพร่อง (Fault Detection)

Modafferi และ Confori [5] ได้เสนอแบบรูปของการกู้ระบบในดับเบิลยูเอส-บีเพล ได้แก่ การเปลี่ยนแปลงค่าของตัวแปรผ่านข้อความภายนอก การตั้งเวลาเพื่อให้ไปทำกิจกรรมอื่น การย้อนการทำงาน การเรียกใช้เซอร์วิซตัวแทน การย้อนการทำงานกลับไปยังจุดที่ปลอดภัย

จากงานวิจัยข้างต้นทำให้ได้แนวคิดในการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องโดยใช้บีเพล ซึ่งพัฒนาตามมาตรฐานดับเบิลยูเอส-บีเพล 2.0 และรวมแบบรูปการทนต่อความผิดพร่องที่สามารถพัฒนาได้ด้วยบีเพลเข้าด้วยกัน โดยนำกระบวนงานที่พัฒนาด้วย ECA Rules จาก [1] มาใช้เป็นแนวทางในการสร้างบีเพลที่ทนต่อความผิดพร่องในแบบรูป Retry, Recovery Block, Wait และจากงานวิจัย [2] ทำให้ได้แนวทางในการออกแบบบีเพลให้ทนต่อความผิดพร่องในแบบรูป Recovery Block, Retry, Voting รวมถึงสามารถพัฒนาต่อยอดการรวมแบบรูปการทนต่อความผิดพร่องแบบอื่นได้ อย่างไรก็ตามก่อนที่จะสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องโดยใช้บีเพล ผู้วิจัยจะได้เพิ่มในส่วนของการแนะนำแบบรูปการทนต่อความผิดพร่องสำหรับเว็บเซอร์วิซด้วย

2.2.2. งานวิจัยที่เกี่ยวข้องกับวิธีการแนะนำแบบรูปการทนต่อความผิดพร่องสำหรับเว็บเซอร์วิซ

Zheng และ Lyu [9, 11, 14, 15] ได้เสนองานวิจัยหลายงานที่เกี่ยวข้องกับแบบรูปสำหรับการทนต่อความผิดพร่องสำหรับเว็บเซอร์วิซ ในงานวิจัย [9, 15] ได้เสนอการประเมินแบบรูปแบบทำซ้ำ (Redundancy) แบบต่างๆ ซึ่งประเมินในด้านความสามารถในการใช้งานได้ ความถูกต้องและสมรรถนะของเซอร์วิซ จากสภาพแวดล้อมที่แตกต่างกัน โดยแบบรูปแบบทำซ้ำจะถูกแบ่งเป็น 2 ด้าน คือ แบบทำซ้ำด้านเวลา (Time Redundancy) และแบบทำซ้ำด้านพื้นที่ (Space Redundancy) แบบทำซ้ำด้านเวลาจะเป็นการใช้เวลาทำซ้ำเพิ่มขึ้นในการคำนวณหรือการติดต่อสื่อสาร เพื่อให้ทนต่อความผิดพร่อง ส่วนแบบทำซ้ำด้านพื้นที่จะขึ้นอยู่กับการมีทรัพยากร เช่น ฮาร์ดแวร์ ซอฟต์แวร์ เป็นต้น ที่เป็นสำเนาเพิ่มขึ้นเพื่อปิดบังความผิดพร่อง แบบทำซ้ำด้านพื้นที่จะแบ่งเป็นแบบ Active ซึ่งหมายถึงสำเนาถูกเรียกใช้งานพร้อมกันแบบขนาน และแบบ Passive ซึ่งหมายถึงสำเนาถูกเรียกใช้งานทีละตัวตามลำดับ การรวมแบบรูปเหล่านี้เข้าด้วยกันเป็นดังตารางที่ 2.1

ตารางที่ 2.1 แบบรูปจากการรวมแบบรูปแบบทำซ้ำ [9]

	Active	Time	Passive
Active	1.Active	4.Active+Time	6.Active+Passive
Time	5.Time+Active	2.Time	8.Time+Passive
Passive	7.Passive+Active	9.Passive+Time	3.Passive

การรวมแบบรูประหว่างแบบ Active, Time และ Passive ทำให้ได้แบบรูปแบบต่างๆ โดยแบบรูป A+B หมายความว่า แบบรูป A จะถูกใช้ก่อน แต่ถ้ายังเกิดข้อผิดพร่องอีกก็จะใช้แบบรูป B

แบบรูปข้างต้นได้ถูกแบ่งออกเป็น 3 กลุ่ม คือ

- Parallel (แบบรูปที่ 1) – เว็บเซอร์วิซสำเนาทุกตัวจะถูกเรียกพร้อมกัน กลุ่มนี้จะมีประสิทธิภาพดีในเรื่อง เวลาในการตอบกลับ แต่จะใช้ทรัพยากรมาก
- Sequential (แบบรูปที่ 2,3,8,9) – เว็บเซอร์วิซสำเนาจะถูกเรียกแบบลำดับ กลุ่มนี้จะใช้ทรัพยากรน้อยกว่า แต่ประสิทธิภาพในเรื่องเวลาในการตอบกลับไม่ดี ในสภาพแวดล้อมที่มีข้อผิดพร่อง
- Hybrid (แบบรูปที่ 4,5,6,7) – เว็บเซอร์วิซสำเนาบางตัวถูกเรียกแบบขนาน กลุ่มนี้จะใช้ทรัพยากรน้อยกว่ากลุ่ม Parallel และมีประสิทธิภาพในเรื่องเวลาในการตอบกลับดีกว่ากลุ่ม Sequential

ในการประเมินแบบรูปจะประเมินจากคุณภาพของเซอร์วิซในด้านอัตราความขัดข้อง (Failure Rate) และด้านเวลาในการตอบกลับ (Response Time) โดยดูจากการเก็บข้อมูลในอดีต เพื่อใช้ในการพิจารณาเลือกแบบรูปการทนต่อความผิดพร่องของเซอร์วิซที่มีคุณภาพดีที่สุด คือ เซอร์วิซมีอัตราความผิดพร่องและใช้เวลาในการตอบกลับน้อย ซึ่งจากงานวิจัยดังกล่าวมีข้อดีคือ สามารถประเมินแบบรูปการทนต่อความผิดพร่องจากคุณภาพของเซอร์วิซโดยตรง แต่ผู้วิจัยมีแนวคิดว่า การประเมินแบบรูปการทนต่อความผิดพร่องจากลักษณะของเซอร์วิซ และการให้บริการของเซอร์วิซก็เป็นอีกแนวความคิดหนึ่งที่จะทำให้ผู้พัฒนาสามารถเลือกแบบรูปได้ตรงกับการใช้งานและความต้องการของผู้พัฒนาเซอร์วิซ

กรอบงานในการประเมินแบบรูปแบบทำซ้ำในงานวิจัยดังกล่าวใช้งานได้กับเว็บเซอร์วิซแบบ Stateless เท่านั้น ในงานวิจัยต่อไป Zheng และ Lyu ต้องการที่จะปรับอัลกอริทึมในการเลือกแบบรูป และความสัมพันธ์ของคุณสมบัติของคุณภาพของเซอร์วิซให้มากขึ้น เพื่อประสิทธิภาพที่ดีขึ้นในการเลือกแบบรูปการทนต่อความผิดพร่อง ทำให้ได้มีงานวิจัย [14] ซึ่งเสนอ

มิดเดิลแวร์สำหรับการประกอบเซอร์วิซ โดยพิจารณาคุณภาพของเซอร์วิซที่นำมาประกอบ และผู้ใช้สามารถกำหนดน้ำหนักของความสำคัญของคุณภาพแต่ละด้านได้ การเลือกแบบรูปสำหรับการทนต่อความผิดพร่องจะถูกเปลี่ยนไปตามข้อมูลคุณภาพของเซอร์วิซที่เปลี่ยนแปลงไป

แบบจำลองคุณภาพของเซอร์วิซที่ผู้ใช้มีส่วนร่วม (User-Collaborated QoS Model) มีคุณภาพด้านต่างๆ ดังนี้

1) Availability (av) - เปอร์เซ็นต์ของเวลาที่เซอร์วิซทำงานในช่วงเวลาที่กำหนด
2) Price (pr) – ค่าธรรมเนียมที่ผู้ใช้เซอร์วิซจำเป็นต้องจ่ายสำหรับการเรียกเซอร์วิซ
3) Popularity (po) – จำนวนเว็บเซอร์วิซที่ได้รับคำร้องขอในช่วงเวลาที่กำหนด
4) Data Size (ds) – ขนาดของข้อมูลที่เว็บเซอร์วิซทำการตอบกลับ
5) Success Rate (sr) – ความน่าจะเป็นที่การร้องขอถูกตอบกลับภายในเวลาที่กำหนด
6) Response Time (rt) – เวลาตั้งแต่ที่ผู้ใช้ส่งการร้องขอจนได้รับการตอบกลับ
7) Overall Success Rate (osr) – ค่าเฉลี่ยของ Success Rate ของผู้ใช้เซอร์วิซทั้งหมด
8) Overall Response Time (ort) – ค่าเฉลี่ยของ Response Time ของผู้ใช้เซอร์วิซทั้งหมด

โดยแบบรูปการทนต่อความผิดพร่องจะแบ่งออกเป็น 2 ประเภท คือ ประเภทลำดับ (Sequential Strategies) คือ เมื่อเซอร์วิซหลักถูกเรียก แล้วเกิดความผิดพร่อง เซอร์วิซสำรองจะถูกเรียกแทน และประเภทขนาน (Parallel Strategies) คือ การเรียกเซอร์วิซที่ทำงานด้วยฟังก์ชันแบบเดียวกันหลายเซอร์วิซพร้อมกัน แบบรูปการทนต่อความผิดพร่องที่เสนอในงานวิจัย ได้แก่

1) Retry –เว็บเซอร์วิซจะถูกเรียกซ้ำตามจำนวนครั้งที่กำหนด เมื่อเว็บเซอร์วิซล้มเหลว

2) Recovery Block – มีเว็บเซอร์วิซสำรองที่พร้อมให้เรียกแบบลำดับ ถ้าเว็บเซอร์วิซหลักล้มเหลว

3) N-Version Programming – ทุกเว็บเซอร์วิซที่มีฟังก์ชันการทำงานแบบเดียวกัน ถูกเรียกพร้อมกันแบบขนาน และผลลัพธ์สุดท้ายจะได้จากการหาผลลัพธ์ส่วนใหญ่

4) Active - ทุกเว็บเซอร์วิซที่มีฟังก์ชันการทำงานแบบเดียวกัน ถูกเรียกพร้อมกันแบบขนาน และผลลัพธ์สุดท้ายจะได้จากเว็บเซอร์วิซที่ตอบกลับเป็นตัวแรกสุด โดยปราศจากการเกิดข้อผิดพร่องในระบบเครือข่าย

ในการหาค่าคุณภาพของเซอร์วิซประกอบรวม พิจารณาจากการนำแบบรูปการทนต่อความผิดพร่องแต่ละแบบรูปไปใช้กับแต่ละเซอร์วิซย่อยที่นำมาประกอบ เพื่อหาว่าแบบรูปการทนต่อความผิดพร่องใดทำให้ค่าคุณภาพของเซอร์วิซประกอบโดยรวมดีที่สุด ซึ่งมีข้อดีคือ ทำให้เลือกได้ว่า จะใช้แบบรูปใดและเซอร์วิซชุดใดที่ทำงานร่วมกัน แล้วทำให้คุณภาพของเซอร์วิซประกอบโดยรวมดี ผู้วิจัยเห็นว่า สำหรับเซอร์วิซหนึ่งๆ ไม่ใช่ว่าจะใช้ได้กับทุกแบบรูปการทนต่อความผิดพร่อง เนื่องจากมีเงื่อนไขของการให้บริการของเซอร์วิซ และลักษณะของเซอร์วิซอยู่ ดังนั้นในการแนะนำแบบรูปให้กับผู้พัฒนาเซอร์วิซ จึงควรที่จะทราบลักษณะของเซอร์วิซด้วย ผู้วิจัยจึงคิดว่า ในการแนะนำแบบรูปควรที่จะมีคำถามให้ผู้พัฒนาเซอร์วิซประเมินว่า เซอร์วิซที่จะถูกพัฒนามีลักษณะแบบใด เพื่อให้สามารถแนะนำแบบรูปการทนต่อความผิดพร่องได้อย่างเหมาะสม

2.2.3. งานวิจัยที่เกี่ยวข้องกับคุณภาพของการบริการของเว็บเซอร์วิซ

Shim และคณะ [16] ได้ออกแบบโมเดลในการประเมินคุณภาพของสถาปัตยกรรมเชิงบริการ (Service-Oriented Architecture) โดยกำหนดคุณภาพที่ต้องการ ได้แก่ ประสิทธิผล (Effectiveness) ความสามารถในการเข้าใจ (Understandability) ความยืดหยุ่น (Flexibility) ความสามารถนำกลับมาใช้ซ้ำ (Reusability) และสร้างเมตริกซ์ที่ใช้วัดคุณภาพ เพื่อนำไปทดสอบ

Yu และคณะ [17] ได้ออกแบบโมเดลในการวัดคุณภาพของการบริการในเว็บเซอร์วิซ ซึ่งได้รวบรวมเทคนิคที่ใช้วัดคุณภาพของการบริการในด้านต่างๆ คือ สมรรถนะ (Performance) ความเชื่อถือได้ (Reliability) การเพิ่มขนาดได้ (Scalability) ความทนทาน (Robustness) ความแม่น (Accuracy) บูรณภาพ (Integrity) ความสามารถใช้งานได้ (Ability) การเข้าถึงได้ (Accessibility) การทำงานร่วมกันได้ (Interoperability) และความมั่นคง (Security)

Zheng และ Lyu [14] ได้เสนอมิดเดิลแวร์ที่รวมโมเดลคุณภาพของการบริการที่ผู้ใช้มีส่วนร่วม กลยุทธ์ของการทนต่อความผิดพร่อง และขั้นตอนวิธี Context-Aware ในการกำหนดกลยุทธ์ของการทนต่อความผิดพร่องที่เหมาะสมสำหรับเว็บเซอร์วิซ โดยโมเดลคุณภาพของการบริการที่ร่วมมือกับผู้ใช้ ได้แก่ สภาพพร้อมใช้งาน(Availability) ราคา (Price) ความนิยม (Popularity) ขนาดของข้อมูล (Data Size) อัตราของความสำเร็จ (Success Rate) เวลาในการตอบกลับ (Response Time) อัตราของความสำเร็จทั้งหมด (Overall Success Rate) เวลาในการตอบกลับทั้งหมด (Overall Response time) ส่วนกลยุทธ์ของการทนต่อความผิดพร่อง ได้แก่ การเรียกเซอร์วิซซ้ำ (Retry) การเรียกเซอร์วิซตัวแทน (Recovery Block) การเรียกใช้เซอร์วิซแบบขนานและเลือกเซอร์วิซจากการโหวต (N-Version Programming) การเรียกใช้เซอร์วิซแบบขนานและเลือกเซอร์

วิซแรกที่ตอบกลับ (Active) ในการประเมินคุณภาพของแต่ละกลยุทธ์ จะประเมินจากความน่าจะเป็นของการมีของแต่ละกลยุทธ์ ซึ่งพิจารณาจากโครงสร้างของบีเพล นอกจากนี้มีอีกหนึ่งงานวิจัยของ Zheng และ Lyu [11] ได้เสนอคุณภาพของการบริการของกลยุทธ์การทนต่อความผิดพร่องของเว็บเซอร์วิซที่สามารถปรับเปลี่ยนได้ตามความต้องการของผู้ใช้ผ่านมิดเดิลแวร์ โดยผู้ใช้สามารถเลือกได้ว่า จะทำการแลกเปลี่ยนข้อมูลเกี่ยวกับคุณภาพของการบริการกับมิดเดิลแวร์หรือไม่ ซึ่งจะมีผลต่อการกำหนดกลยุทธ์ของการทนต่อความผิดพร่องที่เหมาะสม โดยคุณสมบัติของคุณภาพของการบริการที่ใช้ ได้แก่ เวลาเฉลี่ยในการตอบกลับ (Response Time) ส่วนเบี่ยงเบนมาตรฐานของเวลาในการตอบกลับ (Standard Deviation of Response Time) อัตราความขัดข้องทางด้านตรรกะ (Logic Failure Rate) อัตราความขัดข้องทางด้านระบบเครือข่าย (Network Failure Rate) ส่วนกลยุทธ์ของการทนต่อความผิดพร่องเสนอเช่นเดียวกับงานวิจัยก่อนหน้า และเพิ่มขั้นตอนวิธีการเลือกกลยุทธ์ของการทนต่อความผิดพร่องแบบพลวัต (A Dynamic Fault Tolerance Strategy Selection Algorithm) จากการทดสอบ ทำให้ทราบถึงประสิทธิภาพของคุณภาพของการบริการ โดยใช้แต่ละกลยุทธ์ของการทนต่อความผิดพร่อง

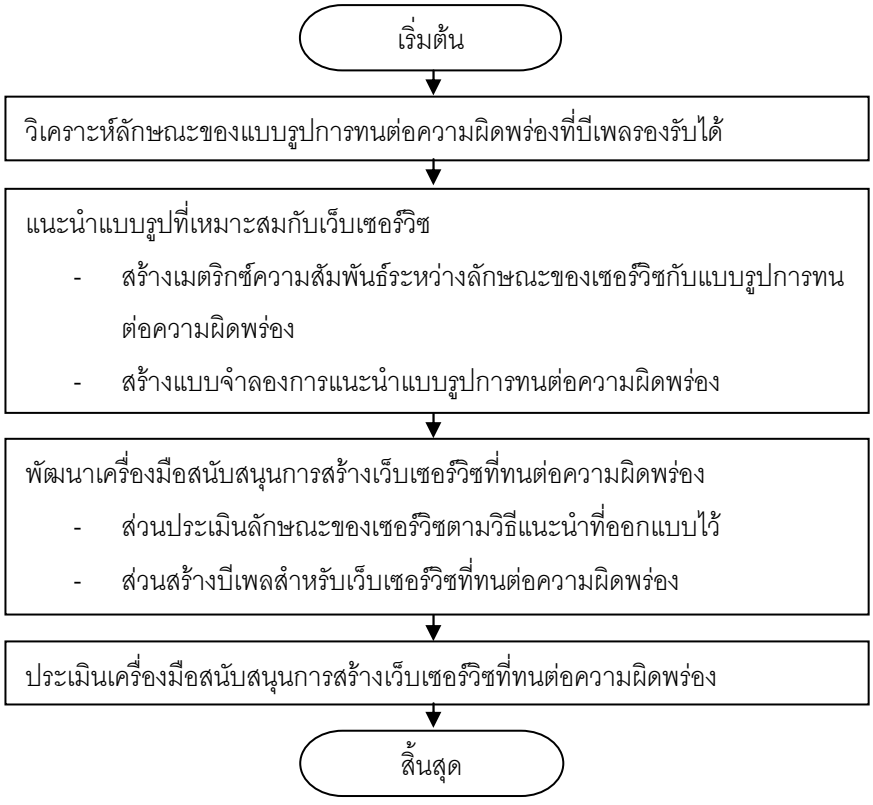
Zeng และคณะ [18] ได้เสนอฟังก์ชันในการวัดคุณภาพของการบริการของเว็บเซอร์วิซในเรื่องต่างๆ ได้แก่ ค่าใช้จ่ายในการประมวลผล (Execution Price) ระยะเวลาในการประมวลผล (Execution Duration) ความเชื่อถือได้ (Reliability) สภาพพร้อมใช้งาน (Availability) ชื่อเสียง (Reputation)

จากงานวิจัยดังกล่าว ทำให้ได้แนวคิดในการนำลักษณะต่างๆ ของคุณภาพของการบริการไปประยุกต์ใช้ในการกำหนดลักษณะต่างๆ ของแบบรูปการทนต่อความผิดพร่องที่เหมาะสมกับเซอร์วิซ ซึ่งมีลักษณะต่างๆ ตามที่ผู้พัฒนาเซอร์วิซกำหนด เพื่อนำไปสู่การแนะนำแบบรูปการทนต่อความผิดพร่องให้กับเซอร์วิซ และจากผลการทดลองของงานวิจัย [11] ซึ่งแสดงคุณภาพของแบบรูปการทนต่อความผิดพร่องในแต่ละแบบ ทำให้สามารถนำผลการทดลองไปใช้ประกอบการพิจารณาลักษณะต่างๆ ของแบบรูปการทนต่อความผิดพร่องในบางแบบรูปได้

บทที่ 3

วิธีดำเนินการวิจัย

งานวิจัยนี้เสนอการพัฒนาเครื่องมือเพื่อช่วยสนับสนุนผู้พัฒนาเว็บเซอร์วิซทางฝั่งผู้ให้บริการ ในการพัฒนาเว็บเซอร์วิซให้สามารถทนต่อความผิดพร่องได้ โดยฟังก์ชันงานหลักของเครื่องมือจะประกอบด้วย (1) การแนะนำผู้พัฒนาเว็บเซอร์วิซเกี่ยวกับแบบรูปการทนต่อความผิดพร่องซึ่งเหมาะสมกับเว็บเซอร์วิซทางฝั่งผู้ให้บริการ โดยใช้แบบจำลองการแนะนำแบบรูปการทนต่อความผิดพร่อง ซึ่งมีการพิจารณาลักษณะของตัวเว็บเซอร์วิซและลักษณะการให้บริการเว็บเซอร์วิซนั้น (2) การสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องตามแบบรูปที่นักพัฒนาเว็บเซอร์วิซเลือกจากข้อ (1) โดยใช้ภาษาบีเพล เครื่องมือนี้จะช่วยสนับสนุนการพัฒนาเว็บเซอร์วิซที่ทนต่อความผิดพร่องได้ ผ่านการประเมินลักษณะต่างๆ ของเว็บเซอร์วิซโดยผู้พัฒนาเอง เครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องมีภาพรวมของขั้นตอนการทำงานดังภาพที่ 3.1



ภาพที่ 3.1 ขั้นตอนการทำงานของเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่อง

3.1. วิเคราะห์ลักษณะของแบบรูปการทนต่อความผิดพร่องที่บีเพลรองรับได้

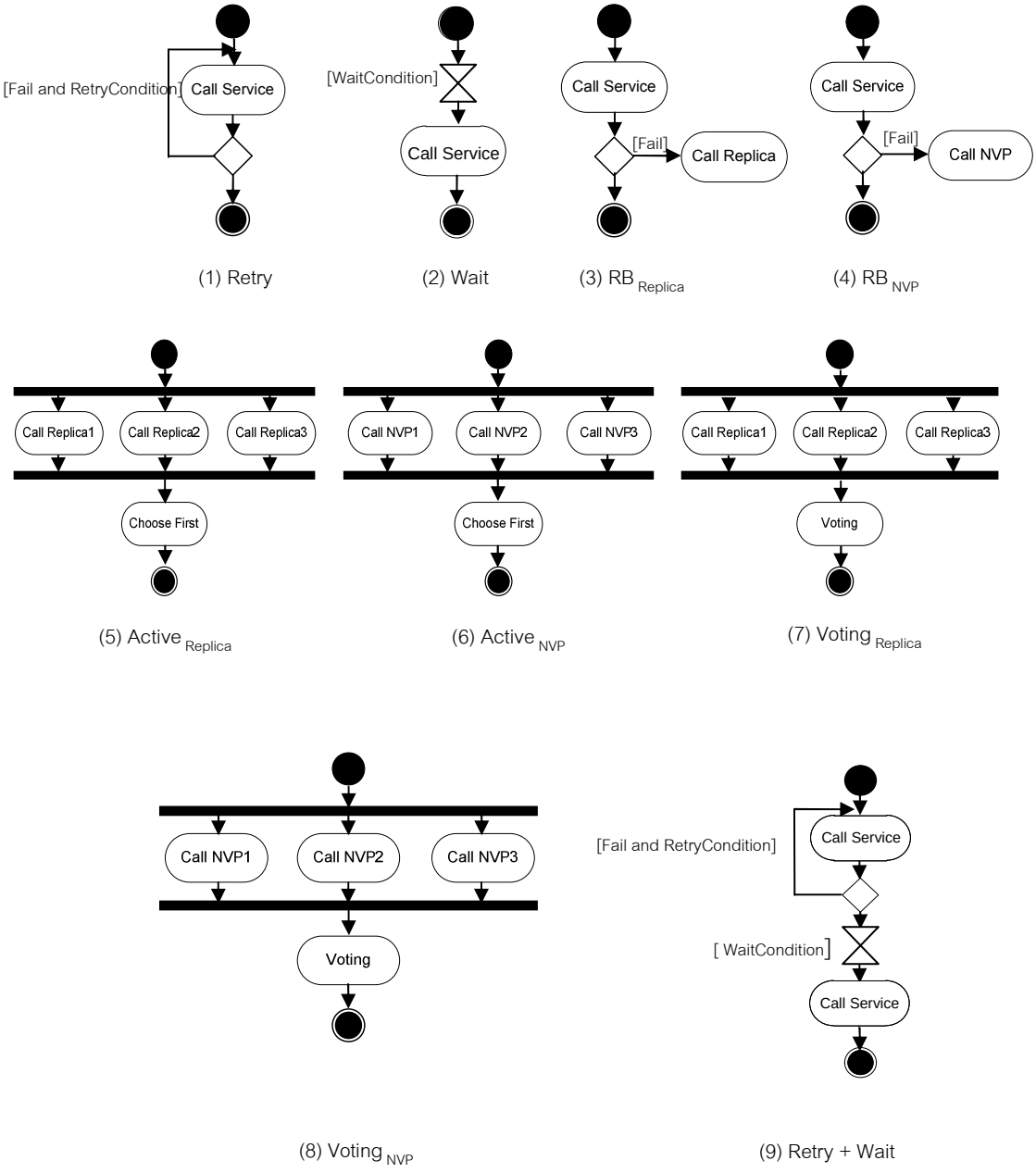
เนื่องจากชุดคำสั่งของดับเบิลยูเอส-บีเพลมีจำนวนค่อนข้างน้อย ทำให้ความสามารถในการออกแบบบีเพลที่ทนต่อความผิดพร่องทำได้ค่อนข้างจำกัด จึงต้องวิเคราะห์หาแบบรูปการทนต่อความผิดพร่องที่ชุดคำสั่งดับเบิลยูเอส-บีเพลสามารถรองรับได้ โดยศึกษาจากงานวิจัย [2] และกระบวนงานที่ใช้รูปแบบ ECA Rule ในการจัดการกับความผิดพร่องแบบต่างๆ [1]

นอกจากนี้ในงานวิจัย [1] ได้เสนอแบบรูปการทนต่อความผิดพร่องที่บีเพลรองรับได้แบบต่างๆ โดยแบ่งแบบรูปการทนต่อความผิดพร่องตามการจัดการกับข้อผิดพร่องออกเป็น 2 กลุ่ม กลุ่มที่ 1 จัดการกับข้อผิดพร่องแบบ Non-Redundancy ได้แก่ แบบรูป Ignore, Notify, Skip, Wait และกลุ่มที่ 2 จัดการกับข้อผิดพร่องแบบ Redundancy ได้แก่ แบบรูป Retry, RetryUntil, Alternate, Replicate ซึ่งได้เสนอให้มีการรวมแบบรูปแบบต่างๆ เข้าด้วยกัน ทำให้มีแบบรูปหลายแบบมาก เพื่อไม่ให้กรอบงานซับซ้อน งานวิจัย [1] จึงได้เสนอให้รวมเฉพาะแบบรูปของกลุ่มที่ 1 และ 2 เข้าด้วยกัน ซึ่งเป็นการเพิ่มความสามารถของการทนต่อความผิดพร่องให้กับเซอร์วิซด้วย ดังนั้นผู้วิจัยจึงได้แนวคิดในการรวมแบบรูปตามแนวคิดของงานวิจัย [1, 11, 12] และได้สรุปแบบรูปการทนต่อความผิดพร่องที่รองรับในงานวิจัยได้ดังนี้

1. Retry – เมื่อเซอร์วิซไม่สามารถใช้งานได้ จะทำการกู้คืน โดยพยายามเรียกเซอร์วิซเดิมใหม่ซ้ำ ตามเงื่อนไขที่กำหนดซึ่งอาจจะเป็นจำนวนรอบหรือเงื่อนไขบางอย่างในการทำงาน และทุกครั้งที่เรียกใหม่ ทุกทรัพยากรยังคงเหมือนเดิม
2. Wait – การรอเรียกเซอร์วิซจนกว่าจะถึงเวลาที่กำหนด ถ้าเซอร์วิซที่เรียกไม่สามารถใช้งานได้หรือมีผู้ใช้งานเป็นจำนวนมาก การเลื่อนเวลาของการเรียกเซอร์วิซออกไป จะช่วยลดโอกาสของการเกิดข้อผิดพร่องได้
3. RecoveryBlock $_{\text{Replica}}$ - เมื่อเซอร์วิซหลักที่ต้องการใช้เกิดข้อผิดพร่อง จะทำการเรียกเซอร์วิซสำรองตามลำดับ ซึ่งเซอร์วิซสำรองจะมีฟังก์ชันการทำงานแบบเดียวกันและพัฒนาเหมือนกัน แต่อยู่ในสภาพแวดล้อมที่ไม่เหมือนกัน
4. RecoveryBlock $_{\text{NVP}}$ – เมื่อเซอร์วิซหลักที่ต้องการใช้เกิดข้อผิดพร่อง จะทำการเรียกเซอร์วิซสำรองตามลำดับ ซึ่งเซอร์วิซสำรองจะมีฟังก์ชันการทำงานแบบเดียวกัน แต่พัฒนาแตกต่างกัน (N-Version Programming: NVP) โดยอาจจะมีทีมพัฒนาคนละทีม หรือมีทีมพัฒนาเดียวกัน แต่พัฒนาด้วยอัลกอริทึมที่แตกต่างกัน
5. Active $_{\text{Replica}}$ – การเรียกเซอร์วิซหลายตัวพร้อมกัน แต่ละตัวมีฟังก์ชันการทำงานแบบเดียวกัน และพัฒนาเหมือนกัน แต่อยู่ในสภาพแวดล้อมที่ไม่เหมือนกัน โดยทำการเลือกผลลัพธ์ของเซอร์วิซตัวแรกสุดที่ทำการตอบกลับเป็นคำตอบของการทำงาน

6. Active $_{NVP}$ – การเรียกเซอร์วิซหลายตัวพร้อมกัน แต่ละตัวมีฟังก์ชันการทำงานแบบเดียวกัน แต่พัฒนาแตกต่างกัน โดยอาจจะมีทีมพัฒนาคนละทีม หรือมีทีมพัฒนาเดียวกันแต่พัฒนาด้วยอัลกอริทึมที่แตกต่างกัน โดยทำการเลือกผลลัพธ์ของเซอร์วิซตัวแรกสุดที่ทำการตอบกลับเป็นคำตอบของการทำงาน
7. Voting $_{Replica}$ – การเรียกเซอร์วิซหลายตัวพร้อมกัน แต่ละตัวมีฟังก์ชันการทำงานแบบเดียวกัน และพัฒนาเหมือนกัน แต่อยู่ในสภาพแวดล้อมที่ไม่เหมือนกัน ซึ่งทำให้ได้ผลลัพธ์หลายค่าจากหลายเซอร์วิซ ดังนั้นจึงต้องมีอัลกอริทึมสำหรับเลือกคำตอบที่จะเป็นคำตอบของการทำงาน
8. Voting $_{NVP}$ – การเรียกเซอร์วิซหลายตัวพร้อมกัน แต่ละตัวมีฟังก์ชันการทำงานแบบเดียวกัน แต่พัฒนาแตกต่างกัน โดยอาจจะมีทีมพัฒนาคนละทีม หรือมีทีมพัฒนาเดียวกันแต่พัฒนาด้วยอัลกอริทึมที่แตกต่างกัน ซึ่งทำให้ได้ผลลัพธ์หลายค่าจากหลายเซอร์วิซ ดังนั้นจึงต้องมีอัลกอริทึมสำหรับเลือกคำตอบที่จะเป็นคำตอบของการทำงาน
9. Retry + Wait – สำหรับแบบรูปนี้ผู้วิจัยเสนอเป็นตัวอย่างของการรวมแบบรูปพื้นฐานกลุ่มที่ 1 และ 2 ตาม [1] เข้าด้วยกัน โดยหากเซอร์วิซไม่สามารถใช้งานได้ จะทำการกู้คืนด้วยวิธี Retry ก่อน ตามเงื่อนไขการเรียกซ้ำที่กำหนด เมื่อมีการเรียกเซอร์วิซครบตามเงื่อนไขแล้ว เซอร์วิซยังไม่สามารถใช้งานได้ จึงจะทำการกู้คืนด้วยวิธี Wait คือ รอจนกว่าจะถึงเวลาตามที่กำหนด แล้วจึงเรียกเซอร์วิซใหม่อีกครั้ง

ลักษณะของแบบรูปการทนต่อความผิดพร่องทั้งหมด 9 แบบ สามารถอธิบายการทำงานได้ดังภาพที่ 3.2



ภาพที่ 3.2 การทำงานของแบบรูปการทนต่อความผิดพร่อง

3.2. แนะนำแบบรูปที่เหมาะสมกับเว็บเซอร์วิซ

การแนะนำแบบรูปการทนต่อความผิดพร่องให้กับเว็บเซอร์วิซประกอบด้วยขั้นตอนต่อไปนี้

3.2.1. สร้างเมตริกซ์ความสัมพันธ์ระหว่างลักษณะของเซอร์วิซกับแบบรูปการทนต่อความผิดพร่อง

แบบรูปที่เหมาะสมขึ้นกับลักษณะของเซอร์วิซ ความหมาย (Semantics) ของเซอร์วิซ และลักษณะการให้บริการเซอร์วิซ ตัวอย่างเช่น เซอร์วิซตรวจสอบยอดเงินคงเหลือในบัญชีธนาคาร ผู้ใช้มีความจำเป็นต้องใช้งานเซอร์วิซตรวจสอบยอดเงินคงเหลือของธนาคารหนึ่ง ไม่สามารถไปใช้เซอร์วิซของธนาคารอื่นที่ไม่ได้เป็นลูกค้าได้ จึงไม่เหมาะสมกับแบบรูปที่ต้องมีการใช้ตัวแทน เช่น RecoveryBlock, Active, Voting เป็นต้น เมื่อตัวแทนเป็นบริการจากธนาคารอื่น ยกเว้นในกรณีที่ธนาคารแห่งนั้นมีการให้บริการสำเนาของเซอร์วิซเอง ดังนั้นในการให้บริการเว็บเซอร์วิซ จึงไม่จำเป็นว่าสามารถจะใช้แบบรูปใดๆ ก็ได้ ทั้งนี้ผู้พัฒนาเว็บเซอร์วิซควรเป็นผู้ประเมินแบบรูปจากลักษณะของเซอร์วิซแต่ละเซอร์วิซ ในการนี้ผู้วิจัยได้เสนอรายการลักษณะของเซอร์วิซ ซึ่งจะส่งผลต่อการเลือกแบบรูปที่เหมาะสม เพราะแบบรูปแต่ละแบบก็เหมาะสมกับการใช้งานในลักษณะที่แตกต่างกันด้วย รายการลักษณะของเซอร์วิซมีดังนี้

1. Transient Failure – ผู้พัฒนาเซอร์วิซสำหรับให้บริการมีความเชื่อมั่นว่าสภาพแวดล้อมการทำงานของเซอร์วิซจะค่อนข้างน่าเชื่อถือ ความขัดข้องของเซอร์วิซจะเกิดขึ้นชั่วคราว ถ้าเรียกเซอร์วิซไม่สำเร็จ ความขัดข้องที่เกิดขึ้น จะหายไปในไม่ช้า เช่น ไม่สามารถเข้าถึงเซอร์วิซได้ เนื่องจากปัญหาทางด้านระบบเครือข่าย ดังนั้นการเรียกเซอร์วิซเดิมอีกครั้งอาจจะทำให้สามารถเข้าถึงเซอร์วิซได้ ถ้าเป็นความขัดข้องที่คงอยู่ไม่นาน การเรียกเซอร์วิซเดิมอีกครั้งอาจทำได้ทันที แต่ถ้าเป็นความผิดพร่องที่คงอยู่ไม่นาน แต่อาจใช้ระยะเวลาช่วงหนึ่งจึงจะสามารถใช้งานเซอร์วิซนั้นได้ อาจหยุดรอเวลาก่อนที่จะเรียกเซอร์วิซ
2. Instance Specificity – เซอร์วิซมีลักษณะการให้บริการที่มีความจำเพาะ ผู้ใช้บริการของเซอร์วิซตัวหนึ่ง ๆ มีความจำเป็นที่ต้องเรียกใช้เซอร์วิซตัวนั้นโดยเฉพาะ เช่น มีผู้ให้บริการเซอร์วิซ (Service Provider) เจ้าเดียวเท่านั้นที่ให้บริการได้ หรือเซอร์วิซตัวนั้นเก็บข้อมูล (State) ของการให้บริการผู้ใช้ไว้ จึงทำให้ผู้ใช้จำเป็นต้องเฉพาะเจาะจงในการใช้งานเซอร์วิซ (ผ่านวิสเดิล) ตัวนั้น ตัวอย่างเช่น กรณีของเซอร์วิซของธนาคารที่ได้กล่าวไปข้างต้น
3. Replica Provision – ผู้พัฒนาเซอร์วิซหรือผู้ให้บริการสามารถพัฒนาสำเนาของเซอร์วิซซึ่งทำงานแบบเดียวกันและพัฒนาเหมือนกัน แต่อาจจะอยู่ในสภาพแวดล้อมที่แตกต่างกันได้ เช่น เว็บเซอร์วิซที่มีสำเนาไว้ แต่ประมวลผลด้วยหน่วยประมวลผลที่แตกต่างกัน หรือ

เก็บข้อมูลไว้ในหน่วยเก็บข้อมูลที่แตกต่างกัน เป็นต้น การเรียกใช้สำเนาต่าง ๆ ของเซอร์วิซอาจจะให้ผลแตกต่างกัน จากการที่สภาพแวดล้อมในการทำงานสามารถแตกต่างกันได้ จึงส่งผลต่อคุณภาพการให้บริการหรือคิวโอเอสของเซอร์วิซได้

4. NVP Provision – ผู้พัฒนาเซอร์วิซหรือผู้ให้บริการมีเซอร์วิซสำรองที่มีการพัฒนาแตกต่างกัน แต่สามารถทำงานแบบเดียวกันได้ โดยอาจจะพัฒนาด้วยผู้พัฒนาคนละทีม หรือทีมเดียวกัน แต่พัฒนาด้วยอัลกอริทึมคนละชุด โดยจะทำงานในสภาพแวดล้อมที่เหมือนกันหรือแตกต่างกันก็ได้
5. Correctness – ผู้พัฒนาเซอร์วิซหรือผู้ให้บริการคาดหวังว่า เซอร์วิซที่ใช้สามารถให้ผลลัพธ์ของการทำงานที่ถูกต้องเสมอ มีความเชื่อถือได้ โดยดูได้จากประสบการณ์ หรือประวัติการให้บริการย้อนหลัง หรือการประเมินสภาพแวดล้อมของการให้บริการ
6. Timeliness – ผู้พัฒนาเซอร์วิซหรือผู้ให้บริการคาดหวังว่า เซอร์วิซที่ใช้สามารถให้ผลลัพธ์ของการทำงานได้ทันเวลาที่ต้องการ หรือใช้เวลาในการทำงานน้อย เซอร์วิซตอบกลับผลลัพธ์ได้อย่างรวดเร็ว โดยดูได้จากประสบการณ์ หรือประวัติการให้บริการย้อนหลัง หรือการประเมินสภาพแวดล้อมของการให้บริการ
7. Simplicity – ผู้พัฒนาเซอร์วิซหรือผู้ให้บริการต้องการให้เซอร์วิซที่ออกแบบให้ทนต่อความผิดพร่องไม่ซับซ้อน ทำงานง่าย เนื่องจากความซับซ้อนในเซอร์วิซจะทำให้การตรวจสอบความถูกต้องทำได้ยาก และการเพิ่มตรรกะการทำงานมากเกินไป อาจทำให้เกิดความผิดพร่องอย่างอื่นเพิ่มขึ้น ซึ่งการออกแบบเซอร์วิซให้ทำงานง่าย เช่น การเขียนโปรแกรมที่มีจำนวนบรรทัดของโปรแกรมน้อย เนื่องจากโปรแกรมที่มีจำนวนบรรทัดน้อยจะมีจำนวนข้อผิดพร่องน้อยกว่า ดูแลรักษาและแก้ไขโปรแกรมได้ง่ายกว่าโปรแกรมที่มีจำนวนบรรทัดมาก
8. Economy – ผู้พัฒนาเซอร์วิซหรือผู้ให้บริการคำนึงถึงค่าใช้จ่ายที่เซอร์วิซใช้ในการประมวลผล ในขณะที่กู้คืน ทั้งค่าใช้จ่ายของทรัพยากร เช่น หน่วยความจำ หน่วยประมวลผล เป็นต้น และค่าใช้จ่ายในการพัฒนาเซอร์วิซสำรอง ทั้งนี้แบบรูปการทนต่อความผิดพร่องจะส่งผลต่อค่าใช้จ่าย แบบรูปที่เรียกเซอร์วิซตามลำดับ จะมีค่าใช้จ่ายในการประมวลผลน้อยกว่าแบบรูปที่มีการเรียกเซอร์วิซหลายตัวพร้อมกันเป็นแบบขนาน และแบบรูปที่มีเซอร์วิซสำรองที่พัฒนาเหมือนกัน จะมีค่าใช้จ่ายในการพัฒนาเซอร์วิซน้อยกว่าแบบรูปที่มีเซอร์วิซสำรองที่พัฒนาแตกต่างกัน

ผู้วิจัยเสนอเมตริกซ์ความสัมพันธ์ระหว่างลักษณะของเซอร์วิซกับแบบรูปการทนต่อความผิดพร่อง (R) ดังตารางที่ 3.1 ซึ่งแต่ละแบบรูปจะตอบสนองต่อลักษณะต่างๆ ของเซอร์วิซได้แตกต่างกันไป โดยกำหนดให้แนวนอนของเมตริกซ์เป็นลักษณะของเซอร์วิซ และแนวตั้งของเมตริกซ์เป็นแบบรูปการทนต่อความผิดพร่อง ในเมตริกซ์จะกำหนดค่าน้ำหนักตามลำดับการตอบสนองต่อลักษณะต่าง ๆ ของเซอร์วิซของแบบรูปการทนต่อความผิดพร่อง ค่าน้ำหนักมีค่าอยู่ระหว่าง 0 ถึง 8 เนื่องจากจะจัดลำดับค่าน้ำหนักจากมากไปน้อยตามจำนวนแบบรูปการทนต่อความผิดพร่องพื้นฐานซึ่งมีทั้งหมด 8 แบบ โดยค่าน้ำหนักที่มากที่สุด คือ 8 หมายความว่า แบบรูปการทนต่อความผิดพร่องนั้นแสดงลักษณะดังกล่าวมากที่สุด และค่าน้ำหนักเป็น 7 หมายความว่า แบบรูปการทนต่อความผิดพร่องนั้นแสดงลักษณะดังกล่าวรองลงมา และเรียงลำดับค่าน้ำหนักเป็นเช่นนี้ไปเรื่อยๆ และถ้าค่าน้ำหนักเป็น 0 แสดงว่า แบบรูปการทนต่อความผิดพร่องนั้น ไม่มีลักษณะดังกล่าว การให้ค่าน้ำหนักมีแนวทางในการพิจารณาดังนี้

ตารางที่ 3.1 เมตริกซ์ความสัมพันธ์ระหว่างลักษณะของเซอร์วิซกับแบบรูปการทนต่อความผิดพร่อง

Service Characteristics	Fault Tolerance Patterns								
	Retry	Wait	RB Replica	RB NVP	Active Replica	Active NVP	Voting Replica	Voting NVP	Retry + Wait
Transient Failure(TF)	8	7	0	0	0	0	0	0	7.5
Instance Specificity(IS)	8	8	7	6	5	4	5	4	8
Replica Provision(RP)	0	0	8	0	8	0	8	0	0
NVP Provision(NP)	0	0	0	8	0	8	0	8	0
Correctness (CO)	2	2	3	4	5	6	7	8	2
Timeliness (TI)	4	1	5	6	7	8	2	3	2.5
Simplicity(SI)	8	8	7	6	5	4	3	2	8
Economy(EC)	7	8	6	5	4	3	2	1	7.5

1. ลักษณะของ Transient Failure เป็นลักษณะที่ผู้พัฒนาเซอร์วิซมีความเชื่อมั่นว่าสภาพแวดล้อมการทำงานของเซอร์วิซจะค่อนข้างน่าเชื่อถือ ความขัดข้องของเซอร์วิซจะเกิดขึ้นชั่วคราว ถ้าเรียกเซอร์วิซไม่สำเร็จ ความขัดข้องที่เกิดขึ้น จะหายไปในไม่ช้า ในการกำหนดค่าน้ำหนักจะพิจารณาให้

(1) แบบรูป Retry ให้ค่าน้ำหนักสูงสุดคือ 8 เนื่องจากเป็นความผิดพร่องที่เกิดขึ้นชั่วคราว ความขัดข้องจะหายไปในไม่ช้า ดังนั้นการเรียกเซอร์วิซซ้ำ จะทำให้เรียกเซอร์วิซได้สำเร็จ

(2) แบบรูป Wait ให้ค่าน้ำหนักเท่ากับ 7 เนื่องจากกรณีที่เป็นความผิดพร่องที่คงอยู่ไม่นาน แต่อาจใช้ระยะเวลาช่วงหนึ่งจึงจะสามารถใช้งานเซอร์วิซนั้นได้ อาจหยุดรอเวลาก่อน จนกระทั่งถึงกำหนดเวลาที่จะเรียกเซอร์วิซ จึงทำการเรียกเซอร์วิซนั้น

(3) แบบรูป Retry+Wait ค่าน้ำหนักจะเกิดจากการนำค่าน้ำหนักของแบบรูป Retry และแบบรูป Wait มาหาค่าเฉลี่ยกัน ซึ่งจะได้ค่าน้ำหนักเท่ากับ (8+7)/2 = 7.5

(4) แบบรูปอื่น มีการทำงานที่ไม่ได้รองรับลักษณะนี้ ค่าน้ำหนักจึงเท่ากับ 0

2. ลักษณะของ Instance Specificity เป็นลักษณะที่ผู้พัฒนาเซอร์วิซมีความจำเป็นที่ต้องเรียกใช้เซอร์วิซตัวนั้นโดยเฉพาะ ไม่สามารถเรียกเซอร์วิซอื่นแทนได้ ในการกำหนดค่าน้ำหนักจะพิจารณาให้

(1) แบบรูป Retry และ Wait ได้ค่าน้ำหนักสูงสุดคือ 8 เนื่องจากเป็นแบบรูปที่จัดการกับความผิดพร่องโดยไม่ได้เรียกเซอร์วิซตัวแทน แต่จะเรียกเซอร์วิซเดิมซ้ำ

(2) แบบรูป $RB_{Replica}$ ให้ค่าน้ำหนักเท่ากับ 7 เนื่องจากเป็นการเรียกเซอร์วิซหลัก แต่ถ้าเกิดข้อผิดพร่องขึ้น จะทำการเรียกเซอร์วิซตัวแทนที่มีการพัฒนาเหมือนกัน ซึ่งถือว่าเป็นเซอร์วิซเดิม แต่อยู่ในสภาพแวดล้อมที่ไม่เหมือนกัน

(3) แบบรูป RB_{NVP} ให้ค่าน้ำหนักเท่ากับ 6 เนื่องจากเป็นการเรียกเซอร์วิซหลัก ซึ่งถ้าเรียกเซอร์วิซสำเร็จ จะไม่ต้องเรียกเซอร์วิซตัวแทน แต่ถ้าเกิดข้อผิดพร่องขึ้น จะทำการเรียกเซอร์วิซตัวแทนที่มีฟังก์ชันการทำงานเหมือนกัน แต่พัฒนาแตกต่างกัน โดยอาจจะมีทีมพัฒนาคนละทีม หรือมีทีมพัฒนาเดียวกัน แต่พัฒนาด้วยอัลกอริทึมที่แตกต่างกัน ซึ่งในการเรียกใช้เซอร์วิซจะคำนึงถึงฟังก์ชันการทำงานที่ให้ผลลัพธ์เหมือนกันมากกว่าที่จะคำนึงว่าเซอร์วิซถูกพัฒนาแตกต่างกันหรือไม่

(4) แบบรูป $Active_{Replica}$ และ $Voting_{Replica}$ ให้ค่าน้ำหนักเท่ากับ 5 เท่ากัน เนื่องจากแบบรูปทั้งสองจะเรียกเซอร์วิซหลายตัวที่พัฒนาเหมือนกัน แต่อยู่ในสภาพแวดล้อมที่ไม่เหมือนกัน พร้อม

กัน ซึ่งจะเห็นว่า มีการเรียกเซอร์วิซหลายตัว ไม่ได้เจาะจงเลือกเซอร์วิซตัวใดตัวหนึ่ง แต่เซอร์วิซหลายตัวมีการพัฒนาเหมือนกัน จึงถือว่าเป็นเซอร์วิซเดียวกัน

(5) แบบรูป $Active_{NVP}$ และ $Voting_{NVP}$ ให้ค่าน้ำหนักเท่ากับ 4 เท่ากัน เนื่องจากแบบรูปทั้งสองจะเรียกเซอร์วิซหลายตัวที่มีฟังก์ชันการทำงานแบบเดียวกัน แต่พัฒนาไม่เหมือนกัน พร้อมกันจะเห็นว่า ถึงแม้จะไม่ใช่เซอร์วิซเดียวกัน แต่เซอร์วิซยังสามารถทำงานแบบเดียวกันได้อยู่ จึงพอที่จะรองรับการทำงานของลักษณะนี้ได้

(6) แบบรูป Retry+Wait ค่าน้ำหนักจะเกิดจากการนำค่าน้ำหนักของแบบรูป Retry และแบบรูป Wait มาหาค่าเฉลี่ยกัน ซึ่งจะได้ค่าน้ำหนักเท่ากับ (8+8)/2 = 8

3. ลักษณะของ Replica Provision เป็นลักษณะที่ผู้พัฒนาเซอร์วิซมีการสำเนาเซอร์วิซ ซึ่งทำงานแบบเดียวกันและพัฒนาเหมือนกัน แต่อาจจะอยู่ในสภาพแวดล้อมที่แตกต่างกัน ในการกำหนดค่าน้ำหนักจะพิจารณาให้

(1) แบบรูป $RB_{Replica}$, $Active_{Replica}$ และ $Voting_{Replica}$ ให้ค่าน้ำหนักเท่ากับ 8

(2) แบบรูปอื่น มีการทำงานที่ไม่ได้รองรับลักษณะนี้ ค่าน้ำหนักจึงเท่ากับ 0

4. ลักษณะของ NVP Provision เป็นลักษณะที่ผู้พัฒนาเซอร์วิซมีเซอร์วิซสำรองที่มีการพัฒนาแตกต่างกัน แต่สามารถทำงานแบบเดียวกันได้ โดยอาจจะพัฒนาด้วยผู้พัฒนาคนละทีมหรือทีมเดียวกัน แต่พัฒนาด้วยอัลกอริทึมคนละชุด โดยจะทำงานในสภาพแวดล้อมที่เหมือนกันหรือแตกต่างกันก็ได้ ในการกำหนดค่าน้ำหนักจะพิจารณาให้

(1) แบบรูป RB_{NVP}, $Active_{NVP}$ และ $Voting_{NVP}$ ให้ค่าน้ำหนักเท่ากับ 8

(2) แบบรูปอื่น มีการทำงานที่ไม่ได้รองรับลักษณะนี้ ค่าน้ำหนักจึงเท่ากับ 0

5. ลักษณะของ Correctness ซึ่งเป็นลักษณะที่ผู้พัฒนาเซอร์วิซคาดหวังว่า เซอร์วิซที่ใช้สามารถให้ผลลัพธ์ของการทำงานที่ถูกต้องเสมอ มีความเชื่อถือได้ ในการกำหนดค่าน้ำหนักจะพิจารณาให้

(1) แบบรูป $Voting_{NVP}$ ได้ค่าน้ำหนักสูงสุดคือ 8 เนื่องจากเป็นแบบรูปที่สามารถป้องกันความขัดข้องแบบ Byzantine คือ กรณีเซอร์วิซให้ผลลัพธ์ที่ไม่ถูกต้อง เนื่องจากแบบรูป $Voting_{NVP}$ จะเรียกเซอร์วิซหลายตัวพร้อมกันแบบขนาน โดยเซอร์วิซแต่ละตัวมีฟังก์ชันการทำงานแบบเดียวกัน แต่พัฒนาแตกต่างกัน ดังนั้นโอกาสที่จะได้ผลลัพธ์ของการทำงานที่ผิด เนื่องจากการเรียกเซอร์วิซใดเซอร์วิซหนึ่งหรือการพัฒนาแบบใดแบบหนึ่งที่มีข้อผิดพลาดจะลดลง และเนื่องจากการ

เรียกเซอร์วิซหลายตัวพร้อมกัน ทำให้ได้ผลลัพธ์หลายค่า ดังนั้นจึงต้องมีอัลกอริทึมสำหรับเลือกคำตอบ เพื่อให้ได้คำตอบที่ถูกที่สุด

(2) แบบรูป $Voting_{Replica}$ ให้ค่าน้ำหนักเท่ากับ 7 เนื่องจากเป็นแบบรูปที่เรียกเซอร์วิซหลายตัวพร้อมกันแบบขนานเหมือนกัน แต่เซอร์วิซแต่ละตัวเป็นสำเนาของเซอร์วิซที่ถูกพัฒนาเหมือนกัน แต่มีสภาพแวดล้อมการทำงานที่แตกต่างกัน ดังนั้นโอกาสที่จะได้คำตอบที่ไม่ถูกต้องจากการโหวตผลลัพธ์ของเซอร์วิซ จึงมีมากกว่าแบบ $Voting_{NVP}$ หากข้อผิดพลาดมีสาเหตุมาจากการพัฒนาเซอร์วิซ

(3) แบบรูป $Active_{NVP}$ ให้ค่าน้ำหนักเท่ากับ 6 เนื่องจากเป็นแบบรูปที่เรียกเซอร์วิซหลายตัวพร้อมกันแบบขนานเหมือนกัน แต่ละตัวมีฟังก์ชันการทำงานแบบเดียวกัน แต่พัฒนาแตกต่างกัน ทำให้ได้ผลลัพธ์หลายค่า โดยทำการเลือกผลลัพธ์ของเซอร์วิซตัวแรกสุดที่ทำการตอบกลับเป็นคำตอบของการทำงาน ซึ่งมีโอกาสของการได้ผลลัพธ์ที่ไม่ถูกต้องน้อยกว่าการเรียกเซอร์วิซเดียว

(4) แบบรูป $Active_{Replica}$ ให้ค่าน้ำหนักเท่ากับ 5 เนื่องจากเป็นแบบรูปที่เรียกเซอร์วิซหลายตัวพร้อมกันแบบขนานเหมือนกัน แต่ละตัวมีการพัฒนาแบบเดียวกัน แต่มีสภาพแวดล้อมการทำงานที่แตกต่างกัน ทำให้ได้ผลลัพธ์หลายค่า แล้วทำการเลือกผลลัพธ์ของเซอร์วิซตัวแรกสุดที่ทำการตอบกลับเป็นคำตอบของการทำงาน จึงมีโอกาสของการได้ผลลัพธ์ที่ไม่ถูกต้องน้อยกว่าการเรียกเซอร์วิซเดียว แต่มากกว่าแบบ $Active_{NVP}$ หากข้อผิดพลาดมีสาเหตุมาจากการพัฒนาเซอร์วิซ

(5) แบบรูป RB_{NVP} ให้ค่าน้ำหนักเป็น 4 เนื่องจากเป็นแบบรูปที่เมื่อเรียกเซอร์วิซหลักไม่สำเร็จ จะทำการเรียกเซอร์วิซตัวแทนที่มีการพัฒนาแตกต่างกัน ดังนั้นโอกาสที่จะให้ผลลัพธ์ถูกต้องจะมากกว่าการเรียกเซอร์วิซเดียว

(6) แบบรูป $RB_{Replica}$ ให้ค่าน้ำหนักเท่ากับ 3 เนื่องจากเป็นแบบรูปที่เมื่อเรียกเซอร์วิซหลักไม่สำเร็จ จะทำการเรียกเซอร์วิซตัวแทนที่มีการพัฒนาเหมือนกัน แต่อยู่ในสภาพแวดล้อมการทำงานที่แตกต่างกัน ดังนั้นโอกาสที่จะให้ผลลัพธ์ถูกต้องจะมากกว่าการเรียกเซอร์วิซเดียว แต่เนื่องจากเป็นเซอร์วิซที่พัฒนาเหมือนกัน ดังนั้นถ้าเป็นความผิดพร่องที่เกิดจากการพัฒนาเซอร์วิซจะทำให้ทั้งเซอร์วิซหลักและเซอร์วิซตัวแทนที่พัฒนาเหมือนกัน เกิดความผิดพร่องทั้งคู่ แบบรูปนี้จึงได้ค่าน้ำหนักน้อยกว่าแบบรูป RB_{NVP}

(7) แบบรูป Retry และแบบรูป Wait ให้ค่าน้ำหนักเท่ากับ 2 เหมือนกัน เนื่องจากแบบรูปทั้งสองมีการเรียกเซอร์วิซเดียว โอกาสที่จะให้ผลลัพธ์ถูกต้องจึงน้อยกว่าแบบรูปที่มีการเรียกหลายเซอร์วิซ

(8) แบบรูป Retry+Wait ค่าน้ำหนักจะเกิดจากการนำค่าน้ำหนักของแบบรูป Retry และแบบรูป Wait มาหาค่าเฉลี่ยกัน ซึ่งจะได้ค่าน้ำหนักเท่ากับ (2+2)/2 = 2

6. ลักษณะของ Timeliness เป็นลักษณะที่ผู้พัฒนาเซอร์วิซคาดหวังว่า เซอร์วิซที่ใช้สามารถให้ผลลัพธ์ของการทำงานได้ทันเวลาที่ต้องการ หรือใช้เวลาในการทำงานน้อย เซอร์วิซตอบกลับผลลัพธ์ได้อย่างรวดเร็ว ในการกำหนดค่าน้ำหนักจะพิจารณาให้

(1) แบบรูป $Active_{NVP}$ ให้ค่าน้ำหนักสูงสุด คือ 8 เนื่องจากเป็นแบบรูปที่เรียกเซอร์วิซหลายตัวที่พัฒนาแตกต่างกันพร้อมกันแบบขนาน และมีอัลกอริทึมในการเลือกคำตอบของเซอร์วิซตัวแรกสุดที่ทำการตอบกลับ ซึ่งเซอร์วิซที่พัฒนาแตกต่างกัน จะส่งผลในเรื่องเวลาที่ใช้ในการประมวลผล เช่น จำนวนบรรทัดของโค้ดที่แตกต่างกัน วิธีการเขียนโปรแกรมที่แตกต่างกัน เป็นต้น ดังนั้นโอกาสที่เซอร์วิซจะตอบกลับไม่พร้อมกันจะมีมากกว่าเซอร์วิซที่มีการพัฒนาเหมือนกัน

(2) แบบรูป $Active_{Replica}$ ให้ค่าน้ำหนักเท่ากับ 7 เนื่องจากเป็นแบบรูปที่เรียกเซอร์วิซหลายตัวที่พัฒนาเหมือนกันพร้อมกันแบบขนาน และมีอัลกอริทึมในการเลือกคำตอบของเซอร์วิซตัวแรกสุดที่ทำการตอบกลับ แต่เนื่องจากมีการพัฒนาเซอร์วิซเหมือนกัน ดังนั้นเวลาในการตอบกลับจึงไม่แตกต่างกันมาก

(3) แบบรูป RB_{NVP} ให้ค่าน้ำหนักเท่ากับ 6 เนื่องจากเป็นแบบรูปที่มีการเรียกเซอร์วิซหลัก ถ้าเซอร์วิซหลักมีข้อผิดพร่อง จะทำการเรียกเซอร์วิซตัวแทนที่พัฒนาแตกต่างกันทันที ดังนั้นโอกาสที่จะได้ผลลัพธ์ตอบกลับอย่างรวดเร็วจึงมากกว่าการเรียกเซอร์วิซหลักที่มีความผิดพร่องเพียงตัวเดียว และเนื่องจากเซอร์วิซตัวแทนมีการพัฒนาแตกต่างกัน ถ้าความผิดพร่องของเซอร์วิซหลักเกิดจากการพัฒนาเซอร์วิซ การเรียกเซอร์วิซตัวแทนที่พัฒนาแตกต่างกัน จึงอาจทำให้การตอบกลับได้เร็วกว่าเซอร์วิซที่พัฒนาเหมือนกัน

(4) แบบรูป $RB_{Replica}$ ให้ค่าน้ำหนักเท่ากับ 5 เนื่องจากเป็นแบบรูปที่มีการเรียกเซอร์วิซหลัก ถ้าเซอร์วิซหลักมีข้อผิดพร่อง จะทำการเรียกเซอร์วิซตัวแทนที่พัฒนาเหมือนกันทันที ดังนั้นโอกาสที่จะได้ผลลัพธ์ตอบกลับอย่างรวดเร็วจึงมากกว่าการเรียกเซอร์วิซหลักตัวเดียว แต่เนื่องจากเซอร์วิซตัวแทนมีการพัฒนาเหมือนกัน โอกาสที่เซอร์วิซจะตอบกลับเร็วกว่าแบบรูป RB_{NVP} จึงน้อยกว่า

(5) แบบรูป Retry ให้ค่าน้ำหนักเท่ากับ 4 เนื่องจากเป็นแบบรูปที่เรียกเซอร์วิซเดียว ถ้าเซอร์วิซเกิดข้อผิดพร่อง จะทำการเรียกเซอร์วิซเดิมซ้ำ ถ้าเซอร์วิซไม่สามารถใช้งานได้ จะทำให้เซอร์วิซเรียกซ้ำตามเงื่อนไขที่กำหนด ทำให้เวลาที่ใช้ในการตอบกลับมาก

(6) แบบรูป $Voting_{NVP}$ ให้ค่าน้ำหนักเท่ากับ 3 เนื่องจากเป็นแบบรูปที่เรียกเซอร์วิซหลายตัวที่พัฒนาแตกต่างกันแบบขนาน และใช้อัลกอริทึมสำหรับเลือกคำตอบ ซึ่งการรอคำตอบจากเซอร์วิซหลายตัว และการประมวลผลของอัลกอริทึมต้องใช้เวลานาน เนื่องจากเซอร์วิซถูกพัฒนาแตกต่างกัน ถ้าความผิดพร่องของเซอร์วิซเกิดจากการพัฒนาเซอร์วิซ การเรียกเซอร์วิซหลายตัวที่พัฒนาแตกต่างกัน จึงทำให้การตอบกลับได้เร็วกว่าเซอร์วิซที่พัฒนาเหมือนกัน

(7) แบบรูป $Voting_{Replica}$ ให้ค่าน้ำหนักเท่ากับ 2 เนื่องจากเป็นแบบรูปที่เรียกเซอร์วิซหลายตัวที่พัฒนาแตกต่างกันแบบขนาน และใช้อัลกอริทึมสำหรับเลือกคำตอบ ซึ่งการรอคำตอบจากเซอร์วิซหลายตัว และการประมวลผลของอัลกอริทึมต้องใช้เวลานาน เนื่องจากเซอร์วิซถูกพัฒนาเหมือนกัน ถ้าความผิดพร่องของเซอร์วิซเกิดจากการพัฒนาเซอร์วิซ การเรียกเซอร์วิซหลายตัวที่พัฒนาเหมือนกัน โอกาสที่เซอร์วิซจะตอบกลับค่อนข้างน้อย ทำให้ตอบกลับผลลัพธ์ได้ช้า

(8) แบบรูป Wait ให้ค่าน้ำหนักเท่ากับ 1 เนื่องจากเป็นแบบรูปที่จะรอเรียกเซอร์วิซจนกว่าจะถึงเวลาที่กำหนด จึงจะทำการเรียกเซอร์วิซ ดังนั้นจึงใช้เวลาในการตอบกลับมากที่สุด

(9) แบบรูป Retry+Wait ค่าน้ำหนักจะเกิดจากการนำค่าน้ำหนักของแบบรูป Retry และแบบรูป Wait มาหาค่าเฉลี่ยกัน ซึ่งจะได้ค่าน้ำหนักเท่ากับ (4+1)/2 = 2.5

เมื่อเปรียบเทียบจากประสิทธิภาพในด้านเวลาของงานวิจัย [2],[3] ซึ่งเรียงลำดับจากแบบรูปที่ใช้เวลาน้อยไปหามาก คือ Active, RB, Retry, Voting และ Wait จึงสอดคล้องกับการพิจารณาให้ค่าน้ำหนักกับแบบรูปต่างๆ

7. ลักษณะของ Simplicity เป็นลักษณะที่ผู้พัฒนาเซอร์วิซต้องการให้เซอร์วิซที่ออกแบบให้ทนต่อความผิดพร่องไม่ซับซ้อน ทำงานง่าย ในการกำหนดค่าน้ำหนักจะพิจารณาให้

(1) แบบรูป Retry และแบบรูป Wait ให้ค่าน้ำหนักสูงสุดคือ 8 เนื่องจากเป็นแบบรูปที่เรียกเซอร์วิซเดียว จึงพัฒนาได้ง่ายกว่าแบบรูปอื่นๆ

(2) แบบรูป $RB_{Replica}$ ให้ค่าน้ำหนักเท่ากับ 7 เนื่องจากเป็นแบบรูปที่เรียกเซอร์วิซหลักก่อน ถ้าเกิดความขัดข้อง จะทำการเรียกเซอร์วิซตัวแทนที่พัฒนาเหมือนกัน ซึ่งไม่ได้เรียกเซอร์วิซหลายตัวพร้อมกัน ประกอบกับเซอร์วิซตัวแทนมีการพัฒนาเหมือนกัน จึงพัฒนาง่ายกว่าเซอร์วิซที่พัฒนาไม่เหมือนกัน

(3) แบบรูป RB_{NVP} ให้ค่าน้ำหนักเท่ากับ 6 เหตุผลเหมือนกับแบบรูป $RB_{Replica}$ แต่แบบรูปนี้เซอร์วิซตัวแทนมีการพัฒนาแตกต่างกัน แต่มีฟังก์ชันการทำงานเหมือนกัน จึงพัฒนาได้ยากกว่าแบบรูป $RB_{Replica}$

(4) แบบรูป $Active_{Replica}$ ให้ค่าน้ำหนักเท่ากับ 5 เนื่องจากเป็นแบบรูปที่เรียกเซอร์วิซหลายตัวพร้อมกัน ซึ่งเซอร์วิซจะมีการพัฒนาเหมือนกัน และเลือกเซอร์วิซตัวแรกสุดที่ตอบกลับ ดังนั้นการเรียกเซอร์วิซหลายตัว จึงมีการทำงานที่ซับซ้อนมากกว่าการเรียกเซอร์วิซตัวเดียว แต่อัลกอริทึมที่ใช้ในการเลือกคำตอบทำงานง่าย ไม่ซับซ้อน เนื่องจากเลือกจากเซอร์วิซที่ให้ผลลัพธ์เร็วที่สุด และเซอร์วิซที่เรียกมีการพัฒนาที่เหมือนกัน จึงพัฒนาง่ายกว่าเซอร์วิซที่พัฒนาแตกต่างกัน

(5) แบบรูป Active$_{NVP}$ ให้ค่าน้ำหนักเท่ากับ 4 เหตุผลเหมือนกับแบบรูป Active$_{Replica}$ แต่แบบรูปนี้เซอร์วิซที่เรียกมีการพัฒนาแตกต่างกัน จึงพัฒนาได้ยากกว่าแบบรูป Active$_{Replica}$

(6) แบบรูป Voting$_{Replica}$ ให้ค่าน้ำหนักเท่ากับ 3 เนื่องจากเป็นแบบรูปที่เรียกเซอร์วิซหลายตัวพร้อมกัน จึงมีการทำงานที่ซับซ้อนมากกว่าการเรียกเซอร์วิซตัวเดียว และการมีอัลกอริทึมสำหรับเลือกคำตอบจากผลลัพธ์ทั้งหมด ทำให้อัลกอริทึมซับซ้อนกว่าแบบ Active$_{Replica}$ ซึ่งเซอร์วิซที่เรียกจะมีการพัฒนาเหมือนกัน จึงพัฒนาได้ง่ายกว่าเซอร์วิซที่พัฒนาแตกต่างกัน

(7) แบบรูป Voting$_{NVP}$ ให้ค่าน้ำหนักเท่ากับ 2 เหตุผลเหมือนกับแบบรูป Voting$_{Replica}$ แต่แบบรูปนี้ เซอร์วิซที่เรียกจะมีการพัฒนาแตกต่างกัน จึงพัฒนาได้ยากกว่าแบบรูป Voting$_{Replica}$

(8) แบบรูป Retry+Wait ค่าน้ำหนักจะเกิดจากการนำค่าน้ำหนักของแบบรูป Retry และแบบรูป Wait มาหาค่าเฉลี่ยกัน ซึ่งจะได้ค่าน้ำหนักเท่ากับ (8+8)/2 = 8

8. ลักษณะของ Economy เป็นลักษณะที่ผู้พัฒนาเซอร์วิซคำนึงถึงค่าใช้จ่ายที่เซอร์วิซใช้ในการประมวลผล ทั้งค่าใช้จ่ายของทรัพยากรและค่าใช้จ่ายในการพัฒนาเซอร์วิซสำรอง ในการกำหนดค่าน้ำหนักจะพิจารณาให้

(1) แบบรูป Wait ให้ค่าน้ำหนักสูงสุดคือ 8 เนื่องจากเป็นแบบรูปที่ไม่มีค่าใช้จ่ายในการพัฒนาเซอร์วิซสำรองและมีค่าใช้จ่ายในการประมวลผลน้อย เพราะเป็นการเรียกเซอร์วิซเดี่ยว โดยจะมีการรอจนกว่าจะครบตามเวลาที่กำหนด จึงจะทำการเรียกเซอร์วิซเพียงครั้งเดียว

(2) แบบรูป Retry ให้ค่าน้ำหนักเท่ากับ 7 เนื่องจากเป็นแบบรูปที่ไม่มีค่าใช้จ่ายในการพัฒนาเซอร์วิซสำรองและมีค่าใช้จ่ายในการประมวลผลน้อย เพราะเป็นการเรียกเซอร์วิซเดี่ยว ตามจำนวนครั้งหรือเงื่อนไขที่กำหนด

(3) แบบรูป RB $_{Replica}$ ให้ค่าน้ำหนักเท่ากับ 6 เนื่องจากเป็นแบบรูปที่มีค่าใช้จ่ายของทรัพยากรสำหรับเซอร์วิซสำรองที่พัฒนาเหมือนกัน แต่อยู่ในสภาพแวดล้อมที่แตกต่างกัน และมีค่าใช้จ่ายในการประมวลผลสำหรับการเรียกเซอร์วิซสำรองแบบลำดับ ซึ่งหมายความว่าสำเนาจะถูกเรียกและเกิดค่าใช้จ่ายก็ต่อเมื่อยังเกิดความขัดข้องอยู่เท่านั้น

(4) แบบรูป RB$_{NVP}$ ให้ค่าน้ำหนักเท่ากับ 5 เนื่องจากเป็นแบบรูปที่มีค่าใช้จ่ายในการพัฒนาเซอร์วิซสำรองที่มีการพัฒนาที่แตกต่างกันและมีค่าใช้จ่ายในการประมวลผลสำหรับการเรียกเซอร์วิซสำรองแบบลำดับ

(5) แบบรูป Active $_{Replica}$ ให้ค่าน้ำหนักเท่ากับ 4 เนื่องจากเป็นแบบรูปที่มีค่าใช้จ่ายของทรัพยากรสำหรับเซอร์วิซสำรองหลายตัวที่พัฒนาเหมือนกัน แต่อยู่ในสภาพแวดล้อมที่แตกต่างกันและมีค่าใช้จ่ายในการประมวลผลมาก เพราะมีการเรียกเซอร์วิซหลายตัวพร้อมกันแบบขนาน

(6) แบบรูป Active $_{NVP}$ ให้ค่าน้ำหนักเท่ากับ 3 เนื่องจากเป็นแบบรูปที่มีค่าใช้จ่ายในการพัฒนาเซอร์วิซสำรองที่มีการพัฒนาแตกต่างกันและมีค่าใช้จ่ายในการประมวลผลมาก เพราะมีการเรียกเซอร์วิซหลายตัวพร้อมกันแบบขนาน

(7) แบบรูป Voting$_{Replica}$ ให้ค่าน้ำหนักเท่ากับ 2 เนื่องจากเป็นแบบรูปที่มีค่าใช้จ่ายของทรัพยากรสำหรับเซอร์วิซสำรองหลายตัวที่พัฒนาเหมือนกัน แต่อยู่ในสภาพแวดล้อมที่แตกต่างกันและมีค่าใช้จ่ายในการประมวลผลมาก เพราะมีการเรียกเซอร์วิซหลายตัวพร้อมกันแบบขนาน และมีค่าใช้จ่ายในการพัฒนาอัลกอริทึมสำหรับเลือกคำตอบให้กับเซอร์วิซ

(8) แบบรูป Voting$_{NVP}$ ให้ค่าน้ำหนักเท่ากับ 1 เนื่องจากเป็นแบบรูปที่มีค่าใช้จ่ายในการพัฒนาเซอร์วิซสำรองที่มีการพัฒนาแตกต่างกันและมีค่าใช้จ่ายในการประมวลผลมาก เพราะมีการเรียกเซอร์วิซหลายตัวพร้อมกันแบบขนาน และมีค่าใช้จ่ายในการพัฒนาอัลกอริทึมสำหรับเลือกคำตอบให้กับเซอร์วิซ

(9) แบบรูป Retry+Wait ค่าน้ำหนักจะเกิดจากการนำค่าน้ำหนักของแบบรูป Retry และแบบรูป Wait มาหาค่าเฉลี่ยกัน ซึ่งจะได้ค่าน้ำหนักเท่ากับ (8+7)/2 = 7.5

3.2.2. **สร้างแบบจำลองการแนะนำแบบรูปการทนต่อความผิดพร่อง**

ในการพัฒนาเว็บเซอร์วิซให้ทนต่อความผิดพร่องได้อย่างเหมาะสมและมีคุณภาพเป็นสิ่งสำคัญสำหรับผู้พัฒนาเว็บเซอร์วิซ ดังนั้นการเลือกแบบรูปการทนต่อความผิดพร่องควรจะเลือกให้ตรงกับลักษณะของเซอร์วิซ

3.2.2.1. **ให้ผู้พัฒนาเซอร์วิซกำหนดลำดับความเด่นของลักษณะของเซอร์วิซ**

ในการเลือกแบบรูปการทนต่อความผิดพร่องสำหรับเซอร์วิซ จะพิจารณาจากความเด่นของลักษณะต่างๆ ของเซอร์วิซ โดยผู้พัฒนาเซอร์วิซจะระบุลักษณะเด่น และให้เรียงลำดับความเด่นจากมากไปน้อย ลักษณะที่เซอร์วิซไม่แสดงออกจะมีค่าเป็น 0 ลักษณะที่เซอร์วิซเด่นมากที่สุดจะมีค่าเป็น 1 ลักษณะที่เซอร์วิซเด่นรองลงมาจะมีค่าเป็น 2 เป็นเช่นนี้ไปเรื่อยๆ จนถึงลักษณะที่เซอร์วิซเด่นน้อยที่สุดจะมีค่ามากที่สุด ทั้งนี้ค่าคะแนนจะต้องมีความต่อเนื่องกัน ยกตัวอย่างเช่น ผู้พัฒนาเซอร์วิซจะพัฒนาเซอร์วิซตรวจสอบยอดเงินคงเหลือในบัญชีธนาคาร ซึ่งผู้พัฒนามีความจำเป็นจะต้องเรียกเซอร์วิซของธนาคารที่มีบัญชีลูกค้าอยู่เท่านั้น ผู้พัฒนาจึงให้ลำดับความเด่นของลักษณะ Instance Specificity เป็นอันดับ 1 และผู้พัฒนาเซอร์วิซเห็นว่าสภาพแวดล้อมในการประมวลผลของธนาคารเป็นที่น่าเชื่อถือได้ และถ้ามีข้อผิดพร่องเกิดขึ้น จะ

เกิดขึ้นไม่นาน ดังนั้นการจัดการกับข้อผิดพร่องจะไม่ซับซ้อน ทำงานง่าย เพื่อไม่ให้มีข้อผิดพร่องอย่างอื่นที่เกิดจากการจัดการกับข้อผิดพร่องที่ซับซ้อน ผู้พัฒนาจึงให้ความเด่นของลักษณะ Transient Failure และ Simplicity เท่ากันเป็นอันดับ 2 และผู้พัฒนามีสำเนาของเซอร์วิซที่ถูกพัฒนาเหมือนกัน แต่อยู่ในสภาพแวดล้อมที่แตกต่างกัน เพื่อใช้ในกรณีที่บางเซอร์วิซมีปัญหา เซอร์วิซอื่นที่เป็นสำเนาจะได้ถูกเรียกใช้งานแทน จึงให้ความเด่นของลักษณะ Replica Provision เป็นอันดับ 3 ในลักษณะเช่นนี้ผู้พัฒนาจะกำหนดลำดับความเด่นของลักษณะของเซอร์วิซเป็นดังตารางที่ 3.2

ตารางที่ 3.2 ตัวอย่างการกำหนดลำดับความเด่นของลักษณะของเซอร์วิซธนาคารแห่งหนึ่ง

	Service Characteristics							
	Transient Failure	Instance Specificity	Replica Provision	NVP Provision	Correctness	Timeliness	Simplicity	Economy
Level	2	1	3	0	0	0	2	0

3.2.2.2. เปลี่ยนลำดับความเด่นของลักษณะของเซอร์วิซเป็นค่าน้ำหนักของลักษณะของเซอร์วิซ

นำลำดับมาเปลี่ยนเป็นค่าน้ำหนักของแต่ละลักษณะ โดยค่าน้ำหนักรวมมีค่าเท่ากับ 1 ด้วยขั้นตอนต่อไปนี้

1) นำลำดับมาเปลี่ยนเป็นคะแนน โดยลักษณะเด่นที่มากกว่าจะได้คะแนนมากกว่าลักษณะเด่นที่น้อยกว่า และให้คะแนนสูงสุดเท่ากับจำนวนลำดับความเด่นที่ผู้พัฒนาเซอร์วิซระบุไว้ จากตัวอย่าง เนื่องจากลำดับความเด่นที่ผู้พัฒนาเซอร์วิซกำหนดมีทั้งหมด 3 ลำดับ ดังนั้นจึงกำหนดให้คะแนนสูงสุดมีค่าเท่ากับ 3 และลักษณะเด่นลำดับที่ 1 จะได้คะแนนมากที่สุด คือ 3 คะแนน ลักษณะเด่นลำดับที่ 2 จะได้ 2 คะแนน และลักษณะเด่นลำดับที่ 3 จะได้ 1 คะแนน ซึ่งสรุปคะแนนได้ดังตารางที่ 3.3

ตารางที่ 3.3 ตัวอย่างคะแนนที่ได้จากการเปลี่ยนลำดับความเด่นเป็นคะแนนสำหรับเซอร์วิซธนาคารแห่งหนึ่ง

	Service Characteristics							
	Transient Failure	Instance Specificity	Replica Provision	NVP Provision	Correctness	Timeliness	Simplicity	Economy
Score	2	3	1	0	0	0	2	0

2) นำคะแนนทั้งหมดมาบวกกัน จากตัวอย่างข้างต้นจะได้คะแนนรวมเท่ากับ 2+3+1+2 = 8

3) ให้เอา 1 ตั้ง หารด้วยคะแนนรวม จากตัวอย่างจะได้เป็น 1/8 = 0.125 และเอาค่าที่ได้ไปคูณกับคะแนนในข้อที่ 1) ซึ่งจะได้ค่าน้ำหนักของแต่ละลักษณะ (D) โดยค่าน้ำหนักรวมมีค่าเท่ากับ 1 เพื่อระบุถึงระดับความเด่นของลักษณะต่างๆ ค่าน้ำหนักของแต่ละลักษณะของเซอร์วิซเป็นดังตารางที่ 3.4

ตารางที่ 3.4 ตัวอย่างค่าน้ำหนักของแต่ละลักษณะของเซอร์วิซธนาคารแห่งหนึ่ง

	Service Characteristics							
	Transient Failure	Instance Specificity	Replica Provision	NVP Provision	Correctness	Timeliness	Simplicity	Economy
D	0.25	0.375	0.125	0	0	0	0.25	0

3.2.2.3. วิธีการแนะนำแบบรูปการทนต่อความผิดพร่อง

ค่าน้ำหนักของแต่ละลักษณะของเซอร์วิซ (D) ที่ได้จากข้อ 3) ในหัวข้อที่ 3.2.2.2. จะถูกนำไปคูณกับเมตริกซ์ความสัมพันธ์ระหว่างลักษณะของเซอร์วิซกับแบบรูปการทนต่อความผิดพร่อง (R) ในตารางที่ 3.1 เพื่อให้ได้คะแนนของแบบรูปการทนต่อความผิดพร่องที่เหมาะกับลักษณะของเซอร์วิซ (P) ดังสมการ (3.1)

$$P = D \times R \qquad (3.1)$$

กำหนดให้ $D = [D_{TF} \;\; D_{IS} \;\; D_{RP} \;\; D_{NP} \;\; D_{CO} \;\; D_{TI} \;\; D_{SI} \;\; D_{EC}]$

$$R = \begin{array}{c} \begin{matrix} \text{Retry} & \text{Wait} & \text{RB}_{\text{Replica}} & \text{RB}_{\text{NVP}} & \text{Active}_{\text{Replica}} & \text{Active}_{\text{NVP}} & \text{Voting}_{\text{Replica}} & \text{Voting}_{\text{NVP}} & \text{Retry+Wait} \end{matrix} \\ \begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} & R_{1,4} & R_{1,5} & R_{1,6} & R_{1,7} & R_{1,8} & R_{1,9} \\ R_{2,1} & R_{2,2} & R_{2,3} & R_{2.4} & R_{2,5} & R_{2,6} & R_{2,7} & R_{2,8} & R_{2,9} \\ R_{3,1} & R_{3,2} & R_{3,3} & R_{3,4} & R_{3,5} & R_{3,6} & R_{3,7} & R_{3,8} & R_{3,9} \\ R_{4,1} & R_{4,2} & R_{4,3} & R_{4,4} & R_{4,5} & R_{4,6} & R_{4,7} & R_{4,8} & R_{4,9} \\ R_{5,1} & R_{5,2} & R_{5,3} & R_{5,4} & R_{5,5} & R_{5,6} & R_{5,7} & R_{5,8} & R_{5,9} \\ R_{6,1} & R_{6,2} & R_{6,3} & R_{6,4} & R_{6,5} & R_{6,6} & R_{6,7} & R_{6,8} & R_{6,9} \\ R_{7,1} & R_{7,2} & R_{7,3} & R_{7,4} & R_{7,5} & R_{7,6} & R_{7,7} & R_{7,8} & R_{7,9} \\ R_{8,1} & R_{8,2} & R_{8,3} & R_{8,4} & R_{8,5} & R_{8,6} & R_{8,7} & R_{8,8} & R_{8,9} \end{bmatrix} \end{array} \begin{matrix} TF \\ IS \\ RP \\ NP \\ CO \\ TI \\ SI \\ EC \end{matrix}$$

$$P = \begin{matrix} \text{Retry} & \text{Wait} & \text{RB}_{\text{Replica}} & \text{RB}_{\text{NVP}} & \text{Active}_{\text{Replica}} & \text{Active}_{\text{NVP}} & \text{Voting}_{\text{Replica}} & \text{Voting}_{\text{NVP}} & \text{Retry+Wait} \end{matrix} \\ \begin{bmatrix} P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 \end{bmatrix}$$

จากตัวอย่างข้างต้นจะได้ผลคูณเมตริกซ์ดังนี้

$$P = \begin{bmatrix} 0.25 & 0.375 & 0.125 & 0 & 0 & 0 & 0.25 & 0 \end{bmatrix} \times \begin{bmatrix} 8 & 7 & 0 & 0 & 0 & 0 & 0 & 0 & 7.5 \\ 8 & 8 & 7 & 6 & 5 & 4 & 5 & 4 & 8 \\ 0 & 0 & 8 & 0 & 8 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & 8 & 0 & 8 & 0 & 8 & 0 \\ 2 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 2 \\ 4 & 1 & 5 & 6 & 7 & 8 & 2 & 3 & 2.5 \\ 8 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 8 \\ 7 & 8 & 6 & 5 & 4 & 3 & 2 & 1 & 7.5 \end{bmatrix}$$

$$P = \begin{matrix} \text{Retry} & \text{Wait} & \text{RB}_{\text{Replica}} & \text{RB}_{\text{NVP}} & \text{Active}_{\text{Replica}} & \text{Active}_{\text{NVP}} & \text{Voting}_{\text{Replica}} & \text{Voting}_{\text{NVP}} & \text{Retry+Wait} \end{matrix} \\ \begin{bmatrix} 7.00 & 6.75 & 5.38 & 3.75 & 4.12 & 2.50 & 3.62 & 2.00 & 2.68 \end{bmatrix}$$

จากเมตริกซ์ P ที่ได้ให้นำค่า P_i มาเปรียบเทียบและเลือกแบบรูปการทนต่อความผิดพร่องที่มีลักษณะใกล้เคียงกับลักษณะเด่นของเซอร์วิซที่ผู้พัฒนาเซอร์วิซระบุ โดยเลือกแบบรูปการทนต่อความผิดพร่องจากค่า P_i ที่มากที่สุด ถ้าค่า P_i ของแบบรูปการทนต่อความผิดพร่องใดมีค่ามากที่สุด แสดงว่า แบบรูปการทนต่อความผิดพร่องนั้นมีลักษณะใกล้เคียงและตอบสนองต่อลักษณะเด่นของเซอร์วิซที่ผู้พัฒนาเซอร์วิซระบุไว้มากที่สุด

จากตัวอย่างจะได้ว่าค่า P ที่มากที่สุด คือ ค่า P ของแบบรูป Retry มีค่าเท่ากับ 7.00 จึงสรุปได้ว่า แบบรูป Retry จะสอดคล้องกับลักษณะเด่นของเซอร์วิซมากที่สุด เนื่องจากลักษณะที่ผู้พัฒนาเซอร์วิซเห็นว่าเด่นและให้ความสำคัญเป็นอันดับ 1 คือ Instance Specificity ซึ่งมีแบบรูปที่มีค่าน้ำหนักของลักษณะนี้สูงสุด คือ Retry, Wait, Retry + Wait ส่วนลักษณะที่ผู้พัฒนาเซอร์วิซเห็นว่าเด่นเป็นอันดับ 2 คือ Transient Failure ซึ่งแบบรูป Retry มีค่าน้ำหนักสูงสุด

และลักษณะ Simplicity ซึ่งแบบรูปที่มีค่าน้ำหนักของลักษณะนี้สูงสุดคือ Retry, Wait, Retry + Wait ส่วนลักษณะที่ผู้พัฒนาเซอร์วิซเห็นว่าเด่นเป็นอันดับ 3 คือ Replica Provision ซึ่งแบบรูปที่มีค่าน้ำหนักของลักษณะนี้สูงสุด คือ $RB_{Replica}$, $Active_{Replica}$, $Voting_{Replica}$ เนื่องจากแบบรูปRetry เป็นแบบรูปที่มีค่าน้ำหนักสูงสุดของลักษณะเซอร์วิซเป็นจำนวนมากที่สุด ดังนั้นแบบรูป Retry จึงเป็นแบบรูปที่เครื่องมือจะแนะนำให้กับผู้พัฒนาเซอร์วิซสำหรับการพัฒนาเซอร์วิซให้ทนต่อความผิดพร่อง

3.3. พัฒนาเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่อง

เครื่องมือสนับสนุนประกอบด้วยการทำงาน 2 ส่วน ดังนี้

3.3.1.ส่วนประเมินลักษณะของเซอร์วิซตามวิธีแนะนำที่ออกแบบไว้

พัฒนาเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่อง ในส่วนประเมินลักษณะของเซอร์วิซตามวิธีแนะนำที่ออกแบบไว้ดังนี้

- พัฒนาแอพพลิเคชันที่มีหน้าต่างสำหรับให้ผู้พัฒนาเซอร์วิซศึกษาคำจำกัดความของลักษณะของเซอร์วิซ และกรอกลำดับความเด่นของลักษณะของเซอร์วิซดังตารางที่ 3.2
- นำลำดับความเด่นที่ผู้พัฒนาเซอร์วิซกรอกมาเป็นค่าพารามิเตอร์สำหรับคำนวณเพื่อหาค่า P ตามวิธีแนะนำที่ออกแบบไว้ในข้อที่ 3.2.2.
- จากค่า P ที่ได้ เครื่องมือจะนำมาใช้ในการแนะนำแบบรูปการทนต่อความผิดพร่องเพื่อสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่อง
- ผู้พัฒนาเซอร์วิซสามารถตัดสินใจเลือกแบบรูปการทนต่อความผิดพร่องได้ว่า จะเลือกแบบรูปตามที่เครื่องมือแนะนำหรือเลือกแบบรูปอื่น

3.3.2.ส่วนสร้างบีเพลสำหรับเว็บเซอร์วิซที่ทนต่อความผิดพร่อง

พัฒนาเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่อง ในส่วนสร้างบีเพลสำหรับเว็บเซอร์วิซที่ทนต่อความผิดพร่องดังนี้

- จากแบบรูปที่ผู้พัฒนาเซอร์วิซเลือกสามารถกำหนดค่าพารามิเตอร์ที่เกี่ยวข้องกับแบบรูปนั้นได้ นอกเหนือไปจากวิสเดิลของเซอร์วิซที่ต้องการให้ทนต่อความผิดพร่อง เช่น

แบบรูป Retry มีพารามิเตอร์ที่เกี่ยวข้อง คือ จำนวนรอบที่เรียกเซอร์วิซซ้ำ แบบรูป RecoveryBlock มีพารามิเตอร์ที่เกี่ยวข้อง คือ รายการวิสเดิลของเซอร์วิซตัวแทน แบบรูป Voting และแบบรูป Active มีพารามิเตอร์ที่เกี่ยวข้อง คือ รายการวิสเดิลของกลุ่มเซอร์วิซที่ถูกเรียกพร้อมกัน แบบรูป Wait มีพารามิเตอร์ที่เกี่ยวข้อง คือ เวลาที่กำหนดให้เรียกเซอร์วิซ เป็นต้น

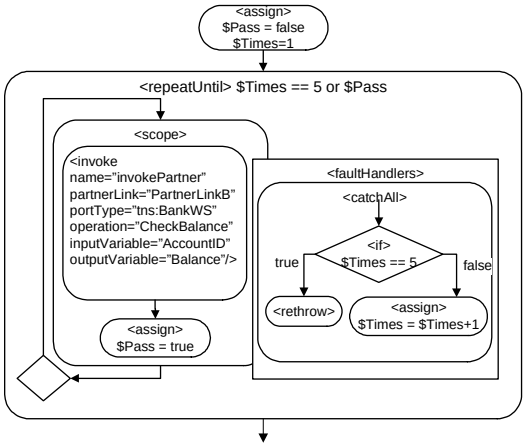
- สร้างบีเพลสำหรับเว็บเซอร์วิซที่ทนต่อความผิดพร่อง โดยใช้ค่าพารามิเตอร์ที่กำหนดด้วย GlassFish ESB v2.2 [19] ซึ่งรองรับการดำเนินงานของกระแสงานบีเพลตามมาตรฐานและสามารถประมวลผลบีเพลได้

โดยอธิบายการทำงานของเครื่องมือในส่วนสร้างบีเพลสำหรับเว็บเซอร์วิซที่ทนต่อความผิดพร่องโดยใช้แบบรูป Retry ได้ดังตัวอย่างในภาพที่ 3.3

Input: WSDL ⟹ Output:

$Times = 5

กำหนดให้ค่าพารามิเตอร์ $Times คือ จำนวนรอบที่เซอร์วิซเรียกซ้ำ เมื่อมีข้อผิดพร่องเกิดขึ้น



ภาพที่ 3.3 การทำงานของเครื่องมือในส่วนสร้างบีเพลสำหรับเว็บเซอร์วิซที่ทนต่อความผิดพร่อง

3.4. ประเมินเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่อง

จากการพัฒนาเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่อง ผู้วิจัยจะทำการประเมินเครื่องมือ โดยการให้ผู้พัฒนาเว็บเซอร์วิซเป็นผู้ประเมิน จำนวนอย่างน้อย 10 ราย โดยบางรายมีการพัฒนาเซอร์วิซโดเมนเดียวกัน หลังจากที่ผู้พัฒนาได้ใช้เครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องแล้ว จะให้ทำแบบประเมินในภาคผนวก ข เพื่อประเมินว่าเครื่องมือที่ใช้ให้ผลลัพธ์ตรงกับความต้องการมากน้อยเพียงใด เครื่องมือและการแนะนำแบบรูปมี

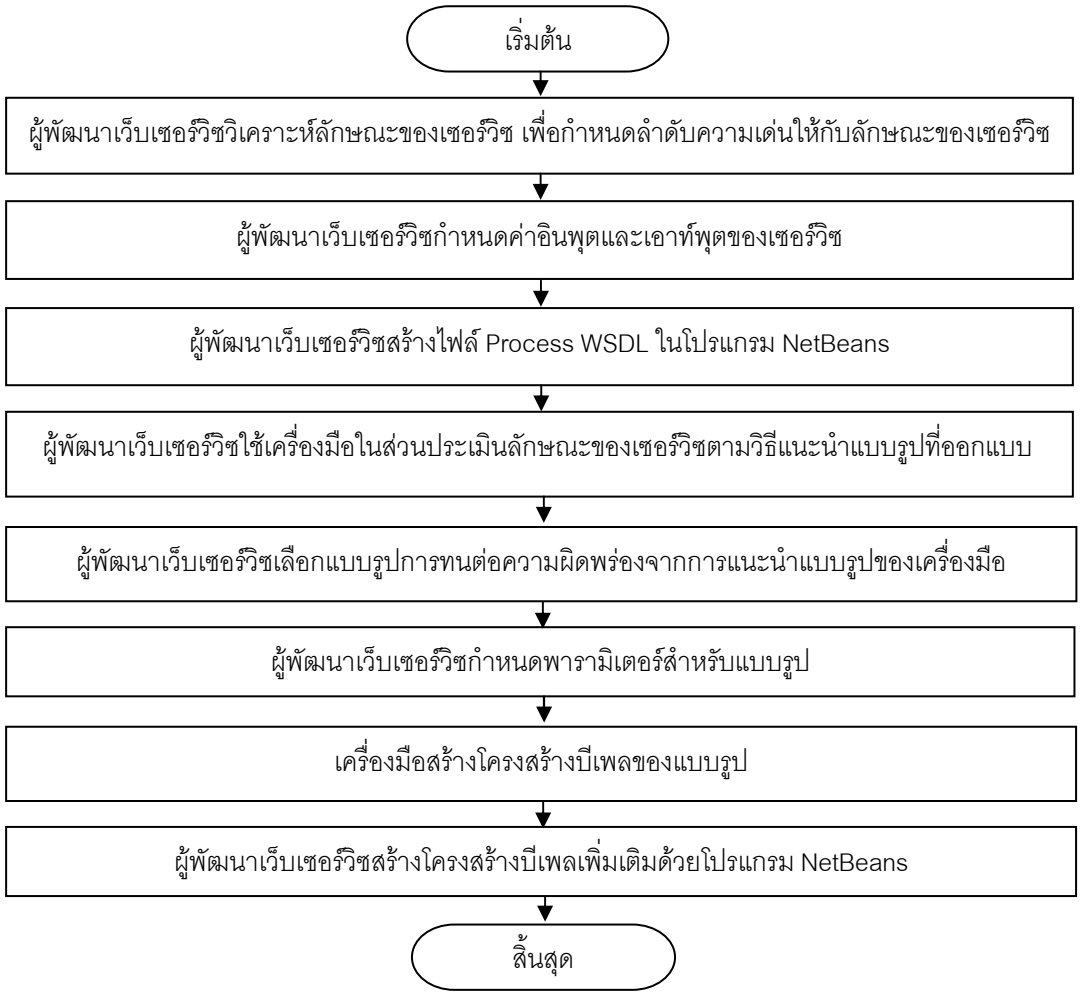
ความสมเหตุสมผลกับเซอร์วิซที่กำลังจะพัฒนาหรือไม่ และมีการทวนสอบความเหมาะสมของแบบรูปที่แนะนำเมื่อเทียบกับแบบรูปที่ไม่ได้แนะนำให้กับเซอร์วิซ

นอกจากนี้จากการใช้งานเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องของผู้พัฒนาเว็บเซอร์วิซ จะทำการเก็บข้อมูลการใช้งาน เพื่อนำมาวิเคราะห์หาความสัมพันธ์ของข้อมูลจากการใช้งาน ตัวอย่างเช่น

- หาความสัมพันธ์ระหว่างโดเมนของเซอร์วิซกับค่าน้ำหนักที่ผู้พัฒนาเซอร์วิซกำหนดให้กับลักษณะต่างๆ เพื่อวิเคราะห์ว่า ลักษณะแบบใดที่ผู้พัฒนาเห็นว่าเด่นในการพัฒนาเซอร์วิซประเภทนี้
- หาความสัมพันธ์ระหว่างโดเมนของเซอร์วิซกับแบบรูปการทนต่อความผิดพร่องที่เครื่องมือสนับสนุนเลือกให้ เพื่อวิเคราะห์ว่า โดเมนของเซอร์วิซมีผลต่อการแนะนำแบบรูปการทนต่อความผิดพร่องหรือไม่ และโดเมนของเซอร์วิซประเภทนี้ส่วนใหญ่ใช้แบบรูปการทนต่อความผิดพร่องแบบใด

บทที่ 4
การทำงานของเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่อง

ในการทำงานของเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องมีภาพรวมของการใช้งานเครื่องมือโดยผู้พัฒนาเว็บเซอร์วิซ ดังภาพที่ 4.1



ภาพที่ 4.1 การใช้งานเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่อง

ผู้พัฒนาเว็บเซอร์วิซจะต้องวิเคราะห์ว่าเซอร์วิซที่ต้องการพัฒนาให้ทนต่อความผิดพร่องนั้นมีลักษณะใดบ้างตามรายการลักษณะของเซอร์วิซที่เครื่องมือกำหนด และกำหนดลำดับความเด่นให้กับลักษณะของเซอร์วิซ จากนั้นกำหนดค่าอินพุตและเอาท์พุตของเซอร์วิซ ซึ่งเครื่องมือจะสร้างโครงสร้างบีเพลของแบบรูปการทนต่อความผิดพร่อง โดยใช้โปรแกรม NetBeans IDE 6.7.1 ประกอบการสร้าง ซึ่งจะประกอบด้วยการทำงานทั้งหมด 3 ส่วน คือ

1) ส่วนสร้างไฟล์ Process WSDL ในโปรแกรม NetBeans

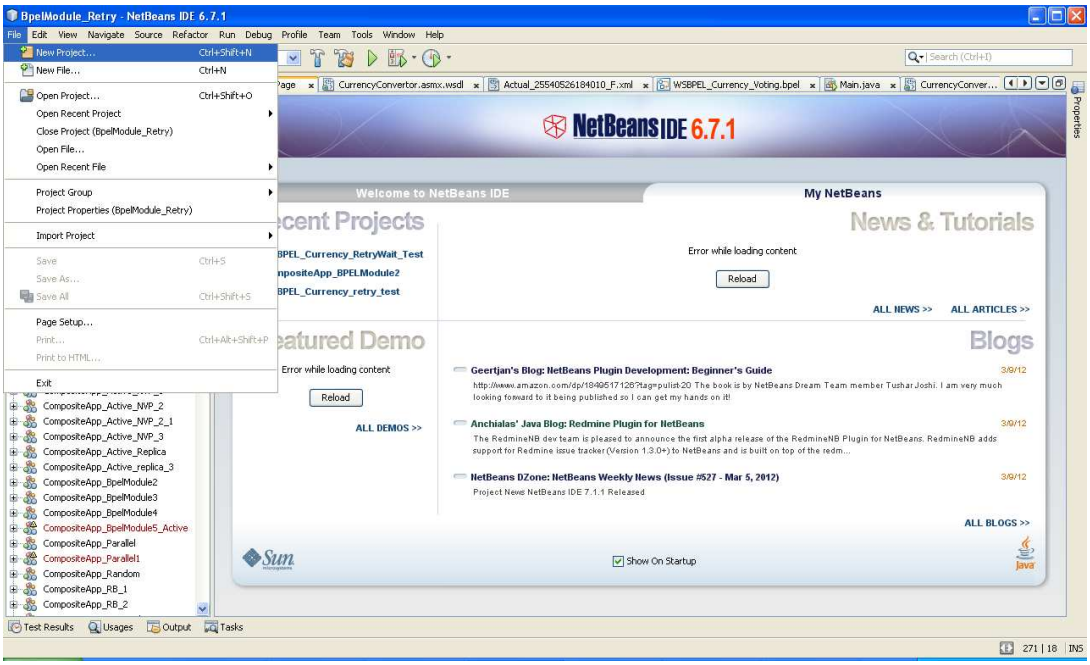
2) ส่วนประเมินลักษณะของเซอร์วิซตามวิธีแนะนำแบบรูปที่ออกแบบ

3) ส่วนสร้างโครงสร้างบีเพลของแบบรูปการทนต่อความผิดพร่อง

4.1. ส่วนสร้างไฟล์ Process WSDL ในโปรแกรม NetBeans

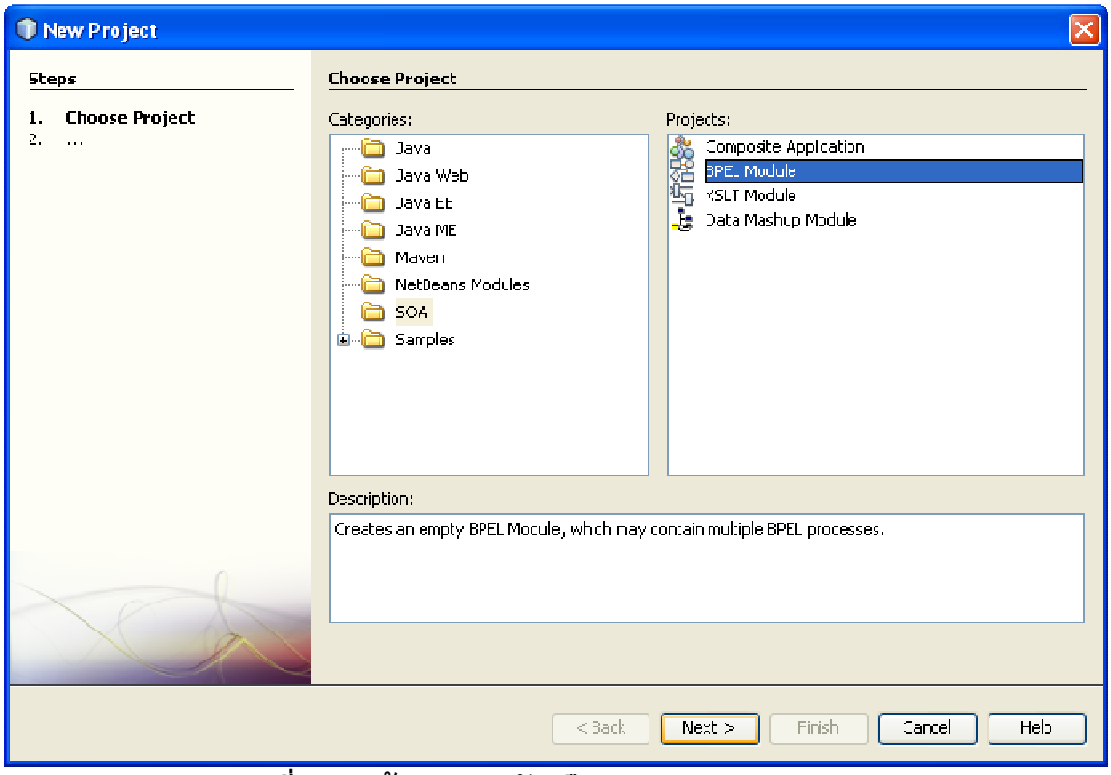
หลังจากวิเคราะห์ลักษณะของเซอร์วิซ เพื่อกำหนดลำดับความเด่นให้กับลักษณะของเซอร์วิซ และกำหนดค่าอินพุตและเอาท์พุตของเซอร์วิซ จะสร้างไฟล์ Process WSDL File ในโปรแกรม NetBeans โดยให้ผู้ใช้ copy folder ชื่อ BPEL ไว้ใน Drive C: ก่อน ซึ่งขั้นตอนการสร้างไฟล์ Process WSDL มีดังต่อไปนี้

1. ให้ผู้ใช้ New Project ในโปรแกรม NetBeans โดยคลิกที่เมนู File เลือก New Project ดังภาพที่ 4.2



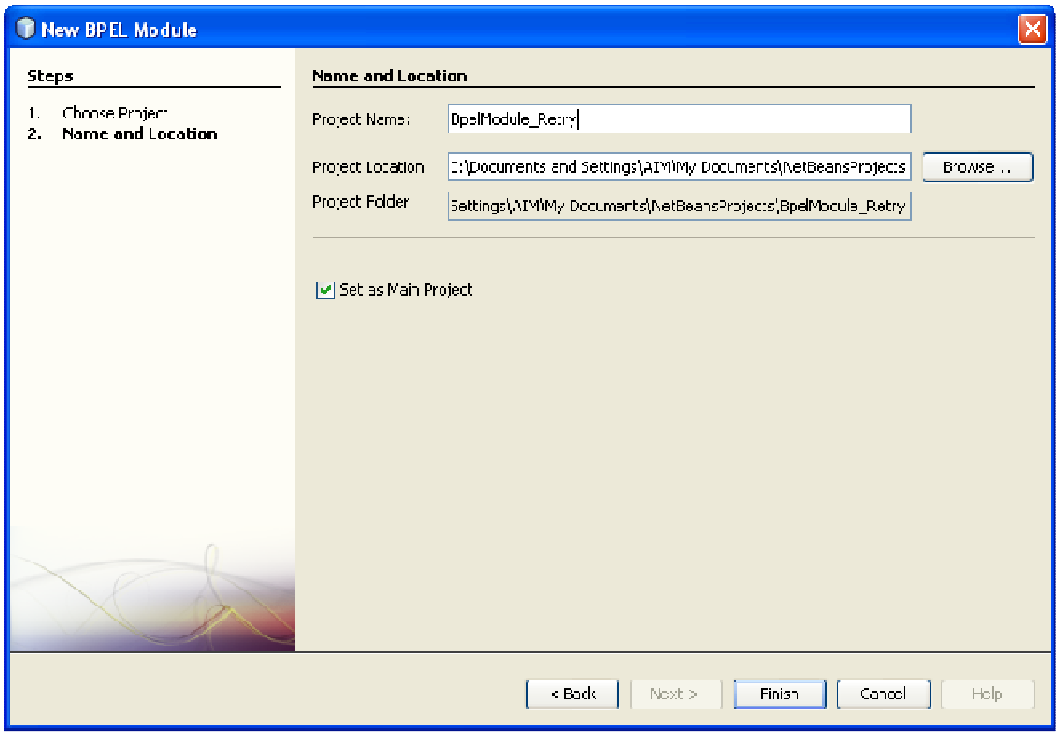
ภาพที่ 4.2 หน้าต่างโปรแกรม NetBeans

2. จะได้หน้าต่างสำหรับเลือก Category และ Project ให้ผู้ใช้เลือก Category เป็น SOA และ Project เป็น BPEL Module ดังภาพที่ 4.3 จากนั้นคลิกปุ่ม Next



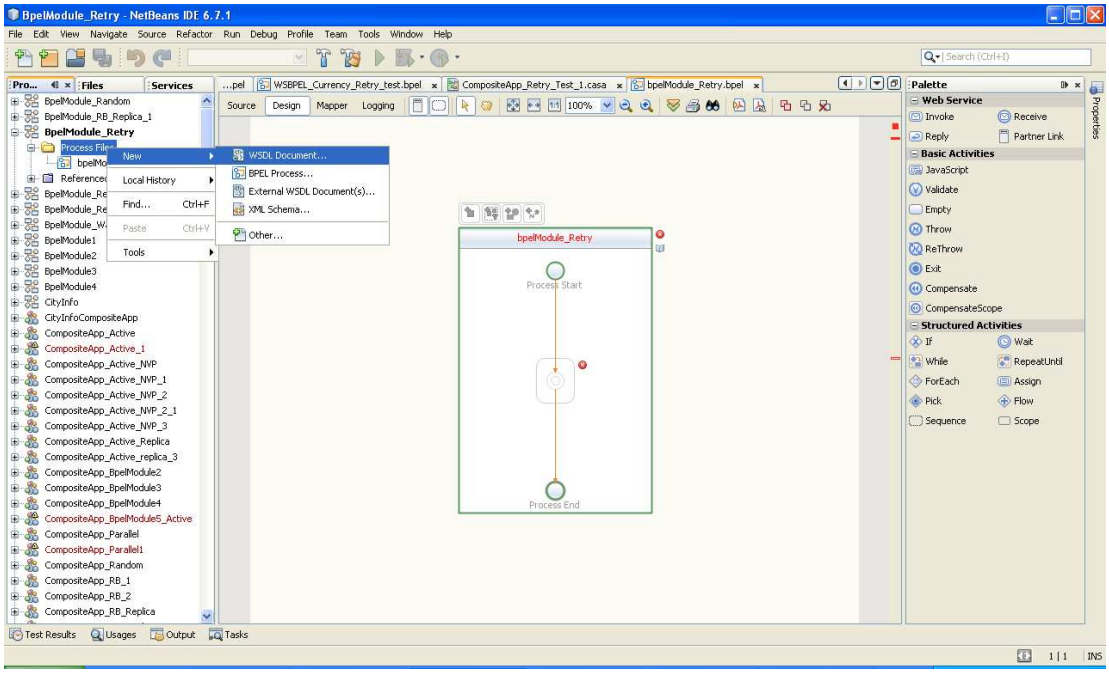
ภาพที่ 4.3 หน้าต่างสำหรับเลือก Category และ Project

3. จะได้หน้าต่างให้ตั้งชื่อ Project และเลือกที่จัดเก็บไฟล์ ดังภาพที่ 4.4 เมื่อผู้ใช้กำหนดแล้ว ให้คลิกปุ่ม Finish โปรแกรมจะสร้างไฟล์ BPEL ให้



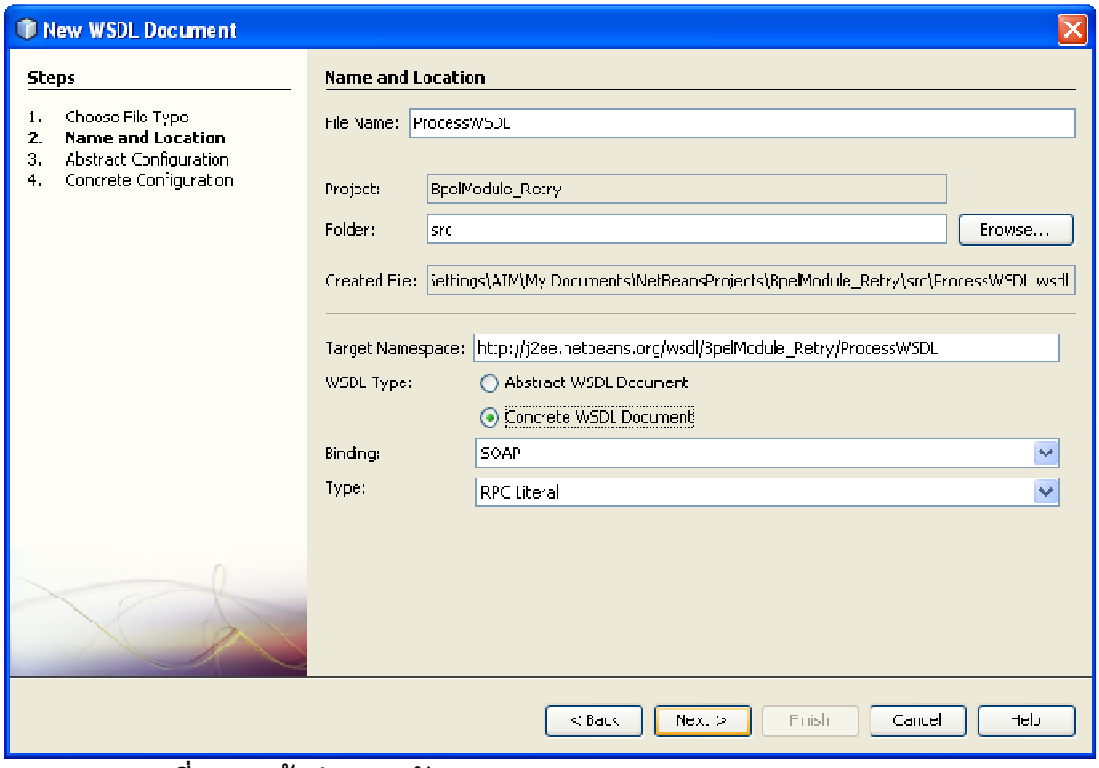
ภาพที่ 4.4 หน้าต่างให้ตั้งชื่อ Project และเลือกที่จัดเก็บไฟล์

4. ให้ผู้ใช้สร้างไฟล์วิสเดิลสำหรับ Process เพื่อใช้ติดต่อกับไฟล์วิสเดิล ของเว็บเซอร์วิซ โดยไปที่หน้าต่าง Project และคลิกขวาที่ Process Files ใน Project ที่สร้างไว้ เลือก New > WSDL Document ดังภาพที่ 4.5 ยกเว้นแบบรูป Voting จะไม่ต้องสร้างไฟล์วิสเดิลตามขั้นตอนที่ 4-7 แต่จะใช้ไฟล์ XML Schema Document



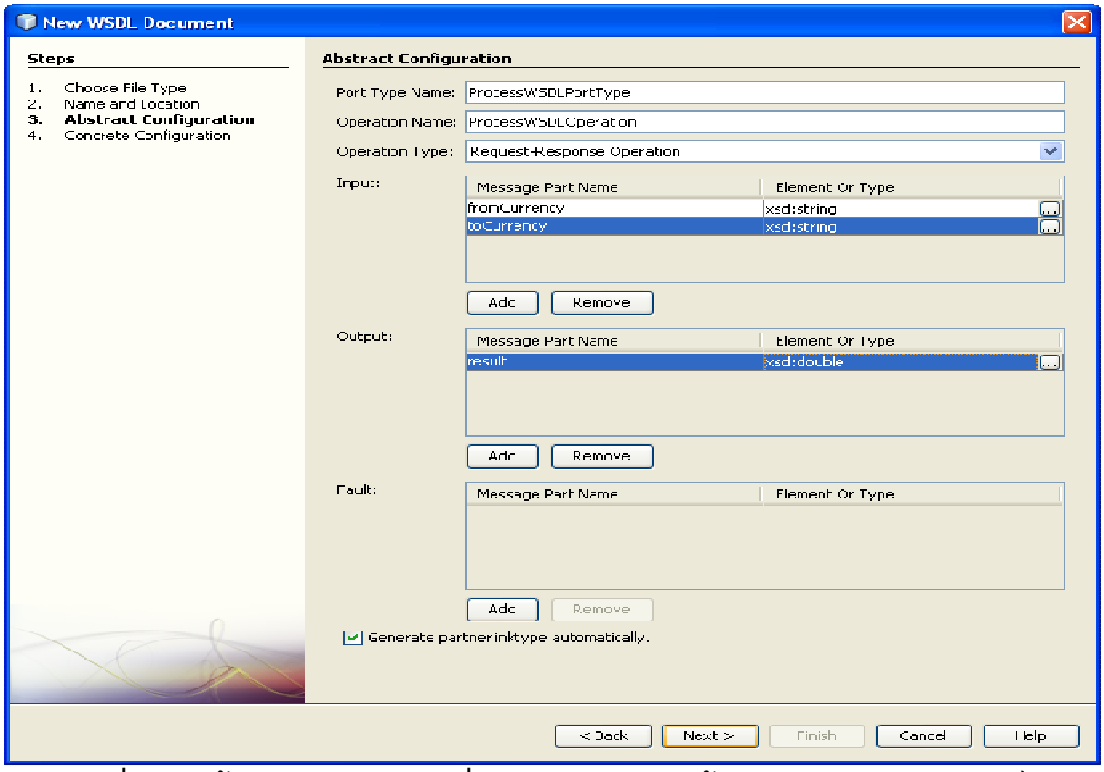
ภาพที่ 4.5 หน้าต่างสำหรับสร้างไฟล์ WSDL สำหรับ Process

5. จะได้หน้าต่างดังภาพที่ 4.6 ให้ผู้ใช้กำหนด File Name, Folder และ WSDL Type ซึ่งเราจะเลือกเป็น Concrete WSDL Document และโพรโทคอลเป็น SOAP ประเภท RPC Literal จากนั้นให้คลิกปุ่ม Next



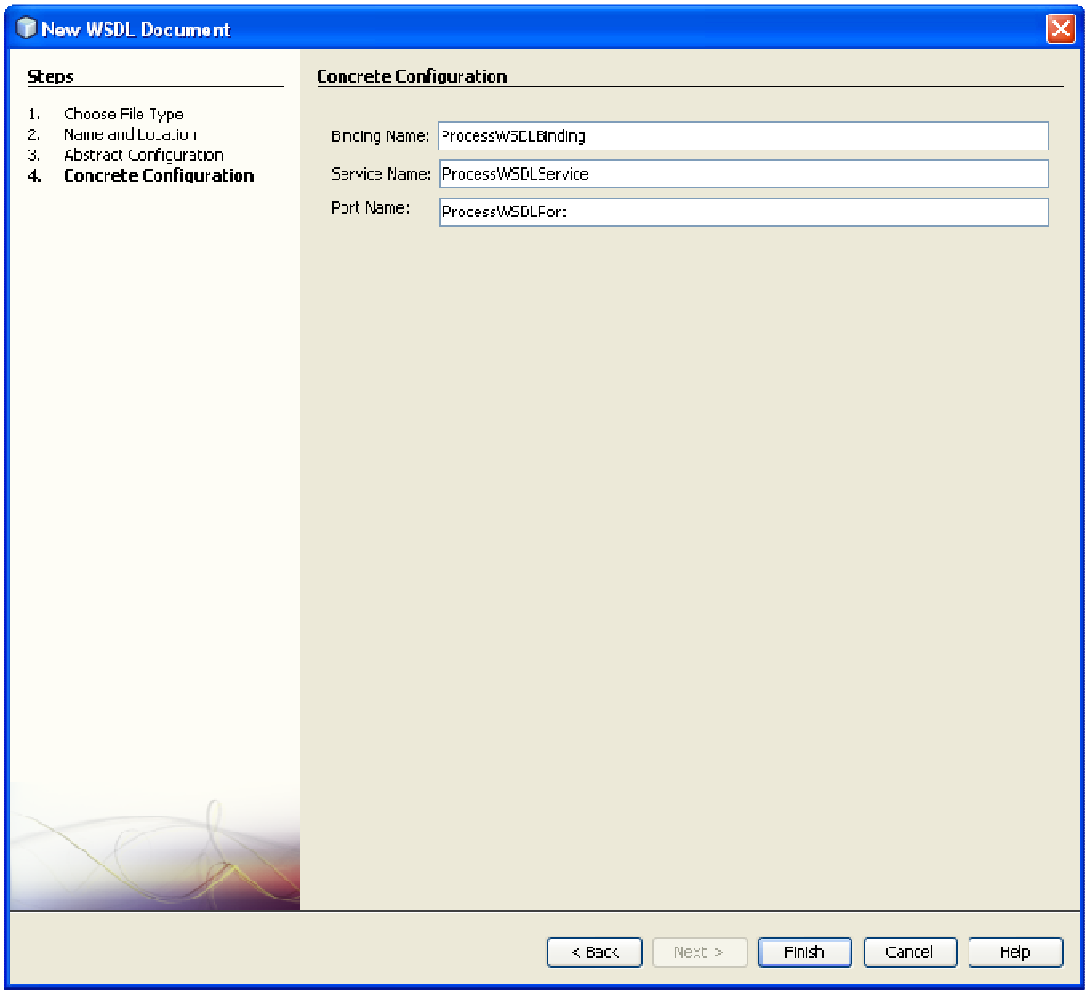
ภาพที่ 4.6 **หน้าต่างสำหรับกำหนด** File Name, Folder **และ** WSDL Type

6. จะได้หน้าต่างดังภาพที่ 4.7 ให้ผู้กรอกชื่อตัวแปรและชนิดข้อมูลของอินพุตและเอาท์พุต แล้วคลิกปุ่ม Next



ภาพที่ 4.7 **หน้าต่างสำหรับกรอกชื่อตัวแปรและชนิดข้อมูลของอินพุตและเอาท์พุต**

7. จะได้หน้าต่างดังภาพที่ 4.8 ให้คลิกปุ่ม Finish



ภาพที่ 4.8 หน้าต่างสำหรับตั้งชื่อ Binding, Service, Port

4.2. ส่วนประเมินลักษณะของเซอร์วิซตามวิธีแนะนำที่ออกแบบ

ส่วนประเมินลักษณะของเซอร์วิซตามวิธีแนะนำที่ออกแบบประกอบด้วยหน้าต่างการทำงานของเครื่องมือทั้งหมด 4 ส่วน คือ หน้าต่างหลัก หน้าต่างสำหรับเลือกลักษณะของเซอร์วิซ หน้าต่างกำหนดลำดับความเด่นของลักษณะของเซอร์วิซ และหน้าต่างแนะนำแบบรูปการทนต่อความผิดพร่อง

4.2.1. หน้าต่างหลัก

ในหน้าต่างหลักแสดงดังภาพที่ 4.9 จะให้ผู้ใช้กรอกค่าพารามิเตอร์ต่างๆ ที่ต้องใช้ในการสร้างบีเพล คือ

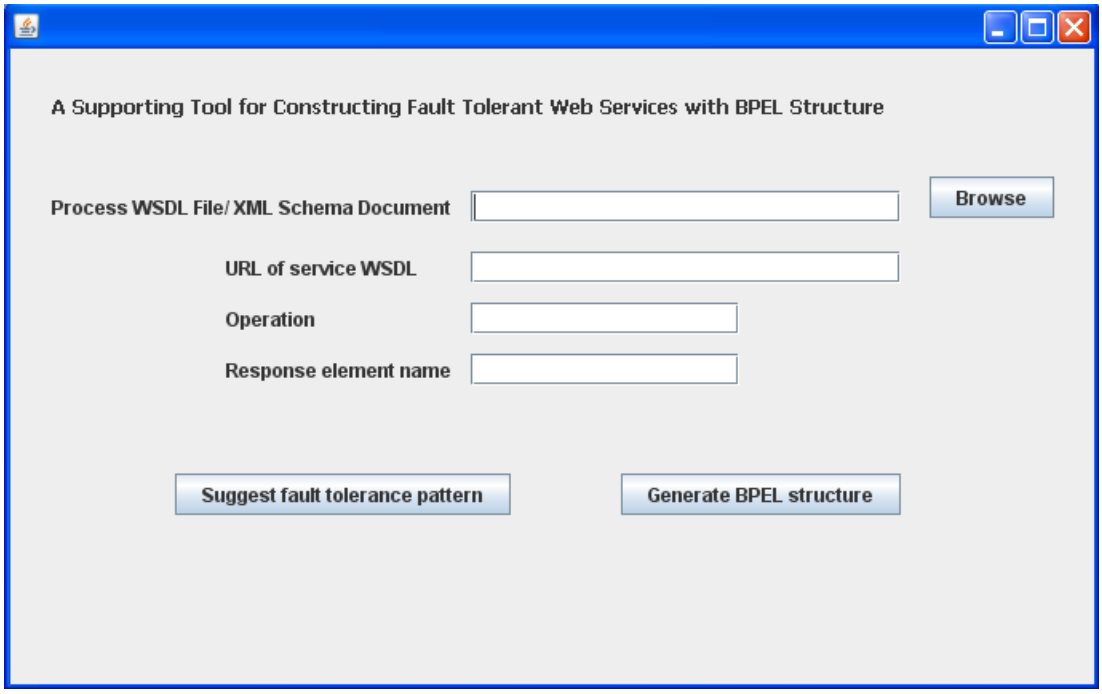
- Process WSDL File/XML Schema Document คือ ไฟล์วิสเดิลหรือไฟล์ XML Schema Document ที่สร้างในโปรแกรม NetBeans ตามขั้นตอนในหัวข้อ 4.1 ซึ่งกำหนดค่าอินพุตและเอาท์พุต ที่ส่งและรับค่าให้กับเซอร์วิซในกระแสงานบีเพล

- URL of service WSDL คือ URL ของไฟล์วิสเดิลของเซอร์วิซที่ต้องการทำให้ทนต่อความผิดพร่อง

- Operation คือ ชื่อ Operation ที่เรียกใช้ในเซอร์วิซ

- Response element name คือ ชื่อตัวแปรที่เก็บค่าผลลัพธ์ของ Operation ซึ่งบาง Operation อาจจะไม่มี ถ้า Operation นั้นมีตัวแปรผลลัพธ์ตัวเดียว

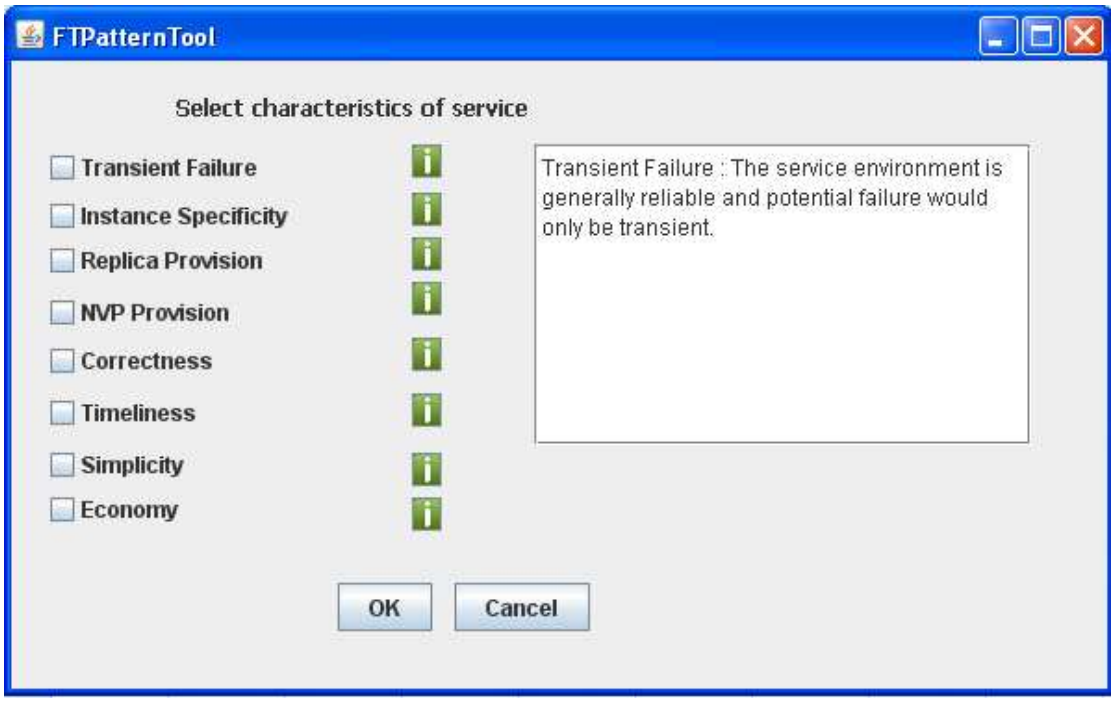
โดยมีปุ่ม 2 อันคือ Suggest fault tolerance pattern เป็นปุ่มสำหรับแนะนำแบบรูปการทนต่อความผิดพร่องให้กับเซอร์วิซ และ Generate BPEL structure เป็นปุ่มสำหรับสร้างโครงสร้างของบีเพลโดยไม่ต้องการให้เครื่องมือแนะนำแบบรูปให้



ภาพที่ 4.9 หน้าต่างหลัก

4.2.2. หน้าต่างสำหรับเลือกลักษณะของเซอร์วิซ

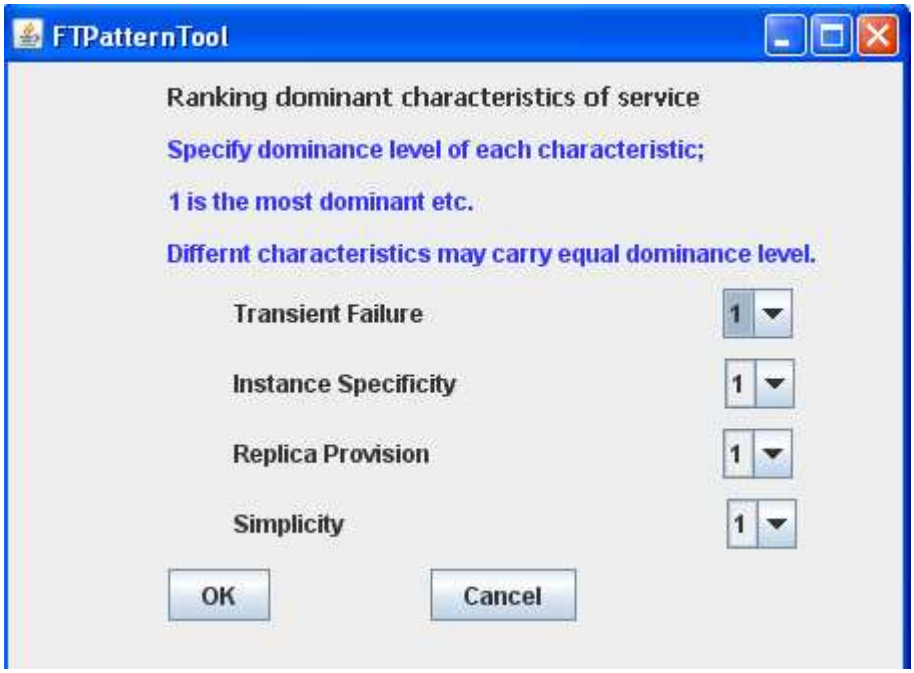
หน้าต่างสำหรับเลือกลักษณะของเซอร์วิซแสดงดังภาพที่ 4.10 โดยหน้าต่างนี้จะแสดงคำอธิบายลักษณะของเซอร์วิซแต่ละลักษณะ โดยการคลิกที่ปุ่ม  และให้ผู้ใช้เลือกลักษณะที่ตรงกับลักษณะของเซอร์วิซของผู้ใช้



ภาพที่ 4.10 หน้าต่างสำหรับเลือกลักษณะของเซอร์วิซ

4.2.3. หน้าต่างกำหนดลำดับความเด่นของลักษณะของเซอร์วิซ

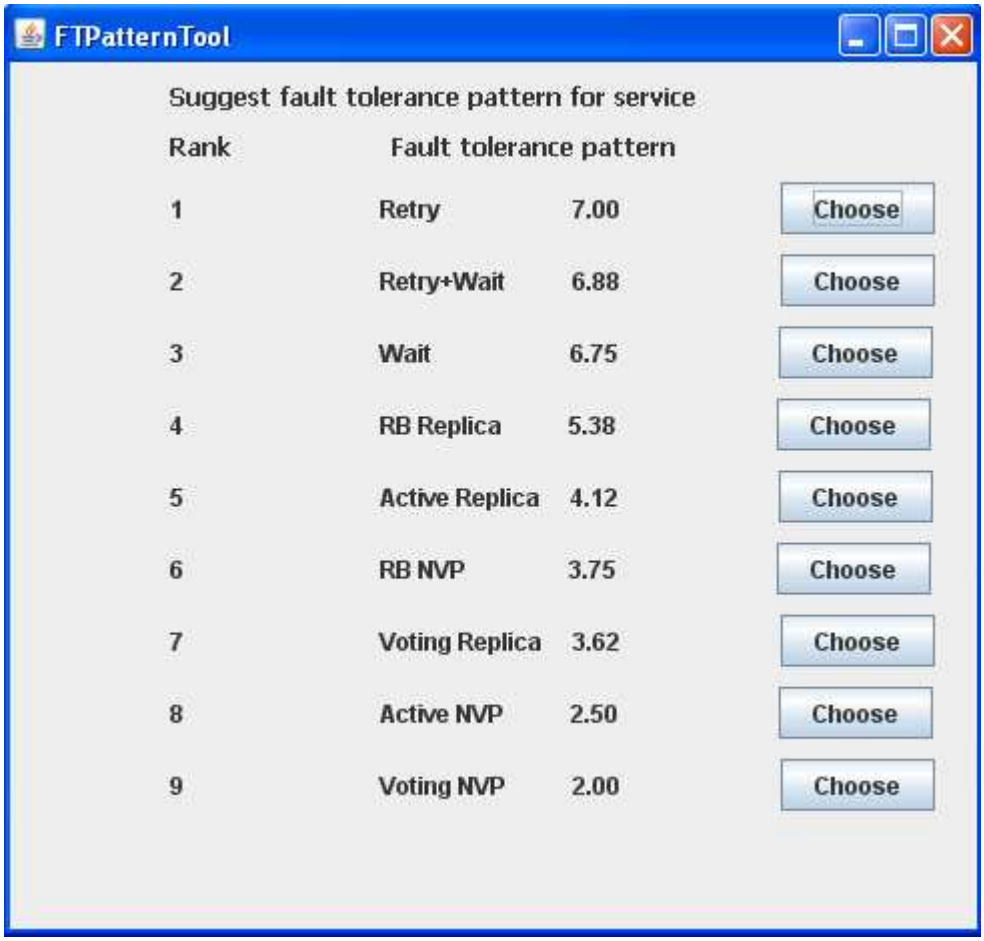
หน้าต่างกำหนดลำดับความเด่นของลักษณะของเซอร์วิซแสดงดังภาพที่ 4.11 โดยหน้าต่างนี้จะแสดงลักษณะของเซอร์วิซที่ผู้ใช้เลือกในหน้าต่างสำหรับเลือกลักษณะของเซอร์วิซ เพื่อให้ผู้ใช้กำหนดลำดับความเด่น ซึ่งจำนวนลำดับจะเท่ากับจำนวนลักษณะของเซอร์วิซ โดยลักษณะที่ได้ลำดับความเด่นอันดับ 1 หมายถึง ลักษณะของเซอร์วิซนั้นมีความเด่นมากที่สุด ลักษณะที่เด่นรองลงมาคือ อันดับ 2 และถ้าลักษณะของเซอร์วิซมีความเด่นเท่ากัน จะมีอันดับเดียวกัน



ภาพที่ 4.11 หน้าต่างกำหนดลำดับความเด่นของลักษณะของเซอร์วิซ

4.2.4. หน้าต่างแนะนำแบบรูปการทนต่อความผิดพร่อง

หน้าต่างแนะนำแบบรูปการทนต่อความผิดพร่องแสดงดังภาพที่ 4.12 เป็นผลลัพธ์ที่เครื่องมือคำนวณด้วยสูตร P = D x R เพื่อแนะนำแบบรูปการทนต่อความผิดพร่องให้กับเซอร์วิซ โดยคะแนน (P) ของแบบรูปใดที่มีคะแนนมากที่สุด แสดงว่าแบบรูปนั้นเหมาะสมกับลักษณะของเซอร์วิซมากที่สุด ซึ่งผู้ใช้สามารถเลือกได้ว่าจะเลือกแบบรูปการทนต่อความผิดพร่องใดด้วยการคลิกที่ปุ่ม Choose หลังแบบรูปนั้น เพื่อให้เครื่องมือสร้างโครงสร้างของบีเพลตามแบบรูปที่เลือกต่อไป



ภาพที่ 4.12 หน้าต่างแนะนำแบบรูปการทนต่อความผิดพร่อง

4.3. ส่วนสร้างโครงสร้างบีเพลของแบบรูปที่ทนต่อความผิดพร่อง

เมื่อผู้ใช้เลือกแบบรูปการทนต่อความผิดพร่องที่ต้องการในภาพที่ 4.12 แล้ว การทำงานในส่วนต่อไป คือ ส่วนสร้างโครงสร้างบีเพลของแบบรูปที่ทนต่อความผิดพร่องประกอบด้วย หน้าต่างการทำงานสำหรับกรอกค่าพารามิเตอร์ของแต่ละแบบรูปการทนต่อความผิดพร่อง ซึ่งมีทั้งหมด 9 แบบรูป โดยพารามิเตอร์ของแต่ละแบบรูปจะไม่เหมือนกัน เมื่อผู้ใช้กรอกค่าพารามิเตอร์แล้ว ผู้ใช้จะได้ไฟล์โครงสร้างบีเพลที่อยู่ใน C:\BPEL เพื่อนำไปใช้ในการสร้างกระบวนการทางธุรกิจ และใช้โปรแกรม NetBeans ประกอบการสร้างโครงสร้างบีเพลเพิ่มเติม จึงจะได้โครงสร้างบีเพลที่ทนต่อความผิดพร่องที่สมบูรณ์

4.3.1. หน้าต่างสำหรับกรอกค่าพารามิเตอร์ของแบบรูป

การกรอกพารามิเตอร์สำหรับแบบรูปต่างๆ ดังตารางที่ 4.1

ตารางที่ 4.1 พารามิเตอร์สำหรับแบบรูปต่าง ๆ

แบบรูป	พารามิเตอร์	คำอธิบาย
Retry	Retry times	จำนวนรอบที่เรียกเซอร์วิซซ้ำ ถ้าเซอร์วิซเกิดข้อผิดพร่อง
Wait	Duration time	ระยะเวลาที่ผู้ใช้รอก่อนเรียกเซอร์วิซให้ทำงาน หน่วยเป็นวินาที
$RB_{Replica}$	Recovery WSDL File from URL	URL ไฟล์วิสเดิลของเซอร์วิซตัวแทนที่เป็นสำเนาของเซอร์วิซหลัก ถ้าเซอร์วิซหลักเกิดข้อผิดพร่อง ผู้พัฒนาเซอร์วิซต้องพัฒนาเซอร์วิซสำเนาเอง
	Operation	ชื่อ Operation ที่เรียกใช้ในเซอร์วิซตัวแทน (ในที่นี้เครื่องมือรองรับเซอร์วิซตัวแทน 1 ตัว)
RB_{NVP}	Recovery WSDL File from URL	URL ไฟล์วิสเดิลของเซอร์วิซตัวแทนที่พัฒนาแตกต่างกัน (N-Version Programming: NVP) ถ้าเซอร์วิซหลักเกิดข้อผิดพร่อง ผู้พัฒนาเซอร์วิซต้องพัฒนาเซอร์วิซตัวแทนเองหรือเลือกใช้เซอร์วิซจากผู้ให้บริการรายอื่นที่ทำงานแบบเดียวกันมาเป็นตัวแทน
	Operation	ชื่อ Operation ที่เรียกใช้ในเซอร์วิซตัวแทน (ในที่นี้เครื่องมือรองรับเซอร์วิซตัวแทน 1 ตัว)

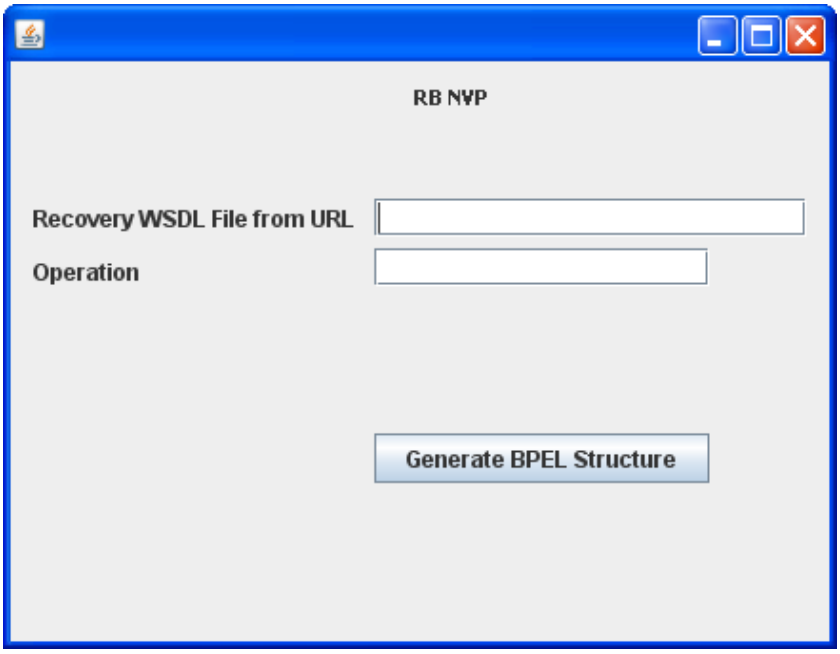
ตารางที่ 4.1 พารามิเตอร์สำหรับแบบรูปต่าง ๆ (ต่อ)

แบบรูป	พารามิเตอร์	คำอธิบาย
$Active_{Replica}$	Number of services	จำนวนเซอร์วิซที่จะเรียกพร้อมกันกับเซอร์วิซหลัก ซึ่งเป็นสำเนาของเซอร์วิซ
	WSDL File n from URL	URL ไฟล์วิสเดิลของเซอร์วิซที่ n ซึ่งจะเรียกพร้อมกัน ผู้พัฒนาเซอร์วิซต้องพัฒนาเซอร์วิซที่ n ที่เป็นสำเนาเอง
	Operation n	ชื่อ Operation ของเซอร์วิซที่ n ที่จะเรียกใช้
	Response element name n	ชื่อตัวแปรที่เก็บค่าผลลัพธ์ของ Operation n ซึ่งบาง Operation อาจจะไม่มี ถ้า Operation นั้นมีตัวแปรผลลัพธ์ตัวเดียว
$Active_{NVP}$	Number of services	จำนวนเซอร์วิซที่จะเรียกพร้อมกันกับเซอร์วิซหลัก ซึ่งเป็นเซอร์วิซที่มีฟังก์ชันการทำงานเหมือนกัน แต่มีการพัฒนาแตกต่างกัน
	WSDL File n from URL	URL ไฟล์วิสเดิลของเซอร์วิซที่ n ซึ่งจะเรียกพร้อมกัน ผู้พัฒนาเซอร์วิซต้องพัฒนาเซอร์วิซตัวแทน n เองหรือเลือกใช้เซอร์วิซจากผู้ให้บริการรายอื่นที่ทำงานแบบเดียวกันมาเป็นตัวแทน
	Operation n	ชื่อ Operation ของเซอร์วิซที่ n ที่จะเรียกใช้
	Response element name n	ชื่อตัวแปรที่เก็บค่าผลลัพธ์ของ Operation n ซึ่งบาง Operation อาจจะไม่มี ถ้า Operation นั้นมีตัวแปรผลลัพธ์ตัวเดียว

ตารางที่ 4.1 พารามิเตอร์สำหรับแบบรูปต่าง ๆ (ต่อ)

แบบรูป	พารามิเตอร์	คำอธิบาย
$Voting_{Replica}$	Number of services	จำนวนเซอร์วิซที่จะเรียกพร้อมกันกับเซอร์วิซหลัก ซึ่งเป็นสำเนาของเซอร์วิซ
	WSDL File n from URL	URL ไฟล์วิสเดิลของเซอร์วิซที่ n ซึ่งจะเรียกพร้อมกัน ผู้พัฒนาเซอร์วิซต้องพัฒนาเซอร์วิซที่ n ที่เป็นสำเนาเอง
	Operation n	ชื่อ Operation ของเซอร์วิซที่ n ที่จะเรียกใช้
	Response element name n	ชื่อตัวแปรที่เก็บค่าผลลัพธ์ของ Operation n ซึ่งบาง Operation อาจจะไม่มี ถ้า Operation นั้นมีตัวแปรผลลัพธ์ตัวเดียว
$Voting_{NVP}$	Number of services	จำนวนเซอร์วิซที่จะเรียกพร้อมกันก่อน ซึ่งเป็นเซอร์วิซที่มีฟังก์ชันการทำงานเหมือนกัน แต่มีการพัฒนาแตกต่างกัน
	WSDL File n from URL	URL ไฟล์วิสเดิลของเซอร์วิซที่ n ซึ่งจะเรียกพร้อมกัน ผู้พัฒนาเซอร์วิซต้องพัฒนาเซอร์วิซตัวแทน n เองหรือเลือกใช้เซอร์วิซจากผู้ให้บริการรายอื่นที่ทำงานแบบเดียวกันมาเป็นตัวแทน
	Operation n	ชื่อ Operation ของเซอร์วิซที่ n ที่จะเรียกใช้
	Response element name n	ชื่อตัวแปรที่เก็บค่าผลลัพธ์ของ Operation n ซึ่งบาง Operation อาจจะไม่มี ถ้า Operation นั้นมีตัวแปรผลลัพธ์ตัวเดียว
Retry+ Wait	Retry times	จำนวนรอบที่เรียกเซอร์วิซซ้ำ ถ้าเซอร์วิซเกิดข้อผิดพร่อง
	Duration time	ระยะเวลาที่ผู้ใช้รอก่อนเรียกเซอร์วิซให้ทำงานอีกครั้ง หลังจากที่เรียกเซอร์วิซซ้ำ แล้วเซอร์วิซยังไม่สามารถทำงานได้ โดยมีหน่วยเป็นวินาที

ยกตัวอย่างเช่นหน้าต่างสำหรับกรอกค่าพารามิเตอร์ของแบบรูป RB_{NVP} ซึ่งผู้ใช้จะต้องเตรียม URL ของไฟล์วิสเดิลสำหรับเซอร์วิซตัวแทนที่มีฟังก์ชันการทำงานเหมือนกัน แต่พัฒนาแตกต่างกัน และชื่อ Operation ที่ใช้ โดยแสดงดังภาพที่ 4.13

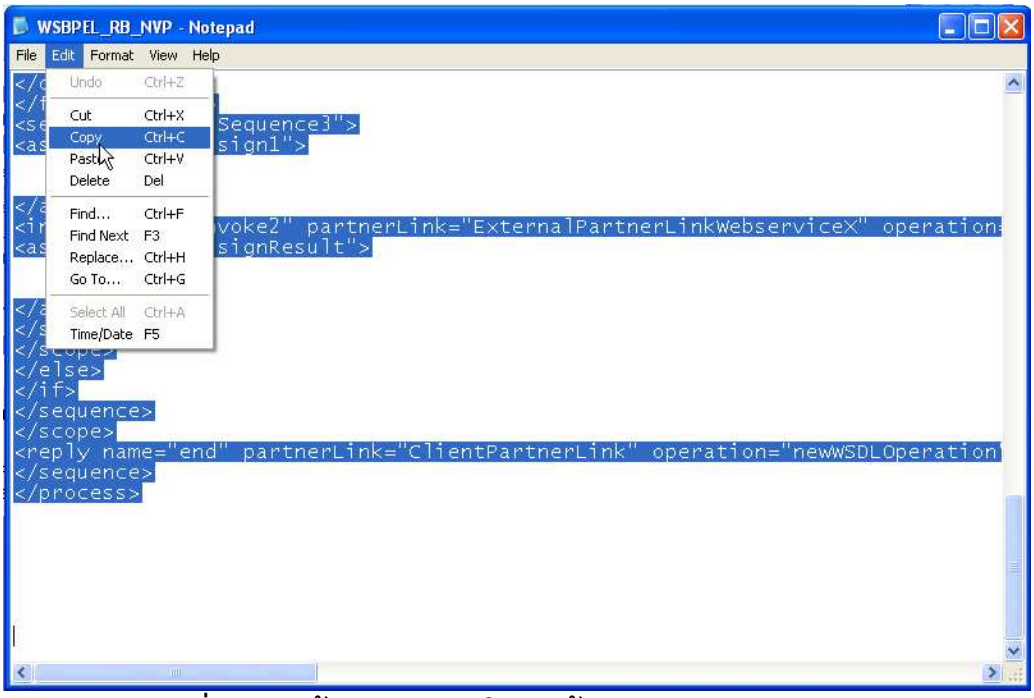


ภาพที่ 4.13 หน้าต่างสำหรับกรอกค่าพารามิเตอร์ของแบบรูป RB_{NVP}

4.3.2. ส่วนสร้างโครงสร้างบีเพลเพิ่มเติมด้วยโปรแกรม NetBeans

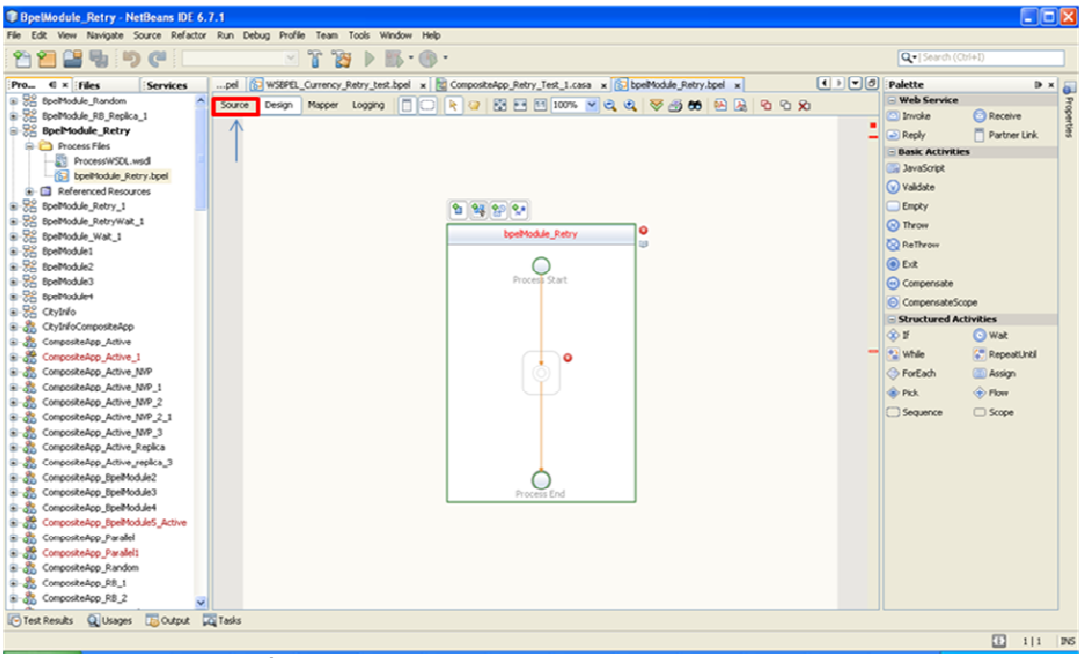
หลังจากผู้ใช้กรอกค่าพารามิเตอร์ของแบบรูปแล้ว เครื่องมือจะสร้างโค้ดโครงสร้างบีเพลไว้ในโฟล์เดอร์ C:\BPEL โดยมีชื่อไฟล์ตามแบบรูปที่เลือก แต่โค้ดที่ได้ยังไม่สมบูรณ์ จึงยังต้องสร้างโครงสร้างบีเพลเพิ่มเติมด้วยโปรแกรม NetBeans โดยจะยกตัวอย่างการสร้างโครงสร้างบีเพลเพิ่มเติมของแบบรูป RB_{NVP} ตามขั้นตอนดังต่อไปนี้

1. ให้ผู้ใช้ copy โครงสร้างบีเพลที่เครื่องมือสร้างในไฟล์ที่อยู่ใน C:\BPEL ชื่อไฟล์ตามแบบรูปที่เลือก ดังภาพที่ 4.14



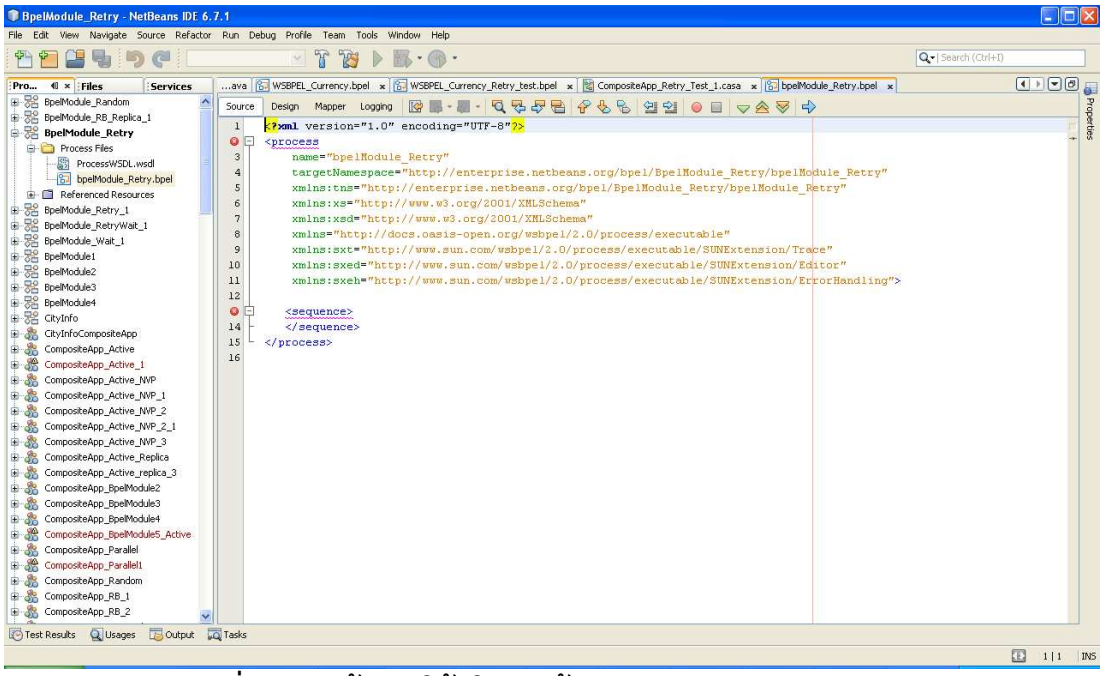
ภาพที่ 4.14 **หน้าต่างคัดลอกโครงสร้างบีเพลของแบบรูป** RB_{NVP}

2. ให้ดับเบิลคลิกที่ไฟล์ BPEL ที่สร้างไว้ในหัวข้อที่ 4.1. จะได้หน้าต่างดังภาพที่ 4.15 แล้วให้คลิกที่ Source



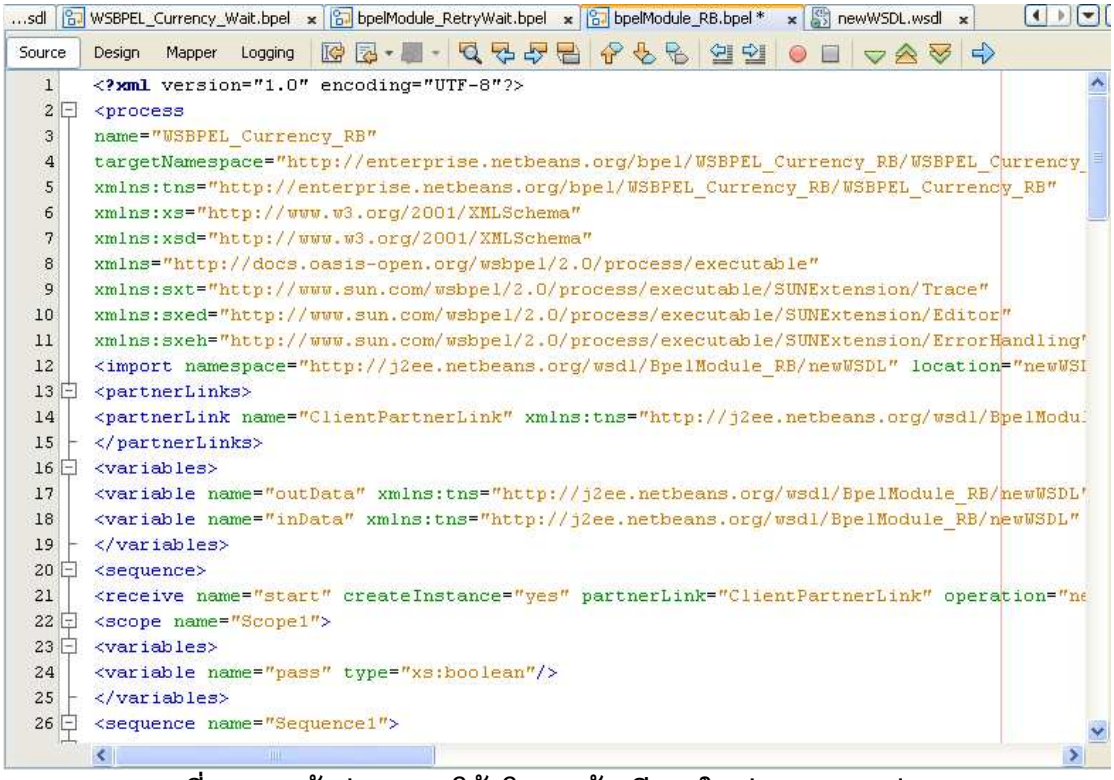
ภาพที่ 4.15 **หน้าต่างเลือกเมนู** Source **ของแบบรูป** RB_{NVP}

3. จะได้โครงสร้างบีเพลดังภาพที่ 4.16 จากนั้นให้ลบโค้ดเดิมออก แล้วคลิกขวาเลือก Paste เพื่อวางโค้ดที่ Copy มาจากขั้นตอนที่ 1



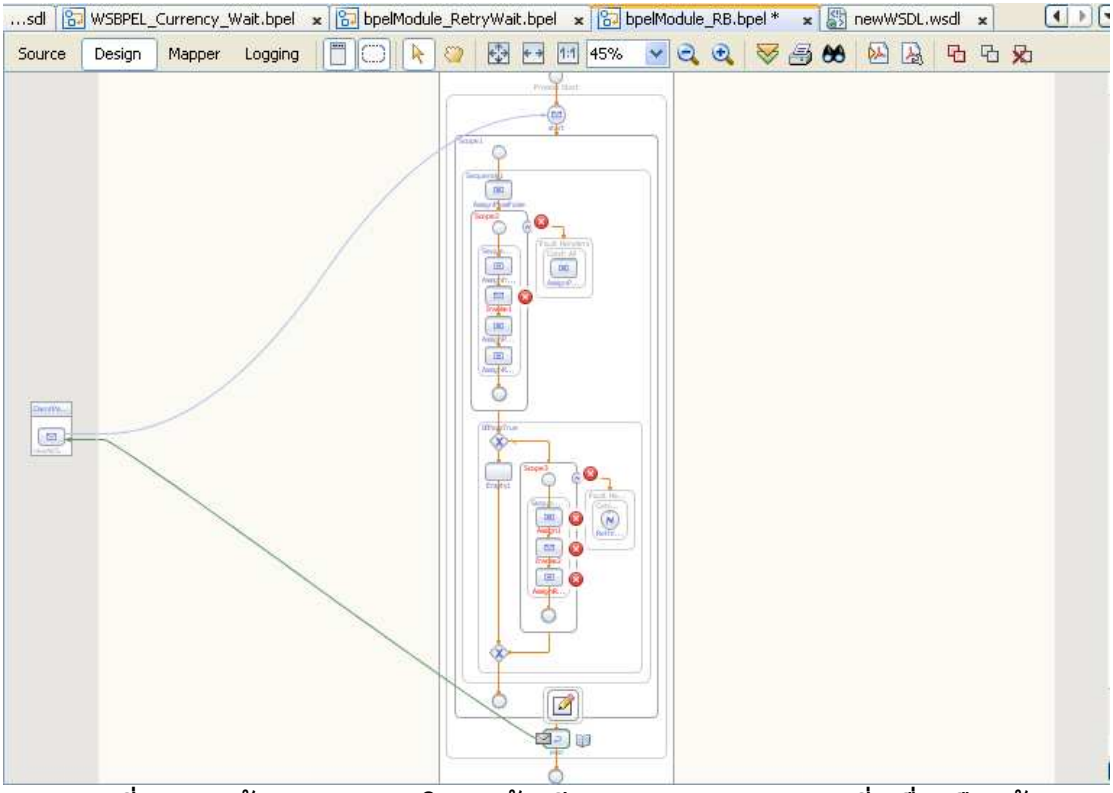
ภาพที่ 4.16 หน้าต่างโค้ดโครงสร้างบีเพลเดิมของแบบรูป RB_{NVP}

4. จะได้โค้ดดังภาพที่ 4.17 ให้ผู้ใช้คลิกที่ Design



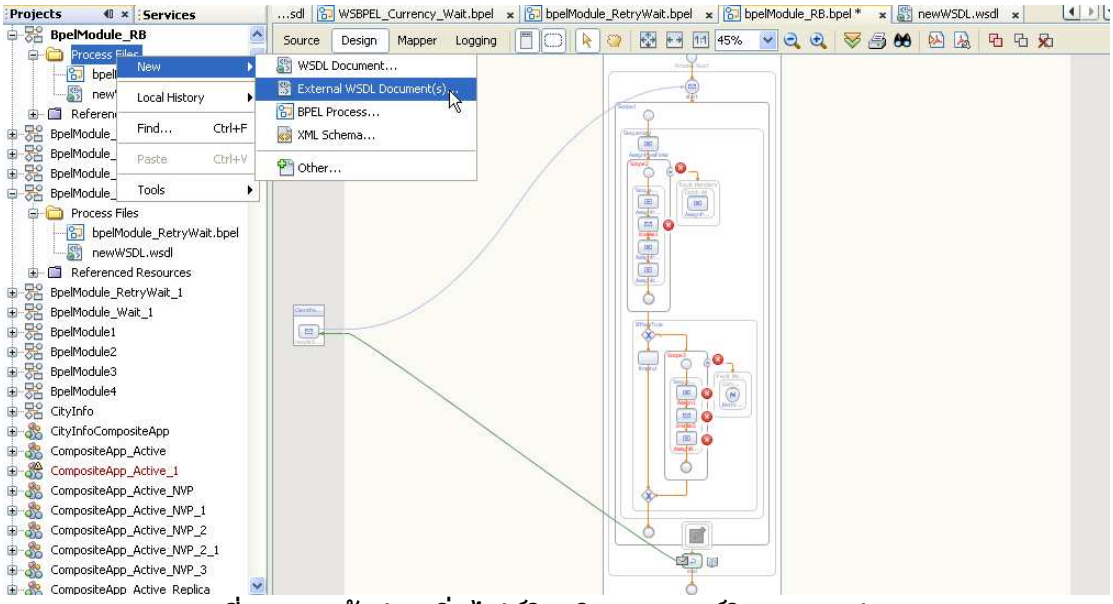
ภาพที่ 4.17 หน้าต่างแสดงโค้ดโครงสร้างบีเพลใหม่ของแบบรูป RB_{NVP}

5. จะได้โครงสร้างบีเพลดังภาพที่ 4.18



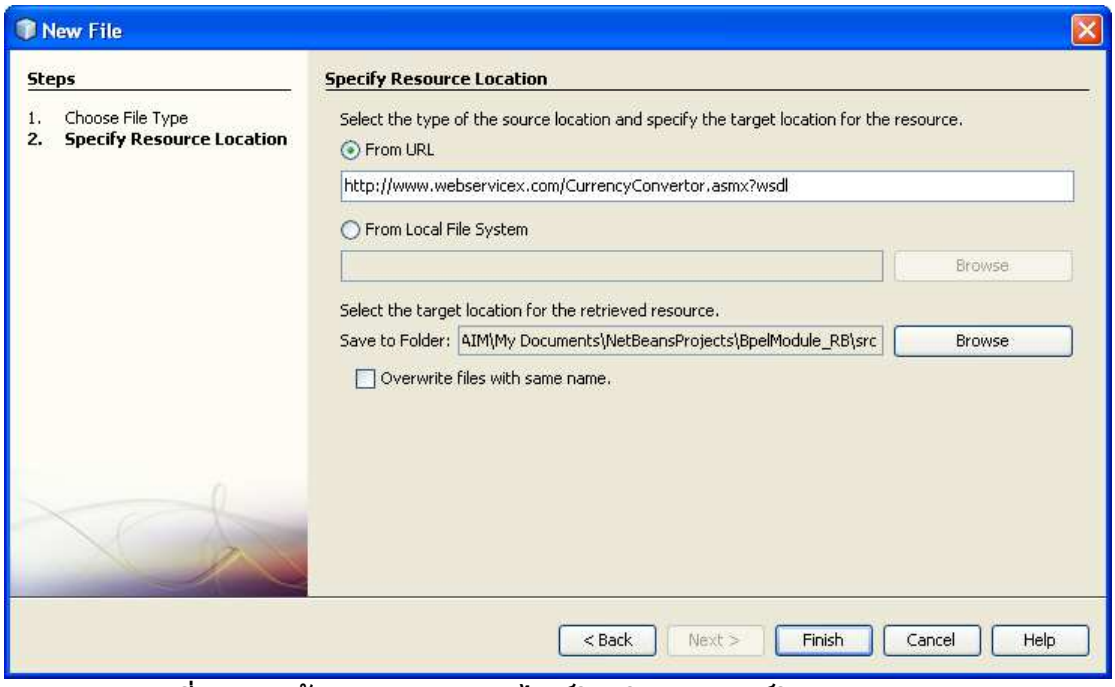
ภาพที่ 4.18 หน้าต่างแสดงรูปโครงสร้างบีเพลของแบบรูป RB_{NVP}ที่เครื่องมือสร้าง

6. ให้ผู้ใช้เพิ่มไฟล์วิสเดิลของเซอร์วิซหลัก โดยการคลิกขวาที่ Process Files เลือก New > External WSDL Document(s) ดังภาพที่ 4.19



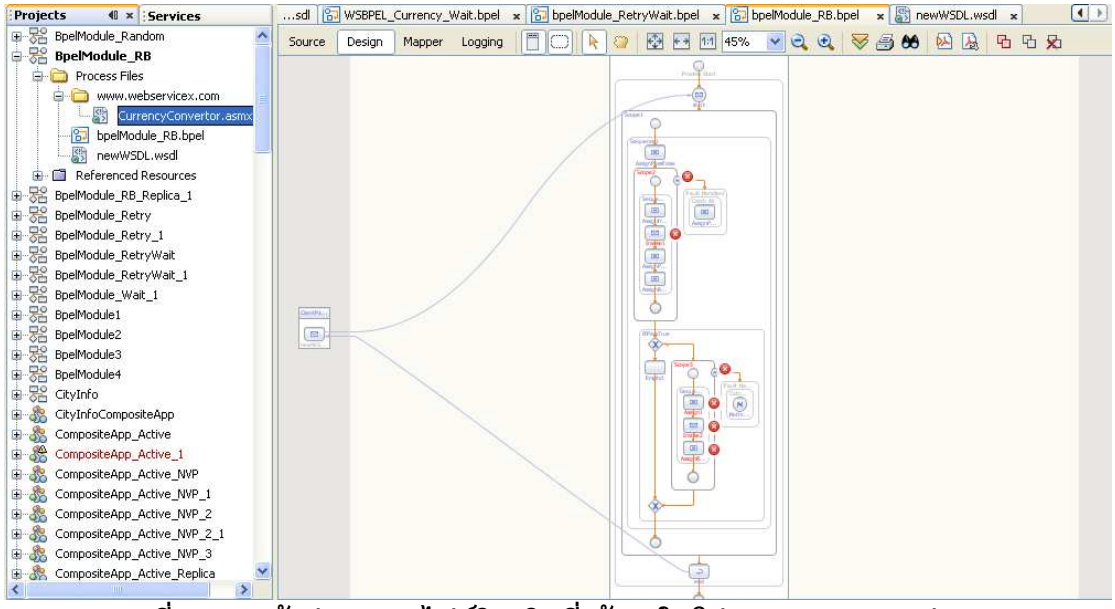
ภาพที่ 4.19 หน้าต่างเพิ่มไฟล์วิสเดิลของเซอร์วิซ แบบรูป RB_{NVP}

7. จะปรากฏหน้าต่าง ดังภาพที่ 4.20 ให้กรอก URL ไฟล์วิสเดิลของเซอร์วิซหลัก แล้วกด Finish



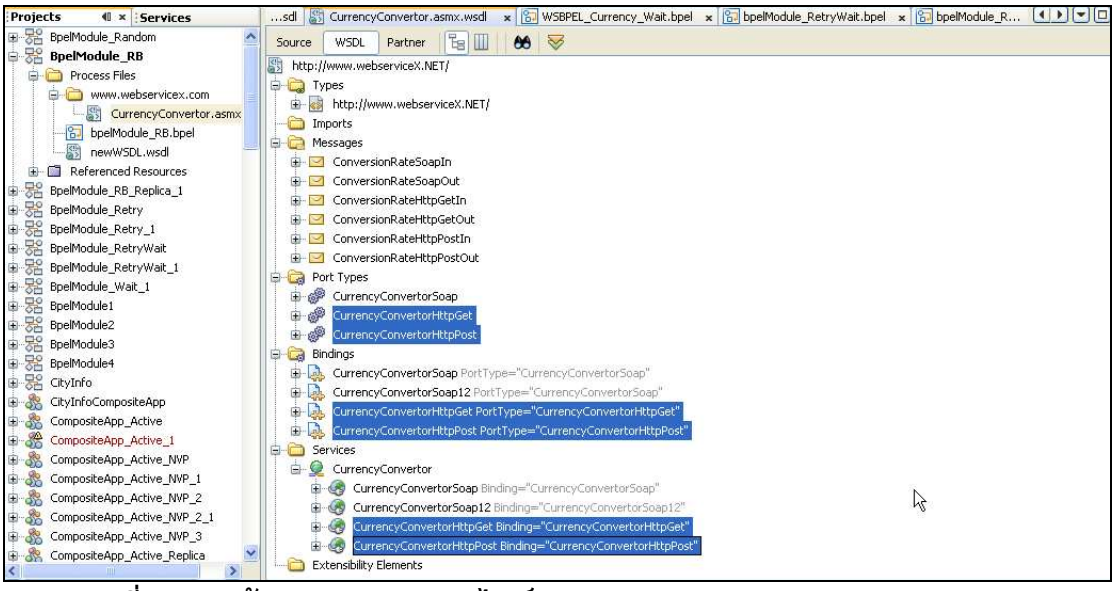
ภาพที่ 4.20 หน้าต่างกรอก URL ไฟล์วิสเดิลของเซอร์วิซ แบบรูป RB_{NVP}

8. จะมีไฟล์วิสเดิลของเซอร์วิซเข้ามาในโปรเจค ดังภาพที่ 4.21 ให้ดับเบิลคลิกที่ไฟล์วิสเดิลดังกล่าว



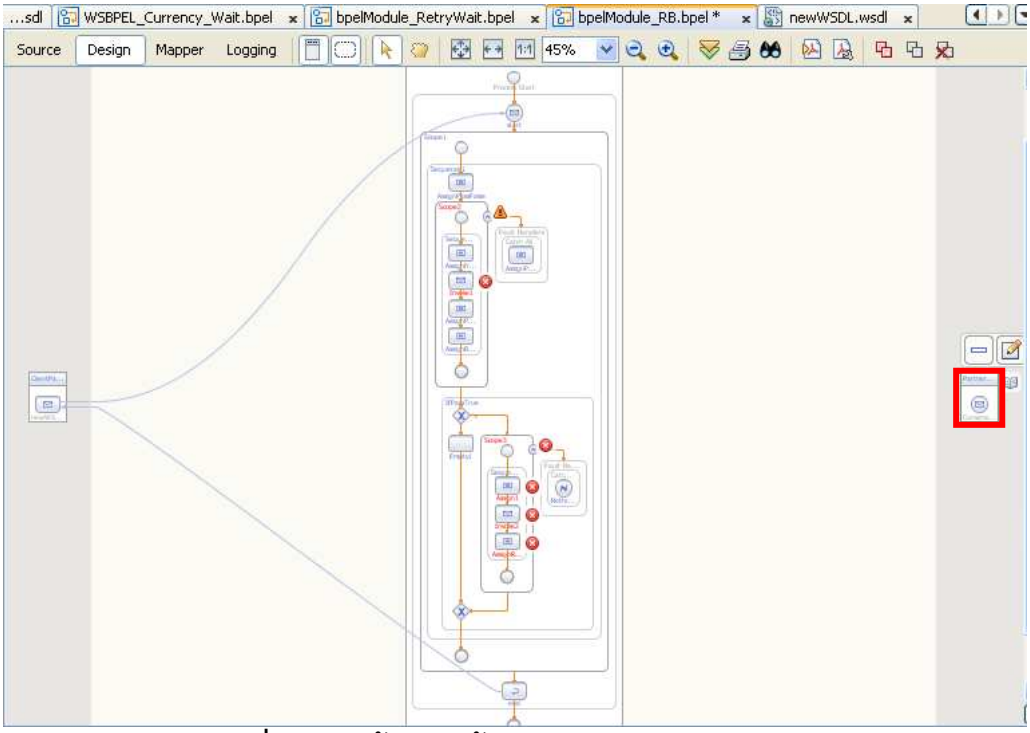
ภาพที่ 4.21 หน้าต่างแสดงไฟล์วิสเดิลที่เข้ามาในโปรเจคของแบบรูป RB_{NVP}

9. จะปรากฏหน้าต่างดังภาพที่ 4.22 ให้ลบไฟล์ HttpGet และ HttpPost ใน PortTypes, Bindings และ Services เพื่อให้สามารถคอมไพล์ไฟล์วิสเดิลได้ โดยการคลิกขวาที่ไฟล์ แล้วกด Delete



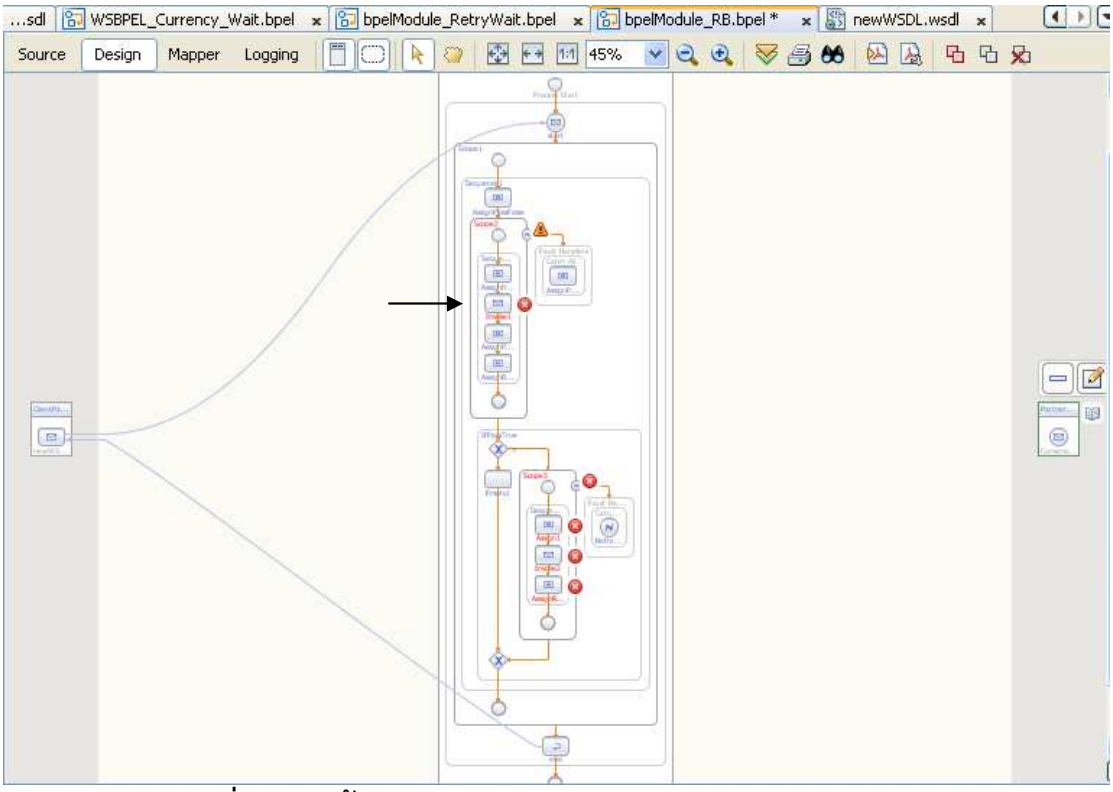
ภาพที่ 4.22 หน้าต่างแสดงการลบไฟล์ HttpGet และHttpPost ของแบบรูป RB_{NVP}

10. ให้ดับเบิลคลิกที่ไฟล์บีเพลที่สร้างในโปรเจคและเลือกมุมมอง Design แล้วลากไฟล์วิสเดิลของเซอร์วิซหลักที่ได้ในขั้นตอนที่ 8 มาไว้ด้านขวามือ ซึ่งมีชื่อว่า PartnerLink1 ดังภาพที่4.23



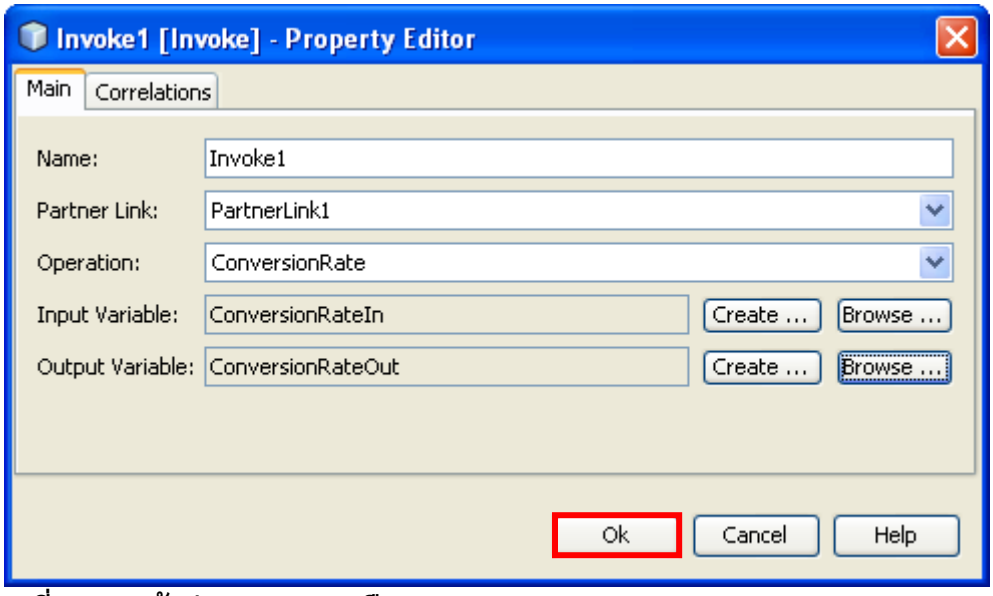
ภาพที่ 4.23 หน้าต่างสร้าง PartnerLink1 ของแบบรูป RB_{NVP}

11. ให้ดับเบิลคลิกที่รูป Invoke1 เพื่อเลือก PartnerLink1, Operation, Input Variable และ Output Variable ของเซอร์วิซหลัก ที่ลากมาในขั้นตอนที่ 10 ดังภาพที่ 4.24



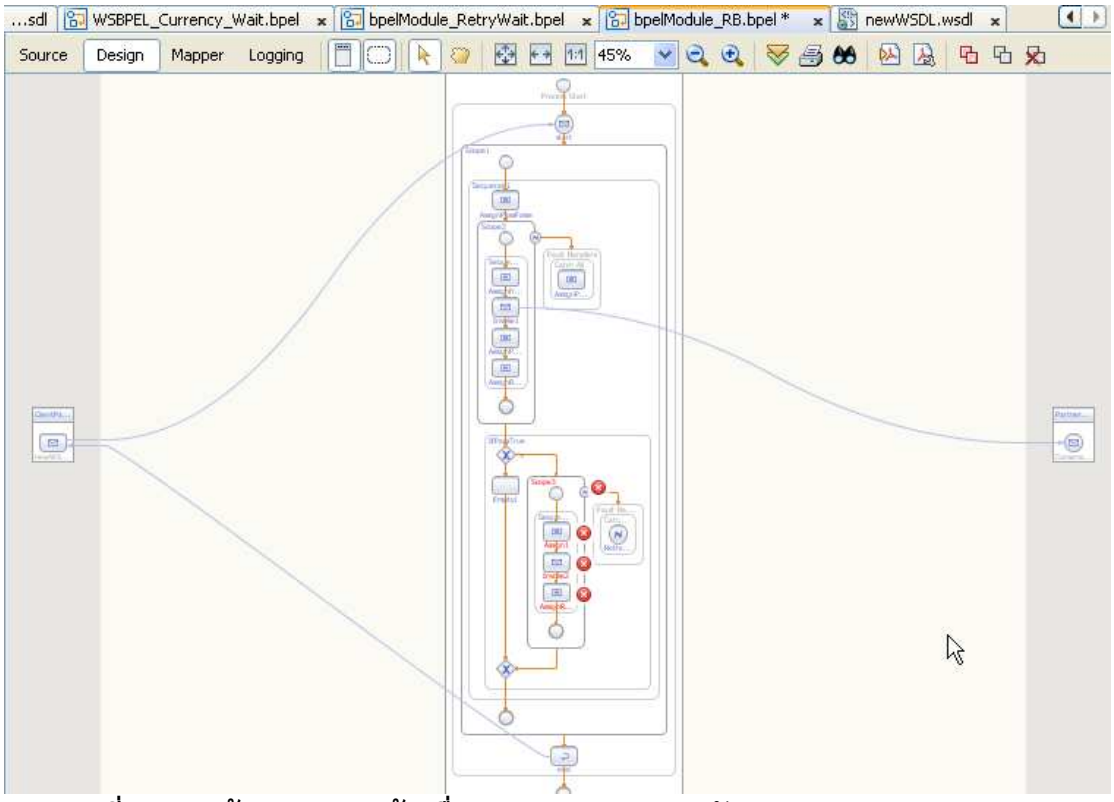
ภาพที่ 4.24 หน้าต่างแสดงการเลือก Invoke1 ของแบบรูป RB_{NVP}

12. จะปรากฏหน้าต่างดังภาพที่ 4.25 ให้เลือก PartnerLink1, Operation ของเซอร์วิซ และคลิกปุ่ม Browse เพื่อเลือก Input Variable และ Output Variable จากนั้นคลิกที่ปุ่ม OK



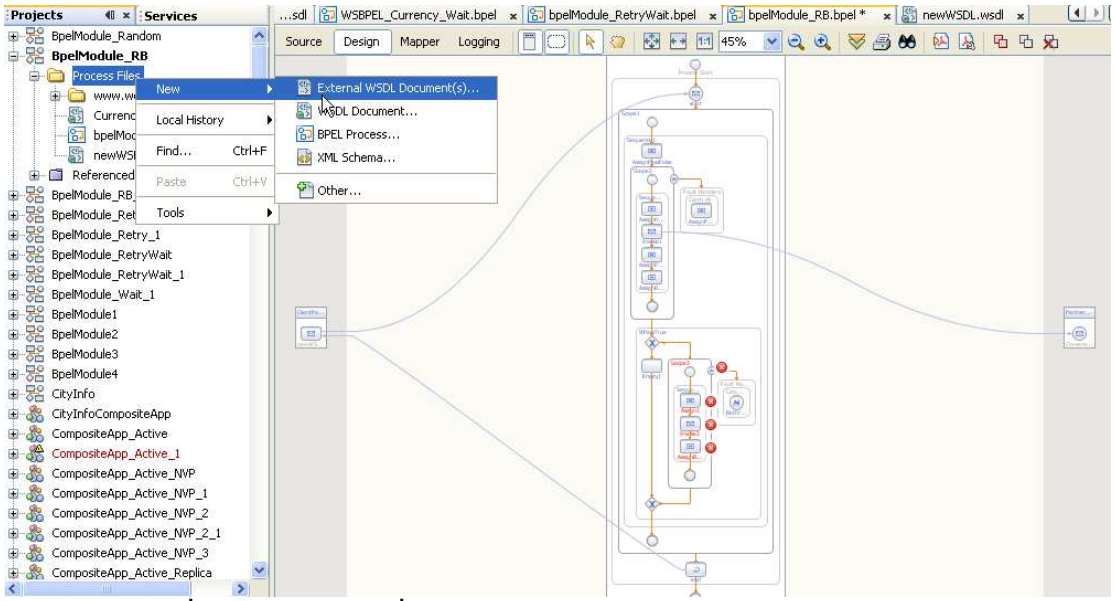
ภาพที่ 4.25 หน้าต่างแสดงการเลือก PartnerLink1, Operation, Input Variable, Output Variable ของแบบรูป RB_{NVP}

13. โปรแกรมจะสร้างเส้นเชื่อมระหว่างรูป Invoke1 และ PartnerLink1 ดังภาพที่ 4.26



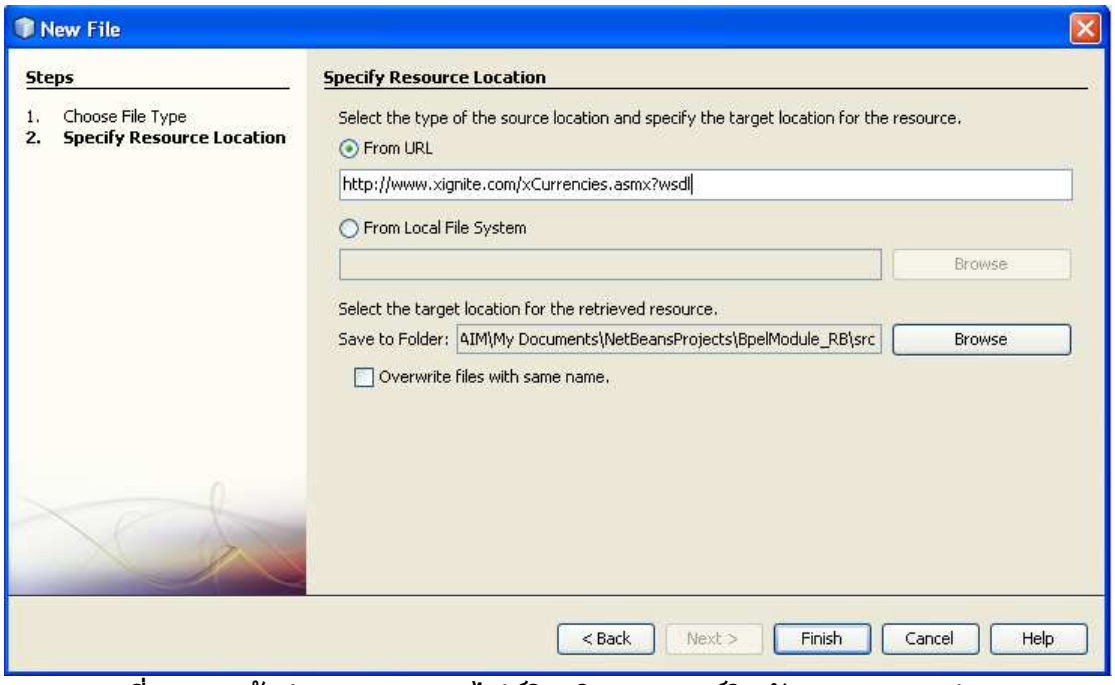
ภาพที่ 4.26 หน้าต่างแสดงเส้นเชื่อมระหว่าง Invoke1 กับ PartnerLink1 ของแบบรูป RB_{NVP}

14. ให้ผู้ใช้เพิ่มไฟล์วิสเดิลของเซอร์วิซตัวแทน โดยการคลิกขวาที่ Process Files เลือก New > External WSDL Document(s) ดังภาพที่ 4.27



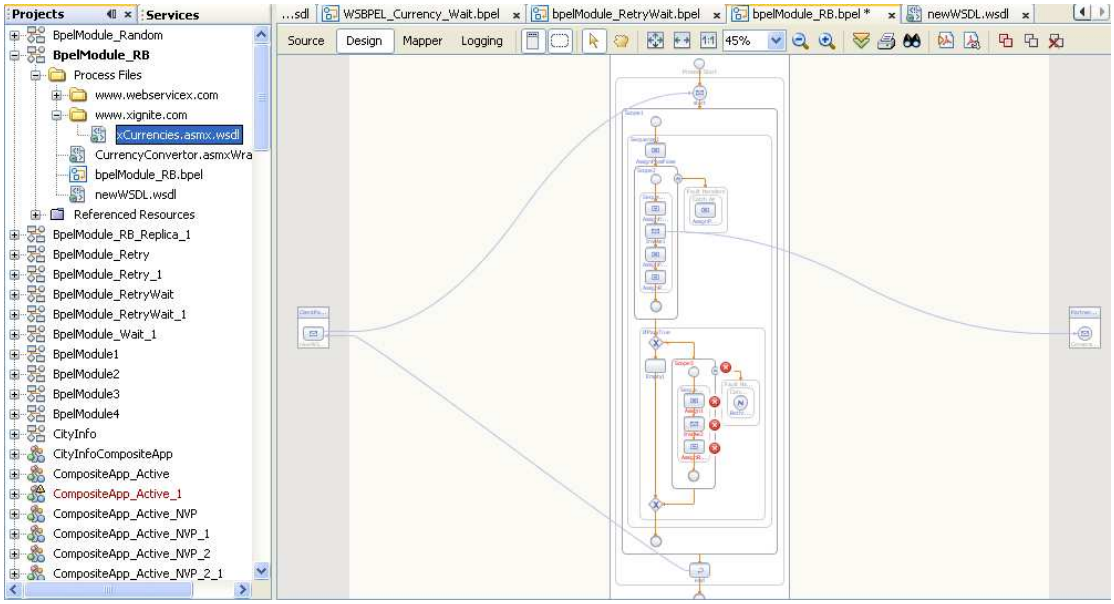
ภาพที่ 4.27 หน้าต่างเพิ่มไฟล์วิสเดิลของเซอร์วิซตัวแทน แบบรูป RB_{NVP}

15. จะปรากฏหน้าต่าง ดังภาพที่ 4.28 ให้กรอก URL ไฟล์วิสเดิลของเซอร์วิซตัวแทน แล้วกด Finish



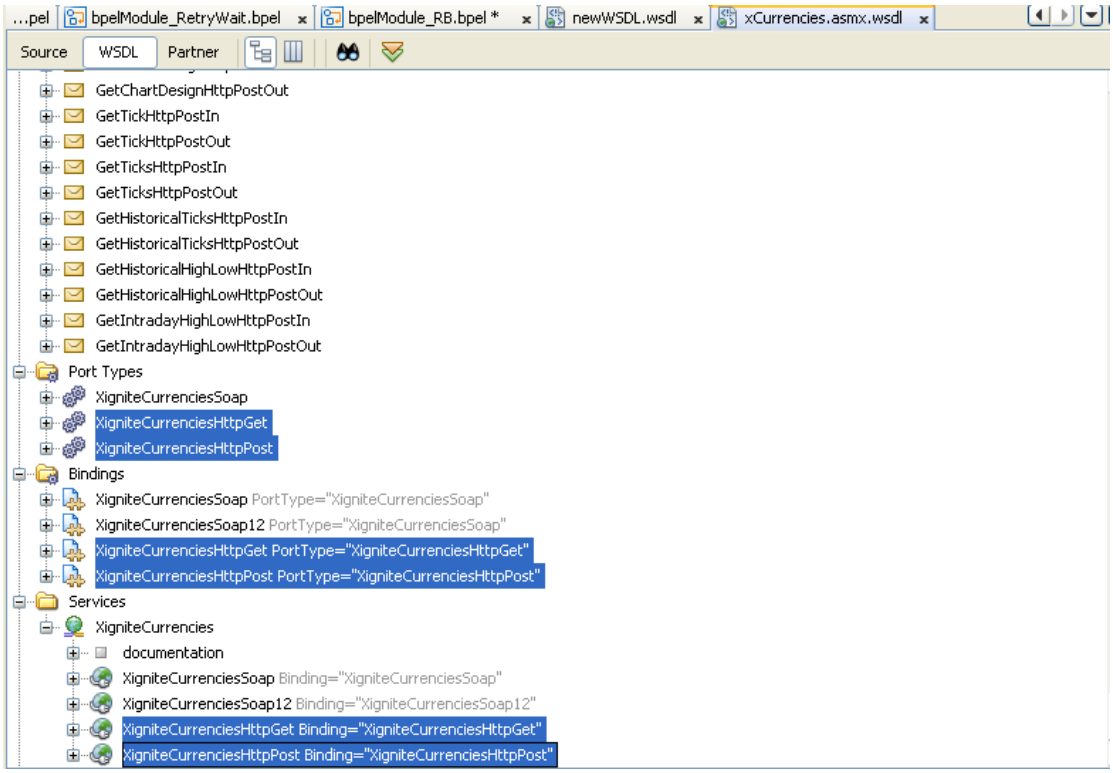
ภาพที่ 4.28 หน้าต่างกรอก URL ไฟล์วิสเดิลของเซอร์วิซตัวแทน แบบรูป RB_{NVP}

16. จะมีไฟล์วิสเดิลของเซอร์วิซตัวแทนเข้ามาในโปรเจค ดังภาพที่ 4.29 ให้ดับเบิลคลิกที่ไฟล์วิสเดิลดังกล่าว



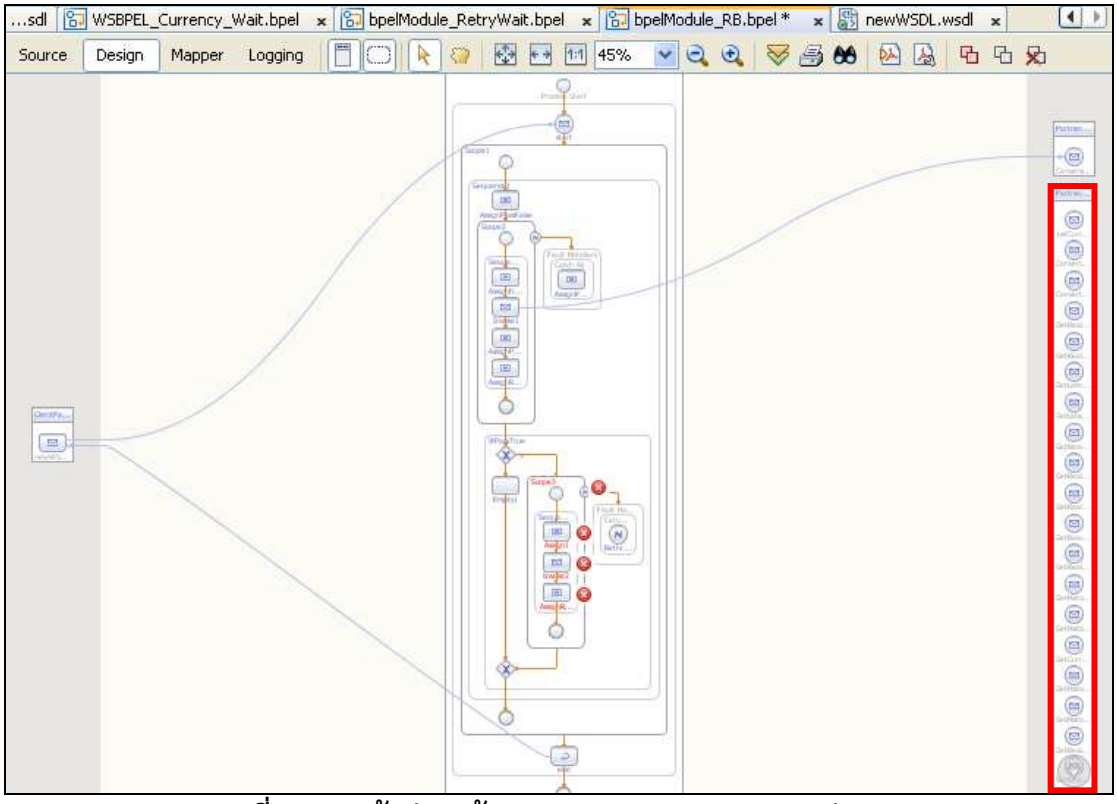
ภาพที่ 4.29 หน้าต่างแสดงไฟล์วิสเดิลของเซอร์วิซตัวแทนที่เข้ามาในโปรเจคของแบบรูป RB_{NVP}

17. จะปรากฏหน้าต่างดังภาพที่ 4.30 ให้ลบไฟล์ HttpGet และ HttpPost ใน PortTypes, Bindings และ Services เพื่อให้สามารถคอมไพล์ไฟล์วิสเดิลได้ โดยการคลิกขวาที่ไฟล์ แล้วกด Delete



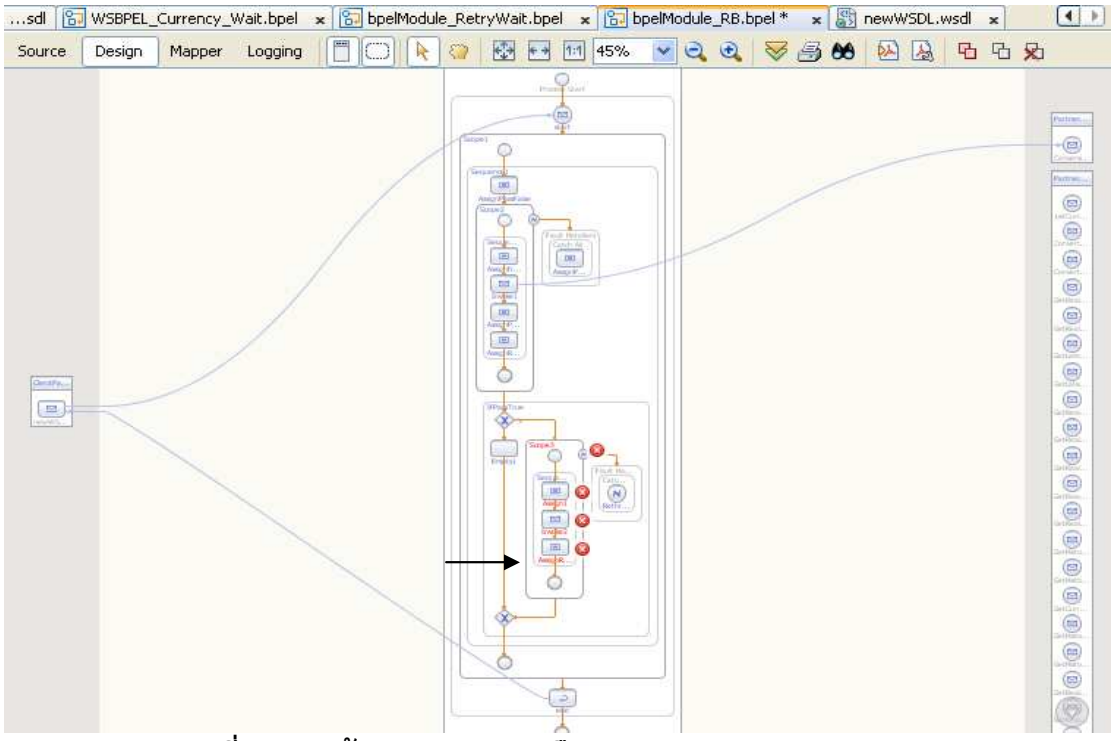
ภาพที่ 4.30 หน้าต่างแสดงการลบไฟล์ HttpGet และHttpPost ของเซอร์วิซตัวแทนแบบรูป RB_{NVP}

18. ให้ดับเบิลคลิกที่ไฟล์บีเพลที่สร้างในโปรเจคและเลือกมุมมอง Design แล้วลากไฟล์วิสเดิลของเซอร์วิซที่ได้ในขั้นตอนที่ 16 มาไว้ด้านขวามือ ซึ่งมีชื่อว่า PartnerLink2 ดังภาพที่ 4.31



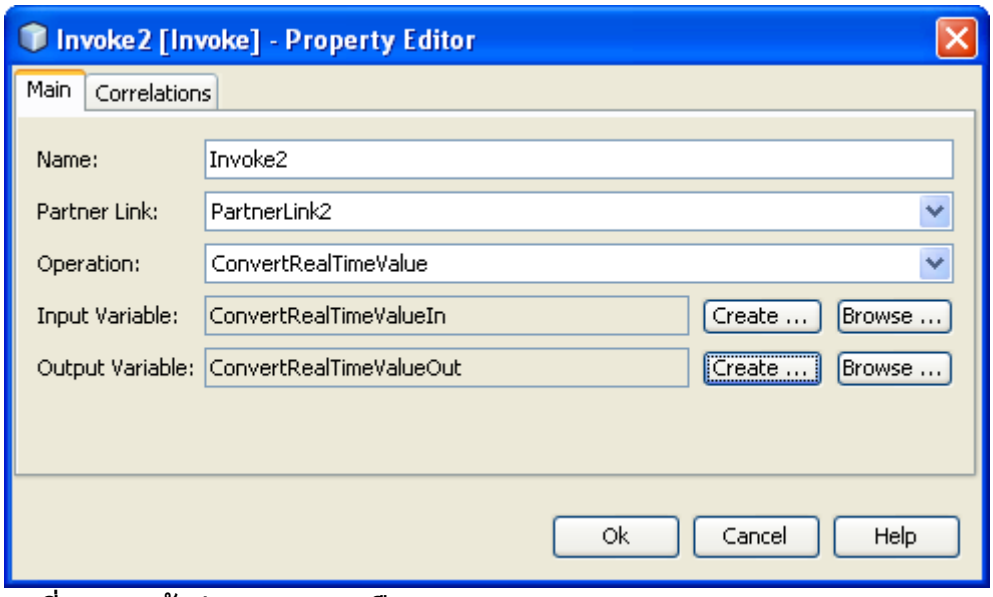
ภาพที่ 4.31 หน้าต่างสร้าง PartnerLink2 ของแบบรูป RB_{NVP}

19. ให้ดับเบิลคลิกที่รูป Invoke2 เพื่อเลือก PartnerLink2, Operation, Input Variable และ Output Variable ของเซอร์วิซ ที่ลากมาในขั้นตอนที่ 18 ดังภาพที่ 4.32



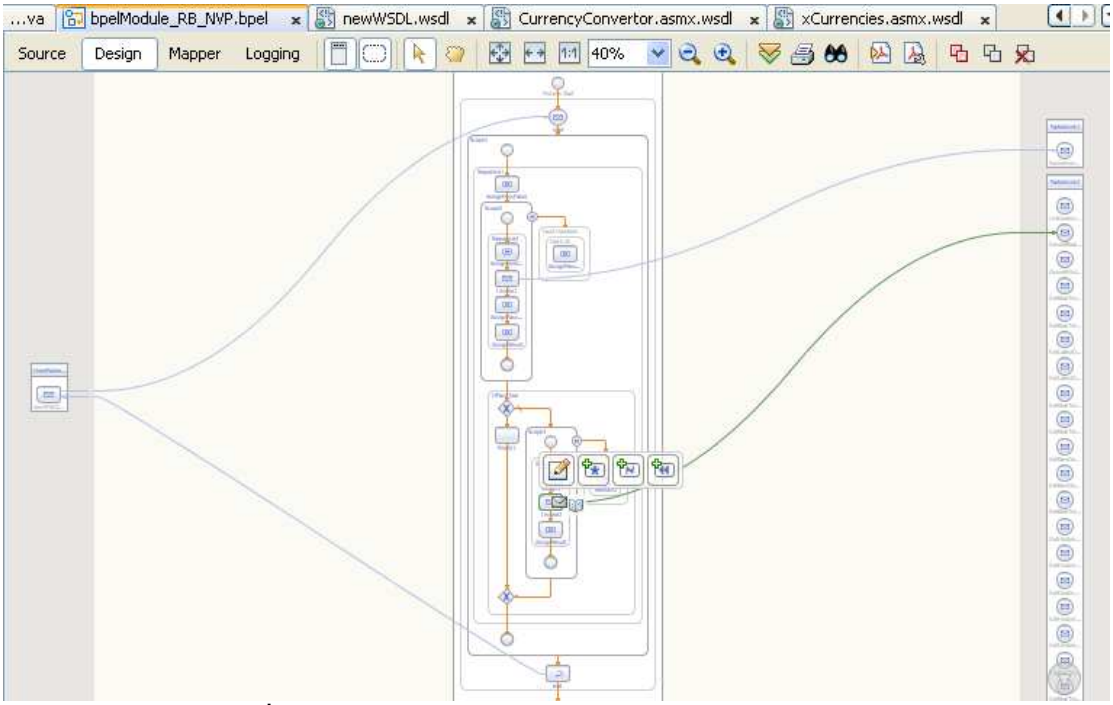
ภาพที่ 4.32 หน้าต่างแสดงการเลือก Invoke2 ของแบบรูป RB_{NVP}

20. จะปรากฏหน้าต่างดังภาพที่ 4.33 ให้เลือก PartnerLink2, Operation ของเซอร์วิส และคลิกปุ่ม Browse เพื่อเลือก Input Variable และ Output Variable จากนั้นคลิกที่ปุ่ม OK



ภาพที่ 4.33 หน้าต่างแสดงการเลือก PartnerLink, Operation, Input Variable, Output Variable **ของเซอร์วิซตัวแทน แบบรูป** RB_{NVP}

21. โปรแกรมจะสร้างเส้นเชื่อมระหว่างรูป Invoke2 และ PartnerLink2 ดังภาพที่ 4.34 เป็นอันเรียบร้อย ผู้ใช้จะได้โครงสร้างบีเพลที่ทนต่อความผิดพร่องตามแบบรูปที่เลือก



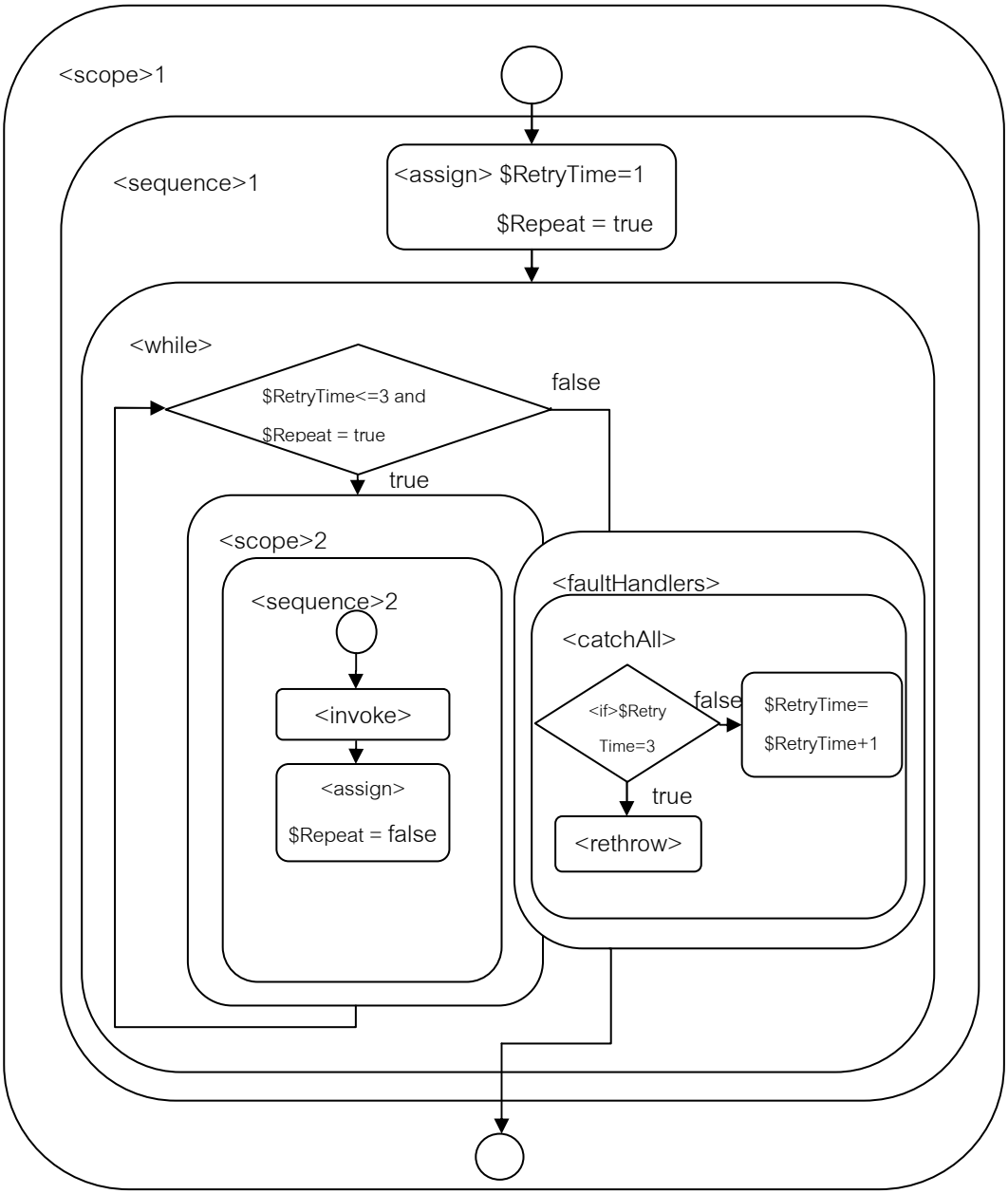
ภาพที่ 4.34 หน้าต่างแสดงโครงสร้างบีเพลของแบบรูป RB_{NVP}

4.3.3. โครงสร้างบีเพลของแบบรูปที่สร้างโดยเครื่องมือ

หลังจากสร้างโครงสร้างบีเพลเพิ่มเติมด้วยโปรแกรม NetBeans จะได้โค้ดโครงสร้างบีเพลของแบบรูปต่างๆ ที่สมบูรณ์ ในที่นี้ขอเสนอโครงสร้างบีเพลในรูปแบบของแผนภาพดังนี้

4.3.3.1. โครงสร้างบีเพลของแบบรูป Retry

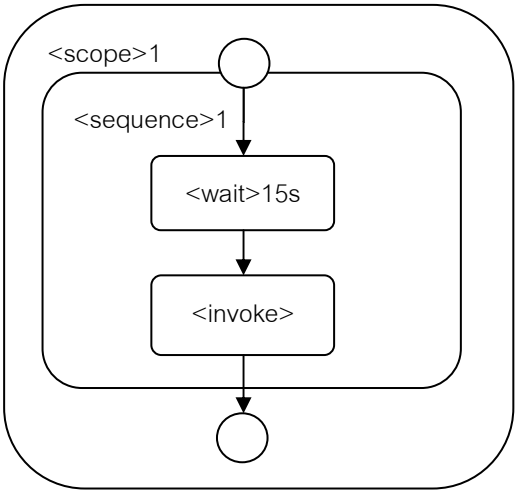
แบบรูป Retry จะทำการเรียกเซอร์วิซซ้ำ ถ้าเกิดข้อผิดพลาดขึ้น โดยจะกำหนดจำนวนรอบที่ให้เรียกเซอร์วิซซ้ำ ถ้าผู้ใช้กรอกจำนวนรอบที่เรียกเซอร์วิซซ้ำเท่ากับ 3 จะได้โครงสร้างบีเพลดังภาพที่ 4.35 โดยลำดับการทำงานจะเริ่มจากการกำหนดค่าเริ่มต้นด้วยแท็ก <assign> ให้กับตัวแปรสำหรับควบคุมจำนวนรอบที่ทำซ้ำ คือ $RetryTime = 1 และตัวแปรควบคุมการวนซ้ำของแท็ก <while> คือ $Repeat = true หลังจากนั้นแท็ก <while> จะมีการตรวจสอบเงื่อนไขในการวนซ้ำว่า $RetryTime น้อยกว่าจำนวนรอบที่ทำซ้ำคือ 3 รอบหรือไม่ และค่าของ $Repeat เป็นจริงหรือไม่ ถ้าเป็นจริง จะทำงานในแท็ก <scope>2 และ <sequence>2 ซึ่งจะทำการเรียกเซอร์วิซด้วยแท็ก <invoke> และกำหนดให้ตัวแปร $Repeat = false เพื่อให้หยุดการวนซ้ำ เนื่องจากสามารถเรียกเซอร์วิซสำเร็จแล้ว แต่ถ้าเงื่อนไขของแท็ก <while> เป็นเท็จ จะให้ <faultHandlers> <catchAll> จัดการ โดยจะตรวจสอบว่าตัวแปร $RetryTime มีค่าเท่ากับจำนวนรอบที่กำหนด คือ 3 รอบหรือไม่ ถ้าไม่ใช่ จะเพิ่มค่า $RetryTime อีก 1 แต่ถ้าใช่ จะใช้แท็ก <rethrow> เพื่อแจ้งข้อผิดพร่อง



ภาพที่ 4.35 โครงสร้างบีเพลของแบบรูป Retry

4.3.3.2. โครงสร้างบีเพลของแบบรูป Wait

แบบรูป Wait คือ ผู้ใช้คาดว่าเซอร์วิซอาจจะเกิดข้อผิดพลาดขึ้น ถ้าทำการเรียกเซอร์วิซทันที จึงทำการรอจนกว่าจะครบเวลาที่กำหนดก่อนการเรียกเซอร์วิซ ถ้าผู้ใช้กรอกระยะเวลาที่ให้รอจนกว่าจะเรียกเซอร์วิซเท่ากับ 15 วินาที จะได้โครงสร้างบีเพลดังภาพที่ 4.36 โดยลำดับการทำงานคือ จะทำการรอเวลาจนกว่าจะครบ 15 วินาทีด้วยแท็ก <wait> แล้วจึงทำการเรียกเซอร์วิซด้วยแท็ก <invoke>



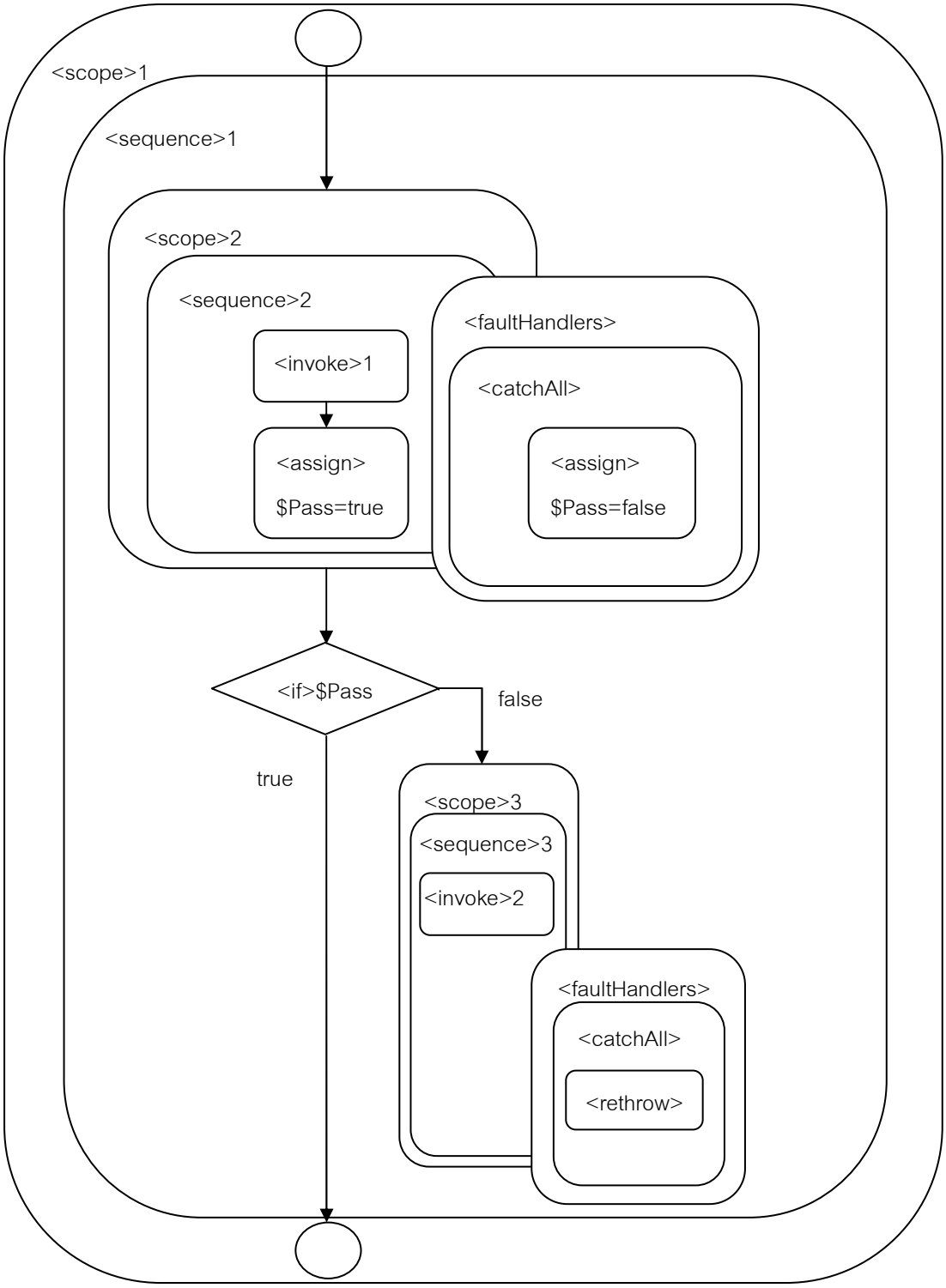
ภาพที่ 4.36 โครงสร้างบีเพลของแบบรูป Wait

4.3.3.3. โครงสร้างบีเพลของแบบรูป $RB_{Replica}$ และ RB_{NVP}

แบบรูป $RB_{Replica}$ และ RB_{NVP} จะมีโครงสร้างบีเพลเหมือนกัน แตกต่างกันที่เซอร์วิซตัวแทนจะแตกต่างกัน โดยแบบรูป $RB_{Replica}$ เซอร์วิซตัวแทนจะถูกพัฒนาเหมือนกับเซอร์วิซหลัก แต่แบบรูป RB_{NVP} เซอร์วิซตัวแทนจะมีฟังก์ชันการทำงานเหมือนกับเซอร์วิซหลัก แต่ถูกพัฒนาไม่เหมือนกัน การทำงานของทั้ง 2 แบบรูป คือ ถ้าเรียกเซอร์วิซหลัก แล้วเกิดข้อผิดพร่อง จะทำการเรียกเซอร์วิซตัวแทน โดยถ้าผู้ใช้กรอก Recovery WSDL File from URL เป็น http://192.168.1.3:8080/CurrencyService2/CurrencyService2Service?wsdl และ Operation เป็น calculateCurrency2 ที่เรียกใช้ในเซอร์วิซตัวแทน จะได้รูปโครงสร้างบีเพลดังภาพที่ 4.37 โดยลำดับการทำงานเริ่มจากในแท็ก <scope>2 <sequence>2 จะทำการเรียกเซอร์วิซหลักด้วยแท็ก <invoke>1 ถ้าเรียกสำเร็จจะทำการกำหนดค่าให้กับตัวแปร $Pass = true เพื่อใช้ในการตรวจสอบเงื่อนไข <if> สำหรับเรียกเซอร์วิซตัวแทนต่อไป แต่ถ้าเซอร์วิซหลักเกิดข้อผิดพร่องขึ้น จะถูกส่งไปให้แท็ก <faultHandlers> <catchAll> จัดการ ซึ่งมีการกำหนดให้ตัวแปร $Pass = false หลังจากนั้นจะทำการตรวจสอบค่าของตัวแปร $Pass ในเงื่อนไข <if> ถ้า $Pass เป็นจริง จะสิ้นสุดการทำงาน แต่ถ้า $Pass เป็นเท็จ จะเข้าสู่การทำงานของแท็ก <scope>3 <sequence>3 ซึ่งมีการเรียกเซอร์วิซตัวแทน แต่ถ้าเกิดข้อผิดพลาดขึ้น จะถูกส่งไปให้แท็ก <faultHandlers> <catchAll> จัดการ ซึ่งจะใช้แท็ก <rethrow> เพื่อแจ้งข้อผิดพร่อง

ผู้วิจัยได้เสนอตัวอย่างของโค้ดบีเพลที่เครื่องมือสร้างสำหรับแบบรูป RB_{NVP} ไว้ในภาคผนวก ก โค้ดบีเพลนี้จะถูกนำเข้าไปยังเครื่องมือ NetBeans และ GlassFish ESB v2.2 เพื่อให้

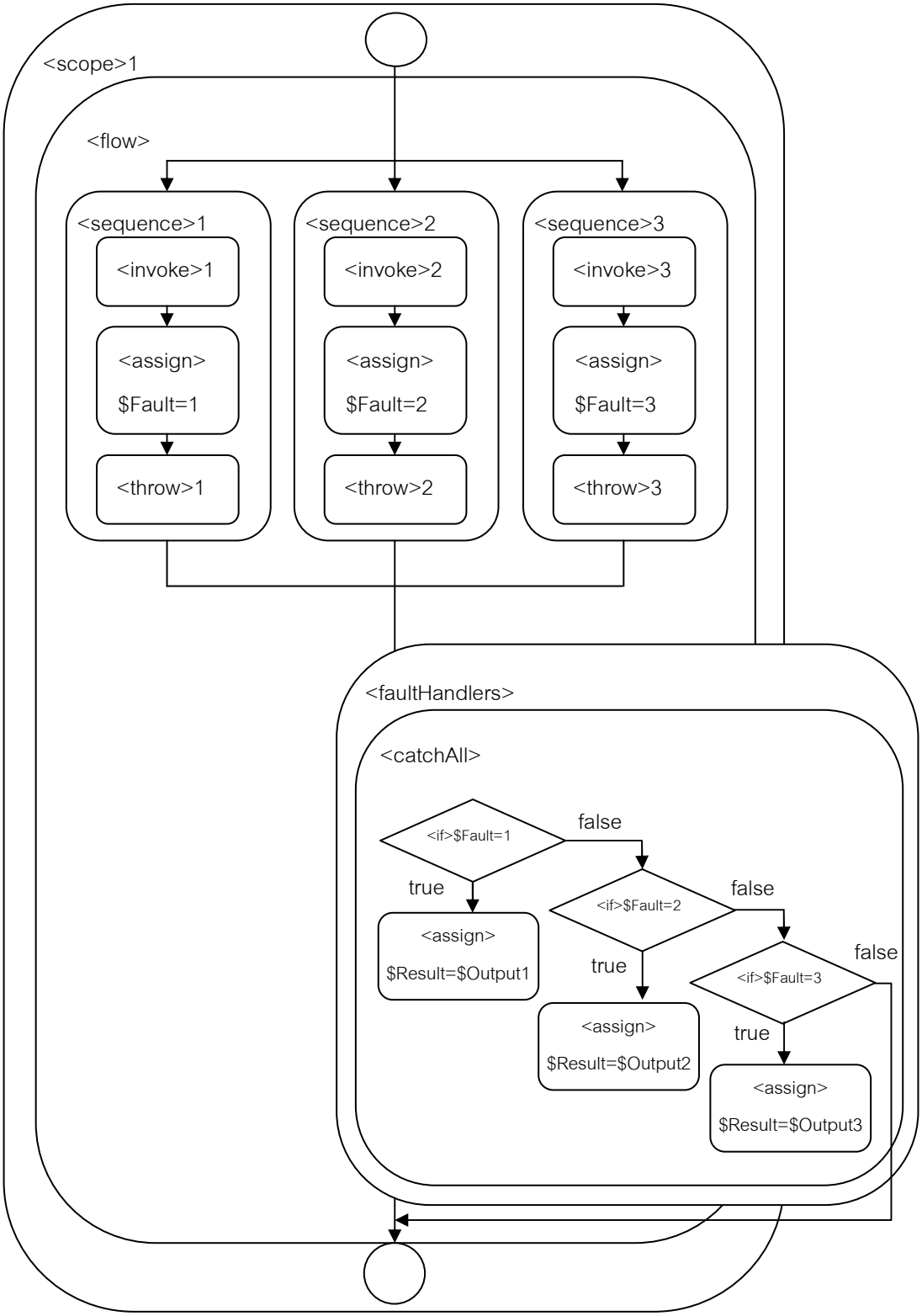
สร้างไฟล์ต่างๆ เพิ่มเติมที่จำเป็นต่อการประมวลผลโดยเครื่องประมวลผลบีเพลของ GlassFish ESB v2.2



ภาพที่ 4.37 โครงสร้างบีเพลของแบบรูป $RB_{Replica}$ และ RB_{NVP}

4.3.3.4. โครงสร้างบีเพลของแบบรูป $Active_{Replica}$และ $Active_{NVP}$

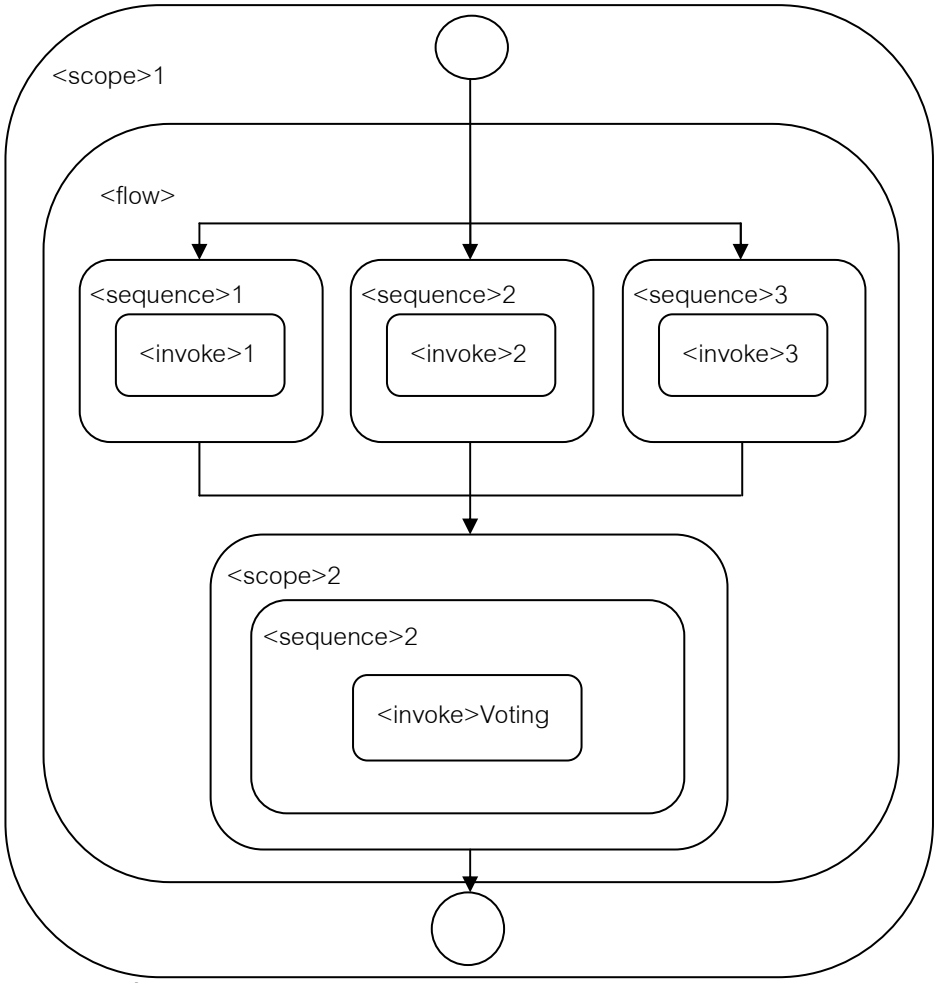
แบบรูป $Active_{Replica}$และ $Active_{NVP}$ จะมีโครงสร้างบีเพลเหมือนกัน แตกต่างกันที่เซอร์วิซที่เรียกพร้อมกันแบบขนานจะแตกต่างกัน โดยแบบรูป $Active_{Replica}$ เซอร์วิซจะถูกพัฒนาเหมือนกัน แต่แบบรูป $Active_{NVP}$ เซอร์วิซจะมีฟังก์ชันการทำงานเหมือนกัน แต่ถูกพัฒนาไม่เหมือนกัน การทำงานของทั้ง 2 แบบรูป คือ จะเรียกเซอร์วิซที่มีฟังก์ชันการทำงานเหมือนกัน พร้อมกันแบบขนาน และจะทำการเลือกเซอร์วิซตัวแรกสุดที่ทำการตอบกลับผลลัพธ์ก่อน ซึ่งถ้าผู้ใช้กำหนดจำนวนเซอร์วิซที่เรียกพร้อมกัน 3 ตัว จะได้รูปโครงสร้างบีเพลดังภาพที่ 4.38 โดยลำดับการทำงานเริ่มจากแท็ก <flow> จะเรียกการทำงานของ <sequence>1 <sequence>2 <sequence>3 พร้อมกันแบบขนาน ภายในแท็ก <sequence>1 จะเรียกเซอร์วิซมาทำงานด้วยแท็ก <invoke>1 ถ้าเรียกเซอร์วิซ1 สำเร็จก่อน จะทำการกำหนดค่าให้กับตัวแปรสำหรับระบุว่าเซอร์วิซใดตอบกลับผลลัพธ์เสร็จก่อน คือ $Fault = 1 ด้วยแท็ก <assign> แล้วส่งต่อการทำงานด้วยแท็ก <throw>1 เช่นเดียวกันภายในแท็ก <sequence>2 จะเรียกเซอร์วิซมาทำงานด้วยแท็ก <invoke>2 ถ้าเรียกเซอร์วิซ2 สำเร็จก่อน จะทำการกำหนดค่าให้กับตัวแปร $Fault = 2 ด้วยแท็ก <assign> แล้วส่งต่อการทำงานด้วยแท็ก <throw>2 เช่นเดียวกันภายในแท็ก <sequence>3 จะเรียกเซอร์วิซมาทำงานด้วยแท็ก <invoke>3 ถ้าเรียกเซอร์วิซ3 สำเร็จก่อน จะทำการกำหนดค่าให้กับตัวแปร $Fault = 3 ด้วยแท็ก <assign> แล้วส่งต่อการทำงานด้วยแท็ก <throw>3 หลังจากที่เรียกเซอร์วิซทั้งหมดพร้อมกันแบบขนาน เซอร์วิซใดตอบกลับผลลัพธ์ก่อน จะส่งต่อการทำงานไปให้ <faultHandlers> <catchAll> จัดการทันที ซึ่งภายในนั้นจะตรวจสอบว่าเป็นการส่งต่อการทำงานของเซอร์วิซใด ด้วยเงื่อนไข <if> เพื่อตรวจสอบว่า ค่า $Fault เป็นเท่าไหร่ ถ้ามีค่าเท่ากับ 1 จะกำหนดค่าผลลัพธ์ของเซอร์วิซ1 ให้ ด้วยแท็ก <assign> $Result=$Output1 แต่ถ้ามีค่าเท่ากับ 2 จะกำหนดค่าผลลัพธ์ของเซอร์วิซ2 ให้ ด้วยแท็ก <assign> $Result=$Output2 แต่ถ้ามีค่าเท่ากับ 3 จะกำหนดค่าผลลัพธ์ของเซอร์วิซ3 ให้ ด้วยแท็ก <assign> $Result=$Output3



ภาพที่ 4.38 โครงสร้างบีเพลของแบบรูป $Active_{Replica}$ และ $Active_{NVP}$

4.3.3.5. โครงสร้างบีเพลของแบบรูป $Voting_{Replica}$ และ $Voting_{NVP}$

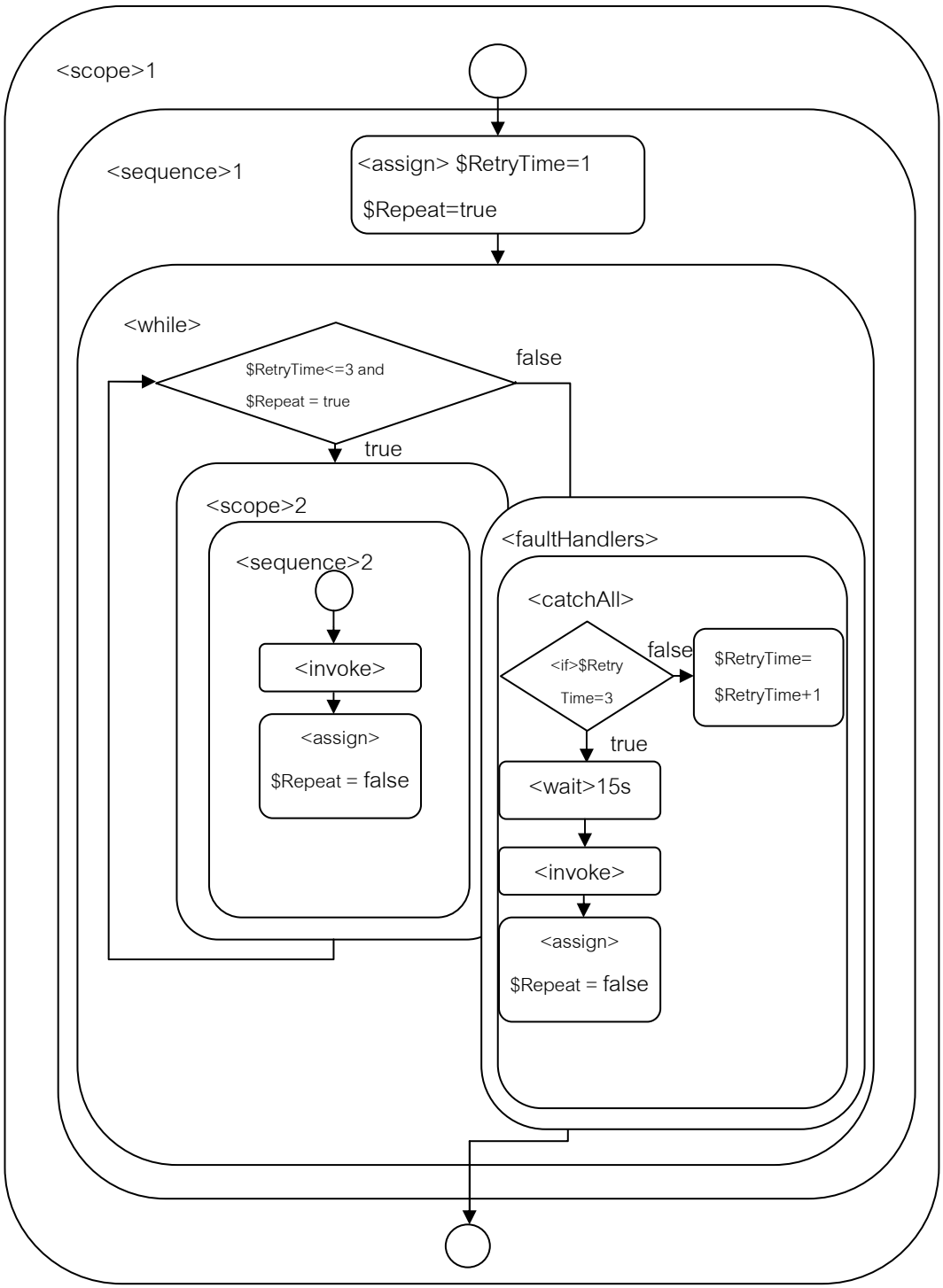
แบบรูป $Voting_{Replica}$และ $Voting_{NVP}$ จะมีโครงสร้างบีเพลเหมือนกัน แตกต่างกันที่เซอร์วิซที่เรียกพร้อมกันแบบขนานจะแตกต่างกัน โดยแบบรูป $Voting_{Replica}$ เซอร์วิซจะถูกพัฒนาเหมือนกัน แต่แบบรูป $Voting_{NVP}$ เซอร์วิซจะมีฟังก์ชันการทำงานเหมือนกัน แต่ถูกพัฒนาไม่เหมือนกัน การทำงานของทั้ง 2 แบบรูป คือ จะเรียกเซอร์วิซที่มีฟังก์ชันการทำงานเหมือนกัน พร้อมกันแบบขนาน และจะทำการเลือกเซอร์วิซด้วยอัลกอริทึมสำหรับโหวต ซึ่งถ้าผู้ใช้กำหนดจำนวนเซอร์วิซที่เรียกพร้อมกัน 3 ตัว จะได้รูปโครงสร้างบีเพลดังภาพที่ 4.39 โดยลำดับการทำงานเริ่มจากแท็ก <flow> จะเรียกการทำงานของ <sequence>1 <sequence>2 <sequence>3 พร้อมกันแบบขนาน ภายในแท็ก <sequence> จะเรียกเซอร์วิซมาทำงานด้วยแท็ก <invoke> เมื่อทั้ง 3 เซอร์วิซตอบกลับ ผลลัพธ์ของทั้ง 3 เซอร์วิซจะถูกนำมาพิจารณา เพื่อเลือกคำตอบด้วยอัลกอริทึมสำหรับโหวต



ภาพที่ 4.39 โครงสร้างบีเพลของแบบรูป $Voting_{Replica}$ และ $Voting_{NVP}$

4.3.3.6. **โครงสร้างบีเพลของแบบรูป** Retry+Wait

แบบรูป Retry+Wait จะทำการเรียกเซอร์วิซก่อน ถ้าเกิดข้อผิดพร่องจะทำการเรียกซ้ำตามจำนวนรอบที่กำหนด ถ้าครบจำนวนรอบแล้ว ยังเกิดข้อผิดพร่องจะทำการรอจนกว่าครบเวลาที่กำหนด จึงจะทำการเรียกเซอร์วิซ ซึ่งถ้าผู้ใช้กรอกค่าพารามิเตอร์ Retry times คือ จำนวนรอบที่เรียกเซอร์วิซซ้ำเป็น 3 รอบและระยะเวลาที่ให้หยุดรอเป็น 15 วินาที จะได้รูปโครงสร้างบีเพลดังภาพที่ 4.40 โดยลำดับการทำงานเริ่มจากการกำหนดค่าเริ่มต้นด้วยแท็ก <assign> ให้กับตัวแปรสำหรับควบคุมจำนวนรอบที่ทำซ้ำ คือ $RetryTime = 1 และตัวแปรควบคุมการวนซ้ำของแท็ก <while> คือ $Repeat = true และตัวแปรควบคุมการหยุดรอ คือ $Wait = true หลังจากนั้นแท็ก <while> จะมีการตรวจสอบเงื่อนไขในการวนซ้ำว่า $RetryTime น้อยกว่าจำนวนรอบที่ทำซ้ำคือ 3 รอบหรือไม่ และค่าของ $Repeat เป็นจริงหรือไม่ ถ้าเป็นจริง จะทำงานในแท็ก <scope>2 และ <sequence>2 ซึ่งจะทำการเรียกเซอร์วิซด้วยแท็ก <invoke> และกำหนดให้ตัวแปร $Repeat = false เพื่อให้หยุดการวนซ้ำ เนื่องจากสามารถเรียกเซอร์วิซสำเร็จแล้ว แต่ถ้าเงื่อนไขของแท็ก <while> เป็นเท็จ จะให้ <faultHandlers> <catchAll> จัดการโดยจะตรวจสอบว่าตัวแปร $RetryTime มีค่าเท่ากับจำนวนรอบที่กำหนด คือ 3 รอบหรือไม่ ถ้าไม่ใช่ จะเพิ่มค่า $RetryTime อีก 1 แต่ถ้าใช่ จะให้หยุดรอ 15 วินาทีก่อนที่จะเรียกเซอร์วิซด้วยแท็ก <invoke> และกำหนดให้ตัวแปร $Repeat = false เพื่อให้หยุดการวนซ้ำ



ภาพที่ 4.40 **โครงสร้างบีเพลของแบบรูป** Retry+Wait

บทที่ 5
ผลการประเมินเครื่องมือ

จากการพัฒนาเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่อง จะทำการประเมินเครื่องมือ โดยการให้ผู้พัฒนาเว็บเซอร์วิซเป็นผู้ประเมิน จำนวน 10 ราย ซึ่งมีประสบการณ์การทำเว็บเซอร์วิซระหว่าง 1-6 ปี โดเมนของเว็บเซอร์วิซจะเกี่ยวกับการให้บริการข้อมูลต่างๆ ภายในองค์กร การติดต่อสื่อสาร การให้ข้อมูลสินค้ากับลูกค้า โลจิสติกส์ และธนาคาร โดยมีบางรายมีการพัฒนาเซอร์วิซโดเมนเดียวกัน คือ การให้บริการข้อมูลภายในองค์กร และการติดต่อสื่อสาร ลักษณะการใช้งานเว็บเซอร์วิซส่วนใหญ่จะใช้ภายในองค์กร และพัฒนาเว็บเซอร์วิซเองประกอบกับใช้บริการเว็บเซอร์วิซอื่นด้วย มาตรฐานโพรโทคอลที่ใช้ส่วนใหญ่จะใช้ SOAP และไม่มีการใช้บีเพลในกระบวนการทางธุรกิจ

5.1. ผลการประเมิน

หลังจากที่ผู้พัฒนาได้ใช้เครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องแล้ว จะให้ทำแบบประเมินในภาคผนวก ข ซึ่งสรุปผลการประเมินได้ 3 ส่วน คือ ส่วนข้อมูลการใช้แบบรูปการทนต่อความผิดพร่อง ส่วนการประเมินการแนะนำแบบรูปการทนต่อความผิดพร่อง ส่วนแนวทางการพัฒนาเครื่องมือต่อไป

5.1.1. ส่วนข้อมูลการใช้แบบรูปการทนต่อความผิดพร่อง

ในการพัฒนาเว็บเซอร์วิซส่วนใหญ่มีการคำนึงถึงการทำให้เซอร์วิซทนต่อความผิดพร่อง โดยวิธีการที่ทำให้เซอร์วิซทนต่อความผิดพร่องผู้พัฒนาเซอร์วิซจะพัฒนาเอง มีการใช้ฮาร์ดแวร์และเทคโนโลยีอื่นมาเสริม เช่น Load balancing, Clustering, Recovery site, Elastic storage, Distributed cache เป็นต้น

ในการออกแบบเว็บเซอร์วิซผู้ออกแบบจะมีการใช้แบบรูปการทนต่อความผิดพร่อง โดยแบบรูปที่ใช้กันมากอันดับ 1 คือ Retry และ Retry+Wait อันดับ 2 คือ Wait และ $RB_{Replica}$ อันดับ 3 คือ RB_{NVP} , $Active_{Replica}$ และ $Voting_{NVP}$

5.1.2. ส่วนการประเมินการแนะนำแบบรูปการทนต่อความผิดพร่อง

จากการใช้เครื่องมือของผู้พัฒนาเซอร์วิซพบว่า ส่วนใหญ่เครื่องมือแนะนำแบบรูปการทนต่อความผิดพร่องได้ตรงกับแบบรูปที่พัฒนา มีผู้พัฒนาเพียง 2 คนที่บอกว่าไม่ตรง โดยแบบรูปที่

ผู้พัฒนาใช้จะเป็นแบบรูปที่เครื่องมือแนะนำให้เป็น อันดับ 2 และ 3 ผู้ใช้คิดว่า เหตุผลที่เครื่องมือแนะนำได้ไม่ตรงกับแบบรูปที่พัฒนา เป็นเพราะค่าคะแนนในเมตริกซ์ความสัมพันธ์ระหว่างลักษณะของเซอร์วิซกับแบบรูปการทนต่อความผิดพร่อง

จากการกำหนดลักษณะของเซอร์วิซและแบบรูปการทนต่อความผิดพร่องในเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่อง ผู้ใช้มีความคิดเห็นว่า ลักษณะของเซอร์วิซมีความครอบคลุมในระดับมากและปานกลางเท่ากัน และมีความคิดเห็นว่า แบบรูปการทนต่อความผิดพร่องมีความครอบคลุมในระดับปานกลาง รองลงมาคือ มาก และเห็นว่าควรมีการเพิ่มแบบรูปการข้ามหรือเพิกเฉยต่อความผิดพร่อง จากการให้คำนิยามของลักษณะของเซอร์วิซและแบบรูปการทนต่อความผิดพร่องที่ให้ผู้ใช้ได้ศึกษาก่อนการทำแบบประเมิน สรุปได้ว่า คำนิยามของลักษณะของเซอร์วิซมีความชัดเจนในระดับปานกลาง และคำนิยามของแบบรูปการทนต่อความผิดพร่องมีความชัดเจนระดับมากและปานกลางเท่ากัน

จากวิธีการแนะนำแบบรูปการทนต่อความผิดพร่องโดยพิจารณาจากลักษณะของเซอร์วิซ ผู้ใช้ส่วนใหญ่มีความคิดเห็นว่า เหมาะสม เนื่องจากผู้ใช้คิดว่า การเลือกรูปแบบการทนต่อความผิดพร่องควรสอดคล้องกับลักษณะของเซอร์วิซ ซึ่งลักษณะของเซอร์วิซจะเป็นตัวสะท้อนความสามารถของการทนต่อความผิดพร่อง และในการพัฒนาเว็บเซอร์วิซในสาขาที่แตกต่างกันย่อมมีความต้องการในการทำให้ทนต่อความผิดพร่องที่แตกต่างกัน รวมถึงทรัพยากรที่มีอย่างจำกัดย่อมทำให้ผู้พัฒนาไม่สามารถเลือกแบบรูปได้ทุกรูปแบบ ดังนั้นการเลือกเฉพาะแบบรูปที่สำคัญตามลักษณะของเซอร์วิซจึงมีความเหมาะสม ส่วนผู้ใช้ที่คิดว่าวิธีการแนะนำแบบรูปโดยพิจารณาจากลักษณะของเซอร์วิซไม่เหมาะสม ได้ให้เหตุผลว่า ในเซอร์วิซหนึ่งมีส่วนการทำงานที่หลากหลาย ถ้าดูละเอียดถึงระบบการทำงานจริง อาจมีวิธีที่ดีที่สุดในการทนต่อความผิดพร่องไม่เหมือนกัน

จากเมตริกซ์แสดงความสัมพันธ์ระหว่างลักษณะของเซอร์วิซกับแบบรูปการทนต่อความผิดพร่อง ผู้ใช้ส่วนใหญ่เห็นว่ามีความเหมาะสม เนื่องจากความสัมพันธ์สะท้อนแบบรูปการทนต่อความผิดพร่องตามความเป็นจริง ส่วนผู้ใช้ที่เห็นว่า ไม่มีความเหมาะสม เป็นเพราะเครื่องมือแนะนำแบบรูปได้ไม่ตรงกับแบบรูปที่ต้องการอาจจะมีสาเหตุจากคะแนนความสัมพันธ์ในเมตริกซ์และเมตริกซ์ยังขาดความยืดหยุ่นในการลดหรือเพิ่มจำนวนของแบบรูปการทนต่อความผิดพร่องที่แนะนำ โดยอาจต้องแก้ไขคะแนนในเมตริกซ์ใหม่ทั้งหมด

จากวิธีคำนวณหาผลคูณเมตริกซ์ P = D x R เพื่อหาแบบรูปการทนต่อความผิดพร่องที่เหมาะสม ผู้ใช้ส่วนใหญ่เห็นว่าเหมาะสม เนื่องจากมีการให้น้ำหนักตามความสำคัญของลักษณะเด่นของเซอร์วิซ ในกรณีที่เซอร์วิซไม่มีลักษณะใด ลักษณะนั้นจะไม่ถูกนำมาคิด และเป็นการคิดจากน้ำหนักของเซอร์วิซประเภทต่างๆ ซึ่งสามารถปรับแต่งได้ตามน้ำหนักของลักษณะของเซอร์วิซ และเนื่องจากเครื่องมือสามารถเลือกแบบรูปได้ตามลักษณะเด่นของเซอร์วิซที่ต้องการในการพัฒนาเว็บเซอร์วิซ จึงเห็นว่าการคำนวณด้วยวิธีนี้น่าจะเหมาะสม

5.1.3. **ส่วนแนวทางการพัฒนาเครื่องมือต่อไป**

ผู้ใช้บางคนเห็นว่า คำนิยามยังคลุมเครือ จากตัวอย่าง เซอร์วิซตรวจสอบยอดเงินคงเหลือในบัญชีธนาคาร เพราะเหตุใดจึงไม่ให้ความสำคัญกับลักษณะ Correctness ซึ่งผลลัพธ์ที่ได้ควรจะคำนึงถึงความถูกต้องด้วย และการให้ระบุว่าลักษณะของเซอร์วิซใดสำคัญเป็นเรื่องที่ยาก เพราะคะแนนที่ได้ใกล้เคียงกัน เนื่องจากมีหลายส่วนการทำงาน ซึ่งมีความต้องการและความสำคัญแตกต่างกัน

ในการแนะนำแบบรูปการทนต่อความผิดพร่องอาจแนะนำได้มากกว่า 1 แบบรูป เพราะคะแนนที่ได้ใกล้เคียงกัน หรือควรมีช่วงคะแนนที่ยอมรับได้ เพื่อบอกว่าแบบรูปที่ได้คะแนนอยู่ในช่วงดังกล่าวมีความเหมาะสม และการให้คะแนนในเมตริกซ์ควรจะมีแนวทางอื่นในการกำหนดค่าที่อิงสถิติหรือหลักการ เพื่อหาค่าที่ดีและเหมาะสมที่สุด

5.2. **ผลการวิเคราะห์ของการใช้งานเครื่องมือในแต่ละโดเมนของเซอร์วิซ**

ในการประเมินเครื่องมือ โดเมนของเซอร์วิซที่ผู้ใช้พัฒนา คือ การให้บริการข้อมูลต่างๆ ภายในองค์กร การติดต่อสื่อสาร การให้ข้อมูลสินค้ากับลูกค้า โลจิสติกส์ และธนาคาร โดยมีบางรายมีการพัฒนาเซอร์วิซโดเมนเดียวกัน คือ การให้บริการข้อมูลภายในองค์กร และการติดต่อสื่อสาร

5.2.1. **โดเมนการให้บริการข้อมูลต่าง ๆ ภายในองค์กร**

โดเมนนี้มีผู้พัฒนาเซอร์วิซทั้งหมด 4 ราย โดยมีผู้พัฒนา 2 รายที่ไม่ได้ใช้แบบรูปการทนต่อความผิดพร่อง ซึ่งรายแรกจัดอันดับความเด่นของลักษณะเรียงจากเด่นมากไปน้อย คือ Economy, Simplicity, Correctness และ Timeliness แบบรูปที่เครื่องมือแนะนำคือ Wait เนื่องจากผู้พัฒนาให้ความสำคัญกับลักษณะ Economy ซึ่งแบบรูป Wait ได้คะแนนมากที่สุด และแบบรูปที่ได้

คะแนนมากที่สุดของลักษณะ Simplicity คือ แบบรูป Wait เช่นกัน แบบรูปที่ได้คะแนนมากที่สุดของลักษณะ Correctness คือ แบบรูป $Voting_{NVP}$ และแบบรูปที่ได้คะแนนมากที่สุดของลักษณะ Timeliness คือ แบบรูป $Active_{NVP}$ และผู้พัฒนารายที่สองให้ความสำคัญกับลักษณะ Timeliness มากที่สุด รองลงมาคือ Correctness แบบรูปที่เครื่องมือแนะนำคือ $Active_{NVP}$ เนื่องจากแบบรูป $Active_{NVP}$ ได้คะแนนมากที่สุดในลักษณะ Timeliness

ส่วนผู้พัฒนาอีก 2 ราย เครื่องมือแนะนำแบบรูปได้ตรงกับแบบรูปที่ผู้พัฒนาใช้ ซึ่งรายแรกให้ลักษณะที่เด่นอันดับ 1 คือ Instance Specificity และ Replica Provision ลักษณะที่เด่นอันดับ 2 คือ Timeliness และ Economy ลักษณะที่เด่นอันดับ 3 คือ Correctness โดยแบบรูปที่เครื่องมือแนะนำและผู้พัฒนาใช้ คือ $RB_{Replica}$ เนื่องจากผู้พัฒนาให้ความสำคัญกับลักษณะ Replica Provision ซึ่งแบบรูปที่ได้คะแนนมากที่สุด คือ $RB_{Replica}$, $Active_{Replica}$ และ $Voting_{Replica}$ และให้ความสำคัญกับลักษณะ Instance Specificity ซึ่งจากทั้ง 3 แบบรูปข้างต้น $RB_{Replica}$ ให้ความจำเพาะกับเซอร์วิซมากที่สุด ดังนั้นเครื่องมือจึงแนะนำแบบรูป $RB_{Replica}$ ให้กับผู้พัฒนา และผู้พัฒนารายที่สองจัดอันดับความเด่นของลักษณะเรียงจากเด่นมากไปน้อย คือ Transient Failure รองลงมาคือ NVP Provision และ Simplicity มีอันดับเท่ากัน อันดับถัดไปคือ Economy และ Correctness ซึ่งแบบรูปที่เครื่องมือแนะนำและผู้พัฒนาใช้คือ Retry เนื่องจากแบบรูป Retry เหมาะกับความผิดพร่องที่เกิดขึ้นชั่วคราว พัฒนาง่าย และประหยัดค่าใช้จ่ายมากกว่าแบบรูปอื่น

จากข้อมูลการเลือกลักษณะของเซอร์วิซโดเมนการให้บริการข้อมูลภายในองค์กรของผู้พัฒนาทั้ง 4 รายสรุปได้ว่า ลักษณะที่ผู้พัฒนาเลือกเหมือนกันมากที่สุด คือ Correctness รองลงมาอันดับสองคือ Timeliness และ Economy และอันดับ 3 คือ Simplicity แสดงว่าผู้พัฒนาคำนึงถึงความถูกต้องของผลลัพธ์ รองลงมาคือ ความรวดเร็วในการตอบกลับ และค่าใช้จ่ายที่ใช้ในการประมวลผล ความง่ายในการพัฒนา และผลของการแนะนำแบบรูปแต่ละเซอร์วิซไม่เหมือนกัน คือ Wait, $Active_{NVP}$, RB_{NVP} และ Retry ทั้งนี้อาจเป็นเพราะลักษณะของแต่ละเซอร์วิซไม่เหมือนกันและขึ้นอยู่กับการให้ความสำคัญของแต่ละลักษณะของผู้พัฒนาเซอร์วิซด้วย

5.2.2. โดเมนการติดต่อสื่อสาร

โดเมนนี้มีผู้พัฒนาเซอร์วิซทั้งหมด 2 ราย ผู้พัฒนารายแรกไม่มีการใช้แบบรูปการทนต่อความผิดพร่อง ซึ่งจัดอันดับความเด่นของลักษณะเรียงจากเด่นมากไปน้อย คือ Correctness, Timeliness, Simplicity และ Economy แบบรูปที่เครื่องมือแนะนำคือ $Active_{NVP}$ เนื่องจากเป็นแบบรูปที่ตอบกลับผลลัพธ์ได้เร็วที่สุด และมีการเรียกเซอร์วิซพร้อมกันหลายตัวแบบขนาน จึงทำให้

มีโอกาสให้คำตอบได้ถูกต้องมากกว่าแบบรูปที่มีการเรียกเซอร์วิซเดียว ส่วนผู้พัฒนารายที่สอง เครื่องมือแนะนำแบบรูปได้ตรงกับแบบรูปที่ผู้ใช้พัฒนา คือ $Active_{Replica}$ โดยจัดอันดับความเด่นให้กับลักษณะ Replica Provision, Correctness และ Timeliness มากที่สุดเท่ากัน เนื่องจากถ้าต้องการให้ได้ผลลัพธ์ที่ถูกต้องและตอบกลับอย่างรวดเร็ว ควรใช้แบบรูป Active และถ้ามีสำเนาของเซอร์วิซ จึงควรใช้แบบรูป $Active_{Replica}$

จากข้อมูลการเลือกลักษณะของเซอร์วิซโดเมนการติดต่อสื่อสารของผู้พัฒนาทั้ง 2 ราย สรุปได้ว่า ลักษณะที่ผู้พัฒนาเห็นว่าสำคัญเหมือนกันคือ Correctness และ Timeliness แสดงว่าผู้พัฒนาคำนึงถึงความถูกต้องและความรวดเร็วในการติดต่อสื่อสาร ซึ่งสอดคล้องกับลักษณะของแบบรูป Active

5.2.3. โดเมนการให้ข้อมูลสินค้ากับลูกค้า

โดเมนนี้มีผู้พัฒนาเซอร์วิซทั้งหมด 1 ราย ผู้พัฒนาจัดอันดับความเด่นของลักษณะเรียงจากเด่นมากไปน้อย คือ Correctness, Transient Failure, Timeliness, Simplicity, Instance Specificity, Replica Provision, NVP Provision และ Economy ซึ่งผู้พัฒนาไม่มีการใช้แบบรูปการทนต่อความผิดพร่อง โดยแบบรูปที่เครื่องมือแนะนำคือ Retry เนื่องจากเป็นแบบรูปที่เหมาะกับความผิดพร่องที่เกิดขึ้นชั่วคราว และเซอร์วิซที่มีความจำเพาะ ง่ายต่อการพัฒนา มีค่าใช้จ่ายในการประมวลผลน้อย

จากข้อมูลการเลือกลักษณะของเซอร์วิซโดเมนการให้ข้อมูลสินค้ากับลูกค้าของผู้พัฒนา สรุปได้ว่า ผู้พัฒนาให้ความสำคัญกับทุกลักษณะ โดยลักษณะที่สำคัญที่สุด คือ Correctness แสดงว่า การให้ข้อมูลสินค้ากับลูกค้าควรมีความถูกต้อง น่าเชื่อถือได้ และถ้าเกิดความผิดพร่องควรเกิดขึ้นแค่ชั่วคราว ให้ผลลัพธ์ที่รวดเร็ว เนื่องจากความรวดเร็วในการให้ข้อมูลสินค้า จะส่งผลต่อการเลือกซื้อสินค้าของลูกค้า ซึ่งลักษณะดังกล่าวสอดคล้องกับแบบรูป Retry ตามที่เครื่องมือแนะนำ โดยผู้พัฒนาเห็นว่าเซอร์วิซที่พัฒนามีลักษณะที่หลากหลาย ยากแก่การเลือกและจัดลำดับความสำคัญ

5.2.4. โดเมนโลจิสติกส์

โดเมนนี้มีผู้พัฒนาเซอร์วิซทั้งหมด 2 ราย ผู้พัฒนารายแรกมีการใช้แบบรูป Wait ซึ่งเครื่องมือแนะนำแบบรูป Retry ให้เป็นอันดับ 1 ส่วนแบบรูป Wait แนะนำให้เป็นอันดับ 3 ผู้พัฒนาจัดอันดับความเด่นของลักษณะเรียงจากเด่นมากไปน้อย คือ Simplicity, Transient Failure,

Economy, Timeliness และ Correctness เนื่องจากแบบรูป Retry และ Wait พัฒนาได้ง่ายและให้ผลลัพธ์ได้ถูกต้อง และมีค่าใช้จ่ายในการประมวลผลใกล้เคียงกัน เพราะเป็นการเรียกเซอร์วิซเดียวเหมือนกัน แต่แบบรูป Retry จัดการกับความผิดพร่องที่เกิดขึ้นชั่วคราวและให้ความรวดเร็วในการตอบกลับได้ดีกว่าแบบรูป Wait ทำให้เครื่องมือแนะนำแบบรูป Retry ให้กับผู้พัฒนาเซอร์วิซ ส่วนผู้พัฒนารายที่สองมีการใช้แบบรูป Retry+Wait ซึ่งเครื่องมือแนะนำแบบรูป Retry ให้เป็นอันดับ 1 ส่วนแบบรูป Retry+Wait เครื่องมือแนะนำให้เป็นอันดับ 2 ผู้พัฒนาจัดอันดับความเด่นของลักษณะเรียงจากเด่นมากไปน้อย คือ Instance Specificity, Transient Failure, Timeliness และ Simplicity เนื่องจากแบบรูป Retry เหมาะกับเซอร์วิซที่มีความจำเพาะและพัฒนาง่าย เพราะเป็นการเรียกเซอร์วิซเดียว และเหมาะกับความผิดพร่องที่เกิดขึ้นชั่วคราว คือ สามารถเรียกเซอร์วิซซ้ำได้ เมื่อเกิดข้อผิดพร่อง และใช้เวลาในการตอบกลับน้อยกว่าแบบรูป Wait เนื่องจาก Wait จะต้องรอก่อนการเรียกเซอร์วิซ ในขณะที่ Retry สามารถเรียกเซอร์วิซได้ทันที ดังนั้นเครื่องมือจึงแนะนำแบบรูป Retry ให้กับผู้พัฒนาเซอร์วิซ

จากข้อมูลการเลือกลักษณะของเซอร์วิซโดเมนโลจิสติกส์ของผู้พัฒนา สรุปได้ว่า ลักษณะที่ผู้พัฒนาเห็นว่าสำคัญเหมือนกันคือ Transient Failure, Timeliness และ Simplicity แสดงว่าผู้พัฒนาคำนึงถึงความรวดเร็วในการตอบกลับ ถ้าเกิดความผิดพร่องขึ้น จะสามารถใช้งานได้ในไม่ช้า และพัฒนาง่าย เพื่อไม่ให้เกิดความผิดพร่องที่เกิดจากการทำให้ทนต่อความผิดพร่อง จากลักษณะดังกล่าวสอดคล้องกับลักษณะของแบบรูป Retry มากที่สุด

5.2.5. โดเมนธนาคาร

โดเมนนี้มีผู้พัฒนาเซอร์วิซทั้งหมด 1 ราย ผู้พัฒนาจัดอันดับความเด่นของลักษณะเรียงจากเด่นมากไปน้อย คือ Instance Specificity, Transient Failure, Simplicity และ Timeliness ซึ่งแบบรูปที่เครื่องมือแนะนำตรงกับแบบรูปที่ผู้พัฒนาใช้คือ Retry เนื่องจากเป็นแบบรูปที่เหมาะกับความผิดพร่องที่เกิดขึ้นชั่วคราว และเซอร์วิซที่มีความจำเพาะ ง่ายต่อการพัฒนา ใช้เวลาในการประมวลผลไม่มาก

จากข้อมูลการเลือกลักษณะของเซอร์วิซโดเมนธนาคารของผู้พัฒนา สรุปได้ว่า ผู้พัฒนาให้ความสำคัญกับความจำเพาะของเซอร์วิซ เนื่องจากเซอร์วิซของธนาคารจะมีข้อมูลเฉพาะของธนาคารตัวเองเท่านั้น และในการให้บริการ ถ้ามีความผิดพร่อง จะต้องเกิดขึ้นชั่วคราวเท่านั้น เนื่องจากลูกค้ามีความจำเป็นที่จะต้องใช้เซอร์วิซในการทำธุรกรรมต่างๆ และเป็นเซอร์วิซที่พัฒนาง่าย เพื่อไม่ให้เกิดความผิดพร่องขึ้น การให้ผลลัพธ์จะต้องรวดเร็ว เพื่อไม่ให้ลูกค้ารอนาน ซึ่งลักษณะดังกล่าวเหมาะกับแบบรูป Retry มากที่สุด

บทที่ 6
สรุปผลการวิจัยและข้อเสนอแนะ

6.1. สรุปผลการวิจัย

งานวิจัยนี้เสนอการพัฒนาเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องด้วยโครงสร้างของบีเพล โดยฟังก์ชันงานหลักของเครื่องมือจะประกอบด้วย (1) การแนะนำผู้พัฒนาเว็บเซอร์วิซเกี่ยวกับการทนต่อความผิดพร่องซึ่งเหมาะสมกับเว็บเซอร์วิซทางฝั่งผู้ให้บริการโดยใช้แบบจำลองการแนะนำแบบรูปการทนต่อความผิดพร่อง ซึ่งมีการพิจารณาลักษณะของตัวเว็บเซอร์วิซและลักษณะการให้บริการเว็บเซอร์วิซนั้น เพื่อกำหนดลำดับความเด่นและนำมาคำนวณกับเมตริกซ์ความสัมพันธ์ระหว่างลักษณะของเซอร์วิซกับแบบรูปการทนต่อความผิดพร่อง ซึ่งจะได้ค่าผลลัพธ์ที่นำมาใช้ในการแนะนำแบบรูป (2) การสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องตามแบบรูปที่นักพัฒนาเว็บเซอร์วิซเลือกจากข้อ (1) โดยใช้ภาษาบีเพล เครื่องมือนี้จะช่วยสนับสนุนการพัฒนาเว็บเซอร์วิซที่ทนต่อความผิดพร่องได้อย่างเหมาะสม ผ่านการประเมินลักษณะต่างๆ ของเว็บเซอร์วิซโดยผู้พัฒนาเอง เครื่องมือจะสร้างโครงสร้างของบีเพลโดยใช้มาตรฐานดับเบิลยูเอส-บีเพล 2.0 และใช้โปรแกรมโอเพนซอร์ซ คือ GlassFish ESB v2.2 ประกอบการสร้างและประมวลผลบีเพล โดยเว็บเซอร์วิซที่พิจารณาจะเป็นเซอร์วิซเดี่ยว ไม่ใช่เซอร์วิซประกอบ

ลักษณะของเซอร์วิซที่ใช้ในการพิจารณา เพื่อเลือกแบบรูปการทนต่อความผิดพร่องที่เหมาะสม มีทั้งหมด 8 ลักษณะ ได้แก่

1. Transient Failure
2. Instance Specificity
3. Replica Provision
4. NVP Provision
5. Correctness
6. Timeliness
7. Simplicity
8. Economy

แบบรูปการทนต่อความผิดพร่องที่แนะนำให้ผู้พัฒนาเซอร์วิซมีทั้งหมด 9 แบบรูป ได้แก่

1. Retry

2. Wait
3. RecoveryBlock $_{Replica}$
4. RecoveryBlock $_{NVP}$
5. Active $_{Replica}$
6. Active $_{NVP}$
7. Voting $_{Replica}$
8. Voting $_{NVP}$
9. Retry + Wait

หลังจากที่สร้างเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องด้วยโครงสร้างของบีเพลเสร็จแล้ว ผู้วิจัยได้ให้ผู้พัฒนาเซอร์วิซจำนวน 10 คนมาประเมินเครื่องมือ ซึ่งจากผลการประเมิน สรุปได้ว่า ผู้พัฒนาเซอร์วิซส่วนใหญ่มีการคำนึงถึงการทำให้เซอร์วิซทนต่อความผิดพร่อง โดยผู้พัฒนาเซอร์วิซพัฒนาวิธีการที่ทำให้เซอร์วิซทนต่อความผิดพร่องเอง มีการใช้ฮาร์ดแวร์และเทคโนโลยีอื่นมาเสริม แบบรูปที่ใช้กันส่วนใหญ่ คือ Retry และ Retry+Wait ในการแนะนำแบบรูปของเครื่องมือส่วนใหญ่แนะนำได้ตรงกับแบบรูปที่ผู้พัฒนาใช้ มีเพียงส่วนน้อยที่เครื่องมือแนะนำได้ไม่ตรง แต่อันดับที่เครื่องมือแนะนำให้คือ อันดับ 2 และ 3 ซึ่งถือว่ายังแนะนำให้เป็นอันดับต้นๆ และผู้พัฒนาส่วนใหญ่เห็นว่า วิธีการแนะนำแบบรูปการทนต่อความผิดพร่องโดยพิจารณาจากลักษณะของเซอร์วิซมีความเหมาะสม และมีผู้พัฒนาแนะนำว่า ในเซอร์วิซหนึ่งมีส่วนการทำงานที่หลากหลาย อาจจะมีวิธีที่ดีที่สุดในการทนต่อความผิดพร่องที่ไม่เหมือนกัน ดังนั้นการเลือกลักษณะและกำหนดลำดับความเด่นจึงทำได้ยาก และผู้พัฒนาส่วนใหญ่เห็นว่า เมตริกซ์แสดงความสัมพันธ์ระหว่างลักษณะของเซอร์วิซกับแบบรูปการทนต่อความผิดพร่องมีการให้คะแนนและมีวิธีการคำนวณหาผลคูณของเมตริกซ์ได้อย่างสมเหตุสมผล เพราะในเมตริกซ์ได้สะท้อนความสัมพันธ์ของแบบรูปตามลักษณะของเซอร์วิซ และการคำนวณจะคำนวณจากการให้น้ำหนักความสำคัญของลักษณะของเซอร์วิซ ในกรณีที่เซอร์วิซไม่มีคุณลักษณะด้านใด ลักษณะนั้นจะไม่ถูกนำมาพิจารณา

6.2. ข้อจำกัดและข้อเสนอแนะ

6.2.1. โครงสร้างของบีเพลที่เครื่องมือสร้างเพื่อจัดการกับการทนต่อความผิดพร่องตามแบบรูป โดยใช้มาตรฐานบีเพล 2.0 ยังไม่สามารถทำงานได้ทันที ต้องอาศัยการใช้โปรแกรม NetBeans และ GlassFish ESB v2.2 ประกอบการสร้าง เพื่อให้มีการสร้างไฟล์ห่อหุ้ม (Wrapper)

ให้กับไฟล์วิสเดิลที่นำเข้ามา รวมทั้งไฟล์อื่นซึ่งจำเป็นต่อการประมวลผลโครงสร้างบีเพลด้วยเครื่องประมวลผลบีเพลของ GlassFish ESB v2.2

6.2.2. เนื่องจากงานวิจัยนี้สนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่อง ซึ่งเป็นเซอร์วิซเดี่ยวและมี 1 โอเปอเรชัน ดังนั้นงานวิจัยต่อไปอาจมีการพัฒนาเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่อง โดยแนะนำแบบรูปการทนต่อความผิดพร่องผ่านการประเมินลักษณะของตัวเว็บเซอร์วิซให้กับเซอร์วิซเดี่ยวที่มีหลายโอเปอเรชัน และเซอร์วิซที่เป็นเซอร์วิซประกอบ

6.2.3. ในการพิจารณาลักษณะของเซอร์วิซเพื่อใช้แนะนำแบบรูปการทนต่อความผิดพร่องอาจเพิ่มลักษณะและแบบรูปอื่นเพิ่มเติมให้มากขึ้น เช่น แบบรูปการข้ามหรือเพิกเฉยต่อความผิดพร่อง เป็นต้น

6.2.4. ในการแนะนำแบบรูป อาจแนะนำได้มากกว่า 1 แบบรูป เนื่องจากบางแบบรูปมีค่าคะแนนที่ใกล้เคียงกัน โดยกำหนดช่วงของคะแนนที่ยอมรับได้ เพื่อบอกว่าแบบรูปใดที่ค่าคะแนนอยู่ในช่วงดังกล่าวมีความเหมาะสม

รายการอ้างอิง

[1] Liu, A., Li, Q., Huang, L., and Xiao, M. FACTS: A Framework for Fault –Tolerant Composition of Transactional Web Services. IEEE Transactions on Services Computing 3(January-March 2010) : 46-59.

[2] Thaisongsuwan, T. and Senivongse, T. Applying Software Fault Tolerance Patterns to WS-BPEL Processes. In Proceedings of the 8th International Joint Conference on Computer Science and Software Engineering (JCSSE), 269-274. Nakorn Pathom, Thailand, May 11-13, 2011.

[3] Liu, A., Li, Q., Huang, L., and Xiao M. A Declarative Approach to Enhancing the Reliability of BPEL Processes. In Proceedings of IEEE International Conference on Web Services (ICWS), 2007.

[4] Dobson, G. Using WS-BPEL to Implement Software Fault Tolerance for Web Services. In Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO-SEAA'06), 2006.

[5] Modafferi, S. and Conforti, E. Methods for Enabling Recovery Actions in Ws-BPEL. In On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE, 219-236. 2006.

[6] Modafferi, S., Mussi, E., and Pernici, B. SH-BPEL-A Self-Healing plug-in for Ws-BPEL engines. In Proceedings of the Middleware for Service Oriented Computing Workshop (MW4SOC), Melbourne, Australia, 2006.

[7] Laranjeiro, N. and Vieira, M. Towards Fault Tolerance in Web Services Compositions. In Proceedings of the 2007 workshop on Engineering fault tolerant systems, Croatia, 2007.

[8] Ezenwoye, O. and Sadjadi, S.M. Enabling Robustness in Existing BPEL Processes. In Proceedings of the 8th International Conference on Enterprise Information Systems (ICEIS), 2006.

[9] Zheng, Z. and Lyu, M.R. A Distributed Replication Strategy Evaluation and Selection Framework for Fault Tolerant Web Services. In Proceedings of IEEE International Conference on Web Services (ICWS), 2008.

[10] Lau, J., Lung, L.C., Fraga, J.d.S., and Santos, G. Designing Fault Tolerant Web Services Using BPEL. In Proceedings of the 7th IEEE/ACIS International Conference on Computer and Information Science (icis), 2008.

[11] Zheng, Z. and Lyu, M.R. An adaptive QOS-aware fault tolerance strategy for web services. Empirical Softw. Eng.15 (2010) : 323-345.

[12] Hanmer, R. Patterns for Fault Tolerant Software. Chichester: Willey Publishing, 2007.

[13] Avizienis, A., Laprie, J.C., Randell, B., and Landwehr, C. Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transactions on Dependable and Secure Computing 1(January-March 2004) : 11-33.

[14] Zheng, Z. and Lyu, M.R. A QOS-Aware Fault Tolerant Middleware for Dependable Service Composition. In Proceedings of IEEE International Conference on Dependable Systems & Networks (DSN '09), 2009.

[15] Zheng, Z. and Lyu, M.R. Optimal Fault Tolerance Strategy Selection for Web Services. International Journal of Web Services Research 7 (October-December 2010) : 21-40.

[16] Shim, B., Choue, S., Kim, S., and Park, S. A Design Quality Model for Service-Oriented Architecture. In Proceedings of the 15th Asia-Pacific Software Engineering Conference, 2008.

[17] Yu, W.D., Radhakrishna, R.B., Pingali, S., and Kolluri, V. Modeling the Measurements of QoS Requirements in Web Service Systems. SIMULATION 83 (January 2007) : 75-91.

[18] Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., and Chang, H. QoS-Aware Middleware for Web Services Composition. IEEE Transactions on Software Engineering 30 (2004)

[19] Oracle. Glassfish ESB [Online]. Available from : http://glassfish.java.net/public/downloadsindex.html#top [January, 2010]

ภาคผนวก

ภาคผนวก ก
โค้ดโครงสร้างบีเพลของแบบรูป RB_{NVP}

โครงสร้างบีเพลของแบบรูป RB_{NVP}

ถ้าผู้ใช้กรอก Service WSDL File from URL เป็น http://www.webservicex.com/CurrencyConvertor.asmx?wsdl และ Operation เป็น ConversionRate ที่เรียกใช้ในเซอร์วิซหลัก Recovery WSDL File from URL เป็นhttp://www.xignite.com/xCurrencies.asmx?wsdl และ Operation เป็น ConversionRealTimeValue ที่เรียกใช้ในเซอร์วิซตัวแทน จะได้โค้ดโครงสร้างบีเพลดังตารางที่ ก-1

ตารางที่ ก-1 โค้ดโครงสร้างบีเพลของแบบรูป RB_{NVP}

```
<?xml version="1.0" encoding="UTF-8"?>
<process
    name="WSBPEL_Currency_RB"
targetNamespace="http://enterprise.netbeans.org/bpel/WSBPEL_Currency_RB/WSBP
EL_Currency_RB"
xmlns:tns=http://enterprise.netbeans.org/bpel/WSBPEL_Currency_RB/WSBPEL_Curre
ncy_RB
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
xmlns:sxt="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/Trace"
xmlns:sxed="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/Editor"
xmlns:sxeh="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/Error
Handling"
xmlns:sxed2="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/Edit
or2" xmlns:ns0="http://www.webserviceX.NET/"
xmlns:ns1="http://www.xignite.com/services/">
   <import
namespace="http://j2ee.netbeans.org/wsdl/WSBPEL_Currency_RB/ExternalWSProces
sWSDL" location="ExternalWSProcessWSDL.wsdl"
```

ตารางที่ ก-1 โค้ดโครงสร้างบีเพลของแบบรูป RB_{NVP} (ต่อ)

```
importType="http://schemas.xmlsoap.org/wsdl/"/>
 <import
namespace="http://enterprise.netbeans.org/bpel/CurrencyConvertor.asmxWrapper"
location="CurrencyConvertor.asmxWrapper.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"/>
  <import namespace="http://www.webserviceX.NET/"
location="http://www.webservicex.com/CurrencyConvertor.asmx?wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"/>
  <import
namespace="http://enterprise.netbeans.org/bpel/xCurrencies.asmxWrapper"
location="xCurrencies.asmxWrapper.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"/>
  <import namespace="http://www.xignite.com/services/"
location="http://www.xignite.com/xCurrencies.asmx?wsdl"
importType="http://schemas.xmlsoap.org/wsdl/"/>
  <partnerLinks>
    <partnerLink name="ExternalPartnerLinkWebserviceX"
xmlns:tns="http://enterprise.netbeans.org/bpel/CurrencyConvertor.asmxWrapper"
partnerLinkType="tns:CurrencyConvertorSoapLinkType"
partnerRole="CurrencyConvertorSoapRole"/>
    <partnerLink name="ExternalPartnerLink_Xignite"
xmlns:tns="http://enterprise.netbeans.org/bpel/xCurrencies.asmxWrapper"
partnerLinkType="tns:XigniteCurrenciesSoapLinkType"
partnerRole="XigniteCurrenciesSoapRole"/>
    <partnerLink name="ClientPartnerLink"
xmlns:tns="http://j2ee.netbeans.org/wsdl/WSBPEL_Currency_RB/ExternalWSProcess
WSDL" partnerLinkType="tns:ExternalWSProcessWSDL"
myRole="ExternalWSProcessWSDLPortTypeRole"/>
  </partnerLinks>
```

ตารางที่ ก-1 โค้ดโครงสร้างบีเพลของแบบรูป RB_{NVP} (ต่อ)

```
<variables>
 <variable name="outData"
xmlns:tns="http://j2ee.netbeans.org/wsdl/WSBPEL_Currency_RB/ExternalWSProcess
WSDL" messageType="tns:ExternalWSProcessWSDLOperationResponse"/>
    <variable name="inData"
xmlns:tns="http://j2ee.netbeans.org/wsdl/WSBPEL_Currency_RB/ExternalWSProcess
WSDL" messageType="tns:ExternalWSProcessWSDLOperationRequest"/>
  </variables>
  <sequence>
    <receive name="start" createInstance="yes" partnerLink="ClientPartnerLink"
operation="ExternalWSProcessWSDLOperation"
xmlns:tns="http://j2ee.netbeans.org/wsdl/WSBPEL_Currency_RB/ExternalWSProcess
WSDL" portType="tns:ExternalWSProcessWSDLPortType" variable="inData"/>
    <scope name="Scope1">
      <sequence name="Sequence1">
        <assign name="AssignPassFalse">
          <copy>
            <from>false()</from>
            <to variable="pass"/>
          </copy>
        </assign>
        <scope name="Scope2">
          <variables>
          <variable name="ConversionRateOut"
xmlns:tns="http://www.webserviceX.NET/"
messageType="tns:ConversionRateSoapOut"/>
            <variable name="ConversionRateIn"
xmlns:tns="http://www.webserviceX.NET/"
messageType="tns:ConversionRateSoapIn"/>
```

ตารางที่ ก-1 โค้ดโครงสร้างบีเพลของแบบรูป RB_{NVP} (ต่อ)

```
        </variables>
        <faultHandlers>
   <catchAll>
            <assign name="AssignPassFalse">
              <copy>
                <from>false()</from>
                <to variable="pass"/>
              </copy>
            </assign>
          </catchAll>
        </faultHandlers>
        <sequence name="Sequence2">
          <assign name="AssignfromtoCurrency">
            <copy>
              <from variable="inData" part="fromCurrency"/>
              <to>$ConversionRateIn.parameters/ns0:FromCurrency</to>
            </copy>
            <copy>
              <from variable="inData" part="toCurrency"/>
              <to>$ConversionRateIn.parameters/ns0:ToCurrency</to>
            </copy>
          </assign>
          <invoke name="Invoke1" partnerLink="ExternalPartnerLinkWebserviceX"
operation="ConversionRate" xmlns:tns="http://www.webserviceX.NET/"
portType="tns:CurrencyConvertorSoap" inputVariable="ConversionRateIn"
outputVariable="ConversionRateOut"/>
          <assign name="AssignPassTrue">
            <copy>
              <from>true()</from>
```

ตารางที่ ก-1 โค้ดโครงสร้างบีเพลของแบบรูป RB_{NVP} (ต่อ)

```
                <to variable="pass"/>
        </copy>
      </assign>
            <assign name="AssignResult">
              <copy>
                <from>concat($pass,
$ConversionRateOut.parameters/ns0:ConversionRateResult)</from>
                <to variable="outData" part="result"/>
              </copy>
            </assign>
          </sequence>
        </scope>
        <if name="IfPassTrue">
          <condition>$pass</condition>
          <empty name="Empty1"/>
          <else>
            <scope name="Scope3">
              <variables>
                <variable name="ConvertRealTimeValueOut"
xmlns:tns="http://www.xignite.com/services/"
messageType="tns:ConvertRealTimeValueSoapOut"/>
                <variable name="ConvertRealTimeValueIn"
xmlns:tns="http://www.xignite.com/services/"
messageType="tns:ConvertRealTimeValueSoapIn"/>
              </variables>
              <faultHandlers>
                <catchAll>
                  <rethrow name="Rethrow2"/>
                </catchAll>
```

ตารางที่ ก-1 โค้ดโครงสร้างบีเพลของแบบรูป RB_{NVP} (ต่อ)

```
    </faultHandlers>
<sequence name="Sequence3">
                <assign name="Assign1">
                  <copy>
                    <from variable="inData" part="fromCurrency"/>
                    <to>$ConvertRealTimeValueIn.parameters/ns1:From</to>
                  </copy>
                  <copy>
                    <from variable="inData" part="toCurrency"/>
                    <to>$ConvertRealTimeValueIn.parameters/ns1:To</to>
                  </copy>
                  <copy>
                    <from variable="inData" part="amount"/>
                    <to>$ConvertRealTimeValueIn.parameters/ns1:Amount</to>
                  </copy>
                </assign>
                <invoke name="Invoke2" partnerLink="ExternalPartnerLink_Xignite"
operation="ConvertRealTimeValue" portType="ns1:XigniteCurrenciesSoap"
inputVariable="ConvertRealTimeValueIn"
outputVariable="ConvertRealTimeValueOut"/>
                <assign name="AssignResult">
                  <copy>
                    <from>concat('RB  ',
$ConvertRealTimeValueOut.parameters/ns1:ConvertRealTimeValueResult/ns1:Result)
</from>
                    <to variable="outData" part="result"/>
                  </copy>
                </assign>
              </sequence>
```

ตารางที่ ก-1 โค้ดโครงสร้างบีเพลของแบบรูป RB_{NVP} (ต่อ)

```
    </scope>
</else>
          </if>
        </sequence>
      </scope>
      <reply name="end" partnerLink="ClientPartnerLink"
operation="ExternalWSProcessWSDLOperation"
xmlns:tns="http://j2ee.netbeans.org/wsdl/WSBPEL_Currency_RB/ExternalWSProcess
WSDL" portType="tns:ExternalWSProcessWSDLPortType" variable="outData"/>
   </sequence>
</process>
```

ภาคผนวก ข
แบบประเมินเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่อง ด้วยโครงสร้างของบีเพล

แบบประเมินเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องด้วยโครงสร้างของบีเพล

ซึ่งเป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย แบบประเมินนี้จัดทำขึ้นเพื่อให้ทราบถึงความเหมาะสมของเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่อง และวิธีการแนะนำแบบรูปของการทนต่อความผิดพร่อง

ชื่อวิทยานิพนธ์ภาษาไทย: การพัฒนาเครื่องมือสนับสนุนการสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องด้วยโครงสร้างของบีเพล

ชื่อวิทยานิพนธ์ภาษาอังกฤษ: A Development of a Supporting Tool for Constructing Fault Tolerant Web Services with BPEL Structure

ชื่อผู้วิจัย: นางสาวธันยธร ลีลาวัชรมาศ

อาจารย์ที่ปรึกษาวิทยานิพนธ์: รศ.ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา

อีเมล์: tunyathorn21@hotmail.com, Tunyathorn.L@student.chula.ac.th

เบอร์โทรติดต่อ: 086-709-0211

ที่มาและคำอธิบายงานวิจัย

การพัฒนาเว็บเซอร์วิซให้มีคุณภาพ ถือเป็นสิ่งสำคัญที่จะทำให้ผู้ใช้บริการนิยมเข้ามาใช้บริการ และแอพพลิเคชันที่เรียกใช้บริการเว็บเซอร์วิซสามารถทำงานได้อย่างมีประสิทธิภาพการทำให้เว็บเซอร์วิซทนต่อความผิดพร่อง (Fault Tolerant) เป็นสิ่งหนึ่งที่ทำให้การเรียกใช้เว็บเซอร์วิซสามารถดำเนินงานต่อไปได้ ถึงแม้ว่าจะเกิดข้อผิดพร่องต่างๆ ระหว่างที่มีการเรียกใช้เซอร์วิซ ทั้งข้อผิดพร่องที่เกิดจากเซอร์วิซเรียกใช้งานไม่ได้หรือผลลัพธ์ที่ได้จากเซอร์วิซไม่ถูกต้อง ในการทำให้เว็บเซอร์วิซทนต่อความผิดพร่องสามารถทำได้โดยประยุกต์ใช้แบบรูปการทนต่อความผิดพร่อง (Fault Tolerance Pattern) เข้ากับเว็บเซอร์วิซ แต่เนื่องจากแบบรูปมีหลายแบบ ดังนั้นคำถามคือผู้พัฒนาเว็บเซอร์วิซควรใช้แบบรูปใดกับเซอร์วิซที่พัฒนาอยู่

ในงานวิจัยนี้ ผู้วิจัยเสนอการพัฒนาเครื่องมือเพื่อช่วยสนับสนุนผู้พัฒนาเว็บเซอร์วิซ (ฝั่งผู้ให้บริการ) ในการพัฒนาเว็บเซอร์วิซให้สามารถทนต่อความผิดพร่องได้ โดยฟังก์ชันงานหลักของเครื่องมือจะประกอบด้วย

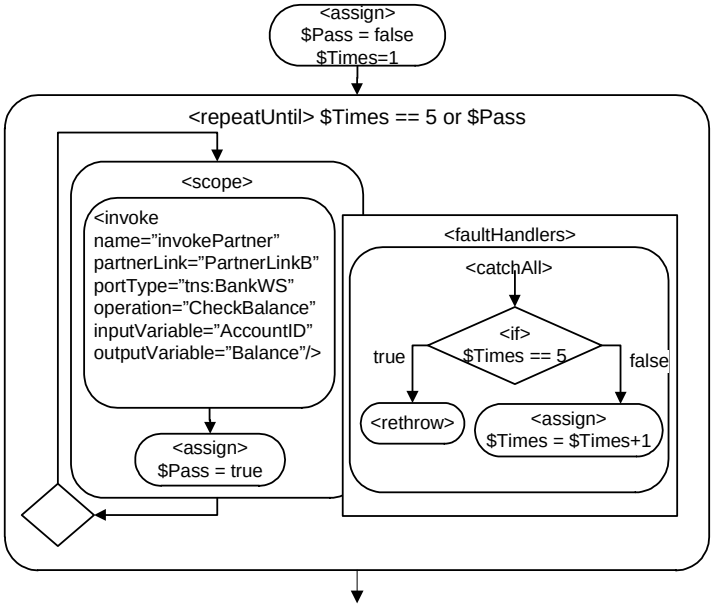
(1)การแนะนำผู้พัฒนาเว็บเซอร์วิซเกี่ยวกับแบบรูปการทนต่อความผิดพร่องซึ่งเหมาะสมกับเว็บเซอร์วิซ โดยใช้แบบจำลองการแนะนำแบบรูปการทนต่อความผิดพร่อง แบบรูปที่สามารถแนะนำประกอบด้วย Retry, Wait, RecoveryBlockReplica, RecoveryBlockNVP, ActiveReplica, ActiveNVP, VotingReplica, VotingNVP และ Retry+Wait ในการแนะนำแบบรูปจะพิจารณาจากลักษณะของตัวเว็บเซอร์วิซและลักษณะหรือสภาพการให้บริการเว็บเซอร์วิซนั้นในประเด็น Transient Failure, Instance Specificity, Replica Provision, NVP Provision, Correctness, Timeliness, Simplicity และ Economy

(2) การสร้างเว็บเซอร์วิซที่ทนต่อความผิดพร่องตามแบบรูปที่นักพัฒนาเว็บเซอร์วิซเลือกจากคำแนะนำที่ได้ โดยใช้ภาษาบีเพล (Business Process Execution Language) ซึ่งเป็นภาษาโครงสร้างกระแสงานของเว็บเซอร์วิซ ผลลัพธ์ที่ได้คือบีเพลเซอร์วิซที่ทำการเรียกใช้เว็บเซอร์วิซของผู้พัฒนาโดยมีโครงสร้างตามแบบรูปการทนต่อความผิดพร่องที่ผู้พัฒนาเลือก ตัวอย่างเช่นดังรูป เซอร์วิซ checkBalance ของ BankWS ถูกทำให้ทนต่อความผิดพร่องด้วยแบบรูป Retry โดยการสร้างบีเพลเซอร์วิซที่สามารถเรียก checkBalance นี้ซ้ำได้หากเมื่อเรียกไปแล้วไม่สำเร็จ โดยกำหนดให้สามารถเรียกซ้ำไม่เกิน 5 ครั้ง

Input: WSDL

$Times=5

Output:



กำหนดให้ค่าพารามิเตอร์ $Times คือ จำนวนรอบที่เซอร์วิซเรียกซ้ำ เมื่อมีข้อผิดพร่องเกิดขึ้น

คุณสมบัติของผู้ประเมิน: เป็นผู้พัฒนาเว็บเซอร์วิซ

ขั้นตอนการกรอกแบบประเมิน

1. กรอกข้อมูลของผู้ประเมิน
2. ศึกษาคำนิยามและแนวคิดของการแนะนำแบบรูปการทนต่อความผิดพร่องให้กับเว็บเซอร์วิซ
3. ศึกษาวิธีการใช้เครื่องมือ
4. ทดลองใช้เครื่องมือซึ่งจะแนะนำแบบรูปการทนต่อความผิดพร่องให้กับเว็บเซอร์วิซของท่าน และพิจารณาผลลัพธ์ที่ได้
5. ทำแบบประเมิน

(กรอกข้อมูลลงในช่องว่างและทำเครื่องหมายถูก / ลงในช่องสี่เหลี่ยม)

ส่วนข้อมูลผู้ประเมิน:

1 ตำแหน่ง ☐ Programmer ☐ Senior Programmer
☐ System Analyst ☐ Project Manager
☐ Software Tester ☐ Help Desk
☐ Researcher ☐ อื่นๆ (โปรดระบุ)_____

2 ประสบการณ์การทำงาน _____ ปี

3 ประสบการณ์การทำเว็บเซอร์วิซ _____ปี

4 ประเภทหน่วยงาน ☐ Software House ☐ Bank ☐ Commerce
☐ Financial and Investment ☐ Communication ☐ Education
☐ Government Organization ☐ อื่นๆ (โปรดระบุ) _____

5 โดเมนของเว็บเซอร์วิซ _____

6 ลักษณะการใช้งานเว็บเซอร์วิซ ☐ ภายในองค์กร ☐ ระหว่างองค์กร

7 ลักษณะการพัฒนาเว็บเซอร์วิซ ☐ พัฒนาเว็บเซอร์วิซเอง

☐ พัฒนาเองและใช้บริการเว็บเซอร์วิซอื่นด้วย

8 มาตรฐานโพรโทคอลที่ใช้พัฒนาเว็บเซอร์วิซ

☐ SOAP ☐ REST ☐ อื่นๆ (โปรดระบุ) _____

9 ในหน่วยงานมีการใช้บีเพล (BPEL) หรือไม่

☐ ใช้ ☐ ไม่ใช้

10 ในการพัฒนาเว็บเซอร์วิซมีการคำนึงถึงการทำให้เซอร์วิซทนต่อความผิดพร่องใช่หรือไม่

☐ ใช่ (ไปทำข้อ 11) ☐ ไม่ใช่

11 วิธีการที่ทำให้เซอร์วิซทนต่อความผิดพร่อง

☐ ผู้พัฒนาเซอร์วิซพัฒนาเอง ☐ ใช้โครงสร้างพื้นฐานที่มีอยู่ในองค์กร

☐อื่น (โปรดระบุ) _____

ส่วนประเมินเครื่องมือ

(กรุณากรอกอันดับความเด่นของลักษณะของเว็บเซอร์วิซของท่าน (ลักษณะที่ไม่ได้เลือกให้ทำเครื่องหมาย X กากบาท)

อันดับ ลักษณะของเว็บเซอร์วิซ

_____ Transient Failure

_____ Instance Specificity

_____ Replica Provision

_____ NVP Provision

_____ Correctness

_____ Timeliness

_____ Simplicity

_____ Economy

(กรอกข้อมูลลงในช่องว่างและทำเครื่องหมายถูก / ลงในช่องสี่เหลี่ยม)

1 ในการออกแบบเว็บเซอร์วิซ ท่านมีการใช้แบบรูปของการทนต่อความผิดพร่อง (Fault Tolerance Pattern) หรือไม่

☐ ไม่มี (ไปทำข้อ 5)

☐ มี แบบรูปที่ใช้คือ ☐Retry ☐Wait ☐RecoveryBlock $_{Replica}$

(สามารถเลือกได้มากกว่า 1 ข้อ) ☐RecoveryBlock $_{NVP}$ ☐Active $_{Replica}$

☐ Active NVP ☐Voting Replica ☐Voting NVP

☐Retry+Wait ☐อื่นๆ (โปรดระบุ)_____

2 เครื่องมือแนะนำแบบรูปการทนต่อความผิดพร่องได้ตรงกับแบบรูปที่พัฒนาหรือไม่

☐ใช่ (ไปทำข้อ 5) ☐ไม่ใช่

3 แบบรูปการทนต่อความผิดพร่องที่ท่านใช้ เครื่องมือแนะนำให้เป็นอันดับที่เท่าไร_____

4 ท่านคิดว่า เพราะเหตุใดแบบรูปที่เครื่องมือแนะนำจึงไม่ตรงกับแบบรูปที่ใช้

_____

_____

_____

5 จากลักษณะของเซอร์วิซที่กำหนดให้เลือก ท่านคิดว่าครอบคลุมลักษณะของเซอร์วิซที่พัฒนามากน้อยเพียงใด

☐มาก ☐ปานกลาง ☐น้อย ท่านคิดว่า ควรมีลักษณะของเซอร์วิซใดเพิ่มขึ้น_____

6 ท่านคิดว่า ลักษณะของเซอร์วิซใดสำคัญที่สุดในการพัฒนาเว็บเซอร์วิซ

☐Transient Failure ☐Instance Specificity ☐Replica Provision

☐NVP Provision ☐Correctness ☐Timeliness

☐Simplicity ☐Economy ☐อื่นๆ (โปรดระบุ)_____

เพราะเหตุใด

_____

_____

_____

7 ท่านคิดว่า แบบรูปการทนต่อความผิดพร่องในเครื่องมือมีความครอบคลุมมากน้อยเพียงใด

☐มาก ☐ปานกลาง ☐น้อย ท่านคิดว่า ควรมีแบบรูปใดเพิ่มขึ้น_____

8 ท่านคิดว่า แบบรูปการทนต่อความผิดพร่องใดสำคัญที่สุด

☐Retry ☐Wait ☐RecoveryBlock $_{Replica}$

☐RecoveryBlock $_{NVP}$ ☐Active $_{Replica}$ ☐ Active NVP

☐Voting Replica ☐Voting NVP ☐Retry+Wait

☐อื่นๆ (โปรดระบุ)_____

เพราะเหตุใด

_____

_____

_____

9 คำนิยามลักษณะของเซอร์วิซมีความชัดเจนมากน้อยเพียงใด

☐มาก ☐ปานกลาง ☐น้อย

10 คำนิยามแบบรูปการทนต่อความผิดพร่องมีความชัดเจนมากน้อยเพียงใด

☐มาก ☐ปานกลาง ☐น้อย

11 ท่านคิดว่า วิธีการแนะนำแบบรูปการทนต่อความผิดพร่องโดยพิจารณาจากลักษณะของเซอร์วิซเหมาะสมหรือไม่

☐เหมาะสม ☐ไม่เหมาะสม

เพราะเหตุใด

_____

_____

_____

12 จากเมตริกซ์แสดงความสัมพันธ์ระหว่างลักษณะของเซอร์วิซกับแบบรูปการทนต่อความผิดพร่อง ท่านคิดว่าการให้คะแนนในเมตริกซ์มีความสมเหตุสมผลใช่หรือไม่

☐ใช่ ☐ไม่ใช่

เพราะเหตุใด

_____

_____

_____

13 ท่านคิดว่า วิธีการหาแบบรูปการทนต่อความผิดพร่องที่เหมาะสมจากการคำนวณหาผลคูณของเมตริกซ์ P=D x R เป็นวิธีที่เหมาะสมใช่หรือไม่

☐ใช่ ☐ไม่ใช่

เพราะเหตุใด

_____

_____

_____

14 ท่านคิดว่า เครื่องมือแนะนำแบบรูปการทนต่อความผิดพร่องได้สมเหตุสมผลกับลักษณะของเซอร์วิซที่พัฒนาหรือไม่

☐ใช่ ☐ไม่ใช่

เพราะเหตุใด

_____

_____

_____

15 ข้อเสนอแนะ

_____

_____

_____

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวธันยธร ลีลาวัชรมาศ เกิดวันที่ 21 ตุลาคม พ.ศ. 2528 ที่จังหวัดกรุงเทพฯ สำเร็จการศึกษาระดับมัธยมศึกษาตอนต้นจากโรงเรียนอุดมศึกษา เมื่อปีการศึกษา 2543 สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลายจากโรงเรียนนวมินทราชินูทิศ เตรียมอุดมศึกษาน้อมเกล้า เมื่อปีการศึกษา 2546 สำเร็จการศึกษาระดับปริญญาตรี ในหลักสูตรวิทยาศาสตรบัณฑิต (วท.บ.) สาขาวิทยาการคอมพิวเตอร์ ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ เมื่อปีการศึกษา 2550 และเข้าศึกษาต่อในระดับปริญญาโท หลักสูตรวิทยาศาสตรมหาบัณฑิต (วท.ม.) สาขาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2552 ที่อยู่ที่สามารถติดต่อได้ คือ 27/600 (130) ถนนลาดพร้าว ซอย 101/1 เขตบางกะปิ แขวงคลองจั่น จังหวัดกรุงเทพฯ 10240 อีเมล์ tunyathorn.l@student.chula.ac.th