การตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะของยูเอ็มแอลโดยใช้ไพแคลคูลัส

นายวราวุฒิ ผ้าเจริญ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2547
ISBN 974-17-7154-1

A CONSISTENCY CHECKING BETWEEN UML COLLABORATION DIAGRAMS AND STATECHART DIAGRAMS USING π-CALCULUS

Mr. Warawoot Pacharoen

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Computer Science

Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University

Academic Year 2004
ISBN 974-17-7154-1

หัวข้อวิทยานิพนธ์	การตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะของยูเอ็มแอลโดยใช้ไพแคลคูลัส
โดย	นาย วราวุฒิ ผ้าเจริญ
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษา	อาจารย์ ดร. อรรถสิทธิ์ สุรฤกษ์
อาจารย์ที่ปรึกษาร่วม	ผู้ช่วยศาสตราจารย์ ดร. ภัทรสินี ภัทรโกศล

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

…….. คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร. ดิเรก ลาวัณย์ศิริ)

คณะกรรมการสอบวิทยานิพนธ์

…… ประธานกรรมการ
(รองศาสตราจารย์ ดร. วันชัย ริ้วไพบูลย์)

…….. อาจารย์ที่ปรึกษา
(อาจารย์ ดร. อรรถสิทธิ์ สุรฤกษ์)

……อาจารย์ที่ปรึกษาร่วม
(ผู้ช่วยศาสตราจารย์ ดร. ภัทรสินี ภัทรโกศล)

……กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. อานนท์ รุ่งสว่าง)

วราวุฒิ ผ้าเจริญ : การตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะของยูเอ็มแอลโดยใช้ไพแคลคูลัส. (A CONSISTENCY CHECKING BETWEEN UML COLLABORATION DIAGRAMS AND STATECHART DIAGRAMS USING π-CALCULUS) อ. ที่ปรึกษา : อาจารย์ ดร.อรรถสิทธิ์ สุรฤกษ์, อ. ที่ปรึกษาร่วม : ผู้ช่วยศาสตราจารย์ ดร.ภัทรสินี ภัทรโกศล, 154 หน้า. ISBN 974-17-7154-1.

วิทยานิพนธ์นี้ นำเสนอวิธีการตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะของยูเอ็มแอลซึ่งเป็นเครื่องมือที่นิยมใช้ในการออกแบบทางด้านวิศวกรรมซอฟต์แวร์ การตรวจสอบทำได้โดยการแปลงแผนภาพทั้งสองไปเป็นไพแคลคูลัส โดยวิทยานิพนธ์นี้ได้เสนอกฎการแปลงแผนภาพความร่วมมือเป็นไพแคลคูลัสจำนวน 8 ข้อ และกฎการแปลงแผนภาพสถานะเป็นไพแคลคูลัสจำนวน 9 ข้อ หลังจากที่ได้ไพแคลคูลัสจากการแปลงแผนภาพทั้งสอง การตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะสามารถทำได้โดยการตรวจสอบว่าเกิดความขัดแย้งกันของเหตุการณ์ที่สามารถเกิดขึ้นได้จากแผนภาพความร่วมมือ เทียบกับเหตุการณ์ที่เป็นไปได้ทั้งหมดของวัตถุในระบบที่ทำงานไปพร้อม ๆ กันซึ่งได้ถูกบรรยายไว้ในแผนภาพสถานะหรือไม่

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต.....

สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา.....

ปีการศึกษา 2547 ลายมือชื่ออาจารย์ที่ปรึกษาร่วม.....

4670476221 : MAJOR COMPUTER SCIENCE
KEY WORD: CONSISTENCY CHECKING / UML / COLLABORATION DIAGRAM / STATECHART DIAGRAM / π-CALCULUS

WARAWOOT PACHAROEN : A CONSISTENCY CHECKING BETWEEN UML COLLABORATION DIAGRAMS AND STATECHART DIAGRAMS USING π-CALCULUS. THESIS ADVISOR : ATHASIT SURARERKS, Ph.D., THESIS COADVISOR : ASST. PROF. PATTARASINEE BHATTARAKOSOL, Ph.D., 154 pp. ISBN 974-17-7154-1.

The thesis proposes a method for consistency checking between UML Collaboration diagrams and Statechart diagrams where UML is a stat-of-the-art tool for designing in software engineering. A verification is started from transforming both diagrams into π–Calculus. This thesis proposes 8 rules for transformation collaboration diagrams into π–Calculus and 9 rules for statechart diagram. As a result, a consistency checking is done by applying the rules to both diagrams and checking a consistence between events from collaboration diagrams and all events from statechart diagrams of all objects in the concurrent system.

Department Computer Engineering	Student's signature.....
Field of study Computer Science	Advisor's signature.....
Academic year 2004	Co-advisor's signature……..

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้ สำเร็จลุล่วงไปได้ด้วยความช่วยเหลืออย่างดียิ่งของ อาจารย์ ดร.อรรถสิทธิ์ สุรฤกษ์ อาจารย์ที่ปรึกษา และผู้ช่วยศาสตราจารย์ ดร.ภัทรสินี ภัทรโกศล อาจารย์ที่ปรึกษาร่วม ที่ให้คำแนะนำ เสนอแนะข้อคิดเห็นต่าง ๆ และให้แนวทางในการค้นคว้าวิจัยด้วยดีเสมอมา และขอขอบคุณ รองศาสตราจารย์ ดร.วันชัย ริ้วไพบูลย์ และผู้ช่วยศาสตราจารย์ ดร. อานนท์ รุ่งสว่าง กรรมการวิทยานิพนธ์ที่กรุณาเสียสละเวลาให้คำแนะนำ ตรวจ และแก้ไขต้นฉบับวิทยานิพนธ์

ขอบคุณ โก้ ต่อ วรรณ กุ้ง ไม้ เมศ สิทธิ์ พี่ชาย พี่ช้าง และเพื่อน ๆ ทุกคนที่เข้าใจให้กำลังใจ และให้ข้อเสนอแนะต่าง ๆ และขอขอบคุณท่านอื่น ๆ ที่มีส่วนช่วยในการทำวิทยานิพนธ์ที่ไม่ได้กล่าวนามมา ณ โอกาสนี้ด้วย

สุดท้ายนี้ ผู้วิจัยใคร่ขอกราบขอบพระคุณ บิดา มารดา พี่อ้อม และน้องลิงค์ที่ให้กำลังใจแก่ผู้วิจัยเสมอมา

สารบัญ

หน้า

บทที่ หน้า

สารบัญตาราง

ตาราง หน้า

สารบัญภาพ

รูปที่ หน้า

รูปที่ หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

คำกล่าวที่ว่า "กันไว้ดีกว่าแก้" นั้นใช้ได้ในทุกสถานการณ์ไม่เว้นแม้กระทั่งงานทางด้านวิศวกรรมซอฟต์แวร์ (software engineering) ดังจะพบได้จากการตรวจหาและแก้ไขข้อผิดพลาด (error) ที่เกิดขึ้นในระบบคอมพิวเตอร์ที่มีการทำงานแบบพร้อมกัน (concurrent system) เช่น ระบบควบคุมการจราจรทางอากาศ หรือระบบโทรศัพท์เคลื่อนที่ เป็นต้น นั้นทำได้ยาก เนื่องจากข้อผิดพลาดต่าง ๆ มักจะเกิดเป็นช่วง ๆ ในระหว่างที่ระบบกำลังทำงาน ไม่เกิดขึ้นอย่างสม่ำเสมอ เกิดปัญหาขึ้นทีก็แก้ไขที ไม่สามารถมั่นใจได้ว่าข้อผิดพลาดทั้งหมดได้รับการแก้ไขแล้ว ดังนั้นวิธีการที่ดีที่สุดสำหรับแก้ไขปัญหานี้ คือ เราจะต้องป้องกันไม่ให้เกิดข้อผิดพลาดขึ้นหรือให้เกิดขึ้นน้อยที่สุด โดยจะต้องตรวจสอบ (verification) ระบบ ตั้งแต่ในช่วงการออกแบบ (designing phase)

แผนภาพความร่วมมือ (collaboration diagram) และแผนภาพสถานะ (statechart diagram) เป็นส่วนหนึ่งของ แผนภาพยูเอ็มแอล (UML diagram) [1, 2] ซึ่งใช้ในการวิเคราะห์และออกแบบระบบ โดยแผนภาพทั้งสองนี้จะให้มุมมองเชิงพลวัติ (dynamic view) ของระบบในด้านที่แตกต่างกัน แผนภาพความร่วมมือจะบรรยายถึงปฏิสัมพันธ์ของวัตถุต่าง ๆ ที่มีอยู่ในระบบ ส่วนแผนภาพสถานะจะบรรยายถึงการเปลี่ยนแปลงพฤติกรรมของวัตถุเมื่อมีเหตุการณ์ต่าง ๆ เกิดขึ้น ซึ่งจากหนังสือของโกมา (Gomaa) [3] ได้บรรยายถึงวิธีการนำแผนภาพยูเอ็มแอลไปใช้ออกแบบระบบที่มีการทำงานแบบพร้อมกัน แต่อย่างไรก็ตามระบบที่ออกแบบด้วยแผนภาพยูเอ็มแอลก็ยังยากต่อการนำไปตรวจสอบความถูกต้อง

ไพแคลคูลัส (π-Calculus) [4, 5] เป็นภาษารูปนัย (formal language) ที่ใช้บรรยายพฤติกรรมของระบบที่มีการทำงานแบบพร้อมกันซึ่งมิลเนอร์ (Milner) เป็นผู้เสนอ โดยไพแคลคูลัสมีพื้นฐานมาจากซีซีเอส (CCS : Calculus of Communicating System) [6] แต่ได้เพิ่มเติมลักษณะพิเศษ คือ มีโครงสร้างการเชื่อมต่อระหว่างวัตถุเป็นแบบพลวัติ เนื่องจากไพแคลคูลัสเป็นภาษารูปนัยทำให้สามารถตรวจสอบความถูกต้องได้ง่ายอีกทั้งมีเครื่องมือต่าง ๆ ที่ช่วยในการตรวจสอบความถูกต้อง เช่น เอ็มดับเบิ้ลยูบี (MWB) [7] หรือ นูเอสเอ็มวี (NuSMV) [8] เป็นต้น แต่ความเป็นรูปนัยของไพแคลคูลัสก็ทำให้เกิดความยุ่งยากต่อการทำความเข้าใจและการ

นำไปใช้ ดังนั้นจึงมีงานวิจัยที่นำไพแคลคูลัสมาประยุกต์ใช้กับบางแผนภาพของยูเอ็มแอลเพื่อให้ง่ายต่อการนำไปใช้ เช่น การนำไพแคลคูลัสมาประยุกต์ใช้กับแผนภาพสถานะ [9-11]

จากที่ได้กล่าวมาข้างต้นจะพบว่าการบรรยายพฤติกรรมของระบบด้วยยูเอ็มแอลจำเป็นที่จะต้องใช้แผนภาพต่าง ๆ มาบรรยายร่วมกัน งานวิจัยนี้จึงมีแนวความคิดที่จะเสนอวิธีการแปลงแผนภาพความร่วมมือและแผนภาพสถานะซึ่งใช้บรรยายพฤติกรรมของระบบที่ทำงานแบบพร้อมกันไปเป็นไพแคลคูลัส เพื่อให้สามารถตรวจสอบความต้องกันระหว่างแผนภาพทั้งสองได้สะดวกและครอบคลุมมากขึ้น

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อเสนอกฎในการแปลงแผนภาพความร่วมมือ และแผนภาพสถานะไปเป็นไพแคลคูลัส

1.2.2 เพื่อนำเสนอวิธีการตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะ

1.3 ขอบเขตของการวิจัย

1.3.1 เสนอกฎในการแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัส

1.3.2 เสนอกฎในการแปลงแผนภาพสถานะไปเป็นไพแคลคูลัส

1.3.3 เสนอวิธีการตรวจสอบความต้องกันของกฎที่ได้จากการแปลงข้อ 1.3.1 และกฎที่ได้จากการแปลงข้อ 1.3.2

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 ได้กฎในการแปลงแผนภาพความร่วมมือ และแผนภาพสถานะไปเป็นไพแคลคูลัส

1.4.2 ได้วิธีการตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะ

1.5 วิธีดำเนินการวิจัย

1.5.1 ศึกษาและทำความเข้าใจในการออกแบบซอฟต์แวร์ ด้วยแผนภาพความร่วมมือ และแผนภาพสถานะ รวมถึงศึกษาการออกแบบระบบที่ทำงานแบบพร้อมกันด้วยยูเอ็มแอล

1.5.2 ศึกษางานวิจัยที่เกี่ยวข้องกับการแปลงแผนภาพต่าง ๆ ของยูเอ็มแอลไปเป็นภาษารูปนัย

1.5.3 ศึกษาภาษารูปนัยที่เหมาะสมเพื่อนำมาใช้เป็นภาษาเป้าหมายของการแปลงแผนภาพความร่วมมือ และแผนภาพสถานะ

1.5.4 ออกแบบขั้นตอนและกฎการแปลงจากแผนภาพความร่วมมือ และแผนภาพสถานะไปเป็นภาษาที่กำหนด

1.5.5 ออกแบบวิธีการในการตรวจสอบความสัมพันธ์ของแผนภาพจากภาษารูปนัยที่ได้จากขั้นตอนการแปลง

1.5.6 สรุปผลการวิจัยและจัดทำรายงานวิทยานิพนธ์

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 งานวิจัยที่เกี่ยวข้อง

การตรวจสอบความถูกต้องของระบบที่ออกแบบด้วยยูเอ็มแอลนั้น ทำได้ยาก เนื่องจากระบบที่ได้จะอยู่ในรูปแบบของแผนภาพต่าง ๆ ที่ใช้บรรยายระบบร่วมกัน ดังนั้นจึงมีงานวิจัยจำนวนมากพยายามแปลงแผนภาพเหล่านั้นให้เป็นภาษารูปนัย เนื่องจากภาษารูปนัยสามารถตรวจสอบความถูกต้องได้สะดวก อีกทั้งยังมีเครื่องมือที่เรียกว่า "เครื่องมือตรวจสอบแบบจำลอง" (model checker) ช่วยในการตรวจสอบความถูกต้อง เช่น จากงานวิจัยของมวนและบัตเลอร์ (Muan and Butler) [12] ได้เสนอวิธีการในการแปลงแผนภาพสถานะของยูเอ็มแอลให้เป็นซีเอสพี (CSP : Communicating and Sequential Processes) และใช้เครื่องมือตรวจสอบแบบจำลองที่เรียกว่าเอฟดีอาร์ (FDR) ในการตรวจสอบความถูกต้องของระบบ

นอกจากความถูกต้องของระบบแล้ว การแปลงแผนภาพต่าง ๆ ของยูเอ็มแอลที่ใช้ออกแบบระบบร่วมกันไปเป็นภาษารูปนัย ทำให้สามารถตรวจสอบความต้องกันระหว่างแผนภาพเหล่านั้นได้ ดังนั้นจากงานวิจัยของโตชิอากิ และคณะ (Toshiaki et al.) [13] ได้เสนอวิธีการแปลงแผนภาพ ต่าง ๆ ของยูเอ็มแอลให้เป็น "ระบบเชิงสัจพจน์" (axiomatic system) และใช้ระบบตรวจสอบทฤษฎีบท (theorem proving system) ที่พัฒนาขึ้นเองที่มีชื่อว่า ฮอล (HOL : Higher Order Logic) ในการตรวจสอบความต้องกันระหว่างแผนภาพเหล่านั้น แต่อย่างไรก็ตาม งานวิจัยของโตชิอากิยังไม่สามารถตรวจสอบระบบที่มีการทำงานแบบพร้อมกัน (concurrent system) ได้อย่างสมบูรณ์ เนื่องจากภาษารูปนัยที่ใช้ไม่สามารถบรรยายการทำงานแบบพร้อมกันได้

ดังนั้น ในงานวิจัยของมิซูตากะ และคณะ (Misutaka et al.) [14] ได้เสนอวิธีการตรวจสอบความต้องกันระหว่างแผนภาพสถานะของเครื่อง (state machine) และแผนภาพลำดับ (sequence diagram) โดยใช้ซีอาร์อี (CRE: Concurrent Regular Expression) ซึ่งมีความสามารถในการบรรยายระบบที่มีการทำงานแบบพร้อมกัน มาเป็นภาษาบรรยายแผนภาพทั้งสอง แต่อย่างไรก็ตามจากงานวิจัยนี้จะพบว่าขั้นตอนการแปลงแผนภาพสถานะของเครื่องไปเป็นซีอาร์อีนั้น มีกฎจำนวนมากทำให้เกิดความยุ่งยาก และซับซ้อนในการแปลง อีกทั้งการบรรยายสถานะของระบบด้วยแผนภาพสถานะของเครื่องนั้น ไม่สามารถบรรยายพฤติกรรมของระบบได้อย่างครบถ้วน เช่น ไม่สามารถกำหนดเงื่อนไขบนการแปลงสถานะได้ และไม่สามารถส่งลักษณะประจำ (attribute) ของข้อความได้ เป็นต้น

ไพแคลคูลัส เป็นภาษารูปนัยที่ใช้บรรยายพฤติกรรมของระบบที่มีการทำงานแบบพร้อมกัน ด้วยความสามารถนี้ไพแคลคูลัสจึงถูกนำมาใช้บรรยายบางแผนภาพของยูเอ็มแอล เช่น แผนภาพสถานะ [9-11] ทำให้สามารถบรรยายแผนภาพสถานะแบบที่มีสถานะประกอบที่สถานะย่อยทำงานแบบพร้อมกันได้ แต่อย่างไรก็ตามยังไม่มีงานวิจัยใดนำความสามารถของไพแคลคูลัสไปใช้บรรยายแผนภาพความร่วมมือของยูเอ็มแอลที่มีการส่งข้อความแบบพร้อมกัน

จากที่ได้กล่าวมา ผู้วิจัยจึงเกิดแนวคิดที่จะนำความสามารถของไพแคลคูลัสไปใช้ในการตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะ โดยการสร้างกฎการแปลงแผนภาพความร่วมมือ และแผนภาพสถานะไปเป็นไพแคลคูลัส ในรูปแบบที่สามารถตรวจสอบความต้องกันของระบบที่บรรยายไว้ด้วยแผนภาพทั้งสองได้

2.2 ทฤษฎีที่เกี่ยวข้อง

ในงานวิจัยนี้มีทฤษฎีที่เกี่ยวข้อง ดังนี้

2.2.1 ยูเอ็มแอล (UML : Unified Modeling Language)

ยูเอ็มแอล [1, 2] เป็นภาษามาตรฐานที่ใช้ในการออกแบบทางด้านวิศวกรรมซอฟต์แวร์ โดยยูเอ็มแอลเป็นภาษากึ่งรูปนัย (semi-formal language) ที่ใช้แผนภาพต่าง ๆ ในการวิเคราะห์และออกแบบระบบเชิงวัตถุเพื่อให้สามารถเก็บข้อตกลง และรายละเอียดต่าง ๆ เกี่ยวกับระบบที่พัฒนา โดยผู้พัฒนาระบบ และบุคคลที่เกี่ยวข้องสามารถเข้าใจรายละเอียด แก้ไข ปรับปรุง และเพิ่มเติมส่วนต่าง ๆ ของระบบที่ออกแบบด้วยยูเอ็มแอลก่อนที่จะสร้างระบบขึ้นมาใช้จริง หรือแม้กระทั่งใช้ยูเอ็มแอลเป็นเอกสารประกอบการบำรุงรักษาระบบในภายหลังได้

แผนภาพของยูเอ็มแอล ประกอบด้วยแผนภาพทั้งหมดจำนวน 9 แผนภาพ โดยสามารถแบ่งออกได้เป็น 2 กลุ่มใหญ่ ๆ ตามลักษณะการบรรยายระบบ ดังนี้

1) แผนภาพเชิงโครงสร้าง (Structural Model) เป็นแผนภาพที่ใช้อธิบายวัตถุต่าง ๆ ที่มีอยู่ในระบบ รวมถึงความสัมพันธ์ระหว่างวัตถุเหล่านั้น โดยประกอบด้วยแผนภาพจำนวน 4 แผนภาพ คือ

- แผนภาพคลาส (Class Diagram)
- แผนภาพวัตถุ (Object Diagram)
- แผนภาพส่วนประกอบ (Component Diagram)
- แผนภาพดิพลอยเมนท์ (Deployment Diagram)

2) แผนภาพเชิงพฤติกรรม (Behavioral Model) เป็นแผนภาพที่ใช้อธิบายพฤติกรรมของระบบ โดยประกอบด้วยแผนภาพจำนวน 5 แผนภาพ คือ

- แผนภาพยูสเคส (Use Case Diagram)
- แผนภาพลำดับ (Sequence Diagram)
- แผนภาพความร่วมมือ (Collaboration Diagram)
- แผนภาพสถานะ (Statechart Diagram)
- แผนภาพกิจกรรม (Activity Diagram)

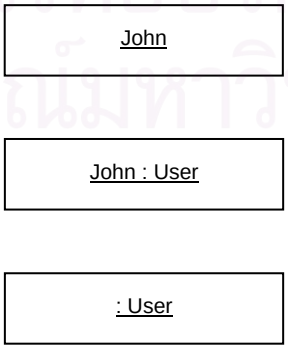
โดยแต่ละแผนภาพของยูเอ็มแอล จะให้มุมมองที่แตกต่างกันของระบบเดียวกัน ความสัมพันธ์ระหว่างแผนภาพนี้เองนำไปสู่การตรวจสอบความต้องกัน (consistency checking) ของแผนภาพต่าง ๆ เพื่อป้องกันความผิดพลาดที่อาจเกิดขึ้นในขั้นตอนการออกแบบระบบ

2.2.1.1 แผนภาพความร่วมมือ (Collaboration diagram)

แผนภาพความร่วมมือ เป็นแผนภาพเชิงพฤติกรรมที่แสดงถึงปฏิสัมพันธ์ (interaction) ของวัตถุต่าง ๆ ที่มีส่วนร่วมอยู่ในระบบ โดยวัตถุที่อยู่ในแผนภาพความร่วมมือแสดงได้ด้วยรูปสี่เหลี่ยมซึ่งมีชื่อของวัตถุนั้นเขียนกำกับอยู่ โดยชื่อของวัตถุสามารถเขียนได้ 3 รูปแบบ คือ

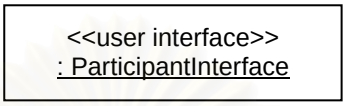
1) ชื่อวัตถุ
2) ชื่อวัตถุ : ชื่อคลาส
3) : ชื่อคลาส

ดังแสดงในรูปที่ 2.1



รูปที่ 2.1 วัตถุในแผนภาพความร่วมมือ

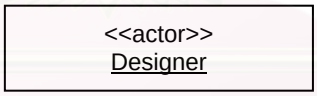
นอกจากนี้บนชื่อของวัตถุอาจมีชื่อประจำรูปแบบ (stereotype name) เขียนกำกับไว้ด้วยก็ได้ โดยเขียนไว้ภายใต้เครื่องหมาย << >> ประโยชน์ของชื่อประจำรูปแบบ คือ เพื่อจำแนกความแตกต่างระหว่างวัตถุต่าง ๆ ในระบบ เช่น <<user interface>> เป็นการระบุว่าวัตถุดังกล่าวเป็นส่วนประสานกับผู้ใช้ เป็นต้น ดังแสดงในรูปที่ 2.2



รูปที่ 2.2 วัตถุในแผนภาพความร่วมมือที่มีชื่อประจำรูปแบบ (stereotype name) เขียนกำกับไว้

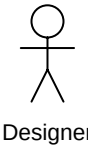
และหากวัตถุใดเป็นผู้กระทำหรือผู้ใช้ (actor) ซึ่งได้แก่ เอนทิตีภายนอก (external entity) ระบบย่อย (sub system) หรือคลาสที่อยู่ภายนอกระบบแต่มีปฏิสัมพันธ์โดยตรงกับระบบ กล่าวคือผู้กระทำสามารถส่งข้อความให้ระบบ หรือได้รับข้อความจากระบบ สัญกรณ์ (notation) ที่ใช้เขียนแสดงผู้กระทำในแผนภาพความร่วมมือมี 2 รูปแบบ คือ

1) แบบสัญกรณ์ของคลาส เป็นรูปสี่เหลี่ยมที่ภายในเขียนชื่อประจำรูปแบบว่า <<actor>> ดังรูปที่ 2.3



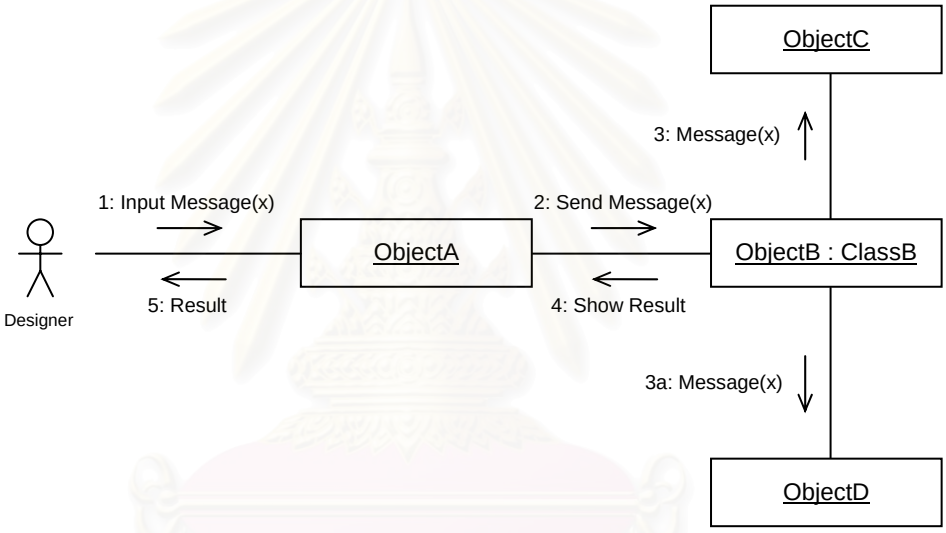
รูปที่ 2.3 ผู้กระทำแบบสัญกรณ์ของคลาส

2) แบบสัญรูปประจำรูปแบบ (stereotype icon) เป็นรูปคน (stick man) โดยเขียนชื่อของผู้กระทำกำกับไว้ทางด้านล่างของรูปคนดังกล่าว ดังแสดงในรูปที่ 2.4



รูปที่ 2.4 ผู้กระทำแบบสัญรูปประจำรูปแบบ

ในแผนภาพความร่วมมือจะมีเส้นเชื่อมต่อระหว่างวัตถุ (object interconnection) เชื่อมระหว่างรูปสี่เหลี่ยมสองรูปที่แทนวัตถุที่มีความสัมพันธ์กัน บนเส้นเชื่อมต่อระหว่างวัตถุอาจมีข้อความ (message) เขียนกำกับไว้ด้วยก็ได้ โดยข้อความหนึ่ง ๆ ประกอบไปด้วย ตัวเลขแสดงลำดับของข้อความ ชื่อของข้อความ และลูกศรแสดงทิศทางของข้อความซึ่งเขียนไว้ด้วยกัน นอกจากนี้บนเส้นเชื่อมต่อระหว่างวัตถุคู่หนึ่ง ๆ สามารถมีข้อความส่งถึงกันได้หลายข้อความทั้งในทิศทางออกจากวัตถุและเข้าหาวัตถุ และหากมีข้อความใดเกิดขึ้นพร้อมกันให้เขียนตัวอักษรภาษาอังกฤษ (ตัวพิมพ์เล็ก) เขียนกำกับไว้ทางด้านหลังของตัวเลขแสดงลำดับ เช่น 1, 1a, 1b,... เป็นต้น ดังแสดงในรูปที่ 2.5



รูปที่ 2.5 แผนภาพความร่วมมือ

จากรูปจะพบว่าชื่อของข้อความนั้นประกอบด้วย

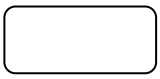
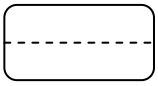
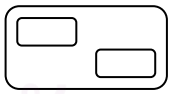
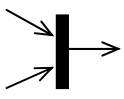
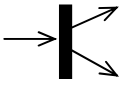
- ตัวเลขซึ่งแสดงลำดับของข้อความ
- ตัวอักษรซึ่งแสดงว่าข้อความเหล่านั้นเกิดขึ้นพร้อมกัน เช่น จากรูปที่ 2.5 จะพบว่าข้อความ 3 และ 3a ถูกส่งไปพร้อมกัน
- ชื่อของข้อความซึ่งก็คือเหตุการณ์ที่เกิดขึ้นกับระบบ รวมถึงลักษณะประจำ (attribute) ที่ส่งมาพร้อมกับข้อความนั้น โดยสามารถเขียนเป็นสมการได้ดังนี้ คือ message = event(message attributes)

แผนภาพความร่วมมือ สามารถบรรยายพฤติกรรมของระบบได้เหมือนกันกับแผนภาพลำดับเพียงแต่แสดงในมิติ (dimension) ที่ต่างกัน [3, 15] โดยแผนภาพความร่วมมือเน้นที่การพิจารณามิติของปฏิสัมพันธ์ระหว่างวัตถุมากกว่าช่วงเวลาที่เกิดปฏิสัมพันธ์ ซึ่งตรงข้ามกับแผนภาพลำดับ

2.2.1.2 แผนภาพสถานะ (Statechart diagram)

แผนภาพสถานะ เป็นแผนภาพเชิงพฤติกรรมที่ใช้บรรยายถึงการเปลี่ยนแปลงพฤติกรรมของวัตถุต่าง ๆ ที่มีอยู่ในระบบ ซึ่งถูกพัฒนาขึ้นโดยฮาเรล (Harel) [16] แผนภาพสถานะมีส่วนประกอบพื้นฐาน 2 ส่วน คือ สถานะ (state) และการแปลงสถานะ (state transition) โดย สัญกรณ์แทนสถานะแสดงด้วยรูปสี่เหลี่ยมมุมมน ยกเว้นแต่สถานะดังกล่าวเป็นสถานะเทียม (pseudostate) เช่น สถานะเริ่มต้น (initial state) ให้ใช้สัญกรณ์วงกลมสีดำ หรือสถานะดังกล่าวเป็นสถานะพิเศษ (special state) เช่น สถานะสิ้นสุด (final state) ให้ใช้สัญกรณ์วงกลมรูปตาวัว (bull's eye) เป็นต้น โดยสัญกรณ์แทนสถานะต่าง ๆ ในแผนภาพสถานะแสดงไว้ในตารางที่ 2.1

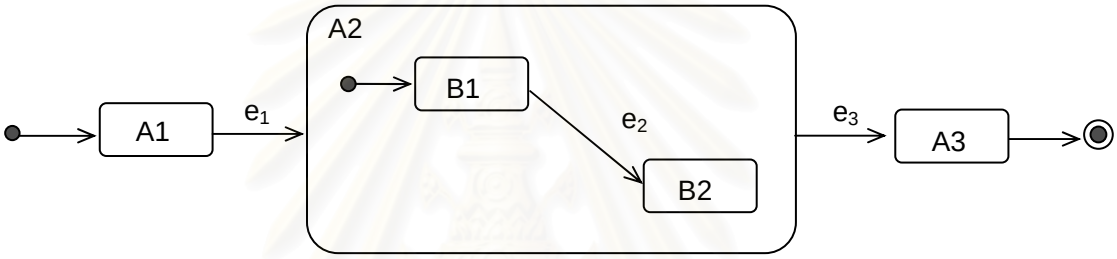
ตารางที่ 2.1 สัญกรณ์แทนสถานะต่าง ๆ ในแผนภาพสถานะ

ประเภทของสถานะ	สัญกรณ์
สถานะทั่วไป (simple state or non-composite state)	
สถานะประกอบแบบสถานะย่อยทำงานพร้อมกัน (concurrent composite state)	
สถานะประกอบแบบสถานะย่อยทำงานไม่พร้อมกัน (non-concurrent composite state)	
สถานะเริ่มต้น(initial state)	
สถานะสิ้นสุด (final state)	
สถานะบรรจบ (join state)	
สถานะแยก (fork state)	

โดยสถานะในแผนภาพสถานะสามารถแบ่งออกได้เป็น 5 ประเภท ดังนี้

1) สถานะทั่วไป (simple state or non-composite state) คือ สถานะที่ไม่มีสถานะย่อยอยู่ภายใน

2) สถานะประกอบแบบสถานะย่อยทำงานไม่พร้อมกัน (non-concurrent composite state) คือ สถานะที่มีสถานะย่อย ๆ อย่างน้อย 1 สถานะทำงานต่อเนื่องกันอยู่ภายใน โดยเมื่อวัตถุอยู่ในสถานะประกอบประเภทนี้จะมีสถานะย่อยเพียงสถานะเดียวเท่านั้นที่ทำงาน ดังตัวอย่างไปนี้ ซึ่งเป็นแผนภาพสถานะของวัตถุ A

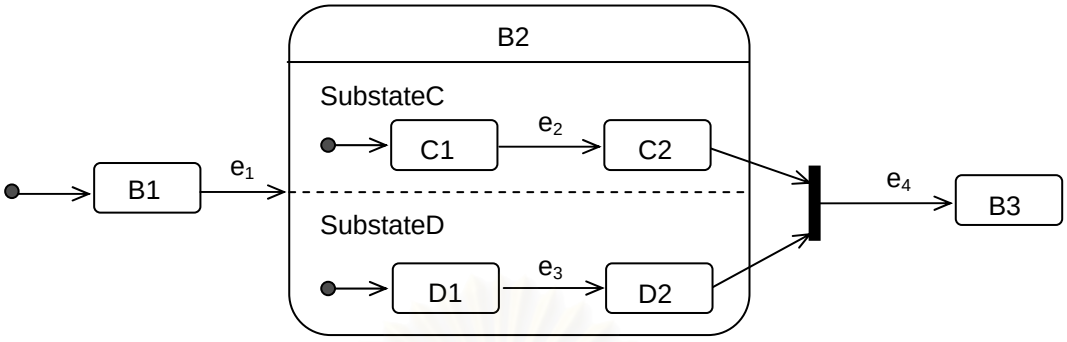


รูปที่ 2.6 ตัวอย่างแผนภาพสถานะที่มีสถานะประกอบแบบสถานะย่อยทำงานไม่พร้อมกัน

จะพบว่าแผนภาพสถานะในรูปที่ 2.6 ประกอบด้วยสถานะทั่วไป 2 สถานะ คือ A_1 และ A_3 และสถานะประกอบ 1 สถานะ คือ A_2 ซึ่งภายในมีสถานะย่อย 2 สถานะ คือ B_1 และ B_2

โดยเมื่อวัตถุ A เริ่มทำงานจะอยู่ที่สถานะ A_1 และหากเกิดเหตุการณ์ e_1 ขึ้น จะทำให้วัตถุ A เปลี่ยนสถานะไปเป็น A_2 พร้อมกับสถานะย่อย B_1 เริ่มทำงาน ณ ขณะนี้ หากเกิดเหตุการณ์ e_2 ขึ้นกับวัตถุ A จะทำให้สถานะย่อย B_2 เริ่มทำงาน แต่หากเกิดเหตุการณ์อื่น ๆ ที่ไม่ใช่ e_2 วัตถุ A จะอยู่ที่สถานะย่อยเดิม สุดท้ายหากวัตถุ A อยู่ที่สถานะย่อย B_2 และเกิดเหตุการณ์ e_3 ขึ้น วัตถุ A จะเปลี่ยนสถานะไปเป็น A_3 และจบการทำงาน

3) สถานะประกอบแบบสถานะย่อยทำงานพร้อมกัน (concurrent composite state) คือ สถานะที่ถูกแบ่งออกเป็นสถานะย่อย ๆ ที่ทำงานพร้อมกันตั้งแต่ 2 สถานะขึ้นไปด้วยเส้นประ โดยเมื่อวัตถุอยู่ในสถานะประกอบประเภทนี้สถานะย่อยทั้งหมดจะทำงานพร้อมกัน ดังตัวอย่างไปนี้ ซึ่งเป็นแผนภาพสถานะของวัตถุ B



รูปที่ 2.7 ตัวอย่างแผนภาพสถานะที่มีสถานะประกอบแบบสถานะย่อยทำงานพร้อมกัน

จะพบว่าแผนภาพสถานะในรูปที่ 2.7 ประกอบด้วยสถานะทั่วไป 2 สถานะ คือ B_1 และ B_3 และสถานะประกอบ 1 สถานะ คือ B_2 ซึ่งภายในมีสถานะย่อย 2 สถานะ คือ *SubstateC* และ *SubstateD* ที่มีการทำงานแบบพร้อมกัน

โดยเมื่อวัตถุ B เริ่มทำงานจะอยู่ที่สถานะ B_1 และหากเกิดเหตุการณ์ e_1 ขึ้น จะทำให้วัตถุ B เปลี่ยนสถานะไปเป็น B_2 พร้อมกับสถานะย่อย *SubstateC* และ *SubstateD* เริ่มทำงานพร้อมกัน โดยอยู่ที่สถานะ C_1 และ D_1 ตามลำดับ ณ ขณะนี้ หากเกิดเหตุการณ์ e_2 ขึ้นกับวัตถุ B จะทำให้สถานะย่อย *SubstateC* เปลี่ยนเป็นสถานะ C_2 โดยสถานะย่อย *SubstateD* ไม่เปลี่ยนสถานะ แต่หากเกิดเหตุการณ์ e_3 จะทำให้สถานะย่อย *SubstateD* เปลี่ยนเป็นสถานะ D_2 โดยสถานะย่อย *SubstateC* ไม่เปลี่ยนสถานะ สุดท้ายหากวัตถุ B อยู่ที่สถานะย่อย C_2 และ D_2 ถ้าเกิดเหตุการณ์ e_4 ขึ้น วัตถุ B จะเปลี่ยนสถานะไปเป็น B_3

4) สถานะเทียม (pseudostate) คือ สถานะที่ไม่มีการประมวลผลเหตุการณ์ โดยอาจเป็นสถานะที่เป็นทางเชื่อมระหว่างการแปลงสถานะ 2 เส้นขึ้นไป ได้แก่ สถานะเริ่มต้น (initial state) สถานะบรรจบ (join state) สถานะแยก (fork state) เป็นต้น

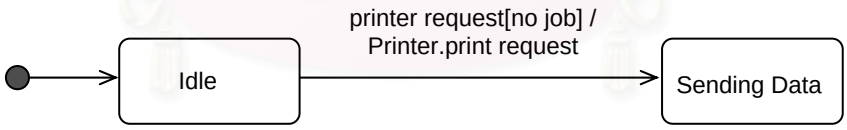
5) สถานะพิเศษ (special state) คือ สถานะที่อาจจะยังคงทำงาน ถึงแม้ว่าการทำงานตามที่บรรยายไว้ในแผนภาพสถานะจะเสร็จสิ้นแล้ว ได้แก่ สถานะสิ้นสุด (final state) สถานะประสานเวลา (sync state)

ส่วนการแปลงสถานะ (state transition) แสดงด้วยลูกศรที่เชื่อมระหว่างสถานะ 2 สถานะ โดยมีทิศทางจากสถานะต้นทาง (source state) ไปยังสถานะปลายทาง (target state) และบนการแปลงสถานะอาจเขียนข้อความกำกับไว้ก็ได้ โดยข้อความที่เขียนประกอบด้วย 3 ส่วน ดังนี้ คือ

- เหตุการณ์ (event) คือ สาเหตุที่ทำให้เกิดการเปลี่ยนแปลงสถานะจากสถานะต้นทางไปเป็นสถานะปลายทาง
- เงื่อนไข (guard-condition) คือ ข้อกำหนดที่มีค่าเป็นจริงหรือเท็จ โดยถ้าเกิดเหตุการณ์ขึ้น และเงื่อนไขเป็นจริงจะเกิดการเปลี่ยนแปลงสถานะ แต่ถ้าเกิดเหตุการณ์ขึ้น แต่เงื่อนไขเป็นเท็จก็จะไม่เกิดการเปลี่ยนแปลงสถานะ
- การกระทำ (action) คือ เหตุการณ์ซึ่งเป็นผลจากการเปลี่ยนแปลงสถานะ โดยสามารถระบุเป็นลำดับของหลาย ๆ กิจกรรมได้

การเขียนข้อความกำกับบนการแปลงสถานะไม่จำเป็นต้องระบุทั้ง 3 ส่วน อาจเว้นส่วนใดส่วนหนึ่งไว้ก็ได้ โดยมีวิธีการเขียน คือ เหตุการณ์[เงื่อนไข] / การกระทำ

พิจารณาตัวอย่างในรูปที่ 2.8 ซึ่งเป็นแผนภาพสถานะของเครื่องคอมพิวเตอร์เครื่องหนึ่งที่ต่ออยู่กับเครื่องพิมพ์ (printer)



รูปที่ 2.8 ตัวอย่างแผนภาพสถานะของเครื่องคอมพิวเตอร์เครื่องหนึ่งที่ต่ออยู่กับเครื่องพิมพ์

จากรูปที่ 2.8 จะพบว่าบนการแปลงสถานะที่เชื่อมระหว่างสถานะ Idle และ Sending Data จะมีข้อความ “printer request[no job] / Printer.print request” เขียนกำกับอยู่ โดยมีความหมายว่า เมื่อเริ่มทำงานเครื่องคอมพิวเตอร์เครื่องนี้จะอยู่ในสถานะ Idle แต่ถ้ามีเหตุการณ์ “printer request” เกิดขึ้น และเงื่อนไขที่กำหนดไว้ คือ “no job” มีค่าความจริงเป็นจริง เครื่องคอมพิวเตอร์เครื่องนี้ก็จะเกิดการกระทำ “Printer.print request” ซึ่งหมายความว่า จะส่งข้อความ “print request” ไปให้วัตถุที่มีชื่อว่า “Printer” แต่ถ้าเงื่อนไขที่กำหนดไว้ คือ “no job” มีค่าความจริงเป็นเท็จการกระทำดังกล่าวก็จะไม่เกิดขึ้น

2.2.2 ไพแคลคูลัส (π-Calculus)

ไพแคลคูลัส [4] เป็นแคลคูลัสที่ใช้บรรยายพฤติกรรมของระบบสื่อสาร (communicating system) ซึ่งพัฒนามาจากซีซีเอส (CCS : Calculus of Comunicating Systems) โดยเพิ่มความสามารถในการเปลี่ยนแปลงโครงสร้างของช่องสื่อสาร ให้สามารถสร้างช่องสื่อสารขึ้นใหม่ได้จากข้อมูลที่ได้รับมาจากช่องสื่อสารเดิมได้

ส่วนประกอบของไพแคลคูลัสมี 2 ส่วน คือ เอเยนต์ (agent) หรือ กระบวนการ (process) และช่องสื่อสาร (communicating channel) โดยการกำหนดชื่อของกระบวนการนิยมใช้ตัวอักษรภาษาอังกฤษเพียงตัวเดียวเขียนด้วยตัวพิมพ์ใหญ่ เช่น P หรือ Q เป็นต้น หรือถ้าจะเขียนเป็นคำ ให้ขึ้นต้นด้วยตัวพิมพ์ใหญ่เช่นกัน เช่น Computer หรือ Car เป็นต้น ส่วนการกำหนดชื่อของช่องสื่อสารนิยมใช้ตัวอักษรภาษาอังกฤษเป็นตัวพิมพ์เล็ก โดยอาจเขียนเพียงตัวเดียว เช่น a หรือ b เป็นต้น หรือจะเขียนเป็นคำก็ได้ เช่น message หรือ signal เป็นต้น

ช่องสื่อสารในไพแคลคูลัส มี 2 ประเภท คือ ช่องสื่อสารที่รอรับข้อมูล และช่องสื่อสารที่ส่งข้อมูล โดยชื่อของช่องสื่อสารที่รอรับข้อมูลให้เขียนตามปกติ แต่ถ้าเป็นชื่อของช่องสื่อสารที่ส่งข้อมูล ให้ขีดเส้นไว้ทางด้านบนของชื่อช่องสื่อสารดังกล่าว เช่น $\overline{a}$ หรือ $\overline{message}$ เป็นต้น ดังตัวอย่างต่อไปนี้

รูปที่ 2.9 ตัวอย่างกระบวนการ และช่องสื่อสารในไพแคลคูลัส

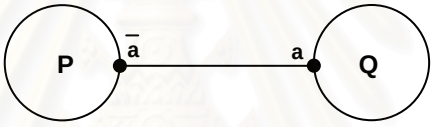
แผนภาพในรูปที่ 2.9 เรียกว่า กราฟการไหล (flowgraph) ประกอบไปด้วยกระบวนการ ซึ่งเขียนแสดงด้วยวงกลมที่มีชื่อของกระบวนการอยู่ตรงกลาง และช่องสื่อสาร ซึ่งเขียนแสดงด้วยวงกลมเล็ก ๆ ที่มีชื่อของช่องสื่อสารเขียนกำกับอยู่เช่นกัน โดยจากรูปหมายความว่ากระบวนการชื่อ *Agent*1 มีช่องสื่อสาร 2 ช่อง คือ *in* และ $\overline{out}$ ซึ่งเป็นช่องสื่อสารที่รอรับข้อมูลและช่องสื่อสารที่ส่งข้อมูล ตามลำดับ

ต่อไปหากต้องการบรรยายพฤติกรรมของกระบวนการด้วยไพแคลคูลัส สามารถทำได้ ดังนี้

$$Agent1 \overset{def}{=} in(x).Agent1'(x)$$
$$Agent1'(x) \overset{def}{=} \overline{out}\langle x\rangle.Agent1$$

ความหมายของไพแคลคูลัสข้างต้นคือ หาก *Agent*1 ได้รับค่าผ่านช่องสื่อสาร *in* จะนำค่าดังกล่าวไปแทนที่ตัวแปร x และเปลี่ยนสถานะเป็น *Agent*1′ ต่อจากนั้นหาก *Agent*1′ ถูกกระตุ้นที่ช่องสื่อสาร $\overline{out}$ ก็จะส่งค่าที่แทนตัวแปร x ออกทางช่องสื่อสารดังกล่าว และเปลี่ยนสถานะเป็น *Agent*1

และหากต้องการให้กระบวนการหลาย ๆ กระบวนการสามารถสื่อสารระหว่างกันได้ ให้นำกระบวนการที่มีชื่อของช่องสื่อสารตรงกันมาเชื่อมต่อกัน ทั้งนี้ต้องให้กระบวนการหนึ่งมีช่องสื่อสารที่ส่งข้อมูล และอีกกระบวนการหนึ่งมีช่องสื่อสารที่รอรับข้อมูล ดังตัวอย่างต่อไปนี้



รูปที่ 2.10 ตัวอย่างกระบวนการ 2 กระบวนการที่เชื่อมต่อกัน

จากรูปที่ 2.10 พบว่ากระบวนการ P และ Q มีช่องสื่อสารที่มีชื่อตรงกัน คือ a โดยกระบวนการ P จะส่งค่าออกทางช่องสื่อสาร a ส่วนกระบวนการ Q จะคอยรับค่าที่ส่งมาทางช่องสื่อสารดังกล่าว เช่นกัน และถ้ากำหนดให้พฤติกรรมของกระบวนการ P และ Q เป็นดังนี้ คือ $P \overset{def}{=} \overline{a}\langle 5\rangle.P'$ และ $Q \overset{def}{=} a(x).Q'$ ตามลำดับ โดยหากกระบวนการ P ถูกกระตุ้นที่ช่องสื่อสาร a กระบวนการ P จะส่งค่า 5 ให้กระบวนการ Q และเมื่อกระบวนการ Q ได้รับค่าดังกล่าวก็จะนำไปแทนที่ตัวแปร x

โดยจากรูปที่ 2.10 สามารถเขียนเป็นไพแคลคูลัสได้ว่า $P \mid Q$ ซึ่งมีความหมายว่าให้กระบวนการ P และกระบวนการ Q มีการติดต่อสื่อสารระหว่างกัน หรืออีกความหมายหนึ่งก็คือ ให้กระบวนการ P และ Q ทำงานแบบขนานกัน (parallel)

การทำงานของไพแคลคูลัสจะใช้วิธีการที่เรียกว่า “การลดทอน” (reduction) เช่น ถ้ามีการสื่อสารกันระหว่างกระบวนการ P และ Q ดังนี้ คือ $y(x).P \mid \overline{y}\langle z\rangle.Q$ จะสามารถลดทอนได้เป็น $P\{z/x\} \mid Q$ ซึ่งจะพบว่าช่องสื่อสาร y หายไปเมื่อการสื่อสารเสร็จเรียบร้อยแล้ว โดย

ความหมายคือ เมื่อมีการสื่อสารผ่านทางช่องสื่อสาร y เกิดขึ้น กระบวนการ P จะได้รับค่า z มาเก็บไว้แทนที่ตัวแปร x ของตนเอง

ดังนั้นหากพิจารณาการทำงานของระบบที่ประกอบด้วยกระบวนการ P และ Q สื่อสารกัน จะสามารถบรรยายด้วยไพแคลคูลัสได้ ดังนี้

$$P \mid Q \quad = \bar{a}\langle 5\rangle.P' \mid a(x).Q' \tag{2.1}$$

$$\xrightarrow{\bar{a}} P' \mid a(x).Q' \tag{2.2}$$

$$\xrightarrow{a} P' \mid Q'\{5/x\} \tag{2.3}$$

ไพแคลคูลัสข้างต้นมีความหมายว่าระบบที่ประกอบด้วยกระบวนการ P และ Q สื่อสารกัน จะรอรับข้อความจากภายนอก โดยจากนิพจน์ (2.1) แสดงว่ามีการกระตุ้นที่ช่องสื่อสาร $\bar{a}$ ทำให้กระบวนการ P ส่งค่า 5 ออกมาทางช่องสื่อสาร $\bar{a}$ และเปลี่ยนสถานะเป็น P' ต่อมาจากนิพจน์ (2.3) แสดงว่าเกิดการกระตุ้นที่ช่องสื่อสาร a ของกระบวนการ Q ทำให้กระบวนการ Q นำค่าที่ได้จากช่องสื่อสาร a ไปแทนที่ตัวแปร x แล้วเปลี่ยนสถานะเป็น Q'

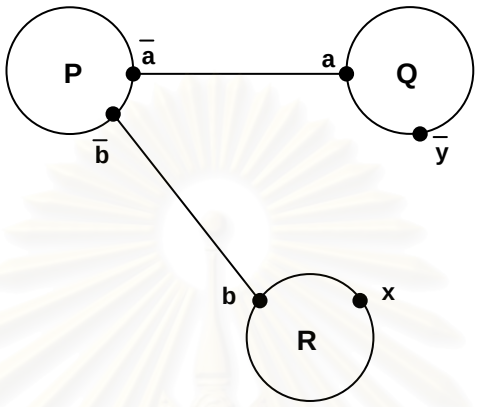
ดังนั้นหากต้องการให้ช่องสื่อสารใดเป็นช่องสื่อสารภายในสามารถทำได้โดยการปกปิด (blind) ช่องสื่อสารนั้น โดยจากรูปที่ 2.10 หากต้องการให้ a เป็นช่องสื่อสารภายในสามารถเขียนเป็นไพแคลคูลัสได้ดังนี้ คือ $new\,a(P \mid Q)$ และเมื่อให้ a เป็นช่องสื่อสารภายใน การทำงานของระบบข้างต้นจะเปลี่ยนไป ดังนี้

$$P \mid Q \quad = \bar{a}\langle 5\rangle.P' \mid a(x).Q' \tag{2.4}$$

$$\xrightarrow{\tau} P' \mid Q'\{5/x\} \tag{2.5}$$

จากนิพจน์ (2.5) จะพบว่ามีสัญลักษณ์ใหม่ คือ τ ซึ่งแสดงว่าเกิดการสื่อสารภายในเกิดขึ้น โดยการสื่อสารภายในจะหมายถึงทั้งการส่ง และรับข้อมูล

ต่อไปพิจารณารูป 2.11 ซึ่งมีกระบวนการ 3 กระบวนการสื่อสารกัน คือ กระบวนการ P Q และ R



รูปที่ 2.11 ตัวอย่างระบบที่ประกอบด้วยกระบวนการที่มีทางเลือกเชิงไม่กำหนด

จากรูปที่ 2.11 ถ้ากำหนดให้พฤติกรรมของกระบวนการ P Q และ R เป็น (2.6) (2.7) และ (2.8) ตามลำดับ ดังนี้

$$P \overset{def}{=} \bar{a}\langle x\rangle.P' + \bar{b}\langle x\rangle.P'' \tag{2.6}$$

$$Q \overset{def}{=} a(y).\bar{y}\langle z\rangle.0 \tag{2.7}$$

$$R \overset{def}{=} b(c).0 + x(c).R' \tag{2.8}$$

ดังนั้นจากรูปที่ 2.11 สามารถเขียนเป็นไพแคลคูลัสได้ว่า $P \mid Q \mid R$ โดยประโยค $P \overset{def}{=} \bar{a}\langle x\rangle.P' + \bar{b}\langle x\rangle.P''$ มีความหมายว่ากระบวนการ P ถ้าได้รับการกระตุ้นที่ช่องสื่อสาร a จะส่งค่า x ไปให้กระบวนการ Q และเปลี่ยนสถานะเป็น P' แต่ถ้ากระบวนการ P ถ้าได้รับการกระตุ้นที่ช่องสื่อสาร b จะส่งค่า x ไปให้กระบวนการ R และเปลี่ยนสถานะเป็น P''

ต่อไปพิจารณาเหตุการณ์ต่อไปนี้

$$\begin{aligned} P \mid Q \mid R &= new\ a\ b\ x\ (\bar{a}\langle x\rangle.P' + \bar{b}\langle x\rangle.P'' \mid a(y).\bar{y}\langle z\rangle.0 \mid b(c).0 + x(c).R') \\ &\xrightarrow{\tau} new\ a\ b\ x\ (P' \mid \bar{x}\langle z\rangle.0 \mid b(c).0 + x(c).R') \\ &\xrightarrow{\tau} new\ a\ b\ x\ (P' \mid 0 \mid R'\{z/c\}) \end{aligned}$$

จากไพแคลคูลัสข้างต้นจะพบว่าหลังจากกระบวนการ Q ได้รับค่า x จากกระบวนการ P ผ่านช่องสื่อสาร a แล้ว กระบวนการ Q นำค่า x ที่ได้รับไปแทนที่ตัวแปร y ทำให้เกิดช่องทางการสื่อสารใหม่ออกจากกระบวนการ Q ไปยังกระบวนการ R ทำให้กระบวนการ Q สามารถส่งค่า z ไปให้กระบวนการ R ได้ ซึ่งความสามารถในการสร้างช่องสื่อสารขึ้นใหม่จากข้อมูลที่ได้รับมาจากช่องสื่อสารเดิมนั้นเป็นจุดเด่นของไพแคลคูลัสดังที่ได้กล่าวมาแล้ว

จากที่ได้อธิบายมาข้างต้น ถ้ากำหนดให้

P Q และ 0 เป็นกระบวนการ
x และ y เป็นช่องสื่อสาร
$\vec{x}$ เป็นลำดับของช่องสื่อสาร โดย $\vec{x} = x_1, x_2, ..., x_n$

วากยสัมพันธ์ (syntax) ของไพแคลคูลัสมี ดังนี้

- 0 หมายถึง กระบวนการที่ไม่มีการทำงาน
- $P + Q$ หมายถึง ทางเลือกเชิงไม่กำหนด (non-deterministic choice) ระหว่างกระบวนการ P หรือ กระบวนการ Q
- $(new\ \vec{x})P$ หมายถึง การสร้างลำดับของช่องสื่อสาร $x_1, x_2, ..., x_n$ ขึ้นใหม่เพื่อใช้ติดต่อกับกระบวนการ P
- $\overline{y}\langle x\rangle.P$ หมายถึง การส่งค่า x ผ่านช่องสื่อสาร y แล้วทำงานต่อไปในฐานะกระบวนการ P
- $y(x).P$ หมายถึง การรับค่า x ผ่านช่องสื่อสาร y แล้วทำงานต่อไปในฐานะกระบวนการ P
- τ หมายถึง การกระทำที่เกิดขึ้นภายในระบบ
- $P \mid Q$ หมายถึง การทำงานพร้อมกันของกระบวนการ P และ Q
- $[x = y]P$ หมายถึง กระบวนการ P จะถูกทำงาน เมื่อค่าของตัวแปร x และ y เท่ากัน

นอกจากนี้การสื่อสารระหว่างกระบวนการต่าง ๆ ของไพแคลคูลัสจะเป็นแบบซิงโครนัส (synchronous) เช่น $y(\vec{x}).P$ หมายความว่า การสื่อสารผ่านช่องสื่อสาร y จะถูกกั้นไม่ให้ทำกระบวนการ P จนกว่าจะได้รับลำดับของช่องสื่อสาร $\vec{x}$ จนครบ และในทางกลับกัน $\bar{y}\langle\vec{x}\rangle.P$ หมายความว่า การสื่อสารผ่านช่องสื่อสาร y จะถูกกั้นไม่ให้ทำกระบวนการ P จนกว่าจะส่งลำดับของช่องสื่อสาร $\vec{x}$ จนครบ

บทที่ 3

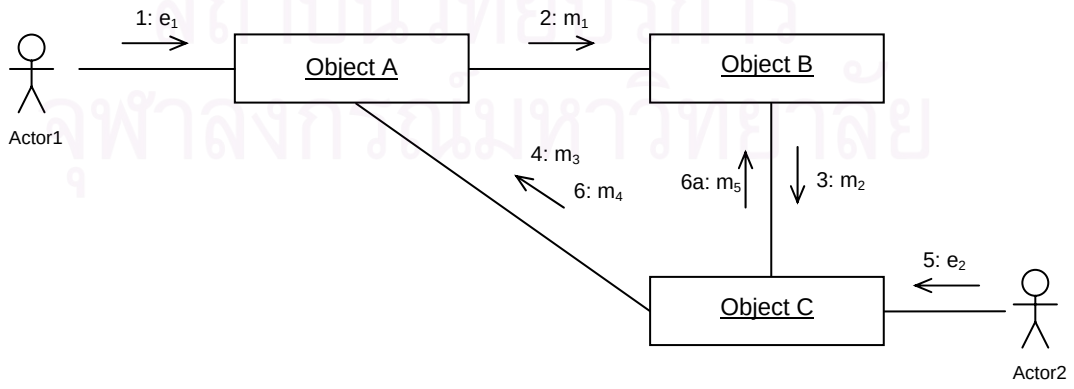
การตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะโดยใช้ไพแคลคูลัส

เนื้อหาในบทนี้จะกล่าวถึงขั้นตอนในการตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะของยูเอ็มแอลโดยใช้ไพแคลคูลัส โดยเริ่มต้นด้วยแนวคิดในการตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะ จากนั้นผู้วิจัยได้เสนอกฎการแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัส และกฎการแปลงแผนภาพสถานะไปเป็นไพแคลคูลัสในรูปแบบที่สามารถตรวจสอบความต้องกันระหว่างแผนภาพทั้งสองได้

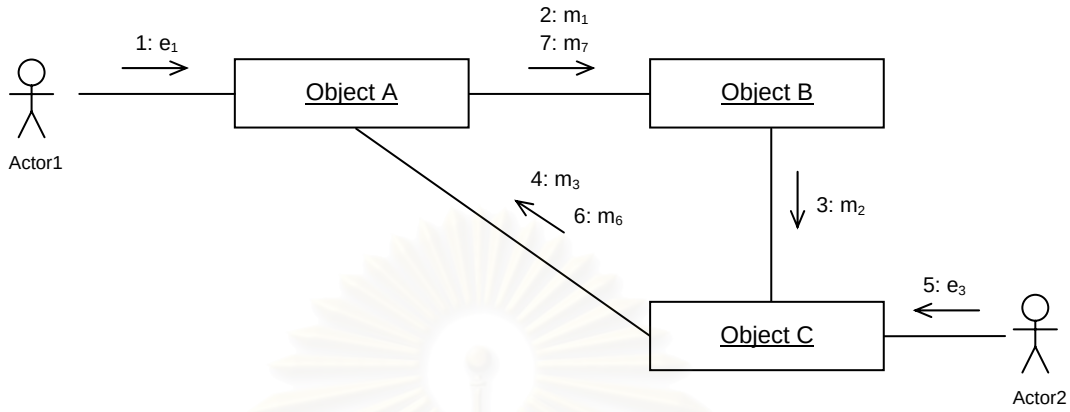
3.1 แนวคิดในการตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะโดยใช้ไพแคลคูลัส

การตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะ หมายถึง การตรวจสอบว่าไม่เกิดการขัดแย้งกันของเหตุการณ์ที่สามารถเกิดขึ้นได้จากแผนภาพความร่วมมือ เทียบกับเหตุการณ์ที่เป็นไปได้ทั้งหมดของวัตถุในระบบที่ทำงานไปพร้อม ๆ กันซึ่งได้ถูกบรรยายไว้ในแผนภาพสถานะ ดังนั้นหากมีบางเหตุการณ์ที่บรรยายไว้ในแผนภาพความร่วมมือแต่ไม่ปรากฏในแผนภาพสถานะแสดงว่าเกิดความไม่สอดคล้องกันระหว่างแผนภาพทั้งสอง

สมมติให้ระบบที่ศึกษามีวัตถุ 3 วัตถุ ทำงานร่วมกัน คือ วัตถุ A B และ C โดยพฤติกรรมของวัตถุทั้งสาม สามารถแสดงได้ด้วยแผนภาพความร่วมมือ ดังรูปที่ 3.1 และ 3.2



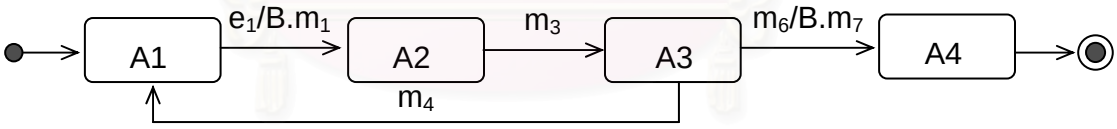
รูปที่ 3.1 แผนภาพความร่วมมือของ Scenario1



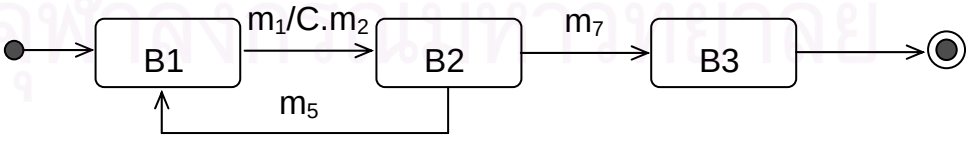
รูปที่ 3.2 แผนภาพความร่วมมือของ Scenario2

จากรูปทั้งสองจะพบว่า ในแผนภาพความร่วมมือมีการบรรยายถึงปฏิสัมพันธ์ระหว่างวัตถุต่าง ๆ ที่มีอยู่ในระบบ โดยบรรยายถึงข้อความที่ส่งระหว่างวัตถุ (อย่างมีลำดับ) แต่จะไม่บรรยายถึงพฤติกรรมของแต่ละวัตถุเมื่อได้รับข้อความ

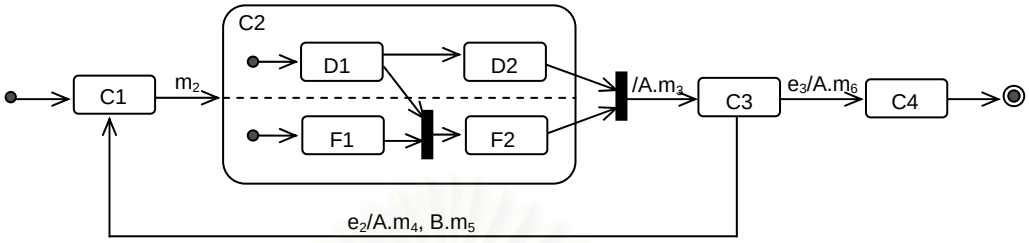
พฤติกรรมของแต่ละวัตถุสามารถบรรยายได้โดยใช้แผนภาพสถานะ ดังรูปที่ 3.3 รูปที่ 3.4 และรูปที่ 3.5 ซึ่งเป็นแผนภาพสถานะของวัตถุ A B และ C ตามลำดับ



รูปที่ 3.3 แผนภาพสถานะของวัตถุ A



รูปที่ 3.4 แผนภาพสถานะของวัตถุ B



รูปที่ 3.5 แผนภาพสถานะของวัตถุ C

ในการบรรยายระบบด้วยแผนภาพสถานะนั้น สามารถพิจารณาถึงลำดับของข้อความ (เหตุการณ์) ต่าง ๆ ที่เกิดขึ้นในระบบได้ โดยการพิจารณาแผนภาพสถานะของวัตถุทั้งสามร่วมกัน

กำหนดให้

E เป็นเซตของเหตุการณ์ที่มีการส่งข้อความมาจากผู้ใช้ (Actor) ภายนอกระบบ

M เป็นเซตของข้อความภายในระบบ

$e_1, e_2, e_3 \in E$

$m_1, m_2, .., m_7 \in M$

ระบบเริ่มต้นทำงานก็ต่อเมื่อวัตถุ A ได้รับเหตุการณ์ e_1 จากผู้ใช้ โดยหลังจากวัตถุ A ได้รับเหตุการณ์ e_1 แล้ววัตถุ A จะส่งข้อความ m_1 ไปให้วัตถุ B (แสดงด้วยสัญลักษณ์ $B.m_1$) และเปลี่ยนสถานะจาก $A1$ ไปเป็น $A2$ ดังรูปที่ 3.3 ส่วนวัตถุ B เมื่อได้รับข้อความ m_1 ก็จะส่งข้อความ m_2 ไปให้วัตถุ C และเปลี่ยนสถานะจาก $B1$ ไปเป็น $B2$ ดังรูปที่ 3.4 หลังจากนั้นเมื่อวัตถุ C ได้รับข้อความจะเกิดเหตุการณ์ต่าง ๆ ขึ้นอีกตามที่ได้บรรยายไว้ในแผนภาพสถานะของวัตถุแต่ละอัน โดยสามารถเขียนสรุปเหตุการณ์ที่เกิดขึ้นกับวัตถุที่อยู่ในระบบได้ ดังนี้

$$Actor1 \xrightarrow{e_1} A(A_1, A_2) \xrightarrow{m_1} B(B_1, B_2) \xrightarrow{m_2} C(C_1, C_2) \xrightarrow{\tau} C(C_2, C_3) \xrightarrow{m_3} A(A_2, A_3)$$

เมื่อ τ เป็นเหตุการณ์ที่เกิดขึ้นภายในวัตถุ

จะพบว่าระบบจะหยุดการทำงานชั่วคราวเมื่อสถานะของ A เป็น $A3$ B เป็น $B2$ และ C เป็น $C3$ และวัตถุ C จะต้องได้รับเหตุการณ์ e_2 หรือ e_3 จากผู้ใช้ คือ $Actor2$ ก่อนจึงจะทำงานต่อได้ โดยถ้าวัตถุ C ได้รับเหตุการณ์ e_2 ระบบจะกลับสู่สถานะเริ่มต้นอีกครั้ง ดังนี้

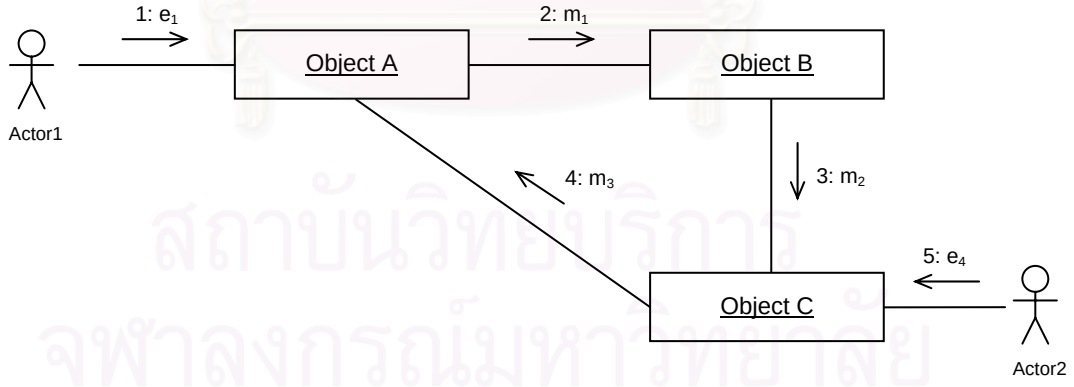
$$Actor2 \xrightarrow{e_2} C(C_3, C_1) \xrightarrow{m_4} A(A_3, A_1)$$
$$\xrightarrow{m_5} B(B_2, B_1)$$

แต่ถ้าวัตถุได้รับเหตุการณ์ e_3 ระบบจะจบการทำงาน (terminate) ดังนี้

$$Actor2 \xrightarrow{e_3} C(C_3, C_4) \xrightarrow{m_6} A(A_3, A_4) \xrightarrow{m_7} B(B_2, B_3)$$

ดังนั้นจากการศึกษาพฤติกรรมของระบบข้างต้นจากแผนภาพสถานะ พบว่าลำดับของเหตุการณ์ที่ระบบรับมาจากผู้ใช้มีรูปแบบเป็น $(e_1e_2)^* e_1e_3$ คือ ระบบเริ่มการทำงานเมื่อได้รับเหตุการณ์ e_1 แล้วจะทำงานไปจนถึงสถานะที่รอรับเหตุการณ์ e_2 หรือ e_3 ถ้าได้รับเหตุการณ์ e_2 จะกลับไปสู่สถานะเริ่มต้น แต่ถ้าได้รับเหตุการณ์ e_3 จะจบการทำงาน

ต่อไปเป็นกรณีที่ระบบมีการบรรยายแผนภาพความร่วมมือเพิ่มเติม ดังรูปที่ 3.6



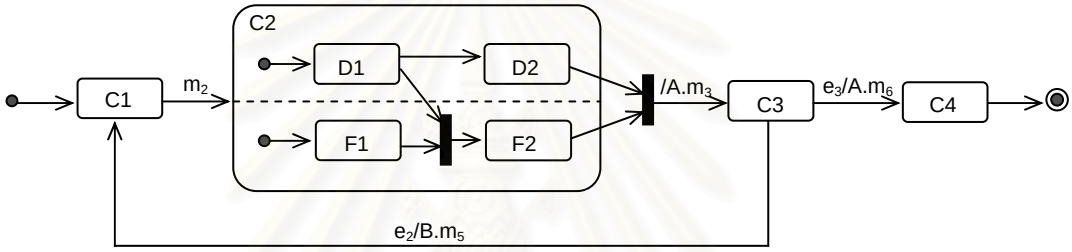
รูปที่ 3.6 แผนภาพความร่วมมือของ Scenario3 ที่เพิ่มในระบบ

จะเห็นว่าหากเกิดลำดับของเหตุการณ์ตามแผนภาพความร่วมมือของ Scenario3 คือ $e_1m_1m_2m_3e_4$ จะเป็นดังนี้

$$Actor1 \xrightarrow{e_1} A(A_1, A_2) \xrightarrow{m_1} B(B_1, B_2) \xrightarrow{m_2} C(C_1, C_2)$$
$$C(C_1, C_2) \xrightarrow{\tau} C(C_2, C_3) \xrightarrow{m_3} A(A_2, A_3) \xrightarrow{e_4} ?$$

แสดงว่ามีเหตุการณ์ที่บรรยายในแผนภาพความร่วมมือแผนภาพนี้ ไม่ถูกบรรยายไว้ในแผนภาพสถานะ เหตุการณ์เช่นนี้ เรียกว่า เกิดความไม่สอดคล้องกันระหว่างแผนภาพทั้งสอง นั่นคือเกิดความขัดแย้งขึ้นภายในระบบ

สุดท้ายในกรณีที่แผนภาพสถานะของวัตถุ C เปลี่ยนไปเป็น ดังรูปที่ 3.7



รูปที่ 3.7 แผนภาพสถานะของวัตถุ C ที่เปลี่ยนการทำงานบางอย่าง

จะพบว่าเมื่อระบบได้รับเหตุการณ์ e_1e_2 จากผู้ใช้ภายนอก ทำให้เกิดลำดับของการส่งข้อความระหว่างวัตถุต่าง ๆ ภายในระบบ ดังนี้

$$Actor1 \xrightarrow{e_1} A(A_1, A_2) \xrightarrow{m_1} B(B_1, B_2) \xrightarrow{m_2} C(C_1, C_2) \xrightarrow{\tau} C(C_2, C_3) \xrightarrow{m_3} A(A_2, A_3)$$
$$Actor2 \xrightarrow{e_2} C(C_3, C_1) \xrightarrow{m_5} B(B_2, B_1)$$

กล่าวคือวัตถุ A อยู่ที่สถานะ A_3 ซึ่งรอข้อความ m_4 จากวัตถุ C ส่วนวัตถุ B อยู่ที่สถานะ B_1 ซึ่งรอข้อความ m_1 จากวัตถุ A และวัตถุ C อยู่ที่สถานะ C_1 ซึ่งรอข้อความ m_2 จากวัตถุ B จะเห็นว่าวัตถุทั้งสามมีการรอคอยข้อความจากกันเป็นวง ลักษณะเช่นนี้ทำให้เกิดเหตุการณ์ที่เรียกว่า “ติดตาย” (deadlock) เกิดขึ้น

จากที่กล่าวมาทำให้ทราบว่าเราสามารถตรวจสอบความต้องกันของแผนภาพความร่วมมือ และแผนภาพสถานะได้ โดยพิจารณาถึงลำดับของข้อความ หรือเหตุการณ์ที่เกิดขึ้นในแผนภาพทั้งสองว่ามีความต้องกันหรือไม่ อย่างไร

3.2 กฎการแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัส

แผนภาพความร่วมมือเป็นแผนภาพที่ใช้บรรยายปฏิสัมพันธ์ระหว่างวัตถุต่าง ๆ ที่มีอยู่ในระบบ ในงานวิจัยนี้ผู้วิจัยได้เสนอให้มีการบรรยายแถวคอยไว้เป็นส่วนประกอบของแผนภาพความร่วมมือโดยใช้เก็บลำดับของข้อความที่ได้บรรยายไว้ในแผนภาพความร่วมมือ เพื่อใช้ตรวจสอบว่าสอดคล้องกับลำดับของเหตุการณ์ที่เกิดขึ้นในแผนภาพสถานะหรือไม่ และจำแนกประเภทข้อความในแผนภาพความร่วมมือเป็น 2 ประเภท คือ ข้อความภายนอกระบบ (external message) และข้อความภายในระบบ (internal message)

โดยการทำงานของแผนภาพความร่วมมือจะพิจารณาข้อความแรกที่อยู่ในแถวคอยว่าเป็นข้อความประเภทใด และมีการทำงาน ดังนี้

1) หากเป็นข้อความภายนอกระบบ ให้พิจารณาว่าข้อความดังกล่าวเกิดจากที่ใด หากเป็นข้อความที่เกิดจากผู้ใช้ภายนอก ให้นำข้อความนั้นไปใส่ไว้ในแถวคอยของแผนภาพสถานะของวัตถุตามที่ได้กำหนดไว้ในแผนภาพความร่วมมือ เพื่อรอการตรวจสอบโดยแผนภาพสถานะต่อไป แต่หากเป็นข้อความที่เกิดจากระบบส่งให้ผู้ใช้ ให้นำข้อความดังกล่าวมารอการตรวจสอบว่าจะเกิดข้อความนั้นจากระบบที่ได้บรรยายไว้ด้วยแผนภาพสถานะหรือไม่

2) หากเป็นข้อความภายในระบบ ให้นำข้อความดังกล่าวมารอการตรวจสอบว่าจะเกิดข้อความนั้นจากระบบที่ได้บรรยายไว้ด้วยแผนภาพสถานะหรือไม่ โดยวิธีพิจารณามี 2 รูปแบบ คือ

- การพิจารณาข้อความที่เกิดแบบไม่พร้อมกัน
- การพิจารณาข้อความที่เกิดแบบพร้อมกัน

ในส่วนต่อไป ผู้วิจัยขอเสนอกฎการแปลงแผนภาพความร่วมมือเป็นไพแคลคูลัส จำนวน 8 ข้อ ดังนี้

กฎข้อที่ 1 ข้อความบนเส้นแสดงความสัมพันธ์จากแผนความร่วมมือ ให้แปลงเป็นช่องสื่อสารในไพแคลคูลัส โดยแบ่งกลุ่มของข้อความข้างต้นออกเป็น 2 กลุ่ม คือ ข้อความภายนอกระบบ ($\vec{e}_{EXT}$) และข้อความภายในระบบ ($\vec{e}_{INT}$)

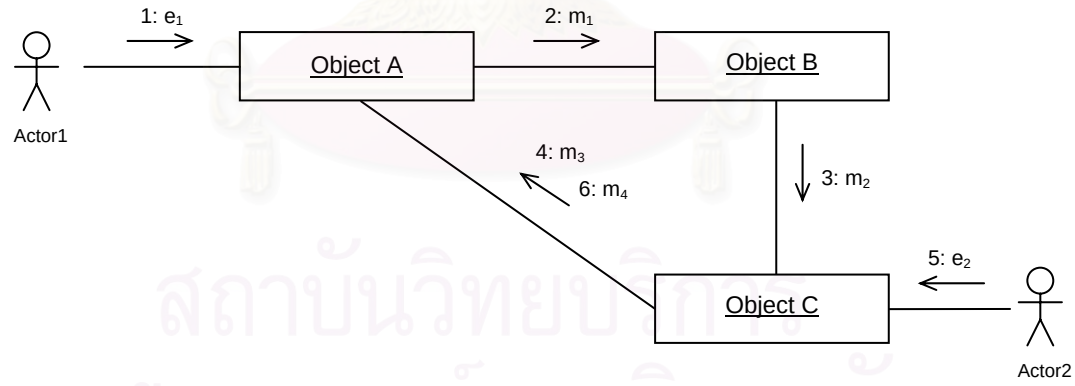
นอกจากนี้ให้นำกลุ่มของข้อความทั้งหมดที่ปรากฏในแผนภาพสถานะ ($\vec{e}_{SYS}$) มาแปลงเป็นช่องสื่อสารในไพแคลคูลัสด้วย

กฎข้อที่ 2 กำหนดแถวคอยของแผนภาพความร่วมมือ ดังนี้

$$Queue_n^F(deq, empty, x_1, x_2, ..., x_n) =$$
$$deq(r).\bar{r}\langle x_1\rangle.Queue_{n-1}^F(deq, empty, x_2, ..., x_n) +$$
$$empty(tf).\bar{f}.Queue_n^F(deq, empty, x_1, ..., x_n)$$
$$Queue_0^F(deq, empty) =$$
$$empty(tf).\bar{t}.Queue_0^F(deq, empty)$$

กฎข้อที่ 3 พิจารณาลำดับของข้อความภายในระบบ แล้วนำไปใส่ไว้ในแถวคอย โดยถ้าเป็นข้อความที่เกิดขึ้นแบบไม่พร้อมกันให้ใส่ข้อความดังกล่าวไว้ในแถวคอยได้ทันที แต่ถ้ากลุ่มข้อความใดเกิดขึ้นแบบพร้อมกันให้ใส่สัญลักษณ์พิเศษ θ และ γ โดยใส่ก่อนและหลังกลุ่มข้อความดังกล่าวตามลำดับ

จากกฎข้อที่ 1 เป็นการแปลงข้อความที่แสดงไว้ในแผนภาพความร่วมมือเป็นช่องสื่อสารในไพแคลคูลัส โดยมีการจำแนกข้อความเป็น 2 กลุ่มดังที่กล่าวข้างต้น ส่วนกฎข้อที่ 2 เป็นการกำหนดแถวคอยของแผนภาพความร่วมมือดังกล่าว และกฎข้อที่ 3 เป็นการนำข้อความที่บรรยายไว้ในแผนภาพความร่วมมือไปใส่ไว้ในแถวคอยข้างต้นอย่างมีลำดับ



รูปที่ 3.8 ตัวอย่างแผนภาพความร่วมมือ

พิจารณาตัวอย่างในรูปที่ 3.8 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้ จากกฎข้อที่ 1 จะได้

$$\vec{e}_{EXT} = e_1, e_2$$
$$\vec{e}_{INT} = m_1, m_2, m_3, m_4$$

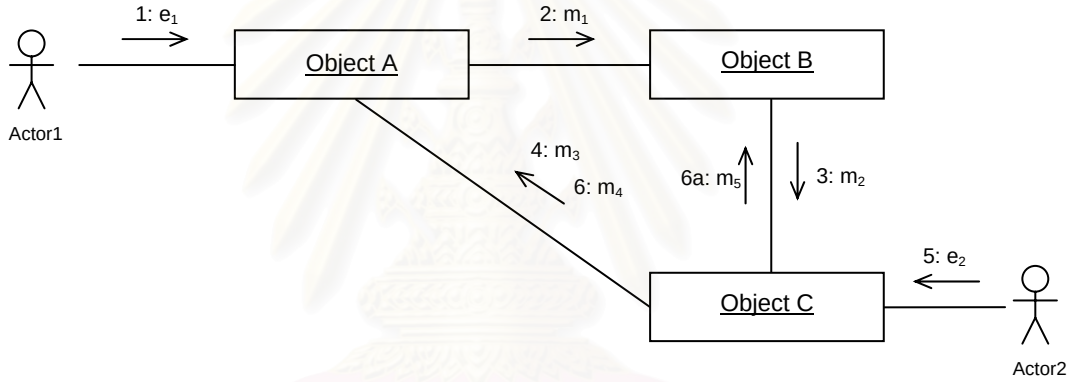
ต่อไปจากกฎข้อที่ 3 พิจารณาลำดับของข้อความที่ได้บรรยายไว้ในรูปที่ 3.8 จะได้ ดังนี้ คือ

$$e_1, m_1, m_2, m_3, e_2, m_4$$

และเมื่อนำข้อความข้างต้นไปใส่ไว้ในแถวคอยที่ได้บรรยายไว้ด้วยกฎข้อที่ 2 จะได้

$$Queue_6^F(deq, empty, e_1, m_1, m_2, m_3, e_2, m_4) =$$
$$deq(r).\bar{r}\langle e_1\rangle.Queue_5^F(deq, empty, m_1, m_2, m_3, e_2, m_4) +$$
$$empty(tf).\bar{f}.Queue_6^F(deq, empty, e_1, m_1, m_2, m_3, e_2, m_4)$$

แต่ถ้าเป็นการบรรยายข้อความที่เกิดขึ้นแบบพร้อมกันสามารถบรรยายได้ ดังตัวอย่างในรูปที่ 3.9



รูปที่ 3.9 ตัวอย่างแผนภาพความร่วมมือ

จากกฎข้อที่ 1 จะได้

$$\vec{e}_{EXT} = e_1, e_2$$
$$\vec{e}_{INT} = m_1, m_2, m_3, m_4, m_5$$

ต่อไปจากกฎข้อที่ 3 พิจารณาลำดับของข้อความที่ได้บรรยายไว้ในรูปที่ 3.9 จะได้ ดังนี้ คือ

$$e_1, m_1, m_2, m_3, e_2, \theta, m_4, m_5, \gamma$$

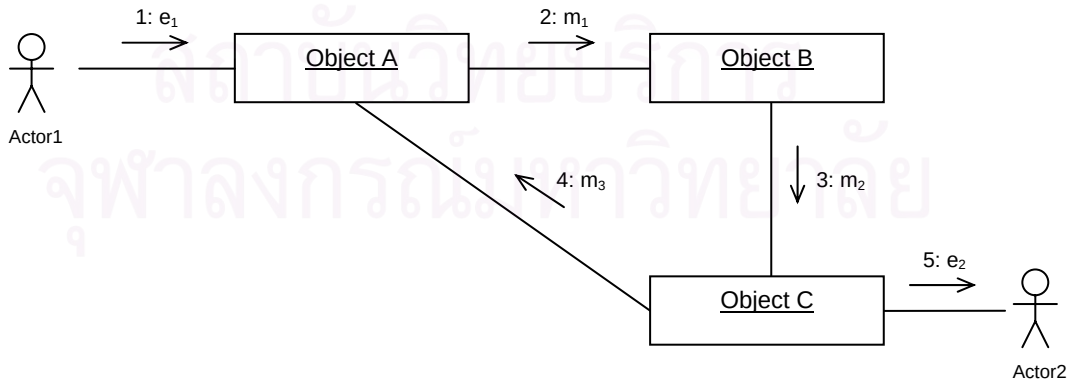
และเมื่อนำข้อความข้างต้นไปใส่ไว้ในแถวคอยที่ได้บรรยายไว้ด้วยกฎข้อที่ 2 จะได้

$$Queue_9^F(deq, empty, e_1, m_1, m_2, m_3, e_2, \theta, m_4, m_5, \gamma) =$$
$$deq(r).\bar{r}\langle e_1\rangle.Queue_8^F(deq, empty, m_1, m_2, m_3, e_2, \theta, m_4, m_5, \gamma) +$$
$$empty(tf).\bar{f}.Queue_9^F(deq, empty, e_1, m_1, m_2, m_3, e_2, \theta, m_4, m_5, \gamma)$$

กฎข้อที่ 4 ตรวจสอบข้อความที่อยู่ในแถวคอย โดยพิจารณาว่ามีข้อความอยู่ในแถวคอยหรือไม่ หากมีให้นำข้อความนั้นมาพิจารณาว่าเป็นข้อความภายนอกระบบ หรือเป็นข้อความภายในระบบ โดยถ้าข้อความนั้นเป็นข้อความภายนอกระบบให้ใช้กฎข้อที่ 5 แต่ถ้าข้อความนั้นเป็นข้อความภายในระบบให้นำข้อความนั้นไปรอการตรวจสอบโดยใช้กฎข้อที่ 6 เนื่องจากเป็นข้อความที่เกิดแบบไม่พร้อมกัน แต่ถ้าพบสัญลักษณ์พิเศษ θ แสดงว่าในแถวคอยมีข้อความที่เกิดแบบพร้อมกันอยู่ให้ตรวจสอบต่อไปโดยใช้กฎข้อที่ 7 และ 8 และถ้าสุดท้ายแถวคอยว่างโดยไม่มีข้อผิดพลาดเกิดขึ้นจากแผนภาพสถานะ แสดงว่าการตรวจสอบเรียบร้อยแล้ว (*Success*) สรุปได้ว่าแผนภาพความร่วมมือ และแผนภาพสถานะสอดคล้องกัน โดยสามารถกำหนดเป็นไพแคลคูลัสได้ ดังนี้

$$Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) = \overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$
$$(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) +$$
$$t.Success$$

กฎข้อที่ 5 พิจารณาข้อความภายนอกระบบว่าข้อความดังกล่าวเกิดจากที่ใด หากเป็นข้อความที่เกิดจากผู้ใช้ภายนอก ให้นำข้อความนั้นไปใส่ไว้ในแถวคอยของแผนภาพสถานะของวัตถุตามที่ได้กำหนดไว้ในแผนภาพความร่วมมือ เพื่อรอการตรวจสอบโดยแผนภาพสถานะต่อไป แต่หากเป็นข้อความที่เกิดจากระบบส่งให้ผู้ใช้ ให้นำข้อความดังกล่าวมารอการตรวจสอบว่าจะเกิดข้อความนั้นจากระบบที่ได้บรรยายไว้ด้วยแผนภาพสถานะหรือไม่



รูปที่ 3.10 ตัวอย่างแผนภาพความร่วมมือ

เช่น จากรูปที่ 3.10 พบว่าเมื่อเหตุการณ์ e_1 ถูกส่งให้ระบบ ระบบจะส่งเหตุการณ์ดังกล่าวให้วัตถุ A ที่อยู่ภายในระบบ แต่ถ้าเป็นเหตุการณ์ e_2 ซึ่งเป็นเหตุการณ์ที่ระบบจะส่งให้ผู้ใช้ ให้นำเหตุการณ์ดังกล่าวมารอการตรวจสอบว่า จะเกิดเหตุการณ์นั้นจากระบบที่ได้บรรยายไว้ด้วยแผนภาพสถานะหรือไม่ โดยสามารถบรรยายด้วยไพแคลคูลัสได้ ดังนี้

$$
\begin{aligned}
&Scenario'(event, ex, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) = \\
&\quad [event = e_1]\overline{enq}_A\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) + \\
&\quad [event = e_2]ex(x). \\
&\qquad ([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) + \\
&\qquad \sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error)
\end{aligned}
$$

กฎข้อที่ 6 นำข้อความภายในระบบมารอการตรวจสอบว่า ข้อความดังกล่าวจะถูกบรรยายไว้ในแผนภาพสถานะด้วยหรือไม่ โดยหากข้อความที่ถูกส่งมาจากแผนภาพสถานะไม่ตรงกับข้อความที่นำมารอการตรวจสอบแสดงว่าเกิดความไม่สอดคล้องกันของลำดับของข้อความที่บรรยายไว้ในแผนภาพความร่วมมือ เช่นจากรูปที่ 3.10 สามารถบรรยายได้ ดังนี้

$$
\begin{aligned}
&Scenario''(event, \overrightarrow{in}, \vec{e}_{INT}, \vec{e}_{SYS}) = \\
&\quad [event = m_1]in_{AB}(x). \\
&\qquad ([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) + \\
&\qquad \sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error) + \\
&\quad [event = m_2]in_{BC}(x). \\
&\qquad ([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) + \\
&\qquad \sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error) + \\
&\quad [event = m_3]in_{CA}(x). \\
&\qquad ([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) + \\
&\qquad \sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error)
\end{aligned}
$$

โดยไพแคลคูลัสข้างต้น หมายความว่า หากข้อความภายในระบบที่ต้องการตรวจสอบ คือ m_1 ให้นำข้อความ m_1 มารอการตรวจสอบว่ามีการส่งข้อความดังกล่าว จากวัตถุ A ไปยังวัตถุ B เกิดขึ้นจากแผนภาพสถานะหรือไม่ โดยถ้าไม่เกิดข้อความดังกล่าวแสดงว่าเกิดข้อขัดแย้งระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะ

กฎข้อที่ 7 นำข้อความที่เกิดขึ้นพร้อมกันภายในระบบมารอการตรวจสอบ โดยดึงข้อความมาจากแถวคอยจนกว่าจะพบสัญลักษณ์พิเศษ γ แสดงว่าหมดช่วงของข้อความที่เกิดขึ้นแบบพร้อมกันแล้ว

$$Scenario_0'''\ (empty, deq, \vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) =$$
$$\overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$
$$Scenario_1'''\ (empty, deq, \vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, event)$$

$$Scenario_n'''\ (empty, deq, \vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, x_1, x_2, ..., x_n) =$$
$$\overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$
$$(\sum_{e\in\{\vec{e}_{INT}\cup\vec{e}_{EXT}\}}[event = e]$$
$$Scenario_{n+1}'''(empty, deq, \vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, x_1, x_2, ..., x_n, event) +$$
$$[event = \gamma](Scenario_n^4(empty, deq, \vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, \overrightarrow{ack}, x_1, x_2, ..., x_n)\ |$$
$$([ack_1 = pos]([ack_2 = pos](...([ack_n = pos]$$
$$Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$$

กฎข้อที่ 8 นำข้อความที่เกิดขึ้นพร้อมกันภายในระบบมาตรวจสอบกับข้อความที่เกิดจากแผนภาพสถานะ โดยถ้าข้อความใดเกิดขึ้นในแผนภาพสถานะแล้วให้เอาออกแล้วรอพิจารณาข้อความที่เหลือ

$$Scenario_n^4(\vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, \overrightarrow{ack}, x_1, x_2, ..., x_n, pos) =$$
$$([x_1 = m_1]in_{AB}(x).$$
$$([x = x_1]\overline{ack}_1\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_1\}}[x = e]Error) +$$
$$[x_1 = m_2]in_{BC}(x).$$
$$([x = x_1]\overline{ack}_1\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_1\}}[x = e]Error) + ...)\ |$$

$$([x_2 = m_1]in_{AB}(x).$$
$$([x = x_2]\overline{ack}_2\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_2\}}[x = e]Error) +$$

$$[x_2 = m_2]in_{BC}(x).$$
$$([x = x_2]\overline{ack}_2\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_2\}}[x = e]Error) + ...) \mid$$
$$([x_3 = m_1]... + [x_3 = m_2]...) \mid (...) \mid$$

$$([x_n = m_1]in_{AB}(x).$$
$$([x = x_n]\overline{ack}_n\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_2\}}[x = e]Error) +$$
$$[x_n = m_2]in_{BC}(x).$$
$$([x = x_n]\overline{ack}_n\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_n\}}[x = e]Error)$$

3.3 กฎการแปลงแผนภาพสถานะไปเป็นไพแคลคูลัส

แผนภาพสถานะเป็นแผนภาพที่ใช้บรรยายถึงพฤติกรรมของวัตถุต่าง ๆ ในระบบ โดยการแปลงแผนภาพสถานะเป็นไพแคลคูลัสที่จะเสนอต่อไปนี้ผู้วิจัยได้พัฒนามาจาก [9] ซึ่งกำหนดให้แผนภาพสถานะของแต่ละวัตถุมีส่วนประกอบหลัก 3 ส่วน คือ แถวคอยของเหตุการณ์ (event queue) โปรแกรมเลือกจ่ายเหตุการณ์ (event dispatcher) และตัวประมวลผลเหตุการณ์ (event processor) แต่อย่างไรก็ตามไพแคลคูลัสที่ได้ยังไม่สามารถตรวจสอบความต้องกันกับแผนภาพความร่วมมือได้

ดังนั้นผู้วิจัยจึงเพิ่มคุณสมบัติบางประการเพื่อให้สามารถตรวจสอบความต้องกันกับแผนภาพความร่วมมือได้ โดยประกอบด้วยกฎรวมทั้งสิ้น 9 ข้อ ดังนี้

กฎข้อที่ 1 เหตุการณ์ (event) ทั้งหมดจากแผนภาพสถานะ ให้แปลงเป็นช่องสื่อสารในไพแคลคูลัส

กฎข้อที่ 2 สถานะในแผนภาพสถานะ ให้แปลงเป็นกระบวนการในไพแคลคูลัส

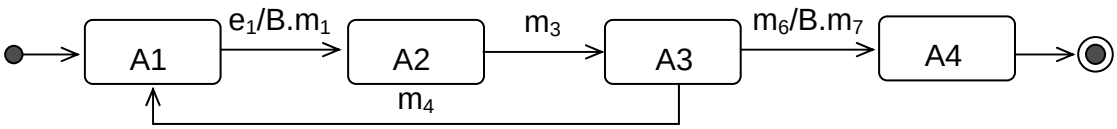
กฎข้อที่ 3 เงื่อนไขในแผนภาพสถานะ สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

$$\overline{g}\langle x\rangle.x(y).([y = true]... + [y = false]...$$

กฎข้อที่ 4 สถานะสถานะทั่วไป สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

$$A\ (step, event_S, \vec{e}_A, \overrightarrow{enq}) =$$
$$event_s(x).([x = e_1]... + [x = e_2]... + ...)$$

พิจารณาตัวอย่างแผนภาพสถานะในรูปที่ 3.11



รูปที่ 3.11 ตัวอย่างแผนภาพสถานะของวัตถุ A

ต่อไปจะแปลงสถานะ A_1 ของวัตถุ A เป็นไพแคลคูลัสได้ ดังนี้

$$A_1(step, event_S, \vec{e}_A, \overrightarrow{enq}) =$$
$$event_s(x).([x = e_1]\overline{enq}_B\langle m_1\rangle.\overline{in}_{AB}\langle m_1\rangle.\overline{step}.A_2(step, event_s, \vec{e}_A) +$$
$$\sum_{e\in\{\vec{e}_A\}\setminus\{e_1\}}[x = e]Error)$$

ความหมายของไพแคลคูลัสข้างต้น คือ ณ สถานะ A_1 หากวัตถุ A ได้รับข้อความ (เหตุการณ์) e_1 จากผู้ใช้แล้ว วัตถุ A จะส่งข้อความ m_1 ไปให้วัตถุ B โดยนำไปใส่ไว้ในแถวคอยของวัตถุ B แล้วจะส่งข้อความดังกล่าวไปตรวจสอบความต้องกันกับแผนภาพสถานะผ่านช่องสื่อสาร $\overline{in}\langle\rangle$ หลังจากนั้นส่งสัญญาณผ่านช่องสื่อสาร $\overline{step}$ ให้ตัวประมวลผลเหตุการณ์ (event processor) นำข้อความถัดไปในแถวคอยจากแผนภาพสถานะของตนเองมาพิจารณาต่อไป แล้วเปลี่ยนสถานะของตนเองเป็น A_2 แต่ถ้าวัตถุ A ได้รับข้อความ อื่น ๆ ที่ไม่ใช่ข้อความ e_1 แสดงว่าเกิดข้อขัดแย้งเกิดขึ้น

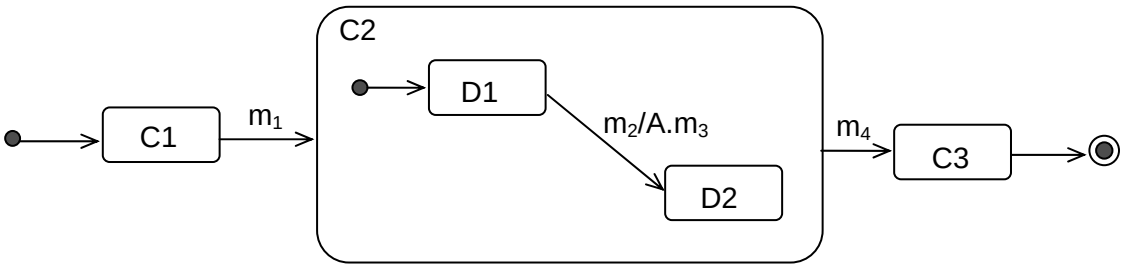
กฎข้อที่ 5 สถานะประกอบแบบสถานะย่อยทำงานไม่พร้อมกัน สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

$$S_1(step, event_S, \vec{e}_A, event_V, pos, neg) =$$
$$event_s(x).(\nu\ ack)$$
$$\overline{event}_V\langle x\ ack\rangle.ack(y).([y = pos]... + [y = neg]...)$$
$$V_1(event_V, \vec{e}_A, pos, neg) =$$
$$event_V(x\ ack).$$
$$([x = e_1]... + [x = e_2]... + ...)$$

กฎข้อที่ 6 สถานะประกอบแบบสถานะย่อยทำงานพร้อมกัน สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

$$S_1(step, event_S, \vec{e}_A, event_{V_1}, event_{V_2}, ..., event_{V_n}, pos, neg, ...) =$$
$$event_s(x).(\nu\ \overrightarrow{ack}).(\overline{event}_{V_1}\langle x\ ack_1\rangle...\overline{event}_{V_n}\langle x\ ack_n\rangle...$$
$$V_i(event_{V_i}, \vec{e}_A, pos, neg, ...) =$$
$$event_{V_i}(x\ ack)...$$
เมื่อ $1 \le i \le n$

พิจารณาตัวอย่างแผนภาพสถานะที่มีสถานะประกอบแบบสถานะย่อยทำงานไม่พร้อมกัน ต่อไปนี้



รูปที่ 3.12 ตัวอย่างแผนภาพสถานะของวัตถุที่มีสถานะประกอบแบบสถานะย่อยทำงานไม่พร้อมกัน

จากรูปที่ 3.12 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

จากกฎข้อที่ 4 จะได้

$$C_1(step, event_S, \vec{e}_A, \overrightarrow{enq}) =$$
$$event_s(x).+([x = m_1]\overline{step}.$$
$$(C_2(step, event_S, \vec{e}_A, event_{V_1}, pos, neg) \mid D_1(event_{V_1}, \vec{e}_A, pos, neg)) +$$
$$\sum_{e \in \{\vec{e}_A\} \setminus \{m_1\}} [x = e] Error)$$

และจากกฎข้อที่ 5 สามารถบรรยายสถานะประกอบ C_2 และสถานะย่อย D_1 และ D_2 ได้ดังนี้

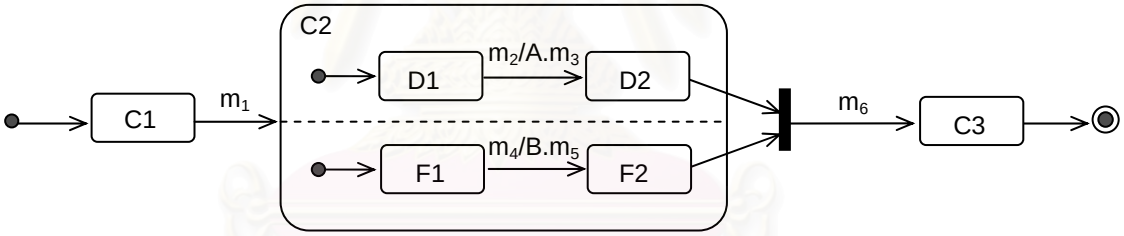
$$C_2(step, event_S, \vec{e}_A, event_{V_1}, pos, neg) =$$
$$event_s(x_1).(\nu\ ack)$$
$$\overline{event}_{V_1}\langle x_1\ ack\rangle.ack(y).([y = pos]([x_1 = m_2]\overline{step}.(\nu\ event_{V_2}).$$
$$(C_2(step, event_S, \vec{e}_A, event_{V_2}, pos, neg) \mid D_2(event_{V_2}, \vec{e}_A, pos, neg)) +$$
$$[x_1 = m_4]\overline{step}.C_3(step, event_S, \vec{e}_A, \overrightarrow{enq})) +$$
$$[y = neg] Error)$$

$$D_1(event_{V_1}, \vec{e}_A, pos, neg) =$$
$$event_{V_1}(x\ ack).$$
$$([x = m_2]\overline{enq}_A\langle m_3\rangle.\overline{in}_{CA}\langle m_3\rangle.\overline{ack}\langle pos\rangle +$$
$$\sum_{e \in \{\vec{e}_A\} \setminus \{m_2\}} [x = e]\overline{ack}\langle neg\rangle.$$
$$D_1(event_{V_1}, \vec{e}_A, pos, neg))$$

$$D_2(event_{V_2}, \vec{e}_A, pos, neg) =$$
$$event_{V_2}(x\ ack).$$
$$([x = m_4]\overline{ack}\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_A\}\setminus\{m_4\}}[x = e]\overline{ack}\langle neg\rangle.$$
$$D_2(event_{V_2}, \vec{e}_A, pos, neg))$$

ความหมายของไพแคลคูลัสข้างต้น คือ หากสถานะประกอบ C_2 ได้รับข้อความใด ๆ ก็ตามจะส่งข้อความดังกล่าวไปให้สถานะย่อยที่ทำงานอยู่ ณ เวลานั้น โดยหากสถานะย่อยตอบรับ (*pos*) ว่าได้รับข้อความดังกล่าวแล้วให้สถานะประกอบเปลี่ยนไปพิจารณาสถานะย่อยถัดไป แต่ถ้าสถานะย่อยตอบปฏิเสธ (*neg*) ข้อความดังกล่าวแล้ว ให้สถานะประกอบอยู่สถานะเดิม

ต่อไปพิจารณาตัวอย่างแผนภาพสถานะที่มีสถานะประกอบแบบสถานะย่อยทำงานพร้อมกัน ต่อไปนี้



รูปที่ 3.13 ตัวอย่างแผนภาพสถานะของวัตถุที่มีสถานะประกอบแบบสถานะย่อยทำงานพร้อมกัน

จากรูปที่ 3.13 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้
จากกฎข้อที่ 4 จะได้

$$C_1(step, event_S, \vec{e}_A, \overrightarrow{enq}) =$$
$$event_s(x).([x = m_1]\overline{step}.$$
$$(C_2(step, event_s, \vec{e}_A) \mid D_1(step, event_s, \vec{e}_A) \mid F_1(step, event_s, \vec{e}_A)) +$$
$$\sum_{e\in\{\vec{e}_A\}\setminus\{m_1\}}[x = e]Error)$$

และจากกฎข้อที่ 6 สามารถบรรยายสถานะประกอบ C_2 และสถานะย่อย D_1 และ F_1 ได้ดังนี้

$$C_2(step, event_S, \vec{e}_A, event_{V_1}, event_{V_2}, pos, neg) =$$
$$event_s(x_1).(\nu\ ack)$$
$$(\overline{event}_{V1}\langle x_1\ ack\rangle.ack(y_1).([y_1 = pos]([x_1 = m_2]$$
$$(C_2(step, event_S, \vec{e}_A, event_{V_1}, event_{V_2}, pos, neg) \mid D_2(event_{V_1}, \vec{e}_A, pos, neg))) +$$
$$[y_1 = neg]C_2(step, event_S, \vec{e}_A, event_{V_1}, event_{V_2}, pos, neg)).$$

$$\overline{event}_{V2}\langle x_1\ ack\rangle.ack(y_2).([y_2 = pos]([x_1 = m_4]$$
$$(C_2(step, event_S, \vec{e}_A, event_{V_1}, event_{V_2}, pos, neg) \mid F_2(event_{V_2}, \vec{e}_A, pos, neg))) +$$
$$[y_2 = neg]C_2(step, event_S, \vec{e}_A, event_{V_1}, event_{V_2}, pos, neg))).$$

$$\overline{step}.([y_1 = pos]([y_2 = pos]([x_1 = m_6]C_3(step, event_S, \vec{e}_A, \overrightarrow{enq})) +$$
$$[y_1 = neg]([y_2 = neg]Error))$$

$$D_1(event_{V_1}, \vec{e}_A, pos, neg) =$$
$$event_{V_1}(x\ ack).$$
$$([x = m_2]\overline{enq}_A\langle m_3\rangle.\overline{in}_{CA}\langle m_3\rangle.\overline{ack}\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_A\}\setminus\{m_2\}}[x = e]\overline{ack}\langle neg\rangle.$$
$$D_1(event_{V_1}, \vec{e}_A, pos, neg))$$

$$F_1(event_{V_2}, \vec{e}_A, pos, neg) =$$
$$event_{V_2}(x\ ack).$$
$$([x = m_4]\overline{enq}_B\langle m_5\rangle.\overline{in}_{CB}\langle m_5\rangle.\overline{ack}\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_A\}\setminus\{m_5\}}[x = e]\overline{ack}\langle neg\rangle.$$
$$F_1(event_V, \vec{e}_A, pos, neg))$$

ความหมายของไพแคลคูลัสข้างต้น คือ หากสถานะประกอบ C_2 ได้รับข้อความใดก็ตามจะกระจายข้อความดังกล่าวไปให้สถานะย่อย ๆ ที่ทำงานอยู่ ณ เวลานั้น โดยหากสถานะย่อยใดตอบรับ (pos) ว่าได้รับข้อความดังกล่าวแล้วให้สถานะประกอบเปลี่ยนไปพิจารณาสถานะย่อย

ถัดไป แต่ถ้าสถานะย่อยใดตอบปฏิเสธ (*neg*) ข้อความดังกล่าวแล้ว ให้สถานะย่อยนั้นอยู่สถานะเดิม

กฎข้อที่ 7 กำหนดแถวคอยของแผนภาพสถานะด้วยไพแคลคูลัสได้ ดังนี้

$$Queue_0^F(enq, deq, empty) =$$
$$enq(x_1).Queue_1^F(enq, deq, empty, x_1) +$$
$$empty(t\ f).\bar{t}.Queue_0^F(enq, deq, empty)$$
$$Queue_n^F(enq, deq, empty, x_1, x_2, ..., x_n) =$$
$$enq(x_{n+1}).Queue_{n+1}^F(enq, deq, empty, x_1, ..., x_n, x_{n+1}) +$$
$$deq(r).\bar{r}\langle x_1\rangle.Queue_{n-1}^F(enq, deq, empty, x_2, ..., x_n) +$$
$$empty(t\ f).\bar{f}.Queue_n^F(enq, deq, empty, x_1, ..., x_n)$$

กฎข้อที่ 8 กำหนดโปรแกรมเลือกจ่ายเหตุการณ์ (event dispatcher) ด้วยไพแคลคูลัสได้ ดังนี้

$$Dispatch^F(t, f, r, deq, empty, execute, complete) =$$
$$\overline{empty}\langle t\ f\rangle.$$
$$t.Dispatch^F(t, f, r, deq, empty, execute, complete) +$$
$$f.\overline{deq}\langle r\rangle.r(event).\overline{execute}\langle event\rangle.complete.$$
$$Dispatch^F(t, f, r, deq, empty, execute, complete))$$

กฎข้อที่ 9 กำหนดตัวประมวลผลเหตุการณ์ S_0 ด้วยไพแคลคูลัสได้ ดังนี้

$$S_0(execute, step, complete, event_s) =$$
$$execute(x).\overline{event}_s\langle x\rangle.step.$$
$$\overline{complete}.S_0(execute, step, complete, event_s)$$

3.4 การประยุกต์ใช้กฎการแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัส

แผนภาพความร่วมมือที่ใช้ในการแปลง ต้องเป็นแผนภาพความร่วมมือแบบที่มีการระบุลำดับของข้อความที่เกิดขึ้นภายในระบบ และหากแผนภาพความร่วมมือใดบรรยายเหตุการณ์ที่มีความต่อเนื่องกัน ให้นำแผนภาพดังกล่าวมาพิจารณาร่วมกันเป็นกรณีเดียว โดยมีขั้นตอนการแปลง ดังนี้

1) นำข้อความทั้งหมดที่แสดงไว้ในแผนภาพความร่วมมือ มาแบ่งออกเป็น 2 กลุ่ม คือ ข้อความภายนอกระบบ และข้อความภายในระบบ จากนั้นนำข้อความทั้ง 2 กลุ่ม มาแปลงเป็นช่องสื่อสารในไพแคลคูลัส โดยใช้กฎข้อที่ 1
2) สร้างแถวคอยที่มีลำดับของข้อความตามที่ได้บรรยายไว้ในแผนภาพความร่วมมือ โดยให้ข้อความแรกอยู่ส่วนต้นสุดของแถวคอย ตามด้วยข้อความลำดับถัด ๆ มา โดยใช้กฎข้อที่ 2 และกฎข้อที่ 3
3) จากกลุ่มของข้อความภายนอกระบบที่ได้จากขั้นตอนที่ 1 ให้พิจารณาข้อความที่เป็นสมาชิกของกลุ่มดังกล่าวทีละข้อความจนครบจำนวนสมาชิก โดยให้ตรวจสอบว่าข้อความนั้นเป็นข้อความภายนอกระบบแบบที่ผู้ใช้ภายนอกส่งให้ระบบ หรือเป็นแบบที่ระบบส่งให้ผู้ใช้ภายนอก จากนั้นแปลงเป็นไพแคลคูลัสโดยใช้กฎข้อ 5 โดยไม่พิจารณาข้อความที่เกิดขึ้นแบบพร้อมกัน
4) จากกลุ่มของข้อความภายในระบบที่ได้จากขั้นตอนที่ 1 ให้พิจารณาข้อความที่เป็นสมาชิกของกลุ่มดังกล่าวทีละข้อความจนครบจำนวนสมาชิก โดยให้ตรวจสอบทิศทางของข้อความนั้นว่าส่งมาจากวัตถุใดและส่งไปยังวัตถุใด จากนั้นแปลงเป็นไพแคลคูลัสโดยใช้กฎข้อที่ 6 โดยไม่พิจารณาข้อความที่เกิดขึ้นแบบพร้อมกัน
5) หากในแผนภาพความร่วมมือมีข้อความที่เกิดแบบพร้อมกัน ให้แปลงเป็นไพแคลคูลัสโดยใช้กฎข้อที่ 7 และกฎข้อที่ 8
6) กำหนดวิธีการนำข้อความออกมาจากแถวคอยที่ได้จากขั้นตอนที่ 2 เพื่อนำไปตรวจสอบความต้องกันกับแผนภาพสถานะ โดยใช้กฎข้อที่ 4

3.5 การประยุกต์ใช้กฎการแปลงแผนภาพสถานะไปเป็นไพแคลคูลัส

แผนภาพสถานะที่จะใช้ในการแปลง ต้องเป็นแผนภาพสถานะแบบที่มีการระบุวัตถุที่จะได้รับข้อความไว้ที่การกระทำ (action) บนการแปลงสถานะ (state transition) โดยมีขั้นตอนการแปลงแผนภาพสถานะของแต่ละวัตถุ ดังนี้

1) นำเหตุการณ์ และการกระทำทั้งหมดที่บรรยายไว้ในแผนภาพสถานะ มาแปลงเป็นช่องสื่อสารในไพแคลคูลัส โดยใช้กฎข้อที่ 1
2) นำสถานะทั้งหมดของวัตถุที่กำลังพิจารณามาแปลงเป็นกระบวนการในไพแคลคูลัส โดยใช้กฎข้อที่ 2 จากนั้นเขียนบรรยายพฤติกรรมของสถานะดังกล่าวทีละสถานะจนครบด้วยไพแคลคูลัส โดยให้พิจารณาว่าเป็นสถานะแบบใด ดังนี้
    2.1) สถานะทั่วไปให้ใช้กฎข้อที่ 4
    2.2) สถานะประกอบแบบสถานะย่อยทำงานไม่พร้อมกันให้ใช้กฎข้อที่ 5
    2.3) สถานะประกอบแบบสถานะย่อยทำงานพร้อมกันให้ใช้กฎข้อที่ 6
3) ข้อความใดบนการแปลงสถานะมีการระบุเงื่อนไข (guard-condition) ให้แปลงเงื่อนไขดังกล่าวไปเป็นไพแคลคูลัส โดยใช้กฎข้อที่ 3
4) สร้างแถวคอย โปรแกรมเลือกจ่ายเหตุการณ์ และตัวประมวลผลเหตุการณ์ของแผนภาพสถานะ โดยใช้กฎข้อที่ 7 8 และ 9 ตามลำดับ

สำหรับตัวอย่างการแปลงแผนภาพความร่วมมือ และแผนภาพสถานะไปเป็นไพแคลคูลัส และการตรวจสอบความต้องกันระหว่างแผนภาพทั้งสองโดยละเอียดนั้นสามารถศึกษาเพิ่มเติมได้จาก ภาคผนวก ก

บทที่ 4

คอมพิวเตอร์เพื่อสนับสนุนการทำงานร่วมกัน

(Computer Supported Cooperative Work: CSCW)

ในบทนี้ ผู้วิจัยจะทดสอบกฎการแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัส และกฎการแปลงแผนภาพสถานะไปเป็นไพแคลคูลัสดังที่ได้อธิบายไว้ในบทที่ 3 ว่าสามารถตรวจสอบความต้องกันระหว่างแผนภาพทั้งสองได้อย่างไร

โดยผู้วิจัยได้เลือกระบบ "คอมพิวเตอร์เพื่อสนับสนุนการทำงานร่วมกัน" มาเป็นกรณีศึกษา เนื่องจากเป็นระบบที่มีการทำงานแบบพร้อมกัน (concurrent system) ที่มีการทำงานค่อนข้างซับซ้อน ซึ่งทำให้ยากต่อการตรวจหาความผิดพลาดต่าง ๆ ของการออกแบบระบบ เช่น การตรวจสอบความต้องกันระหว่างแผนภาพต่าง ๆ ที่ใช้ในการออกแบบระบบ

แผนภาพความร่วมมือ และแผนภาพสถานะของระบบคอมพิวเตอร์เพื่อสนับสนุนการทำงานร่วมกันในบทนี้ ผู้วิจัยได้เลือกมาแสดงเป็นบางแผนภาพตามกรณีที่เกิดขึ้นได้ สำหรับการแปลงแผนภาพความร่วมมือ และแผนภาพสถานะของระบบดังกล่าวไปเป็นไพแคลคูลัส พร้อมทั้งการตรวจสอบความต้องกันระหว่างแผนภาพทั้งสองโดยละเอียดนั้น สามารถศึกษาเพิ่มเติมได้จากภาคผนวก ข

4.1 ความเป็นมาของระบบคอมพิวเตอร์เพื่อสนับสนุนการทำงานร่วมกัน

การพัฒนาซอฟต์แวร์เป็นการทำงานที่จำเป็นต้องอาศัยการทำงานร่วมกัน เนื่องจากเป็นการทำงานที่เกิดขึ้นจากบทบาทของบุคคลหลายคน อาทิเช่น ผู้ใช้ระบบ ลูกค้า ผู้วิเคราะห์ระบบ ผู้ออกแบบระบบ ผู้บริหาร เป็นต้น ดังนั้นการติดต่อสื่อสารระหว่างสมาชิกในทีมผู้พัฒนาจึงเป็นสิ่งที่สำคัญอย่างยิ่ง โดยการติดต่อสื่อสารนั้นอาจจะทำได้โดยการพูดคุยกันโดยตรงหรือผ่านทางเครื่องมือต่าง ๆ ในระหว่างกระบวนการพัฒนานั้นขั้นตอนการวิเคราะห์และออกแบบระบบนั้นนับว่าสำคัญมาก เพราะเป็นส่วนสำคัญที่จะทำให้ได้ซอฟต์แวร์ที่ถูกต้อง (verification) และถูกใจ (validation) ซึ่งมักเกิดปัญหาต่าง ๆ ขึ้นในระหว่างการทำงานดังกล่าว ดังนี้ คือ

ความล่าช้า เนื่องจากหากผู้ออกแบบระบบต้องส่งข้อมูลของแผนภาพ (diagram) ต่าง ๆ ซึ่งตนได้ออกแบบ เป็นไฟล์ผ่านทาง E-mail หรือส่งเป็นเอกสารเพื่อให้สมาชิกคนอื่น ๆ ในทีมตรวจสอบว่าสิ่งที่ตนทำนั้นถูกต้องหรือไม่ หรือมีข้อบกพร่องประการใด บางครั้งอาจต้องรอเป็นเวลานานกว่าจะมีการตอบกลับมา

มีค่าใช้จ่ายสูง เนื่องจากหากผู้ออกแบบระบบต้องจัดทำเอกสารให้สมาชิกคนอื่น ๆ ภายในทีม เพื่อตรวจสอบในกรณีที่ไม่ส่งเป็นไฟล์ข้อมูล จะพบว่าจะต้องจัดทำเอกสารหลายชุด และจัดทำหลายครั้ง

ความไม่คล่องตัว เนื่องจากหากต้องการให้การทำงานเป็นไปอย่างรวดเร็วจะต้องนัดสมาชิกทุกคนมาร่วมกันทำงานในสถานที่เดียวกัน ซึ่งบางครั้งอาจติดปัญหาที่มีสมาชิกบางคนไม่สามารถมาได้เนื่องจากอยู่ในสถานที่ที่ไกลออกไป

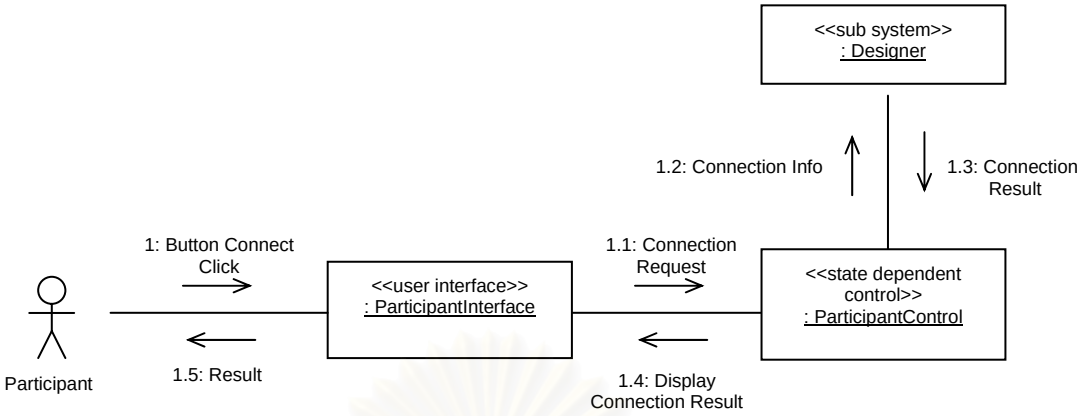
ด้วยเหตุนี้จึงเกิดแนวคิดที่จะสร้างเครื่องมือเพื่อช่วยในการติดต่อสื่อสารระหว่างสมาชิกในทีมผู้พัฒนาขึ้นโดยให้ผู้ออกแบบระบบสามารถส่งแผนภาพต่าง ๆ ที่ตนออกแบบให้สมาชิกคนอื่นในทีมได้ตรวจสอบ และให้ข้อเสนอแนะได้ในทันที เพื่อทำให้การทำงานเป็นไปได้อย่างมีประสิทธิภาพ และสะดวกสบายมากที่สุด

4.2 การแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัส

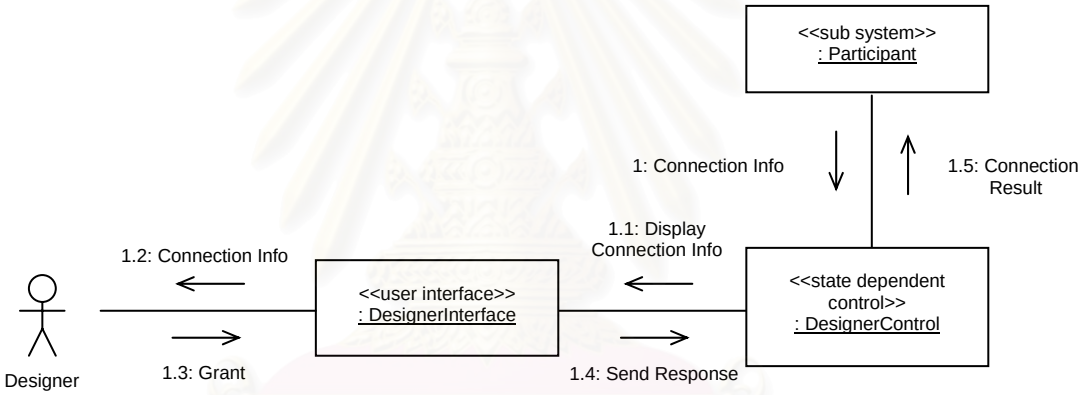
สำหรับการแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัส จะแบ่งการพิจารณาเป็น 2 กรณี ดังนี้ คือ

4.2.1 กรณีที่ไม่มีการส่งข้อความแบบพร้อมกัน

แผนภาพความร่วมมือของระบบคอมพิวเตอร์เพื่อสนับสนุนการทำงานร่วมกัน ที่นำมาเป็นตัวอย่างในการแปลงในหัวข้อนี้ เป็นแผนภาพความร่วมมือของ “ยูสเคสการเข้าร่วมโครงการของผู้ที่มีส่วนร่วม” (CSCW Participant Join Project use case) โดยมีขั้นตอนการทำงาน ดังรูปที่ 4.1 และ 4.2

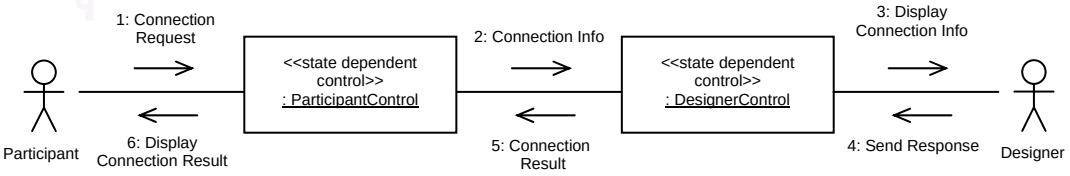


รูปที่ 4.1 แผนภาพความร่วมมือ “Participant Join Project”



รูปที่ 4.2 แผนภาพความร่วมมือ “Designer Grant”

จากรูปที่ 4.1 และ 4.2 จะพบว่าเหตุการณ์ที่บรรยายไว้ด้วยแผนภาพความร่วมมือของทั้ง 2 แผนภาพ มีความต่อเนื่องกัน ดังนั้นให้นำแผนภาพทั้ง 2 มาพิจารณาร่วมกันเป็นกรณีเดียว ดังรูป 4.3



รูปที่ 4.3 แผนภาพความร่วมมือประกอบของ “Participant Join Project” และ “Designer Grant”

จากตัวอย่างในรูปที่ 4.3 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้
ใช้กฎข้อที่ 1 จะได้

$$\vec{e}_{EXT} = connection\_request, display\_connection\_Info, send\_response,$$
$$display\_connection\_result$$
$$\vec{e}_{INT} = connection\_Info, connection\_result$$

ต่อไปจากกฎข้อที่ 3 พิจารณาลำดับของข้อความที่ได้บรรยายไว้ในรูปที่ 4.3 จะได้ ดังนี้ คือ

$$connection\_request, connection\_Info, display\_connection\_Info,$$
$$send\_response, connection\_result, display\_connection\_result$$

และเมื่อนำข้อความข้างต้นไปใส่ไว้ในแถวคอยที่ได้บรรยายไว้ด้วยกฎข้อที่ 2 จะได้

$$Queue_6^F(deq, empty, connection\_request, ..., display\_connection\_result)$$
$$deq(r).\bar{r}\langle connection\_request\rangle.Queue_5^F(deq, empty, connection\_Info, ...) +$$
$$empty(tf).\bar{f}.Queue_6^F(deq, empty, connection\_request, ...)$$

และจากกฎข้อที่ 4 5 และ 6 สามารถแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัสได้ ดังนี้

$$Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) = \overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$
$$(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) +$$
$$t.Success$$

$$Scenario'(event, ex, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) =$$
$$[event = connection\_request]\overline{enq}_{Par}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = send\_response]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = display\_connection\_Info]ex(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error) +$$
$$[event = display\_connection\_result]ex(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error)$$

$$Scenario''(event, \vec{in}, \vec{e}_{INT}, \vec{e}_{SYS}) =$$

$$[event = connection\_Info]in_{ParDes}(x).$$

$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$

$$\sum_{e \in \{\vec{e}_{SYS}\} \setminus \{event\}} [x = e]Error) +$$

$$[event = connection\_result]in_{DesPar}(x).$$

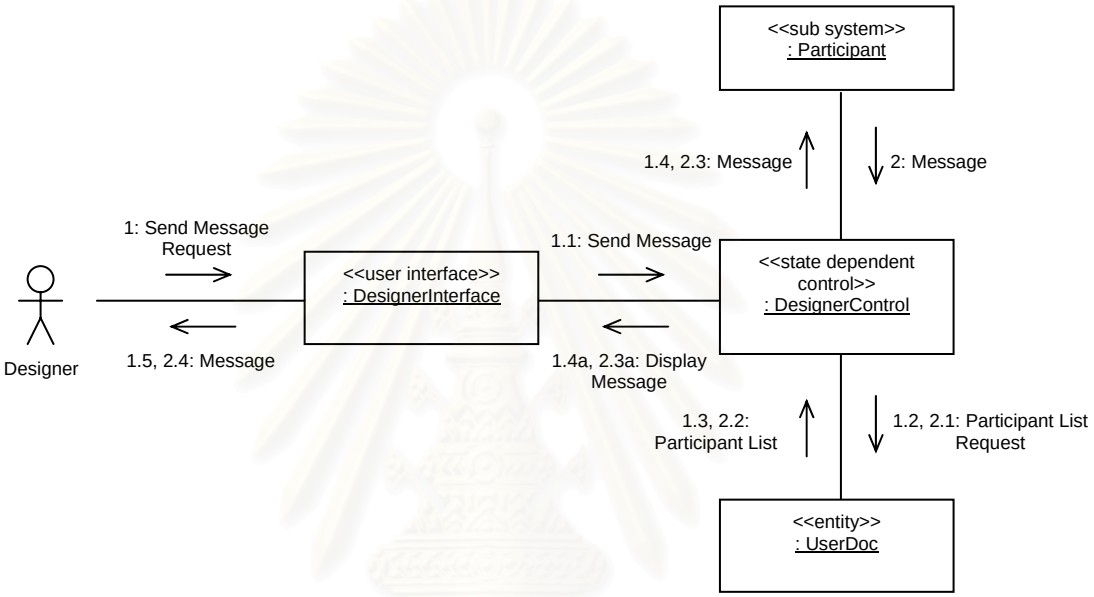
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$

$$\sum_{e \in \{\vec{e}_{SYS}\} \setminus \{event\}} [x = e]Error)$$

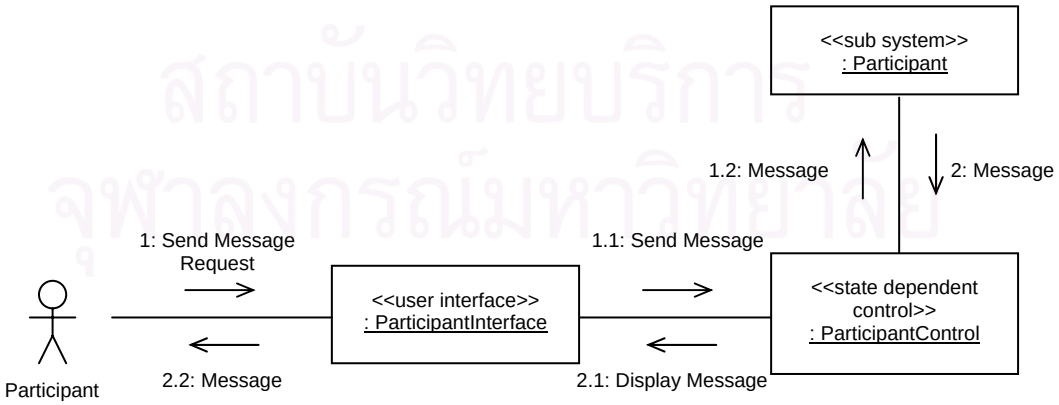
และจากแผนภาพความร่วมมือในรูปที่ 4.3 พบว่าไม่มีการส่งข้อความแบบพร้อมกัน ดังนั้นจึงสามารถละการบรรยาย $Scenario'''$ และ $Scenario^4$ ได้

4.2.2 กรณีที่มีการส่งข้อความแบบพร้อมกัน

ต่อไปเป็นการตัวอย่างการแปลงแผนภาพความร่วมมือ ในกรณีที่มีการส่งข้อความแบบพร้อมกัน โดยแผนภาพความร่วมมือดังกล่าวเป็นแผนภาพความร่วมมือของ “ยูสเคสการสนทนา” (CSCW Chat use case) โดยมีขั้นตอนการทำงาน ดังรูปที่ 4.4 และ 4.5



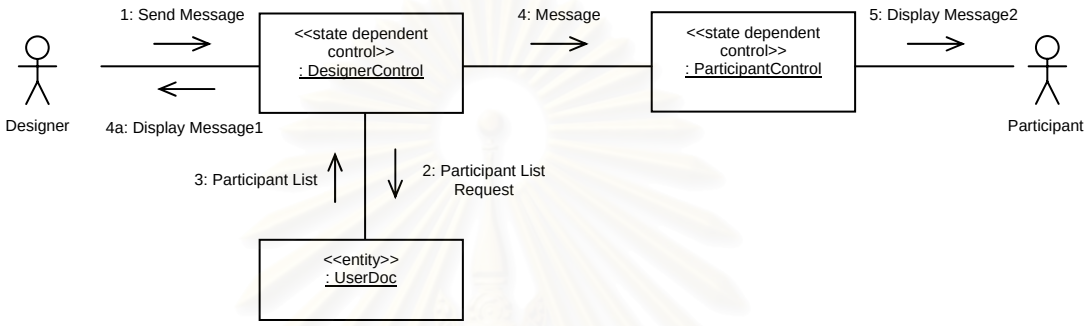
รูปที่ 4.4 แผนภาพความร่วมมือ ”Designer Chat”



รูปที่ 4.5 แผนภาพความร่วมมือ ”Participant Chat”

จากรูปที่ 4.4 และ 4.5 จะพบว่าเหตุการณ์ที่บรรยายไว้ด้วยแผนภาพความร่วมมือของทั้ง 2 แผนภาพ มีความต่อเนื่องกัน ดังนั้นให้นำแผนภาพทั้ง 2 มาพิจารณาร่วมกัน โดยสามารถแบ่งได้เป็น 2 กรณี คือ กรณีที่ Designer ส่งข้อความ และกรณีที่ Participant ส่งข้อความ ดังรูป 4.6 และ 4.7 ตามลำดับ

กรณีที่ Designer ส่งข้อความ



รูปที่ 4.6 แผนภาพความร่วมมือประกอบของ “Participant Chat” และ “Designer Chat”

จากตัวอย่างในรูปที่ 4.6 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้
ใช้กฎข้อที่ 1 จะได้

$$\vec{e}_{EXT} = send\_message, participant\_list\_request, participant\_list, display\_message1, display\_message2$$

$$\vec{e}_{INT} = message$$

ต่อไปพิจารณาลำดับของข้อความที่ได้บรรยายไว้ในรูปที่ 4.6 จะได้ ดังนี้ คือ

$$send\_message, participant\_list\_request, participant\_list, \theta, message, display\_message1, \gamma, display\_message2$$

และจากกฎข้อที่ 4 ถึงกฎข้อที่ 8 สามารถแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัสได้ ดังนี้

$$Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) = \overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$
$$(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) +$$
$$t.Success$$

$$Scenario'(event, ex, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) =$$

$$[event = send\_message]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$

$$[event = participant\_list\_request]ex(x).$$

$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$

$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error) +$$

$$[event = participant\_list]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$

$$[event = display\_message2]ex(x).$$

$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$

$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error)$$

$$Scenario'''_0(empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) =$$

$$\overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$

$$Scenario'''_1(empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, event)$$

$$Scenario'''_n(empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, x_1, x_2, ..., x_n) =$$

$$\overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$

$$(\sum_{e\in\{\vec{e}_{INT}\cup\vec{e}_{EXT}\}}[event = e]$$

$$Scenario'''_{n+1}(empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, x_1, x_2, ..., x_n, event) +$$

$$[event = \gamma](Scenario^4_n(empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, \overrightarrow{ack}, x_1, x_2, ..., x_n)\ |$$

$$([ack_1 = pos]([ack_2 = pos](...([ack_n = pos]$$

$$Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$$

$$Scenario^4_n(\overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, \overrightarrow{ack}, x_1, x_2, ..., x_n, pos) =$$

$$([x_1 = message]in_{DesPar}(x).$$

$$([x = x_1]\overline{ack}_1\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_1\}}[x = e]Error) +$$

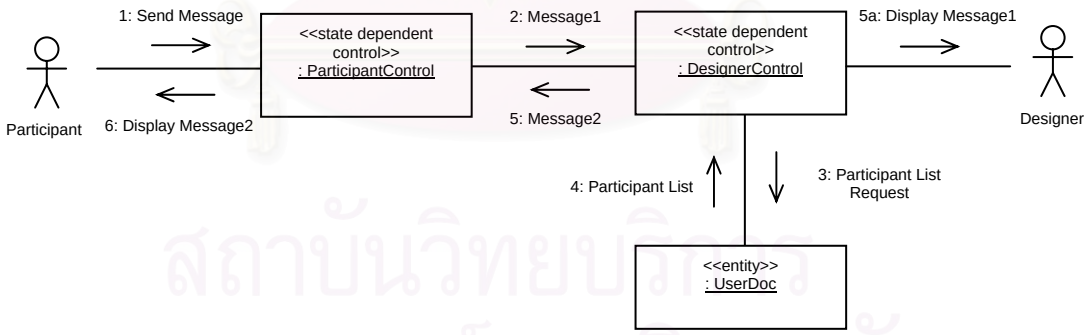
$$[x_1 = display\_message1]ex(x).$$

$$([x = x_1]\overline{ack}_1\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_1\}}[x = e]Error) + ...)\ |$$

$$
\begin{aligned}
&([x_2 = message]in_{DesPar}(x).\\
&\qquad([x = x_2]\overline{ack}_2\langle pos\rangle +\\
&\qquad\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_2\}}[x = e]Error) +\\
&[x_2 = display\_message1]ex(x).\\
&\qquad([x = x_2]\overline{ack}_2\langle pos\rangle +\\
&\qquad\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_2\}}[x = e]Error) + ...) |\\
&([x_3 = message]... + [x_3 = display\_message1]...) | (...) |\\
&\\
&([x_n = message]in_{DesPar}(x).\\
&\qquad([x = x_n]\overline{ack}_n\langle pos\rangle +\\
&\qquad\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_2\}}[x = e]Error) +\\
&[x_n = display\_message1]ex(x).\\
&\qquad([x = x_n]\overline{ack}_n\langle pos\rangle +\\
&\qquad\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_n\}}[x = e]Error)
\end{aligned}
$$

กรณีที่ Participant **ส่งข้อความ**



รูปที่ 4.7 แผนภาพความร่วมมือประกอบของ “Participant Chat” และ “Designer Chat”

จากตัวอย่างในรูปที่ 4.7 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้
ใช้กฎข้อที่ 1 จะได้

$$
\begin{aligned}
\vec{e}_{EXT} = {} & send\_message, participant\_list\_request, participant\_list,\\
& display\_message1, display\_message2
\end{aligned}
$$

$$\vec{e}_{INT} = message1, message2$$

ต่อไปพิจารณาลำดับของข้อความที่ได้บรรยายไว้ในรูปที่ 4.7 จะได้ ดังนี้ คือ

$$send\_message, message1, participant\_list\_request, participant\_list,$$
$$\theta, message2, display\_message1, \gamma, display\_message2$$

และจากกฎข้อที่ 4 ถึงข้อที่ 8 สามารถแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัสได้ ดังนี้

$$Scenario'(event, ex, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) =$$
$$[event = send\_message]\overline{enq}_{Par}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = participant\_list\_request]ex(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error) +$$
$$[event = participant\_list]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = display\_message2]ex(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error)$$

$$Scenario''(event, \overrightarrow{in}, \vec{e}_{INT}, \vec{e}_{SYS}) =$$
$$[event = message1]in_{ParDes}(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error)$$

$$Scenario_0'''(empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) =$$
$$\overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$
$$Scenario_1'''(empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, event)$$

$$Scenario'''_n\ (empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, x_1, x_2, ..., x_n) =$$

$$\overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$

$$(\sum_{e\in\{\vec{e}_{INT}\cup\vec{e}_{EXT}\}}[event = e]$$

$$Scenario'''_{n+1}(empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, x_1, x_2, ..., x_n, event) +$$

$$[event = \gamma](Scenario^4_n(empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, \overrightarrow{ack}, x_1, x_2, ..., x_n)\ |$$

$$([ack_1 = pos]([ack_2 = pos](...([ack_n = pos]$$

$$Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$$

$$Scenario^4_n(\overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, \overrightarrow{ack}, x_1, x_2, ..., x_n, pos) =$$

$$([x_1 = message2]in_{DesPar}(x).$$

$$([x = x_1]\overline{ack}_1\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_1\}}[x = e]Error) +$$

$$[x_1 = display\_message1]ex(x).$$

$$([x = x_1]\overline{ack}_1\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_1\}}[x = e]Error) + ...)\ |$$

$$([x_2 = message2]in_{DesPar}(x).$$

$$([x = x_2]\overline{ack}_2\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_2\}}[x = e]Error) +$$

$$[x_2 = display\_message1]ex(x).$$

$$([x = x_2]\overline{ack}_2\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_2\}}[x = e]Error) + ...)\ |$$

$$([x_3 = message2]... + [x_3 = display\_message1]...)\ |\ (...)\ |$$

$$([x_n = message2]in_{DesPar}(x).$$

$$([x = x_n]\overline{ack}_n\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_2\}}[x = e]Error) +$$

$$[x_n = display\_message1]ex(x).$$

$$([x = x_n]\overline{ack}_n\langle pos\rangle +$$

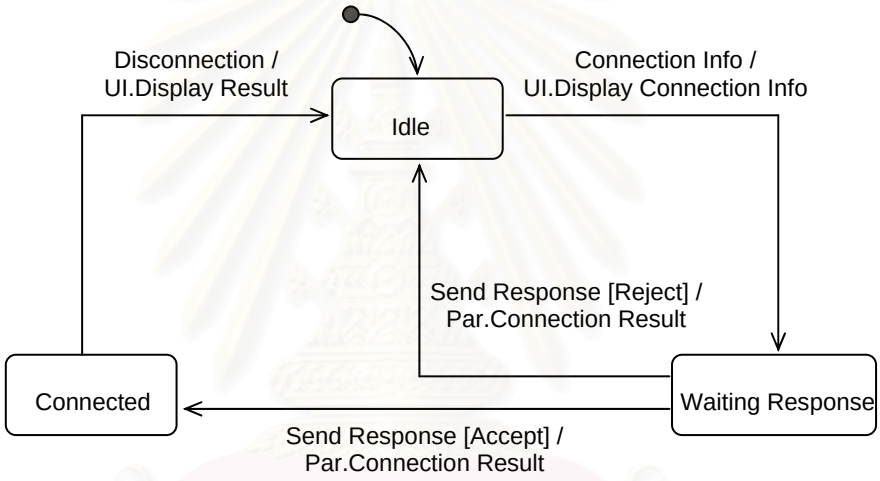
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_n\}}[x = e]Error)$$

4.3 การแปลงแผนภาพสถานะไปเป็นไพแคลคูลัส

สำหรับการแปลงแผนภาพสถานะไปเป็นไพแคลคูลัส จะแบ่งการพิจารณาเป็น 3 กรณี ดังนี้ คือ

4.3.1 กรณีที่เป็นสถานะทั่วไป

ต่อไปเป็นตัวอย่างการแปลงแผนภาพสถานะในกรณีที่เป็นสถานะทั่วไป โดยแผนภาพสถานะที่ใช้เป็นตัวอย่างเป็นแผนภาพสถานะของวัตถุ “Designer Control” ซึ่งพฤติกรรมของวัตถุดังกล่าวถูกบรรยายไว้ในแผนภาพสถานะดังรูปที่ 4.8



รูปที่ 4.8 Top-level statechart for Designer control

จากรูปที่ 4.8 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

ให้ $\vec{e}_{Des} = connection\_Info, display\_connection\_Info$
$send\_response, connection\_result,$
$disconnection, display\_result$

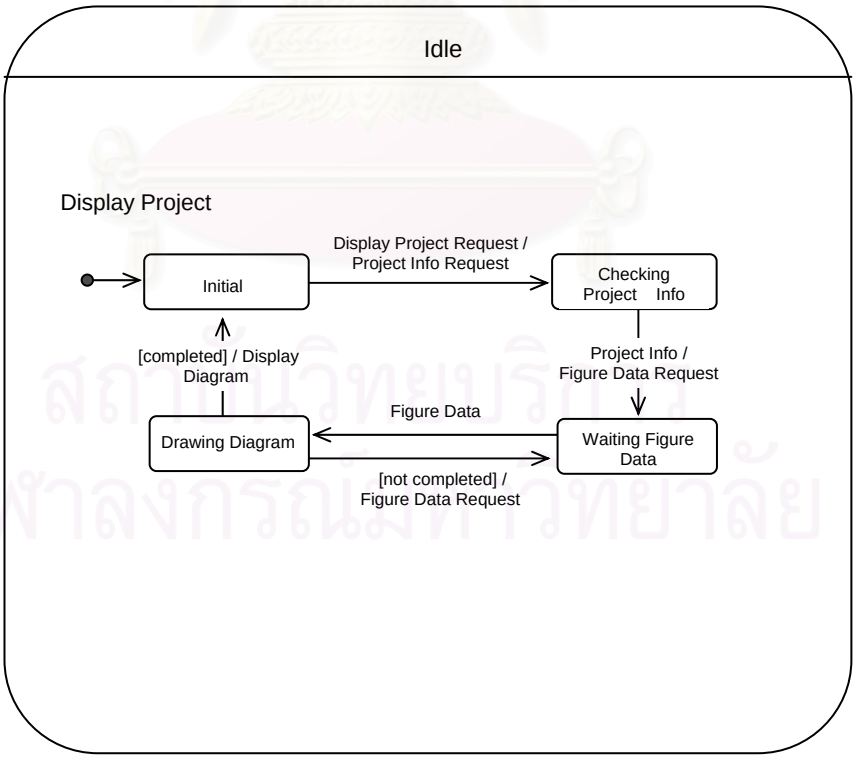
$Idle(step, event_S, \vec{e}_{Des}, \overrightarrow{enq}, ex, \vec{in}) = event_s(x).$
$([x = connection\_Info]\overline{enq}_{Par}\langle display\_connection\_Info\rangle.$
$\overline{ex}\langle display\_connection\_Info\rangle.\overline{step}.$
$Waitng\_response(step, g_1, event_S, \vec{e}_{Des}, \overrightarrow{enq}, ex, \vec{in}) +$
$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{connection\_Info\}}[x = e]Error)$

$$Waiting\_response(step, g_1, event_S, \vec{e}_{Des}, \overrightarrow{enq}, ex, \vec{in}) = event_s(x).$$
$$([x = send\_response]\overline{g}_1\langle z\rangle.z(y).$$
$$([y = accept]\overline{in}_{DesPar}\langle connection\_result\rangle.\overline{enq}_{Par}\langle connection\_result\rangle.$$
$$\overline{step}.Connected(step, event_s, \vec{e}_{Par}, \overrightarrow{enq}, ex, \vec{in}) +$$
$$[y = reject]\overline{in}_{DesPar}\langle connection\_result\rangle.\overline{enq}_{Par}\langle connection\_result\rangle.$$
$$\overline{step}.Idle(step, event_s, \vec{e}_{Par}, \overrightarrow{enq}, ex, \vec{in})) +$$
$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{send\_response\}}[x = e]Error)$$

$$Connected(step, event_S, \vec{e}_{Des}, \overrightarrow{enq}, ex, \vec{in}) = event_s(x).$$
$$([x = disconnection]\overline{ex}\langle display\_result\rangle.\overline{step}.Idle(step, event_S, \vec{e}_{Des}, \overrightarrow{enq}, \vec{in}) +$$
$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{disconnection\}}[x = e]Error)$$

4.3.2 กรณีที่เป็นสถานะประกอบแบบสถานะย่อยทำงานไม่พร้อมกัน

พิจารณาตัวอย่างต่อไปนี้ ซึ่งเป็นแผนภาพสถานะ Idle superstate ของวัตถุ Participant Control



รูปที่ 4.9 แผนภาพสถานะของ Participant Control: Idle superstate

จากรูปที่ 4.9 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

จากกฎข้อที่ 5 สามารถบรรยายสถานะประกอบ *Idle* และสถานะย่อย *Display _ project* ได้ดังนี้

$$
\begin{aligned}
&Idle(step, event_S, \vec{e}_{Par}, event_{V_1}, pos, neg, \overrightarrow{enq}) = \\
&\quad event_s(x_1).(\nu\ ack) \\
&\quad (\overline{event}_{V1}\langle x_1\ ack\rangle.ack(y_1). \\
&\quad ([y_1 = pos](\\
&\qquad [x_1 = display\_project\_request]\overline{step}. \\
&\qquad (Idle(step, event_S, \vec{e}_{Par}, event_{V_1}, pos, neg, \overrightarrow{enq}) \mid \\
&\qquad Checking\_project\_Info(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq})) + \\
&\qquad [x_1 = project\_Info]\overline{step}. \\
&\qquad (Idle(step, event_S, \vec{e}_{Par}, event_{V_1}, pos, neg, \overrightarrow{enq}) \mid \\
&\qquad Waiting\_figure\_data(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq})) + \\
&\qquad [x_1 = figure\_data]\overline{step}. \\
&\qquad (Idle(step, event_S, \vec{e}_{Par}, event_{V_1}, pos, neg, \overrightarrow{enq}) \mid \\
&\qquad Drawing\_diagram(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq})) + \\
&\qquad [x_1 = connection\_request]\overline{step}. \\
&\qquad Waiting\_response(step, g_1, event_S, \vec{e}_{Par}, \overrightarrow{enq})) \\
&\quad [y_1 = neg]Error) \\
&Display\_project(event_{V_1}, \vec{e}_{Par}, pos, neg) = \\
&\quad event_{V_1}(x\ ack). \\
&\quad ([x = display\_project\_request] \\
&\qquad \overline{ex}\langle project\_Info\_request\rangle.\overline{ack}\langle pos\rangle + \\
&\quad \sum_{e \in \{\vec{e}_{Par}\} \setminus \{display\_project\_request\}} [x = e]\overline{ack}\langle neg\rangle. \\
&\qquad Display\_project(event_{V_1}, \vec{e}_{Par}, pos, neg))
\end{aligned}
$$

$$Checking\_project\_Info(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}) =$$

$$event_{V_1}(x\ ack).$$

$$([x = project\_Info]\overline{ex}\langle figure\_data\_request\rangle.\ \overline{ack}\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{Par}\}\setminus\{project\_Info\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Checking\_project\_Info(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}))$$

$$Waiting\_figure\_data(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, g_1) =$$

$$event_{V_1}(x\ ack).$$

$$([x = figure\_data]\ \overline{ack}\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{Par}\}\setminus\{figure\_data\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Waiting\_figure\_data(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}))$$

$$Drawing\_diagram(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, g_1) =$$

$$\overline{g}\langle z\rangle.z(y).$$

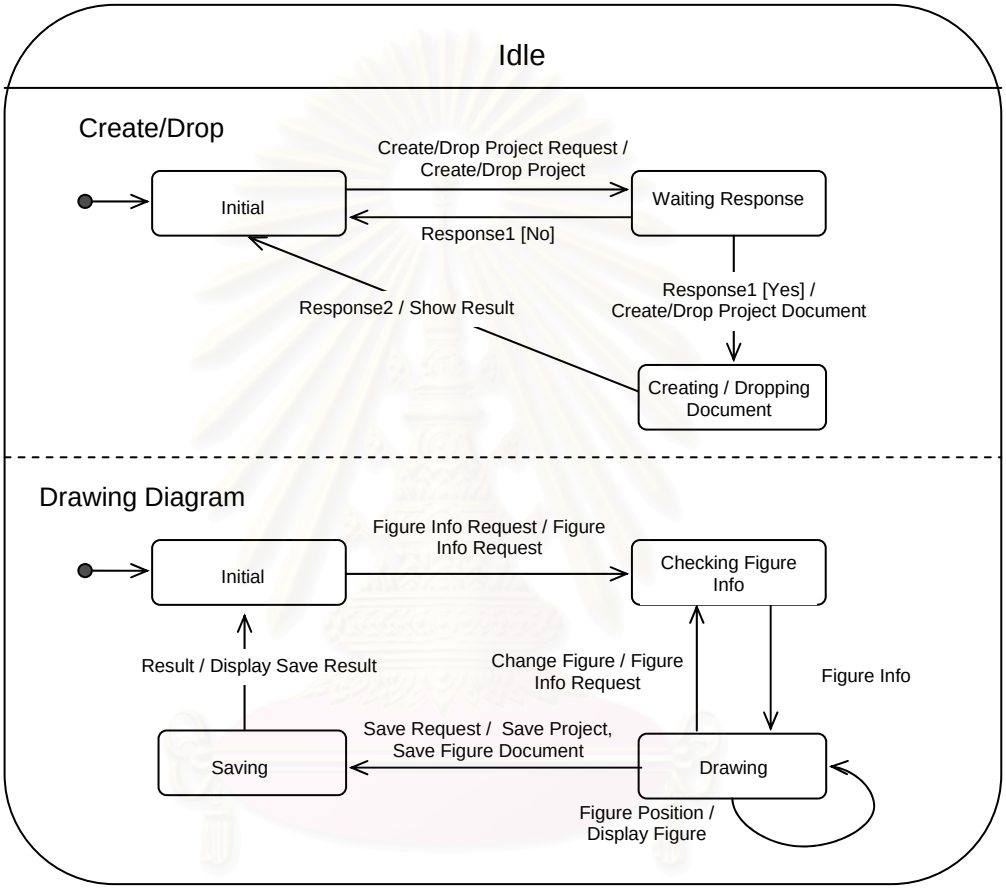
$$([y = completed]\overline{ex}\langle display\_diagram\rangle\ \overline{ack}\langle pos\rangle +$$

$$[y = not\_completed]\overline{ex}\langle figure\_data\_request\rangle\overline{ack}\langle neg\rangle.$$

$$Drawing\_diagram(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, g_1)$$

4.3.3 กรณีที่เป็นสถานะประกอบแบบสถานะย่อยทำงานพร้อมกัน

พิจารณาตัวอย่างต่อไปนี้ ซึ่งเป็นแผนภาพสถานะ Idle superstate ของวัตถุ Designer Control



รูปที่ 4.10 แผนภาพสถานะของ Designer Control: Idle superstate

จากรูปที่ 4.10 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

จากกฎข้อที่ 6 สามารถบรรยายสถานะประกอบ *Idle* และสถานะย่อย $Create\_Drop$ และ $Drawing\_Diagram$ ได้ดังนี้

$$\vec{e}_{Des} = create\_drop\_project\_request, create\_drop\_project$$
$$response1, create\_drop\_project\_doc, response2,$$
$$show\_result$$

$$Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg) =$$

$$event_s(x_1).(\nu\ ack_1\ ack_2)\ (\overline{event}_{V1}\langle x_1\ ack_1\rangle.ack_1(y_1).$$

$$([y_1 = pos]($$

$$[x_1 = create\_drop\_proj\_request]\overline{step}.$$

$$(Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg)\ |$$

$$Waiting\_response(event_{V_1}, \vec{e}_{Des}, pos, neg)) +$$

$$[x_1 = response]\overline{step}.$$

$$(Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg)\ |$$

$$Creating\_Dropping\_Doc(event_{V_1}, \vec{e}_{Des}, pos, neg))) +$$

$$[y_1 = neg]\, Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg)).$$

$$\overline{event}_{V2}\langle x_1\ ack_2\rangle.ack_2(y_2).$$

$$([y_2 = pos]($$

$$[x_1 = figure\_Info\_request1]\overline{step}.$$

$$(Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg)\ |$$

$$Checking\_figure\_Info(event_{V_2}, \vec{e}_{Des}, pos, neg)) +$$

$$[x_1 = figure\_Info]\ \overline{step}.$$

$$(Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg)\ |$$

$$Drawing(event_{V_2}, \vec{e}_{Des}, pos, neg)) +$$

$$[x_1 = figure\_position]\ \overline{step}.$$

$$(Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg)\ |$$

$$Drawing(event_{V_2}, \vec{e}_{Des}, pos, neg)) +$$

$$[x_1 = change\_figure]\ \overline{step}.$$

$$(Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg)\ |$$

$$Checking\_figure\_Info(event_{V_2}, \vec{e}_{Des}, pos, neg)) +$$

$$[x_1 = save\_request]\ \overline{step}.$$
$$(Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg) \mid$$
$$Saving(event_{V_2}, \vec{e}_{Des}, pos, neg)) +$$

$$[x_1 = result]\ \overline{step}.$$
$$(Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg) \mid$$
$$Drawing\_diagram(event_{V_2}, \vec{e}_{Des}, pos, neg))) +$$

$$[y_2 = neg]Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg)).$$

$$\overline{step}.([y_1 = pos]([y_2 = pos]([x_1 = connection\_Info]$$
$$Waiting\_response(step, event_S, \vec{e}_{Des}, \overrightarrow{enq})))$$

$$Create\_Drop(event_{V_1}, \vec{e}_{Des}, pos, neg) =$$
$$event_{V_1}(x\ ack).$$
$$([x = create\_drop\_proj\_request]\overline{in}_{DesPar}\langle create\_drop\_proj\_request\rangle.$$
$$\overline{enq}_{Par}\langle create\_drop\_proj\_request\rangle.\ \overline{ack}\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{create\_drop\_proj\_request\}}[x = e]\overline{ack}\langle neg\rangle.$$
$$Create\_Drop(event_{V_1}, \vec{e}_{Des}, pos, neg))$$

$$Waiting\_response(event_{V_1}, \vec{e}_{Des}, pos, neg) = event_{V_1}(x\ ack).($$
$$[x = response1]($$
$$[y = yes]\overline{ex}\langle create\_drop\_proj\_doc\rangle.\overline{ack}\langle pos\rangle +$$
$$[y = no]\overline{ack}\langle neg\rangle.Waiting\_response(event_{V_1}, \vec{e}_{Des}, pos, neg)) +$$
$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{response1\}}[x = e]\overline{ack}\langle neg\rangle.$$
$$Waiting\_response(event_{V_1}, \vec{e}_{Des}, pos, neg))$$

$$Creating\_Dropping\_Doc(event_{V_1}, \vec{e}_{Des}, pos, neg) =$$

$$event_{V_1}(x\ ack).$$

$$([x = response2]\ \overline{ex}\langle show\_result\rangle.\ \overline{ack}\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{response2\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Creating\_Dropping\_Doc(event_{V_1}, \vec{e}_{Des}, pos, neg))$$

$$Drawing\_diagram(event_{V_2}, \vec{e}_{Des}, pos, neg) =$$

$$event_{V_2}(x\ ack).$$

$$([x = figure\_Info\_request1]\overline{ex}\langle figure\_Info\_request2\rangle.\overline{ack}\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{figure\_Info\_request1\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Drawing\_diagram(event_{V_2}, \vec{e}_{Des}, pos, neg)$$

$$Checking\_figure\_Info(event_{V_2}, \vec{e}_{Des}, pos, neg) =$$

$$event_{V_2}(x\ ack).$$

$$([x = figure\_Info]\ \overline{ack}\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{figure\_Info\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Checking\_figure\_Info(event_{V_2}, \vec{e}_{Des}, pos, neg) =$$

$$Drawing(event_{V_2}, \vec{e}_{Des}, pos, neg) =$$

$$event_{V_2}(x\ ack).($$

$$([x = figure\_position]\overline{ex}\langle display\_figure\rangle.\overline{ack}\langle pos\rangle +$$

$$[x = saving\_request]\overline{ex}\langle save\_project\rangle.\overline{ex}\langle save\_figure\_doc\rangle.\overline{ack}\langle pos\rangle) +$$

$$[x = change\_figure]\overline{ex}\langle figure\_Info\_request\rangle.\overline{ack}\langle pos\rangle) +$$

$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{figure\_position, saving\_request, change\_figure\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Drawing(event_{V_2}, \vec{e}_{Des}, pos, neg))$$

$$Saving(event_{V_2}, \vec{e}_{Des}, pos, neg) =$$

$$event_{V_2}(x\ ack).$$

$$([x = result]\overline{ex}\langle display\_save\_result\rangle.\overline{ack}\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{result\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Saving(event_{V_2}, \vec{e}_{Des}, pos, neg))$$

4.4 การตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะโดยใช้ไพแคลคูลัส

ในขั้นตอนนี้จะตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะโดยใช้ไพแคลคูลัสที่ได้จากการแปลงในหัวข้อที่ 4.2 และ 4.3 โดยจะยกตัวอย่างการตรวจสอบเป็น 2 กรณี คือ กรณีที่ในระบบไม่มีการส่งข้อความแบบพร้อมกัน และกรณีที่ในระบบมีการส่งข้อความแบบพร้อมกัน โดยวัตถุ Participant Control และ Designer Control จะทำงานแบบขนานกัน (parallel)

4.4.1 กรณีที่ในระบบไม่มีการส่งข้อความแบบพร้อมกัน

พิจารณาลำดับของข้อความที่ได้บรรยายไว้ในรูปที่ 4.3 จะได้ ดังนี้ คือ

$connection\_request, connection\_Info, display\_connection\_Info,$
$send\_response, connection\_result, display\_connection\_result$

$$Queue_6^F\,(deq, empty, connection\_request, \quad | \quad Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$$
$$connection\_Info, display\_connection\_Info,$$
$$send\_response, connection\_result,$$
$$display\_connection\_result) \qquad (4.1)$$

$= \quad deq(r).\bar{r}\langle connection\_request\rangle.Queue_5^F(deq, empty, ...) + empty(t\ f).\bar{f}.Queue_6^F(deq, empty, ..) \quad | \quad \overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) + [event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) + t.Success$

$\xrightarrow{\tau} \quad \bar{f}.Queue_6^F(deq, empty, connection\_request, connection\_Info, display\_connection\_Info, send\_response, connection\_result, display\_connection\_result) \quad | \quad f.\overline{deq}\langle y\rangle.y(event).(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) + [event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) + t.Success$

$\xrightarrow{\tau} \quad Queue_6^F(deq, empty, connection\_request, connection\_Info, display\_connection\_Info, send\_response, connection\_result, display\_connection\_result) \quad | \quad \overline{deq}\langle y\rangle.y(event).(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) + [event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$= \quad deq(r).\bar{r}\langle connection\_request\rangle.Queue_5^F(deq, empty, ...) + empty(t\ f).\bar{f}.Queue_6^F(deq, empty, ..) \quad | \quad \overline{deq}\langle y\rangle.y(event).(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$

$$[event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$$

$\xrightarrow{\tau}$ $\bar{y}\langle connection\_request\rangle.Queue_5^F(deq, empty, ...)$ | $y(event).$
$(\sum_{e\in\{\bar{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$
$\sum_{e\in\{\bar{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$
$[event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$\xrightarrow{\tau}$ $Queue_5^F(deq, empty, ...)$ | $(\sum_{e\in\{\bar{e}_{EXT}\}}[connection\_request = e]$
$.Scenario'(connection\_request, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$
$\sum_{e\in\{\bar{e}_{INT}\}}[connection\_request = e]$
$Scenario''(connection\_request, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$
$[connection\_request = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$\xrightarrow{\tau}$ $Queue_5^F(deq, empty, display\_connection\_Info,$ | $Scenario'(connection\_request, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS})$
$send\_response, connection\_result,$
$display\_connection\_result)$

$=$ $deq(r).\bar{r}\langle m_1\rangle.Queue_4^F(deq, empty, ...) +$ | $\overline{enq}_{Par}\langle connection\_request\rangle$
$empty(tf).\bar{f}.Queue_5^F(deq, ...)$ | $.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$

ต่อไปพิจารณาการทำงานของ *Scenario* ที่นำข้อความมาใส่ไว้ในแถวคอยของวัตถุ *Participant Control* ดังนี้

$$Queue_0^{Par}(enq_{Par}, deq_{Par}, empty) \mid Scenario'(connection\_request, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS})$$

$$= enq_{Par}(x_1).Queue_1^{Par}(enq_{Par}, deq_{Par}, empty, x_1) + \mid \overline{enq}_{Par}\langle connection\_request\rangle.$$
$$empty(t\,f).\bar{t}.Queue_0^{Par}(enq_{Par}, deq_{Par}, empty) \quad Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$$

$$\xrightarrow{\tau} Queue_1^{Par}(enq_{Par}, deq_{Par}, empty, connection\_request) \mid Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$$

(4.2) (4.3)

จาก (4.3) พบว่า *Scenario* พร้อมที่จะนำข้อความถัดไปในแถวคอยของแผนภาพความร่วมมือมาพิจารณาต่อเหมือน (4.1) และจาก (4.2) จะพบว่าขณะนี้มีข้อความ *connection_request* อยู่ในแถวคอยของวัตถุ *Participant Control* โดยจะอธิบายการทำงานต่อไป ดังนี้

$$Queue_1^{Par}(enq_{Par}, deq_{Par}, empty, connection\_request) \mid Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete)$$

$$= enq_{Par}(x_2).Queue_2^{Par}(enq_{Par}, deq_{Par}, empty, x_1, x_2) + \mid \overline{empty}\langle t\,f\rangle.$$
$$deq_{Par}(r).\bar{r}\langle connection\_request\rangle. \quad (t.Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete) +$$
$$Queue_0^{Par}(enq_{Par}, deq_{Par}, empty) + \quad f.\overline{deq}_{Par}\langle r\rangle.r(event).\overline{execute}\langle event\rangle.complete.$$
$$empty_{Par}(t\,f).\bar{f}. \quad Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete))$$
$$Queue_1^{Par}(enq_{Par}, deq_{Par}, empty, connection\_request)$$

$$\xrightarrow{\tau} \ \overline{f}.Queue_1^{Par}(enq_{Par}, deq_{Par}, empty, connection\_request) \quad | \quad (t.Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete) +$$
$$f.\overline{deq}_{Par}\langle r\rangle.r(event).\overline{execute}\langle event\rangle.complete.$$
$$Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete))$$

$$\xrightarrow{\tau} \ Queue_1^{Par}(enq_{Par}, deq_{Par}, empty, connection\_request) \quad | \quad \overline{deq}_{Par}\langle r\rangle.r(event).\overline{execute}\langle event\rangle.complete.$$
$$Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete)$$

$$= \ enq_{Par}(x_2).Queue_2^{Par}(enq_{Par}, deq_{Par}, empty, x_1, x_2) + \quad | \quad \overline{deq}_{Par}\langle r\rangle.r(event).\overline{execute}\langle event\rangle.complete.$$
$$deq_{Par}(r).\overline{r}\langle connection\_request\rangle. \qquad Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete)$$
$$Queue_0^{Par}(enq_{Par}, deq_{Par}, empty) +$$
$$empty_{Par}(t f).\overline{f}.$$
$$Queue_1^{Par}(enq_{Par}, deq_{Par}, empty, connection\_request)$$

$$\xrightarrow{\tau} \ \overline{r}\langle connection\_request\rangle.Queue_0^{Par}(enq_{Par}, deq_{Par}, empty) \quad | \quad r(event).\overline{execute}\langle event\rangle.complete.$$
$$Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete)$$

$$\xrightarrow{\tau} \ Queue_0^{Par}(enq_{Par}, deq_{Par}, empty) \quad | \quad \overline{execute}\langle connection\_request\rangle.complete.$$
$$Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete)$$

จากนิพจน์ข้างต้นเป็นการทำงานของวัตถุ *Participant Control* ที่นำข้อความออกมาจากแถวคอยด้วยโปรแกรมเลือกจ่ายเหตุการณ์

ต่อไปเป็นการทำงานระหว่างตัวประมวลผลเหตุการณ์และโปรแกรมเลือกจ่ายเหตุการณ์

$$
\begin{aligned}
& Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete) \mid S_0(execute, step, complete, event_s) \\
= \; & \overline{execute}\langle connection\_request\rangle.complete. \mid execute(x).\overline{event}_s\langle x\rangle.step. \\
& Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete) \quad \overline{complete}.S_0(execute, step, complete, event_s) \\
\xrightarrow{\tau} \; & complete. \mid \overline{event}_s\langle connection\_request\rangle.step. \\
& Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete) \quad \overline{complete}.S_0(execute, step, complete, event_s)
\end{aligned}
$$

เมื่อตัวประมวลผลเหตุการณ์ได้รับข้อความจากโปรแกรมเลือกจ่ายเหตุการณ์จะนำข้อความที่ได้ไปให้วัตถุ *Participant Control* ดังนี้

$$
\begin{aligned}
& S_0(execute, step, complete, event_s) \mid Idle(step, event_S, \vec{e}_{Des}, \overrightarrow{enq}) \\
= \; & \overline{event}_s\langle connection\_request\rangle.step. \mid event_s(x). \\
& \overline{complete}.S_0(execute, step, complete, event_s) \quad ([x = connection\_request]\overline{enq}_{Des}\langle connection\_Info\rangle. \\
& \quad \overline{in}_{ParDes}\langle connection\_Info\rangle.\overline{step}. \\
& \quad Waitng\_response(step, event_s, \vec{e}_{Des}, \overrightarrow{enq}) + \\
& \quad \Sigma_{e\in\{\vec{e}_{Des}\}\setminus\{connection\_request\}}[x = e]Error) \\
\xrightarrow{\tau} \; & step. \mid \overline{enq}_{Des}\langle connection\_Info\rangle.\ \overline{in}_{ParDes}\langle connection\_Info\rangle.\overline{step}. \\
& \overline{complete}.S_0(execute, step, complete, event_s) \quad Waitng\_response(step, event_s, \vec{e}_{Des}, \overrightarrow{enq}) +
\end{aligned}
$$

จะพบว่าเมื่อวัตถุ *Participant Control* ได้รับข้อความ $connection\_request$ จะส่งข้อความ $connection\_Info$ ไปให้วัตถุ *Designer Control* โดยนำข้อความดังกล่าวไปใส่ไว้ในแถวคอยของวัตถุดังกล่าว ดังนี้

$$
\begin{aligned}
& Queue_0^{Par}(enq, deq, empty) && | \quad Idle(step, event_S, \vec{e}_{Des}, \overrightarrow{enq}) \\
= \; & enq_{Des}(x_1).Queue_1^{Des}(enq_{Des}, deq_{Des}, empty, x_1) + && | \quad \overline{enq}_{Des}\langle connection\_Info\rangle.\ \overline{in}_{ParDes}\langle connection\_Info\rangle.\overline{step}. \\
& empty(tf).\bar{t}.Queue_0^{Des}(enq_{Des}, deq_{Des}, empty) && \quad Waitng\_response(step, event_s, \vec{e}_{Des}, \overrightarrow{enq}) \\
\xrightarrow{\tau} \; & Queue_1^{Des}(enq, deq, empty, connection\_Info) && | \quad \overline{in}_{ParDes}\langle connection\_Info\rangle.\overline{step}. \\
& && \quad Waitng\_response(step, event_s, \vec{e}_{Des}, \overrightarrow{enq})
\end{aligned}
$$

(4.4) (4.5)

นอกจากวัตถุ *Participant Control* จะส่งข้อความ $connection\_Info$ ไปให้วัตถุ *Designer Control* แล้ว ยังส่งข้อความดังกล่าวไปตรวจสอบความต้องกันกับแผนภาพความร่วมมือผ่านทาง *Scenario″* อีกด้วย

$$
\begin{aligned}
& Scenario''(connection\_Info, \overrightarrow{in}, \vec{e}_{INT}, \vec{e}_{SYS}) = && | \quad Idle(step, event_S, \vec{e}_{Des}, \overrightarrow{enq}) \\
= \; & in_{ParDes}(x). && | \quad \overline{in}_{ParDes}\langle connection\_Info\rangle.\overline{step}. \\
& ([x = connection\_Info] && \quad Waitng\_response(step, event_s, \vec{e}_{Des}, \overrightarrow{enq}) \\
& \quad Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) + \\
& \quad \sum_{e \in \{\vec{e}_{SYS}\} \setminus \{connection\_Info\}} [x = e] Error)
\end{aligned}
$$

$\xrightarrow{\tau}$ $Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$ | $\overline{step}.Waitng\_response(step, event_s, \vec{e}_{Des}, \overrightarrow{enq})$

ทำคล้าย (4.1) ไปเรื่อย ๆ จนไม่มีข้อความในแถวคอยของแผนภาพความร่วมมือแล้ว หากไม่พบข้อผิดพลาดใด ๆ จากวัตถุอื่นที่บรรยายไว้ด้วยแผนภาพสถานะ แสดงว่าแผนภาพความร่วมมือและแผนภาพสถานะข้างต้นมีความต้องกัน

4.4.2 กรณีที่ในระบบมีการส่งข้อความแบบพร้อมกัน

พิจารณาลำดับของข้อความที่ได้บรรยายไว้ในรูปที่ 4.6 จะได้ ดังนี้ คือ

$send\_message, participant\_list\_request, participant\_list,$
$\theta, message, display\_message1, \gamma, display\_message2$

$Queue_8^F(deq, empty, send\_message,$ | $Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$
$participant\_list\_request, participant\_list, \theta, message,$
$display\_message1, \gamma, display\_message2)$ (4.6)

$= deq(r).\bar{r}\langle send\_message\rangle.Queue_7^F(deq, empty, ...) +$ | $\overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$
$empty(t\ f).\bar{f}.Queue_8^F(deq, empty, ..)$ $(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$
$\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$
$[event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) +$
$t.Success$

$$\xrightarrow{\tau} \ \overline{f}.Queue_8^F\,(deq, empty, send\_message,\ |\ f.\overline{deq}\langle y\rangle.y(event).$$
$$participant\_list\_request,\ participant\_list, \theta, message,$$
$$(\textstyle\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'\,(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$$
$$display\_message1, \gamma, display\_message2)$$
$$\textstyle\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) +$$
$$t.Success$$

$$\xrightarrow{\tau} \ Queue_8^F\,(deq, empty, send\_message,\ |\ \overline{deq}\langle y\rangle.y(event).$$
$$participant\_list\_request,\ participant\_list, \theta, message,$$
$$(\textstyle\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'\,(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$$
$$display\_message1, \gamma, display\_message2)$$
$$\textstyle\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$$

$$= \ deq(r).\overline{r}\langle send\_message\rangle.Queue_7^F\,(deq, empty, ...) +\ |\ \overline{deq}\langle y\rangle.y(event).$$
$$empty(t\,f).\overline{f}.Queue_8^F\,(deq, empty, ..)$$
$$(\textstyle\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'\,(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$$
$$\textstyle\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$$

$$\xrightarrow{\tau} \ \overline{y}\langle send\_message\rangle.Queue_7^F\,(deq, empty, ...)\ |\ y(event).$$
$$(\textstyle\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'\,(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$$
$$\textstyle\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$$

$\xrightarrow{\tau}$ $Queue_7^F(deq, empty, ...)$ | $(\sum_{e\in\{\vec{e}_{EXT}\}}[send\_message = e]$
$.Scenario'(sent\_message, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$
$\sum_{e\in\{\vec{e}_{INT}\}}[send\_message = e]$
$Scenario''(send\_message, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$
$[send\_message = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$\xrightarrow{\tau}$ $Queue_7^F(deq, empty, participant\_list\_request,$ | $Scenario'(send\_message, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS})$
$participant\_list, \theta, message, display\_message1, \gamma,$
$display\_message2)$

$=$ $deq(r).\bar{r}\langle participant\_list\_request\rangle.$ | $\overline{enq}_{Des}\langle event\rangle.\ Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$
$Queue_6^F(deq, empty, ...) +$
$empty(tf).\bar{f}.Queue_7^F(deq, ...)$

ต่อไปพิจารณาการทำงานของ *Scenario* ที่นำข้อความมาใส่ไว้ในแถวคอยของวัตถุ *Designer Control* ดังนี้

$Queue_0^{Des}(enq_{Des}, deq_{Des}, empty)$ | $Scenario'(send\_message, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS})$

$=$ $enq_{Des}(x_1).Queue_1^{Des}(enq_{Des}, deq_{Des}, empty, x_1) +$ | $\overline{enq}_{Des}\langle send\_message\rangle.\ Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$
$empty(tf).\bar{t}.Queue_1^{Des}(enq_{Par}, deq_{Par}, empty)$

$\xrightarrow{\tau}$ $Queue_1^{Des}(enq_{Par}, deq_{Par}, empty, connection\_request)$ | $Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$

(4.7) (4.8)

จาก (4.8) พบว่า *Scenario* พร้อมที่จะนำข้อความถัดไปในแถวคอยของแผนภาพความร่วมมือมาพิจารณาต่อเหมือน (4.6) และจาก (4.7) จะพบว่าขณะนี้มีข้อความ *send _ message* อยู่ในแถวคอยของวัตถุ *Designer Control*

ต่อไปเมื่อนำข้อความจากแถวคอยของแผนภาพความร่วมมือมาพิจารณาจนกระทั่งถึงข้อความที่เกิดขึ้นแบบพร้อมกัน จะมีวิธีการตรวจสอบ ดังนี้

$\xrightarrow{\tau}$ $Queue_5^F(deq, empty, \theta, message, display\_message1, \gamma, display\_message2)$ | $\overline{deq}\langle y\rangle.y(event).(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) + [event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$= deq(r).\bar{r}\langle\theta\rangle.Queue_4^F(deq, empty, ...) + empty(tf).\bar{f}.Queue_5^F(deq, empty, ..)$ | $\overline{deq}\langle y\rangle.y(event).(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) + [event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$\xrightarrow{\tau}$ $\bar{y}\langle\theta\rangle.Queue_4^F(deq, empty, ...)$ | $y(event).$
$(\sum_{e\in\{\bar{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$
$\sum_{e\in\{\bar{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$
$[event = \theta]Scenario_0'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$\xrightarrow{\tau}$ $Queue_4^F(deq, empty, message,$ | $Scenario_0'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$
$display\_message1, \gamma, display\_message2)$

$=$ $deq(r).\bar{r}\langle message\rangle.Queue_3^F(deq, empty, ...) +$ | $\overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$
$empty(t\ f).\bar{f}.Queue_4^F(deq, empty, ..)$ $Scenario_1'''(empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, event)$

$\xrightarrow{\tau}$ $\bar{f}.Queue_4^F(deq, empty, ..)$ | $f.\overline{deq}\langle y\rangle.y(event).$
$Scenario_1'''(empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, event)$

$\xrightarrow{\tau}$ $Queue_4^F(deq, empty, message,$ | $\overline{deq}\langle y\rangle.y(event).$
$display\_message1, \gamma, display\_message2)$ $Scenario_1'''(empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, event)$

$=$ $deq(r).\bar{r}\langle message\rangle.Queue_3^F(deq, empty, ...) +$ | $\overline{deq}\langle y\rangle.y(event).$
$empty(t\ f).\bar{f}.Queue_4^F(deq, empty, ..)$ $Scenario_1'''(empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, event)$

$\xrightarrow{\tau}$ $\bar{y}\langle message\rangle.Queue_3^F(deq, empty, ...)$ | $y(event).Scenario_1'''(empty, deq, \vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, event)$

$\xrightarrow{\tau}$ $Queue_3^F(deq, empty, display\_message1, \gamma,$ | $Scenario_1'''(empty, deq, \vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, message)$
$display\_message2)$

จะพบว่าเมื่อนำข้อความจากแถวคอยของแผนภาพความร่วมมือมาใส่ไว้ในกระบวนการ $Scenario'''$ จนกระทั่งถึงข้อความ γ ขั้นตอนต่อไปจะมีวิธีการตรวจสอบ ดังนี้

$\xrightarrow{\tau}$ $\bar{y}\langle\gamma\rangle.Queue_1^F(deq, empty, display\_message2)$ | $Scenario_2^4(\vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, \overrightarrow{ack}, message,$
$([ack_1 = pos]([ack_2 = pos]$
$Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$

$\xrightarrow{\tau}$ $Queue_1^F(deq, empty, display\_message2)$ | $Scenario_2^4(\vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, \overrightarrow{ack}, message,$
$([ack_1 = pos]([ack_2 = pos]$
$Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$

$=$ $deq(r).\bar{r}\langle display\_message2\rangle.Queue_0^F(deq, empty, ...) +$ | $in_{DesPar}(x).([x = message]\overline{ack}_1\langle pos\rangle +$
$empty(tf).\bar{f}.Queue_0^F(deq, empty)$ $\sum_{e\in\{\bar{e}_{SYS}\}\setminus\{message\}}[x = e]Error)$ |

$$ex(x).([x = display\_message1]\overline{ack}_2\langle pos\rangle + \sum_{e\in\{\vec{e}_{SYS}\}\setminus\{display\_message1\}}[x = e]Error)$$

จากการทดสองพบว่าเกิดข้อความ *message* และ *display _ message*1 ขึ้น และไม่พบข้อผิดพลาดใด ๆ จากวัตถุอื่นที่บรรยายไว้ด้วยแผนภาพสถานะ แสดงว่าแผนภาพความร่วมมือและแผนภาพสถานะข้างต้นมีความต้องกัน

4.5 ผลการตรวจสอบ

จากการทดสอบกับ “ระบบคอมพิวเตอร์เพื่อสนับสนุนการทำงานร่วมกัน” ดังที่ได้กล่าวมาแล้ว พบว่าการตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะสามารถทำได้โดยการนำนิพจน์ในรูปแบบที่เป็นไพแคลคูลัส ที่ได้จากการใช้กฎการแปลงแผนภาพทั้งสองไปเป็นไพแคลคูลัส โดยจากการตรวจสอบไม่พบว่าเกิดข้อขัดแย้งใด ๆ ระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะที่ใช้บรรยายพฤติกรรมของระบบดังกล่าว

บทที่ 5

สรุปผลการวิจัย

5.1 สรุปผลการวิจัย

จากการทดสอบกับกรณีศึกษาดังที่กล่าวมาแล้วพบว่า การตรวจสอบความต้องกันระหว่างแผนภาพทั้งสองสามารถทำได้ โดยใช้กฎการแปลงแผนภาพความร่วมมือเป็นไพแคลคูลัส และกฎการแปลงแผนภาพความสถานะเป็นไพแคลคูลัส จากนั้นใช้ความเป็นรูปนัยของไพแคลคูลัสตรวจสอบความต้องกันระหว่างแผนภาพทั้งสอง

5.2 ปัญหาและข้อจำกัดที่พบจากการวิจัย

5.2.1 การตรวจสอบจะไม่สามารถจะทำได้ในกรณีที่ไม่ได้ระบุแผนภาพสถานะของวัตถุบางวัตถุไว้

5.2.2 กรณีที่เกิดความผิดพลาด เนื่องจากการระบุชื่อของข้อความเดียวกันบนแผนภาพทั้งสองไม่ตรงกันจะไม่สามารถตรวจสอบได้

5.3 ข้อเสนอแนะ

5.3.1 การตรวจสอบความต้องกันระหว่างแผนภาพอื่น ๆ เช่น แผนภาพลำดับ และแผนภาพความร่วมมือ ของยูเอ็มแอล เป็นกรณีที่น่าสนใจ

5.3.2 การสร้างโปรแกรมคอมพิวเตอร์เพื่อใช้แปลงแผนภาพความร่วมมือ และแผนภาพสถานะไปเป็นไพแคลคูลัส และตรวจสอบความต้องกันของไพแคลคูลัสที่ได้จากแผนภาพทั้งสองอย่างอัตโนมัติ เป็นสิ่งที่น่าสนใจ

รายการอ้างอิง

[1] Booch, G., Rumbaugh, J., and Jacobson, I. The Unified Modeling Language reference manual. Reading, MC: Addison-Wesley, 1998.

[2] OMG Unified Modeling Language specification version 1.5 [online]. 2003. Available from: http://www.omg.org/technology/documents/formal/uml.htm [2004, August 15]

[3] Gomaa, H. Designing concurrent, distributed, and real-time applications with UML. 2 nd ed. Reading, MC: Addison-Wesley, 2001.

[4] Milner, R. Communicating and mobile systems: The π-Calculus. Cambridge: Cambridge University Press, 1999.

[5] Milner, R., Parrow, J., and Walker, D. A calculus of mobile process (Parts I and II). Information and Computation 100(1) (September 1992): 1–77.

[6] Milner, R. Communication and concurrency. Englewood Cliffs, NJ: Prentice Hall, 1989.

[7] Victor, B., and Moller, F. The mobility workbench: A tool for the π-calculus. Proceedings of the 6th International Conference on Computer Aided Verification, Lecture Notes In Computer Science 818 (1994): 428–440.

[8] Cavada, R., Cimatti, A., Olivetti, E., Pistore, M., and Roveri, M. NuSMV 2.1 user manual [online]. (n.d.). Available from: http://nusmv.irst.itc.it [2004, May 17]

[9] Lam, V.S.W., and Padget, J. Formalization of UML statechart diagrams in π-calculus. Proceedings 2001 Australian Software Engineering Conference, IEEE Computer Society, 2001, pp. 213–223.

[10] Lam, V.S.W., and Padget, J. Analyzing equivalences of UML statechart diagrams by structural congruence and open bisimulations. In 2003 IEEE Symposium on Human Centric Computing Languages and Environments, IEEE Computer Society, 2003, pp. 137–144.

[11] Lam, V.S.W., and Padget, J. Symbolic model checking of UML statechart diagrams with an integrated approach. 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, 2004, pp. 337-346

[12] Ng, M. Y., and Butler, M. Towards formalizing UML state diagrams in CSP. Proceedings of the first International Conference on Software Engineering and Formal Methods (SEFM'03), 2003, pp. 138-147.

[13] Aoki, T., Tateishi, T., and Katayama, T. An axiomatic formalization of UML models [online]. 2001. Available from: http://kt-www.jaist.ac.jp/students/toshiaki/reserch/ruml.pdf [2004, August 16]

[14] Okazaki, M., Aoki, T., and Katayama, T. Formalizing sequence diagrams and state machines using Concurrent Regular Expression. Proceedings of 2nd International Workshop on Scenarios and State Machines: Models, Algorithms, and Tools (SCESM'03), 2003, pp. 74-79.

[15] Khriss, I., Elkoutbi, M., and Keller, R. K. Automating the synthesis of UML statecharts diagrams from multiple collaboration diagrams. In <<UML>>'98, Lecture Notes In Computer Science 1618 (1999): 132-147.

[16] Harel, D. Statecharts: A visual formalism for complex system. Science of Computer Programming 8. (1987): 231-274.

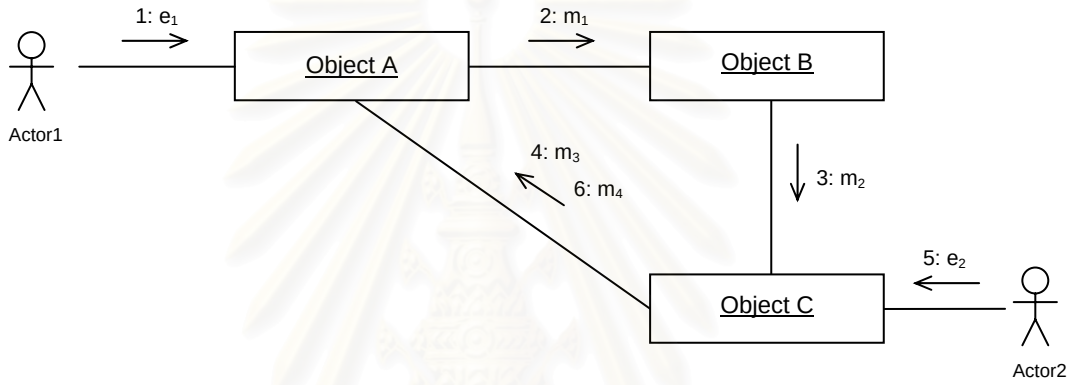
ภาคผนวก

ภาคผนวก ก

ตัวอย่างการแปลงแผนภาพความร่วมมือ และแผนภาพสถานะไปเป็นไพแคลคูลัส และการตรวจสอบอย่างละเอียด

การแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัส

สมมติให้ระบบที่ศึกษามีวัตถุ 3 วัตถุ ทำงานร่วมกัน คือ วัตถุ A B และ C โดยพฤติกรรมของวัตถุทั้งสาม สามารถแสดงได้ด้วยแผนภาพความร่วมมือ ดังรูปที่ ก-1 และ ก-2



รูปที่ ก-1 แผนภาพความร่วมมือของ Scenario1

จากกฎข้อที่ 1 จะได้

$$\vec{e}_{EXT} = e_1, e_2$$
$$\vec{e}_{INT} = m_1, m_2, m_3, m_4$$
$$\overrightarrow{enq} = enq_A, enq_B, enq_C$$

ต่อไปจากกฎข้อที่ 3 พิจารณาลำดับของข้อความที่ได้บรรยายไว้ในรูปที่ ก-1 จะได้ดังนี้ คือ

$$e_1, m_1, m_2, m_3, e_2, m_4$$

และเมื่อนำข้อความข้างต้นไปใส่ไว้ในแถวคอยที่ได้บรรยายไว้ด้วยกฎข้อที่ 2 จะได้

$$Queue_6^F(deq, empty, e_1, m_1, m_2, m_3, e_2, m_4)$$
$$deq(r).\bar{r}\langle e_1\rangle.Queue_5^F(deq, empty, m_1, m_2, m_3, e_2, m_4) +$$
$$empty(tf).\bar{f}.Queue_6^F(deq, empty, e_1, m_1, m_2, m_3, e_2, m_4)$$
$$Queue_0^F(deq, empty) =$$
$$empty(tf).\bar{t}.Queue_0^F(deq, empty)$$

และจากกฎข้อที่ 4 5 และ 6 จะสามารถแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัสได้ ดังนี้

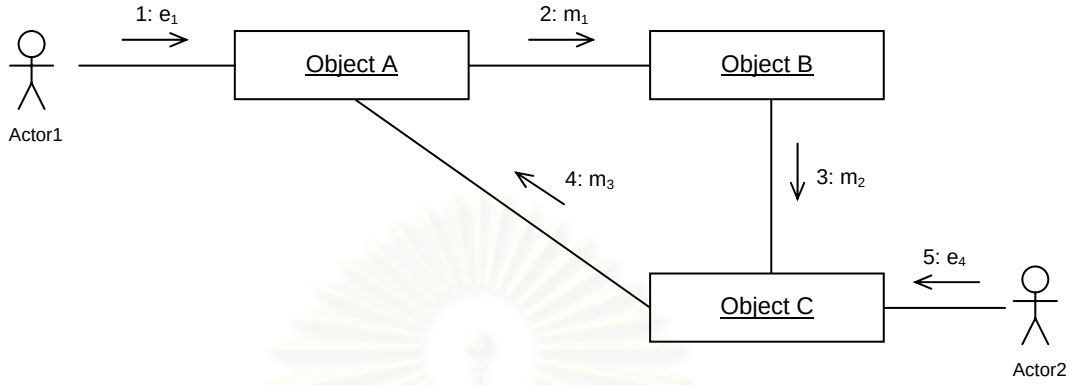
$$Scenario_1(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) = \overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$
$$(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario_1'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario_1''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) +$$
$$t.Success$$

$$Scenario_1'\ (event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) =$$
$$[event = e_1]\overline{enq}_A\langle event\rangle.Scenario_1(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = e_2]\overline{enq}_C\langle event\rangle.Scenario_1\ (empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error)$$

$$Scenario_1''(event, \overrightarrow{in}, \vec{e}_{INT}, \vec{e}_{SYS}) =$$
$$[event = m_1]in_{AB}(x).$$
$$([x = event]Scenario_1(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error) +$$
$$[event = m_2]in_{BC}(x).$$
$$([x = event]Scenario_1(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error) +$$
$$[event = m_3]in_{CA}(x).$$
$$([x = event]Scenario_1(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error) +$$
$$[event = m_4]in_{CA}(x).$$
$$([x = event]Scenario_1(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error)$$

และจากแผนภาพความร่วมมือในรูปที่ ก-1 พบว่าไม่มีการส่งข้อความแบบพร้อมกัน ดังนั้นจึงสามารถละการบรรยาย $Scenario'''$ และ $Scenario^4$ ได้

ต่อไปพิจารณารูปที่ ก-2



รูปที่ ก-2 แผนภาพความร่วมมือของ Scenario2

จากกฎข้อที่ 1 จะได้

$$\vec{e}_{EXT} = e_1, e_4$$
$$\vec{e}_{INT} = m_1, m_2, m_3$$

และจากกฎข้อที่ 2 และ 3 จะได้

$$Queue_5^F\,(deq, empty, e_1, m_1, m_2, m_3, e_4)$$

การแปลงแผนภาพสถานะไปเป็นไพแคลคูลัส

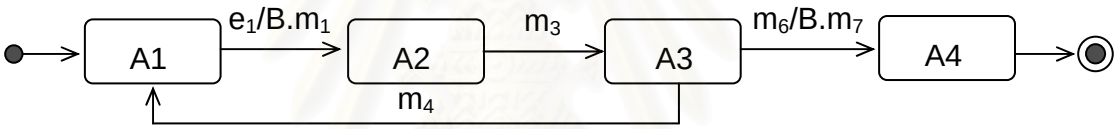
ขั้นตอนแรกของการแปลงแผนภาพสถานะไปเป็นไพแคลคูลัส คือ กำหนดให้มีแถวคอยของเหตุการณ์ (event queue) โปรแกรมเลือกจ่ายเหตุการณ์ (event dispatcher) และตัวประมวลผลเหตุการณ์ (event processor) สำหรับแผนภาพสถานะของแต่ละวัตถุที่มีอยู่ในระบบ ซึ่งสามารถบรรยายด้วยไพแคลคูลัสได้โดยใช้กฎข้อ 7 8 และ 9 ได้ ดังนี้

$$Queue_0^F\,(enq, deq, empty) =$$
$$enq(x_1).Queue_1^F\,(enq, deq, empty, x_1) +$$
$$empty(tf).\bar{t}.Queue_0^F\,(enq, deq, empty)$$
$$Queue_n^F\,(enq, deq, empty, x_1, x_2, ..., x_n) =$$
$$enq(x_{n+1}).Queue_{n+1}^F(enq, deq, empty, x_1, ..., x_n, x_{n+1}) +$$
$$deq(r).\bar{r}\langle x_1\rangle.Queue_{n-1}^F(enq, deq, empty, x_2, ..., x_n) +$$
$$empty(tf).\bar{f}.Queue_n^F\,(enq, deq, empty, x_1, ..., x_n)$$

$$Dispatch^{F}(t, f, r, deq, empty, execute, complete) =$$
$$\overline{empty}\langle t\, f\rangle.$$
$$t.Dispatch^{F}(t, f, r, deq, empty, execute, complete) +$$
$$f.\overline{deq}\langle r\rangle.r(event).\overline{execute}\langle event\rangle.complete.$$
$$Dispatch^{F}(t, f, r, deq, empty, execute, complete))$$

$$S_0(execute, step, complete, event_s) =$$
$$execute(x).\overline{event_s}\langle x\rangle.step.$$
$$\overline{complete}.S_0(execute, step, complete, event_s)$$

ขั้นตอนต่อไปจะแปลงแผนภาพสถานะของวัตถุ A B และ C ไปเป็นไพแคลคูลัส โดยเริ่มจากวัตถุ A ซึ่งพฤติกรรมของวัตถุ A บรรยายไว้ในแผนภาพสถานะ ดังรูปที่ ก-2 และสามารถแปลงได้ ดังนี้



รูปที่ ก-3 แผนภาพสถานะของวัตถุ A

ให้ $\vec{e}_A = e_1, m_1, m_3, m_4, m_6, m_7$

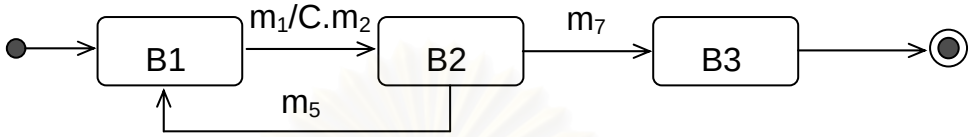
$$A_1(step, event_S, \vec{e}_A, \overrightarrow{enq}) =$$
$$event_s(x).([x = e_1]\overline{enq}_B\langle m_1\rangle.\overline{in}_{AB}\langle m_1\rangle.\overline{step}.A_2(step, event_s, \vec{e}_A, \overrightarrow{enq}) +$$
$$\sum_{e\in\{\vec{e}_A\}\setminus\{e_1\}}[x = e]Error)$$

$$A_2(step, event_S, \vec{e}_A, \overrightarrow{enq}) =$$
$$event_s(x).([x = m_3]\overline{step}.A_3(step, event_s, \vec{e}_A, \overrightarrow{enq}) +$$
$$\sum_{e\in\{\vec{e}_A\}\setminus\{m_3\}}[x = e]Error)$$

$$A_3(step, event_S, \vec{e}_A, \overrightarrow{enq}) =$$
$$event_s(x).([x = m_6]\overline{enq}_B\langle m_7\rangle.\overline{in}_{AB}\langle m_7\rangle.\overline{step}.A_1(step, event_s, \vec{e}_A, \overrightarrow{enq}) +$$
$$[x = m_4]\overline{step}.A_1(step, event_s, \vec{e}_A, \overrightarrow{enq}) +$$
$$\sum_{e\in\{\vec{e}_A\}\setminus\{m_6\}\cup\{m_4\}}[x = e]Error)$$

$$A_4(step, event_s, \vec{e}_A, \overrightarrow{enq}) = 0$$

ต่อไปพิจารณาวัตถุ B ซึ่งพฤติกรรมของวัตถุ B บรรยายไว้ในแผนภาพสถานะดังรูปที่ ก-3 และสามารถแปลงได้ ดังนี้



รูปที่ ก-4 แผนภาพสถานะของวัตถุ B

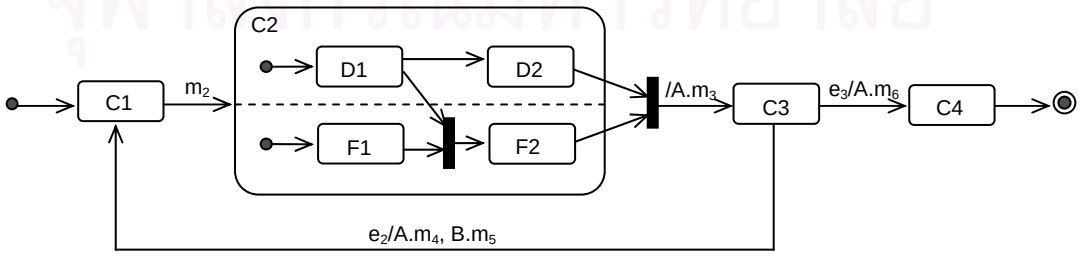
ให้ $\vec{e}_B = m_1, m_2, m_5, m_7$

$$B_1(step, event_S, \vec{e}_B, \overrightarrow{enq}) =$$
$$event_s(x).([x = m_1]\overline{enq}_C\langle m_2\rangle.\overline{in}_{AC}\langle m_2\rangle.\overline{step}.B_2(step, event_s, \vec{e}_B, \overrightarrow{enq}) +$$
$$\sum_{e\in\{\vec{e}_B\}\setminus\{m_1\}}[x = e]Error)$$

$$B_2(step, event_S, \vec{e}_B, \overrightarrow{enq}) =$$
$$event_s(x).([x = m_7]\overline{step}.B_3(step, event_s, \vec{e}_B, \overrightarrow{enq}) +$$
$$[x = m_5]\overline{step}.B_1(step, event_s, \vec{e}_B, \overrightarrow{enq}) +$$
$$\sum_{e\in\{\vec{e}_B\}\setminus\{m_5\}\cup\{m_7\}}[x = e]Error)$$

$$B_3(step, event_S, \vec{e}_B, \overrightarrow{enq}) = 0$$

ต่อไปพิจารณาวัตถุ C ซึ่งพฤติกรรมของวัตถุ C บรรยายไว้ในแผนภาพสถานะดังรูปที่ ก-4 และสามารถแปลงได้ ดังนี้



รูปที่ ก-5 แผนภาพสถานะของวัตถุ C

ให้ $\vec{e}_C = e_2, e_3, m_2, m_3, m_4, m_5, m_6$

$$C_1(step, event_S, \vec{e}_C, \overrightarrow{enq}) =$$
$$event_s(x).([x = m_2]\overline{step}.C_2(step, event_s, \vec{e}_C, \overrightarrow{enq}) +$$
$$\sum_{e \in \{\vec{e}_C\} \setminus \{m_2\}} [x = e] Error)$$

$$C_2(step, event_S, \vec{e}_C, \overrightarrow{enq}) =$$
$$\overline{enq}_A \langle m_3 \rangle . \overline{in}_{CA} \langle m_3 \rangle . \overline{step}.C_3(step, event_s, \vec{e}_C, \overrightarrow{enq})$$

$$C_3(step, event_S, \vec{e}_C, \overrightarrow{enq}) =$$
$$event_s(x).([x = e_2]\overline{enq}_A \langle m_4 \rangle . \overline{in}_{CA} \langle m_4 \rangle .$$
$$\overline{enq}_B \langle m_5 \rangle . \overline{in}_{CB} \langle m_5 \rangle . \overline{step}.C_1(step, event_s, \vec{e}_C, \overrightarrow{enq}) +$$
$$[x = e_3]\overline{enq}_A \langle m_6 \rangle . \overline{in}_{CA} \langle m_6 \rangle . \overline{step}.C_4(step, event_s, \vec{e}_C, \overrightarrow{enq}) +$$
$$\sum_{e \in \{\vec{e}_C\} \setminus \{e_2\} \cup \{e_3\}} [x = e] Error)$$

$$C_4(step, event_S, \vec{e}_C, \overrightarrow{enq}) = 0$$

การตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะโดยใช้ไพแคลคูลัส

ในขั้นตอนนี้จะตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะ โดยวัตถุ A B และ C จะทำงานแบบขนานกัน (parallel) แต่เพื่อความเข้าใจจะบรรยายการทำงานของวัตถุเป็นคู่ ๆ โดยเป็นคู่ที่กำลังทำงานอยู่ ณ เวลานั้น เริ่มที่แถวคอย และ $Scenario_1$ ของแผนภาพความร่วมมือ ดังนี้

$$Queue_6^F(deq, empty, e_1, m_1, m_2, m_3, e_2, m_4) \quad | \quad Scenario_1(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$$

$$= \quad deq(r).\bar{r}\langle e_1\rangle.Queue_5^F(deq, empty, m_1, m_2, m_3, e_2, m_4) + \quad | \quad \overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$
$$empty(t\ f).\bar{f}.Queue_6^F(deq, empty, e_1, m_1, m_2, m_3, e_2, m_4) \quad (\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario_1'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario_1''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) +$$
$$t.Success$$

$$\xrightarrow{\tau} \quad \bar{f}.Queue_6^F(deq, empty, e_1, m_1, m_2, m_3, e_2, m_4) \quad | \quad f.\overline{deq}\langle y\rangle.y(event).$$
$$(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario_1'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario_1''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) +$$
$$t.Success$$

$\xrightarrow{\tau}$ $Queue_6^F(deq, empty, e_1, m_1, m_2, m_3, e_2, m_4)$ | $\overline{deq}\langle y\rangle.y(event).$
$(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario_1'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$
$\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario_1''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$
$[event = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$=$ $deq(r).\bar{r}\langle e_1\rangle.Queue_5^F(deq, empty, m_1, m_2, m_3, e_2, m_4) +$
$empty(tf).\bar{f}.Queue_6^F(deq, empty, e_1, m_1, m_2, m_3, e_2, m_4)$ | $\overline{deq}\langle y\rangle.y(event).$
$(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario_1'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$
$\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario_1''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$
$[event = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$\xrightarrow{\tau}$ $\bar{y}\langle e_1\rangle.Queue_5^F(deq, empty, m_1, m_2, m_3, e_2, m_4)$ | $y(event).$
$(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario_1'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$
$\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario_1''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$
$[event = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$\xrightarrow{\tau}$ $Queue_5^F(deq, empty, m_1, m_2, m_3, e_2, m_4)$ | $(\sum_{e\in\{\vec{e}_{EXT}\}}[e_1 = e].Scenario_1'(e_1, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$
$\sum_{e\in\{\vec{e}_{INT}\}}[e_1 = e]Scenario_1''(e_1, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$
$[e_1 = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$$\xrightarrow{\tau} \quad Queue_5^F(deq, empty, m_1, m_2, m_3, e_2, m_4) \quad | \quad Scenario_1'(e_1, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS})$$

$$= \quad deq(r).\bar{r}\langle m_1\rangle.Queue_4^F(deq, empty, m_2, m_3, e_2, m_4) + \quad | \quad [e_1 = e_1]\overline{enq}_A\langle e_1\rangle.Scenario_1(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$empty(t\,f).\bar{f}.Queue_5^F(deq, empty, m_1, m_2, m_3, e_2, m_4) \quad | \quad [e_1 = e_2]\overline{enq}_C\langle e_1\rangle.Scenario_1(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[e_1 = e]Error)$$

$$\xrightarrow{\tau} \quad deq(r).\bar{r}\langle m_1\rangle.Queue_4^F(deq, empty, m_2, m_3, e_2, m_4) + \quad | \quad \overline{enq}_A\langle e_1\rangle.Scenario_1(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$$
$$empty(t\,f).\bar{f}.Queue_5^F(deq, empty, m_1, m_2, m_3, e_2, m_4)$$

ต่อไปพิจารณาการทำงานของ $Scenario_1$ ที่นำข้อความมาใส่ไว้ในแถวคอยของวัตถุ A ดังนี้

$$Queue_0^A(enq_A, deq_A, empty_A) \quad | \quad \overline{enq}_A\langle e_1\rangle.Scenario_1(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$$

$$= \quad enq_A(x_1).Queue_1^A(enq_A, deq_A, empty_A, x_1) + \quad | \quad \overline{enq}_A\langle e_1\rangle.Scenario_1(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$$
$$empty_A(t\,f).\bar{t}.Queue_0^A(enq_A, deq_A, empty_A)$$

$$\xrightarrow{\tau} \quad Queue_1^A(enq_A, deq_A, empty_A, e_1) \qquad \text{(ก-1)} \quad | \quad Scenario_1(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) \qquad \text{(ก-2)}$$

จะพบว่าขณะนี้มีข้อความ e_1 อยู่ในแถวคอยของวัตถุ A (ก-1) และ $Scenario_1$ พร้อมที่จะนำข้อความถัดไปจากแถวคอยของแผนภาพความร่วมมือมาพิจารณาต่อไป (ก-2) ซึ่งวัตถุทั้งสองคู่จะทำงานไปพร้อม ๆ กันโดยจะอธิบายการทำงานทีละคู่ได้ดังนี้

จาก (ก-2) พิจารณาการทำงานร่วมกันของ $Scenario_1$ และแถวคอยของแผนภาพความร่วมมือ

$$Queue_5^F(deq, empty, m_1, m_2, m_3, e_2, m_4) \mid Scenario_1(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$$

$$= deq(r).\bar{r}\langle m_1\rangle.Queue_4^F(deq, empty, m_2, m_3, e_2, m_4) + empty(t\ f).\bar{f}.Queue_5^F(deq, empty, m_1, m_2, m_3, e_2, m_4) \mid \overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario_1'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario_1''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) + [event = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) + t.Success$$

$$\xrightarrow{\tau} \bar{f}.Queue_5^F(deq, empty, m_1, m_2, m_3, e_2, m_4) \mid f.\overline{deq}\langle y\rangle.y(event).(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario_1'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario_1''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) + [event = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) + t.Success$$

$$\xrightarrow{\tau} Queue_5^F(deq, empty, m_1, m_2, m_3, e_2, m_4) \mid \overline{deq}\langle y\rangle.y(event).(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario_1'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario_1''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) + [event = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$$

$$
\begin{aligned}
= \quad & deq(r).\bar{r}\langle m_1\rangle.Queue_4^F(deq, empty, m_2, m_3, e_2, m_4) + \\
& empty(tf).\bar{f}.Queue_5^F(deq, empty, m_1, m_2, m_3, e_2, m_4)
\end{aligned}
\quad | \quad
\begin{aligned}
& \overline{deq}\langle y\rangle.y(event). \\
& (\textstyle\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario_1'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \\
& \textstyle\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario_1''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) + \\
& [event = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))
\end{aligned}
$$

$$
\xrightarrow{\tau} \quad \bar{y}\langle m_1\rangle.Queue_4^F(deq, empty, m_2, m_3, e_2, m_4)
\quad | \quad
\begin{aligned}
& y(event). \\
& (\textstyle\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario_1'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \\
& \textstyle\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario_1''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) + \\
& [event = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))
\end{aligned}
$$

$$
\xrightarrow{\tau} \quad Queue_4^F(deq, empty, m_2, m_3, e_2, m_4)
\quad | \quad
\begin{aligned}
& (\textstyle\sum_{e\in\{\vec{e}_{EXT}\}}[m_1 = e].Scenario_1'(m_1, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \\
& \textstyle\sum_{e\in\{\vec{e}_{INT}\}}[m_1 = e]Scenario_1''(m_1, in, \vec{e}_{INT}, \vec{e}_{SYS}) + \\
& [m_1 = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))
\end{aligned}
$$

$$
\xrightarrow{\tau} \quad Queue_4^F(deq, empty, m_2, m_3, e_2, m_4)
\quad | \quad
Scenario_1''(m_1, in, \vec{e}_{INT}, \vec{e}_{SYS})
$$

$$
\begin{aligned}
= \quad & deq(r).\bar{r}\langle m_2\rangle.Queue_3^F(deq, empty, m_3, e_2, m_4) + \\
& empty(tf).\bar{f}.Queue_4^F(deq, empty, m_2, m_3, e_2, m_4)
\end{aligned}
\quad | \quad
\begin{aligned}
& in_{AB}(x). \\
& ([x = m_1]Scenario_1(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) + \\
& \textstyle\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{m_1\}}[x = e]Error) \qquad \text{(ก-3)}
\end{aligned}
$$

จาก (ก-1) พิจารณาการทำงานของวัตถุ A ที่นำข้อความออกมาจากแถวคอยด้วยโปรแกรมเลือกจ่ายเหตุการณ์ ดังนี้

$$
\begin{aligned}
& Queue_1^A(enq_A, deq_A, empty_A, e_1) && | \quad Dispatch^A(t, f, r, deq_A, empty_A, execute, complete) \\
= \quad & enq_A(x).Queue_1^A(enq_A, deq_A, empty_A, e_1, x) + && | \quad \overline{empty}_A\langle t\, f\rangle.(\\
& deq_A(r).\bar{r}\langle e_1\rangle.Queue_0^A(enq_A, deq_A, empty_A) + && \quad t.Dispatch^A(t, f, r, deq_A, empty_A, execute, complete) + \\
& empty_A(t\, f).\bar{f}.Queue_1^A(enq_A, deq_A, empty_A, e_1) && \quad f.\overline{deq}_A\langle r\rangle.r(event).\overline{execute}\langle event\rangle.complete. \\
& && \quad Dispatch^A(t, f, r, deq_A, empty_A, execute, complete)) \\
\xrightarrow{\tau} \quad & \bar{f}.Queue_1^A(enq_A, deq_A, empty_A, e_1) && | \quad t.Dispatch^A(t, f, r, deq_A, empty_A, execute, complete) + \\
& && \quad f.\overline{deq}_A\langle r\rangle.r(event).\overline{execute}\langle event\rangle.complete. \\
& && \quad Dispatch^A(t, f, r, deq_A, empty_A, execute, complete) \\
\xrightarrow{\tau} \quad & Queue_1^A(enq_A, deq_A, empty_A, e_1) && | \quad \overline{deq}_A\langle r\rangle.r(event).\overline{execute}\langle event\rangle.complete. \\
& && \quad Dispatch^A(t, f, r, deq_A, empty_A, execute, complete) \\
= \quad & enq_A(x).Queue_1^A(enq_A, deq_A, empty_A, e_1, x) + && | \quad \overline{deq}_A\langle r\rangle.r(event).\overline{execute}\langle event\rangle.complete. \\
& deq_A(r).\bar{r}\langle e_1\rangle.Queue_0^A(enq_A, deq_A, empty_A) + && \quad Dispatch^A(t, f, r, deq_A, empty_A, execute, complete) \\
& empty_A(t\, f).\bar{f}.Queue_1^A(enq_A, deq_A, empty_A, e_1) && \\
\xrightarrow{\tau} \quad & .\bar{r}\langle e_1\rangle.Queue_0^A(enq_A, deq_A, empty_A) && | \quad r(event).\overline{execute}\langle event\rangle.complete. \\
& && \quad Dispatch^A(t, f, r, deq_A, empty_A, execute, complete)
\end{aligned}
$$

$\xrightarrow{\tau}$ $Queue_0^A(enq_A, deq_A, empty_A)$ | $\overline{execute}\langle e_1\rangle.complete.$
$Dispatch^A(t, f, r, deq_A, empty_A, execute, complete)$

ต่อไปเป็นการทำงานระหว่างตัวประมวลผลเหตุการณ์และโปรแกรมเลือกจ่ายเหตุการณ์

$Dispatch^A(t, f, r, deq_A, empty_A, execute, complete)$ | $S_0(execute, step, complete, event_s)$

$=$ $\overline{execute}\langle e_1\rangle.complete.$ | $execute(x).\overline{event}_s\langle x\rangle.step.$
$Dispatch^A(t, f, r, deq_A, empty_A, execute, complete)$ $\overline{complete}.S_0(execute, step, complete, event_s)$

$\xrightarrow{\tau}$ $complete.$ | $\overline{event}_s\langle e_1\rangle.step.$
$Dispatch^A(t, f, r, deq_A, empty_A, execute, complete)$ $\overline{complete}.S_0(execute, step, complete, event_s)$

เมื่อตัวประมวลผลเหตุการณ์ได้รับข้อความจากโปรแกรมเลือกจ่ายเหตุการณ์จะนำข้อความที่ได้ไปให้วัตถุ A ดังนี้

$S_0(execute, step, complete, event_s)$ | $A_1(step, event_S, \vec{e}_A, \overrightarrow{enq})$

$=$ $\overline{event}_s\langle e_1\rangle.step.$ | $event_s(x).([x = e_1]\overline{enq}_B\langle m_1\rangle.\overline{in}_{AB}\langle m_1\rangle$
$\overline{complete}.S_0(execute, step, complete, event_s)$ $.\overline{step}.A_2(step, event_s, \vec{e}_A, \overrightarrow{enq}) +$
$\sum_{e\in\{\vec{e}_A\}\setminus\{e_1\}}[x = e]Error)$

$\xrightarrow{\tau}$ $step.$ | $\overline{enq}_B\langle m_1\rangle.\overline{in}_{AB}\langle m_1\rangle.\overline{step}.A_2(step, event_s, \vec{e}_A, \overrightarrow{enq})$
$\overline{complete}.S_0(execute, step, complete, event_s)$

จะพบว่าเมื่อวัตถุ A ได้รับข้อความ e_1 จะส่งข้อความ m_1 ไปให้วัตถุ B โดยนำข้อความดังกล่าวไปใส่ไว้ในแถวคอยของวัตถุ ดังนี้

$$Queue_0^B(enq_B, deq_B, empty_B) \quad | \quad A_1(step, event_S, \vec{e}_A, \overrightarrow{enq})$$

$$= enq_B(x_1).QueueB(enq_B, deq_B, empty_B, x_1) + empty_B(tf).\bar{t}.Queue_0^B(enq_B, deq_B, empty_B) \quad | \quad \overline{enq}_B\langle m_1\rangle.\overline{in}_{AB}\langle m_1\rangle.\overline{step}.A_2(step, event_s, \vec{e}_A, \overrightarrow{enq})$$

$$\xrightarrow{\tau} Queue_1^B(enq, deq, empty, m_1) \quad \text{(ก-4)} \quad | \quad \overline{in}_{AB}\langle m_1\rangle.\overline{step}.A_2(step, event_s, \vec{e}_A, \overrightarrow{enq}) \quad \text{(ก-5)}$$

นอกจากวัตถุ A จะส่งข้อความ m_1 ไปให้วัตถุ B แล้ว ยังส่งข้อความดังกล่าวไปตรวจสอบความต้องกันกับแผนภาพความร่วมมือผ่านทาง $Scenario_1''$ อีกด้วย (จาก (ก-3) และ (ก-5))

$$Scenario_1''(m_1, in, \vec{e}_{INT}, \vec{e}_{SYS}) \quad | \quad A_1(step, event_S, \vec{e}_A, \overrightarrow{enq})$$

$$= in_{AB}(x). ([x = m_1]Scenario_1(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) + \sum_{e \in \{\vec{e}_{SYS}\} \setminus \{m_1\}} [x = e]Error) \quad | \quad \overline{in}_{AB}\langle m_1\rangle.\overline{step}.A_2(step, event_s, \vec{e}_A, \overrightarrow{enq})$$

$$\xrightarrow{\tau} Scenario_1(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) \quad | \quad \overline{step}.A_2(step, event_s, \vec{e}_A, \overrightarrow{enq})$$

ทำคล้าย (ก-1) ไปเรื่อย ๆ จนข้อความที่อยู่ในแถวคอยของ $Queue_6^F(deq, empty, e_1, m_1, m_2, m_3, e_2, m_4)$ หมด ขั้นตอนต่อไปจะเป็น ดังนี้

$$Queue_0^F(deq, empty) \mid Scenario_1(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$$

$$= empty(t\ f).\bar{t}.Queue_0^F(deq, empty) \mid \overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$
$$(\sum_{e\in\{\bar{e}_{EXT}\}}[event = e].Scenario_1'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\bar{e}_{INT}\}}[event = e]Scenario_1''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) +$$
$$t.Success$$

$$\xrightarrow{\tau} \bar{t}.Queue_0^F(deq, empty) \mid t.Success$$

$$\xrightarrow{\tau} Queue_0^F(deq, empty) \mid Success$$

จากตัวอย่างข้างต้นแสดงว่าลำดับของเหตุการณ์ที่บรรยายไว้ในแผนภาพความร่วมมือของ Scenario1 ถูกบรรยายไว้แล้วในแผนภาพสถานะ เนื่องจากไม่มีข้อความในแถวคอยของแผนภาพความร่วมมือแล้ว และไม่พบข้อผิดพลาดใด ๆ จากวัตถุอื่นที่บรรยายไว้ด้วยแผนภาพสถานะ

ต่อไปพิจารณากรณีที่ระบบมีการบรรยายแผนภาพความร่วมมือดังรูป ก-2 โดยข้อความที่อยู่ในแถวคอยเป็น ดังนี้

$$Queue_5^F\,(deq, empty, e_1, m_1, m_2, m_3, e_4)$$

และเมื่อทำตามขั้นตอนดังที่กล่าวมาข้างต้นพบว่าเมื่อข้อความในแถวคอยของแผนภาพความร่วมมือเป็น e_4 จะเกิดเหตุการณ์ดังนี้

$$Queue_1^F\,(deq, empty, e_4) \;|\; Scenario_1(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$$

$$= deq(r).\bar{r}\langle e_4\rangle.Queue_0^F\,(deq, empty) + empty(t\,f).\bar{f}.Queue_1^F\,(deq, empty, e_4) \;|\; \overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).(\textstyle\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario_1'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario_1''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) + [event = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) + t.Success$$

$$\xrightarrow{\tau} \bar{f}.Queue_1^F\,(deq, empty, e_4) \;|\; f.\overline{deq}\langle y\rangle.y(event).(\textstyle\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario_1'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario_1''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) + [event = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) + t.Success$$

$\xrightarrow{\tau}$ $Queue_1^F(deq, empty, e_4)$ | $\overline{deq}\langle y\rangle.y(event).$
$(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario_1'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$
$\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario_1''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$
$[event = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$= deq(r).\bar{r}\langle e_4\rangle.Queue_0^F(deq, empty) +$
$empty(tf).\bar{f}.Queue_1^F(deq, empty, e_4)$ | $\overline{deq}\langle y\rangle.y(event).$
$(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario_1'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$
$\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario_1''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$
$[event = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$\xrightarrow{\tau}$ $\bar{y}\langle e_4\rangle.Queue_0^F(deq, empty)$ | $y(event).$
$(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario_1'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$
$\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario_1''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$
$[event = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$\xrightarrow{\tau}$ $Queue_0^F(deq, empty)$ | $(\sum_{e\in\{\vec{e}_{EXT}\}}[e_4 = e].Scenario_1'(e_4, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$
$\sum_{e\in\{\vec{e}_{INT}\}}[e_4 = e]Scenario_1''(e_4, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$
$[e_4 = \theta]Scenario_1'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$\xrightarrow{\tau}$ $Queue_0^F(deq, empty)$ | $Scenario_1'(e_4, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS})$

จากตัวอย่างข้างต้นแม้ว่าจะไม่มีข้อความในแถวคอยของแผนภาพความร่วมมือแล้ว แต่เมื่อพิจารณาจากแผนภาพสถานะของวัตถุ C หลังจากได้รับข้อความ e_4

$$
\begin{aligned}
& S_0(execute, step, complete, event_s) && | && C_3(step, event_S, \vec{e}_C, \overrightarrow{enq}) \\
= \; & \overline{event}_s\langle e_4\rangle.step. && | && event_s(x).([x = e_2]\overline{enq}_A\langle m_4\rangle.\overline{in}_{CA}\langle m_4\rangle. \\
& \overline{complete}.S_0(execute, step, complete, event_s) && && \overline{enq}_B\langle m_5\rangle.\overline{in}_{CB}\langle m_5\rangle.\overline{step}.C_1(step, event_s, \vec{e}_C, \overrightarrow{enq}) + \\
& && && [x = e_3]\overline{enq}_A\langle m_6\rangle.\overline{in}_{CA}\langle m_6\rangle.\overline{step}.C_4(step, event_s, \vec{e}_C, \overrightarrow{enq}) + \\
& && && \Sigma_{e\in\{\vec{e}_C\}\setminus\{e_2\}\cup\{e_3\}}[x = e]Error) \\
\xrightarrow{\tau} \; & step. && | && Error \\
& \overline{complete}.S_0(execute, step, complete, event_s)
\end{aligned}
$$

จะพบว่าเกิดข้อผิดพลาดขึ้น เนื่องจากวัตถุ C ได้รับข้อความ e_4 จากแผนภาพความร่วมมือซึ่งไม่ได้ถูกบรรยายไว้ในแผนภาพสถานะของวัตถุ C ดังนั้นสรุปได้ว่าเกิดความไม่ต้องกันระหว่างแผนภาพความร่วมมือและแผนภาพสถานะขึ้น

ภาคผนวก ข

การตรวจสอบความถูกต้องของการออกแบบ ระบบคอมพิวเตอร์เพื่อสนับสนุนการทำงานร่วมกัน

การตรวจสอบความถูกต้องของการออกแบบระบบคอมพิวเตอร์เพื่อสนับสนุนการทำงานร่วมกัน ประเด็นหนึ่งที่น่าสนใจคือการตรวจสอบความต้องกันระหว่างแผนภาพต่าง ๆ ที่ใช้ในการออกแบบระบบ

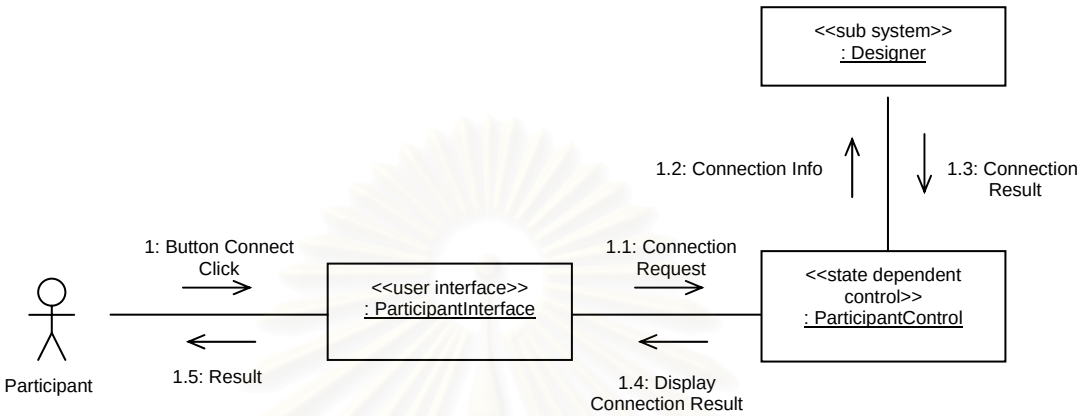
เนื้อหาในส่วนนี้จะตรวจสอบความต้องกันระหว่างแผนความร่วมมือและแผนภาพสถานะที่ใช้ออกแบบระบบ โดยเริ่มด้วยการแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัส การแปลงแผนภาพสถานะไปเป็นไพแคลคูลัส และสุดท้ายเป็นการตรวจสอบความต้องกันระหว่างไพแคลคูลัสที่ได้

การแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัส

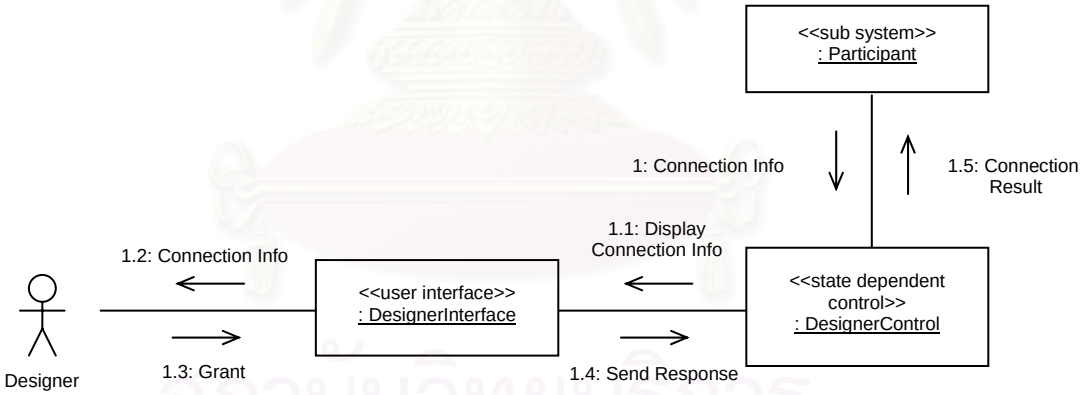
ขั้นตอนการแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัส เริ่มต้นด้วยการพิจารณาเหตุการณ์ที่ได้บรรยายไว้ในแผนภาพความร่วมมือแต่ละแผนภาพ โดยหากแผนภาพใดมีลำดับของเหตุการณ์ต่อเนื่องกันให้พิจารณาแผนภาพดังกล่าวร่วมกัน

นอกจากนี้ในการแปลงแผนภาพความร่วมมือจะไม่พิจารณาวัตถุที่ไม่มีการเปลี่ยนสถานะ เช่น วัตถุที่เป็นส่วนต่อประสานกับผู้ใช้ (user interface) หรือวัตถุที่เป็นเอนทิตี (entity) เนื่องจากวัตถุดังกล่าวไม่มีการบรรยายการเปลี่ยนสถานะไว้ในแผนภาพสถานะ จึงไม่สามารถนำมาตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะได้ แต่ให้พิจารณาวัตถุที่ไม่มีการเปลี่ยนสถานะเสมือนว่าเป็นเอนทิตีภายนอก (external entity) ที่มีปฏิสัมพันธ์กับระบบ

แผนภาพความร่วมมือ “Participant Join Project” และ “Designer Grant”

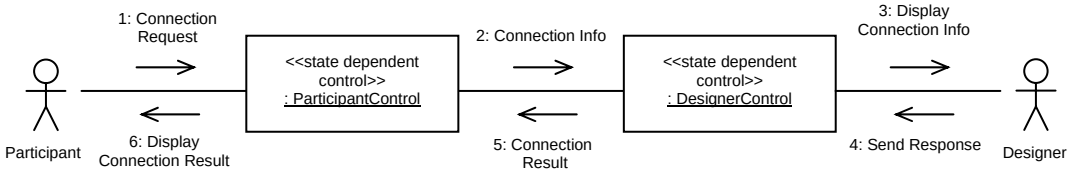


รูปที่ ข-1 แผนภาพความร่วมมือ “Participant Join Project”



รูปที่ ข-2 แผนภาพความร่วมมือ “Designer Grant”

จากรูปที่ ข-1 และ ข-2 จะพบว่าเหตุการณ์ที่บรรยายไว้ด้วยแผนภาพความร่วมมือของทั้ง 2 แผนภาพ มีความต่อเนื่องกัน ดังนั้นให้นำแผนภาพทั้ง 2 มาพิจารณาร่วมกันเป็นกรณีเดียว ดังรูป ข-3



รูปที่ ข-3 แผนภาพความร่วมมือประกอบของ “Participant Join Project” และ “Designer Grant”

จากตัวอย่างในรูปที่ ข-3 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

ใช้กฎข้อที่ 1 จะได้

$$\vec{e}_{EXT} = connection\_request, display\_connection\_Info, send\_response, display\_connection\_result$$

$$\vec{e}_{INT} = connection\_Info, connection\_result$$

ต่อไปจากกฎข้อที่ 3 พิจารณาลำดับของข้อความที่ได้บรรยายไว้ในรูปที่ ข-3 จะได้ ดังนี้ คือ

$$connection\_request, connection\_Info, display\_connection\_Info, send\_response, connection\_result, display\_connection\_result$$

และเมื่อนำข้อความข้างต้นไปใส่ไว้ในแถวคอยที่ได้บรรยายไว้ด้วยกฎข้อที่ 2 จะได้

$$Queue_6^F(deq, empty, connection\_request, ..., display\_connection\_result) = deq(r).\bar{r}\langle connection\_request\rangle.Queue_5^F(deq, empty, connection\_Info, ...) + empty(t\ f).\bar{f}.Queue_6^F(deq, empty, connection\_request, ...)$$

และจากกฎข้อที่ 4 5 และ 6 จะสามารถแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัสได้ ดังนี้

$$Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) = \overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$
$$(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) +$$
$$t.Success$$

$$Scenario'(event, ex, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) =$$

$$[event = connection\_request]\overline{enq}_{Par}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$

$$[event = send\_response]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$

$$[event = display\_connection\_Info]ex(x).$$

$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$

$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error) +$$

$$[event = display\_connection\_result]ex(x).$$

$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$

$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error)$$

$$Scenario''(event, \overrightarrow{in}, \vec{e}_{INT}, \vec{e}_{SYS}) =$$

$$[event = connection\_Info]in_{ParDes}(x).$$

$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$

$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error) +$$

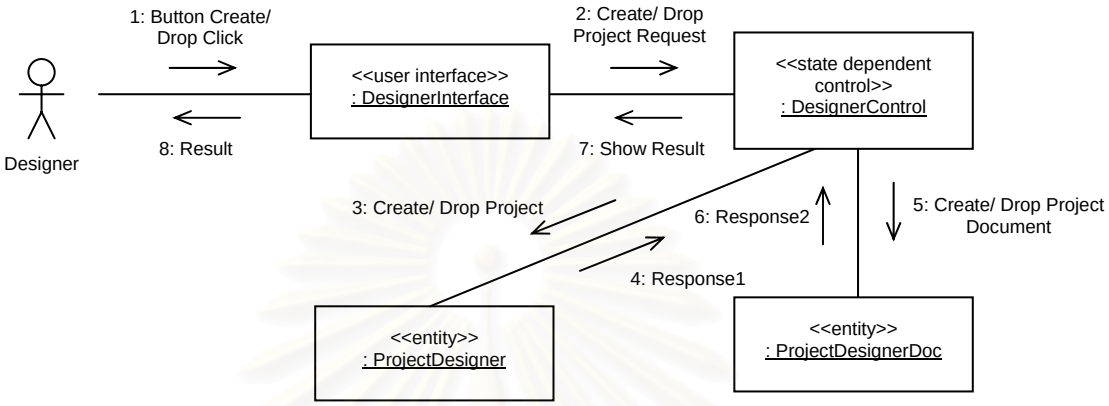
$$[event = connection\_result]in_{DesPar}(x).$$

$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$

$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error)$$

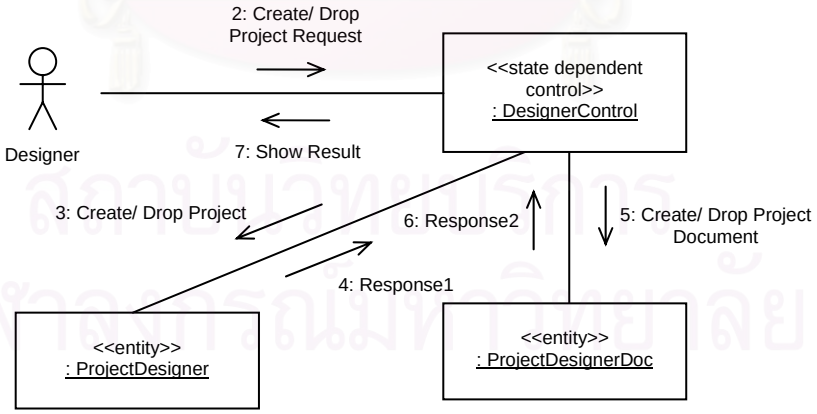
และจากแผนภาพความร่วมมือในรูปที่ ข-3 พบว่าไม่มีการส่งข้อความแบบพร้อมกัน ดังนั้นจึงสามารถละการบรรยาย $Scenario'''$ และ $Scenario^4$ ได้

แผนภาพความร่วมมือ “Designer Create/Drop Project”



รูปที่ ข-4 แผนภาพความร่วมมือ “Designer Create/Drop Project”

จากที่ได้กล่าวข้างต้นว่าการพิจารณาวัตถุที่ไม่มีการเปลี่ยนสถานะ ให้พิจารณาเสมือนว่าวัตถุนั้นเป็นเอนทิตีภายนอก (external entity) ที่มีปฏิสัมพันธ์กับระบบ ทำให้สามารถพิจารณารูปที่ ข-4 ได้เสมือนกับรูป ข-5



รูปที่ ข-5 แผนภาพความร่วมมือ “Designer Create/Drop Project“ ที่แก้ไขแล้ว

จากตัวอย่างในรูปที่ ข-5 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

ใช้กฎข้อที่ 1 จะได้

$$\vec{e}_{EXT} = create\_drop\_project\_request, create\_drop\_project\ response1, create\_drop\_project\_doc, response2, show\_result$$

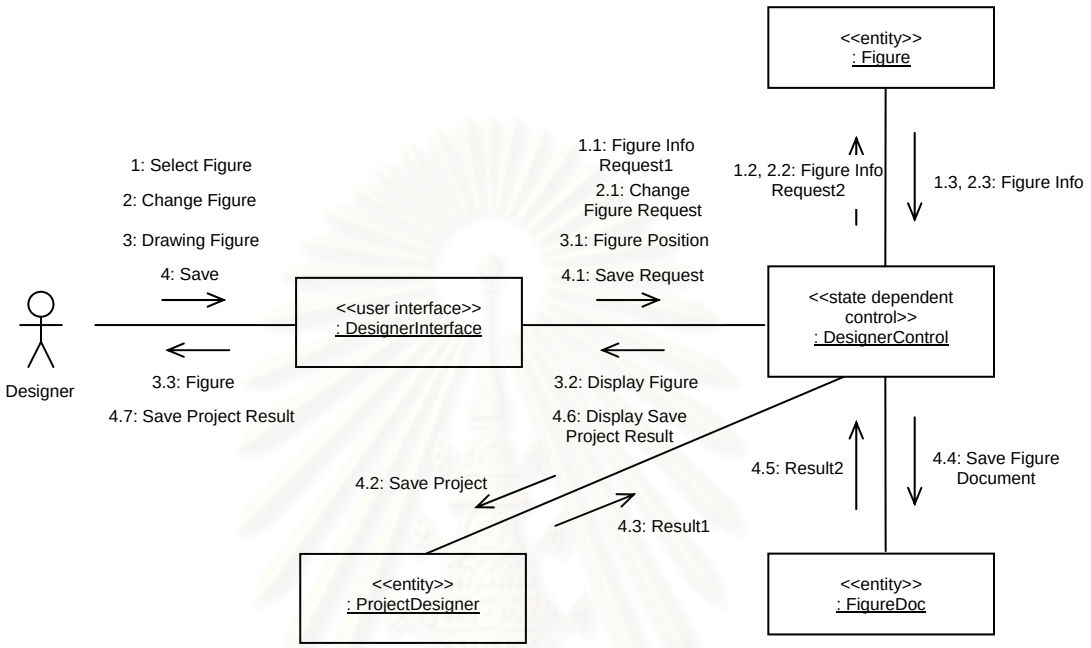
ต่อไปพิจารณาลำดับของข้อความที่ได้บรรยายไว้ในรูปที่ ข-5 จะได้ ดังนี้ คือ

$$create\_drop\_project\_request, create\_drop\_project, response1$$
$$create\_drop\_project\_doc, response2, show\_result$$

และจากกฎข้อที่ 5 จะสามารถแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัสได้ ดังนี้

$$Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}) =$$
$$[event = create\_drop\_project\_request]\overline{enq}_{Des}\langle event\rangle.$$
$$Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) +$$
$$[event = response1]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) +$$
$$[event = response2]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) +$$
$$[event = create\_drop\_project]ex(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT})) +$$
$$[event = create\_drop\_project\_doc]ex(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT})) +$$
$$[event = show\_result]ex(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}))$$

แผนภาพความร่วมมือ “Designer Drawing Diagram”



รูปที่ ข-6 แผนภาพความร่วมมือ “Designer Drawing Diagram”

จากตัวอย่างในรูปที่ ข-6 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

ใช้กฎข้อที่ 1 จะได้

$$\vec{e}_{EXT} = figure\_Info\_request1, figure\_Info\_request2, figure\_Info, change\_figure\_request, figure\_position, display\_figure, save\_request, save\_project, result1, save\_figure\_doc, result2, display\_save\_project\_result$$

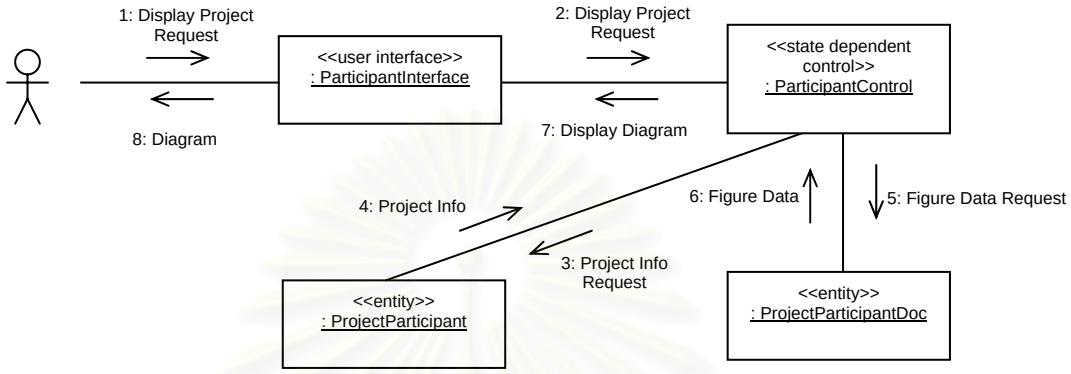
ต่อไปพิจารณาลำดับของข้อความที่ได้บรรยายไว้ในรูปที่ ข-6 จะได้ ดังนี้ คือ

$$figure\_Info\_request1, figure\_Info\_request2, figure\_Info, change\_figure\_request, figure\_Info\_request2, figure\_Info, figure\_position, display\_figure, save\_request, save\_project, result1, save\_figure\_doc, result2, display\_save\_project\_result$$

และจากกฎข้อที่ 5 จะสามารถแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัสได้ ดังนี้

$$
\begin{aligned}
&Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}) = \\
&\quad [event = figure\_Info\_request1]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) + \\
&\quad [event = figure\_Info\_request2]ex(x). \\
&\qquad ([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT})) + \\
&\quad [event = figure\_Info]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) + \\
&\quad [event = change\_figure\_request]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) + \\
&\quad [event = figure\_position]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) + \\
&\quad [event = display\_figure]ex(x). \\
&\qquad ([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT})) + \\
&\quad [event = save\_request]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) + \\
&\quad [event = save\_project]ex(x). \\
&\qquad ([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT})) + \\
&\quad [event = result1]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) + \\
&\quad [event = save\_figure\_doc]ex(x). \\
&\qquad ([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT})) + \\
&\quad [event = result2]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) + \\
&\quad [event = display\_save\_project\_result]ex(x). \\
&\qquad ([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}))
\end{aligned}
$$

แผนภาพความร่วมมือ “Participant Display Diagram”



รูปที่ ข-7 แผนภาพความร่วมมือ “Participant Display Diagram”

จากตัวอย่างในรูปที่ ข-7 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

ใช้กฎข้อที่ 1 จะได้

ให้ $\vec{e}_{EXT} = display\_project\_request, project\_Info\_request, project\_Info, figure\_data\_request, figure\_data, display\_diagram$

ต่อไปพิจารณาลำดับของข้อความที่ได้บรรยายไว้ในรูปที่ ข-7 จะได้ ดังนี้ คือ

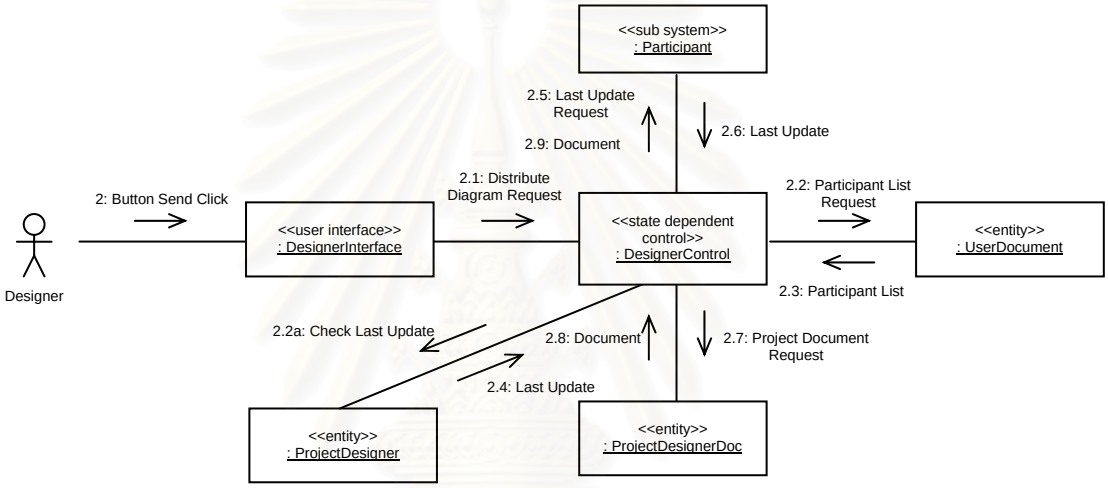
$$display\_project\_request, project\_Info\_request, project\_Info, figure\_data\_request, figure\_data, display\_diagram$$

และจากกฎข้อที่ 5 จะสามารถแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัสได้ ดังนี้

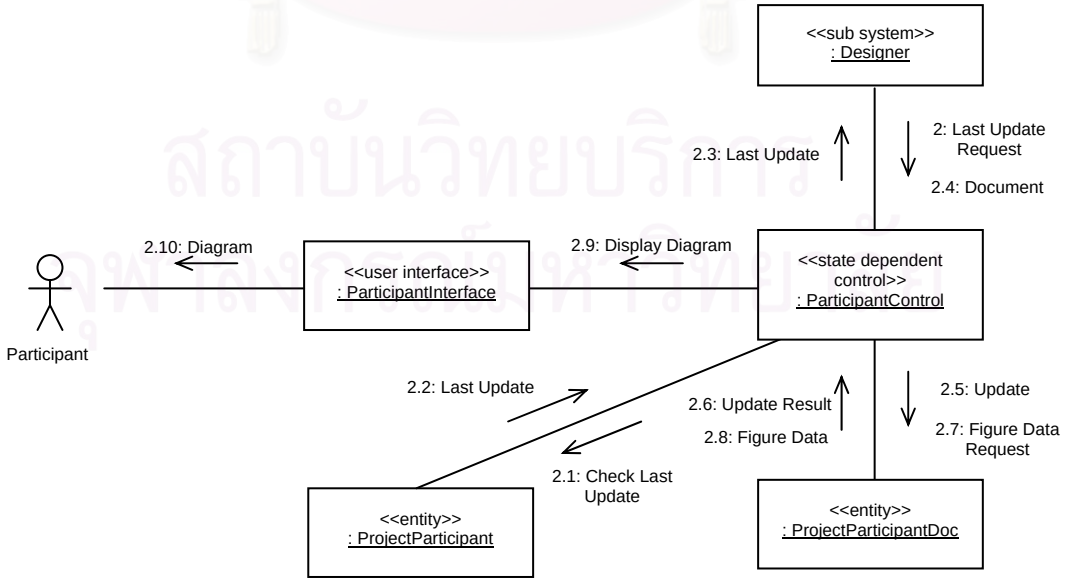
$$Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}) =$$
$$[event = display\_project\_request]\overline{enq}_{Par}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) +$$
$$[event = project\_Info\_request]ex(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT})) +$$
$$[event = project\_Info]\overline{enq}_{Par}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) +$$

$$[event = figure\_data\_request]ex(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT})) +$$
$$[event = figure\_data]\overline{enq}_{Par}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) +$$
$$[event = display\_diagram]ex(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}))$$

แผนภาพความร่วมมือ “Designer Distributed Diagram” และ “Participant Distributed Diagram”

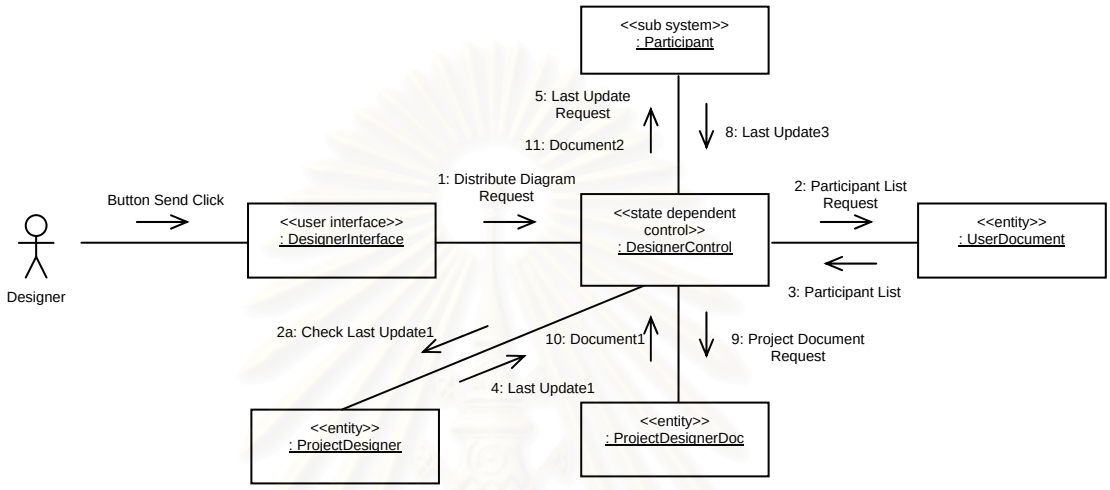


รูปที่ ข-8 แผนภาพความร่วมมือ ”Designer Distributed Diagram”

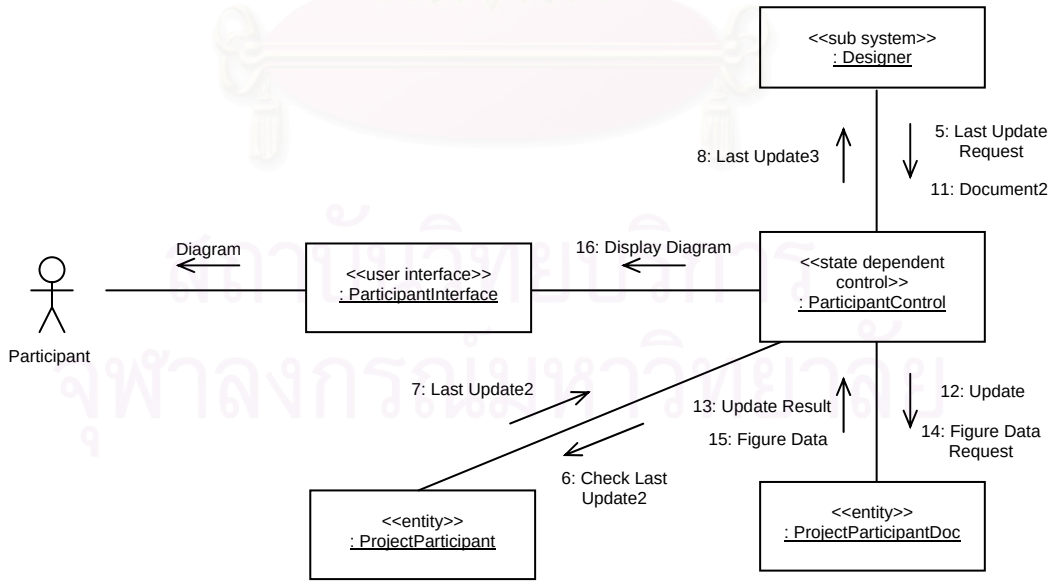


รูปที่ ข-9 แผนภาพความร่วมมือ ”Participant Distributed Diagram”

จากรูปที่ ข-8 และ ข-9 จะพบว่าเหตุการณ์ที่บรรยายไว้ด้วยแผนภาพความร่วมมือของทั้ง 2 แผนภาพ มีความต่อเนื่องกัน ดังนั้นให้นำแผนภาพทั้ง 2 มาพิจารณาร่วมกันเป็นกรณีเดียว ดังรูป ข-10 และ ข-11



รูปที่ ข-10 แผนภาพความร่วมมือ "Designer Distributed Diagram" ที่แก้ไขแล้ว



รูปที่ ข-11 แผนภาพความร่วมมือ "Participant Distributed Diagram" ที่แก้ไขแล้ว

จากตัวอย่างในรูปที่ ข-10 และ ข-11 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

ใช้กฎข้อที่ 1 จะได้

$$\vec{e}_{EXT} = distribute\_diagram\_request, participant\_list\_request,$$
$$check\_last\_update1, participant\_list, last\_update1,$$
$$check\_last\_update2, last\_update2, project\_doc\_request,$$
$$document1, update, update\_result, figure\_data\_request,$$
$$figure\_data, display\_diagram$$

$$\vec{e}_{INT} = last\_update\_request, last\_update3, document2$$

ต่อไปพิจารณาลำดับของข้อความที่ได้บรรยายไว้ในรูปที่ ข-10 และ ข-11 จะได้ ดังนี้ คือ

$$distribute\_diagram\_request, \theta, participant\_list\_request,$$
$$check\_last\_update1, \gamma, participant\_list, last\_update1,$$
$$last\_update\_request, check\_last\_update2, last\_update2,$$
$$last\_update3, project\_doc\_request, document1, document2, update,$$
$$update\_result, figure\_data\_request, figure\_data, display\_diagram$$

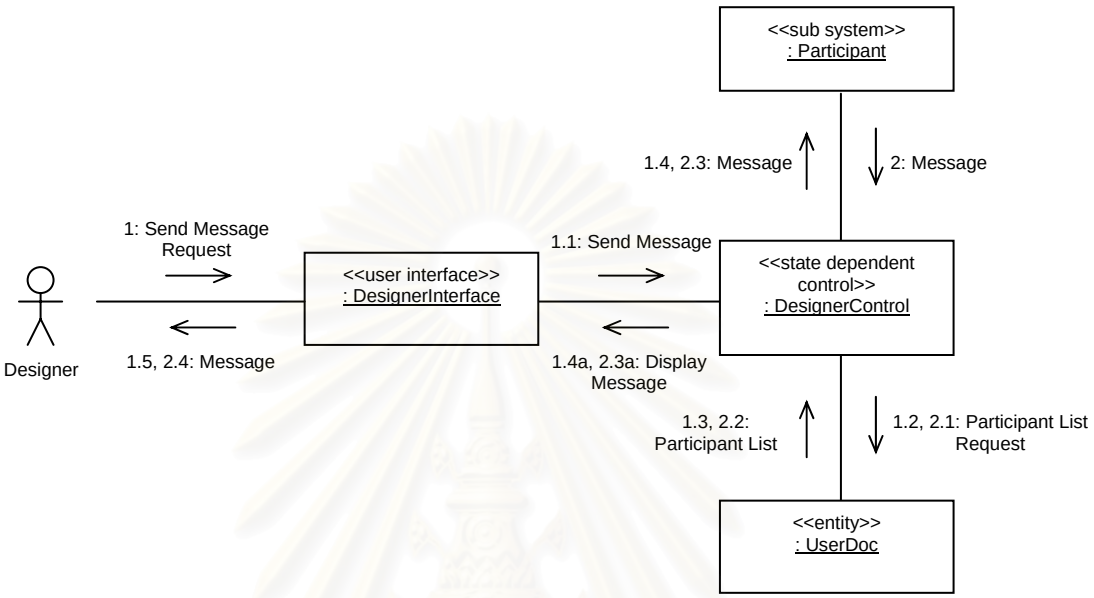
และจากกฎข้อที่ 5 และ 6 จะสามารถแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัสได้ ดังนี้

$$Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}) =$$
$$[event = distribute\_diagram\_request]\overline{enq}_{Des}\langle event\rangle.$$
$$Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) +$$
$$[event = participant\_list]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) +$$
$$[event = check\_last\_update1]ex(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT})) +$$
$$[event = last\_update1]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) +$$
$$[event = check\_last\_update2]ex(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT})) +$$
$$[event = last\_update2]\overline{enq}_{Par}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) +$$
$$[event = project\_doc\_request]ex(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT})) +$$
$$[event = document1]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) +$$

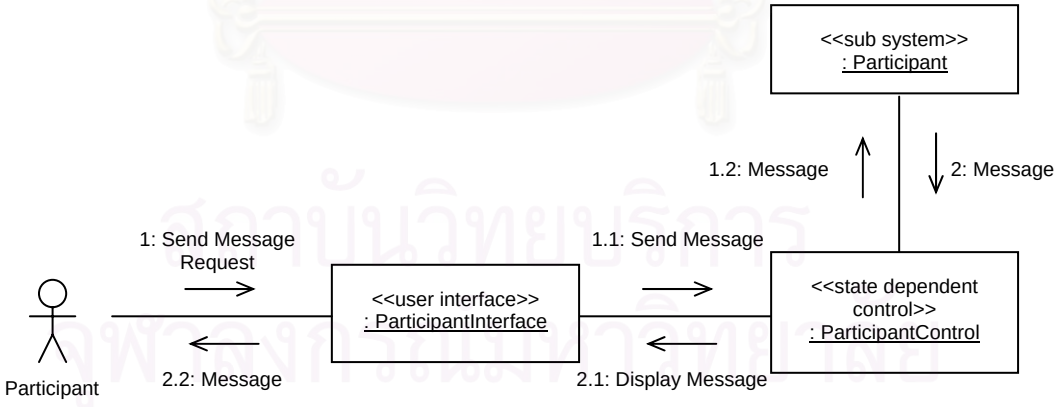
$$[event = update]ex(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT})) +$$
$$[event = update\_result]\overline{enq}_{Par}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) +$$
$$[event = figure\_data\_request]ex(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT})) +$$
$$[event = figure\_data)]\overline{enq}_{Par}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}) +$$
$$[event = display\_diagram]ex(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}))$$

$$Scenario''(event, \vec{in}, \vec{e}_{INT}, \vec{e}_{SYS}) =$$
$$[event = last\_update\_request]in_{DesPar}(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$\sum_{e \in \{\vec{e}_{SYS}\} \setminus \{event\}} [x = e]Error) +$$
$$[event = last\_update3]in_{ParDes}(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$\sum_{e \in \{\vec{e}_{SYS}\} \setminus \{event\}} [x = e]Error) +$$
$$[event = document2]in_{DesPar}(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$\sum_{e \in \{\vec{e}_{SYS}\} \setminus \{event\}} [x = e]Error)$$

แผนภาพความร่วมมือ “Designer Chat” และ “Participant Chat”



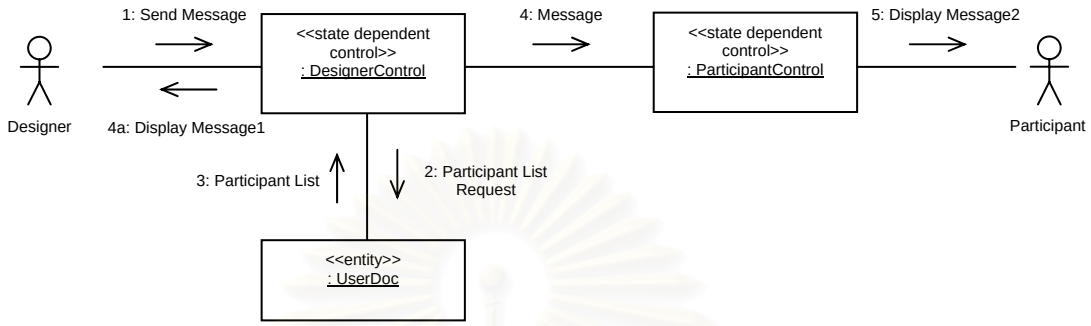
รูปที่ ข-12 แผนภาพความร่วมมือ ”Designer Chat”



รูปที่ ข-13 แผนภาพความร่วมมือ ”Participant Chat”

จากรูปที่ ข-12 และ ข-13 จะพบว่าเหตุการณ์ที่บรรยายไว้ด้วยแผนภาพความร่วมมือของทั้ง 2 แผนภาพ มีความต่อเนื่องกัน ดังนั้นให้นำแผนภาพทั้ง 2 มาพิจารณาร่วมกัน

โดยสามารถแบ่งได้เป็น 2 กรณี คือ กรณีที่ Designer ส่งข้อความ และกรณีที่ Participant ส่งข้อความ ดังรูป ข-14 และ ข-15 ตามลำดับ



รูปที่ ข-14 แผนภาพความร่วมมือประกอบของ “Participant Chat” และ “Designer Chat”

จากตัวอย่างในรูปที่ ข-14 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

ใช้กฎข้อที่ 1 จะได้

$$\vec{e}_{EXT} = send\_message, participant\_list\_request, participant\_list, display\_message1, display\_message2$$

$$\vec{e}_{INT} = message$$

ต่อไปพิจารณาลำดับของข้อความที่ได้บรรยายไว้ในรูปที่ ข-14 จะได้ ดังนี้ คือ

$$send\_message, participant\_list\_request, participant\_list, \theta, message, display\_message1, \gamma, display\_message2$$

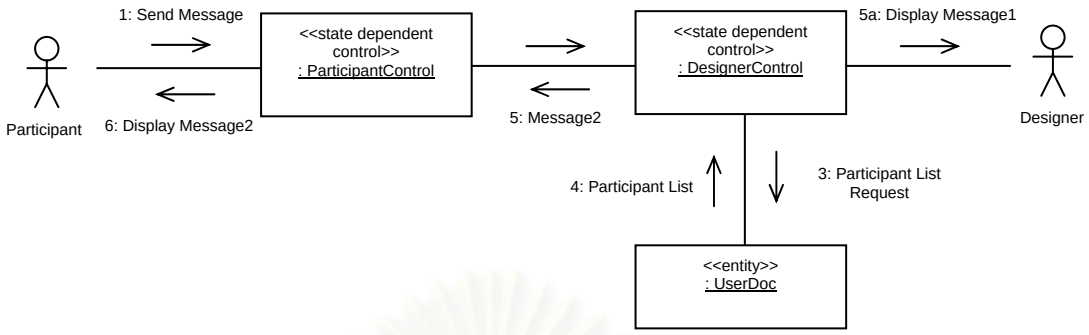
และจากกฎข้อที่ 5 ถึงกฎข้อที่ 8 จะสามารถแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัสได้ ดังนี้

$$Scenario'(event, ex, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) =$$
$$[event = send\_message]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = participant\_list\_request]ex(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error) +$$
$$[event = participant\_list]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = display\_message2]ex(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error)$$

$$Scenario_0'''\ (empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) = \overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$
$$Scenario_1'''\ (empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, event)$$

$$Scenario_n'''\ (empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, x_1, x_2, ..., x_n) =$$
$$\overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$
$$(\sum_{e\in\{\vec{e}_{INT}\cup\vec{e}_{EXT}\}}[event = e]$$
$$Scenario_{n+1}'''(empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, x_1, x_2, ..., x_n, event) +$$
$$[event = \gamma](Scenario_n^4(empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, \overrightarrow{ack}, x_1, x_2, ..., x_n)\ |$$
$$([ack_1 = pos]([ack_2 = pos](...([ack_n = pos]$$
$$Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$$

$$Scenario_n^4(\overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, \overrightarrow{ack}, x_1, x_2, ..., x_n, pos) =$$
$$([x_1 = message]in_{DesPar}(x).$$
$$([x = x_1]\overline{ack}_1\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_1\}}[x = e]Error) +$$
$$[x_1 = display\_message1]ex(x).$$
$$([x = x_1]\overline{ack}_1\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_1\}}[x = e]Error) + ...)\ |$$
$$([x_2 = message]in_{DesPar}(x).$$
$$([x = x_2]\overline{ack}_2\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_2\}}[x = e]Error) +$$
$$[x_2 = display\_message1]ex(x).$$
$$([x = x_2]\overline{ack}_2\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_2\}}[x = e]Error) + ...)\ |$$
$$([x_3 = message]... + [x_3 = display\_message1]...)\ |\ (...)\ |$$
$$([x_n = message]in_{DesPar}(x).$$
$$([x = x_n]\overline{ack}_n\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_2\}}[x = e]Error) +$$
$$[x_n = display\_message1]ex(x).$$
$$([x = x_n]\overline{ack}_n\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_n\}}[x = e]Error)$$



รูปที่ ข-15 แผนภาพความร่วมมือประกอบของ “Participant Chat” และ “Designer Chat”

จากตัวอย่างในรูปที่ ข-15 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

ใช้กฎข้อที่ 1 จะได้

$$\vec{e}_{EXT} = send\_message, participant\_list\_request, participant\_list, display\_message1, display\_message2$$

$$\vec{e}_{INT} = message1, message2$$

ต่อไปพิจารณาลำดับของข้อความที่ได้บรรยายไว้ในรูปที่ ข-15 จะได้ ดังนี้ คือ

$$send\_message, message1, participant\_list\_request, participant\_list, \theta, message2, display\_message1, \gamma, display\_message2$$

และจากกฎข้อที่ 5 ถึงกฎข้อที่ 8 จะสามารถแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัสได้ ดังนี้

$$\begin{aligned}
&Scenario'(event, ex, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) = \\
&\quad [event = send_message]\overline{enq}_{Par}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) + \\
&\quad [event = participant_list_request]ex(x). \\
&\qquad ([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) + \\
&\qquad \sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error) + \\
&\quad [event = participant_list]\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) + \\
&\quad [event = display_message2]ex(x). \\
&\qquad ([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) \\
&\qquad \sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error)
\end{aligned}$$

$$Scenario''(event, \vec{in}, \vec{e}_{INT}, \vec{e}_{SYS}) =$$

$$[event = message1]in_{ParDes}(x).$$

$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$

$$\sum_{e \in \{\vec{e}_{SYS}\} \setminus \{event\}} [x = e]Error)$$

$$Scenario_0'''(empty, deq, \vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) =$$

$$\overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$

$$Scenario_1'''(empty, deq, \vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, event)$$

$$Scenario_n'''(empty, deq, \vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, x_1, x_2, ..., x_n) =$$

$$\overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$

$$(\sum_{e \in \{\vec{e}_{INT} \cup \vec{e}_{EXT}\}} [event = e]$$

$$Scenario'''_{n+1}(empty, deq, \vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, x_1, x_2, ..., x_n, event) +$$

$$[event = \gamma](Scenario_n^4(empty, deq, \vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, \overrightarrow{ack}, x_1, x_2, ..., x_n) |$$

$$([ack_1 = pos]([ack_2 = pos](...([ack_n = pos]$$

$$Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$$

$$Scenario_n^4(\vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, \overrightarrow{ack}, x_1, x_2, ..., x_n, pos) =$$

$$([x_1 = message2]in_{DesPar}(x).$$

$$([x = x_1]\overline{ack}_1\langle pos\rangle +$$

$$\sum_{e \in \{\vec{e}_{SYS}\} \setminus \{x_1\}} [x = e]Error) +$$

$$[x_1 = display\_message1]ex(x).$$

$$([x = x_1]\overline{ack}_1\langle pos\rangle +$$

$$\sum_{e \in \{\vec{e}_{SYS}\} \setminus \{x_1\}} [x = e]Error) + ...) |$$

$$([x_2 = message2]in_{DesPar}(x).$$

$$([x = x_2]\overline{ack}_2\langle pos\rangle +$$

$$\sum_{e \in \{\vec{e}_{SYS}\} \setminus \{x_2\}} [x = e]Error) +$$

$$[x_2 = display\_message1]ex(x).$$

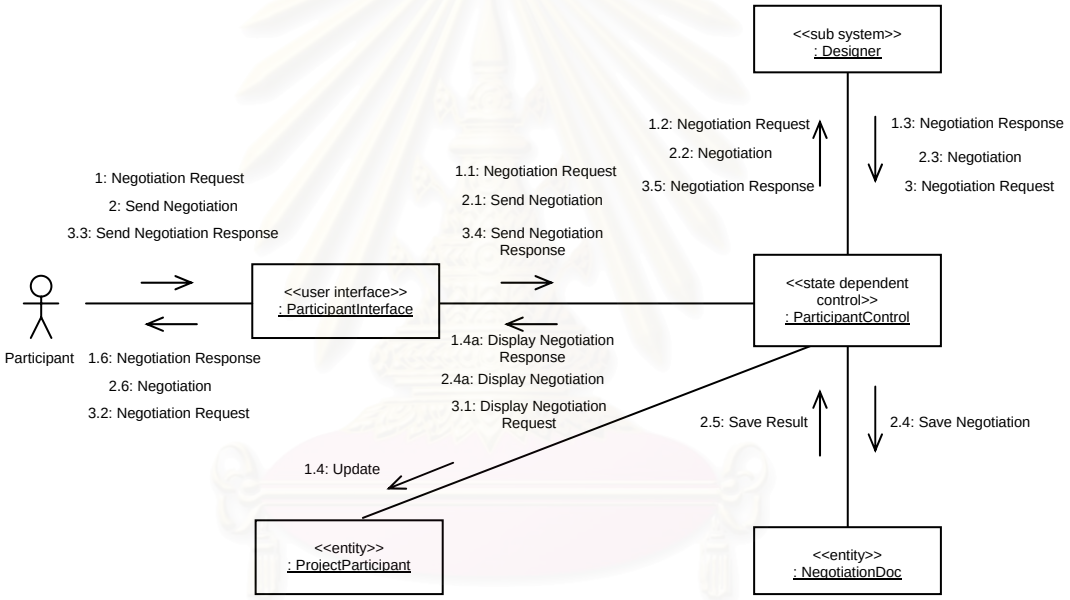
$$([x = x_2]\overline{ack}_2\langle pos\rangle +$$

$$\sum_{e \in \{\vec{e}_{SYS}\} \setminus \{x_2\}} [x = e]Error) + ...) |$$

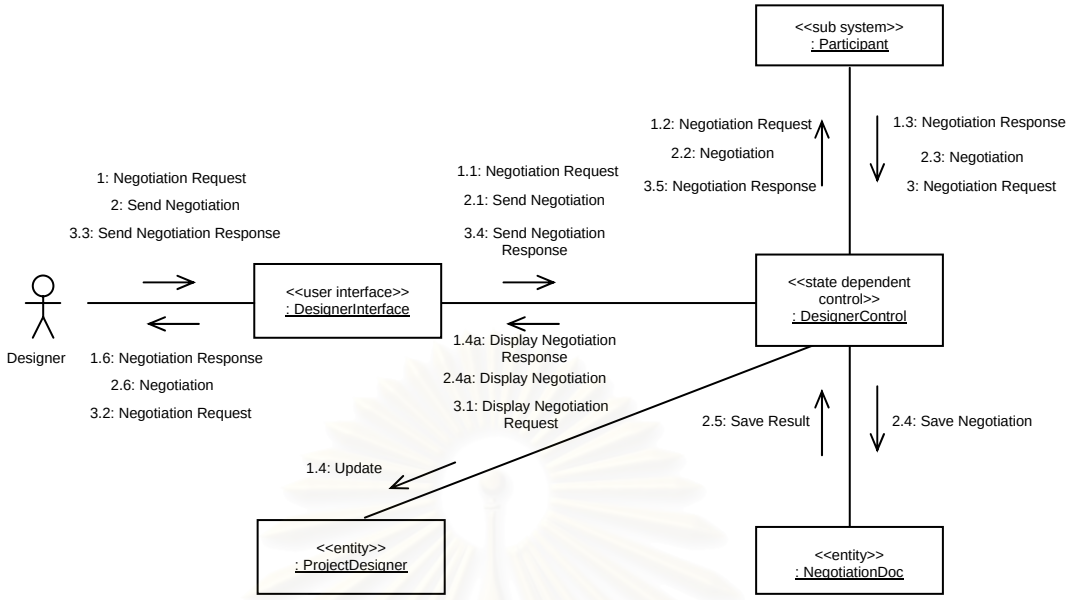
$$([x_3 = message2]... + [x_3 = display\_message1]...) | (...) |$$

$$([x_n = message2]in_{DesPar}(x).$$
$$([x = x_n]\overline{ack}_n\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_2\}}[x = e]Error) +$$
$$[x_n = display\_message1]ex(x).$$
$$([x = x_n]\overline{ack}_n\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_n\}}[x = e]Error)$$

แผนภาพความร่วมมือ “Designer Negotiation” และ “Participant Negotiation”

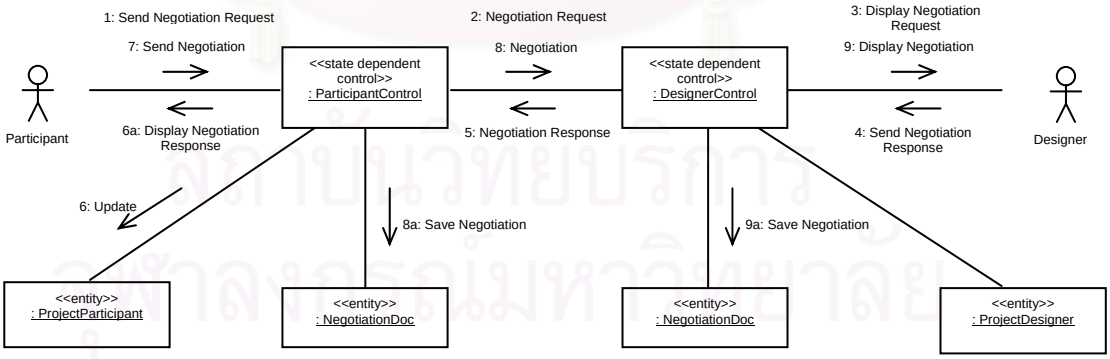


รูปที่ ข-16 แผนภาพความร่วมมือ ”Participant Negotiation”



รูปที่ ข-17 แผนภาพความร่วมมือ "Designer Negotiation"

จากรูปที่ ข-16 และ ข-17 จะพบว่าเหตุการณ์ที่บรรยายไว้ด้วยแผนภาพความร่วมมือของทั้ง 2 แผนภาพ มีความต่อเนื่องกัน ดังนั้นให้นำแผนภาพทั้ง 2 มาพิจารณาร่วมกันเป็นกรณีเดียว ดังรูป ข-18 ซึ่งเป็นเหตุการณ์ที่ Participant เป็นผู้ส่งคำขอการเจรจา (negotiation request) และส่งคำเจรจา (negotiation)



รูปที่ ข-18 แผนภาพความร่วมมือประกอบของ "Participant Negotiation" และ "Designer Negotiation"

จากตัวอย่างในรูปที่ ข-18 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

ใช้กฎข้อที่ 1 จะได้

$$\vec{e}_{EXT} = send\_negotiation\_request, display\_negotiation\_request,$$
$$send\_negotiation\_response, update, display\_negotiation\_response,$$
$$send\_negotiation, display\_negotiation, save\_negotiation$$

$$\vec{e}_{INT} = negotiation\_request, negotiation\_response, negotiation$$

ต่อไปพิจารณาลำดับของข้อความที่ได้บรรยายไว้ในรูปที่ ข-18 จะได้ ดังนี้ คือ

$$send\_negotiation\_request, negotiation\_request,$$
$$display\_negotiation\_request, send\_negotiation\_response,$$
$$negotiation\_response, \theta, update, display\_negotiation\_response, \gamma,$$
$$send\_negotiation, display\_negotiation, save\_negotiation$$

และจากกฎข้อที่ 5 ถึงกฎข้อที่ 8 จะสามารถแปลงแผนภาพความร่วมมือไปเป็นไพแคลคูลัสได้ ดังนี้

$$Scenario'(event, ex, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) =$$
$$[event = send\_negotiation\_request]$$
$$\overline{enq}_{Par}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = display\_negotiation\_request]ex(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error) +$$
$$[event = send\_negotiation\_response]$$
$$\overline{enq}_{Des}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = send\_negotiation]$$
$$\overline{enq}_{Par}\langle event\rangle.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$

$$Scenario''(event, \overrightarrow{in}, \vec{e}_{INT}, \vec{e}_{SYS}) =$$
$$[event = negotiation\_request]in_{ParDes}(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{event\}}[x = e]Error) +$$

$$[event = negotiation\_response]in_{DesPar}(x).$$
$$([x = event]Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$\sum_{e \in \{\vec{e}_{SYS}\} \setminus \{event\}} [x = e]Error) +$$

$$Scenario_0'''(empty, deq, \vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) =$$
$$\overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$
$$Scenario_1'''(empty, deq, \vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, event)$$

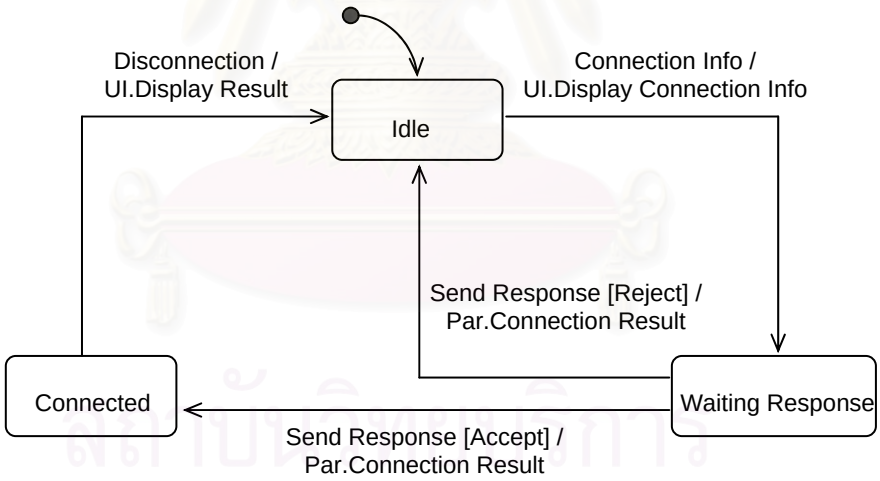
$$Scenario_n'''(empty, deq, \vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, x_1, x_2, ..., x_n) =$$
$$\overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$
$$(\sum_{e \in \{\vec{e}_{INT} \cup \vec{e}_{EXT}\}} [event = e]$$
$$Scenario'''_{n+1}(empty, deq, \vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, x_1, x_2, ..., x_n, event) +$$
$$[event = \gamma](Scenario_n^4(empty, deq, \vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, \overrightarrow{ack}, x_1, x_2, ..., x_n) |$$
$$([ack_1 = pos]([ack_2 = pos](...([ack_n = pos]$$
$$Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$$

$$Scenario_n^4(\vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, \overrightarrow{ack}, x_1, x_2, ..., x_n, pos) =$$
$$([x_1 = update]ex(x).$$
$$([x = x_1]\overline{ack}_1\langle pos\rangle +$$
$$\sum_{e \in \{\vec{e}_{SYS}\} \setminus \{x_1\}} [x = e]Error) +$$
$$[x_1 = display\_negotiation\_response]ex(x).$$
$$([x = x_1]\overline{ack}_1\langle pos\rangle +$$
$$\sum_{e \in \{\vec{e}_{SYS}\} \setminus \{x_1\}} [x = e]Error) +$$
$$[x_1 = negotiation]in_{ParDes}(x).$$
$$([x = x_1]\overline{ack}_1\langle pos\rangle +$$
$$\sum_{e \in \{\vec{e}_{SYS}\} \setminus \{x_1\}} [x = e]Error) +$$
$$[x_1 = save\_negotiation]ex(x).$$
$$([x = x_1]\overline{ack}_1\langle pos\rangle +$$
$$\sum_{e \in \{\vec{e}_{SYS}\} \setminus \{x_1\}} [x = e]Error) +$$
$$[x_1 = display\_negotiation]ex(x).$$
$$([x = x_1]\overline{ack}_1\langle pos\rangle +$$
$$\sum_{e \in \{\vec{e}_{SYS}\} \setminus \{x_1\}} [x = e]Error) +$$

$$[x_1 = save\_negotiation]ex(x).$$
$$([x = x_1]\overline{ack}_1\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_1\}}[x = e]Error) + ...) \mid ... \mid$$
$$([x_n = update]ex(x).$$
$$([x = x_n]\overline{ack}_n\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_n\}}[x = e]Error) + ... +$$
$$[x_n = save\_negotiation]ex(x).$$
$$([x = x_n]\overline{ack}_n\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{SYS}\}\setminus\{x_n\}}[x = e]Error)$$

การแปลงแผนภาพสถานะ

แผนภาพสถานะของ Designer



รูปที่ ข-19 Top-level statechart for Designer control

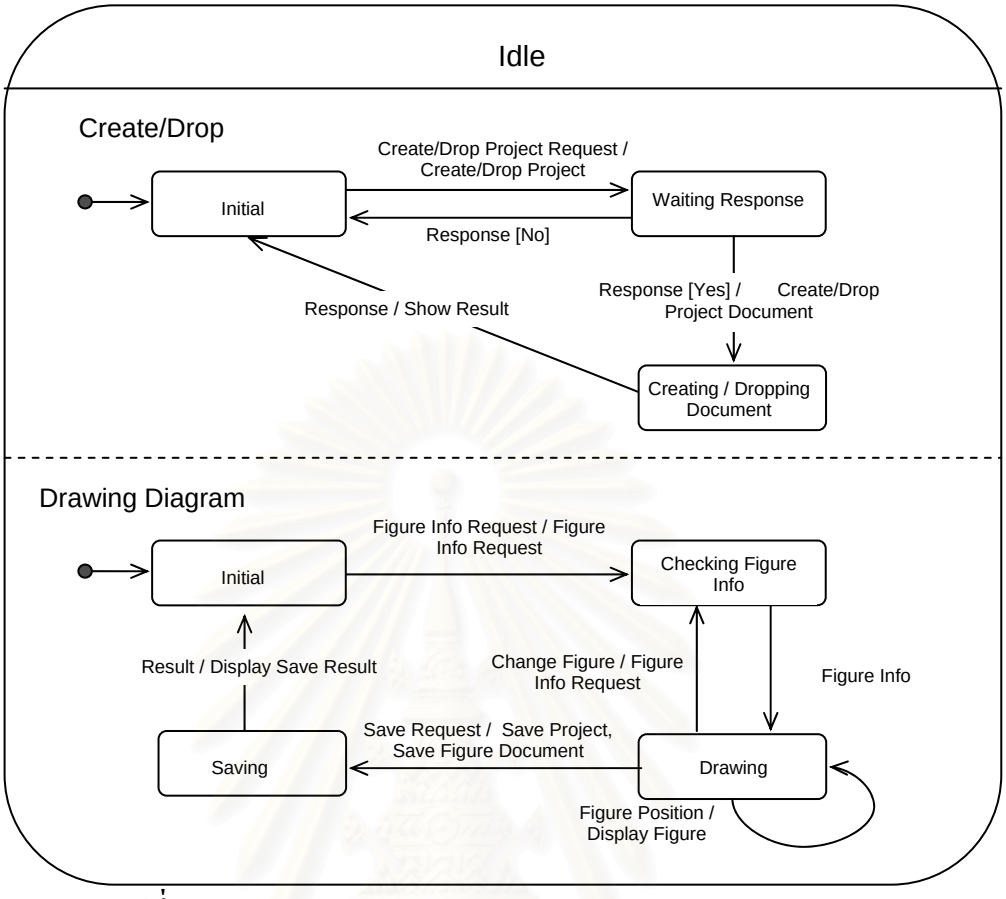
จากรูปที่ ข-19 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

ให้ $\vec{e}_{Des} = connection\_Info, display\_connection\_Info$
$send\_response, connection\_result,$
$disconnection, display\_result$

$$Idle(step, event_S, \vec{e}_{Des}, \overrightarrow{enq}, ex, \vec{in}) = event_s(x).$$
$$([x = connection\_Info]\overline{enq}_{Par}\langle display\_connection\_Info\rangle.$$
$$\overline{ex}\langle display\_connection\_Info\rangle.\overline{step}.$$
$$Waitng\_response(step, g_1, event_S, \vec{e}_{Des}, \overrightarrow{enq}, ex, \vec{in}) +$$
$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{connection\_Info\}}[x = e]Error)$$

$$Waiting\_response(step, g_1, event_S, \vec{e}_{Des}, \overrightarrow{enq}, ex, \vec{in}) = event_s(x).$$
$$([x = send\_response]\overline{g}_1\langle z\rangle.z(y).$$
$$([y = accept]\overline{in}_{DesPar}\langle connection\_result\rangle.\overline{enq}_{Par}\langle connection\_result\rangle.$$
$$\overline{step}.Connected(step, event_s, \vec{e}_{Par}, \overrightarrow{enq}, ex, \vec{in}) +$$
$$[y = reject]\overline{in}_{DesPar}\langle connection\_result\rangle.\overline{enq}_{Par}\langle connection\_result\rangle.$$
$$\overline{step}.Idle(step, event_s, \vec{e}_{Par}, \overrightarrow{enq}, ex, \vec{in})) +$$
$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{send\_response\}}[x = e]Error)$$

$$Connected(step, event_S, \vec{e}_{Des}, \overrightarrow{enq}, ex, \vec{in}) = event_s(x).$$
$$([x = disconnection]\overline{ex}\langle display\_result\rangle.\overline{step}.Idle(step, event_S, \vec{e}_{Des}, \overrightarrow{enq}, \vec{in}) +$$
$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{disconnection\}}[x = e]Error)$$



รูปที่ ข-20 แผนภาพสถานะของ Designer Control: Idle superstate

จากรูปที่ ข-20 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

$$Idle(step, event_s, \vec{e}_{Des}, ...) \mid Create\_Drop(event_{v_1}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in})$$
$$\mid Drawing\_Diagram(step, event_{v_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in})$$

และจากกฎข้อที่ 6 สามารถบรรยายสถานะประกอบ *Idle* และสถานะย่อย *Create_Drop* และ *Drawing_Diagram* ได้ดังนี้

$$\vec{e}_{Des} = create\_drop\_project\_request, create\_drop\_project$$
$$response1, create\_drop\_project\_doc, response2,$$
$$show\_result$$

$$Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg) =$$
$$event_s(x_1).(\nu\ ack_1\ ack_2)\ (\overline{event}_{V1}\langle x_1\ ack_1\rangle.ack_1(y_1).$$
$$([y_1 = pos]($$
$$[x_1 = create\_drop\_proj\_request]\overline{step}.$$

$$(Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg) |$$
$$Waiting\_response(event_{V_1}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) +$$

$$[x_1 = response]\overline{step}.$$
$$(Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg) |$$
$$Creating\_Dropping\_Doc(event_{V_1}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))) +$$

$$[y_1 = neg] Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg)).$$

$$\overline{event}_{V2}\langle x_1\ ack_2\rangle.ack_2(y_2).$$
$$([y_2 = pos]($$
$$[x_1 = figure\_Info\_request1]\overline{step}.$$
$$(Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg) |$$
$$Checking\_figure\_Info(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) +$$

$$[x_1 = figure\_Info]\ \overline{step}.$$
$$(Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg) |$$
$$Drawing(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) +$$

$$[x_1 = figure\_position]\ \overline{step}.$$
$$(Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg) |$$
$$Drawing(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) +$$

$$[x_1 = change\_figure]\ \overline{step}.$$
$$(Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg) |$$
$$Checking\_figure\_Info(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) +$$

$$[x_1 = save\_request]\ \overline{step}.$$
$$(Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg) |$$
$$Saving(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) +$$

$$[x_1 = result]\ \overline{step}.$$
$$(Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg) |$$
$$Drawing\_diagram(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))) +$$

$$[y_2 = neg]\,Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg)).$$

$$\overline{step}.([y_1 = pos]([y_2 = pos]([x_1 = connection\_Info]$$
$$Waiting\_response(step, event_S, \vec{e}_{Des}, \overrightarrow{enq}, ex, \vec{in})))$$

$$Create\_Drop(event_{V_1}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$
$$event_{V_1}(x\ ack).$$
$$([x = create\_drop\_proj\_request]\overline{in}_{DesPar}\langle create\_drop\_proj\_request\rangle.$$
$$\overline{enq}_{Par}\langle create\_drop\_proj\_request\rangle.\ \overline{ack}\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{create\_drop\_proj\_request\}}[x = e]\overline{ack}\langle neg\rangle.$$
$$Create\_Drop(event_{V_1}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

$$Waiting\_response(event_{V_1}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) = event_{V_1}(x\ ack).($$
$$[x = response1]($$
$$[y = yes]\overline{ex}\langle create\_drop\_proj\_doc\rangle.\overline{ack}\langle pos\rangle +$$
$$[y = no]\overline{ack}\langle neg\rangle.Waiting\_response(event_{V_1}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) +$$
$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{response1\}}[x = e]\overline{ack}\langle neg\rangle.$$
$$Waiting\_response(event_{V_1}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

$$Creating\_Dropping\_Doc(event_{V_1}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$
$$event_{V_1}(x\ ack).$$
$$([x = response2]\,\overline{ex}\langle show\_result\rangle.\ \overline{ack}\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{response2\}}[x = e]\overline{ack}\langle neg\rangle.$$
$$Creating\_Dropping\_Doc(event_{V_1}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

$$Drawing\_diagram(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$
$$event_{V_2}(x\ ack).$$
$$([x = figure\_Info\_request1]\overline{ex}\langle figure\_Info\_request2\rangle.\overline{ack}\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{figure\_Info\_request1\}}[x = e]\overline{ack}\langle neg\rangle.$$
$$Drawing\_diagram(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

$$Checking\_figure\_Info(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$

$$event_{V_2}(x\ ack).$$

$$([x = figure\_Info]\ \overline{ack}\langle pos\rangle +$$

$$\sum_{e \in \{\vec{e}_{Des}\} \setminus \{figure\_Info\}} [x = e]\overline{ack}\langle neg\rangle.$$

$$Checking\_figure\_Info(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

$$Drawing(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$

$$event_{V_2}(x\ ack).($$

$$([x = figure\_position]\overline{ex}\langle display\_figure\rangle.\overline{ack}\langle pos\rangle +$$

$$[x = saving\_request]\overline{ex}\langle save\_project\rangle.\overline{ex}\langle save\_figure\_doc\rangle.\overline{ack}\langle pos\rangle) +$$

$$[x = change\_figure]\overline{ex}\langle figure\_Info\_request\rangle.\overline{ack}\langle pos\rangle) +$$

$$\sum_{e \in \{\vec{e}_{Des}\} \setminus \{figure\_position, saving\_request, change\_figure\}} [x = e]\overline{ack}\langle neg\rangle.$$

$$Drawing(event_V, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

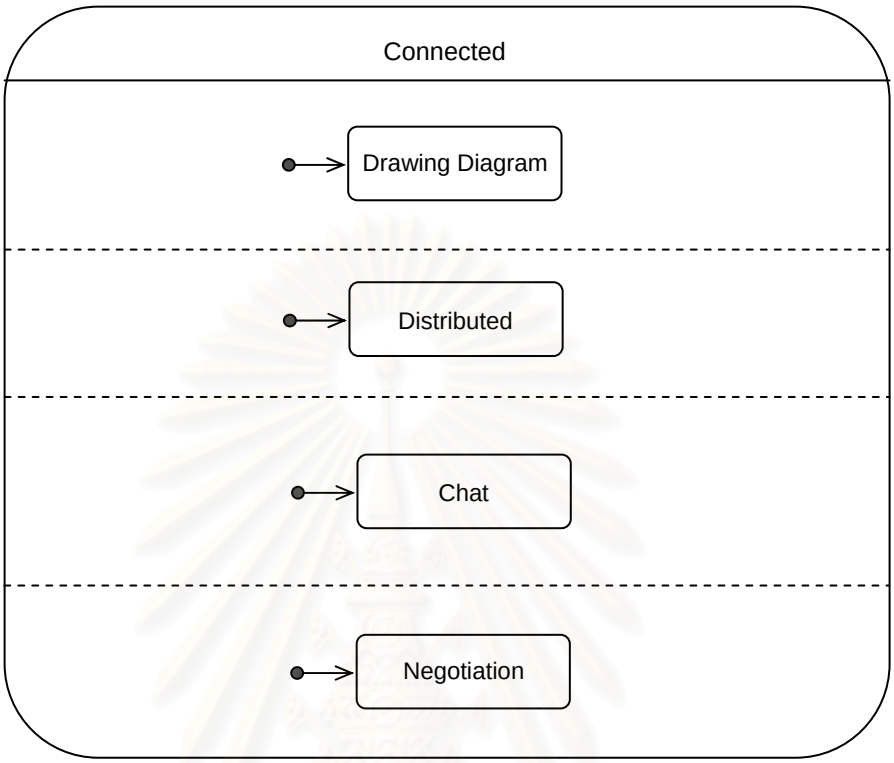
$$Saving(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$

$$event_{V_2}(x\ ack).$$

$$([x = result]\overline{ex}\langle display\_save\_result\rangle.\overline{ack}\langle pos\rangle +$$

$$\sum_{e \in \{\vec{e}_{Des}\} \setminus \{result\}} [x = e]\overline{ack}\langle neg\rangle.$$

$$Saving(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$



รูปที่ ข-21 แผนภาพสถานะของ Designer Control: Connected superstate

จากรูปที่ ข-21 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

จากกฎข้อที่ 4 จะได้

$$Connected(step, event_s, \vec{e}_{Des}, ...) \mid Drawing\_Diagram(event_{v_1}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) \mid$$
$$Distributed(event_{v_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) \mid Chat(event_{v_3}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) \mid$$
$$Negotiation(event_{v_4}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in})$$

และจากกฎข้อที่ 6 สามารถบรรยายสถานะประกอบ *Connected* และสถานะย่อย *Drawing_Diagram* *Distributed* *Chat* และ *negotiation* ได้ดังนี้

$$Connected(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, event_{V_3}, event_{V_4}, pos, neg) =$$
$$event_s(x_1).(\nu\ ack_1\ ack_2\ ack_3\ ack_4)\ (\overline{event}_{V1}\langle x_1\ ack_1\rangle.ack_1(y_1).$$
$$([y_1 = pos]($$
$$[x_1 = figure\_Info\_request1]\overline{step}.$$

$$(Connected(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, event_{V_3}, event_{V_4}, pos, neg) |$$
$$Checking\_figure\_Info(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) +$$

$$[x_1 = figure\_Info]\ \overline{step}.$$
$$(Connected(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, event_{V_3}, event_{V_4}, pos, neg) |$$
$$Drawing(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) +$$

$$[x_1 = figure\_position]\ \overline{step}.$$
$$(Connected(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, event_{V_3}, event_{V_4}, pos, neg) |$$
$$Drawing(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) +$$

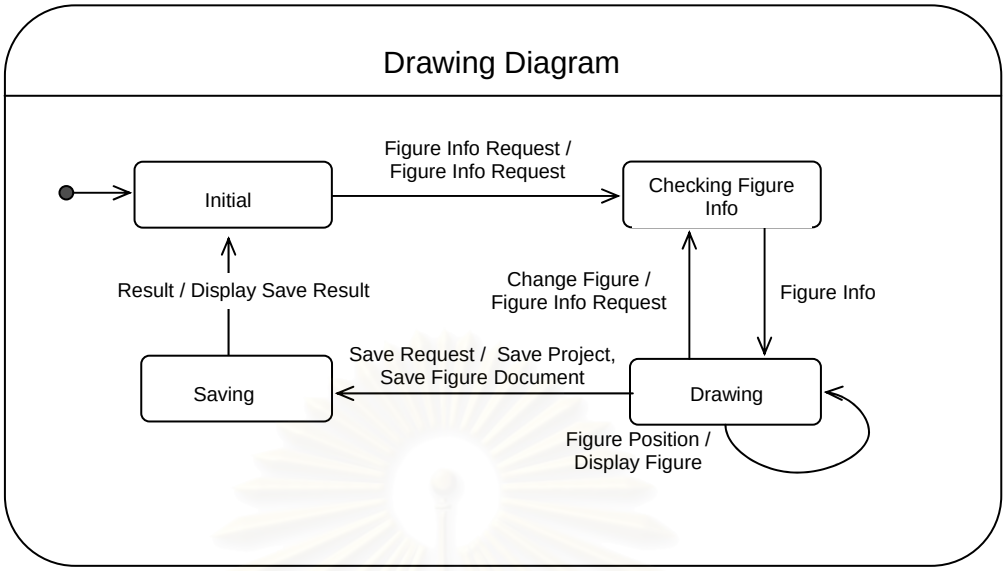
$$[x_1 = change\_figure]\ \overline{step}.$$
$$(Connected(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, event_{V_3}, event_{V_4}, pos, neg) |$$
$$Checking\_figure\_Info(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) +$$

$$[x_1 = save\_request]\ \overline{step}.$$
$$(Connected(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, event_{V_3}, event_{V_4}, pos, neg) |$$
$$Saving(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) +$$

$$[x_1 = result]\ \overline{step}.$$
$$(Connected(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, event_{V_3}, event_{V_4}, pos, neg) |$$
$$Drawing\_diagram(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))) +$$

$$[y_1 = neg] Connected(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, event_{V_3}, event_{V_4}, pos, neg)).$$

$$\overline{event}_{V2}\langle x_1\ ack_2\rangle.ack_2(y_2)...$$
$$\overline{event}_{V3}\langle x_1\ ack_3\rangle.ack_3(y_3)...$$
$$\overline{event}_{V4}\langle x_1\ ack_4\rangle.ack_4(y_4)...$$
$$\overline{step}.([y_1 = pos]([y_2 = pos]([y_3 = pos]([y_4 = pos]([x_1 = disconnection]$$
$$Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, event_{V_2}, pos, neg)$$



รูปที่ ข-22 แผนภาพสถานะของ Designer Control: Drawing Diagram superstate

$$Drawing\_diagram(event_{V_1}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$
$$event_{V_1}(x\ ack).$$
$$([x = figure\_Info\_request1]\overline{ex}\langle figure\_Info\_request2\rangle.\overline{ack}\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{figure\_Info\_request1\}}[x = e]\overline{ack}\langle neg\rangle.$$
$$Drawing\_diagram(event_{V_1}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in})$$

$$Checking\_figure\_Info(event_{V_1}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$
$$event_{V_1}(x\ ack).$$
$$([x = figure\_Info]\ \overline{ack}\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{figure\_Info\}}[x = e]\overline{ack}\langle neg\rangle.$$
$$Checking\_figure\_Info(event_{V_1}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$

$$Drawing(event_{V_1}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$
$$event_{V_1}(x\ ack).($$
$$([x = figure\_position]\overline{ex}\langle display\_figure\rangle.\overline{ack}\langle pos\rangle +$$
$$[x = saving\_request]\overline{ex}\langle save\_project\rangle.\overline{ex}\langle save\_figure\_doc\rangle.\overline{ack}\langle pos\rangle) +$$
$$[x = change\_figure]\overline{ex}\langle figure\_Info\_request\rangle.\overline{ack}\langle pos\rangle) +$$
$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{figure\_position, saving\_request, change\_figure\}}[x = e]\overline{ack}\langle neg\rangle.$$
$$Drawing(event_{V_1}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

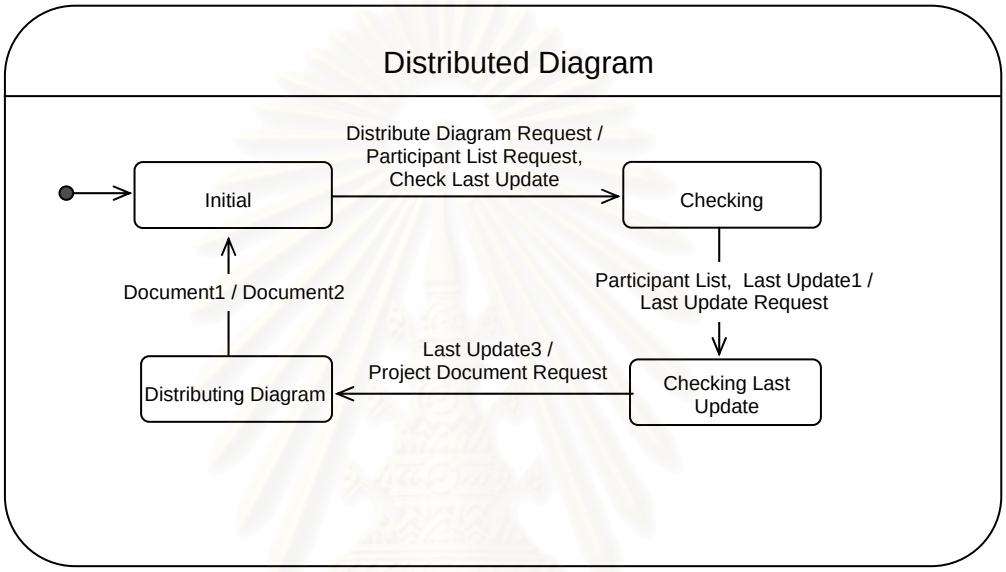
$$Saving(event_{V_1}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$

$$event_{V_1}(x\ ack).$$

$$([x = result]\overline{ex}\langle display\_save\_result\rangle.\overline{ack}\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{result\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Saving(event_{V_1}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$



รูปที่ ข-23 แผนภาพสถานะของ Designer Control: Distributed Diagram superstate

$$Distributed(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$

$$event_{V_2}(x\ ack).$$

$$([x = distribute\_diagram\_request]\overline{ex}\langle participant\_list\_request\rangle.$$

$$\overline{ex}\langle check\_last\_update1\rangle.\overline{ack}\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{distribute\_diagram\_request\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Distributed(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

$$Checking(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$

$$event_{V_2}(x\ ack).$$

$$(\sum_{e\in\{participant\_list, last\_update1\}}[x = e]\overline{in}_{DesPar}\langle last\_update\_request\rangle.$$

$$\overline{enq}_{Par}\langle last\_update\_request\rangle.\overline{ack}\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{participant\_list, last\_update1\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Checking(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

$$Checking\_last\_update(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$

$$event_{V_2}(x\ ack).$$

$$([x = last\_update3]\overline{ex}\langle project\_doc\_request\rangle.\overline{ack}\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{last\_update\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Checking\_last\_update(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

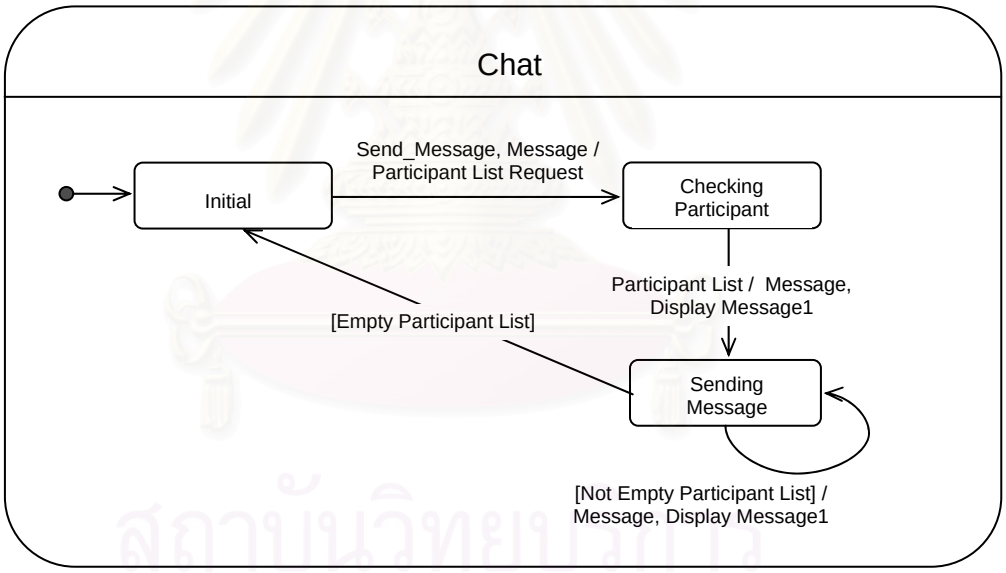
$$Distributing\_Diagram(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$

$$event_{V_2}(x\ ack).$$

$$([x = document1]\overline{in}_{DesPar}\langle document2\rangle.\overline{enq}_{Par}\langle document2\rangle.\overline{ack}\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{document1\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Distributing\_Diagram(event_{V_2}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$



รูปที่ ข-24 แผนภาพสถานะของ Designer Control: Chat superstate

$$Chat(event_{V_3}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$

$$event_{V_3}(x\ ack).$$

$$(\sum_{e\in\{send\_message, message\}}[x = e]\overline{ex}\langle participant\_list\_request\rangle.\overline{ack}\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{send\_message, message\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Chat(event_{V_3}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

$$Checking\_Participant(event_{V_3}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$

$$event_{V_3}(x\ ack).$$

$$([x = participant\_list]\overline{in}_{DesPar}\langle message\rangle.\overline{enq}_{Par}\langle message\rangle.$$

$$\overline{ex}\langle display\_message1\rangle.\overline{ack}\langle pos\rangle +$$

$$\sum_{e\in\{\bar{e}_{Des}\}\setminus\{participant\_list\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Checking\_Participant(event_{V_3}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

$$Sending\_Message(event_{V_3}, g_1, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$

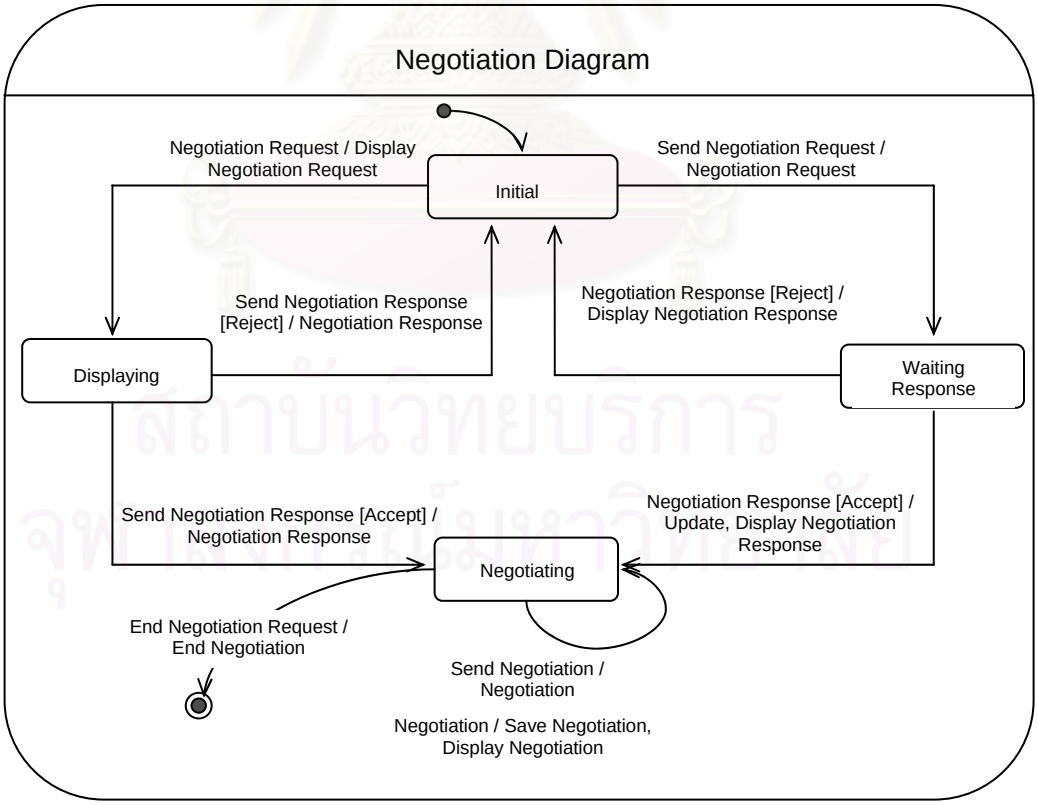
$$event_{V_3}(x\ ack).\overline{g}_1\langle z\rangle.z(y).$$

$$([y = not\_empty\_participant\_list]\overline{in}_{DesPar}\langle message\rangle.\overline{enq}_{Par}\langle message\rangle.$$

$$\overline{ex}\langle display\_message\rangle.\overline{ack}\langle pos\rangle +$$

$$[y = empty\_participant\_list]\overline{ack}\langle pos\rangle$$

$$Sending\_Message(event_{V_3}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$



รูปที่ ข-25 แผนภาพสถานะของ Designer Control: Negotiation superstate

$$Negotiation(event_{V_4}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$

$$event_{V_4}(x\ ack).$$

$$([x = send\_negotiation\_request]\overline{in}_{DesPar}\langle negotiation\_request\rangle.$$

$$\overline{enq}_{Par}\langle negotiation\_request\rangle.\overline{ack}\langle pos\rangle +$$

$$[x = negotiation\_request]\overline{ex}\langle display\_negotiation\_request\rangle.$$

$$\overline{ack}\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{send\_negotiation\_request, negotiation\_request\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Negotiation(event_{V_4}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

$$Waiting\_response(event_{V_4}, \vec{e}_{Des}, pos, neg, g_1, \overrightarrow{enq}, ex, \vec{in}) =$$

$$event_{V_4}(x\ ack).$$

$$([x = negotiation\_response]\overline{g}_1\langle z\rangle.z(y).$$

$$([y = accept]\overline{ex}\langle update\rangle.$$

$$\overline{ex}\langle negotiation\_response\rangle.\overline{ack}\langle pos\rangle +$$

$$[y = reject].\ \overline{ex}\langle negotiation\_response\rangle.\overline{ack}\langle pos\rangle) +$$

$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{negotiation\_response\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Waiting\_response(event_{V_4}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

$$Displaying(event_{V_4}, \vec{e}_{Des}, pos, neg, g_2, \overrightarrow{enq}, ex, \vec{in}) =$$

$$event_{V_4}(x\ ack).$$

$$([x = send\_negotiation\_response]\ \overline{g}_2\langle z\rangle.z(y).$$

$$([y = accept]\overline{in}_{DesPar}\langle negotiation\_response\rangle.$$

$$\overline{enq}_{Par}\langle negotiation\_response\rangle.\overline{ack}\langle pos\rangle +$$

$$[y = reject]\ \overline{in}_{DesPar}\langle negotiation\_response\rangle.$$

$$\overline{enq}_{Par}\langle negotiation\_response\rangle.\overline{ack}\langle pos\rangle) +$$

$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{send\_negotiation\_response\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Displaying(event_{V_4}, \vec{e}_{Des}, pos, neg, g_2, \overrightarrow{enq}, ex, \vec{in}))$$

$$Negotiating(event_{V_4}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$

$$event_{V_4}(x\ ack).$$

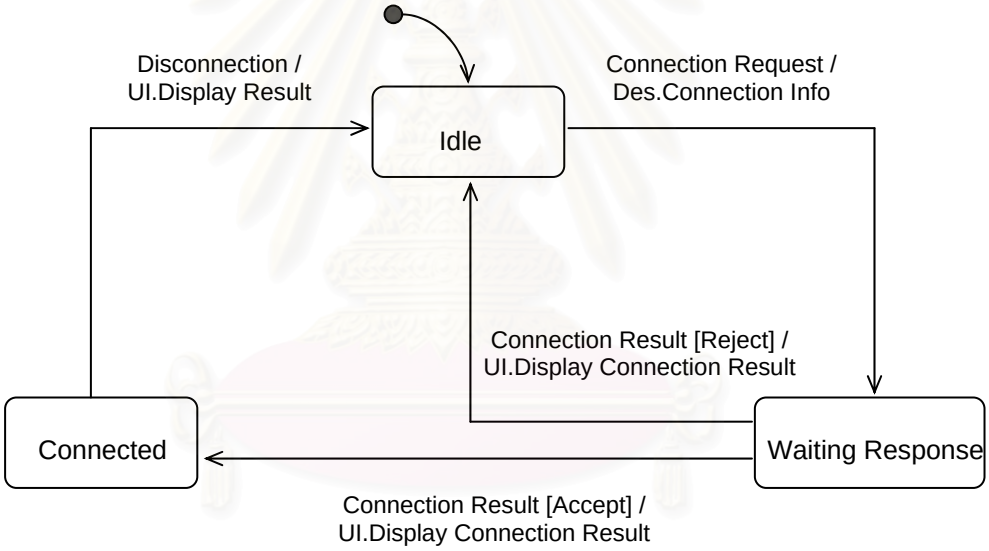
$$([x = send\_negotiation]\overline{in}_{DesPar}\langle negotiation\rangle.$$

$$\overline{enq}_{Par}\langle negotiation\rangle.\overline{ack}\langle pos\rangle +$$

$$[x = negotiation]\overline{ex}\langle save\_negotiation\rangle.$$
$$\overline{ex}\langle display\_negotiation\rangle.\overline{ack}\langle pos\rangle) +$$
$$[x = end\_negotiation\_request]\overline{in}_{DesPar}\langle end\_negotiation\rangle.$$
$$\overline{enq}_{Par}\langle end\_negotiation\rangle.\overline{ack}\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{Des}\}\setminus\{send\_negotiation, negotiation, end\_negotiation\_request\}}[x = e]\overline{ack}\langle neg\rangle.$$
$$Negotiating(event_{V_4}, \vec{e}_{Des}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

แผนภาพสถานะของ Participant

CSCW Participant Join Project



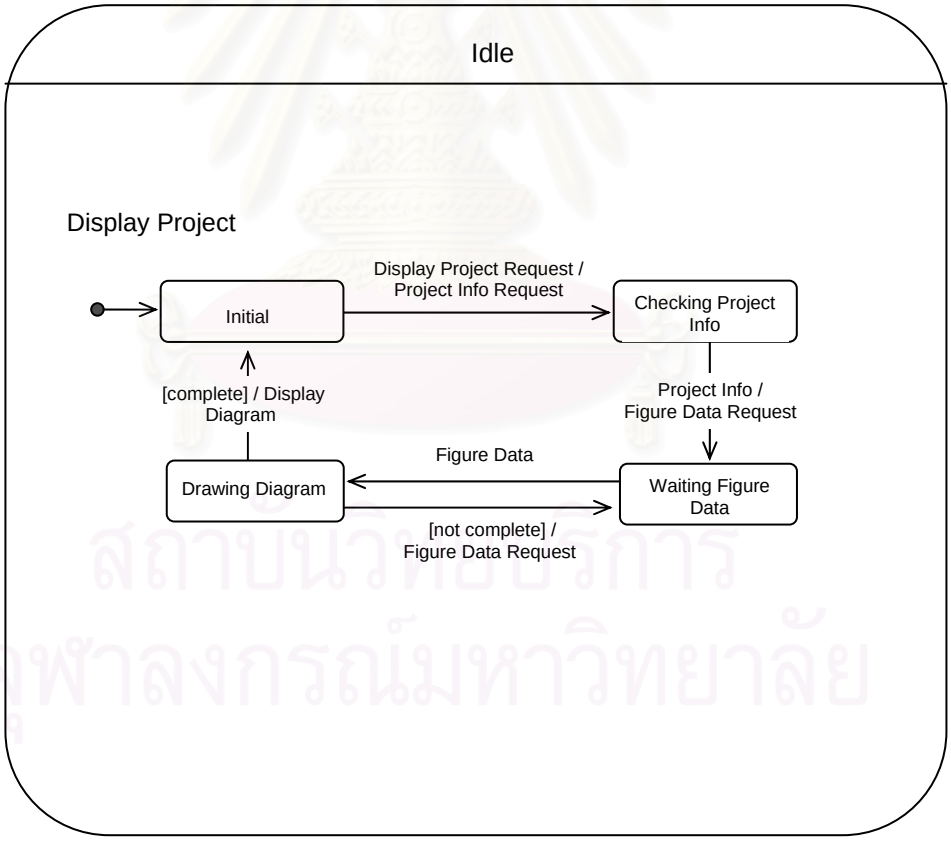
รูปที่ ข-26 Top-level statechart for Participant control

ให้ $\vec{e}_{Par} =$ $connection\_request, connection\_Info,$
$connection\_result, display\_connection\_result$
$disconnection, display\_result$

$$Idle(step, event_S, \vec{e}_{Par}, \overrightarrow{enq}, ex, \vec{in}) = event_s(x).$$
$$([x = connection\_request]\overline{enq}_{Des}\langle connection\_Info\rangle.$$
$$\overline{in}_{ParDes}\langle connection\_Info\rangle.\overline{step}.Waitng\_response(step, event_s, \vec{e}_{Par}, \overrightarrow{enq}) +$$
$$\sum_{e\in\{\vec{e}_{Par}\}\setminus\{connection\_request\}}[x = e]Error)$$

$$Waiting\_response(step, g, event_S, \vec{e}_{Par}, \overrightarrow{enq}, ex, \vec{in}) = event_s(x).$$
$$([x = connection\_result]\overline{g}\langle z\rangle.z(y).$$
$$([y = accept]\overline{ex}\langle display\_connection\_result\rangle.$$
$$\overline{step}.Connected(step, event_s, \vec{e}_{Par}, \overrightarrow{enq}, ex, \vec{in}) +$$
$$[y = reject]\overline{ex}\langle display\_connection\_result\rangle.$$
$$\overline{step}.Idle(step, event_s, \vec{e}_{Par}, \overrightarrow{enq}, ex, \vec{in})) +$$
$$\sum_{e\in\{\vec{e}_{Par}\}\setminus\{connection\_result\}}[x = e]Error)$$

$$Connected(step, event_S, \vec{e}_{Par}, \overrightarrow{enq}, ex, \vec{in}) = event_s(x).$$
$$([x = disconnection]\overline{ex}\langle display\_result\rangle.\overline{step}.Idle(step, event_s, \vec{e}_{Par}, \overrightarrow{enq}, ex, \vec{in}) +$$
$$\sum_{e\in\{\vec{e}_{Par}\}\setminus\{disconnection\}}[x = e]Error)$$



รูปที่ ข-27 แผนภาพสถานะของ Participant Control: Idle superstate

จากรูปที่ ข-27 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

จากกฎข้อที่ 4 จะได้

$$Idle(step, event_s, \vec{e}_{Par}, event_{V_1}, pos, neg) \mid Display\_project(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in})$$

และจากกฎข้อที่ 5 สามารถบรรยายสถานะประกอบ *Idle* และสถานะย่อย *Display _ project* ได้ดังนี้

$$\begin{aligned}
&Idle(step, event_S, \vec{e}_{Par}, event_{V_1}, pos, neg) = \\
&\quad event_s(x_1).(\nu\ ack) \\
&\quad (\overline{event}_{V1}\langle x_1\ ack\rangle.ack(y_1). \\
&\quad ([y_1 = pos](\\
&\qquad [x_1 = display_project_request]\overline{step}. \\
&\qquad (Idle(step, event_S, \vec{e}_{Par}, event_{V_1}, pos, neg) \mid \\
&\qquad Checking_project_Info(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) + \\
&\qquad [x_1 = project_Info]\overline{step}. \\
&\qquad (Idle(step, event_S, \vec{e}_{Par}, event_{V_1}, pos, neg) \mid \\
&\qquad Waiting_figure_data(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) + \\
&\qquad [x_1 = figure_data]\overline{step}. \\
&\qquad (Idle(step, event_S, \vec{e}_{Par}, event_{V_1}, pos, neg) \mid \\
&\qquad Drawing_diagram(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) + \\
&\qquad [x_1 = connection_request]\overline{step}. \\
&\qquad Waiting_response(step, g_1, event_S, \vec{e}_{Par}, \overrightarrow{enq}, ex, \vec{in})) \\
&\quad [y_1 = neg]Error)
\end{aligned}$$

$$\begin{aligned}
&Display_project(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) = \\
&\quad event_{V_1}(x\ ack). \\
&\quad ([x = display_project_request] \\
&\qquad \overline{ex}\langle project_Info_request\rangle.\overline{ack}\langle pos\rangle + \\
&\quad \sum_{e \in \{\vec{e}_{Par}\} \setminus \{display_project_request\}} [x = e]\overline{ack}\langle neg\rangle. \\
&\qquad Display_project(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))
\end{aligned}$$

$$Checking\_project\_Info(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$

$$event_{V_1}(x\ ack).$$

$$([x = project\_Info]\overline{ex}\langle figure\_data\_request\rangle.\ \overline{ack}\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{Par}\}\setminus\{project\_Info\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Checking\_project\_Info(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

$$Waiting\_figure\_data(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$

$$event_{V_1}(x\ ack).$$

$$([x = figure\_data]\ \overline{ack}\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{Par}\}\setminus\{figure\_data\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Waiting\_figure\_data(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

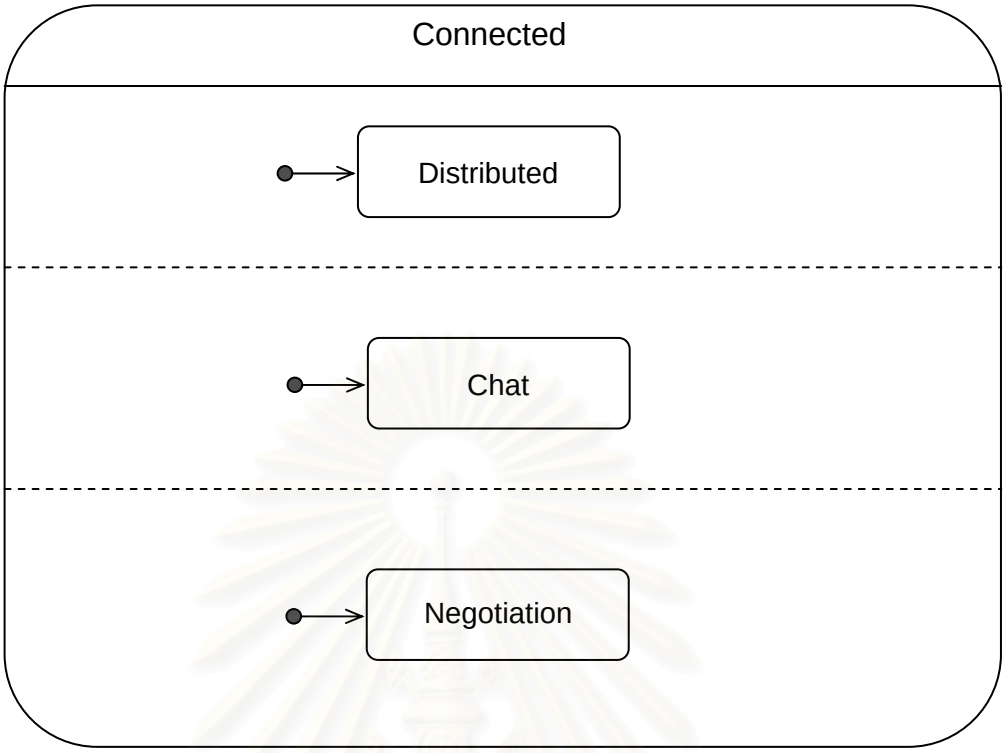
$$Drawing\_diagram(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, g_1, \overrightarrow{enq}, ex, \vec{in}) =$$

$$\overline{g}_1\langle z\rangle.z(y).$$

$$([y = completed]\overline{ex}\langle display\_diagram\rangle\ \overline{ack}\langle pos\rangle +$$

$$[y = not\_completed]\overline{ex}\langle figure\_data\_request\rangle\ \overline{ack}\langle pos\rangle.$$

$$Drawing\_diagram(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, g_1, \overrightarrow{enq}, ex, \vec{in})$$



รูปที่ ข-28 แผนภาพสถานะของ Participant Control: Connected superstate

จากรูปที่ ข-28 สามารถแปลงเป็นไพแคลคูลัสได้ ดังนี้

จากกฎข้อที่ 4 จะได้

$$Connected(step, event_s, \vec{e}_{Par}, ...) \mid Distributed(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) \mid$$
$$Chat(event_{v_2}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) \mid Negotiation(event_{v_3}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in})$$

และจากกฎข้อที่ 6 สามารถบรรยายสถานะประกอบ *Connected* และสถานะย่อย *Distributed* *Chat* และ *Negotiation* ได้ดังนี้

$$Connected(step, event_S, \vec{e}_{Par}, event_{V_1}, event_{V_2}, event_{V_3}, pos, neg) =$$
$$event_s(x_1).(\nu\ ack_1\ ack_2\ ack_3)\ (\overline{event}_{V1}\langle x_1\ ack_1\rangle.ack_1(y_1).$$
$$([y_1 = pos]($$
$$[x_1 = last\_update\_request]\overline{step}.$$
$$(Connected(step, event_S, \vec{e}_{Par}, event_{V_1}, event_{V_2}, event_{V_3}, pos, neg) \mid$$
$$Checking\_Last\_Update(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) +$$

$$[x_1 = document2]\ \overline{step}.$$
$$(Connected(step, event_S, \vec{e}_{Par}, event_{V_1}, event_{V_2}, event_{V_3}, pos, neg) \mid$$
$$Updating(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) +$$

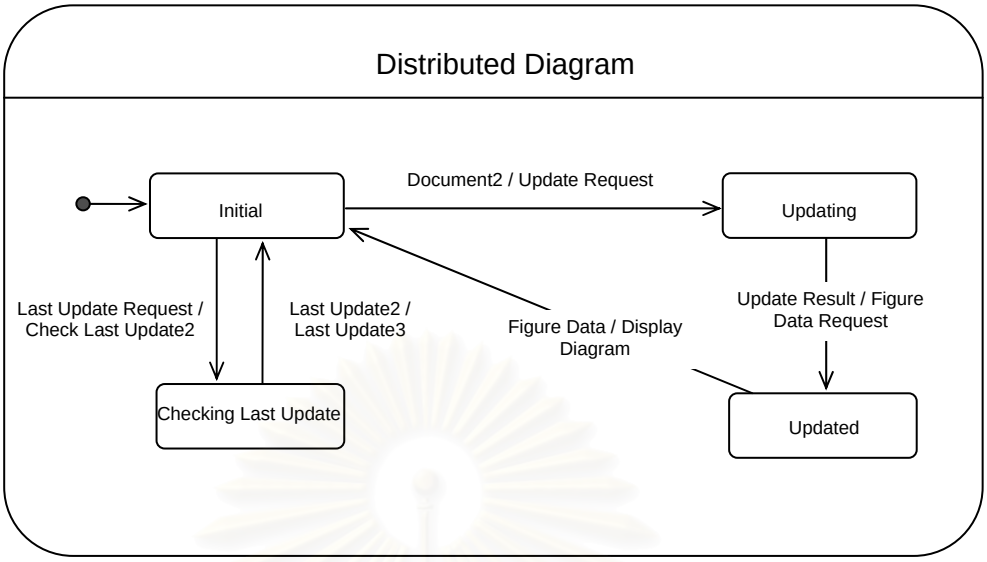
$$[x_1 = update\_result]\ \overline{step}.$$
$$(Connected(step, event_S, \vec{e}_{Par}, event_{V_1}, event_{V_2}, event_{V_3}, pos, neg) \mid$$
$$Updated(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) +$$

$$[x_1 = figure\_data]\ \overline{step}.$$
$$(Connected(step, event_S, \vec{e}_{Par}, event_{V_1}, event_{V_2}, event_{V_3}, pos, neg) \mid$$
$$Distributed(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) +$$

$$[x_1 = last\_update2]\ \overline{step}.$$
$$(Connected(step, event_S, \vec{e}_{Par}, event_{V_1}, event_{V_2}, event_{V_3}, pos, neg) \mid$$
$$Distributed(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in})) +$$

$$[y_1 = neg] Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, pos, neg)).$$

$$\overline{event}_{V2}\langle x_1\ ack_2\rangle.ack_2(y_2)...$$
$$\overline{event}_{V3}\langle x_1\ ack_3\rangle.ack_3(y_3)...$$
$$\overline{step}.([y_1 = pos]([y_2 = pos]([y_3 = pos]([x_1 = disconnection]$$
$$Idle(step, event_S, \vec{e}_{Des}, event_{V_1}, pos, neg)$$



รูปที่ ข-29 แผนภาพสถานะของ Participant Control: Distributed Diagram superstate

$$Distributed(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$
$$event_{V_1}(x\ ack).$$
$$([x = last\_update\_request]\overline{ex}\langle check\_last\_update2\rangle.\overline{ack}\langle pos\rangle +$$
$$[x = document2]\overline{ex}\langle update\_request\rangle.\overline{ack}\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{Par}\}\setminus\{last\_update\_request, document2\}}[x = e]\overline{ack}\langle neg\rangle.$$
$$Distributed(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

$$Checking\_Last\_Update(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$
$$event_{V_1}(x\ ack).$$
$$([x = last\_update2]\overline{in}_{ParDes}\langle last\_update3\rangle.\overline{enq}_{Des}\langle last\_update3\rangle.\overline{ack}\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{Par}\}\setminus\{last\_update2\}}[x = e]\overline{ack}\langle neg\rangle.$$
$$Checking\_Last\_Update(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

$$Updating(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$
$$event_{V_1}(x\ ack).$$
$$([x = update\_result]\overline{ex}\langle figure\_data\_request\rangle.\overline{ack}\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{Par}\}\setminus\{update\_result\}}[x = e]\overline{ack}\langle neg\rangle.$$
$$Updating(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

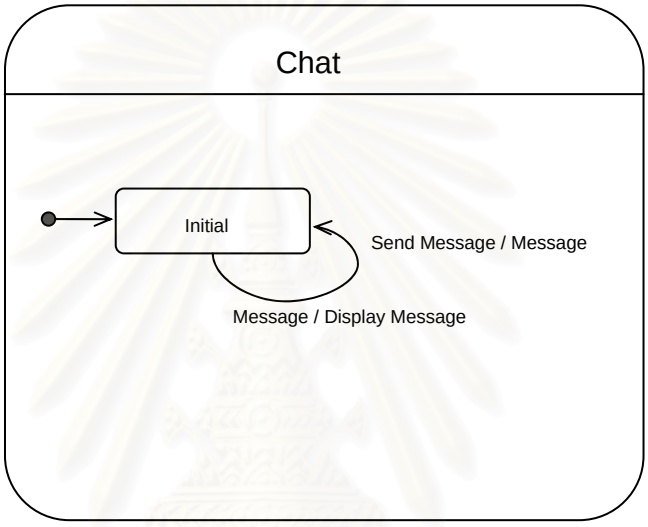
$$Updated(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$

$$event_{V_1}(x\ ack).$$

$$([x = figure\_data]\overline{ex}\langle display\_diagram\rangle.\overline{ack}\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{Par}\}\setminus\{figure\_data\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Updated(event_{V_1}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$



รูปที่ ข-30 แผนภาพสถานะของ Participant Control: Chat superstate

$$Chat(event_{V_2}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$

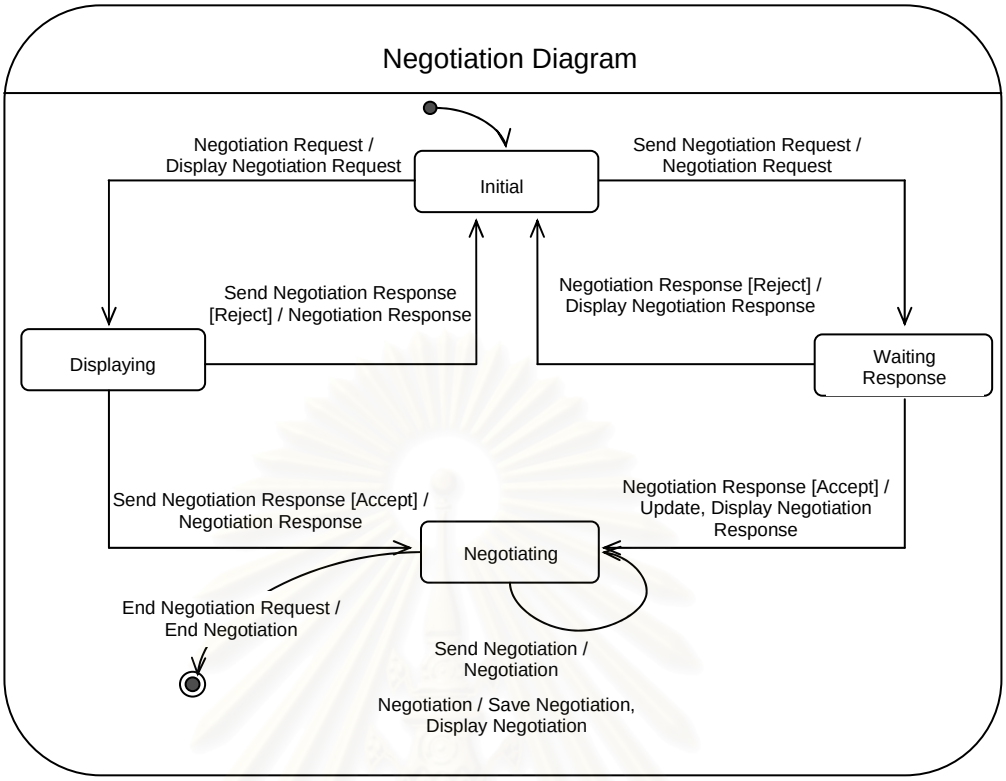
$$event_{V_2}(x\ ack).$$

$$([x = send\_message]\overline{in}_{ParDes}\langle message\rangle.\overline{enq}_{Des}\langle message\rangle.\overline{ack}\langle pos\rangle +$$

$$[x = message]\overline{ex}\langle display\_message\rangle.\overline{ack}\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{Par}\}\setminus\{send\_message, message\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Chat(event_{V_2}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$



รูปที่ ข-31 แผนภาพสถานะของ Participant Control: Negotiation superstate

$$Negotiation(event_{V_3}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}) =$$

$$event_{V_3}(x\ ack).$$

$$([x = send\_negotiation\_request]\overline{in}_{ParDes}\langle negotiation\_request\rangle.$$

$$\overline{enq}_{Des}\langle negotiation\_request\rangle.\overline{ack}\langle pos\rangle +$$

$$[x = negotiation\_request]\overline{ex}\langle display\_negotiation\_request\rangle.$$

$$\overline{ack}\langle pos\rangle +$$

$$\sum_{e\in\{\vec{e}_{Par}\}\setminus\{send\_negotiation\_request, negotiation\_request\}}[x = e]\overline{ack}\langle neg\rangle.$$

$$Negotiation(event_{V_3}, \vec{e}_{Par}, pos, neg, \overrightarrow{enq}, ex, \vec{in}))$$

$$Waiting\_response(event_{V_3}, \vec{e}_{Par}, pos, neg, g_1, \overrightarrow{enq}, ex, \vec{in}) =$$

$$event_{V_3}(x\ ack).$$

$$([x = negotiation\_response]\overline{g}_1\langle z\rangle.z(y).$$

$$([y = accept]\overline{ex}\langle update\rangle.$$

$$\overline{ex}\langle negotiation\_response\rangle.\overline{ack}\langle pos\rangle +$$

$$[y = reject].\ \overline{ex}\langle negotiation\_response\rangle.\overline{ack}\langle pos\rangle) +$$

$$\sum_{e\in\{\vec{e}_{Par}\}\setminus\{negotiation\_response\}}[x=e]\overline{ack}\langle neg\rangle.$$
$$Waiting\_response(event_{V_3},\vec{e}_{Par},pos,neg,g_1,\overrightarrow{enq},ex,\vec{in}))$$

$$Displaying(event_{V_3},\vec{e}_{Par},pos,neg,g_2,\overrightarrow{enq},ex,\vec{in}) =$$
$$event_{V_3}(x\ ack).$$
$$([x=send\_negotiation\_response]\ \overline{g}_2\langle z\rangle.z(y).$$
$$([y=accept]\overline{in}_{ParDes}\langle negotiation\_response\rangle.$$
$$\overline{enq}_{Des}\langle negotiation\_response\rangle.\overline{ack}\langle pos\rangle +$$
$$[y=reject]\ \overline{in}_{ParDes}\langle negotiation\_response\rangle.$$
$$\overline{enq}_{Des}\langle negotiation\_response\rangle.\overline{ack}\langle pos\rangle) +$$
$$\sum_{e\in\{\vec{e}_{Par}\}\setminus\{send\_negotiation\_response\}}[x=e]\overline{ack}\langle neg\rangle.$$
$$Displaying(event_{V_3},\vec{e}_{Par},pos,neg,g_2,\overrightarrow{enq},ex,\vec{in}))$$

$$Negotiating(event_{V_3},\vec{e}_{Par},pos,neg,\overrightarrow{enq},ex,\vec{in}) =$$
$$event_{V_3}(x\ ack).$$
$$([x=send\_negotiation]\overline{in}_{ParDes}\langle negotiation\rangle.$$
$$\overline{enq}_{Des}\langle negotiation\rangle.\overline{ack}\langle pos\rangle +$$
$$[x=negotiation]\overline{ex}\langle save\_negotiation\rangle.$$
$$\overline{ex}\langle display\_negotiation\rangle.\overline{ack}\langle pos\rangle) +$$
$$[x=end\_negotiation\_request]\overline{in}_{ParDes}\langle end\_negotiation\rangle.$$
$$\overline{enq}_{Des}\langle end\_negotiation\rangle.\overline{ack}\langle pos\rangle +$$
$$\sum_{e\in\{\vec{e}_{Par}\}\setminus\{send\_negotiation,negotiation,end\_negotiation\_request\}}[x=e]\overline{ack}\langle neg\rangle.$$
$$Negotiating(event_{V_3},\vec{e}_{Par},pos,neg,\overrightarrow{enq},ex,\vec{in}))$$

การตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะโดยใช้ไพแคลคูลัส

ในขั้นตอนนี้จะตรวจสอบความต้องกันระหว่างแผนภาพความร่วมมือ และแผนภาพสถานะโดยใช้ไพแคลคูลัสที่ได้จากการแปลงในหัวข้อที่แล้ว โดยจะยกตัวอย่างการตรวจสอบเป็น 2 กรณี คือ กรณีที่ในระบบไม่มีการส่งข้อความแบบพร้อมกัน และกรณีที่ในระบบมีการส่งข้อความแบบพร้อมกัน โดยวัตถุ Participant Control และ Designer Control จะทำงานแบบขนานกัน (parallel)

1. กรณีที่ในระบบไม่มีการส่งข้อความแบบพร้อมกัน

พิจารณาลำดับของข้อความที่ได้บรรยายไว้ในรูปที่ ข-3 จะได้ ดังนี้ คือ

$connection\_request, connection\_Info, display\_connection\_Info,$
$send\_response, connection\_result, display\_connection\_result$

$$Queue_6^F(deq, empty, connection\_request, connection\_Info, display\_connection\_Info, send\_response, connection\_result, display\_connection\_result) \quad | \quad Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) \qquad (ข\text{-}1)$$

$$= \quad deq(r).\bar{r}\langle connection\_request\rangle.Queue_5^F(deq, empty, ...) + empty(t\ f).\bar{f}.Queue_6^F(deq, empty, ..) \quad | \quad \overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) + [event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) + t.Success$$

$$\xrightarrow{\tau} \quad \bar{f}.Queue_6^F(deq, empty, connection\_request, connection\_Info, display\_connection\_Info, send\_response, connection\_result, display\_connection\_result) \quad | \quad f.\overline{deq}\langle y\rangle.y(event).(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) + [event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) + t.Success$$

$$\xrightarrow{\tau} \quad Queue_6^F(deq, empty, connection\_request, connection\_Info, display\_connection\_Info, send\_response, connection\_result, display\_connection\_result) \quad | \quad \overline{deq}\langle y\rangle.y(event).(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) + [event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$$

$$= \quad deq(r).\bar{r}\langle connection\_request\rangle.Queue_5^F(deq, empty, ...) + empty(t\ f).\bar{f}.Queue_6^F(deq, empty, ..) \quad | \quad \overline{deq}\langle y\rangle.y(event).(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$$

$$[event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$$

$\xrightarrow{\tau}$ $\bar{y}\langle connection\_request\rangle.Queue_5^F(deq, empty, ...)$ | $y(event).$
$(\sum_{e\in\{\bar{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$
$\sum_{e\in\{\bar{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$
$[event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$\xrightarrow{\tau}$ $Queue_5^F(deq, empty, ...)$ | $(\sum_{e\in\{\bar{e}_{EXT}\}}[connection\_request = e]$
$.Scenario'(connection\_request, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$
$\sum_{e\in\{\bar{e}_{INT}\}}[connection\_request = e]$
$Scenario''(connection\_request, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$
$[connection\_request = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$\xrightarrow{\tau}$ $Queue_5^F(deq, empty, display\_connection\_Info,$ | $Scenario'(connection\_request, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS})$
$send\_response, connection\_result,$
$display\_connection\_result)$

$=$ $deq(r).\bar{r}\langle m_1\rangle.Queue_4^F(deq, empty, ...) +$ | $\overline{enq}_{Par}\langle connection\_request\rangle$
$empty(tf).\bar{f}.Queue_5^F(deq, ...)$ | $.Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$

ต่อไปพิจารณาการทำงานของ *Scenario* ที่นำข้อความมาใส่ไว้ในแถวคอยของวัตถุ *Participant Control* ดังนี้

$$Queue_0^{Par}(enq_{Par}, deq_{Par}, empty) \mid Scenario'(connection\_request, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS})$$

$$= enq_{Par}(x_1).Queue_1^{Par}(enq_{Par}, deq_{Par}, empty, x_1) + empty(tf).\bar{t}.Queue_0^{Par}(enq_{Par}, deq_{Par}, empty) \mid \overline{enq}_{Par}\langle connection\_request\rangle. Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$$

$$\xrightarrow{\tau} Queue_1^{Par}(enq_{Par}, deq_{Par}, empty, connection\_request) \mid Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$$

(ข-2) (ข-3)

จาก (ข-3) พบว่า *Scenario* พร้อมที่จะนำข้อความถัดไปในแถวคอยของแผนภาพความร่วมมือมาพิจารณาต่อเหมือน (ข-1) และจาก (ข-2) จะพบว่าขณะนี้มีข้อความ *connection_request* อยู่ในแถวคอยของวัตถุ *Participant Control* โดยจะอธิบายการทำงานต่อไป ดังนี้

$$Queue_1^{Par}(enq_{Par}, deq_{Par}, empty, connection\_request) \mid Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete)$$

$$= enq_{Par}(x_2).Queue_2^{Par}(enq_{Par}, deq_{Par}, empty, x_1, x_2) + deq_{Par}(r).\bar{r}\langle connection\_request\rangle. Queue_0^{Par}(enq_{Par}, deq_{Par}, empty) + empty_{Par}(tf).\bar{f}. Queue_1^{Par}(enq_{Par}, deq_{Par}, empty, connection\_request) \mid \overline{empty}\langle tf\rangle. (t.Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete) + f.\overline{deq}_{Par}\langle r\rangle.r(event).\overline{execute}\langle event\rangle.complete. Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete))$$

$$\xrightarrow{\tau} \ \overline{f}.Queue_1^{Par}(enq_{Par}, deq_{Par}, empty, connection\_request) \quad | \quad (t.Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete) +$$
$$f.\overline{deq}_{Par}\langle r\rangle.r(event).\overline{execute}\langle event\rangle.complete.$$
$$Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete))$$

$$\xrightarrow{\tau} \ Queue_1^{Par}(enq_{Par}, deq_{Par}, empty, connection\_request) \quad | \quad \overline{deq}_{Par}\langle r\rangle.r(event).\overline{execute}\langle event\rangle.complete.$$
$$Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete)$$

$$= \ enq_{Par}(x_2).Queue_2^{Par}(enq_{Par}, deq_{Par}, empty, x_1, x_2) + \quad | \quad \overline{deq}_{Par}\langle r\rangle.r(event).\overline{execute}\langle event\rangle.complete.$$
$$deq_{Par}(r).\bar{r}\langle connection\_request\rangle. \quad Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete)$$
$$Queue_0^{Par}(enq_{Par}, deq_{Par}, empty) +$$
$$empty_{Par}(t f).\bar{f}.$$
$$Queue_1^{Par}(enq_{Par}, deq_{Par}, empty, connection\_request)$$

$$\xrightarrow{\tau} \ \bar{r}\langle connection\_request\rangle.Queue_0^{Par}(enq_{Par}, deq_{Par}, empty) \quad | \quad r(event).\overline{execute}\langle event\rangle.complete.$$
$$Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete)$$

$$\xrightarrow{\tau} \ Queue_0^{Par}(enq_{Par}, deq_{Par}, empty) \quad | \quad \overline{execute}\langle connection\_request\rangle.complete.$$
$$Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete)$$

จากนิพจน์ข้างต้นเป็นการทำงานของวัตถุ *Participant Control* ที่นำข้อความออกมาจากแถวคอยด้วยโปรแกรมเลือกจ่ายเหตุการณ์

ต่อไปเป็นการทำงานระหว่างตัวประมวลผลเหตุการณ์และโปรแกรมเลือกจ่ายเหตุการณ์

$$
\begin{aligned}
& Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete) \mid S_0(execute, step, complete, event_s) \\
= \; & \overline{execute}\langle connection\_request\rangle.complete. \mid execute(x).\overline{event}_s\langle x\rangle.step. \\
& Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete) \quad \overline{complete}.S_0(execute, step, complete, event_s) \\
\xrightarrow{\tau} \; & complete. \mid \overline{event}_s\langle connection\_request\rangle.step. \\
& Dispatch^{Par}(t, f, r, deq_{Par}, empty_{Par}, execute, complete) \quad \overline{complete}.S_0(execute, step, complete, event_s)
\end{aligned}
$$

เมื่อตัวประมวลผลเหตุการณ์ได้รับข้อความจากโปรแกรมเลือกจ่ายเหตุการณ์จะนำข้อความที่ได้ไปให้วัตถุ *Participant Control* ดังนี้

$$
\begin{aligned}
& S_0(execute, step, complete, event_s) \mid Idle(step, event_S, \vec{e}_{Des}, \overrightarrow{enq}) \\
= \; & \overline{event}_s\langle connection\_request\rangle.step. \mid event_s(x). \\
& \overline{complete}.S_0(execute, step, complete, event_s) \quad ([x = connection\_request]\overline{enq}_{Des}\langle connection\_Info\rangle. \\
& \quad \overline{in}_{ParDes}\langle connection\_Info\rangle.\overline{step}. \\
& \quad Waitng\_response(step, event_s, \vec{e}_{Des}, \overrightarrow{enq}) + \\
& \quad \Sigma_{e\in\{\vec{e}_{Des}\}\setminus\{connection\_request\}}[x = e]Error) \\
\xrightarrow{\tau} \; & step. \mid \overline{enq}_{Des}\langle connection\_Info\rangle.\ \overline{in}_{ParDes}\langle connection\_Info\rangle.\overline{step}. \\
& \overline{complete}.S_0(execute, step, complete, event_s) \quad Waitng\_response(step, event_s, \vec{e}_{Des}, \overrightarrow{enq}) +
\end{aligned}
$$

จะพบว่าเมื่อวัตถุ *Participant Control* ได้รับข้อความ $connection\_request$ จะส่งข้อความ $connection\_Info$ ไปให้วัตถุ *Designer Control* โดยนำข้อความดังกล่าวไปใส่ไว้ในแถวคอยของวัตถุดังกล่าว ดังนี้

$$Queue_0^{Par}(enq, deq, empty) \quad | \quad Idle(step, event_S, \vec{e}_{Des}, \overrightarrow{enq})$$

$$= enq_{Des}(x_1).Queue_1^{Des}(enq_{Des}, deq_{Des}, empty, x_1) + empty(tf).\bar{t}.Queue_0^{Des}(enq_{Des}, deq_{Des}, empty) \quad | \quad \overline{enq}_{Des}\langle connection\_Info\rangle.\ \overline{in}_{ParDes}\langle connection\_Info\rangle.\overline{step}.\ Waitng\_response(step, event_s, \vec{e}_{Des}, \overrightarrow{enq})$$

$$\xrightarrow{\tau} Queue_1^{Des}(enq, deq, empty, connection\_Info) \quad | \quad \overline{in}_{ParDes}\langle connection\_Info\rangle.\overline{step}.\ Waitng\_response(step, event_s, \vec{e}_{Des}, \overrightarrow{enq})$$

(ข-4) (ข-5)

นอกจากวัตถุ *Participant Control* จะส่งข้อความ $connection\_Info$ ไปให้วัตถุ *Designer Control* แล้ว ยังส่งข้อความดังกล่าวไปตรวจสอบความต้องกันกับแผนภาพความร่วมมือผ่านทาง *Scenario″* อีกด้วย

$$Scenario''(connection\_Info, \vec{in}, \vec{e}_{INT}, \vec{e}_{SYS}) = \quad | \quad Idle(step, event_S, \vec{e}_{Des}, \overrightarrow{enq})$$

$$= in_{ParDes}(x).([x = connection\_Info]\ Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}) + \sum_{e \in \{\vec{e}_{SYS}\} \setminus \{connection\_Info\}} [x = e]\, Error) \quad | \quad \overline{in}_{ParDes}\langle connection\_Info\rangle.\overline{step}.\ Waitng\_response(step, event_s, \vec{e}_{Des}, \overrightarrow{enq})$$

$\xrightarrow{\tau}$ $Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$ | $\overline{step}.Waitng\_response(step, event_s, \vec{e}_{Des}, \overrightarrow{enq})$

ทำคล้าย (ข-1) ไปเรื่อย ๆ จนไม่มีข้อความในแถวคอยของแผนภาพความร่วมมือแล้ว หากไม่พบข้อผิดพลาดใด ๆ จากวัตถุอื่นที่บรรยายไว้ด้วยแผนภาพสถานะ แสดงว่าแผนภาพความร่วมมือและแผนภาพสถานะข้างต้นมีความต้องกัน

2. กรณีที่ในระบบมีการส่งข้อความแบบพร้อมกัน

พิจารณาลำดับของข้อความที่ได้บรรยายไว้ในรูปที่ ข-6 จะได้ ดังนี้ คือ

$send\_message, participant\_list\_request, participant\_list,$
$\theta, message, display\_message1, \gamma, display\_message2$

$Queue_8^F(deq, empty, send\_message,$ | $Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$
$participant\_list\_request, participant\_list, \theta, message,$
$display\_message1, \gamma, display\_message2)$ (ข-6)

$= deq(r).\bar{r}\langle send\_message\rangle.Queue_7^F(deq, empty, ...) +$ | $\overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$
$empty(t\ f).\bar{f}.Queue_8^F(deq, empty, ..)$ $(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$
$\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$
$[event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) +$
$t.Success$

$$\xrightarrow{\tau} \ \overline{f}.Queue_8^F\,(deq, empty, send\_message,\ |\ f.\overline{deq}\langle y\rangle.y(event).$$
$$participant\_list\_request, participant\_list, \theta, message,$$
$$(\textstyle\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'\,(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$$
$$display\_message1, \gamma, display\_message2)$$
$$\textstyle\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})) +$$
$$t.Success$$

$$\xrightarrow{\tau} \ Queue_8^F\,(deq, empty, send\_message,\ |\ \overline{deq}\langle y\rangle.y(event).$$
$$participant\_list\_request, participant\_list, \theta, message,$$
$$(\textstyle\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'\,(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$$
$$display\_message1, \gamma, display\_message2)$$
$$\textstyle\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$$

$$= \ deq(r).\bar{r}\langle send\_message\rangle.Queue_7^F\,(deq, empty, ...) +\ |\ \overline{deq}\langle y\rangle.y(event).$$
$$empty(t\,f).\bar{f}.Queue_8^F\,(deq, empty, ..)$$
$$(\textstyle\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'\,(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$$
$$\textstyle\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$$

$$\xrightarrow{\tau} \ \bar{y}\langle send\_message\rangle.Queue_7^F\,(deq, empty, ...)\ |\ y(event).$$
$$(\textstyle\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'\,(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$$
$$\textstyle\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$$

$\xrightarrow{\tau}$ $Queue_7^F(deq, empty, ...)$ | $(\sum_{e\in\{\vec{e}_{EXT}\}}[send\_message = e]$
$.Scenario'(sent\_message, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$
$\sum_{e\in\{\vec{e}_{INT}\}}[send\_message = e]$
$Scenario''(send\_message, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$
$[send\_message = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$\xrightarrow{\tau}$ $Queue_7^F(deq, empty, participant\_list\_request,$ | $Scenario'(send\_message, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS})$
$participant\_list, \theta, message, display\_message1, \gamma,$
$display\_message2)$

$=$ $deq(r).\bar{r}\langle participant\_list\_request\rangle.$ | $\overline{enq}_{Des}\langle event\rangle.\ Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$
$Queue_6^F(deq, empty, ...) +$
$empty(tf).\bar{f}.Queue_7^F(deq, ...)$

ต่อไปพิจารณาการทำงานของ *Scenario* ที่นำข้อความมาใส่ไว้ในแถวคอยของวัตถุ *Designer Control* ดังนี้

$Queue_0^{Des}(enq_{Des}, deq_{Des}, empty)$ | $Scenario'(send\_message, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS})$

$=$ $enq_{Des}(x_1).Queue_1^{Des}(enq_{Des}, deq_{Des}, empty, x_1) +$ | $\overline{enq}_{Des}\langle send\_message\rangle.\ Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$
$empty(tf).\bar{t}.Queue_1^{Des}(enq_{Par}, deq_{Par}, empty)$

$\xrightarrow{\tau}$ $Queue_1^{Des}(enq_{Par}, deq_{Par}, empty, connection\_request)$ | $Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$

(ข-7) (ข-8)

จาก (ข-8) พบว่า *Scenario* พร้อมที่จะนำข้อความถัดไปในแถวคอยของแผนภาพความร่วมมือมาพิจารณาต่อเหมือน (ข-6) และจาก (ข-7) จะพบว่าขณะนี้มีข้อความ *send _ message* อยู่ในแถวคอยของวัตถุ *Designer Control*

ต่อไปเมื่อนำข้อความจากแถวคอยของแผนภาพความร่วมมือมาพิจารณาจนกระทั่งถึงข้อความที่เกิดขึ้นแบบพร้อมกัน จะมีวิธีการตรวจสอบ ดังนี้

$\xrightarrow{\tau}$ $Queue_5^F(deq, empty, \theta, message, display\_message1, \gamma, display\_message2)$ | $\overline{deq}\langle y\rangle.y(event).(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) + [event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$= deq(r).\bar{r}\langle\theta\rangle.Queue_4^F(deq, empty, ...) + empty(tf).\bar{f}.Queue_5^F(deq, empty, ..)$ | $\overline{deq}\langle y\rangle.y(event).(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) + \sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) + [event = \theta]Scenario'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$

$$\xrightarrow{\tau} \quad \bar{y}\langle\theta\rangle.Queue_4^F(deq, empty, ...) \quad | \quad y(event).$$
$$(\sum_{e\in\{\vec{e}_{EXT}\}}[event = e].Scenario'(event, \overrightarrow{enq}, \vec{e}_{EXT}, \vec{e}_{SYS}) +$$
$$\sum_{e\in\{\vec{e}_{INT}\}}[event = e]Scenario''(event, in, \vec{e}_{INT}, \vec{e}_{SYS}) +$$
$$[event = \theta]Scenario_0'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}))$$

$$\xrightarrow{\tau} \quad Queue_4^F(deq, empty, message, \quad | \quad Scenario_0'''(in, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$$
$$display\_message1, \gamma, display\_message2)$$

$$= \quad deq(r).\bar{r}\langle message\rangle.Queue_3^F(deq, empty, ...) + \quad | \quad \overline{empty}\langle t\ f\rangle.f.\overline{deq}\langle y\rangle.y(event).$$
$$empty(t\ f).\bar{f}.Queue_4^F(deq, empty, ..) \qquad Scenario_1'''(empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, event)$$

$$\xrightarrow{\tau} \quad \bar{f}.Queue_4^F(deq, empty, ..) \quad | \quad f.\overline{deq}\langle y\rangle.y(event).$$
$$Scenario_1'''(empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, event)$$

$$\xrightarrow{\tau} \quad Queue_4^F(deq, empty, message, \quad | \quad \overline{deq}\langle y\rangle.y(event).$$
$$display\_message1, \gamma, display\_message2) \qquad Scenario_1'''(empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, event)$$

$$= \quad deq(r).\bar{r}\langle message\rangle.Queue_3^F(deq, empty, ...) + \quad | \quad \overline{deq}\langle y\rangle.y(event).$$
$$empty(t\ f).\bar{f}.Queue_4^F(deq, empty, ..) \qquad Scenario_1'''(empty, deq, \overrightarrow{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, event)$$

$\xrightarrow{\tau}$ $\bar{y}\langle message\rangle.Queue_3^F\,(deq, empty, ...)$ | $y(event).Scenario_1'''\ (empty, deq, \vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, event)$

$\xrightarrow{\tau}$ $Queue_3^F\,(deq, empty, display\_message1, \gamma,$ | $Scenario_1'''\ (empty, deq, \vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, message)$
$display\_message2)$

จะพบว่าเมื่อนำข้อความจากแถวคอยของแผนภาพความร่วมมือมาใส่ไว้ในกระบวนการ $Scenario'''$ จนกระทั่งถึงข้อความ γ ขั้นตอนต่อไปจะมีวิธีการตรวจสอบ ดังนี้

$\xrightarrow{\tau}$ $\bar{y}\langle\gamma\rangle.Queue_1^F\,(deq, empty, display\_message2)$ | $Scenario_2^4(\vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, \overrightarrow{ack}, message,$
$([ack_1 = pos]([ack_2 = pos]$
$Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$

$\xrightarrow{\tau}$ $Queue_1^F\,(deq, empty, display\_message2)$ | $Scenario_2^4(\vec{in}, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS}, \overrightarrow{ack}, message,$
$([ack_1 = pos]([ack_2 = pos]$
$Scenario(empty, deq, \vec{e}_{EXT}, \vec{e}_{INT}, \vec{e}_{SYS})$

$=$ $deq(r).\bar{r}\langle display\_message2\rangle.Queue_0^F\,(deq, empty, ...) +$ | $in_{DesPar}(x).\,([x = message]\overline{ack}_1\langle pos\rangle +$
$empty(tf).\bar{f}.Queue_0^F\,(deq, empty)$ $\sum_{e\in\{\bar{e}_{SYS}\}\setminus\{message\}}[x = e]Error)$ |

$$ex(x).\ ([x = display\_message1]\overline{ack}_2\langle pos\rangle + \sum_{e\in\{\vec{e}_{SYS}\}\setminus\{display\_message1\}}[x = e]Error)$$

จากการทดสองพบว่าเกิดข้อความ *message* และ *display _ message*1 ขึ้น และไม่พบข้อผิดพลาดใด ๆ จากวัตถุอื่นที่บรรยายไว้ด้วยแผนภาพสถานะ แสดงว่าแผนภาพความร่วมมือและแผนภาพสถานะข้างต้นมีความต้องกัน

ประวัติผู้เขียนวิทยานิพนธ์

นายวราวุฒิ ผ้าเจริญ เกิดเมื่อวันที่ 16 ธันวาคม พ.ศ. 2523 ที่จังหวัดตราด สำเร็จการศึกษาหลักสูตรวิทยาศาสตรบัณฑิต (วท.บ.) สาขาวิทยาการคอมพิวเตอร์ จากภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2545 จากนั้นในปีการศึกษา 2546 ได้เข้าศึกษาต่อหลักสูตรวิทยาศาสตรมหาบัณฑิต (วท.ม.) สาขาวิทยาศาสตร์คอมพิวเตอร์ ณ ภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย โดยได้รับทุนพัฒนาอาจารย์ (UDC) จากมหาวิทยาลัยอุบลราชธานี