การตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็มโดยใช้ออนโทโลยี

นางสาววิสาข์รัตน์ ศรีสูงเนิน

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2560

Consistency Checking of PDM Network Diagram using Ontology

Miss Wisarat Srisungnoen

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Computer Science
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2017

หัวข้อวิทยานิพนธ์	การตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็มโดยใช้ออนโทโลยี
โดย	นางสาววิสาข์รัตน์ ศรีสูงเนิน
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร. วิวัฒน์ วัฒนาวุฒิ

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์

(รองศาสตราจารย์ ดร. สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ

(รองศาสตราจารย์ ดร. สมชาย ประสิทธิ์จูตระกูล)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(รองศาสตราจารย์ ดร. วิวัฒน์ วัฒนาวุฒิ)

.....กรรมการ

(รองศาสตราจารย์ ดร. พรศิริ หมื่นไชยศรี)

.....กรรมการภายนอกมหาวิทยาลัย

(ดร. เดชานุชิต กตัญญูทวีทิพย์)

วิสาข์รัตน์ ศรีสูงเนิน : การตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็มโดยใช้ออนโทโลยี (Consistency Checking of PDM Network Diagram using Ontology) อ. ที่ปรึกษาวิทยานิพนธ์หลัก: รศ. ดร. วิวัฒน์ วัฒนาวุฒิ, 149 หน้า.

แผนภาพข่ายงานแบบพีดีเอ็ม (Precedence Diagram Method: PDM) เป็นวิธีการสร้างแผนภาพข่ายงานกิจกรรมเพื่อแสดงลำดับและความสัมพันธ์ของกิจกรรมภายใต้งานโครงการ โดยใช้โหนดในการอธิบายระยะเวลาในการดำเนินงานของแต่ละกิจกรรม ที่ถูกเชื่อมโยงกันด้วยลูกศรที่แสดงการพึ่งพาในรูปแบบต่าง ๆ ซึ่งในระหว่างการดำเนินกิจกรรมนั้น อาจมีการพิจารณาเปลี่ยนแปลงระยะเวลาในการดำเนินงานซึ่งอาจจะส่งผลให้ช่วงเวลาของการดำเนินงานในแต่ละกิจกรรมไม่สอดคล้องกัน ดังนั้นงานวิจัยนี้จึงนำเสนอวิธีการตรวจสอบความต้องกันเชิงความหมายของช่วงเวลาในแผนภาพข่ายงานแบบพีดีเอ็ม โดยการสร้างแบบจำลองออนโทโลยีที่อธิบายความหมายของแผนภาพข่ายงานแบบพีดีเอ็มด้วยภาษาอาวล์ (OWL) และออกแบบกฎด้วยภาษาเอสดับบลิวอาร์แอล (SWRL) ที่สามารถสรุปผลจากจากข้อเท็จจริงเชิงตรรกะที่กำหนดขึ้นมาได้อย่างอัตโนมัติ และแสดงผลลัพธ์ของการตรวจสอบความต้องกันด้วยรูปแบบการสืบค้นของภาษาเอสคิวดับเบิลยูอาร์แอล (SQWRL)

5871011721 : MAJOR COMPUTER SCIENCE
KEYWORDS: PRECEDENCE DIAGRAM METHOD / CONSISTENCY CHECKING / ONTOLOGY / OWL / SWRL / SQWRL

WISARAT SRISUNGNOEN: Consistency Checking of PDM Network Diagram using Ontology. ADVISOR: ASSOC. PROF. WIWAT VATANAWOOD, Ph.D., 149 pp.

Precedence Diagram Method (PDM) is a visual representation technique that depicts the activities involved in a project. It is a tool for scheduling activities using nodes to represent activities, time intervals and their connections with arrows to illustrate activity dependencies. During project execution, activities in the project may be changed and may lead to inconsistency of time interval in the project. An ontology describes the relationship between the concepts within a domain. Thus, the ontology could be used to represent knowledge of PDM and check consistencies semantically. This paper proposes PDM Ontology in OWL with the SWRL rules to infer the new knowledge from existing PDM and return the consistency result of PDM using SQWRL.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยการให้ความช่วยเหลือแนะนำของ รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ ซึ่งเป็นอาจารย์ที่ปรึกษาวิทยานิพนธ์ ผู้เขียนจึงขอกราบขอบพระคุณไว้ ณ โอกาสนี้

ผู้เขียนขอกราบขอบพระคุณ รองศาสตราจารย์ ดร.สมชาย ประสิทธิ์จูตระกูล ที่กรุณาให้เกียรติเป็นประธานโดยมี รองศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี และดร.เดชานุชิต กตัญญูทวีทิพย์ เป็นกรรมการในการสอบวิทยานิพนธ์ ซึ่งได้กรุณาตรวจแก้ไขวิทยานิพนธ์ฉบับนี้ให้ถูกต้องสมบูรณ์ยิ่งขึ้น รวมถึงอาจารย์ในภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้ประสิทธิ์ประสาทวิชาความรู้แก่ข้าพเจ้า ตลอดจนเจ้าหน้าที่ภาควิชาวิศวกรรมคอมพิวเตอร์และบัณฑิตวิทยาลัยทุกท่านที่ให้ความสะดวกด้านอำนวยการ และประสานงาน ในการจัดทำวิทยานิพนธ์ให้ผู้เขียนด้วยดีตลอดมา

ท้ายนี้ผู้เขียนขอน้อมรำลึกถึงอำนาจบารมีของคุณพระศรีรัตนตรัย และสิ่งศักดิ์สิทธิ์ทั้งหลายที่อยู่ในสากลโลก อันเป็นที่พึ่งให้ผู้เขียนมีสติปัญญาในการจัดทำวิทยานิพนธ์ให้สำเร็จลุล่วงไปด้วยดี ผู้เขียนขอให้เป็นกตเวทิตาแด่บิดา มารดา ครอบครัวของผู้เขียน สำหรับกำลังใจที่มีค่ายิ่ง รวมถึงขอขอบคุณผู้บังคับบัญชาในสายงาน เพื่อนร่วมงาน และมิตรสหาย ที่คอยติดตามให้กำลังใจ ให้การสนับสนุนและช่วยเหลือในด้านต่าง ๆ และท่านอื่น ๆ ที่มิได้กล่าวนามไว้ ณ ที่นี้ ตลอดจนผู้เขียนหนังสือ และบทความต่าง ๆ ที่ให้ความรู้แก่ผู้เขียนจนทำให้วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยดี

สารบัญ

หน้า

หน้า

หน้า

สารบัญรูป

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การบริหารงานโครงการ (Project Management) จำเป็นต้องอาศัยการวิเคราะห์และออกแบบขั้นตอนการดำเนินงานที่เกี่ยวข้องกันอย่างละเอียดเพื่อให้โครงการสามารถบรรลุข้อจำกัด ทั้ง 3 ด้าน (Triple constraints) ได้แก่ ขอบเขตของงาน (Scope) เวลา (Time) และงบประมาณ (Cost) [1] ในเรื่องของการบริหารจัดการเวลานั้นมักจะเกี่ยวข้องกับการวางแผนการจัดการตารางงาน (Planning Schedule Management) ซึ่งจะต้องมีการกำหนดระยะเวลาเริ่มต้นและสิ้นสุดของแต่ละกิจกรรม เพื่อให้แต่ละกิจกรรมสามารถสำเร็จหรือบรรลุเป้าหมายได้ตามที่วางแผนไว้

เครื่องมือที่นิยมนำมาใช้ในการอธิบายลำดับและขั้นตอนการทำงานตามแผนงานโครงการก็คือ วิธีการสร้างแผนภาพเพื่อแสดงความสัมพันธ์ระหว่างกิจกรรมตามลำดับก่อนหลัง หรือแผนภาพข่ายงานแบบพีดีเอ็ม (Precedence Diagram Method - PDM) โดยใช้โหนดเป็นตัวแทนของกิจกรรม แผนภาพข่ายงานแบบพีดีเอ็มสามารถอธิบายขั้นตอนการทำงานตั้งแต่เริ่มต้นจนสิ้นสุดกิจกรรมอย่างเป็นลำดับ มีการระบุรายละเอียดในแต่ละกิจกรรม เช่น ระยะเวลาดำเนินการหรือจำนวนวัน วันที่เริ่มต้น วันที่สิ้นสุด เงื่อนไขความสัมพันธ์ของแต่ละคู่กิจกรรม เช่น ก่อนที่จะเริ่มทำกิจกรรมต่อไปได้จะต้องรอให้กิจกรรมก่อนหน้าเสร็จเรียบร้อยก่อน เป็นต้น สามารถกำหนดผู้รับผิดชอบของแต่ละกิจกรรม ข้อดีของการใช้แผนภาพข่ายงานแบบพีดีเอ็มคือมีการกำหนดกิจกรรมที่จะต้องดำเนินการไม่มาก ไม่มีข้อจำกัด ง่ายต่อใช้งานและการตีความโดยมนุษย์ เนื่องจากเป็นเครื่องมือที่แสดงระยะเวลาของการดำเนินงานที่เกิดขึ้นจริงได้ด้วยแผนภาพ แต่หากเป็นงานโครงการขนาดใหญ่ที่ประกอบไปด้วยกิจกรรมที่ต้องดำเนินการจำนวนมาก มีระยะเวลาที่ต้องใช้ในการดำเนินงานค่อนข้างนาน โดยเฉพาะงานโครงการที่มีการปรับเปลี่ยนแผนงานที่ค่อนข้างบ่อย เช่น การปรับระยะเวลาดำเนินการของกิจกรรมใดกิจกรรมหนึ่งให้เพิ่มมากขึ้น ซึ่งการปรับระยะเวลาดังกล่าวอาจจะส่งผลกระทบกับกิจกรรมอื่น ๆ ที่มีความสัมพันธ์กันในรูปแบบต่าง ๆ ดังนั้นการตรวจสอบความต้องกันของแผนรูปที่มีการเปลี่ยนแปลงด้วยมนุษย์จึงอาจจะไม่สะดวกนัก

ดังนั้นงานวิจัยนี้จึงนำเสนอวิธีการตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม โดยการสร้างแบบจำลองออนโทโลยีที่อธิบายความหมายของแผนภาพข่ายงานแบบพีดีเอ็ม ด้วยภาษาอาวล์ (OWL) และออกแบบกฎด้วยภาษาเอสดับบลิวอาร์แอล (SWRL) ที่สามารถสรุปผลจากข้อเท็จจริงเชิงตรรกะที่กำหนดขึ้นมาได้อย่างอัตโนมัติ อีกทั้งยังออกแบบกฎเพื่อตรวจสอบความ

ต้องกันของแผนภาพข่ายงานแบบพีดีเอ็มด้วยภาษา (SQWRL) เนื่องจากออนโทโลยีมีความสามารถในการอธิบายแนวคิดเชิงความหมายของแผนภาพดังกล่าวได้ และยังถูกเก็บอยู่ในรูปแบบที่คอมพิวเตอร์สามารถเข้าใจและประมวลผลได้อีกด้วย

1.2 วัตถุประสงค์ของงานวิจัย

1. ออกแบบกฎตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม ด้วยวิธีออนโทโลยี
2. เพื่อพัฒนาเครื่องมือเพื่อตรวจสอบความต้องกันของแผนภาพพีดีเอ็ม ด้วยเว็บแอพพลิเคชัน

1.3 ขอบเขตการวิจัย

1. แผนภาพข่ายงานแบบพีดีเอ็มที่ใช้ในการตรวจสอบจะต้องเป็นโหนดกิจกรรมที่มีการระบุระยะเวลาในการดำเนินการ วันที่เริ่มทำงานที่เร็วที่สุด วันที่งานเสร็จได้เร็วที่สุด วันที่เริ่มงานที่ช้าที่สุด วันที่งานเสร็จได้ช้าที่สุด มีการระบุเส้นเชื่อมระหว่างกิจกรรมใด ๆ และข้อมูลต่าง ๆ ในรูปแบบไฟล์ Excel ตามที่ออกแบบไว้
2. ออกแบบและพัฒนาออนโทโลยีของแผนภาพข่ายงานแบบพีดีเอ็มด้วยโปรแกรมโปรเทเจและจัดเก็บเป็นไฟล์ภาษาอาวล์
3. การสร้างกฎการตรวจความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็มด้วยภาษาเอสดับเบิลยูอาร์แอล
4. สามารถตรวจสอบความต้องกันกรณีที่มีการเปลี่ยนแปลงข้อมูลในแผนภาพข่ายงานแบบพีดีเอ็มดังต่อไปนี้
   - ระยะเวลาในการดำเนินงานแต่ละกิจกรรม (Duration)
   - วันเริ่มต้น และสิ้นสุดของแต่ละกิจกรรม (ES, EF, LS, LF)
   - ประเภทความสัมพันธ์ของกิจกรรม (FS, SS, FF, SF)
   - การจัดสรรหรือมอบหมายงานให้กับบุคคลากร (Resource)
5. เครื่องมือตรวจสอบมีความสามารถดังต่อไปนี้
   - นำเข้าโครงร่างออนโทโลยีต้นแบบของแผนภาพข่ายงานแบบพีดีเอ็มในรูปแบบของภาษาอาวล์
   - นำเข้าข้อมูลแผนภาพข่ายงานแบบพีดีเอ็มที่ต้องการตรวจสอบความต้องกันที่อยู่ในรูปแบบไฟล์ Excel
   - นำเข้าข้อมูลเข้ากฎเอสดับเบิลยูอาร์แอล (SWRL) ที่ใช้ในการอนุมานองค์ความรู้ที่ซ่อนเร้น (Inference Rules)
   - ตรวจสอบความต้องกัน

- ประมวลผลและแสดงผลลัพธ์ของความต้องกัน
- พัฒนาด้วยระบบเว็บแอพพลิเคชัน

6. ใช้กรณีศึกษา 4 กรณี โดยกำหนดให้ทำการทดสอบในกรณีที่แผนภาพข่ายงานแบบพีดีเอ็มมีความต้องกัน 2 กรณี และไม่ต้องกัน 2 กรณี ทั้งนี้จะทำการตรวจสอบในกรณีที่มีการเปลี่ยนแปลงข้อมูลต่าง ๆ ตามข้อ 4. ได้แก่ ระยะเวลาในการดำเนินงาน วันเริ่มต้นและสิ้นสุดของกิจกรรม ประเภทความสัมพันธ์ของกิจกรรม และการจัดสรรหรือมอบหมายงานให้กับบุคคลากร

1.4 ขั้นตอนการวิจัย

1. ศึกษาองค์ประกอบแผนภาพข่ายงานแบบพีดีเอ็มและการใช้งาน
2. ศึกษากระบวนการออกแบบและพัฒนาออนโทโลยี
3. ออกแบบเมตาโมเดลของแผนภาพข่ายงานแบบพีดีเอ็ม
4. ออกแบบออนโทโลยีจากเมตาโมเดลของแผนภาพพีดีเอ็ม
5. ออกแบบกฎที่ใช้ในการตรวจสอบความต้องกันของแผนภาพพีดีเอ็มด้วยภาษาเอสดับเบิลยูอาร์แอล
6. พัฒนาออนโทโลยีด้วยโปรแกรมประยุกต์โปรเทเจ
7. ออกแบบและพัฒนาระบบเว็บแอพพลิเคชั่นสำหรับผู้ใช้งาน ทั้งรูปแบบของข้อมูลนำเข้าหน้าจอการแสดงผล และส่วนที่เชื่อมต่อกับออนโทโลยี ตลอดจนกฎที่ใช้ในการตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม
8. ทดสอบระบบและประเมินผล
9. สรุปผลการวิจัยและข้อเสนอแนะ
10. ตีพิมพ์ผลงานทางวิชาการ
11. จัดทำวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็มด้วยกฎที่ออกแบบได้
2. สามารถนำเข้าข้อมูลแผนภาพข่ายงานแบบพีดีเอ็ม ในรูปแบบของไฟล์ภาษาอาวล์ เพื่อตรวจสอบความต้องกัน ด้วยเครื่องมือเว็บแอพพลิเคชัน

1.6 ลำดับการจัดเรียงเนื้อหาในวิทยานิพนธ์

วิทยานิพนธ์นี้มีทั้งหมด 6 บท ดังต่อไปนี้ บทที่ 1 บทนำความเป็นมาและความสำคัญของปัญหา วัตถุประสงค์ของการวิจัย ขอบเขตของการวิจัย ประโยชน์ที่คาดว่าจะได้รับและผลงานตีพิมพ์

บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง บทที่ 3 การออกแบบออนโทโลยีและกฎของแผนภาพข่ายงานแบบพีดีเอ็ม บทที่ 4 การออกแบบและพัฒนาระบบ บทที่ 5 วิธีการประเมินและวัดผลการทดลอง และบทที่ 6 สรุปผลการวิจัย ข้อเสนอแนะและแนวทางสำหรับอนาคต

1.7 ผลงานที่ตีพิมพ์จากวิทยานิพนธ์

1. Wisarat Srisungnoen; Wiwat Vatanawood, “An Ontology-based Knowledge Acquisition for PDM” in Proceedings - 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2018, 2018, pp 287-292

2. Wisarat Srisungnoen; Wiwat Vatanawood, “Dependency Types Validation of PDM using Ontology” Eng.J., vol.22, no.5, 2018

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

ทฤษฎีที่เกี่ยวข้องกับการออกแบบและพัฒนางานวิจัยนี้ ประกอบด้วยทฤษฎีแผนภาพข่ายงานแบบพีดีเอ็ม ทฤษฎีความต้องกัน การสร้างองค์ความรู้โดยวิธีออนโทโลยีด้วยภาษาอาวล์ และการสร้างกฎด้วยภาษาเอสดับเบิลยูอาร์แอล เพื่อใช้ในการตรวจสอบความต้องกัน

2.1.1 แผนภาพข่ายงานแบบพีดีเอ็ม (PDM)

แผนภาพข่ายงานแบบพีดีเอ็ม [2] เป็นส่วนต่อขยายของแผนภาพเอโอเอ็น (Activity on Node : AON) ซึ่งจัดเป็นแผนผังงานข่าย (Network Diagram) ชนิดหนึ่ง ที่แสดงถึงกิจกรรมทั้งหมดของโครงการด้วยโหนด (Node) ที่ถูกเชื่อมโยงกันด้วยเส้นเชื่อม (Edge) ซึ่งเส้นเชื่อมนี้จะแสดงความสัมพันธ์ (Dependency) ของการเริ่มต้นและสิ้นสุดของการดำเนินกิจกรรมในแต่ละคู่กิจกรรม แผนภาพข่ายงานแบบพีดีเอ็มจะแสดงให้เห็นขั้นตอนต่าง ๆ ของโครงการตั้งแต่จุดเริ่มต้นจนถึงจุดสิ้นสุดของโครงการ ซึ่งมักถูกนำมาใช้ในงานโครงการที่เกี่ยวข้องกับการบริหารจัดการตารางเวลา (Project Schedule) หรือการทำแผนบริหารจัดการเวลา ตั้งแต่ปี ค.ศ. 1964 ซึ่งแผนภาพข่ายงานแบบพีดีเอ็ม เป็นที่นิยมมากกว่าเอโอเอ็นเพราะใช้งานง่ายและสามารถทำงานแบบซ้อนทับกันได้ (Overlap) และยังนิยมถูกนำไปพัฒนาต่อยอดให้เป็นเครื่องมือที่ช่วยในการวางแผนในโปรแกรมสำเร็จรูปอีกด้วย [3]

ถึงแม้ว่าแผนภาพข่ายงานแบบพีดีเอ็ม จะมีความคล้ายคลึงกับวิธีซีพีเอ็ม (Critical Path Method : CPM) ตรงที่มีการกำหนดประเภทของการดำเนินงานเป็น 4 รูปแบบ ได้แก่ วันที่เริ่มทำงานที่เร็วที่สุด (Earliest Start : ES) วันที่งานเสร็จได้เร็วที่สุด (Earliest Finish : EF) วันที่เริ่มงานที่ช้าที่สุด (Latest Start : LS) และ วันที่งานเสร็จได้ช้าที่สุด (Latest Finish : LF) แต่จุดเด่นที่สำคัญของวิธีแผนภาพข่ายงานแบบพีดีเอ็มคือความสัมพันธ์ระหว่างกิจกรรมที่สามารถรองรับได้ถึง 4 รูปแบบ คือ สิ้นสุด-เริ่มต้น (Finish-to-Start) เริ่มต้น-เริ่มต้น (Start-to-Start) เริ่มต้น-สิ้นสุด (Start-to-Finish) และ สิ้นสุด-สิ้นสุด (Finish-to-Finish) ในขณะที่ CPM มีรูปแบบของความสัมพันธ์ของกิจกรรมเพียงรูปแบบเดียวคือ สิ้นสุด-เริ่มต้น (Finish-to-Start) นอกจากนี้ยังสามารถคำนวณเวลาที่ใช้ในการรอคอย (Lag time) เพื่อให้กิจกรรมก่อนหน้านั้นสิ้นสุดลงก่อนได้ หรือเวลาที่นำหน้า (Lead time) เพื่อให้กิจกรรมที่ตามหลังสามารถเริ่มต้นงานได้ก่อนที่กิจกรรมก่อนหน้าจะสิ้นสุดลง หรือเป็นลักษณะการทำงานที่ทำพร้อมกัน (Overlapping) ควบคู่กันไปทั้งสองกิจกรรม การที่แผนภาพสามารถ

คำนวณเวลารอคอยและเวลานำหน้าได้ก็จะช่วยให้แผนนั้นมีความยืดหยุ่นมากยิ่งขึ้น ซึ่งถือว่าเป็นเครื่องมือที่ช่วยให้ผู้บริหารสามารถที่จะบริหารจัดงานโครงการเป็นไปอย่างมีประสิทธิภาพมากขึ้น เช่น การพิจารณาจัดสรรบุคลากร (Resource) ตามลำดับความสำคัญ และยังช่วยให้ลูกค้าสามารถพิจารณาเร่งหรือขยายเวลาของการดำเนินงานได้อีกด้วย นอกจากนี้สถาบันการบริหารการจัดการโครงการ (The Project Management Institute : PMI) ได้จัดพิมพ์เอกสารองค์ความรู้ของการบริหารจัดการโครงการ (The Project Management Body of Knowledge : PMBOK) [4] ซึ่งได้รวบรวมองค์ความรู้ในการบริหารจัดการโครงการเพื่อเป็นแนวทางในการบริหารงานโครงการให้กับผู้จัดการโครงการ เนื้อหาส่วนหนึ่งยังได้กล่าวถึงและแนะนำแผนภาพข่ายงานแบบพีดีเอ็ม ให้เป็นเครื่องมือในการจัดลำดับของกิจกรรมอีกด้วย

การกำหนดสัญลักษณ์ของแผนภาพข่ายงานแบบพีดีเอ็มพื้นฐาน ประกอบไปด้วย

- กิจกรรม (Activity) คือ กิจกรรมที่ต้องทำในโครงการ
- ระยะเวลา (Duration: DUR) คือ ระยะเวลาที่ใช้ในการดำเนินกิจกรรม มีหน่วยเป็นวัน (day)
- วันที่เริ่มต้นที่เร็วที่สุด (Early start: ES) คือ วันที่ที่เร็วที่สุดที่สามารถเริ่มต้นทำกิจกรรม
- วันที่สิ้นสุดที่เร็วที่สุด (Early finish: EF) คือ วันที่ที่เร็วที่สุดที่สามารถทำกิจกรรมให้แล้วเสร็จ
- วันที่เริ่มต้นที่ช้าที่สุด (Late start: LS) คือ วันที่ที่ช้าที่สุดที่สามารถเริ่มต้นทำกิจกรรม
- วันที่สิ้นสุดที่ช้าที่สุด (Late finish: LF) คือ วันที่ที่ช้าที่สุดที่สามารถทำกิจกรรมให้เสร็จ
- เวลาลอยรวม (Total float: TF) คือ เวลาของกิจกรรมที่สามารถเลื่อนเวลาเริ่มต้นออกไปจากที่กำหนดไว้ โดยไม่มีผลกระทบต่อเวลาแล้วเสร็จของโครงการที่จะเสร็จล่าช้ากว่าที่กำหนด แต่อาจทำให้เวลาเริ่มต้นเร็วที่สุดของกิจกรรมที่อยู่ข้างหลังต้องเลื่อนตามไปด้วยสามารถหาได้จาก TF = LF - EF หรือ TF = LS – ES

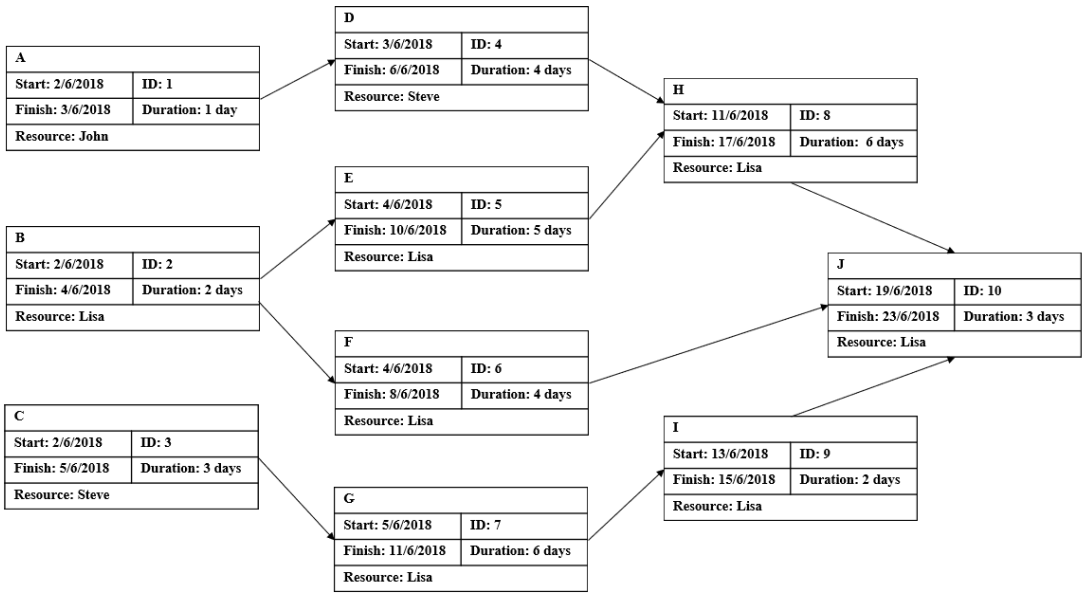
Early start	Duration (day)	Early finish
Activity		
Late start	Total float	Late finish

รูปที่ 2.1 ส่วนประกอบพื้นฐานของแผนภาพข่ายงานแบบพีดีเอ็ม

คำนิยามศัพท์อื่น ๆ ที่เกี่ยวข้องกับแผนภาพข่ายงานแบบพีดีเอ็ม ที่ไม่ได้แสดงในรูปที่ 2.1 นี้คือ

- เวลาลอยอิสระ (Free Float : FF) คือ เวลาของกิจกรรมที่สามารถเลื่อนเวลาเริ่มต้นช้าออกไปจากที่กำหนดไว้ โดยไม่มีผลกระทบต่อเวลาแล้วเสร็จของ โครงการที่จะล่าช้ากว่ากำหนด และไม่มีผลทำให้กำหนดเวลาเริ่มต้นของกิจกรรมอื่น ๆที่ อยู่ข้างหลังต้องเลื่อนตามไปด้วย สามารถหาได้จาก FF = ES ของเส้นทางถัดไปข้างหน้า – EF ของเส้นทางที่พิจารณาอยู่
- กิจกรรมวิกฤต (Critical Activity) คือ กิจกรรมที่มีความสำคัญ ซึ่งถ้ากิจกรรมนั้นเกิดการล่าช้าจะมีผลทำให้เวลาทั้งหมดของโครงการล่าช้ากว่าที่ได้วางแผนไว้
- เส้นทางวิกฤติ (Critical Path) คือ เส้นทางของงานหรือกิจกรรมในโครงการตั้งแต่จุดเริ่มต้นจนถึงจุดสิ้นสุดที่ใช้เวลามากที่สุด หรือเป็นเส้นทางที่ยาวที่สุดของผังเครือข่าย ถ้าเวลาที่ใช้ในเส้นทางเหล่านี้ล่าช้าจะเป็นผลทำให้โครงการล่าช้ากว่าที่ได้วางแผนไว้ เพื่อหาเส้นวิกฤตจะต้องคำนวณค่าเวลาเริ่มและเวลาสิ้นสุดของแต่ละกิจกรรม
- ลำดับก่อนหน้า (Precedence) คือ ลำดับความสำคัญที่ถูกนิยามไว้ว่า กิจกรรมที่ตามหลังไม่สามารถเริ่มต้นได้ จนกว่ากิจกรรมก่อนหน้าทั้งหมดนั้นจะแล้วเสร็จ
- กิจกรรมก่อนหน้า (Predecessor Activity) คือ กิจกรรมใด ๆ ที่ต้องทำให้แล้วเสร็จก่อนที่จะเริ่มต้นกิจกรรมที่กำหนดถัดไป
- กิจกรรมตามหลัง (Successor Activity) คือ กิจกรรมใด ๆ ที่ไม่สามารถเริ่มต้นได้จนกว่ากิจกรรมที่กำหนดก่อนหน้าจะแล้วเสร็จ

เมื่อนำแต่ละกิจกรรมจากรูปที่ 2.1 มาเรียงต่อกันเป็นลำดับเพื่อประกอบกันเป็นแผนภาพข่ายงานแบบพีดีเอ็ม ที่มีการพึ่งพา (Dependency) หรือความสัมพันธ์ระหว่างกิจกรรม ซึ่งมีการกำหนดประเภทของความสัมพันธ์ไว้ทั้งหมด 4 รูปแบบ คือ Finish to Start (FS), Start to Start (SS), Finish to Finish (FF) และ Start to Finish (SF) ซึ่งถือว่าเป็นจุดเด่นที่สำคัญของแผนภาพข่ายงานแบบพีดีเอ็ม ในแผนภาพตัวอย่างจะเป็นความสัมพันธ์ในรูปแบบของการทำกิจกรรมแรกให้เสร็จก่อนที่จะเริ่มกิจกรรมต่อไป (Finish to Start : FS) ซึ่งสามารถแทนค่าต่าง ๆ ลงในแผนภาพข่ายงานแบบพีดีเอ็ม ดังรูปที่ 2.2 จากรูปตัวอย่างแผนภาพข่ายงานแบบพีดีเอ็มจะพบว่า ประกอบไปด้วยหลายกิจกรรม ได้แก่ กิจกรรมที่ชื่อ A, B, C, D, E, F, G, H, I, และ J ซึ่งแต่ละกิจกรรมจะมีการกำหนดวันที่เริ่มต้น วันที่สิ้นสุด ระยะเวลาในการดำเนินงาน รวมไปถึงการมอบหมายผู้ที่รับผิดชอบกิจกรรมนั้น ๆ จากนั้นแต่ละกิจกรรมก็จะถูกเชื่อมโยงกันเป็นลำดับด้วยรูปแบบการพึ่งพาประเภทสิ้นสุด-เริ่มต้น

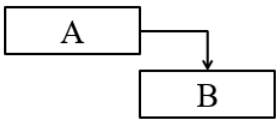
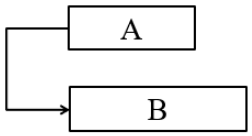
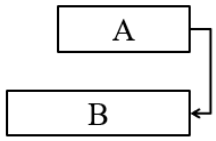
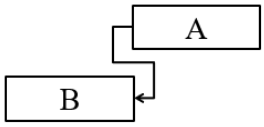
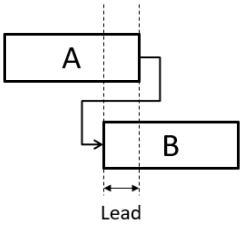
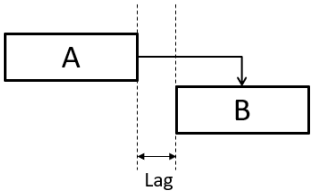
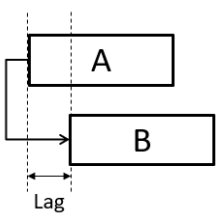


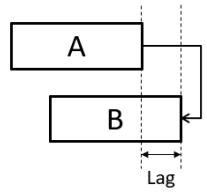
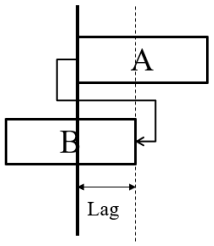
รูปที่ 2.2 ตัวอย่างแผนภาพข่ายงานแบบพีดีเอ็ม

นอกจากประเภทความสัมพันธ์ทั้ง 4 รูปแบบนี้แล้ว แผนภาพข่ายงานแบบพีดีเอ็มยังสามารถใช้ในการคำนวณเวลารอคอย (Lag) และเวลานำ (Lead) ได้อีกด้วย การใช้งานของเวลารอคอยและเวลานำสำหรับงานใด ๆ นั้น เป็นการกำหนดระยะเวลาของกิจกรรมก่อนหน้า (Predecessor) ว่าจะให้เริ่มต้นช้าหรือเร็ว ขึ้นอยู่กับประเภทความสัมพันธ์ที่ได้ระบุไว้ว่าเป็นแบบใดซึ่งสามารถอธิบายความหมายได้ดังนี้

- เวลารอคอย (Lag) เป็นรูปแบบกิจกรรมที่ล่าช้า (Delay) โดยจะเกิดขึ้นเมื่อกิจกรรมที่ตามหลัง (Successor) มานั้น จะเริ่มต้นได้ต่อเมื่อกิจกรรมก่อนหน้า (Predecessor) สิ้นสุดลงโดยที่มีการล่าช้ากว่าแผน ทำให้ต้องเลื่อนเวลาเริ่มต้นของกิจกรรมที่ตามหลังออกไป ซึ่งเวลารอคอยนี้สามารถเกิดขึ้นได้กับทุกรูปแบบความสัมพันธ์
- เวลานำ (Lead) เป็นรูปแบบของกิจกรรมที่ตามหลัง (Successor) ซึ่งจะเหลื่อมกับกิจกรรมก่อนหน้า (Predecessor) นั่นหมายความว่ากิจกรรมที่ตามหลังนั้น จะสามารถเริ่มต้นทำได้โดยที่งานก่อนหน้ายังไม่เสร็จสิ้น ซึ่งเวลานำนี้สามารถเกิดขึ้นกับกิจกรรมที่มีความสัมพันธ์ระหว่างกันเป็นแบบ Finish to Start (FS) เท่านั้น [5]

ตารางที่ 2.1 ความหมายและสัญลักษณ์ของประเภทความสัมพันธ์ของแผนภาพพีดีเอ็ม [2]

ลำดับ	ประเภทการพึ่งพา (Dependency)	สัญลักษณ์	คำอธิบาย
1.	Finish-to-Start (FS)		กิจกรรม A จะต้องดำเนินการให้แล้วเสร็จก่อนที่จะเริ่มกิจกรรม B หรืออีกนัยหนึ่งคือ B จะเริ่มทำได้ต่อเมื่อ A ทำเสร็จแล้ว
2.	Start-to-Start (SS)		กิจกรรม B จะเริ่มทำงานได้ต่อเมื่อกิจกรรม A เริ่มดำเนินการ หรือทั้ง 2 กิจกรรมจะต้องเริ่มพร้อมกัน
3.	Finish-to-Finish (FF)		เมื่อกิจกรรม A ดำเนินการแล้วเสร็จ กิจกรรม B ก็ต้องแล้วเสร็จด้วยเช่นกัน หรือ กิจกรรม B จะต้องเสร็จพร้อมกับกิจกรรม A
4.	Start-to-Finish (SF)		กิจกรรม B จะดำเนินการแล้วเสร็จเมื่อกิจกรรม A เริ่มดำเนินการ
5.	Finish-to-Start (FS) with Lead		กิจกรรม B จะสามารถเริ่มต้นได้โดยที่กิจกรรม A ยังไม่แล้วเสร็จ หรือเรียกว่า Finish-to-Start (Negative Lag)
6.	Finish-to-Start (FS) with Lag		กิจกรรม B จะเริ่มต้นล่าช้าเนื่องจากมีระยะเวลารอคอยหลังจากที่กิจกรรม A แล้วเสร็จ หรือเรียกว่า Finish-to-Start (Positive Lag)
7.	Start-to-Start (SS) with Lag		กิจกรรม B จะเริ่มล่าช้า เนื่องจากมีระยะเวลารอคอยให้กิจกรรม A เริ่มต้นไปก่อนสักระยะหนึ่ง

ลำดับ	ประเภทการพึ่งพา (Dependency)	สัญลักษณ์	คำอธิบาย
8.	Finish-to-Finish (FF) with Lag		กิจกรรม B จะเสร็จสิ้น เนื่องจากมีระยะเวลารอคอยให้กิจกรรม A เสร็จสิ้นไปก่อนสักระยะหนึ่ง
9.	Start-to-Finish (SF) with Lag		กิจกรรม B จะสิ้นสุดได้ต่อเมื่อกิจกรรม A เริ่มต้น โดยที่กิจกรรม B มีการสิ้นสุดล่าช้ากว่าแผน หรือกิจกรรม A เริ่มไปแล้วระยะหนึ่ง กิจกรรม B จึงจะสิ้นสุด

การคำนวณระยะเวลาในการดำเนินกิจกรรมในแผนภาพข่ายงานแบบพีดีเอ็ม [6] อย่างน้อยจะต้องมีการคำนวณ Early Start, Late Start, Early Finish, Latest Finish จากนั้นจึงคำนวณเวลาลอยรวม (Total float) ของแต่ละกิจกรรม เวลาลอยรวมนี้นับว่ามีความสำคัญต่อการบริหารจัดการโครงการเป็นอย่างมาก โดยมากจะนิยมใช้หน่วยนับเป็นจำนวนวันในการติดตามงานในโครงการ ซึ่งการคำนวณระยะเวลาในการดำเนินกิจกรรมแบ่งออกเป็น 2 รูปแบบ

1) การคำนวณเส้นทางไปข้างหน้า (Forward Pass) เป็นการกำหนดตารางเวลาตามเส้นทางไปข้างหน้า เพื่อหา ระยะเวลาที่เร็วที่สุดในการเริ่มต้น (ES) และระยะเวลาที่เร็วที่สุดในการทำกิจกรรมนั้นให้แล้วเสร็จ (EF) รวมไปถึงการหาระยะเวลาที่ต้องใช้ในการดำเนินกิจกรรมทั้งหมดในโครงการ การคำนวณเส้นทางไปข้างหน้ามีสูตรและกฎดังนี้

   - ระยะเวลาที่เร็วที่สุดในการเริ่มต้น (ES)

   ES = Max EF of related predecessor

   จากสูตรสามารถอธิบายได้ว่า ก่อนที่กิจกรรมจะสามารถเริ่มได้ ทุกกิจกรรมที่มาก่อนหน้าของกิจกรรมที่ต้องการหา ES ต้องทำเสร็จ โดยที่กิจกรรมเริ่มต้นมีค่า ES และ EF เป็น 0

   ถ้ากิจกรรมมีเพียงกิจกรรมเดียวที่มาก่อน (Predecessor) ค่า ES ของกิจกรรมมีค่าเท่ากับ EF ของกิจกรรมที่มาก่อน

   ถ้ากิจกรรมนั้นมีหลายกิจกรรมที่มาก่อน (Predecessor) ค่า ES ของกิจกรรมคือ ค่ามากที่สุดของค่า EF (Max EF) ของกิจกรรมที่มาก่อนกิจกรรมที่กำลังพิจารณาทุกกิจกรรม

- ระยะเวลาที่เร็วที่สุดในการเสร็จสิ้น (EF)

EF = ES + Duration

จากสูตรสามารถคำนวณ EF ของกิจกรรมได้จากผลรวมของ ES และระยะเวลาดำเนินงาน (Duration) ของกิจกรรม

2) การคำนวณเส้นทางย้อนกลับ (Backward Pass) ซึ่งหลังจากที่คำนวณเส้นทางไปข้างหน้าเสร็จสิ้นแล้ว จะได้ของระยะเวลาที่สิ้นสุดที่เร็วที่สุดของกิจกรรมสุดท้าย (EF) ซึ่งจะเท่ากับระยะเวลาที่สิ้นสุดที่ช้าที่สุดของกิจกรรมสุดท้าย (LF) หรือระยะเวลาการดำเนินของโครงการทั้งหมด ซึ่งการคำนวณจะต้องเริ่มจากกิจกรรมท้ายสุดของแผนภาพ เพื่อหาระยะเวลาที่ช้าที่สุดในการเสร็จสิ้น (LF) แล้วจึงทำการหาระยะเวลาที่ช้าที่สุดในการเริ่มต้น (LS) การคำนวณเส้นทางย้อนกลับมีสูตรและกฎดังนี้

- ระยะเวลาที่ช้าที่สุดในการเริ่มต้น (LS)

LS = LF - Duration

จากกิจกรรมสุดท้าย จะได้ว่า EF = LF ดังนั้นจากสูตรสามารถคำนวณ LS ของกิจกรรมได้จากการลบระยะเวลาดำเนินงาน (Duration) ของกิจกรรม

- ระยะเวลาที่ช้าที่สุดในการเสร็จสิ้น (LF)

LF = Min LS of related successor

เมื่อได้ระยะเวลาที่ช้าที่สุดในการเริ่มต้น (LS) ของกิจกรรมสุดท้าย ถัดมาในการหา LF ของกิจกรรมถัดไป ให้คำนวณจาก ค่า LS ของกิจกรรมที่ตามหลัง (successor) ถ้ากิจกรรมนั้นมีกิจกรรมที่ตามมาเพียงกิจกรรมเดียว ค่า LF ของกิจกรรมนั้นเท่ากับค่า LS ของกิจกรรมที่ตามมา แต่ถ้ากิจกรรมนั้นมีกิจกรรมที่ตามมามากกว่าหนึ่งกิจกรรม ค่า LF นั้นจะเท่ากับค่าของกิจกรรมที่ตามมาที่มีค่า LS น้อยที่สุด (Min LS)

นอกจากการออกแบบและวางแผนกิจกรรมในโครงการโดยใช้แผนภาพข่ายงานแบบพีดีเอ็มแล้ว การบริหารจัดการทรัพยากร (Resource) นับว่าเป็นปัจจัยที่สำคัญที่ส่งผลให้การดำเนินโครงการในแต่ละกิจกรรมสำเร็จลุล่วงไปได้ ซึ่งข้อจำกัด [7] ที่พบในการบริหารจัดการทรัพยากรก็คือ

- จำนวนทรัพยากรที่มีอยู่อย่างจำกัด (Limited number of resources)
- จำนวนทรัพยากรที่สามารถใช้งานได้มีอยู่อย่างจำกัด (Limited availability of resources)

- การขาดงานของพนักงานอย่างไม่มีกำหนดแน่นอน เช่น พนักงานลาป่วย ลาคลอด (Unplanned absence of resources)
- การสูญเสียทรัพยากรโดยไม่ได้มีการวางแผน เช่น พนักงานลาออก หรือ เสียชีวิต (Unplanned loss of resources)
- การไล่พนักงานออก โดยไม่มีการการวางแผน (Unplanned turnover of personnel)

ปัญหาต่าง ๆ เหล่านี้สามารถเกิดขึ้นได้ในระหว่างที่ดำเนินโครงการ ดังนั้นจึงควรมีการบริหารจัดการทรัพยากรอย่างถูกต้อง มีประสิทธิภาพ เพื่อไม่ให้แผนงานในการดำเนินโครงการล่าช้า

ส่วนประกอบของทรัพยากรที่มักใช้ในการวางแผนงานโครงการจะต้องประกอบไปด้วยข้อมูลต่าง ๆ ดังนี้

- ชื่อของทรัพยากร (Resource name) คือ ชื่อเรียกเฉพาะเจาะจงของทรัพยากร ในการบริหารงานโครงการโดยมากจะหมายถึงชื่อบุคคลากร
- ประเภทของทรัพยากร (Resource type) คือ ประเภทของทรัพยากร ซึ่งแบ่งออกเป็น 3 ประเภท ได้แก่ บุคลากร (Work) เครื่องจักรอุปกรณ์ต่าง ๆ (Material) และค่าใช้จ่าย (Cost)
- กลุ่มของทรัพยากร (Resource group) คือ การจัดกลุ่มทรัพยากร ซึ่งสามารถจัดกลุ่มได้ตามหน้าที่ หรือแผนกที่สังกัด เช่น ฝ่ายขาย ฝ่ายจัดหา ฝ่ายออกแบบ เป็นต้น
- จำนวนงานที่ทรัพยากรสามารถทำได้สูงสูด (Maximum unit) คือ จำนวนงานทั้งหมดที่ทรัพยากรสามารถทำได้ มีหน่วยเป็น จำนวนวัน
- อัตรามาตรฐาน (Standard rate) คือ อัตราค่าจ้างปกติที่จ่ายให้กับบุคลากร ในชั่วโมงทำงานปกติ
- อัตราล่วงเวลา (Overtime rate) คือ อัตราค่าจ้างพิเศษที่จ่ายให้กับบุคลากร ที่มีการทำงานล่วงเวลา
- ค่าใช้จ่ายต่อการทำงาน (Cost per use) คือค่าใช้จ่ายที่ใช้ในการคำนวณต้นทุนที่เกิดขึ้นจริง สำหรับทรัพยากรที่เป็นบุคลากร
- การกำหนดการคำนวณค่าใช้จ่าย (Accrue at) คือการกำหนดว่าค่าใช้จ่ายต่อการทำงานทั้งอัตรามาตรฐานและอัตราล่วงเวลา จะต้องถูกคิดเมื่อใด ซึ่งแบ่งออกเป็น 3 ช่วงเวลา คือ คำนวณตอนเริ่มงาน คำนวณหลังจากงานเสร็จสิ้น และคำนวณตามสัดส่วนที่เกิดขึ้นจริง (Prorated)

2.1.2 ความต้องกัน (Consistency)

ความต้องกัน คือ ความประสานกันทำงานของแบบจำลองที่เป็นไปอย่างสอดคล้องกัน [8] ซึ่งในมุมมองของแบบจำลองหรือแผนภาพยูเอ็มแอลนั้น ได้แบ่งประเภทของความต้องกันออกเป็น 3 มุมมองดังนี้ [9]

1. มุมมองในแนวคิดเชิงนามธรรมของแบบจำลองในรูปแบบต่าง ๆ ได้แก่

- ความต้องกันในแนวนอน (Horizontal consistency) หรือเรียกอีกอย่างหนึ่งว่า ความต้องกันภายในระหว่างแบบจำลอง (Intra-model consistency) หมายถึง ความต้องกันระหว่างแบบจำลองหรือแผนรูปที่มีแนวคิดเชิงนามธรรมของแบบจำลองในระดับเดียวกันและอยู่ภายใต้เวอร์ชันเดียวกัน
- ความต้องกันในแนวตั้ง (Vertical consistency) หรือเรียกอีกอย่างหนึ่งว่า ความต้องกันภายนอกระหว่างแบบจำลอง (Inter-model consistency) หมายถึง ความต้องกันระหว่างแบบจำลองหรือแผนภาพ ที่มีแนวคิดเชิงนามธรรมของแบบจำลองที่ต่างระดับกัน แต่อยู่ภายใต้เวอร์ชันเดียวกัน
- ความต้องกันตามวิวัฒนาการ (Evolution consistency) ความต้องกันประเภทนี้จะถูกนำมาใช้ก็ต่อเมื่อเวอร์ชันของแบบจำลองหรือแผนภาพนั้นมีการปรับปรุง แก้ไข เพิ่มเติมให้เป็นอีกเวอร์ชันหนึ่ง ซึ่งจะเป็นความต้องกันของแบบจำลองประเภทเดียวกันแต่ต่างเวอร์ชัน

2. มุมมองในเชิงวากยสัมพันธ์ (Syntactic) และเชิงความหมาย (Semantic)

- ความต้องกันเชิงวากยสัมพันธ์ (Syntactic consistency) เป็นการตรวจสอบความต้องกันระหว่างข้อกำหนด (Specification) ที่ได้ออกแบบ กับไวยากรณ์เชิงนามธรรม (Abstract syntax) ที่ได้ระบุไว้ในเมตาดาต้า
- ความต้องกันเชิงความหมาย (Semantic consistency) เป็นการตรวจสอบความต้องกันเชิงความหมายตามที่ได้ระบุไว้ในเมตาดาต้า ซึ่งจะเน้นไปในเชิงพฤติกรรมของแผนภาพ (Behavior of diagram) ให้มีความเข้ากันได้อย่างมีความหมาย (Semantically compatible)

3. มุมมองในเชิงสังเกตการณ์ (Observation) และเชิงร้องขอ (Invocation)

- ความต้องกันในเชิงสังเกตการณ์ (Observation consistency) เป็นความต้องกันระหว่างคลาสลูก (Subclass) และคลาสแม่ (Superclass) โดยที่อินแสตนซ์ของคลาสลูกจะต้องถูกถ่ายทอดพฤติกรรม และสามารถใช้งานได้เช่นเดียวกับอินสแตนซ์ของคลาสแม่
- ความต้องกันในเชิงร้องขอ (Invocation consistency) เป็นความต้องกันระหว่างคลาสลูก (Subclass) และคลาสแม่ (Superclass) โดยที่อินแสตนซ์ของคลาสลูกจะสามารถใช้งานได้ก็ต่อเมื่อมีการร้องขอจากคลาสแม่

ในงานวิจัยนี้มุ่งเน้นตรวจสอบความต้องกันเชิงวากยสัมพันธ์ (Syntactic Consistency) และเชิงความหมาย (Semantic consistency) ของแผนภาพข่ายงานแบบพีดีเอ็มเป็นหลัก โดยการอธิบายความหมายของแผนภาพในรูปแบบของแบบจำลองออนโทโลยี แล้วจึงนำแบบจำลองดังกล่าวมาตรวจสอบด้วยกฎซึ่งเป็นข้อจำกัดหรือเงื่อนไขที่แผนภาพนั้นจำเป็นจะต้องมีเพื่อรักษาความต้องกัน

2.1.3 ออนโทโลยี (Ontology)

ออนโทโลยี เป็นการอธิบายความสัมพันธ์ (Relationship) ระหว่างแนวคิด (Concept) ที่อยู่ภายใต้ขอบเขตเนื้อหาเรื่องใดเรื่องหนึ่ง (Domain) [7] ดังนั้นเพื่อให้แนวคิดเหล่านี้สามารถอธิบายความหมายได้ถูกต้องเข้าใจตรงกันจึงต้องจัดเก็บให้อยู่ในรูปแบบหรือข้อกำหนดที่เป็นทางการ (Formal specification) [8] คืออยู่ในรูปแบบที่คอมพิวเตอร์สามารถที่จะเข้าใจและประมวลผลอย่างมีประสิทธิภาพ การที่ใช้ข้อกำหนดที่เป็นทางการจะช่วยให้การตีความแนวคิดและความสัมพันธ์ชัดเจนยิ่งขึ้น อีกทั้งยังสนับสนุนการแบ่งปัน (Sharing) การนำกลับมาใช้ใหม่ได้ (Reusing) และสามารถรองรับการอนุมานความรู้ใหม่ภายใต้แนวคิดนี้ได้อีกด้วย [9] ออนโทโลยีประกอบไปด้วยส่วนต่าง ๆ ดังนี้ [10]

1) **แนวคิด (Concept)** คือ ขอบเขตของเนื้อหาเรื่องใดเรื่องหนึ่งที่สนใจ
2) **ความสัมพันธ์ (Relationship)** คือ รูปแบบของความสัมพันธ์ระหว่างแนวคิด ตัวอย่างประเภทของความสัมพันธ์สามารถมีได้ดังนี้
    - ความสัมพันธ์แบบลำดับชั้น (Subclass-of) คือ ความสัมพันธ์ที่มีคุณสมบัติการถ่ายทอด คุณสมบัติของแนวคิดแม่ไปยังแนวคิดลูก
    - ความสัมพันธ์แบบเป็นส่วนหนึ่ง (Part-of) คือ ความสัมพันธ์ที่หมายถึงการเป็นส่วนประกอบ

- ความสัมพันธ์เชิงความหมาย (Syn-of) คือ ความสัมพันธ์ที่แสดงถึงแนวคิดที่มีความเหมือนเชิงความหมายต่อกัน
- ความสัมพันธ์การเป็นตัวแทน (Instance-of) คือ ความสัมพันธ์ที่แสดงถึงการเป็นตัวแทน หรือสมาชิกของแนวคิด

3) **คุณลักษณะ (Property)** คือ คุณสมบัติที่ใช้ในการอธิบายแนวคิดนั้น

4) **สัจพจน์ (Axiom)** คือ ข้อกำหนดในการสร้างความสัมพันธ์ โดยที่โครงสร้างความสัมพันธ์นั้นจะต้องเป็นจริงเสมอ หรืออีกนัยหนึ่งคือ ข้อกำหนดหรือเงื่อนไข (Constraint) ที่ใช้ในการแปลงความสัมพันธ์ระหว่างแนวคิดกับคุณสมบัติ หรือแนวคิดกับแนวคิด เพื่อให้แปลงความหมายได้ถูกต้อง และยังสามารถนำมาใช้ในการอนุมานความรู้เพื่อสร้างความรู้ใหม่จากออนโทโลยีที่สร้างขึ้นได้อีกด้วย

5) **อินสแตนซ์ (Instances)** คือ ตัวอย่างข้อมูลที่ใช้อธิบายองค์ประกอบของออนโทโลยี

เมื่อนำส่วนประกอบต่าง ๆ ของออนโทโลยีมาประกอบกันก็จะสามารถพัฒนาออนโทโลยีเพื่อสร้างองค์ความรู้ในด้านต่าง ๆ ได้ โดยการพัฒนาออนโทโลยีโดยส่วนใหญ่มักจะมีวัตถุประสงค์ดังต่อไปนี้ [11]

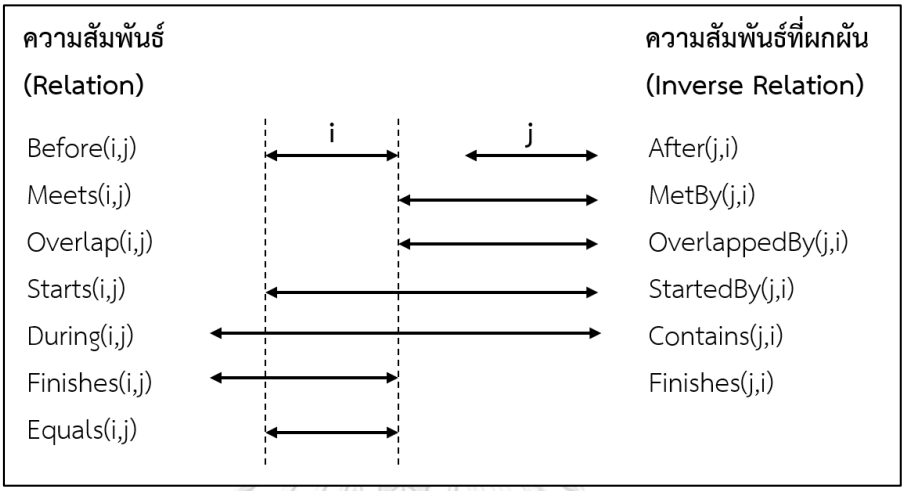
- เพื่อเข้าใจภาษาธรรมชาติ (Natural Language)
- เพื่อค้นคืนข้อมูล (Information Retrieval)
- เพื่อตรวจสอบทฤษฎี (Theoretical Investigation)
- เพื่อแบ่งปันความรู้และนำกลับมาใช้ใหม่ (Sharing and Reuse)
- เพื่อสร้างแบบจำลอง (Modeling)

นอกจากนี้ออนโทโลยียังมีภาษาที่สนับสนุนเครื่องมือที่ใช้ในการให้เหตุผล (Reasoning tool) [10] การให้เหตุผลนับเป็นสิ่งที่สำคัญมากเพราะนอกจากรับประกันเรื่องคุณภาพของการออกแบบออนโทโลยีแล้ว ยังใช้ในการสร้างองค์ความรู้ใหม่ๆ ให้เกิดขึ้นได้จากข้อมูลที่มีอยู่ ซึ่งคุณสมบัติเด่นที่นักวิจัยมักนำออนโทโลยีมาประยุกต์ใช้คือการตรวจสอบความต้องกัน (Consistency) ของแบบจำลองต่าง ๆ ในช่วงการออกแบบ ซึ่งในงานวิจัยนี้ใช้ข้อดีของออนโทโลยีในการตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็มเช่นเดียวกัน

2.1.4 ออนโทโลยีเวลา (Time Ontology)

ออนโทโลยีเวลา [12] ถือได้ว่ามีความสำคัญเป็นอย่างมากในการพัฒนาระบบเว็บเชิงความหมาย (Semantic Web) ซึ่งออนโทโลยีเวลานี้ได้พัฒนาต่อยอดมาจากทฤษฎีช่วงเวลา 13 ข้อ

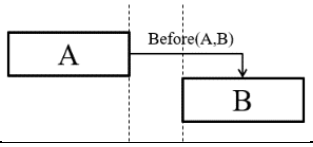
ของ Allen (Allen's interval based time theory) [13] ที่ได้อธิบายเกี่ยวกับความสัมพันธ์ด้านเวลาในลักษณะที่มีการพึ่งพากันระหว่างกิจกรรมสองกิจกรรม คือกิจกรรมที่นำหน้า (Predecessor) และกิจกรรมที่ตามหลัง (Successor) โดยแบ่งออกเป็นใน 13 รูปแบบพื้นฐาน ดังรูปที่ 2.3

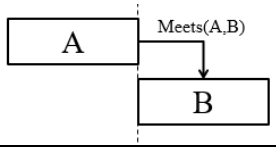
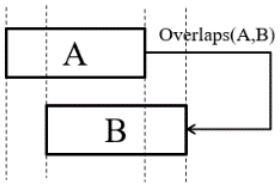
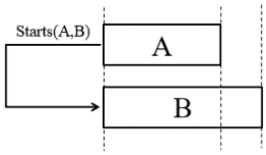


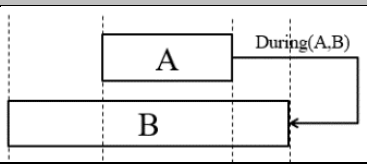
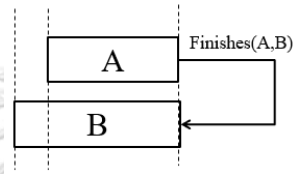
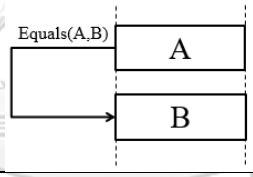
รูปที่ 2.3 ความสัมพันธ์ของช่วงเวลาตามทฤษฎีทั้ง 13 ข้อของ Allen

จากรูปที่ 2.3 จะพบว่ามี 6 ความสัมพันธ์ที่มีความสัมพันธ์ที่ผกผัน (Inverse Relation) เช่น กรณีของ Before(i,j) หมายความว่า ช่วงเวลาของกิจกรรม i มีความสัมพันธ์แบบก่อนหน้า (Before) กับกิจกรรม j แต่ในขณะเดียวกัน กิจกรรม j ก็มีความสัมพันธ์แบบตามหลัง (After) กับกิจกรรม i เป็นต้น ยกเว้นความสัมพันธ์แบบ Equals ที่ไม่มีความสัมพันธ์แบบผกผัน เพราะมีลักษณะเป็นความสัมพันธ์แบบสมมาตร (Symmetric) เมื่ออธิบายความสัมพันธ์จากรูปที่ 2.3 ให้อยู่ในช่วงเวลาในการทำกิจกรรม A และ B จะได้ดังตารางที่ 2.2

ตารางที่ 2.2 อธิบายความสัมพันธ์ของทฤษฎีช่วงเวลาทั้ง 13 ข้อ

ความสัมพันธ์ (Relation)	**คำนิยาม (Definition)**	**ความสัมพันธ์ที่ผกผัน (Inverse Relation)**
Before	ถ้ากิจกรรม A มีความสัมพันธ์แบบ Before กับกิจกรรม B แล้ว ดังนั้น เมื่อสิ้นสุดกิจกรรม A แล้ว กิจกรรม B จึงจะเริ่มต้นได้ 	After

ความสัมพันธ์ (Relation)	คำนิยาม (Definition)	ความสัมพันธ์ที่ผกผัน (Inverse Relation)
Meets	ถ้ากิจกรรม A มีความสัมพันธ์แบบ Meet กับกิจกรรม B แล้ว ดังนั้น เวลาที่สิ้นสุดกิจกรรม A แล้วจะเป็นเวลาเดียวกันกับที่กิจกรรม B เริ่มต้น 	MetBy
Overlaps	ถ้ากิจกรรม A มีความสัมพันธ์แบบ Overlaps กับกิจกรรม B แล้ว ดังนั้น เวลาที่เริ่มต้นของกิจกรรม A จะเกิดขึ้นก่อนเวลาที่เริ่มต้นกิจกรรม B และเวลาที่สิ้นสุดกิจกรรม A จะเกิดขึ้นหลังจากเวลาที่เริ่มต้นกิจกรรม B และเวลาที่กิจกรรม A สิ้นสุดจะเกิดก่อนเวลาที่กิจกรรม B จะสิ้นสุด 	OverlappedBy
Starts	ถ้ากิจกรรม A มีความสัมพันธ์แบบ Starts กับกิจกรรม B แล้ว ดังนั้น เวลาที่เวลาที่เริ่มต้นของกิจกรรม A และ B จะเป็นเวลาเดียวกัน และกิจกรรม A จะสิ้นสุดก่อนกิจกรรม B 	StartedBy
During	ถ้ากิจกรรม A มีความสัมพันธ์แบบ Durings กับกิจกรรม B แล้ว ดังนั้น เวลาที่เริ่มต้นกิจกรรม A จะเกิดหลังเวลาเริ่มต้นของกิจกรรม B และเวลาที่สิ้นสุดกิจกรรม A จะเกิดก่อนเวลาที่สิ้นสัดของกิจกรรม B	Contains

ความสัมพันธ์ (Relation)	คำนิยาม (Definition)	ความสัมพันธ์ที่ผกผัน (Inverse Relation)
		
Finishes	ถ้ากิจกรรม A มีความสัมพันธ์แบบ Finishes กับกิจกรรม B แล้ว ดังนั้น กิจกรรม A จะเริ่มต้นหลังกิจกรรม B และเวลาที่สิ้นสุดของกิจกรรม A และ B จะเป็นเวลาเดียวกัน 	FinishedBy
Equals	ถ้ากิจกรรม A มีความสัมพันธ์แบบ Equals กับกิจกรรม B แล้ว ดังนั้น เวลาที่เริ่มต้นและสิ้นสุดของทั้งกิจกรรม A และ B จะเป็นเวลาเดียวกัน 	-

โดยที่ออนโทโลยีเวลานี้สามารถรองรับภาษาที่ใช้ในการอธิบายแบบจำลองเชิงความหมายได้อย่างหลากหลาย อาทิ DAML, OWL, KSL, KIF และ Cyc เป็นต้น ดังนั้นในงานวิจัยนี้จึงได้ใช้ทฤษฎีของ Allen มาปรับใช้ในการสร้างกฎ เพื่อใช้ในการอนุมานความรู้ใหม่จากข้อมูลนำเข้าเพื่อทำการจำแนกและอธิบายความสัมพันธ์และลำดับของกิจกรรมบนแผนภาพข่ายงานแบบพีดีเอ็มให้อยู่ในรูปแบบภาษาอาวล์ ซึ่งจะช่วยให้คอมพิวเตอร์สามารถประมวลผลและตรวจสอบความต้องกันต่อไปได้

2.1.5 ภาษาอาวล์ (OWL)

ภาษาอาวล์ (OWL) [17] เป็นภาษาที่ถูกพัฒนาขึ้นโดยโดยหน่วยงานเวิลด์ไวด์เว็บคอนซอร์เทียม (W3C) ในปี ค.ศ. 2004 มีวัตถุประสงค์เพื่อเพิ่มขีดความสามารถของภาษาอาร์ดีเอฟ (Resource Description Framework : RDF) และโครงร่างอาร์ดีเอฟ (Resource Description Framework Schema : RDFS) [18] ซึ่งมีพื้นฐานมาจากภาษาเอ็กซ์เอ็มแอล (Extensible Markup

Language : XML) ให้สามารถอธิบายแนวคิดเชิงความหมายได้อย่างครบถ้วนสมบูรณ์ยิ่งขึ้น เนื่องจากภาษาอาวล์นั้น สามารถที่จะรองรับการอธิบายเชิงความหมายของออนโทโลยี ที่มีการกำหนดโครงสร้างของข้อมูลเป็นคลาส ลำดับชั้นของคลาส และความสัมพันธ์ระหว่างคลาส ทั้งนี้ยังสามารถรองรับการอธิบายข้อมูลเชิงตรรกะ (Logic) ชนิดข้อมูล (Data type) และตัวบ่งปริมาณได้ (Quantifier) จึงทำให้การอธิบายข้อมูลเหล่านี้มีความหมายมากยิ่งขึ้น อีกทั้งยังสามารถใช้ในการประมวลผลคอมพิวเตอร์ได้อีกด้วย [19]

ปัจจุบันมีการพัฒนาภาษาอาวล์ 2 เวอร์ชันคือ อาวล์ เวอร์ชัน 1 (OWL 1) และ อาวล์ เวอร์ชัน 2 (OWL 2) สำหรับภาษาอาวล์ เวอร์ชัน 1 นั้นสามารถแบ่งได้เป็น 3 ประเภท [17] ขึ้นอยู่กับลักษณะการใช้งาน ดังนี้

1) อาวล์ไลท์ (OWL Lite) เหมาะกับการใช้งานเพื่อการจัดลำดับชั้น (Hierarchy) และการแบ่งประเภท (Classification) ของแนวคิดที่มีการกำหนดเกณฑ์ในการจัดลำดับหรือแบ่งกลุ่มอย่างง่าย ซึ่งจะช่วยให้สามารถค้นหาคำตอบได้อย่างรวดเร็ว เนื่องจากมีการจัดเส้นทางการค้นหาในลักษณะลำดับตามพจนานุกรม (Thesauri) และการจัดหมวดหมู่ (Taxonomy)

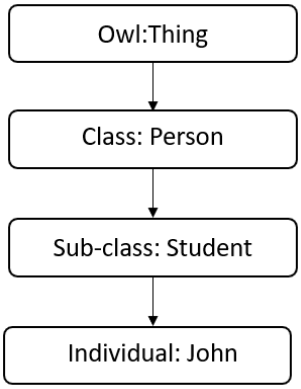
2) อาวล์ดี-แอล (OWL DL) เหมาะกับการใช้งานในลักษณะที่ต้องใช้วิธีการอธิบายเชิงความหมายที่ซับซ้อนมากขึ้น โดยที่ยังคงความครบถ้วนสมบูรณ์ของการคำนวณ มีการตรวจสอบทุกเงื่อนไข และสามารถประมวลผลได้ในระยะเวลาที่จำกัด ถึงแม้ว่า ภาษาอาวล์ดีแอล จะมีโครงสร้างตามภาษา ภาษาอาวล์ ทุกประการ หรือสามารถรองรับคำสั่งงานทุกอย่างของภาษาภาษาอาวล์ ได้ แต่ก็มีข้อจำกัดในการใช้งาน เช่น คลาสหนึ่งๆ อาจจะเป็นคลาสย่อย (Sub class) ของหลายคลาสได้ แต่คลาสนั้นจะไม่สามารถเป็นอินสแตนซ์ (Instance) ของคลาสอื่น ๆ ได้ เนื่องจากภาษาอาวล์ดีแอลได้รับการถ่ายทอดคุณสมบัติมาจากภาษาการอธิบายเชิงตรรกะ (Description Logic) [20] ที่ใช้ในการอธิบายแนวคิดภายใต้ขอบเขตที่สนใจ และยังสามารถใช้ในการให้เหตุผลเชิงตรรกะสำหรับออนโทโลยีได้

3) อาวล์ฟูล (OWL Full) เหมาะกับการใช้งานที่มีการอธิบายเชิงความหมายขั้นสูง มีความซับซ้อนมากเป็นพิเศษ โดยสามารถที่จะกำหนดไวยากรณ์ภาษา (Syntax) เพิ่มเติมขึ้นมาเองจาก อาร์ดีเอฟ ได้ และยังสามารถกำหนดให้คลาสสามารถเป็นอินแสตนซ์ (instance) ได้อีกด้วย

ต่อมาได้มีการพัฒนาเวอร์ชันเป็นภาษาอาวล์ เวอร์ชัน 2 [21] ในปี ค.ศ. 2009 เนื่องจากภาษาอาวล์ เวอร์ชัน 1 นั้นยังมีข้อจำกัดบางประการในการสร้างแบบจำลองที่มีความซับซ้อน โดยเฉพาะการสร้างแบบจำลองที่ต้องอาศัยการให้เหตุผลด้วยเครื่องมือต่าง ๆ เช่น FacT++, Pallet, RACER และ HermiT ทำให้มีความจำเป็นที่จะต้องพัฒนาภาษาให้รองรับกับเครื่องมือเหล่านี้

โดยเฉพาะภาษาอาวล์ เวอร์ชัน 2 ดีแอล (OWL 2 DL) ที่สามารถรองรับเครื่องมือการอนุมานเหตุผลเหล่านี้ได้เป็นอย่างดี อีกทั้งยังสนับสนุนส่วนเชื่อมต่อภาษาอาวล์ (OWL API) ทำให้ผู้พัฒนาสามารถเลือกใช้แพลตฟอร์มที่มีประสิทธิภาพในการเชื่อมต่อด้วยภาษาจาวา (Java) ได้อย่างสะดวกและหลากหลายมากยิ่งขึ้น นอกจากนี้ยังมีการจัดกลุ่มย่อยของภาษาตามโปรไฟล์ (Profile) ดังนี้ อาวล์ 2 อี-แอล (OWL 2 EL) อาวล์ 2 คิว-แอล (OWL 2 QL) และอาล์ว 2 อาร์-แอล (OWL 2 RL) [22]

ภาษาอาวล์เป็นชุดของข้อเท็จจริงหรือสัจพจน์ ที่มีการแสดงตรรกะ (Logic Assertion) ไว้อย่างชัดเจน ซึ่งประกอบด้วย 3 ส่วน ได้แก่ คลาส (Class) อินดิวิดวล (Individual) ภายใต้ขอบเขตที่กำหนด และคุณลักษณะ (Property) จากนั้นใช้เครื่องมืออนุมานเหตุผล (Reasoner) ในการอนุมานข้อเท็จจริงอื่น ๆ ภายใต้ขอบเขตที่สนใจที่ถูกอธิบายในออนโทโลยี



รูปที่ 2.4 แสดงการแทนความรู้ด้วยออนโทโลยี โดยใช้คลาส คลาสย่อย

การประกาศตัวแปรต่าง ๆ ในภาษาอาวล์นั้นสามารถเขียนให้อยู่ในรูปของไวยากรณ์ (Syntax) ได้หลากหลายรูปแบบ เช่น XML/RDF, Functional Syntax, Manchester และ Turtle โดยที่ XML/RDF จะเป็นรูปแบบที่นิยมใช้งานกันมากที่สุด

1. คลาส (Class)

คลาส คือ กลุ่มของชุดข้อมูลที่ใช้ในการอธิบายสิ่งต่าง ๆ ภายใต้ขอบเขตที่สนใจ เช่น กำหนดคลาส ชื่อว่า “Person” ที่มีคลาสย่อย (Sub Class) เป็น “Student” โดยที่คลาสย่อยนั้นมี อินดิวิดวล ชื่อ “John” ตัวอย่างการประกาศคลาส Person และคลาสย่อย Student ด้วยไวยากรณ์ XML/RDF

```
<owl:Class rdf:ID="Person"/>
<owl:Class rdf:ID="Student">
  <rdfs:subClassOf rdf:resource="Person" />
```

```
</owl:Class>
```

ซึ่งคลาสในภาษาอาล์วนั้นจะมีคลาสพิเศษ (Special Classes) อยู่ 2 ประเภทได้แก่

- **Top (a):** เป็นเซตของอินดิวิดวลทั้งหมด ใช้แทนด้วยสัญลักษณ์ owl:Thing ซึ่งจะเป็นคลาสที่อยู่บนสุดของทุกคลาสภายใต้ขอบเขตที่สนใจ
- **Bottom (⊥):** เป็นเซตว่าง (Empty Set)

นอกจากนี้คลาสต่าง ๆ สามารถที่จะนำมารวมกันด้วยการใช้ตัวดำเนินการทางเซต (Set Operator) ดังต่อไปนี้

- เซ็ตย่อย (Subset)
- เซ็ตที่ไม่ได้เชื่อมโยงกัน (Disjoint sets)
- เซ็ตที่รวมกัน (Union sets)
- เซ็ตที่ทับซ้อนกัน (Intersection sets)
- เซ็ตที่ตรงข้ามกัน (Complement sets)

2. อินสแตนซ์หรืออินดิวิดวล (Instance/Individual)

อินสแตนซ์หรืออินดิวิดวล คือ การระบุข้อความที่เป็นสัจพจน์ (Axiom) หรือข้อเท็จจริง (Fact) โดยแสตนซ์จะต้องอยู่ภายใต้คลาสหรือคลาสย่อย และจะต้องมีความสัมพันธ์กับอินแสตนซ์อื่น ๆ หรือข้อมูลอื่น ๆ ผ่านทางคุณลักษณะ (Property) ตัวอย่างการประกาศอินแสตนซ์ ของบุคคลที่ชื่อว่า John ซึ่งอยู่ภายใต้คลาสย่อย Student

```
<Student rdf:ID="John" />
```


3. คุณลักษณะ (Property)

คุณลักษณะ (Property) คือ ความสัมพันธ์ระหว่างคู่อินดิวิดวล (Object Property) และข้อมูล (Data Property) ภายใต้ขอบเขตที่สนใจ ซึ่งในภาษาอาล์วแบ่งออกเป็น 2 ประเภท

- **คุณลักษณะของออบเจ็ค (Object Property)** เป็นการแสดงความสัมพันธ์ระหว่างอินดิวิดวลแต่ละคู่ โดยอินดิวิดวลหลักที่เป็นผู้กำหนดหรือกระทำคุณลักษณะจะเรียกว่าโดเมน (Domain) ส่วนอินดิวิดวลที่ได้รับการถ่ายทอดคุณลักษณะจะเรียกว่าเรนจ์ (Range) เช่น John กับ Steve โดยที่ John มีการกำหนดคุณลักษณะเป็น “isStudentOf” Steve โดยที่ทั้ง 2 คนนั้นอยู่ภายใต้คลาส Person แต่ John อยู่ภายใต้คลาสย่อย “Student” ในขณะที่ Steve อยู่ภายใต้คลาสย่อย “Teacher” ดังนั้นหาก

เขียนในลักษณะของความสัมพันธ์ที่ว่า John is a student of Steve จะมีความหมายว่าคลาส Student เป็นโดยเมน ส่วนคลาส Teacher เป็นเรนจ์ ซึ่งสามารถที่จะประกาศคุณลักษณะได้ดังนี้

```
<owl:ObjectProperty rdf:ID="isStudentOf">
  <rdfs:domain rdf:resource="#Student"/>
  <rdfs:range rdf:resource="#Teacher"/>
</owl:ObjectProperty>
```

- **คุณลักษณะของข้อมูล (Data Property)** เป็นการแสดงความสัมพันธ์ระหว่างอินดิวิดวลกับประเภทของข้อมูล (Data Type) เช่น อินดิวิดวล John มีคุณลักษณะของข้อมูลเป็น “hasAge” ซึ่งมีประเภทข้อมูลเป็นจำนวนเต็มบวก (Positive Integer) เป็นต้น

```
<owl:DatatypeProperty rdf:ID="hasAge">
  <rdfs:domain rdf:resource="#Student" />
  <rdfs:range  rdf:resource="&xsd;positiveInteger"/>
</owl:DatatypeProperty>
```

ลักษณะพิเศษของคุณลักษณะ (Property Characteristics) มีดังต่อไปนี้

- คุณสมบัติการถ่ายทอด (Transitive Property) : เป็นประเภทของคุณสมบัติที่ได้รับการถ่ายทอดมาจากคลาสใดคลานหนึ่งภายใต้ขอบเขตที่สนใจ เช่น คุณสมบัติของการเป็นบรรพบุรุษ (hasAncestor) เป็นต้น
- คุณสมบัติสมมาตร (Symmetric Property) : เป็นประเภทของคุณสมบัติที่สามารถสลับที่กันได้ระหว่างโดนเมนกับเรนจ์ได้ โดยที่ความหมายไม่เปลี่ยนแปลง เช่น คุณลักษณะของการเป็นพี่น้อง (isSiblingOf) เป็นต้น
- คุณสมบัติตามฟังก์ชัน (Functional Property) : เป็นคุณลักษณะเฉพาะตัวของแต่ละคลาส เช่น คลาสครอบครัว (Family) จะต้องประกอบไปด้วยคุณลักษณะที่แสดงการเป็นภรรยา (isWifeOf) เป็นต้น ซึ่งคุณสมบัติตามฟังก์ชันมักจะมีคุณสมบัติที่ตรงข้ามกัน (Inverse Of) ที่เรียกว่า คุณสมบัติอินเวิร์ส (Inverse Functional Property) เช่น คุณลักษณะที่ตรงกันข้ามกับคุณลักษณะของการเป็นภรรยาคือ คุณลักษณะของการเป็นสามี (isHusbandOf) เป็นต้น

การที่จะให้คำนิยามว่าอินดิวิดวลใดควรจะอยู่ภายใต้คลาสใดนั้น จำเป็นจะต้องมีการกำหนดข้อจำกัดของคุณลักษณะ (Property Restrictions) ของแต่ละคลาสให้ชัดเจน โดยที่ข้อจำกัดของคุณลักษณะของคลาสจะเป็นตัวกำหนดอินดิวิดวลภายใต้คลาสนั้น ในการระบุข้อจำกัดของคุณลักษณะดังกล่าวสามารถใช้ไวยากรณ์ภาษา XML ในภาษาอาวล์ มาใช้ในการแบ่งประเภทข้อจำจัดออกเป็น 2 ประเภท ได้แก่ ข้อจำกัดของข้อมูล (Data Restriction) คือข้อจำกัดของความสัมพันธ์ที่มีเรนจ์เป็นข้อความหรือตัวอักษร และ ข้อจำกัดของอ็อบเจค (Object Restriction) คือข้อจำกัดของความสัมพันธ์ที่มีเรนจ์เป็นอินดิวิดวล ตัวอย่างของข้อจำกัดทั้ง 2 ประเภทที่นิยมใช้กันก็คือ

- ข้อจำกัดตัวบ่งปริมาณ (Quantifier Restriction): เป็นข้อจำกัดด้านปริมาณที่ใช้ตัวบ่งชี้ปริมาณในการกำหนดข้อจำกัดของคุณลักษณะ เช่น At least, All of, Some of เป็นต้น
- ข้อจำกัดคาร์ดินัลลิตี้ (Cardinality Restriction): เป็นข้อจำกัดด้านปริมาณที่ระบุเป็นตัวเลข เช่น 1 มือ จะต้องประกอบไปด้วย 5 นิ้วเท่านั้น

ซึ่งในงานวิจัยนี้ได้เลือกใช้ ภาษาอาวล์เวอร์ชัน 2 ดีแอล (OWL 2 DL) ในการอธิบายโครงสร้างของออนโทโลยี เพราะมีความสามารถที่จะรองรับการให้เหตุผลด้วยการอนุมานจากข้อเท็จจริงและกฎต่าง ๆ ที่กำหนดขึ้นได้และเป็นภาษาเวอร์ชันใหม่ล่าสุดที่ W3C แนะนำให้ใช้งาน

2.1.6 ภาษาเอสดับเบิลยูอาร์แอล (SWRL)

ภาษาเอสดับเบิลยูอาร์แอล (Semantic Web Rule Language : SWRL) [14] เป็นภาษาที่ใช้สำหรับเขียนกฎ ถูกพัฒนามาจากภาษาที่ใช้ในการอธิบายกฎ (Description Logic) เช่นเดียวกับภาษาอาวล์ โดยเป็นส่วนต่อขยายมาจากภาษาอาวล์ ช่วยให้ออนโทโลยีที่พัฒนาขึ้นมานั้นสามารถที่จะอนุมาน (Inference) ผ่านกฎที่ถูกออกแบบไว้ได้ ซึ่งจะทำให้ออนโทโลยีสามารถเรียนรู้ความรู้ใหม่ๆ ผ่านกฎดังกล่าวได้ ซึ่งจำเป็นต้องใช้ติดตั้งเครื่องมือที่ใช้ในการอนุมาน (Inference Engine) เช่น JESS [15], Drools [16] เป็นต้น

โครงสร้างภาษาเอสดับเบิลยูอาร์แอลนั้นต้องเขียนให้อยู่ในรูปคู่ของเหตุ (Antecedent Part) และผล (Consequent Part) ซึ่งเหตุหรือเรียกอีกอย่างหนึ่งว่า ส่วนตัว (Body) นั้น จะใช้ในการอธิบายประโยคหรือข้อความที่เป็นข้อเท็จจริง สำหรับผลหรือเรียกอีกอย่างหนึ่งว่า ส่วนหัว (Head) นั้นจะใช้ในการอธิบายผลหรือข้อสรุปที่ตามมา ซึ่งทั้งสองส่วนจะต้องอยู่ในรูปแบบของเงื่อนไขที่มีตัวเชื่อม (Connective) ตามตรรกะเพรดิเคต (Predicate Logic) [17] “และ” (AND) โดยมีการกำหนดสัญลักษณ์แทน “และ” ด้วยเครื่องหมาย “^” ซึ่งแต่ละประโยคหรือข้อความที่เป็นข้อเท็จจริง

จากองค์ความรู้ที่สร้างขึ้นมานั้นเรียกว่าสูตรอะตอม (Atom) ซึ่งเป็นสูตรที่เล็กที่สุดที่ถูกต้องตามหลักไวยากรณ์ เช่น การแสดงความสัมพันธ์ของบุคคลต่างๆ ภายใต้ขอบเขตของครอบครัว โดยกำหนดว่า จอห์นเป็นพ่อของลิซ่า สามารถแทนความรู้ดังกล่าวได้ด้วยหลักไวยากรณ์ตามตรรกะเพรติเคตได้ดังนี้

Father (John, Lisa)

โดยที่ Father ใช้แทนสัญลักษณ์เพรดิเขต (Predicate Symbol) ส่วน John และ Lisa เป็นอินดิวิดวล (Individual) ภายใต้ขอบเขตที่กำหนด จากนั้นนำแต่ละอะตอมมาเชื่อมกันด้วยเครื่องหมาย “^” และเชื่อมส่วนหน้าและส่วนหลังด้วยตัวเชื่อมถ้า-แล้ว (IF-THEN) ซึ่งมีการกำหนดสัญลักษณ์เป็น “$\rightarrow$”

atom ^ atom .... $\rightarrow$ atom ^ atom

ในภาษาเอสดับเบิลยูอาร์แอลนั้นไม่สามารถรองรับการเชื่อมประโยคที่เป็นเท็จ (Disjuction) ได้ ดังนั้นแต่ละประโยคที่เชื่อมกันจะต้องเป็นประโยคบอกเล่า (Positive Conjunction) เท่านั้น ซึ่งในภาษาเอสดับเบิลยูอาร์แอลจะประกอบไปด้วยพริดิเคตใน 5 ลักษณะด้วยกัน

- คลาสในภาษาอาวล์ (OWL Class)
- คุณลักษณะในภาษาอาวล์ (OWL Property)
- ประเภทของข้อมูล (Data Type)
- ช่วงของข้อมูล (Data Range)
- เพรดิเขตที่กำหนดขึ้นมาเอง (Built-In)

ในแต่ละอะตอมจะต้องถูกกำหนดให้อยู่ใน 7 รูปแบบนี้เท่านั้น

- **อะตอมของคลาส (Class Atoms)** จะแสดงอยู่ในรูปแบบของประโยค (expression) ที่มีชื่อคลาสนำหน้าและตามด้วยวงเล็บที่แสดงถึงอาร์กิวเมนต์ (Argument) ซึ่งจะมีได้เพียงค่าเดียวเท่านั้น โดยที่อาร์กิวเมนต์สามารถเป็นได้ทั้งตัวแปรที่ไม่ระบุค่า ซึ่งจะมีเครื่องหมายคำถาม (?) นำหน้า และอินดิวิดวลที่อยู่ภายใต้คลาสที่กำหนด เช่น การใช้ตัวแปร ?p เพื่อแทนค่าอินดิวิดวลทุกตัวที่อยู่ภายใต้คลาส Person หรือการใช้อินดิวิดวล John เพื่อระบุค่าเฉพาะเจาะจงในคลาส Man ซึ่งประโยคดังกล่าวสามารถเขียนให้อยู่ในรูปแบบของอะตอมของคลาสในภาษาเอสดับเบิลยูอาร์แอลได้ดังนี้

Person(?p)
Man(John)

เมื่อนำประโยคข้างต้นมาเขียนให้อยู่ในรูปของภาษากฎอย่างง่าย เพื่อใช้ในการอนุมานความรู้ว่าทุกอินดิวิดวลที่อยู่ภายใต้คลาส Man จะอยู่ภายใต้คลาส Person ด้วย

Man(?p) ⟶ Person(?p)

- **อะตอมคุณลักษณะของอินดิวิดวล (Individual Property atoms)** จะแสดงอยู่ในรูปแบบของประโยคที่มีคุณลักษณะของออบเจ็ค (Object Property) นำหน้าและตามด้วยวงเล็บที่แสดงถึงอาร์กิวเมนต์ 2 ค่า เพื่อแสดงความสัมพันธ์ระหว่างคู่ของตัวแปรที่ใช้แทนค่าอินดิวิดวล หรืออินดิวิดวลที่มีการระบุค่า

hasBrother(?x, ?y)
hasSibling(Fred, ?y)

เมื่อนำประโยคข้างต้นมาเขียนให้อยู่ในรูปของภาษากฎเพื่อแสดงว่า.... จะเขียนได้ดังนี้

Person(?p) ^ hasSibling(?p, ?s) ^ Man(?s) -> hasBrother(?p,?s)

- **อะตอมคุณลักษณะของข้อมูล (Data Valued Property atoms)** จะแสดงอยู่ในรูปแบบของประโยคที่มีคุณลักษณะของข้อมูล (Data Property) นำหน้าและตามด้วยวงเล็บที่แสดงถึงอาร์กิวเมนต์ 2 ค่า เพื่อแสดงความสัมพันธ์ระหว่างคู่ของตัวแปรที่ค่าแรกของอาร์กิวเมนต์ใช้แทนค่าอินดิวิดวลหรืออินดิวิดวลที่มีการระบุค่า ส่วนค่าที่สองเป็นตัวแปรที่แทนด้วยค่าของข้อมูลหรือ ข้อมูลที่มีการระบุค่า

hasAge(?x, ?age)
hasHeight(Fred, ?h)
hasAge(?x, 232)
hasName(?x, "Fred")

เมื่อนำอะตอมแต่ละลักษณะมาประกอบกันในรูปแบบของภาษากฎ

- **อะตอมที่แสดงความแตกต่างของอินดิวิดวล (Different Individuals atoms)** จะแสดงอยู่ในรูปแบบของประโยคที่มีคุณลักษณะของข้อมูล (Data Property) ชื่อว่า ‘differentFrom’ แล้วตามด้วยวงเล็บที่แสดงถึงอาร์กิวเมนต์ 2 ค่า ซึ่งเป็นอินดิวิดวลทั้ง 2 ค่า

differentFrom(?x, ?y)

differentFrom(Fred, Joe)

- **อะตอมที่แสดงความเหมือนกันของอินดิวิดวล (Same Individual atoms)** จะแสดงอยู่ในรูปแบบของประโยคที่มีคุณลักษณะของข้อมูล (Data Property) ชื่อว่า ‘sameAs’ แล้วตามด้วยวงเล็บที่แสดงถึงอาร์กิวเมนต์ 2 ค่า ซึ่งเป็นอินดิวิดวลทั้ง 2 ค่า

sameAs(?x, ?y)
sameAs(Fred, Joe)

- **อะตอมที่แสดงช่วงค่าของข้อมูล (Data Range atoms)** จะแสดงอยู่ในรูปแบบของประโยคที่มีประเภทของข้อมูล (Data Type) ที่เป็นตัวเลขหรือตัวอักษรต่าง ๆ แล้วตามด้วยวงเล็บที่แสดงถึงอาร์กิวเมนต์ 1 ค่า ซึ่งเป็นอินดิวิดวล

xsd:int(?x)
[3, 4, 5](?x)

- **อะตอมบิลท์อิน (Built-in atoms)** จะเป็นการผสมกันระหว่างอะตอมในรูปแบบต่างๆ โดยแต่ละอะตอมจะเชื่อมกันด้วยเครื่องหมาย “^” ซึ่งเป็นรูปแบบที่นิยมใช้งานมากที่สุด เพราะการที่มีเงื่อนไขของหลาย ๆ อะตอมมาประกอบกันจะช่วยให้กฎที่สร้างขึ้นนั้นมีความถูกต้องแม่นยำมากขึ้น

Person(?p) ^ hasAge(?p, ?age) ^ swrlb:greaterThan(?age, 17) → Adult(?p)

จากกฎของอะตอมประเทบิลท์อินข้างต้นสามารถซึ่งประกอบไปด้วยเหตุ 3 อะตอม คือ Person(?p), hasAge(?p, ?age), swrlb:greaterThan(?age, 17) และส่วนผล 1 อะตอม คือ Adult(?p) แต่ละหน่วยมีความหมายดังต่อไปนี้

Person(?p)	หมายถึง p เป็นสมาชิกของคลาสบุคคล (Person)
hasAge(?p, ?age)	หมายถึง p มีอายุที่มีค่าเป็น age
swrlb:greaterThan(?age, 17)	หมายถึง age มีค่ามากกว่า 17
Adult(?p)	หมายถึง p เป็นสมาชิกของคลาสผู้ใหญ่ (Adult)
เครื่องหมาย ^	หมายถึง และ

เครื่องหมาย $\rightarrow$ หมายถึง ดังนั้น

จากกฎด้านบนสามารถอธิบายได้ว่า หากบุคคลในตัวแปร p เป็นสมาชิกของคลาสบุคคล (Person) และบุคคลในตัวแปร p มีอายุเป็นค่า age และค่า age มีค่ามากกว่า 17 ดังนั้นจึงสามารถอนุมานได้ว่าบุคคลในตัวแปร p นั้น เป็นสมาชิกของคลาสผู้ใหญ่ (Adult) ด้วย ซึ่งจากกฎดังกล่าวทำให้สามารถอนุมานความรู้ใหม่ ๆ ภายใต้เงื่อนไขที่กำหนดได้ไม่มีข้อจำกัด

ดังนั้นงานวิจัยนี้จึงออกแบบกฎด้วยภาษาเอสดับเบิลยูอาร์แอล เพื่ออนุมานข้อเท็จจริงต่าง ๆ ที่เกี่ยวข้องกับแผนภาพข่ายงานแบบพีดีเอ็ม อีกทั้งยังใช้ในการตรวจสอบความต้องกันของแผนภาพดังกล่าวอีกด้วย

2.1.7 ภาษาเอสคิวดับเบิลยูอาร์แอล (SQWRL)

ภาษาเอสคิวดับเบิลยูอาร์แอล (Semantic Query-Enhanced Web Rule Language : SQWRL) [18] หรือเรียกว่า สควอรอล (Squirrel) เป็นภาษาที่ใช้สำหรับการสืบค้นข้อมูลจากออนโทโลยีที่สร้างขึ้นด้วยภาษาอาวล์ โดยใช้หลักการและไวยากรณ์พื้นฐานของภาษาเอสดับเบิลยูอาร์แอล (SWRL) และภาษาเอสคิวแอล (SQL) [19] เข้าไว้ด้วยกัน ซึ่งภาษา SQWRL มีการเขียนรูปแบบการสืบค้นโดยใช้หลักของภาษา SWRL ที่แบ่งออกเป็นส่วนเหตุ (Antecedent) และส่วนผล (Consequent) โดยที่ส่วนเหตุจะใช้ในการระบุเงื่อนไขที่ต้องการสืบค้น และในส่วนผลจะใช้ในการระบุค่าที่ต้องการนำมาแสดงผล ตัวอย่างตัวกระทำการ ที่นิยมใช้มีดังต่อไปนี้

sqwrl:select	หมายถึง	การเลือก
sqwrl:selectDistinct	หมายถึง	การเลือกแบบไม่ซ้ำ
sqwrl:count	หมายถึง	การนับ
sqwrl:countDistinct	หมายถึง	การนับแบบไม่ซ้ำ
sqwrl:min	หมายถึง	การสืบค้นค่าที่น้อยที่สุด
sqwrl:max	หมายถึง	การสืบค้นค่าที่มากที่สุด
sqwrl:sum	หมายถึง	การรวมค่า
sqwrl:avg	หมายถึง	การหาค่าเฉลี่ย
sqwrl:orderBy	หมายถึง	การเรียงลำดับ

ตัวอย่างการสืบค้นข้อมูลผู้ที่มีอายุมากกว่า 17 ปี สามารถจะเขียนแสดงด้วยไวยากรณ์ภาษา SQWRLได้ดังนี้

Person(?p) ^ hasAge(?p, ?age) ^ swrlb:greaterThan(?age, 17) → sqwrl:select(?p, ?age)

จากโครงสร้างด้านบนสามารถอธิบายได้ดังนี้

Person(?p)	หมายถึง	p เป็นสมาชิกของคลาสบุคคล
hasAge(?p, ?age)	หมายถึง	p มีอายุที่มีค่าเป็น age
swrlb:greaterThan(?age, 17)	หมายถึง	age มีค่ามากกว่า 17
sqwrl:select(?p, ?age)	หมายถึง	ให้เลือก p ซึ่งเป็นบุคคล และ age ซึ่งเป็นอายุมาแสดงตามเงื่อนไขที่กำหนด

ซึ่งในงานวิจัยนี้จะใช้ภาษา SQWRL ในการสืบค้นรูปแบบที่ต้องกันและไม่ต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม และทำการส่งคืนค่าการสืบค้นกลับมาให้ผู้ใช้งาน

2.1.8 โปรเทเจ (Protégé)

โปรเทเจ (Protégé) [20] เป็นเครื่องมือสนับสนุนการพัฒนาออนโทโลยีที่ถูกพัฒนาโดยมหาวิทยาลัยสแตนฟอร์ด (Stanford University School of Medicine) โดยมีความสามารถในการสร้างแบบจำลององค์ความรู้ด้วยวิธีออนโทโลยี ที่ประกอบไปด้วย การสร้างคลาส การสร้างความสัมพันธ์ของคลาส การสร้างกฎ (SWRL) และการสร้างภาษาที่ใช้ในการสืบค้น (SQWRL) เป็นต้น ซึ่งสามารถที่จะส่งออกโครงสร้างของแบบจำลองเหล่านี้ได้ในรูปแบบที่หลากหลาย ไม่ว่าจะเป็นรูปแบบของภาษาเอ็กซ์เอ็มแอล (XML) ภาษาอาร์ดีเอฟ (RDF) หรือภาษาอาวล์ (OWL) อีกทั้งยังสามารถติดตั้งโปรแกรมเสริมที่เพิ่มความสามารถให้กับโปรแกรมหลัก (Plug-in) นอกจากนี้ยังมีส่วนต่อเชื่อมต่อโปรแกรมประยุกต์ (Application Programming Interface : API) ที่มีพื้นฐานเป็นภาษาจาวา (Java) ในการสร้างเครื่องมือและโปรแกรมประยุกต์จากองค์ความรู้ที่สร้างโดยโปรเทเจ

โดยงานวิจัยชิ้นนี้จะใช้โปรเทเจ เป็นเครื่องมือหลักในการพัฒนาแบบจำลองออนโทโลยีของแผนภาพข่ายงานแบบพีดีเอ็ม รวมไปถึงการสร้างกฎที่ใช้ในการอนุมานความรู้ใหม่ และกฎที่ใช้ในการตรวจสอบความต้องกันของแผนภาพดังกล่าว จากนั้นจึงส่งออกแบบโครงสร้างจำลองในรูปแบบของภาษาอาวล์ และกฎในรูปแบบของภาษาเอสดับเบิลยูอาร์แอล นอกจากนี้โปรเทเจยังรองรับการนำเข้าข้อมูลด้วยแฟ้มข้อมูล Excel ซึ่งเป็นส่วนเสริมที่ถูกติดตั้งมากับโปรเทเจ เวอร์ชัน 5.2 ซึ่งในงานวิจัยนี้ก็จะใช้เครื่องมือดังกล่าวในการนำเข้าข้อมูลแผนภาพแบบพีดีเอ็มจากแฟ้มข้อมูล Excel

2.2 งานวิจัยที่เกี่ยวข้อง

ในส่วนนี้จะเป็นการศึกษางานวิจัยที่เกี่ยวข้องกับการนำแผนภาพที่เกี่ยวข้องกับการบริหารจัดการโครงการมาแสดงให้อยู่ในรูปแบบเชิงความหมาย จากนั้นจึงทำการศึกษางานวิจัยที่เกี่ยวข้องกับการนำแผนภาพแสดงการทำงานของกิจกรรมในรูปแบบต่าง ๆ มาทำการแปลงเป็นออนโทโลยีเพื่อตรวจสอบความต้องกัน

2.2.1 งานวิจัยที่เกี่ยวข้องกับการนำแบบจำลองเกี่ยวข้องกับการบริหารจัดการงานโครงการมาแสดงให้อยู่ในรูปแบบเชิงความหมาย

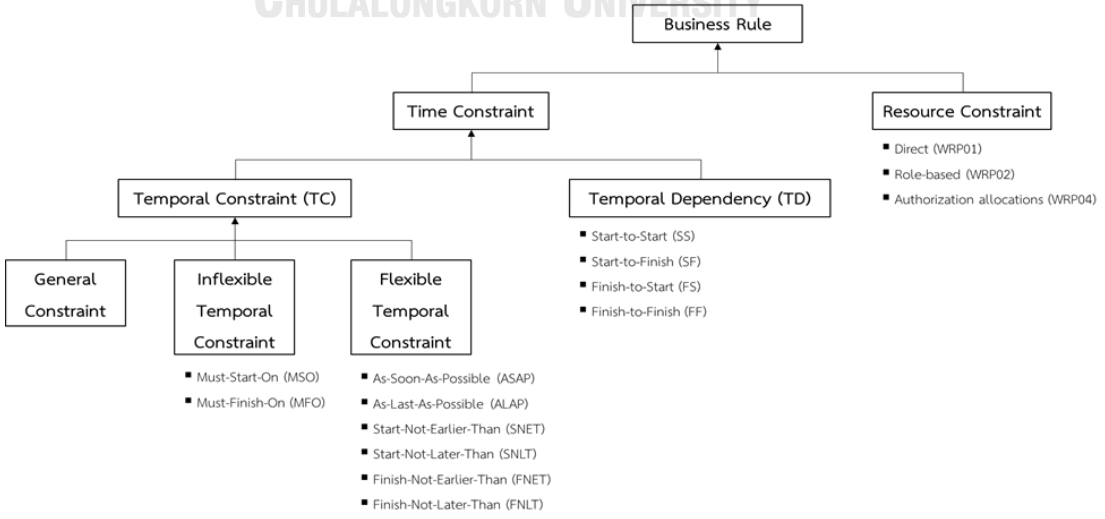
ในปี 2005 มีงานวิจัยที่ชื่อว่า Contact Workflow Model Patterns Using BPMN [21] โดย Kabilan, V. ได้นำเสนอแนวคิดในการเชื่อมโยงข้อมูลสัญญาที่ประกอบไปด้วยเงื่อนไขทางสัญญาและข้อจำกัดด้านต่าง ๆ จากเดิมที่ใช้แบบจำลองการดำเนินงานสัญญา (Contract Workflow Model : CWM) ในการติดตามงาน แต่แบบจำลองดังกล่าวมีการพัฒนาในรูปแบบที่เน้นเข้าใจง่ายโดยมนุษย์ แต่ยากต่อการนำไปประมวลผลโดยคอมพิวเตอร์ จากข้อจำกัดดังกล่าวงานวิจัยนี้นึงได้เลือกใช้แบบจำลองกระบวนการทางธุรกิจ (Business Process Models) มาช่วยในการนำเสนอการติดตามข้อมูลสัญญาให้สามารถอยู่ในรูปแบบที่เป็นทางการมากขึ้น (Formal Representation Notations) โดยจัดทำเป็นแบบจำลองที่มีชื่อว่า Multi-Tier Contract Ontology (MTCO) ซึ่งแนวคิดหลักของงานวิจัยนี้คือต้องการที่จะแบ่งปันข้อมูลต่าง ๆ ที่เกี่ยวข้องกับสัญญา ไม่ว่าจะ กฎ ระเบียบ ข้อบังคับที่เกี่ยวข้องกับสัญญา ภาระหน้าที่และบทบาทความรับผิดชอบของแต่ละคนที่อยู่ภายใต้สัญญานั้น ๆ รวมไปถึงการติดตามสถานะของสัญญาที่มีการเปลี่ยนแปลงตลอดเวลา ซึ่งส่งผลกระทบต่อการดำเนินงานภายใต้สัญญาและผู้ที่เกี่ยวข้อง โดยมีการออกแบบสถาปัตยกรรมออกเป็น 3 ส่วนหลัก ๆ ด้วยกัน โดยด้านบนสุดเป็นออนโทโลยีเกี่ยวกับสัญญาโดยทั่วไป ในชั้นถัดมาจะเป็นออนโทโลยีที่มีลักษณะเฉพาะของแต่ละประเภทสัญญา และในชั้นล่างสุดจะเป็นออนโทโลยีที่มีการกำหนดรูปแบบ (template) ของสัญญาไว้ล่วงหน้า ซึ่งในการออกแบบออนโทยีทั้ง 3 ส่วนนั้น มีการแบ่งช่วงในการพัฒนาออกเป็นสองระยะคือ ระยะแรกจะเน้นที่การออกแบบแนวคิดโดยภาพรวม จากนั้นในระยะที่สองจะใช้วิธีสร้างแบบจำลองตามแนวคิดที่ได้ออกแบบ ซึ่งในขั้นตอนของการสร้างแบบจำลองนั้นผู้วิจัยได้ใช้ภาษา RDFS, ภาษา DAML+OIL และภาษา OWL โดยใช้เครื่องมือ โปรเทเจ

ต่อมาในปี 2015 มีงานวิจัยที่ชื่อว่า Discovering Business Models for Software Process Management-An Approach for Integrating Time and Resource Perspectives from Legacy Information Systems [22] โดย C. Arévalo Maldonado, I. Ramos Román, and M. J. Escalona Cuaresma งานวิจัยนี้มีวัตถุประสงค์เพื่อผสานระบบดั้งเดิม (Legacy Information

System : LIS) และระบบการจัดการเนื้อหาขององค์กร (Enterprise Content Management : ECM) ให้เป็นการจัดการกระบวนการทางธุรกิจ (Business Process Management : BPM) ที่สามารถรองรับกระบวนการทางธุรกิจสมัยใหม่ที่เกี่ยวข้องกับการทำงานร่วมกันของบุคคลากรจากหลากหลายหน้าที่ รวมไปถึงผู้มีส่วนได้ส่วนเสียที่เกี่ยวข้องกับธุรกิจนั้น ๆ ตั้งแต่เริ่มต้นจนสิ้นสุดกระบวนการ โดยใช้วิธีการทางวิศวกรรมในการสร้างแบบจำลอง (Model-Driven Engineering : MDE) ของทั้งสองระบบให้เป็นแบบจำลองเมตาโมเดลของกระบวนการทางธุรกิจ (Business Process Definition Metamodel : BPDM) จากนั้นจึงแปลงโค้ดในระบบเดิมให้เป็นแบบจำลองที่เป็นนามธรรมยิ่งขึ้น โดยกำหนดเงื่อนไขเฉพาะที่เกี่ยวข้องกับข้อจำกัดด้านเวลาและการจัดสรรทรัพยากรในงานบริหารโครงการ มีขั้นตอนในการแปลงดังรูปที่ 2.4 โดยมีขั้นตอนดังต่อไปนี้

1) แปลงตาราง (Table) และค่าเชื่อมโยง (Foreign Key) ระหว่าง 2 ตาราง ให้เป็นคลาส ส่วนคอลัมน์ (Column) ในตารางก็ทำการแปลงให้เป็นคุณลักษณะของคลาส
2) แปลงข้อจำกัดของฐานข้อมูล (Database constraint) ให้เป็นข้อจำกัดในภาษาโอซีแอล (Object Constraint Language : OCL)
3) แปลงข้อจำกัดด้านเวลาและทรัพยากรที่ใช้ด้วยข้อจำกัดภาษาโอซีแอลและหลักการอนุมาน (Infer) โดยคำนึงถึงอนุกรมวิธาน (Taxonomy) ของกฎทางกระบวนการธุรกิจดังแสดงในรูปที่ 2.4 เพื่อสร้างคลาสใหม่

ซึ่งแนวคิดดังกล่าวสามารถประยุกต์ใช้ได้กับระบบดั้งเดิมได้เป็นอย่างดี นอกจากนี้ยังสามารถทำงานร่วมกับเมตาโมเดลของ BPMN และ OCL ได้ซึ่งทำให้เพิ่มส่วนต่อขยายให้สามารถเพิ่มกฎการจัดสรรทรัพยากร เพื่อมุมมองใหม่ๆ ให้กับผู้บริหารเพื่อใช้เป็นข้อมูลในการตัดสินใจ



รูปที่ 2.5 อนุกรมวิธาน (Taxonomy) ของกฎทางกระบวนการธุรกิจ [22]

2.2.2 งานวิจัยที่เกี่ยวข้องกับการนำแผนภาพแสดงการทำงานของกิจกรรมอื่น ๆ มาแปลงเป็นออนโทโลยีเพื่อตรวจสอบความต้องกัน

ในปี ค.ศ. 2014 มีงานวิจัยที่ชื่อว่า A Semantic Mapping Representation and Generation Tool Using UML for System Engineers [23] โดย S.-H. Chung, W. Tai, D. O'Sullivan, and A. Boran ได้นำเสนอแนวคิดในการแปลงภาษายูเอ็มแอล ให้มีความสามารถในการอธิบายเชิงความหมาย (Semantic) ได้อย่างเป็นรูปธรรมมากขึ้น โดยใช้แนวคิดของการอธิบายแบบจำลองยูเอ็มแอลด้วยเมตาเดต้า โดยเปลี่ยนแนวคิดจากนามธรรมให้เป็นรูปธรรมหรือการอธิบายด้วยภาษาที่เป็นทางการ ซึ่งในขณะนั้นมีวิธีการสร้างแบบจำลองที่นิยมใช้กัน 3 วิธี คือ

1) การใช้ภาษาอาวล์ เนื่องจากภาษาอาวล์ มีการใช้ XML elements เป็นหลักซึ่งเปรียบได้กับ Class และ Property ใน UML แต่ ภาษาอาวล์ ก็มีข้อจำกัดในเรื่องของการระบุเครื่องหมายการดำเนินการทางคณิตศาสตร์ (Arithmetic operations)

2) การใช้ภาษาเว็บกฎเชิงความหมาย (Semantic Web Rule Language - SWRL) เป็นภาษาที่สามารถนิยามกฎต่าง ๆเข้าไปในออนโทโลยีได้ นอกเหนือจากคำจำกัดความของแนวคิด ซึ่งสามารถที่จะอธิบายและครอบคลุมความหมายของแนวคิดได้เป็นอย่างดี แต่การใช้งานก็ค่อนข้างยากด้วยไวยากรณ์ (Syntax) ภาษาที่ค่อนข้างซับซ้อน จำเป็นต้องมีความรู้และความเชี่ยวชาญในการเขียนกฎ

3) SPARQL Protocol และ อาร์ดีเอฟ Query Language SPARQL เป็นภาษาที่ใช้ในการเขียนคำสั่ง query สำหรับ อาร์ดีเอฟ ซึ่งข้อมูลจะอยู่ในรูปแบบกราฟที่เชื่อมต่อกันโดยตรง (directed graph data) และมีการระบุป้ายชื่อเพื่อแสดงข้อมูล สามารถแปลงให้เป็นข้อมูลเชิงความหมายในขณะ query time ได้ทันที แต่ไวยากรณ์ก็ยังมีความยากสำหรับผู้ที่ไม่คุ้นเคยกับเทคโนโลยีเว็บเชิงความหมาย (Semantic Web Ontology)

สำหรับงานวิจัยนี้ได้เลือกใช้ ภาษาอาวล์ และภาษาเอสดับเบิลยูอาร์แอล เนื่องจากเมตาโมเดลของภาษายูเอ็มแอลประกอบไปด้วยคลาส (Class) ซับคลาส (Sub class) ลำดับชั้นของคลาส (Hierarchy) ความสัมพันธ์ระหว่างคลาส (Relationship) ข้อกำหนดด้านคุณลักษณะของคลาส (Class attribute definition) และสัจพจน์ (Axiom) ที่ใช้ในการระบุข้อข้อจำกัด (Constraint) ซึ่งส่วนประกอบเหล่านี้ สามารถแปลงมาเป็นออนโทโลยี เพื่อใช้การอธิบายความสัมพันธ์ (Relationship) ระหว่างแนวคิด (Concept) ที่อยู่ภายใต้ขอบเขต (Domain) ที่สนใจ ให้อยู่ในรูปแบบหรือข้อกำหนดที่เป็นทางการ (Formal specification) ได้ [9] นอกจากนี้ยังสามารถอธิบายแนวคิดให้อยู่ในรูปของคลาสและความสัมพันธ์โดยใช้ความจริงหรือสัจพจน์ (axiom) ได้อีกด้วย จะเห็นได้ว่าทั้ง

UML และออนโทโลยีต่างก็สามารถอธิบายแนวคิดและความสัมพันธ์ระหว่างคลาสได้เช่นเดียวกัน ดังนั้นในงานวิจัยชิ้นนี้จะใช้เทคนิคการแปลงแผนภาพข่ายงานแบบพีดีเอ็ม ให้เป็นยูเอ็มแอลเมตาโมเดลก่อน จากนั้นจึงนำไปออกแบบออนโทโลยีในต่อไป

ต่อมาในปี ค.ศ. 2015 มีงานวิจัยที่ชื่อว่า Consistency of UML class, object and statechart diagrams using ontology reasoners [8] โดย A. H. Khan and I. Porres ได้นำเสนอวิธีการวิเคราะห์ความต้องกันของแบบจำลองภาษายูเอ็มแอลในรูปแบบต่าง ๆ ที่ถูกนำมาผสานเข้าไว้ด้วยกัน ซึ่งประกอบไปด้วย แผนภาพคลาส แผนภาพอ็อบเจ็ค และแผนภาพแสดงสถานะ (State chart diagram) โดยทำการแปลงแบบจำลองภาษายูเอ็มแอลแต่ละรูปแบบให้อยู่ในรูปของภาษาอาวล์ เวอร์ชัน 2 จากนั้นจึงใช้เครื่องมือในการให้เหตุผล (Reasoner) ของภาษาอาวล์ ซึ่งในงานวิจัยนี้ได้เลือกใช้เครื่องมือ 2 ชนิดด้วยกันคือ Pellet และ HermiT ซึ่งทั้งคู่เป็นเครื่องมือชนิดโอเพนซอร์ซภาษาจาวาที่รองรับการใช้งานภาษาอาวล์ เวอร์ชัน 2 และยังสนับสนุนการอนุมานด้วยภาษาเอสดับเบิลยูอาร์แอลอีกด้วย ซึ่งจะช่วยให้การวิเคราะห์ความต้องกันของแบบจำลองเป็นไปอย่างอัตโนมัติ การที่เลือกใช้เครื่องมือ 2 ชนิดก็เพื่อที่จะวัดประสิทธิภาพในการประมวลผลตามแนวคิดหรือวิธีการที่ได้ออกแบบมา จากนั้นจึงทำการทดสอบแนวคิดดังกล่าวกับแบบจำลองกว่า 2,000 รูปแบบ เพื่อตรวจจับความผิดพลาดในระหว่างขั้นตอนการตรวจสอบความความต้องกัน ซึ่งในการทดสอบนั้นผู้วิจัยได้ทำการออกแบบสถานการณ์เป็น 3 รูปแบบ คือ 1) การตรวจความต้องกันของแบบจำลองที่ผสานกันหลายรูปแบบ เช่น การนำแผนภาพคลาสมาผสานกับแผนภาพอ็อบเจ็ค เป็นต้น 2) ตรวจสอบความสอดคล้องระหว่างอ็อบเจ็คและภาษาโอซีแอล (OCL) และ 3) ตรวจสอบความต้องกันของแผนภาพคลาส และแผนภาพแสดงสถานะ

และในปีเดียวกันก็ได้มีการนำเสนองานวิจัยที่ชื่อว่า Ontology-based workflow validation โดย T. A. Pham and N. Le Thanh ปี ค.ศ. 2015 [24] ซึ่งงานวิจัยนี้ได้นำเสนอแนวคิดในการอธิบายกระบวนการทำงาน (Workflow) ที่ประมวลผลด้วย วิธีซีพีเอ็น (Coloured Petri Net : CPN) ให้เป็นออนโทโลยีเพื่อให้สามารถอธิบายเชิงความหมายได้ โดยการออกแบบให้สามารถตรวจสอบความต้องกันในเรื่องของเวลาและความต้องกันขณะประมวลผลโดยอัตโนมัติด้วยออนโทโลยี ภาษาอาวล์ DL นอกจากนี้ยังออกแบบให้ผู้ใช้งานสามารถปรับเปลี่ยนรูปแบบของกระบวนการทำงานได้ในขณะที่การอธิบายเชิงความหมายยังคงถูกต้องและครบถ้วนอยู่ โดยเริ่มต้นจากการกำหนดวิธีการอธิบายความหมายในรูปแบบที่เป็นทางการของข้อกำหนดต่าง ๆ ในกระบวนการทำงาน จากนั้นก็พัฒนาออนโทโลยีที่เพื่ออธิบายความหมายของข้อจำกัดนั้น ๆ แล้วจึงกำหนดชุดของกระบวนการที่ผู้ใช้สามารถเปลี่ยนแปลงได้ สุดท้ายคือการพัฒนาแบบสอบถามเพื่อตรวจสอบความต้องกันของกระบวนการด้วยภาษาสปาเคิล (SPARQL) เมื่อนำมาเปรียบเทียบกับงานวิจัยที่จะทำนี้ ซึ่ง

เป็นการใช้แผนภาพข่ายงานแบบพีดีเอ็มเป็นหลักและยังไม่มีเครื่องมือใดที่ใช้ในการแปลงเป็นภาษาอาวล์ โดยอัตโนมัติ จึงจำเป็นต้องออกแบบแผนภาพข่ายงานแบบพีดีเอ็ม ให้อยู่ในรูปแบบของ เมตาโมเดลก่อนที่จะทำการแปลงเป็นภาษาอาวล์ ด้วยเครื่องมือ โปรเทเจ แต่ในส่วนของการตรวจสอบนั้นจะใช้กฎของเอสดับเบิลยูอาร์แอล มาใช้ในการการให้เหตุผล (Reasoning) เพื่ออนุมาน (Inference) ความรู้ใหม่จากข้อมูลที่มีอยู่มา เพื่อเพิ่มความสามารถของออนโทโลยีให้สามารถให้คำแนะนำเพิ่มเติมกรณีที่มีการปรับเปลี่ยนจำนวนวันในแผนภาพข่ายงานแบบพีดีเอ็มได้

นอกจากนี้ยังมีงานวิจัยอื่น ๆ ที่นิยมใช้แผนภาพแสดงการทำงานของแบบจำลองภาษายูเอ็มแอล (Unified Modeling Language : UML) [25] ซึ่งถือเป็นหนึ่งในภาษาที่นิยมใช้ในการอธิบายแบบจำลอง โดยเฉพาะแบบจำลองเชิงวัตถุ (Object-oriented Modeling) โดยใช้แนวคิดของการอธิบายแบบจำลองยูเอ็มแอลด้วยเมตาเดต้า โดยเปลี่ยนแนวคิดจากนามธรรมให้เป็นรูปธรรมหรือการอธิบายด้วยภาษาที่เป็นทางการด้วยวิธีออนโทโลยี โดยทำการแปลงแผนแสดงการทำงานของกิจกรรมต่างๆ ไม่ว่าจะเป็น แผนภาพแสดงกิจกรรม (Activity diagram) [26] [27] [28] แผนภาพแสดงสถานะ (Statechart diagram [8]) แผนภาพแสดงการทำงาน (Workflow diagram) [24] [29] แผนภาพแสดงลำดับการทำงาน (Sequence diagram) [30] แผนภาพแสดงการทำงานร่วมกัน (Collaboration diagram) โดยสรุปแล้วพบว่างานวิจัยเหล่านี้ได้ใช้หลักการการวิเคราะห์เชิงเปรียบเทียบการใช้งานเครื่องมือสำหรับสร้างแบบจำลองระหว่างโปรเทเจที่ใช้สำหรับภาษาอาวล์ และเครื่องมือแสดงกระบวนทัศน์ของภาษายูเอ็มแอล ถึงแม้ว่าวัตถุประสงค์การใช้งานระหว่างภาษาอาวล์และภาษายูเอ็มแอลนั้นจะแตกต่างกัน แต่การอธิบายโครงสร้างของส่วนประกอบต่าง ๆ กลับมีลักษณะที่คล้ายกัน เช่น การนำเสนอโครงสร้างด้วยแผนภาพคลาส เป็นต้น หากเป็นคลาสในภาษาอาวล์จะประกอบไปด้วยข้อมูลตัวอย่าง (Individual) โดยที่ข้อมูลตัวอย่างเหล่านี้จะถูกระบุให้เป็นอิสระจากคลาส คือไม่ขึ้นกับคลาสใดคลาสหนึ่งโดยตรง คลาสจะถูกแบ่งตามลำดับชั้นในลักษณะของซูเปอร์คลาสและซับคลาสหรือที่เรียกกันว่าอนุกรมวิธาน (Taxonomy) ในการที่จะจับคู่คลาสและข้อมูลตัวอย่างจะต้องเป็นไปตามเงื่อนไขที่ระบุ และคลาสจะต้องถูกสร้างขึ้นจากคำอธิบายของเงื่อนไขเหล่านี้เช่นกัน ส่วนคลาสในภาษายูเอ็มแอลนั้นจะใช้ในการแสดงโครงสร้างของระบบโดยทั่วไป โดยมีการอธิบายคุณลักษณะของโครงสร้างและการทำงานในลักษณะของพฤติกรรมและความสัมพันธ์กับคลาสอื่น ๆ แต่ทั้งสองภาษานี้ต่างก็สนับสนุนการสร้างแบบจำลองเชิงโครงสร้าง โดยที่ภาษาอาวล์จะเรียกว่าเป็นออนโทโลยี ส่วนในภาษายูเอ็มแอลจะเรียกว่าเป็นแพ็คเก็จ (Package) โดยสามารถจับคู่ลักษณะที่คล้ายกันของภาษายูเอ็มแอลและภาษาอาวล์ ได้ดังตารางที่ 2.3

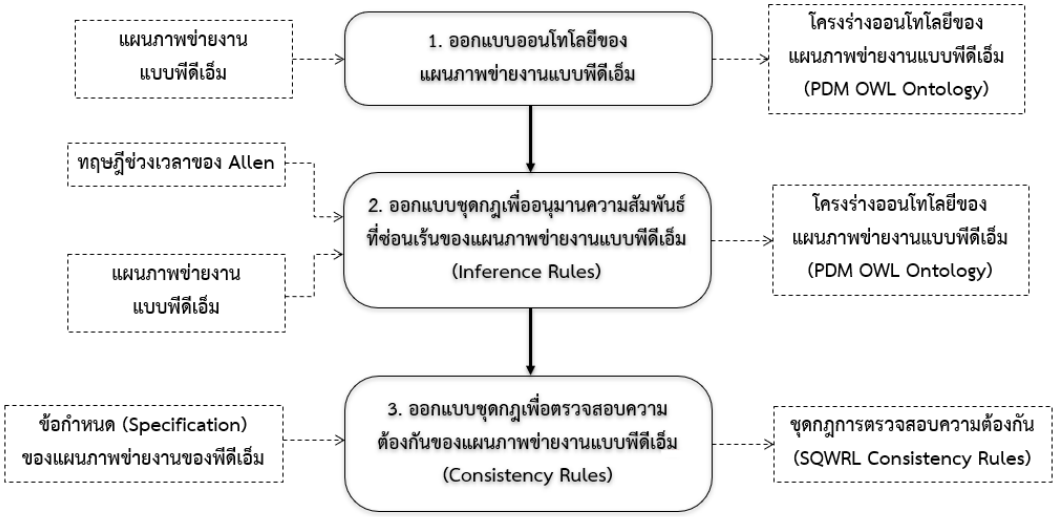
ตารางที่ 2.3 เปรียบเทียบส่วนประกอบระหว่าง ภาษายูเอ็มแอล และ ภาษาอาวล์ [31]

ส่วนประกอบในภาษายูเอ็มแอล	ส่วนประกอบในภาษาอาวล์
Class	Class
Subclass	Subclass
Instance	Individual
Attribute, Binary association	Property
Enumeration	OneOff
Multiplicity	minCardinality maxCardinality
Package	Ontology

โดยในงานวิจัยได้ทำการทดสอบการสร้างแผนภาพยูเอ็มแอลด้วยเครื่องมือที่สามารถแสดงกระบวนทัศน์ของภาษายูเอ็มแอล (Visual Paradigm for UML) จากนั้นจึงออกแบบออนโทโลยีโดยการอธิบายความสัมพันธ์ระหว่างคลาสต่าง ๆภายใต้แบบจำลองยูเอ็มแอลที่สร้างขึ้นมา โดยใช้โปรเทเจเป็นเครื่องมือในสร้างออนโทโลยีและส่งออกมาเป็นไฟล์ภาษาอาวล์ จากนั้นนำไฟล์ภาษาอาวล์ที่สร้างได้ไปใช้งานในลักษณะของเว็บเชิงความหมาย (Semantic Web) พบว่าสามารถประมวลผลได้โดยคอมพิวเตอร์และคอมพิวเตอร์ยังสามารถรวบรวม (Integrate) ข้อมูลระหว่างเว็บต่าง ๆ เข้าไว้ด้วยกัน

บทที่ 3
การออกแบบออนโทโลยีและกฎของแผนภาพข่ายงานแบบพีดีเอ็ม

งานวิจัยนี้เป็นการสร้างแบบจำลองของแผนภาพข่ายงานแบบพีดีเอ็มด้วยวิธีออนโทโลยี เพื่ออธิบายความรู้และตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม ซึ่งประกอบด้วย 3 ขั้นตอนหลัก ๆ ตามรูปที่ 3.1



รูปที่ 3.1 ขั้นตอนในการออกแบบออนโทโลยีและกฎของแผนภาพข่ายงานแบบพีดีเอ็ม

ในขั้นตอนแรกจะเป็นการออกแบบออนโทโลยีของแผนภาพข่ายงานแบบพีดีเอ็ม โดยจะใช้ข้อมูลแผนภาพข่ายงานแบบพีดีเอ็มมาเป็นข้อมูลนำเข้า โดยจะพิจารณาองค์ประกอบต่าง ๆ บนแผนภาพ แล้วจึงทำการออกแบบออนโทโลยี ซึ่งในหัวข้อที่ 3.1 จะอธิบายขั้นตอนการออกแบบโดยละเอียด ผลลัพธ์ที่ได้จากขั้นตอนนี้คือ โครงร่างออนโทโลยีของแผนภาพข่ายงานแบบพีดีเอ็ม ที่จะถูกนำไปสร้างเป็นออนโทโลยีในภาษาอาวล์ ด้วยเครื่องมือโปรเทเจต่อไป

ต่อมาจะทำการออกแบบชุดกฎ 2 ประเภท คือ ชุดกฎเพื่ออนุมานความสัมพันธ์ที่ซ่อนเร้นของแผนภาพข่ายงานแบบพีดีเอ็ม (Inference Rules) และชุดกฎเพื่อตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม (Consistency Rules) โดยชุดกฎที่ใช้อนุมานความสัมพันธ์ที่ซ่อนเร้นจะทำการสกัดองค์ความรู้เพิ่มเติมนอกเหนือจากองค์ความรู้ที่ชัดแจ้งที่ได้จากข้อมูลนำเข้า โดยการอนุมานจะทำได้โดยการพิจารณาองค์ความรู้เดิมที่เป็นโครงสร้างของ PDM ว่ามีความสัมพันธ์หรือการพึ่งพารูปแบบใดที่สอดคล้องกับทฤษฎีช่วงเวลาของ Allen ทั้ง 13 ข้อ ได้แก่ Before, After, Meet,

MetBy, Overlaps, OverlappedBy, Starts, StartedBy, During, Contains, Finishes, FinishedBy, และ Equals จากนั้นทำการแปลงความสัมพันธ์ดังกล่าวออกมาให้อยู่ในรูปของภาษาเอสดับเบิลยูอาร์แอล (SWRL) แล้วจึงนำกฎชุดนี้ไปสร้างด้วยเครื่องมือโปรเทเจ จากนั้นทำการประมวลผลกฎผ่านตัวให้เหตุผล (Reasoner) ที่ชื่อว่า Drools ผลลัพธ์จากการประมวลผลด้วยเครื่องมือดังกล่าวจะทำให้ได้ข้อมูลหรือองค์ความรู้ใหม่ ที่จะสามารถนำไปใช้เป็นข้อมูลสำหรับการตรวจสอบความต้องกันของประเภทการพึ่งพา (Dependency Type) ต่าง ๆ ตามนิยามของแผนภาพข่ายงานแบบพีดีเอ็มได้ ซึ่งในหัวข้อที่ 3.2 จะอธิบายขั้นตอนการออกแบบกฎที่ซ่อนเร้นโดยละเอียด

กฎชุดที่สอง คือ ชุดกฎที่ใช้ตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม กฎชุดนี้จะพิจารณาจากองค์ประกอบหลักของแผนภาพข่ายงานแบบพีดีเอ็ม ว่ามีส่วนใดบ้างที่จำเป็นจะต้องตรวจสอบความต้องกัน ซึ่งประกอบไปด้วย 3 ส่วน ได้แก่ กิจกรรมในแต่ละโหนด (Node) ความสัมพันธ์หรือความพึ่งพากัน (Dependency) และทรัพยากรบุคคล (Resource) ดังนั้นในการออกแบบกฎ จะพิจารณาจากข้อกำหนด (Specification) ของแผนภาพข่ายงานของพีดีเอ็มเป็นหลัก เช่น ในกรณีพิจารณาความต้องกันของโหนด ซึ่งประกอบไปด้วยค่าของช่วงเวลาต่าง ๆ จากข้อกำหนดที่กล่าวว่า เวลาที่เริ่มต้นของกิจกรรมใด ๆ จะต้องน้อยกว่าเวลาที่กิจกรรมนั้นสิ้นสุด ดังนั้น กฎที่ออกแบบจะต้องสามารถตรวจสอบได้ว่าค่าที่นำเข้ามานั้นจะต้องสอดคล้องตามข้อกำหนด จากนั้นทำการแปลงข้อกำหนดของแผนภาพข่ายงานแบบพีดีเอ็มให้อยู่ในรูปแบบภาษาเอสคิวดับเบิลยูอาร์แอล (SQWRL) ซึ่งในหัวข้อที่ 3.3 จะอธิบายขั้นตอนการออกแบบกฎที่ใช้ตรวจสอบความต้องกันโดยละเอียด ซึ่งกฎชุดนี้จะถูกนำไปใช้งานในเครื่องมือที่ผู้วิจัยได้ทำการพัฒนาขึ้น โดยจะถูกจัดเก็บไว้ใน SQWRL Rules Configuration File เพื่อให้โปรแกรมจาวาสามารถเรียกใช้งานกฎใน Configuration File ได้ ซึ่งจะอธิบายรายละเอียดในบทถัดไป

3.1 การออกแบบออนโทโลยีของแผนภาพข่ายงานแบบพีดีเอ็ม

ในส่วนของการออกแบบออนโทโลยี ผู้วิจัยได้เริ่มต้นจากการศึกษาข้อมูลที่เกี่ยวข้องกับแผนภาพข่ายงานแบบพีดีเอ็ม เพื่อนำมาวิเคราะห์ส่วนประกอบ จากนั้นจึงทำการออกแบบและพัฒนาออนโทโลยีแผนภาพข่ายงานแบบพีดีเอ็ม

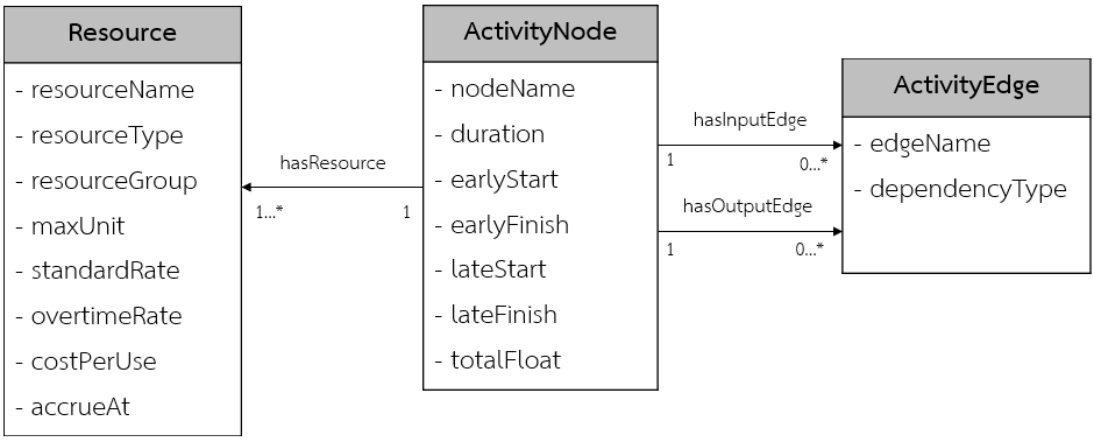
3.1.1 ศึกษาข้อมูลที่เกี่ยวข้องกับแผนภาพข่ายงานแบบพีดีเอ็ม

ในขั้นตอนนี้จะเป็นการศึกษารายละเอียดที่เกี่ยวข้องกับแผนภาพข่ายงานแบบพีดีเอ็ม ตั้งแต่ส่วนประกอบของแผนภาพ คำอธิบายเกี่ยวกับข้อมูลที่แสดงบนแผนภาพ ตลอดจนข้อมูลอื่น ๆ ที่เกี่ยวข้องที่ไม่ได้แสดงบนแผนภาพ การคำนวณและการได้มาของข้อมูลต่าง ๆ ทั้งนี้ประโยชน์สูงสุดของแผนภาพข่ายงานแบบพีดีเอ็ม นอกจากจะใช้ในการวางแผนเพื่อจัดการตารางงานโครงการให้มี

ประสิทธิภาพแล้ว ยังใช้ในการวางแผนทรัพยากรโดยเฉพาะทรัพยากรบุคคล ดังนั้นจึงทำการศึกษาเพิ่มเติมเกี่ยวกับข้อมูลที่เกี่ยวข้องกับทรัพยากรบุคคล เพื่อใช้ในการมอบหมายงานในแต่ละกิจกรรม รวมถึงค่าใช้จ่ายที่เกิดขึ้นในโครงการอีกด้วย ซึ่งทั้งหมดนี้ได้อธิบายรายละเอียดไว้ในบทที่ 2

3.1.2 วิเคราะห์ส่วนประกอบของแผนภาพข่ายงานแบบพีดีเอ็ม

จากข้อมูลต่าง ๆ บนแผนภาพข่ายงานแบบพีดีเอ็ม ดังแสดงตามรูปที่ 2.1 และ 2.2 ผู้วิจัยได้วิเคราะห์ส่วนประกอบบนแผนภาพอย่างละเอียดและพบว่า ส่วนประกอบที่สำคัญของแผนภาพข่ายงานแบบพีดีเอ็ม ถูกแบ่งออกเป็น 3 ส่วนหลัก คือ ส่วนที่เป็นกิจกรรม (Activity node) ที่มีการอธิบายรายละเอียดเกี่ยวกับเวลาที่ใช้ในการดำเนินแต่ละกิจกรรมในรูปแบบต่าง ๆ เช่น เวลาเริ่มต้นกิจกรรม เวลาสิ้นสุดกิจกรรม ระยะเวลาที่ใช้ในการดำเนินกิจกรรม เป็นต้น และส่วนที่เป็นเส้นเชื่อม (Edge) ที่ใช้ในการกำหนดความสัมพันธ์ (Dependency) ระหว่างกิจกรรมที่อยู่ก่อนหน้า (Predecessor) และกิจกรรมที่ตามหลัง ซึ่งจะมีรูปแบบความสัมพันธ์ใน 4 ลักษณะ ได้แก่ Finish to Start (FS), Start to Start (SS), Finish to Finish (FF) และ Start to Finish (SF) และส่วนที่เป็นการจัดสรรทรัพยากร ซึ่งจะแสดงรายละเอียดต่าง ๆ ที่เกี่ยวข้องกับทรัพยากรนั้น ๆ เช่น ชื่อ ตำแหน่งหน้าที่ ความสามารถในการรองรับงาน ดังนั้นเมื่อนำแนวคิดนี้มาออกแบบเป็นเมตาโมเดลโดยใช้การอธิบายด้วยแผนภาพคลาส (Class Diagram) จะได้ดังรูปที่ 3.2



รูปที่ 3.2 เมตาโมเดลของแผนภาพข่ายงานแบบพีดีเอ็ม

จากรูปที่ 3.2 เมตาโมเดลของแผนภาพข่ายงานที่ออกแบบได้นั้นประกอบไปด้วย 3 คลาสหลัก คือ คลาส ActivityNode คลาส ActivityEdge และคลาส Resource ซึ่งคลาส ActivityNode ใช้ในการอธิบายคุณลักษณะ (attribute) ของกิจกรรมดังนี้

- nodeName คือ ชื่อของกิจกรรม

- duration คือ ระยะเวลาที่ใช้ในการดำเนินกิจกรรม มีหน่วยเป็นวัน
- earlyStart คือ วันที่เริ่มทำงานที่เร็วที่สุดของกิจกรรม
- earlyFinish คือ วันที่งานเสร็จได้เร็วที่สุดของกิจกรรม
- lateStart คือ วันที่เริ่มงานที่ช้าที่สุดของกิจกรรม
- lateFinsih คือ วันที่งานเสร็จได้ช้าที่สุดของกิจกรรม
- totalFloat คือ เวลาสำรองของแต่ละกิจกรรม สามารถหาได้จาก (LF-EF) หรือ (LS-ES)

ส่วนคลาส ActivityEdge นั้น ใช้ในการอธิบายคุณลักษณะ ของความสัมพันธ์ระหว่าง 2 กิจกรรมดังนี้

- edgeName คือ ชื่อของเส้นเชื่อม
- dependencyType คือ ความสัมพันธ์ของเส้นเชื่อม ซึ่งประกอบไปด้วย 4 ลักษณะ ได้แก่ FS, FF, SF, SS

และสุดท้ายคือคลาส Resource นั้นใช้ในการอธิบายคุณลักษณะ ของทรัพยากรที่ใช้ภายในงานโครงการ ในที่นี้จะหมายถึงทรัพยากรบุคคลที่จะต้องรับผิดชอบงานในแต่ละกิจกรรม ซึ่งประกอบด้วยคุณลักษณะดังนี้

- resourceName คือ ชื่อของทรัพยากร
- resourceType คือ ประเภทของทรัพยากร ซึ่งประกอบไปด้วย 3 ประเภทได้แก่ บุคลากร (Work) วัสดุอุปกรณ์ (Material) และค่าใช้จ่าย (Cost)
- resourceGroup คือ กลุ่มของทรัพยากร เช่น หากเป็นทรัพยากรประเภทบุคลากร ก็จะสามารถแบ่งกลุ่มได้เป็น กลุ่มผู้พัฒนา กลุ่มผู้วิเคราะห์ระบบ กลุ่มผู้ทดสอบระบบ เป็นต้น
- maxUnit คือ ความสามารถในการรองรับงานของทรัพยากรในช่วงเวลาที่กำหนด
- standardRate คือ อัตราค่าจ้างในราคามาตรฐาน หรือค่าจ้างที่จ่ายให้แก่ทรัพยากรกรณีที่มีการทำงานในเวลาปกติ
- overtimeRate คือ อัตราค่าจ้างในราคาพิเศษ หรือค่าจ้างที่จ่ายให้แก่ทรัพยากรกรณีที่มีการทำงานนอกเวลาปกติ
- costPerUse คือ ค่าใช้จ่ายที่เกิดขึ้นเมื่อมีการใช้งานทรัพยากร
- accrueAt คือ ใช้เพื่อระบุช่วงเวลาในการคำนวณค่าใช้จ่ายที่จะเกิดขึ้น ซึ่งแบ่งออกเป็น 3 วิธี ได้แก่ คำนวณตอนเริ่มต้นงาน (Start) คำนวณตอนทำงานแล้วเสร็จ (Finish) และคำนวณตามสัดส่วนที่ทำงานจริง (Prorated)

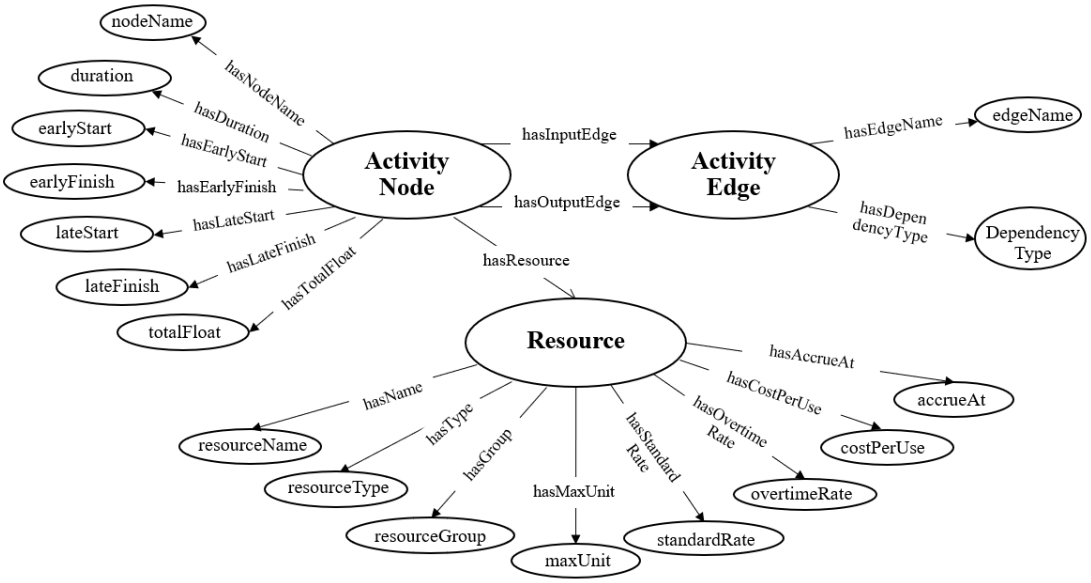
นอกจากนี้เส้นที่ปรากฏระหว่างคลาส ActivityNode และ ActivityEdge ยังใช้ในการแสดงความสัมพันธ์ระหว่างกิจกรรมและเส้นเชื่อม ซึ่งประกอบไปด้วย ความสัมพันธ์ 2 ประเภท ได้แก่

- hasInputEdge เป็นการอธิบายว่ากิจกรรมใดใด มีเส้นเชื่อมขาเข้าเป็นเส้นใด
- hasOutputEdge เป็นการอธิบายว่ากิจกรรมใดใด มีเส้นเชื่อมขาออกเป็นเส้นใด

และเส้นเชื่อมที่ปรากฏระหว่างคลาส ActivityNode และ Resource มีความสัมพันธ์ในรูปแบบ hasResource คือใช้อธิบายว่า ActivityNode มีการใช้งาน Resource อย่างไรบ้าง จะเห็นได้ว่าองค์ประกอบต่าง ๆ ที่อยู่ในแผนภาพข่ายงานแบบพีดีเอ็มสามารถนำมาออกแบบให้เป็นเมตาโมเดล เพื่อใช้ในการอธิบายแผนภาพข่ายงานแบบพีดีเอ็มได้อย่างครบถ้วน

3.1.3 ออกแบบและพัฒนาออนโทโลยีแผนภาพข่ายงานแบบพีดีเอ็ม

เมื่อได้เมตาโมเดลของแผนภาพข่ายงานแบบพีดีเอ็มจากขั้นตอนที่ 3.1.2 แล้ว ในส่วนถัดมาเป็นขั้นตอนการแปลงพีดีเอ็มเมตาโมเดลให้เป็นออนโทโลยีด้วยภาษาอาวล์ เนื่องจากยูเอ็มแอลเมตาโมเดล ใช้ในการอธิบายโครงสร้างของแบบจำลองที่มีลักษณะเป็นคลาส โดยอธิบายคุณลักษณะต่าง ๆ ของคลาสนั้น ๆ ในรูปแบบของโครงสร้างและความสัมพันธ์ระหว่างคลาส ซึ่งสอดคล้องกับลักษณะโครงสร้างของภาษาอาวล์ [31] ดังนั้นจากเมตาโมเดลของแผนภาพข่ายงานแบบพีดีเอ็มในรูปที่ 3.2 สามารถอ้างอิงจากงานวิจัยที่เปรียบเทียบส่วนประกอบระหว่างแบบจำลองภาษายูเอ็มแอล (UML) กับภาษาอาวล์ (OWL) [31] ในตารางที่ 2.3 ซึ่งทำการเปรียบเทียบส่วนประกอบระหว่าง ภาษายูเอ็มแอลและภาษาอาวล์



รูปที่ 3.3 แบบจำลองโครงร่างออนโทโลยีของแผนภาพข่ายงานแบบพีดีเอ็มต้นแบบ

จากงานวิจัยพบว่าสามารถแปลงคลาส (Class) ในภาษายูเอ็มแอลให้เป็นคลาส (Class) ในภาษาอาวล์ จากนั้นแปลงคุณลักษณะ (Attribute) ของคลาสในภาษายูเอ็มแอล ให้เป็นคุณลักษณะ

(Property) ในภาษาอาวล์ และข้อมูลอินสแตนซ์ (Instance) ในภาษายูเอ็มแอลให้เป็นข้อมูลอินดิวิดวล (Individual) ในภาษาอาวล์ ผลลัพธ์ที่ได้จากการแปลงสามารถแสดงออกมาเป็นโครงสร้างออนโทโลยีด้วยภาษาอาวล์ ได้ดังรูปที่ 3.3

จากรูปที่ 3.3 จะได้แบบจำลองโครงร่างออนโทโลยีของแผนภาพข่ายงานแบบพีดีเอ็มต้นแบบที่ได้จากการสกัดองค์ความรู้มาจากแผนภาพข่ายงานแบบพีดีเอ็ม ที่ประกอบด้วย 3 คลาสหลัก ได้แก่ คลาส ActivityNode คลาส ActivityEdge และคลาส Resource ซึ่งแต่ละคลาสจะประกอบด้วยคุณลักษณะ (Attribute) ตามคุณลักษณะที่อธิบายบนแผนภาพ โดยที่ ActivityNode จะประกอบไปด้วย nodeName, duration, earlyStart, earlyFinish, lateStart, lateFinish, totalFloat ส่วนคลาส ActivityEdge จะประกอบไปด้วย edgeName และ DependencyType สำหรับคลาส Resource จะประกอบไปด้วย resourceName, resourceType, resourceGroup, maxUnit, standardRate, overtimeRate, costPerUse, และ accrueAt

จากนั้นนำโครงร่างที่ออกแบบได้นี้ไปพัฒนาต่อด้วยเครื่องมือโปรเทเจ แล้วจึงส่งออกมาเป็นไฟล์ภาษาอาวล์เพื่อนำไปใช้ร่วมกับกฎการอนุมานด้วยภาษาเอสดับเบิลยูอาร์แอลต่อไป

3.2 การออกแบบชุดกฎเพื่ออนุมานความสัมพันธ์ที่ซ่อนเร้นของแผนภาพข่ายงานแบบพีดีเอ็ม

โครงร่างออนโทโลยีที่ได้จากการออกแบบในข้อ 3.2.3 นั้น เป็นเพียงการอธิบายลักษณะและส่วนประกอบของแผนภาพข่ายงานแบบพีดีเอ็มซึ่งเป็นองค์ความรู้ที่สามารถแสดงออกมาให้ในรูปแบบของโครงสร้างที่ชัดเจนสามารถเขียนบรรยายลักษณะและรายละเอียดได้ หรือที่เรียกว่าความรู้ชัดแจ้ง (Explicit Knowledge) [32] ซึ่งในงานวิจัยนี้ได้นำเสนอออกมาในรูปแบบของแบบจำลองออนโทโลยีที่ได้ผ่านการวิเคราะห์มาแล้ว นอกจากความรู้ชัดแจ้งที่สามารถนำมาวิเคราะห์และออกแบบเป็นแบบจำลองออนโทโลยีแล้ว ในการที่จะตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็มนั้นพบว่ายังมีความจำเป็นที่จะต้องสกัดองค์ความรู้ที่ซ่อนเร้น (Implicit Knowledge) เพื่อนำมาอธิบายประกอบกับโครงสร้างออนโทโลยีที่ออกแบบได้ เพื่อให้องค์ความรู้ของแผนภาพข่ายงานแบบพีดีเอ็มนั้นมีความสมบูรณ์ยิ่งขึ้น

จากงานวิจัยเกี่ยวกับทฤษฎีความสัมพันธ์ของช่วงเวลาของ Allen (Allen's Theory) [13] พบว่าความสัมพันธ์ทั้ง 13 รูปแบบนั้นสามารถนำมาอธิบายความสัมพันธ์ของลักษณะการเริ่มต้นและสิ้นสุดกิจกรรมของแผนภาพข่ายงานแบบพีดีเอ็มได้อย่างครบถ้วนสมบูรณ์ ผู้วิจัยจึงนำหลักการดังกล่าวมาใช้ในการเขียนกฎเพื่ออนุมานความรู้ที่ซ่อนเร้นในแผนภาพข่ายงานแบบพีดีเอ็ม ซึ่งความสัมพันธ์ทั้ง 13 ข้อ สามารถนำมาสร้างเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ทั้งหมด 7 ข้อ เนื่องจากคุณสมบัติของออนโทโลยีที่สามารถอธิบายความสัมพันธ์แบบผกผันได้โดยอัตโนมัติ ดังนั้นจึง

ทำการออกแบบกฎเพียง 7 ข้อ และกำหนดว่าความสัมพันธ์ทั้ง 7 ข้อนี้มีความสัมพันธ์แบบผกผันกับความสัมพันธ์ใดในเครื่องมือโปรเทเจ ก็จะสามารถอนุมานความสัมพันธ์อีก 6 รูปแบบที่เหลือได้

นอกจากความสัมพันธ์ของช่วงเวลาตามทฤษฎีของ Allen แล้วยังมีความสัมพันธ์ในลักษณะของลำดับการเกิดกิจกรรมก่อน (Predecessor) และหลัง (Successor) ดังนั้นจึงทำการเพิ่มกฎความสัมพันธ์แบบลำดับการเกิดก่อน รวมเป็นกฎทั้งหมด 8 ข้อ ดังตารางที่ 3.1

ตารางที่ 3.1 แสดงกฎการอนุมานความสัมพันธ์ที่ซ่อนเร้นและวัตถุประสงค์ของกฎ

ลำดับ	กฎ	วัตถุประสงค์ของกฎ
1.	กฎความสัมพันธ์แบบก่อนหน้า (intervalBefore)	เพื่ออธิบายความสัมพันธ์ของกิจกรรมที่อยู่ก่อนหน้า และกิจกรรมที่ตามหลัง โดยที่จะต้องรอให้กิจกรรมก่อนหน้าแล้วเสร็จก่อน จึงจะเริ่มต้นกิจกรรมที่ตามหลังได้
2.	กฎความสัมพันธ์แบบประชิด (intervalMeets)	เพื่ออธิบายความสัมพันธ์ที่เมื่อกิจกรรมก่อนหน้าเสร็จสิ้นแล้ว กิจกรรมที่ตามหลังจะเริ่มต้นพร้อมกันในทันที
3.	กฎความสัมพันธ์แบบทับซ้อน (intervalOverlaps)	เพื่ออธิบายความสัมพันธ์ที่มีช่วงระยะเวลาเวลาใดเวลาหนึ่ง ที่ทั้งสองกิจกรรมดำเนินการไปพร้อมกัน
4.	กฎความสัมพันธ์แบบเริ่มต้น (invervalStarts)	เพื่ออธิบายความสัมพันธ์ที่กิจกรรมทั้งสองเริ่มต้นพร้อมกัน
5.	กฎความสัมพันธ์แบบท่ามกลาง (intervalDuring)	เพื่ออธิบายความสัมพันธ์ที่กิจกรรมที่ตามหลังมีระยะเวลาการทำงานครอบคลุมระยะการทำงานของกิจกรรมแรก
6.	กฎความสัมพันธ์แบบสิ้นสุด (intervalFinishes)	เพื่ออธิบายความสัมพันธ์ที่กิจกรรมทั้งสองสิ้นสุดพร้อมกัน
7.	กฎความสัมพันธ์แบบเท่ากัน (intervalEquals)	เพื่ออธิบายความสัมพันธ์ที่กิจกรรมทั้งสองเริ่มต้นและสิ้นสุดพร้อมกัน
8.	กฎความสัมพันธ์แบบลำดับก่อนหน้า (Predecessor)	เพื่ออธิบายความสัมพันธ์ของลำดับกิจกรรม ที่เกิดก่อนหน้าและเกิดตามหลัง

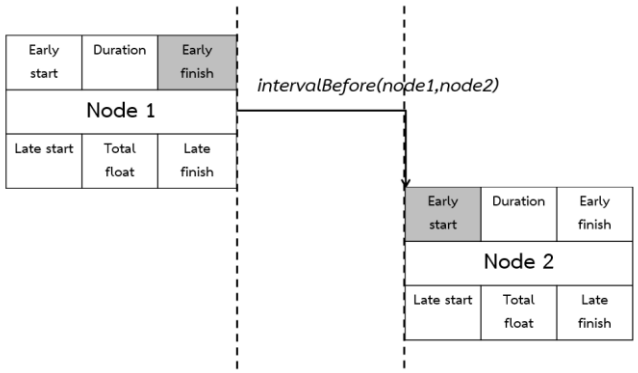
เนื่องจากการพัฒนาออนโทโลยีด้วยภาษาอาวล์นั้นสามารถกำหนดคุณสมบัติของความสัมพันธ์ในลักษณะที่ผกผัน (Inverse) กันได้ ดังนั้นเมื่อคู่กิจกรรมใดมีเงื่อนไขตรงตามกฎที่ได้ระบุ

ไว้ ก็จะสามารถอนุมานความรู้จากทฤษฎีของ Allen ได้ 2 ความสัมพันธ์ (ยกเว้นกฎในลำดับที่ 7 ที่ไม่มีคุณสมบัติของการผกผัน) โดยที่ความสัมพันธ์แรกจะถูกอนุมานให้เป็นคุณสมบัติของกิจกรรมที่อยู่ก่อนหน้า และความสัมพันธ์ผกผันจะถูกอนุมานให้เป็นคุณสมบัติของกิจกรรมที่ตามหลัง

จากนั้นผู้วิจัยจึงนำกฎทั้ง 7 ข้อนี้มาอธิบายด้วยภาษากฎเอสดับเบิลยูอาร์แอล เพื่อความสะดวกในการอธิบายกฎจะกำหนดให้ตัวแปร (?node1) แทนกิจกรรมที่อยู่ก่อนหน้า และกำหนดให้ตัวแปร (?node2) แทนกิจกรรมที่ตามหลัง

3.2.1 กฎความสัมพันธ์แบบก่อนหน้า (intervalBefore)

จากคำนิยามของกฎความสัมพันธ์แบบก่อนหน้าที่กล่าวว่า ถ้ากิจกรรม node1 มีความสัมพันธ์แบบก่อนหน้า กับกิจกรรม node2 แล้ว ดังนั้น เมื่อสิ้นสุดกิจกรรม node1 แล้ว กิจกรรม node2 จึงจะเริ่มต้นได้ สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปภาพ 3.4 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.5



รูปที่ 3.4 กฎความสัมพันธ์ก่อนหน้า (intervalBefore) ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็ม

Node(?node1) ^ Node(?node2) ^
earlyFinish(?node1, ?ef) ^ earlyStart(?node2, ?es) ^
swrlb:lessThan(?ef, ?es) ⟶ intervalBefore(?node1, ?node2)

ชื่อความสัมพันธ์: intervalBefore

วัตถุประสงค์ของกฎ: เพื่ออธิบายความสัมพันธ์ของกิจกรรมที่อยู่ก่อนหน้า และกิจกรรมที่ตามหลัง โดยที่จะต้องรอให้กิจกรรมก่อนหน้าแล้วเสร็จก่อน จึงจะเริ่มต้นกิจกรรมที่ตามหลังได้

คำอธิบาย: ถ้า (?node1) คือค่าตัวแปรของกิจกรรมที่เป็นสมาชิกของคลาส Node และ (?node2) คือค่าตัวแปรของกิจกรรมที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีค่า earlyFinish

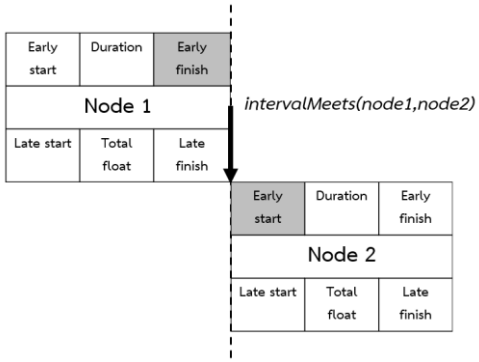
ที่แทนด้วยตัวแปร (?ef) และกิจกรรม (?node2) มีค่า earlyStart ที่แทนด้วยตัวแปร (?es) และค่าของตัวแปร (?ef) น้อยกว่า ค่าของตัวแปร (?es) แล้วจะอนุมานได้ว่า กิจกรรม (?node1) มีความสัมพันธ์แบบก่อนหน้ากับกิจกรรม (?node2)

ชื่อความสัมพันธ์ผกผัน (InverseOf): intervalAfter

รูปที่ 3.5 กฎความสัมพันธ์แบบก่อนหน้า (intervalBefore) ที่อธิบายด้วยภาษา SWRL

3.2.2 กฎความสัมพันธ์แบบประชิด (intervalMeets)

จากคำนิยามของกฎความสัมพันธ์แบบประชิดที่กล่าวว่า ถ้ากิจกรรม node1 มีความสัมพันธ์แบบประชิด กับกิจกรรม node2 แล้ว ดังนั้น เวลาที่สิ้นสุดกิจกรรม node1 จะเท่ากับเวลาเริ่มต้นของกิจกรรม node2 สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.6 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.7



รูปที่ 3.6 กฎความสัมพันธ์แบบประชิด (intervalMeets) ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็ม

Node(?node1) ^ Node(?node2) ^

earlyFinish(?node1, ?ef) ^ earlyStart(?node2, ?es) ^

swrlb:equal(?ef, ?es) ⟶ intervalMeets(?node1, ?node2)

ชื่อความสัมพันธ์: intervalMeets

วัตถุประสงค์ของกฎ: เพื่ออธิบายความสัมพันธ์ที่เมื่อกิจกรรมก่อนหน้าเสร็จสิ้นแล้ว กิจกรรมที่ตามหลังจะเริ่มต้นพร้อมกันในทันที

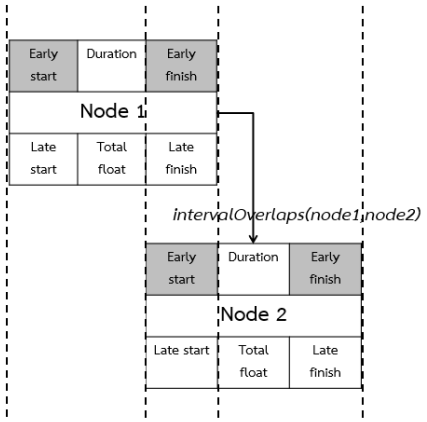
คำอธิบาย: ถ้า (?node1) คือค่าตัวแปรของกิจกรรมที่เป็นสมาชิกของคลาส Node และ (?node2) คือค่าตัวแปรของกิจกรรมที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีค่า earlyFinish ที่แทนด้วยตัวแปร (?ef) และกิจกรรม (?node2) มีค่า earlyStart ที่แทนด้วยตัวแปร (?es) และค่า

ของตัวแปร (?ef) เท่ากับ ค่าของตัวแปร (?es) แล้วจะอนุมานได้ว่า กิจกรรม (?node1) มีความสัมพันธ์แบบประชิดกับกิจกรรม (?node2) **ชื่อความสัมพันธ์ผกผัน (InverseOf):** intervalMetBy

รูปที่ 3.7 กฎความสัมพันธ์แบบประชิด (intervalMeets) ที่อธิบายด้วยภาษา SWRL

3.2.3 กฎความสัมพันธ์แบบทับซ้อน (intervalOverlaps)

จากคำนิยามของกฎความสัมพันธ์แบบทับซ้อนที่กล่าวว่า ถ้ากิจกรรม node1 มีความสัมพันธ์แบบทับซ้อนกับกิจกรรม node2 แล้ว ดังนั้นเวลาที่เริ่มต้นของกิจกรรม node1 จะเกิดก่อนเวลาที่เริ่มต้นของกิจกรรม node2 และเวลาที่สิ้นสุดของกิจกรรม node1 จะเกิดหลังจากเวลาที่เริ่มต้นของกิจกรรม node2 และเวลาที่สิ้นสุดของกิจกรรม node1 จะเกิดก่อนเวลาที่สิ้นสุดของกิจกรรม node2 สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.8 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.9



รูปที่ 3.8 กฎความสัมพันธ์แบบทับซ้อน (intervalOverlaps)
ที่อธิบายด้วยแผนภาพข่ายงายแบบพีดีเอ็ม

Node(?node1) ^ Node(?node2) ^ earlyStart(?node1, ?es1) ^ earlyStart(?node2, ?es2) ^ swrlb:lessThan(?es1, ?es2) ^ earlyFinish(?node1, ?ef1) ^ swrlb:greaterThan(?ef1, ?es2) earlyFinish(?node2, ?ef2) ^ swrlb:lessThan(?ef1, ?ef2) ⟶ intervalOverlaps(?node1, ?node2) **ชื่อความสัมพันธ์**: intervalOverlaps

วัตถุประสงค์ของกฎ: เพื่ออธิบายความสัมพันธ์ที่มีช่วงระยะเวลาเวลาใดเวลาหนึ่ง ที่ทั้งสองกิจกรรมดำเนินการไปพร้อมกัน

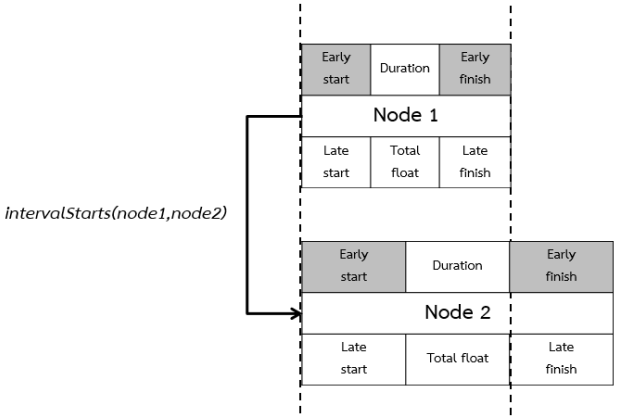
คำอธิบาย: ถ้า (?node1) คือค่าตัวแปรของกิจกรรมที่เป็นสมาชิกของคลาส Node และ (?node2) คือค่าตัวแปรของกิจกรรมที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีค่า earlyStart ที่แทนด้วยตัวแปร (?es1) และกิจกรรม (?node2) มีค่า earlyStart ที่แทนด้วยตัวแปร (?es2) โดยที่ค่าของตัวแปร (?es1) น้อยกว่า ค่าของตัวแปร (?es2) และกิจกรรม (?node1) มีค่า earlyFinish แทนด้วยตัวแปร (?ef1) โดยที่ค่าของตัวแปร (?ef1) มากกว่าค่าของตัวแปร (?es2) และกิจกรรม (?node2) มีค่า earlyFinish แทนด้วยตัวแปร (?ef2) โดยที่ค่าของตัวแปร (?ef1) น้อยกว่าค่าของตัวแปร (?ef2) แล้วจะอนุมานได้ว่า กิจกรรม (?node1) มีความสัมพันธ์แบบทับซ้อนกับกิจกรรม (?node2)

ชื่อความสัมพันธ์ผกผัน (InverseOf): intervalOverlappedBy

รูปที่ 3.9 กฎความสัมพันธ์แบบทับซ้อน (intervalOverlaps) ที่อธิบายด้วยภาษา SWRL

3.2.4 กฎความสัมพันธ์แบบเริ่มต้น (intervalStarts)

จากคำนิยามของกฎความสัมพันธ์แบบทับซ้อนที่กล่าวว่า ถ้ากิจกรรม node1 มีความสัมพันธ์แบบเริ่มต้นกับกิจกรรม node2 แล้ว ดังนั้นเวลาที่เริ่มต้นของกิจกรรม node1 จะเท่ากับเวลาที่เริ่มต้นของกิจกรรม node2 และเวลาที่สิ้นสุดของกิจกรรม node1 จะเกิดก่อนเวลาที่สิ้นสุดของกิจกรรม node2 สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.10 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.11



รูปที่ 3.10 กฎความสัมพันธ์แบบเริ่มต้น (intervalStarts)
ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็ม

Node(?node1) ^ Node(?node2) ^
earlyStart(?node1, ?es1) ^ earlyStart(?node2, ?es2) ^ swrlb:equal(?es1, ?es2) ^
earlyFinish(?node1, ?ef1) ^ earlyFinish(?node2, ?ef2) ^ swrlb:lessThan(?ef1, ?ef2)
⟶ intervalStarts(?node1, ?node2)

ชื่อความสัมพันธ์: intervalStarts

วัตถุประสงค์ของกฎ: เพื่ออธิบายความสัมพันธ์ที่กิจกรรมทั้งสองเริ่มต้นพร้อมกัน

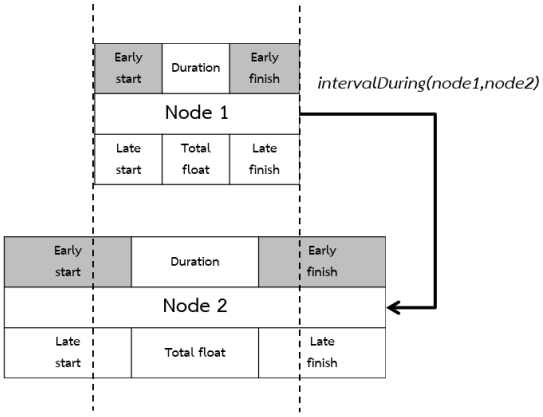
คำอธิบาย: ถ้า (?node1) คือค่าตัวแปรของกิจกรรมที่เป็นสมาชิกของคลาส Node และ (?node2) คือค่าตัวแปรของกิจกรรมที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีค่า earlyStart ที่แทนด้วยตัวแปร (?es1) และกิจกรรม (?node2) มีค่า earlyStart ที่แทนด้วยตัวแปร (?es2) โดยที่ค่าของตัวแปร (?es1) เท่ากับ ค่าของตัวแปร (?es2) และกิจกรรม (?node1) มีค่า earlyFinish แทนด้วยตัวแปร (?ef1) และกิจกรรม (?node2) มีค่า earlyFinish แทนด้วยตัวแปร (?ef2) โดยที่ค่าของตัวแปร (?ef1) น้อยกว่าค่าของตัวแปร (?ef2) แล้วจะอนุมานได้ว่า กิจกรรม (?node1) มีความสัมพันธ์แบบเริ่มต้นกับกิจกรรม (?node2)

ชื่อความสัมพันธ์ผกผัน (InverseOf): intervalStartedBy

รูปที่ 3.11 กฎความสัมพันธ์แบบเริ่มต้น (intervalStarts) ที่อธิบายด้วยภาษา SWRL

3.2.5 กฎความสัมพันธ์แบบท่ามกลาง (intervalDuring)

จากคำนิยามของกฎความสัมพันธ์แบบท่ามกลางที่กล่าวว่า ถ้ากิจกรรม node1 มีความสัมพันธ์แบบท่ามกลางกับกิจกรรม node2 แล้ว ดังนั้นเวลาที่เริ่มต้นของกิจกรรม node1 จะเกิดหลังเวลาที่เริ่มต้นของกิจกรรม node2 และเวลาที่สิ้นสุดของกิจกรรม node 1 จะเกิดก่อนเวลาที่สิ้นสุดของกิจกรรม node2 สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.12 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.13



รูปที่ 3.12 กฎความสัมพันธ์แบบท่ามกลาง (intervalDuring)
ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็ม

Node(?node1) ^ Node(?node2) ^ earlyStart(?node1, ?es1) ^
earlyStart(?node2, ?es2) ^ swrlb:greaterThan(?es1, ?es2) ^
earlyFinish(?node1, ?ef1) ^ earlyFinish(?node2, ?ef2) ^
swrlb:lessThan(?ef1, ?ef2) ⟶ intervalDuring(?node1, ?node2)

ชื่อความสัมพันธ์: intervalDuring

วัตถุประสงค์ของกฎ: เพื่ออธิบายความสัมพันธ์ที่กิจกรรมที่ตามหลังมีระยะเวลาการทำงานครอบคลุมระยะการทำงานของกิจกรรมแรก

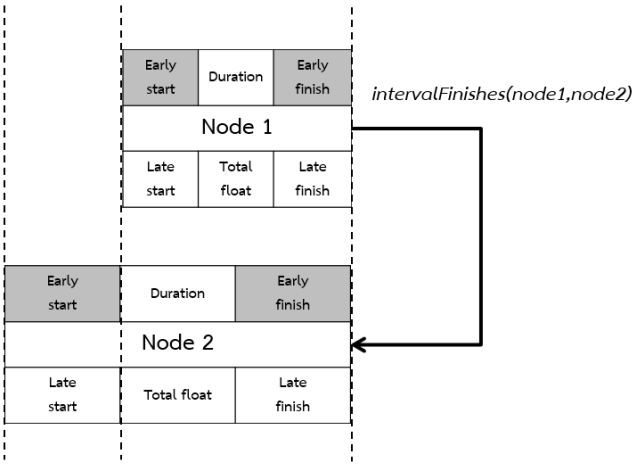
คำอธิบาย: ถ้า (?node1) คือค่าตัวแปรของกิจกรรมที่เป็นสมาชิกของคลาส Node และ (?node2) คือค่าตัวแปรของกิจกรรมที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีค่า earlyStart ที่แทนด้วยตัวแปร (?es1) และกิจกรรม (?node2) มีค่า earlyStart ที่แทนด้วยตัวแปร (?es2) โดยที่ค่าของตัวแปร (?es1) มากกว่า ค่าของตัวแปร (?es2) และกิจกรรม (?node1) มีค่า earlyFinish แทนด้วยตัวแปร (?ef1) และกิจกรรม (?node2) มีค่า earlyFinish แทนด้วยตัวแปร (?ef2) โดยที่ค่าของตัวแปร (?ef1) น้อยกว่า ค่าของตัวแปร (?ef2) แล้วจะอนุมานได้ว่า กิจกรรม (?node1) มีความสัมพันธ์แบบท่ามกลางกับกิจกรรม (?node2)

ชื่อความสัมพันธ์ผกผัน (InverseOf): intervalContains

รูปที่ 3.13 กฎความสัมพันธ์แบบท่ามกลาง (intervalDuring) ที่อธิบายด้วยภาษา SWRL

3.2.6 กฎความสัมพันธ์แบบสิ้นสุด (intervalFinishes)

จากคำนิยามของกฎความสัมพันธ์แบบสิ้นสุดที่กล่าวว่า ถ้ากิจกรรม node1 มีความสัมพันธ์แบบสิ้นสุดกับกิจกรรม node2 แล้ว ดังนั้นเวลาที่เริ่มต้นของกิจกรรม node1 จะเกิดหลังเวลาที่เริ่มต้นของกิจกรรม node2 และเวลาที่สิ้นสุดของกิจกรรม node1 จะเท่ากับเวลาที่สิ้นสุดของกิจกรรม node2 สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.14 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.15



รูปที่ 3.14 กฎความสัมพันธ์แบบสิ้นสุด (intervalFinishes)
ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็ม

Node(?node1) ^ Node(?node2) ^ earlyStart(?node1, ?es1) ^
earlyStart(?node2, ?es2) ^ swrlb:greaterThan(?es1, ?es2) ^
earlyFinish(?node1, ?ef1) ^ earlyFinish(?node2, ?ef2) ^
swrlb:equal(?ef1, ?ef2) ⟶ intervalFinishes(?node1, ?node2)

ชื่อความสัมพันธ์: intervalFinishes

วัตถุประสงค์ของกฎ: เพื่ออธิบายความสัมพันธ์ที่กิจกรรมทั้งสองสิ้นสุดพร้อมกัน

คำอธิบาย: ถ้า (?node1) คือค่าตัวแปรของกิจกรรมที่เป็นสมาชิกของคลาส Node และ (?node2) คือค่าตัวแปรของกิจกรรมที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีค่า earlyStart ที่แทนด้วยตัวแปร (?es1) และกิจกรรม (?node2) มีค่า earlyStart ที่แทนด้วยตัวแปร (?es2) โดยที่ค่าของตัวแปร (?es1) มากกว่า ค่าของตัวแปร (?es2) และกิจกรรม (?node1) มีค่า earlyFinish แทนด้วยตัวแปร (?ef1) และกิจกรรม (?node2) มีค่า earlyFinish แทนด้วยตัวแปร (?ef2) โดยที่

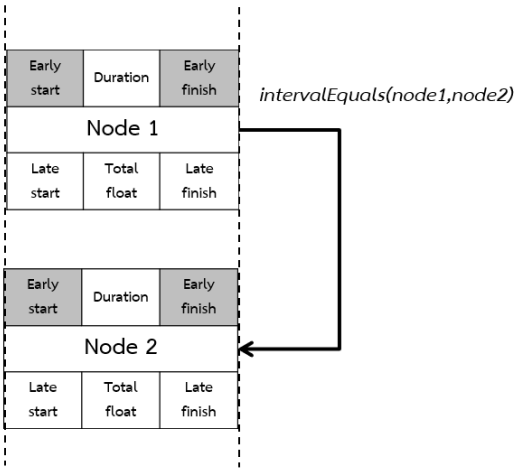
ค่าของตัวแปร (?ef1) เท่ากับ ค่าของตัวแปร (?ef2) แล้วจะอนุมานได้ว่า กิจกรรม (?node1) มีความสัมพันธ์แบบสิ้นสุดกับกิจกรรม (?node2)

ชื่อความสัมพันธ์ผกผัน (InverseOf): intervalFinishedBy

รูปที่ 3.15 กฎความสัมพันธ์แบบสิ้นสุด (intervalFinishes) ที่อธิบายด้วยภาษา SWRL

3.2.7 กฎความสัมพันธ์แบบเท่ากัน (intervalEquals)

จากคำนิยามของกฎความสัมพันธ์แบบเท่ากันที่กล่าวว่า ถ้ากิจกรรม node1 มีความสัมพันธ์แบบสิ้นสุดกับกิจกรรม node2 แล้ว ดังนั้นเวลาที่เริ่มต้นของกิจกรรม node1 จะเท่ากับเวลาที่เริ่มต้นของกิจกรรม node2 และเวลาที่สิ้นสุดของกิจกรรม node1 จะเท่ากับเวลาที่สิ้นสุดของกิจกรรม node2 สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.16 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.17



รูปที่ 3.16 กฎความสัมพันธ์แบบเท่ากัน (intervalEquals)
ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็ม

Node(?node1) ^ Node(?node2) ^ earlyStart(?node1, ?es1) ^
earlyStart(?node2, ?es2) ^ swrlb:equal(?es1, ?es2) ^
earlyFinish(?node1, ?ef1) ^ earlyFinish(?node2, ?ef2) ^
swrlb:equal(?ef1, ?ef2) ⟶ intervalEquals(?node1, ?node2)

ชื่อความสัมพันธ์: intervalEquals

วัตถุประสงค์ของกฎ: เพื่ออธิบายความสัมพันธ์ที่กิจกรรมทั้งสองเริ่มต้นและสิ้นสุดพร้อมกัน

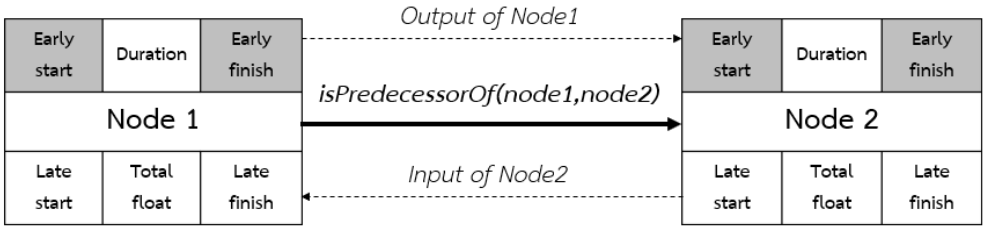
คำอธิบาย: ถ้า (?node1) คือค่าตัวแปรของกิจกรรมที่เป็นสมาชิกของคลาส Node และ (?node2) คือค่าตัวแปรของกิจกรรมที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีค่า earlyStart ที่แทนด้วยตัวแปร (?es1) และกิจกรรม (?node2) มีค่า earlyStart ที่แทนด้วยตัวแปร (?es2) โดยที่ค่าของตัวแปร (?es1) เท่ากับ ค่าของตัวแปร (?es2) และกิจกรรม (?node1) มีค่า earlyFinish แทนด้วยตัวแปร (?ef1) และกิจกรรม (?node2) มีค่า earlyFinish แทนด้วยตัวแปร (?ef2) โดยที่ค่าของตัวแปร (?ef1) เท่ากับ ค่าของตัวแปร (?ef2) แล้วจะอนุมานได้ว่า กิจกรรม (?node1) มีความสัมพันธ์แบบเท่ากับกับกิจกรรม (?node2)

ชื่อความสัมพันธ์ผกผัน (InverseOf): ไม่มี

รูปที่ 3.17 กฎความสัมพันธ์แบบเท่ากัน (intervalEquals) ที่อธิบายด้วยภาษา SWRL

3.2.8 กฎความสัมพันธ์แบบลำดับก่อนหน้า (isPredecessorOf)

นอกจากความสัมพันธ์ของช่วงเวลาตามทฤษฎีของ Allen แล้ว ลำดับการเกิดของกิจกรรมตามนิยามของแผนภาพข่ายงานแบบพีดีเอ็มก็มีความสำคัญเช่นกัน เพื่อบอกลำดับการเกิดกิจกรรมระหว่าง 2 กิจกรรมว่ากิจกรรมใดเกิดก่อนหรือหลังกิจกรรมใด สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.18 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.19



รูปที่ 3.18 กฎความสัมพันธ์แบบลำดับก่อนหน้า (isPredecessorOf)
ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็ม

Node(?node1) ^ Node(?node2) ^ hasOutputEdge(?node1, ?edge1) ^
hasInputEdge(?node2, ?edge2) ^ sameAs(?edge1, ?edge2) ⟶
isPredecessorOf(?node1, ?node2)

ชื่อความสัมพันธ์: isPredecessorOf

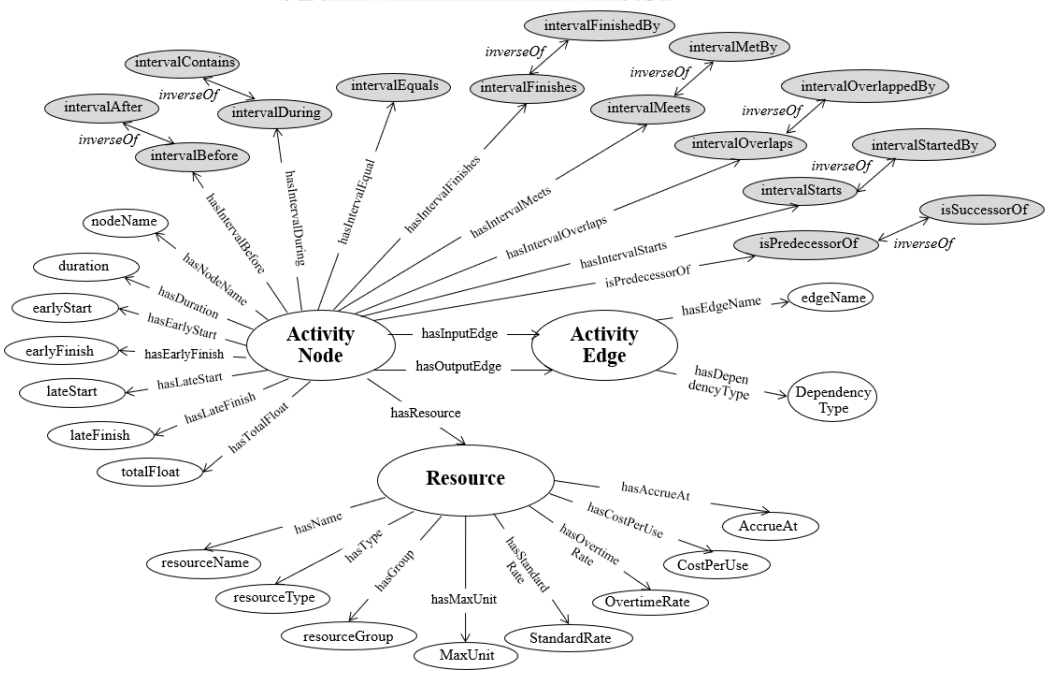
วัตถุประสงค์ของกฎ: เพื่ออธิบายความสัมพันธ์ของลำดับกิจกรรม ที่เกิดก่อนหน้าและเกิดตามหลัง

คำอธิบาย: ถ้า (?node1) คือค่าตัวแปรของกิจกรรมที่เป็นสมาชิกของคลาส Node และ (?node2) คือค่าตัวแปรของกิจกรรมที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีความสัมพันธ์แบบมีเส้นเชื่อมที่วิ่งออก (hasOutputEdge) เป็น (?edge1) และกิจกรรม (?node2) มีความสัมพันธ์แบบมีเส้นเชื่อมที่วิ่งเข้า (hasInputEdge) เป็น (?edge2) และทั้ง (?edge1) และ (?edge2) เป็นเส้นเชื่อมเดียวกัน (sameAs) แล้วจะอนุมานได้ว่า กิจกรรม (?node1) มีความสัมพันธ์แบบลำดับก่อนหน้ากับกิจกรรม (?node2)

ชื่อความสัมพันธ์ผกผัน (InverseOf): isSuccessorOf

รูปที่ 3.19 กฎความสัมพันธ์แบบลำดับก่อนหน้า (isPredecessorOf) ที่อธิบายด้วยภาษา SWRL

เมื่อมีการอนุมานกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้น ทำให้เกิดความสัมพันธ์รูปแบบใหม่หรือองค์ความรู้ใหม่เพิ่มขึ้นมาจากเดิมซึ่งเป็นข้อดีของออนโทโลยี ดังนั้นแบบโครงร่างออนโทโลยีของแผนภาพข่ายงานแบบพีดีเอ็มต้นแบบที่ออกแบบได้ในขั้นตอนที่ 3.1.3 จึงต้องมีการปรับปรุงเพิ่มเติมเพื่อรองรับความสัมพันธ์ใหม่ที่เกิดขึ้น โดยการเพิ่มความสัมพันธ์ของช่วงเวลาตามทฤษฎีของ Allen ดังนั้นโครงร่างออนโทโลยีใหม่จะแสดงดังรูปที่ 3.20 โดยที่แต่ละความสัมพันธ์ที่เพิ่มขึ้นมาจะมีความสัมพันธ์ในลักษณะผกผัน (InverseOf) ยกเว้นความสัมพันธ์แบบเท่ากัน (intervalEquals) ที่จะมีความสัมพันธ์ในลักษณะสมมาตร (Symmetric)



รูปที่ 3.20 แบบจำลองโครงร่างออนโทโลยีของแผนภาพข่ายงานแบบพีดีเอ็ม
ที่แสดงความสัมพันธ์ของช่วงเวลา

3.3 การออกแบบชุดกฎเพื่อตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม

แบบจำลองออนโทโลยีของแผนภาพข่ายงานแบบพีดีเอ็มที่ได้จากการออกแบบในขั้นตอนที่ 3.1.3 นั้น ประกอบไปด้วย 3 แนวคิดหลัก ได้แก่ แนวคิดเกี่ยวกับกิจกรรมในแต่ละโหนด ซึ่งถูกอธิบายอยู่ภายใต้คลาส ActivityNode แนวคิดเกี่ยวกับความสัมพันธ์หรือความพึ่งพากัน (Dependency) ของแต่ละกิจกรรม ซึ่งถูกอธิบายอยู่ภายใต้คลาส ActivityEdge และสุดท้ายคือแนวคิดเกี่ยวกับทรัพยากรซึ่งในงานวิจัยนี้จะเน้นเฉพาะทรัพยากรบุคคลเป็นหลัก ซึ่งถูกอธิบายอยู่ภายใต้คลาส Resource ดังนั้นในการตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม ก็จะทำการตรวจสอบความต้องกันเชิงความหมายภายใต้ 3 แนวคิดดังกล่าว โดยมีจำนวนกฎดังนี้

- กฎตรวจสอบความต้องกันของโหนดกิจกรรม จำนวน 6 กฎ
- กฎการตรวจสอบการพึ่งพาที่ต้องกัน จำนวน 13 กฎ
- กฎการตรวจสอบการพึ่งพาที่ไม่ต้องกัน จำนวน 12 กฎ
- กฎการตรวจสอบการจัดสรรทรัพยากร จำนวน 2 กฎ

โดยจะใช้ภาษาเอสคิวดับเบิลยูอาร์แอล (SQWRL) ในการอธิบายชุดกฎตามที่ได้ออกแบบไว้ อีกทั้งยังสามารถทำการสืบค้นรูปแบบที่ไม่ต้องกันของแผนภาพข่ายงานแบบพีดีเอ็มได้อีกด้วย

3.3.1 กฎตรวจสอบความต้องกันของโหนดกิจกรรม

จากนิยามของโหนดกิจกรรมที่แสดงบนแผนภาพข่ายงานแบบพีดีเอ็มในรูปที่ 2.1 ตามที่ได้อธิบายในรายละเอียดในเนื้อหาบทที่ 2 นั้น พบว่ามีความเกี่ยวข้องกับเรื่องช่วงของเวลาที่จำเป็นจะต้องเกิดขึ้นอย่างเป็นลำดับก่อนและหลัง ซึ่งข้อมูลช่วงเวลาที่จะนำมาพิจารณาเพื่อตรวจสอบความต้องกันสามารถแบ่งเป็นชุดกฎทั้งหมด 6 ข้อดังแสดงในตารางที่ 3.2

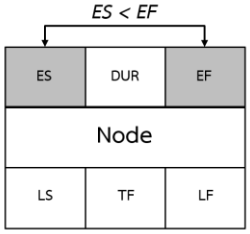
ตารางที่ 3.2 แสดงกฎตรวจสอบความต้องกันของโหนดกิจกรรมและวัตถุประสงค์ของกฎ

ลำดับ	กฎ	วัตถุประสงค์ของกฎ
1.	กฎตรวจสอบวันที่เริ่มต้นและวันที่สิ้นสุดที่เร็วที่สุด (ES_EF_Consistency)	เพื่อตรวจสอบว่าวันที่เริ่มต้นที่เร็วที่สุดของแต่ละกิจกรรม จะต้องเกิดก่อน วันที่สิ้นสุดที่เร็วที่สุด
2.	กฎตรวจสอบวันที่เริ่มต้นและวันที่สิ้นสุดที่ช้าที่สุด (LS_LF_Consistency)	เพื่อตรวจสอบว่าวันที่เริ่มต้นที่ช้าที่สุดของแต่ละกิจกรรม จะต้องเกิดก่อน วันที่สิ้นสุดที่ช้าที่สุด

ลำดับ	กฎ	วัตถุประสงค์ของกฎ
3.	กฎตรวจสอบวันที่เริ่มต้นที่เร็วที่สุดและวันที่เริ่มต้นที่ช้าที่สุด (ES_LS_Consistency)	เพื่อตรวจสอบว่าวันที่เริ่มต้นที่เร็วที่สุดของแต่ละกิจกรรม จะต้องเกิดก่อนหรือเกิดพร้อมกัน กับวันที่เริ่มต้นที่ช้าที่สุด
4.	กฎตรวจสอบวันที่สิ้นสุดที่เร็วที่สุดและวันที่สิ้นสุดที่ช้าที่สุด (EF_LF_Consistency)	เพื่อตรวจสอบว่าวันที่สิ้นสุดที่เร็วที่สุดของแต่ละกิจกรรม จะต้องเกิดก่อนหรือเกิดพร้อมกัน กับวันที่สิ้นสุดที่ช้าที่สุด
5.	กฎตรวจสอบระยะเวลา (DUR_Consistency)	เพื่อตรวจสอบว่าระยะเวลาในการดำเนินงานของแต่ละกิจกรรมจะต้องมีอย่างน้อย 1 วัน และจะต้องเท่ากับ ผลต่างของวันที่สิ้นสุดที่เร็วที่สุดกับวันที่เริ่มต้นที่เร็วที่สุด และจะต้องเท่ากับ ผลต่างของวันที่วันที่สิ้นสุดที่ช้าที่สุดกับวันที่สิ้นสุดที่เร็วที่สุด
6.	กฎตรวจสอบเวลาลอยรวม (TF_Consistency)	เพื่อตรวจสอบเวลาลอยรวมของแต่ละกิจกรรม ที่จะต้องเท่ากับ ผลต่างของวันที่เริ่มต้นที่ช้าที่สุดกับวันที่เริ่มต้นที่เร็วที่สุด และจะต้องเท่ากับ ผลต่างของวันที่สิ้นสุดที่ช้าที่สุดกับวันที่เริ่มต้นที่ช้าที่สุด

3.3.1.1 กฎตรวจสอบวันที่เริ่มต้นและวันที่สิ้นสุดที่เร็วที่สุด (ES_EF_Consistency)

จากเงื่อนไขของแผนภาพข่ายงานแบบพีดีเอ็ม เวลาเริ่มต้นที่เร็วที่สุดของกิจกรรมใด ๆ จะต้องน้อยกว่าเวลาสิ้นสุดที่เร็วที่สุดของกิจกรรมนั้น ๆ สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.21 และสามารถอธิบายเป็นภาษากฎเอสคิวดับเบิลยูอาร์แอลได้ดังรูปที่ 3.22



รูปที่ 3.21 กฎตรวจสอบวันที่เริ่มต้นและวันที่สิ้นสุดที่เร็วที่สุด (ES_EF_Consistency) ที่อธิบายด้วยแผนภายข่ายงานแบบพีดีเอ็ม

Node(?node) ^ earlyStart(?node, ?es) ^ earlyFinish(?node, ?ef) ^
swrlb:lessThan(?es, ?ef) ^ Result(ES_EF_Consistency) -> sqwrl:select(?node)

ชื่อกฎ: กฎตรวจสอบวันที่เริ่มต้นและวันที่สิ้นสุดที่เร็วที่สุด (ES_EF_Consistency)

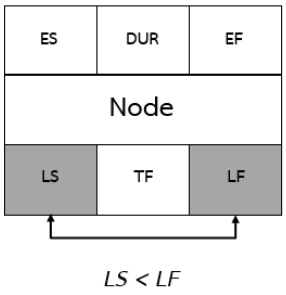
วัตถุประสงค์ของกฎ: เพื่อตรวจสอบว่าวันที่เริ่มต้นที่เร็วที่สุดของแต่ละกิจกรรม จะต้องเกิดก่อนวันที่สิ้นสุดที่เร็วที่สุด

คำอธิบาย: ถ้า (?node) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node) มีค่า earlyStart ที่แทนด้วยตัวแปร (?es) และกิจกรรม (?node) มีค่า earlyFinish ที่แทนด้วยตัวแปร (?ef) โดยที่ค่าของตัวแปร (?es) น้อยกว่า ค่าของตัวแปร (ef) แล้วให้ค้นคืนค่าผลลัพธ์ของการตรวจสอบกิจกรรมตามเงื่อนไขดังกล่าว

รูปที่ 3.22 กฎตรวจสอบวันที่เริ่มต้นและวันที่สิ้นสุดที่เร็วที่สุด (ES_EF_Consistency) ที่อธิบายด้วยภาษา SQWRL

3.3.1.2 กฎตรวจสอบวันที่เริ่มต้นและวันที่สิ้นสุดที่ช้าที่สุด (LS_LF_Consistency)

จากเงื่อนไขของแผนภาพข่ายงานแบบพีดีเอ็ม เวลาเริ่มต้นที่ช้าที่สุดของกิจกรรมใด ๆ จะต้องน้อยกว่าเวลาสิ้นสุดที่ช้าที่สุดของกิจกรรมนั้น ๆ สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.23 และสามารถอธิบายเป็นภาษากฎเอสคิวดับเบิลยูอาร์แอลได้ดังรูปที่ 3.24



รูปที่ 3.23 กฎตรวจสอบวันที่เริ่มต้นและวันที่สิ้นสุดที่ช้าที่สุด (LS_LF_Consistency) ที่อธิบายด้วยแผนภายข่ายงานแบบพีดีเอ็ม

Node(?node) ^ lateStart(?node, ?ls) ^ lateFinish(?node, ?lf) ^
swrlb:lessThan(?ls, ?lf) ^ Result(LS_LF_Consistency) -> sqwrl:select(?node)

ชื่อกฎ: **กฎตรวจสอบวันที่เริ่มต้นและวันที่สิ้นสุดที่ช้าที่สุด** (LS_LF_Consistency)

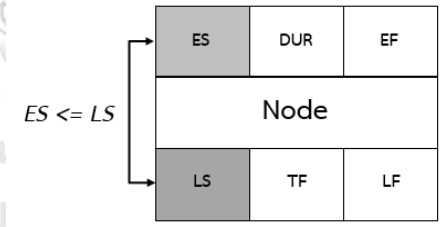
วัตถุประสงค์ของกฎ: เพื่อตรวจสอบว่าวันที่เริ่มต้นที่ช้าที่สุดของแต่ละกิจกรรม จะต้องเกิดก่อนวันที่สิ้นสุดที่ช้าที่สุด

คำอธิบาย: ถ้า (?node) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node) มีค่า lateStart ที่แทนด้วยตัวแปร (?ls) และกิจกรรม (?node) มีค่า lateFinish ที่แทนด้วยตัวแปร (?lf) โดยที่ค่าของตัวแปร (?ls) น้อยกว่า ค่าของตัวแปร (lf) แล้วให้ค้นคืนค่าผลลัพธ์ของการตรวจสอบกิจกรรมตามเงื่อนไขดังกล่าว

รูปที่ 3.24 กฎตรวจสอบวันที่เริ่มต้นและวันที่สิ้นสุดที่เร็วที่สุด (LS_LF_Consistency) ที่อธิบายด้วยภาษา SQWRL

3.3.1.3 กฎตรวจสอบวันที่เริ่มต้นที่เร็วที่สุดและวันที่เริ่มต้นที่ช้าที่สุด (ES_LS_Consistency)

จากเงื่อนไขของแผนภาพข่ายงานแบบพีดีเอ็ม เวลาเริ่มต้นที่เร็วที่สุดของกิจกรรมใด ๆ จะต้องน้อยกว่าเวลาเริ่มต้นที่ช้าที่สุดของกิจกรรมนั้น ๆ สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.25 และสามารถอธิบายเป็นภาษากฎเอสคิวดับเบิลยูอาร์แอลได้ดังรูปที่ 3.26



รูปที่ 3.25 กฎตรวจสอบวันที่เริ่มต้นที่เร็วที่สุดและวันที่เริ่มต้นที่ช้าที่สุด (ES_LS_Consistency) ที่อธิบายด้วยแผนภายข่ายงานแบบพีดีเอ็ม

Node(?node) ^ earlyStart(?node, ?es) ^ lateStart(?node, ?ls) ^
swrlb:lessThanOrEqual(?es, ?ls) ^ Result(ES_LS_Consistency) -> sqwrl:select(?node)

ชื่อกฎ: กฎตรวจสอบวันที่เริ่มต้นที่เร็วที่สุดและวันที่เริ่มต้นที่ช้าที่สุด (ES_LS_Consistency)

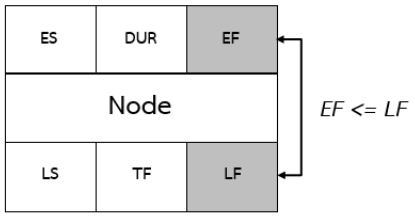
วัตถุประสงค์ของกฎ: เพื่อตรวจสอบว่าวันที่เริ่มต้นที่เร็วที่สุดของแต่ละกิจกรรม จะต้องเกิดก่อนหรือเกิดพร้อมกัน กับวันที่เริ่มต้นที่ช้าที่สุด

คำอธิบาย: ถ้า (?node) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node) มีค่า earlyStart ที่แทนด้วยตัวแปร (?es) และกิจกรรม (?node) มีค่า lateStart ที่แทนด้วยตัวแปร (?ls) โดยที่ค่าของตัวแปร (?es) น้อยกว่าหรือเท่ากับ ค่าของตัวแปร (ls) แล้วให้ค้นคืนค่าผลลัพธ์ของการตรวจสอบกิจกรรมตามเงื่อนไขดังกล่าว

รูปที่ 3.26 กฎตรวจสอบวันที่เริ่มต้นที่เร็วที่สุดและวันที่เริ่มต้นที่เร็วที่สุด (ES_LS_Consistency) ที่อธิบายด้วยภาษา SQWRL

3.3.1.4 กฎตรวจสอบวันที่สิ้นสุดที่เร็วที่สุดและวันที่สิ้นสุดที่ช้าที่สุด (EF_LF_Consistency)

จากเงื่อนไขของแผนภาพข่ายงานแบบพีดีเอ็ม เวลาสิ้นสุดที่เร็วที่สุดของกิจกรรมใด ๆ จะต้องน้อยกว่าหรือเท่ากับเวลาสิ้นสุดที่ช้าที่สุดของกิจกรรมนั้น ๆ สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.27 และสามารถอธิบายเป็นภาษากฎเอสคิวดับเบิลยูอาร์แอลได้ดังรูปที่ 3.28



รูปที่ 3.27 กฎตรวจสอบวันที่สิ้นสุดที่เร็วที่สุดและวันที่สิ้นสุดที่ช้าที่สุด (EF_LF_Consistency) ที่อธิบายด้วยแผนภายข่ายงานแบบพีดีเอ็ม

Node(?node) ^ earlyFinish(?node, ?ef) ^ lateFinish(?node, ?lf) ^ swrlb:lessThanOrEqual(?ef, ?lf) ^ Result(EF_LF_Consistency) -> sqwrl:select(?node)

ชื่อกฎ: กฎตรวจสอบวันที่เริ่มต้นที่เร็วที่สุดและวันที่เริ่มต้นที่ช้าที่สุด (ES_LS_Consistency)

วัตถุประสงค์ของกฎ: เพื่อตรวจสอบว่าวันที่สิ้นสุดที่เร็วที่สุดของแต่ละกิจกรรม จะต้องเกิดก่อนหรือเกิดพร้อมกัน กับวันที่สิ้นสุดที่ช้าที่สุด

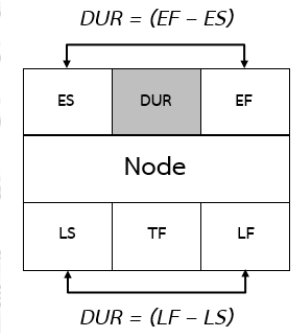
คำอธิบาย: ถ้า (?node) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node) มีค่า earlyFinish ที่แทนด้วยตัวแปร (?ef) และกิจกรรม (?node) มีค่า

lateFinish ที่แทนด้วยตัวแปร (?lf) โดยที่ค่าของตัวแปร (?ef) น้อยกว่าหรือเท่ากับ ค่าของตัวแปร (lf) แล้วให้ค้นคืนค่าผลลัพธ์ของการตรวจสอบกิจกรรมตามเงื่อนไขดังกล่าว

รูปที่ 3.28 กฎตรวจสอบวันที่เริ่มต้นที่เร็วที่สุดและวันที่เริ่มต้นที่เร็วที่สุด (EF_LF_Consistency) ที่อธิบายด้วยภาษา SQWRL

3.3.1.5 กฎตรวจสอบระยะเวลา (DUR_Consistency)

จากเงื่อนไขของแผนภาพข่ายงานแบบพีดีเอ็ม ระยะเวลาของการดำเนินงานกิจกรรมสามารถคำนวณได้จากผลต่างของวันที่สิ้นสุดที่เร็วที่สุด (early finish) กับวันที่เริ่มต้นที่เร็วที่สุด (early start) ซึ่งจะต้องเท่ากับการคำนวณจากผลต่างของวันที่สิ้นสุดที่ช้าที่สุด (late finish) กับที่เริ่มต้นที่ช้าที่สุด (late start) สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.29 และสามารถอธิบายเป็นภาษากฎเอสคิวดับเบิลยูอาร์แอลได้ดังรูปที่ 3.30



รูปที่ 3.29 กฎตรวจสอบระยะเวลา (DUR_Consistency) ที่อธิบายด้วยแผนภายข่ายงานแบบพีดีเอ็ม

Node(?node) ^ earlyStart(?node, ?es) ^ earlyFinish(?node, ?ef) ^ lateStart(?node, ?ls) ^ lateFinish(?node, ?lf) ^ duration(?node, ?dur) ^ swrlb:subtract(?dur1, ?ef, ?es) ^ swrlb:subtract(?dur2, ?lf, ?ls) ^ swrlb:equal(?dur1, ?dur2) ^ swrlb:equal(?dur, ?dur1) ^ swrlb:equal(?dur, ?dur2) ^ Result(DUR_Consistency) -> sqwrl:select(?node)
ชื่อกฎ: กฎตรวจสอบระยะเวลา (DUR_Consistency)

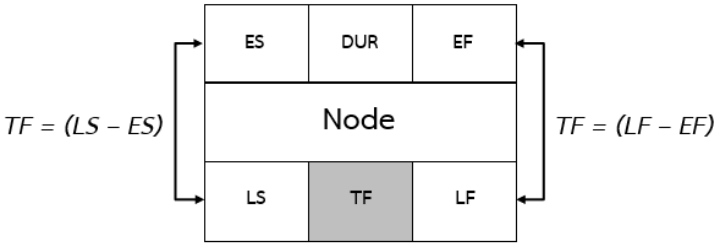
วัตถุประสงค์ของกฎ: เพื่อตรวจสอบว่าระยะเวลาในการดำเนินงานของแต่ละกิจกรรมจะต้องมีอย่างน้อย 1 วัน และจะต้องเท่ากับ ผลต่างของวันที่สิ้นสุดที่เร็วที่สุดกับวันที่เริ่มต้นที่เร็วที่สุด และจะต้องเท่ากับ ผลต่างของวันที่วันที่สิ้นสุดที่ช้าที่สุดกับวันที่สิ้นสุดที่เร็วที่สุด

คำอธิบาย: ถ้า (?node) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node) มีค่า earlyStart ที่แทนด้วยตัวแปร (?es) และกิจกรรม (?node) มีค่า earlyFinish ที่แทนด้วยตัวแปร (?ef) และกิจกรรม (?node) มีค่า lateStart ที่แทนด้วยตัวแปร (?ls) และกิจกรรม (?node) มีค่า lateFinish ที่แทนด้วยตัวแปร (?lf) และกิจกรรม (?node) มีค่า duration ที่แทนด้วยตัวแปร (?dur) และค่าของตัวแปร (?dur1) คำนวณมาจาก earlyFinish ซึ่งแทนค่าด้วยตัวแปร (?ef) ลบด้วย earlyStart ซึ่งแทนค่าด้วยตัวแปร (?es) และค่าของตัวแปร (?dur2) คำนวณมาจาก lateFinish ซึ่งแทนค่าด้วยตัวแปร (?lf) ลบด้วย lateStart ซึ่งแทนค่าด้วยตัวแปร (?ls) และค่าของตัวแปร (?dur1) เท่ากับ (?dur2) และค่าของตัวแปร (?dur) เท่ากับ (?dur1) และค่าของตัวแปร (?dur) เท่ากับ (?dur2) แล้วให้ค้นคืนค่าผลลัพธ์ของการตรวจสอบกิจกรรมตามเงื่อนไขดังกล่าว

รูปที่ 3.30 กฎตรวจสอบระยะเวลา (DUR_Consistency) ที่อธิบายด้วยภาษา SQWRL

3.3.1.6 กฎตรวจสอบเวลาลอยรวม (TF_Consistency)

จากเงื่อนไขของแผนภาพข่ายงานแบบพีดีเอ็ม เวลาลอยรวมสามารถคำนวณได้จากผลต่างของวันที่เริ่มต้นที่ช้าที่สุด (late start) กับวันที่เริ่มต้นที่เร็วที่สุด (early start) ซึ่งจะต้องเท่ากับการคำนวณผลต่างของวันที่สิ้นสุดที่ช้าที่สุด (late finish) กับวันที่สิ้นสุดที่เร็วที่สุด (early finish) สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.31 และสามารถอธิบายเป็นภาษากฎเอสคิวดับเบิลยูอาร์แอลได้ดังรูปที่ 3.32



รูปที่ 3.31 กฎตรวจสอบเวลาลอยรวม (TF_Consistency)
ที่อธิบายด้วยแผนภายข่ายงานแบบพีดีเอ็ม

Node(?node) ^ earlyStart(?node, ?es) ^ earlyFinish(?node, ?ef) ^
lateStart(?node, ?ls) ^ lateFinish(?node, ?lf) ^ totalFloat(?node, ?tf) ^
swrlb:subtract(?tf1, ?ls, ?es) ^ swrlb:subtract(?tf2, ?lf, ?ef) ^
swrlb:equal(?tf1, ?tf2) ^ swrlb:equal(?tf, ?tf1) ^
swrlb:equal(?tf, ?tf2) ^ Result(TF_Consistency) -> sqwrl:select(?node)

ชื่อกฎ: กฎตรวจสอบเวลาลอยรวม (TF_Consistency)

วัตถุประสงค์ของกฎ: เพื่อตรวจสอบเวลาลอยรวมของแต่ละกิจกรรม ที่จะต้องเท่ากับ ผลต่างของวันที่เริ่มต้นที่ช้าที่สุดกับวันที่เริ่มต้นที่เร็วที่สุด และจะต้องเท่ากับ ผลต่างของวันที่สิ้นสุดที่ช้าที่สุดกับวันที่เริ่มต้นที่ช้าที่สุด

คำอธิบาย: ถ้า (?node) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node) มีค่า earlyStart ที่แทนด้วยตัวแปร (?es) และกิจกรรม (?node) มีค่า earlyFinish ที่แทนด้วยตัวแปร (?ef) และกิจกรรม (?node) มีค่า lateStart ที่แทนด้วยตัวแปร (?ls) และกิจกรรม (?node) มีค่า lateFinish ที่แทนด้วยตัวแปร (?lf) และกิจกรรม (?node) มีค่า totalFloat ที่แทนด้วยตัวแปร (?tf) และค่าของตัวแปร (?tf1) คำนวณมาจาก lateStart ซึ่งแทนค่าด้วยตัวแปร (?ls) ลบด้วย earlyStart ซึ่งแทนค่าด้วยตัวแปร (?es) และค่าของตัวแปร (?tf2) คำนวณมาจาก lateFinish ซึ่งแทนค่าด้วยตัวแปร (?lf) ลบด้วย earlyFinish ซึ่งแทนค่าด้วยตัวแปร (?ef) และค่าของตัวแปร (?tf1) เท่ากับ (?tf2) และค่าของตัวแปร (?tf) เท่ากับ (?tf1) และค่าของตัวแปร (?tf) เท่ากับ (?tf2) แล้วให้ค้นคืนค่าผลลัพธ์ของการตรวจสอบกิจกรรมตามเงื่อนไขดังกล่าว

รูปที่ 3.32 กฎตรวจสอบเวลาลอยรวม (TF_Consistency)ที่อธิบายด้วยภาษา SQWRL

3.3.2 กฎตรวจสอบความต้องกันของการพึ่งพาของกิจกรรม

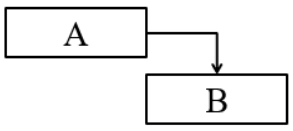
รูปแบบของความสัมพันธ์หรือการพึ่งพา (Dependency) ของแผนภาพข่ายงานแบบพีดีเอ็มมีอยู่ด้วยกัน 4 รูปแบบคือ การพึ่งพาแบบเริ่มต้น-สิ้นสุด (Finish-to-Start : FS) การพึ่งพาแบบเริ่มต้น-เริ่มต้น (Start-to-Start : SS) การพึ่งพาแบบสิ้นสุด-สิ้นสุด (Finish-to-Finish : FF) และความสัมพันธ์แบบเริ่มต้น-สิ้นสุด (Start-to-Finish) ดังนั้น ในการออกแบบกฎเพื่อตรวจสอบความต้องกัน ก็จะทำการตรวจสอบการพึ่งพาของกิจกรรมทั้ง 4 รูปแบบดังกล่าว

3.3.2.1 กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้น (FS)

รูปแบบทั่วไปของการพึ่งพาแบบสิ้นสุด-เริ่มต้น จะแสดงดังรูปที่ 3.33 คือเมื่อกิจกรรมก่อนหน้าสิ้นสุดลง จึงจะเริ่มต้นกิจกรรมถัดไปได้ ซึ่งเมื่อทำการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าตรงกับความสัมพันธ์แบบประชิด (intervalMeets)

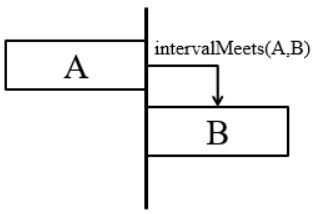
แต่ในทางปฏิบัติพบว่ากิจกรรมที่ตามหลังอาจจะเริ่มต้นก่อนที่กิจกรรมก่อนหน้าจะสิ้นสุด หรือในระหว่างที่กิจกรรมก่อนหน้ากำลังดำเนินการ เรียกการพึ่งพาลักษณะนี้ว่าการสิ้นสุด-เริ่มต้นแบบมีเวลานำ (Finish-to-Start with Lead) ซึ่งมักจะเกิดขึ้นในกรณีที่มีความจำเป็นในการเร่งการวันที่แล้วเสร็จของโครงการให้ไวขึ้น ในทางกลับกันก็อาจจะมีการล่าช้าในการเริ่มต้นของกิจกรรมที่ตามหลัง เรียกการพึ่งพาลักษณะนี้ว่าการสิ้นสุดเริ่มต้นแบบมีเวลารอคอย (Finish-to-Start with Lag) ซึ่งมักจะเกิดขึ้นในกรณีที่ไม่สามารถเริ่มต้นกิจกรรมได้ตามแผน

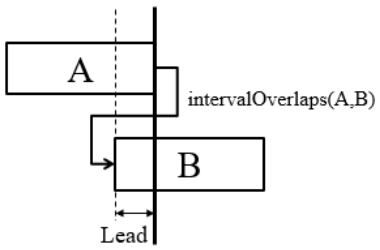
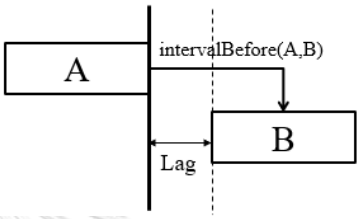
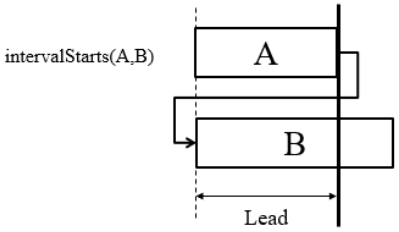
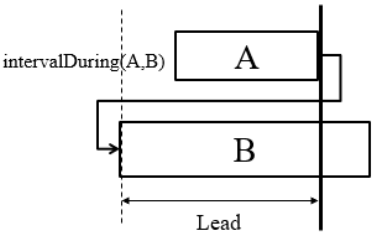
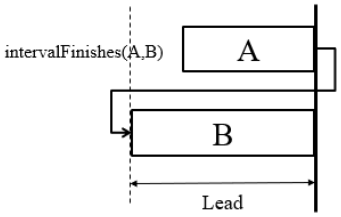
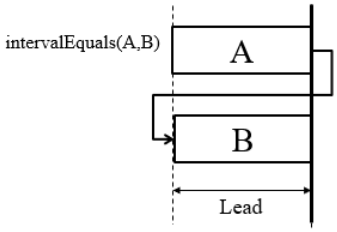
การพึ่งพาแบบมีเวลานำและเวลารอคอยนั้น เมื่อทำการอนุมานด้วยกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าตรงกับความสัมพันธ์แบบทับซ้อน (intervalOverlaps) และความสัมพันธ์แบบก่อนหน้า (intervalBefore) ตามลำดับ ดังนั้นหากพบว่าความสัมพันธ์ของกิจกรรมคู่ใดที่มีการกำหนดให้มีการพึ่งพาแบบสิ้นสุด-เริ่มต้น (Finish-to-Start : FS) จะต้องถูกอธิบายด้วย 3 ช่วงเวลาตามทฤษฎี Allen ได้แก่ intervalMeets, intervalBefore และ intervalOverlaps จึงจะอนุมานได้ว่ามีความต้องกัน



รูปที่ 3.33 รูปแบบทั่วไปของการพึ่งพาแบบสิ้นสุด-เริ่มต้น

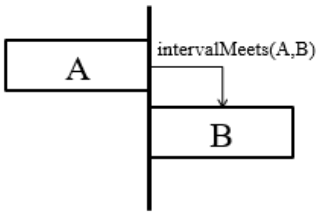
ตารางที่ 3.3 แสดงกฎการตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้น

ลำดับ	กฎ	ทฤษฎีช่วงเวลาของ Allen	มีความต้องกัน
1.	กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบทั่วไปที่ต้องกัน (FS_Consistency)		✓

ลำดับ	กฎ	ทฤษฎีช่วงเวลาของ Allen	มีความต้องกัน
2.	กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบมีเวลานำที่ต้องกัน (FS_Lead_Consistency)		✓
3.	กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบมีเวลารอคอยที่ต้องกัน (FS_Lag_Consistency)		✓
4.	กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้นที่ไม่ต้องกันแบบที่ 1 (FS_ Inconsistency1)		✗
5.	กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้นที่ไม่ต้องกันแบบที่ 2 (FS_Inconsistency2)		✗
6.	กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้นที่ไม่ต้องกันแบบที่ 3 (FS_Inconsistency3)		✗
7.	กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้นที่ไม่ต้องกันแบบที่ 4 (FS_Inconsistency4)		✗

3.3.2.1.1 กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบทั่วไปที่ต้องกัน (FS_Consistency)

จากการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าการพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบทั่วไปจะตรงกับความสัมพันธ์แบบประชิด (intervalMeets) ดังนั้นหากการพึ่งพาของกิจกรรมเป็นแบบสิ้นสุด-เริ่มต้น และมีความสัมพันธ์แบบประชิด ก็จะสามารถสรุปได้ว่ามีความต้องกัน สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.34 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.35



รูปที่ 3.34 การพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบทั่วไปที่ต้องกัน (FS_Consistency) ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็มและทฤษฎีช่วงเวลาของ Allen

Node(?node1) ^ Node(?node2) ^ nodeName(?node1, ?name1) ^ nodeName(?node2, ?name2) ^ isPredecessorOf(?node1, ?node2) ^ intervalMeets(?node1, ?node2) ^ Edge(?edge1) ^ Edge(?edge2) ^ hasOutputEdge(?node1, ?edge1) ^ hasInputEdge(?node2, ?edge2) ^ sameAs(?edge1, ?edge2) ^ dependencyType(?edge1, "FS") -> sqwrl:select(?node1, ?edge1, ?node2)

ชื่อกฎ: กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบทั่วไปที่ต้องกัน (FS_Consistency)

วัตถุประสงค์ของกฎ: เพื่อตรวจสอบความต้องกันของความสัมพันธ์ของกิจกรรมที่อยู่ก่อนหน้าและกิจกรรมที่ตามหลัง ที่มีการพึ่งพาแบบสิ้นสุด-เริ่มต้นในรูปแบบทั่วไปที่ต้องกัน

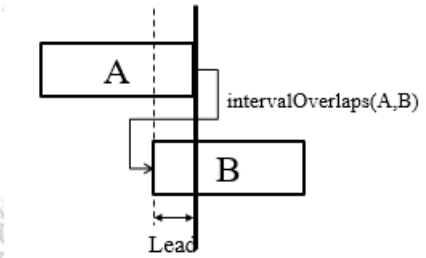
คำอธิบาย: ถ้า (?node1) และ (?node2) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีความสัมพันธ์แบบลำดับก่อนหน้ากับกิจกรรม (?node2) และกิจกรรม (?node1) มีความสัมพันธ์แบบประชิดหน้ากับกิจกรรม (?node2) และ (?edge1) และ (?edge2) คือค่าของตัวแปรของเส้นเชื่อมใด ๆ ที่เป็นสมาชิกของคลาส Edge และกิจกรรม (?node1) มีเส้นเชื่อมที่วิ่งออกเป็น (?edge1) และกิจกรรม (?node2) มีเส้นเชื่อมที่วิ่งเข้าเป็น

(?edge2) และเส้นเชื่อม (?edge1) และ (?edge2) เป็นเส้นเชื่อมเดียวกัน และเส้นเชื่อม (?edge1) มีประเภทการพึ่งพามีค่าเป็น “FS” แล้วสรุปได้ว่าการพึ่งพาแบบสิ้นสุด-เริ่มต้นมีความต้องกัน

รูปที่ 3.35 กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบทั่วไปที่ต้องกัน (FS_Consistency) ที่อธิบายด้วยภาษา SWRL

3.3.2.1.2 กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบมีเวลานำ (FS_Lead_Consistency)

จากการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าการพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบที่มีเวลานำ (Lead) จะตรงกับความสัมพันธ์แบบทับซ้อน (intervalOverlaps) ดังนั้นหากการพึ่งพาของกิจกรรมเป็นแบบสิ้นสุด-เริ่มต้นแบบที่มีเวลานำและมีความสัมพันธ์แบบทับซ้อน ก็จะสามารถสรุปได้ว่ามีความต้องกัน สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.36 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.37



รูปที่ 3.36 การพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบมีเวลานำที่ต้องกัน (FS_Lead_Consistency) ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็มและทฤษฎีช่วงเวลาของ Allen

Node(?node1) ^ Node(?node2) ^ nodeName(?node1, ?name1) ^
nodeName(?node2, ?name2) ^ isPredecessorOf(?node1, ?node2) ^
intervalOverlaps(?node1, ?node2) ^ Edge(?edge1) ^ Edge(?edge2) ^
hasOutputEdge(?node1, ?edge1) ^ hasInputEdge(?node2, ?edge2) ^
sameAs(?edge1, ?edge2) ^ dependencyType(?edge1, "FS") ->
sqwrl:select(?node1, ?edge1, ?node2)

ชื่อกฎ: กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบมีเวลานำที่ต้องกัน (FS_Lead_Consistency)

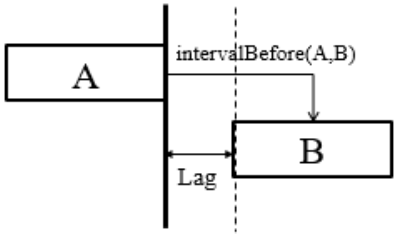
วัตถุประสงค์ของกฎ: เพื่อตรวจสอบความต้องกันของความสัมพันธ์ของกิจกรรมที่อยู่ก่อนหน้าและกิจกรรมที่ตามหลัง ที่มีการพึ่งพาแบบสิ้นสุด-เริ่มต้นในแบบมีเวลานำที่ต้องกัน

คำอธิบาย: ถ้า (?node1) และ (?node2) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีความสัมพันธ์แบบลำดับก่อนหน้ากับกิจกรรม (?node2) และกิจกรรม (?node1) มีความสัมพันธ์แบบทับซ้อนหน้ากับกิจกรรม (?node2) และ (?edge1) และ (?edge2) คือค่าของตัวแปรของเส้นเชื่อมใด ๆ ที่เป็นสมาชิกของคลาส Edge และกิจกรรม (?node1) มีเส้นเชื่อมที่วิ่งออกเป็น (?edge1) และกิจกรรม (?node2) มีเส้นเชื่อมที่วิ่งเข้าเป็น (?edge2) และเส้นเชื่อม (?edge1) และ (?edge2) เป็นเส้นเชื่อมเดียวกัน และเส้นเชื่อม (?edge1) มีประเภทการพึ่งพามีค่าเป็น “FS” แล้วสรุปได้ว่าการพึ่งพาแบบเริ่มต้น-สิ้นสุดแบบมีเวลานำมีความต้องกัน

รูปที่ 3.37 กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบมีเวลานำที่ต้องกัน (FS_Lead_Consistency) ที่อธิบายด้วยภาษา SWRL

3.3.2.1.3 กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบมีเวลารอคอย (FS_Lag_Consistency)

จากการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าการพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบที่มีเวลารอคอย (Lag) จะตรงกับความสัมพันธ์แบบก่อนหน้า (intervalBefore) ดังนั้นหากการพึ่งพาของกิจกรรมเป็นแบบสิ้นสุด-เริ่มต้นแบบที่มีเวลารอคอย และมีความสัมพันธ์แบบก่อนหน้า ก็จะสามารถสรุปได้ว่ามีความต้องกัน สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.38 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.39



รูปที่ 3.38 การพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบมีเวลารอคอยที่ต้องกัน (FS_Lag_Consistency) ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็มและทฤษฎีช่วงเวลาของ Allen

Node(?node1) ^ Node(?node2) ^ nodeName(?node1, ?name1) ^ nodeName(?node2, ?name2) ^ isPredecessorOf(?node1, ?node2) ^ intervalBefore(?node1, ?node2) ^ Edge(?edge1) ^ Edge(?edge2) ^ hasOutputEdge(?node1, ?edge1) ^ hasInputEdge(?node2, ?edge2) ^ sameAs(?edge1, ?edge2) ^ dependencyType(?edge1, "FS") -> sqwrl:select(?node1, ?edge1, ?node2)

ชื่อกฎ: กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบมีเวลารอคอยที่ต้องกัน (FS_Lag_Consistency)

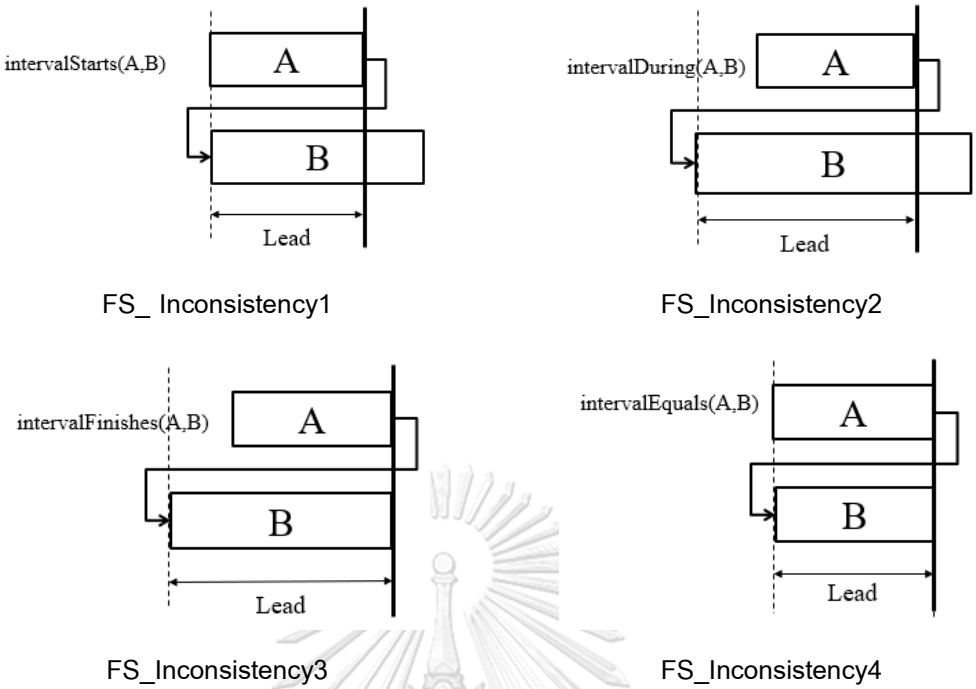
วัตถุประสงค์ของกฎ: เพื่อตรวจสอบความต้องกันของความสัมพันธ์ของกิจกรรมที่อยู่ก่อนหน้าและกิจกรรมที่ตามหลัง ที่มีการพึ่งพาแบบสิ้นสุด-เริ่มต้นในแบบมีเวลาตามที่ต้องกัน

คำอธิบาย: ถ้า (?node1) และ (?node2) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีความสัมพันธ์แบบลำดับก่อนหน้ากับกิจกรรม (?node2) และกิจกรรม (?node1) มีความสัมพันธ์แบบก่อนหน้าหน้ากับกิจกรรม (?node2) และ (?edge1) และ (?edge2) คือค่าของตัวแปรของเส้นเชื่อมใด ๆ ที่เป็นสมาชิกของคลาส Edge และกิจกรรม (?node1) มีเส้นเชื่อมที่วิ่งออกเป็น (?edge1) และกิจกรรม (?node2) มีเส้นเชื่อมที่วิ่งเข้าเป็น (?edge2) และเส้นเชื่อม (?edge1) และ (?edge2) เป็นเส้นเชื่อมเดียวกัน และเส้นเชื่อม (?edge1) มีประเภทการพึ่งพามีค่าเป็น “FS” แล้วสรุปได้ว่าการพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบมีเวลารอคอยมีความต้องกัน

รูปที่ 3.39 กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบมีเวลารอคอยที่ต้องกัน (FS_Lag_Consistency)

3.3.2.1.4 กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้นที่ไม่ต้องกัน (FS_ Inconsistency)

จากการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าการพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบทั่วไปจะตรงกับความสัมพันธ์แบบประชิด (intervalMeets) หรือหากมีเวลานำ (Lead) จะมีความสัมพันธ์แบบทับซ้อน (intervalOverlaps) หรือหากมีเวลารอคอย (Lag) จะมีความสัมพันธ์แบบก่อนหน้า (intervalBefore) ดังนั้นหากการพึ่งพาของกิจกรรมแบบสิ้นสุด-เริ่มต้นมีความสัมพันธ์ของช่วงเวลาใน 3 รูปแบบข้างต้นก็จะสรุปได้ว่ามีความต้องกัน ในทางตรงกันข้ามหากกิจกรรมใด ๆ มีความสัมพันธ์ในรูปแบบอื่น ๆ นอกเหนือจากนี้ก็จะสรุปได้ว่าไม่ต้องกัน ซึ่งกรณีของการไม่ต้องกันสามารถเป็นได้ 4 รูปแบบตามรูปที่ 3.40



รูปที่ 3.40 การพึ่งพาแบบสิ้นสุด-เริ่มต้นที่ไม่ต้องกันในรูปแบบต่าง ๆ
ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็มและทฤษฎีช่วงเวลาของ Allen

3.3.2.2 กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-เริ่มต้น (SS)

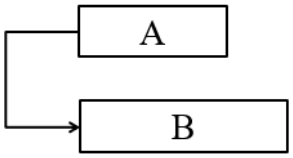
รูปแบบทั่วไปของการพึ่งพาแบบเริ่มต้น-เริ่มต้น จะแสดงดังรูปที่ 3.41 คือเมื่อกิจกรรมก่อนหน้าและกิจกรรมที่ตามหลังเริ่มต้นพร้อมกัน ซึ่งเมื่อแปลงให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าตรงกับความสัมพันธ์แบบเริ่มต้น (intervalStarts) และความสัมพันธ์แบบเท่ากัน (intervalEquals) ตามลำดับ

แต่ในทางปฏิบัติพบว่ากิจกรรมที่ตามหลังอาจจะไม่สามารถเริ่มต้นได้พร้อมกันกับกิจกรรมก่อนหน้า เรียกการพึ่งพาลักษณะนี้ว่าการเริ่มต้น-เริ่มต้นแบบมีเวลารอคอย (Start-to-Start with Lag) ซึ่งมักจะเกิดขึ้นในกรณีที่ไม่สามารถเริ่มต้นกิจกรรมได้ตามแผน การพึ่งพาแบบมีเวลารอคอยนั้นเมื่อทำการอนุมานด้วยกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าตรงกับความสัมพันธ์แบบทับซ้อน (intervalOverlaps)

แต่หากมีการขยายเวลารอคอย (Lag) ในการเริ่มต้นของกิจกรรมที่ตามหลัง จนกิจกรรมก่อนหน้าแล้วเสร็จ ดังรูปลำดับที่ 4 และ 5 ในตารางที่ 3.4 ซึ่งเมื่อแปลงให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าตรงกับความสัมพันธ์แบบก่อนหน้า (intervalBefore) และความสัมพันธ์แบบประชิด (intervalMeets) ตามลำดับ จะพบว่ามีความไม่ต้องกันเชิงความหมายของการเริ่มต้น

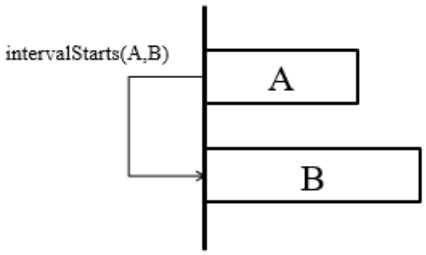
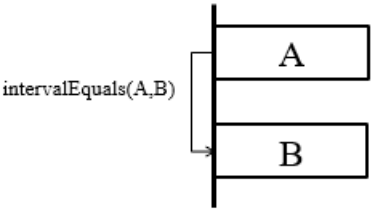
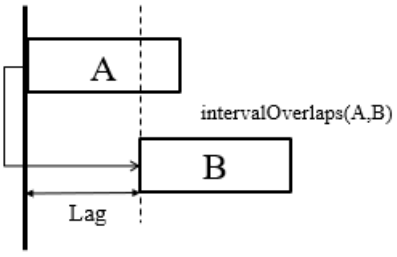
กิจกรรมพร้อมกัน ดังนั้นหากพบว่าความสัมพันธ์ของกิจกรรมคู่ใดที่มีการพึ่งพาแบบเริ่มต้น-เริ่มต้น (Start-to-Start) แต่มีความสัมพันธ์แบบก่อนหน้าและประชิด จะอนุมานได้ว่ามีความไม่ต้องกัน

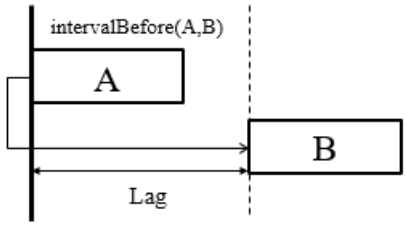
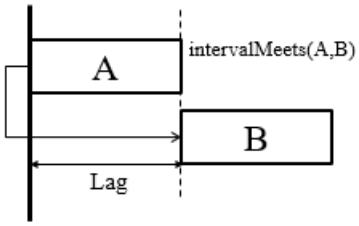
เนื่องจากการพึ่งพาแบบเริ่มต้น-เริ่มต้น ไม่มีการนิยามเวลานำ (Lead) ดังนั้นจึงไม่นำมาพิจารณาในการตรวจสอบความต้องกัน



รูปที่ 3.41 รูปแบบทั่วไปของการพึ่งพาแบบเริ่มต้น-เริ่มต้น

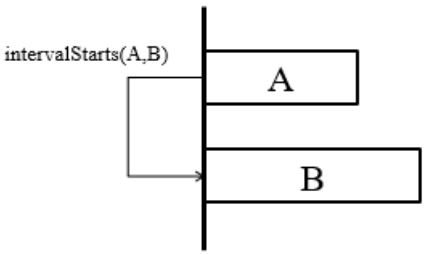
ตารางที่ 3.4 กฎการตรวจสอบการพึ่งพาแบบเริ่มต้น-เริ่มต้น

ลำดับ	กฎ	ทฤษฎีช่วงเวลาของ Allen	มีความต้องกัน
1.	กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-เริ่มต้น แบบที่ต้องกัน (SS_Consistency1)		✓
2.	กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-เริ่มต้นแบบที่ต้องกัน (SS_Consistency2)		✓
3.	กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-เริ่มต้น แบบมีเวลารอคอยที่ต้องกัน (SS_Lag_Consistency)		✓

ลำดับ	กฎ	ทฤษฎีช่วงเวลาของ Allen	มีความต้องกัน
4.	กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-เริ่มต้นที่ไม่ต้องกันแบบที่ 1 (SS_ Inconsistency1)		x
5.	กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-เริ่มต้นที่ไม่ต้องกันแบบที่ 2 (SS_Inconsistency2)		x

3.3.2.2.1 กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-เริ่มต้นแบบที่ต้องกัน (SS_Consistency1)

จากการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าการพึ่งพาแบบเริ่มต้น-เริ่มต้นแบบทั่วไปจะตรงกับความสัมพันธ์แบบเริ่มต้น (intervalStarts) ดังนั้นหากการพึ่งพาของกิจกรรมเป็นแบบเริ่มต้น-เริ่มต้น ก็จะสามารถสรุปได้ว่ามีความต้องกัน สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.42 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.43



รูปที่ 3.42 การพึ่งพาแบบเริ่มต้น-เริ่มต้นที่ต้องกัน (SS_Consistency1)
ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็มและทฤษฎีช่วงเวลาของ Allen

```
Node(?node1) ^ Node(?node2) ^ nodeName(?node1, ?name1) ^
nodeName(?node2, ?name2) ^ isPredecessorOf(?node1, ?node2) ^
```

intervalStarts(?node1, ?node2) ^ Edge(?edge1) ^ Edge(?edge2) ^
hasOutputEdge(?node1, ?edge1) ^ hasInputEdge(?node2, ?edge2) ^ sameAs(?edge1, ?edge2) ^ dependencyType(?edge1, "SS") ->
sqwrl:select(?node1, ?name1, ?edge1, ?node2, ?name2)

ชื่อกฎ: กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-เริ่มต้นแบบที่ต้องกัน (SS_Consistency1)

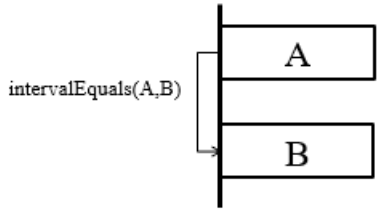
วัตถุประสงค์ของกฎ: เพื่อตรวจสอบความต้องกันของความสัมพันธ์ของกิจกรรมที่อยู่ก่อนหน้าและกิจกรรมที่ตามหลัง ที่มีการพึ่งพาแบบเริ่มต้น-เริ่มต้นที่ต้องกัน

คำอธิบาย: ถ้า (?node1) และ (?node2) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีความสัมพันธ์แบบลำดับก่อนหน้ากับกิจกรรม (?node2)และกิจกรรม (?node1) มีความสัมพันธ์แบบเริ่มต้นกับกิจกรรม (?node2)และ (?edge1) และ (?edge2) คือค่าของตัวแปรของเส้นเชื่อมใด ๆ ที่เป็นสมาชิกของคลาส Edge และกิจกรรม (?node1) มีเส้นเชื่อมที่วิ่งออกเป็น (?edge1) และกิจกรรม (?node2) มีเส้นเชื่อมที่วิ่งเข้าเป็น (?edge2) และเส้นเชื่อม (?edge1) และ (?edge2) เป็นเส้นเชื่อมเดียวกัน และเส้นเชื่อม (?edge1) มีประเภทการพึ่งพา มีค่าเป็น “SS” แล้วสรุปได้ว่าการพึ่งพาแบบเริ่มต้น-เริ่มต้นมีความต้องกัน

รูปที่ 3.43 กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-เริ่มต้นที่ต้องกัน (SS_Consistency) ที่อธิบายด้วยภาษา SWRL

3.3.2.2.2 กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-เริ่มต้นแบบที่ต้องกัน (SS_Consistency2)

จากการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าการพึ่งพาแบบเริ่มต้น-เริ่มต้นจะตรงกับความสัมพันธ์แบบเท่ากัน (intervalEquals) ดังนั้นหากการพึ่งพาของกิจกรรมเป็นแบบเริ่มต้น-เริ่มต้น และมีความสัมพันธ์แบบเท่ากัน ก็จะสามารถสรุปได้ว่ามีความต้องกัน สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.44 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.45



รูปที่ 3.44 การพึ่งพาแบบเริ่มต้น-เริ่มต้นที่ต้องกัน (SS_Consistency2) ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็มและทฤษฎีช่วงเวลาของ Allen

Node(?node1) ^ Node(?node2) ^ nodeName(?node1, ?name1) ^
nodeName(?node2, ?name2) ^ isPredecessorOf(?node1, ?node2) ^
intervalEquals(?node1, ?node2) ^ Edge(?edge1) ^ Edge(?edge2) ^
hasOutputEdge(?node1, ?edge1) ^ hasInputEdge(?node2, ?edge2) ^
sameAs(?edge1, ?edge2) ^ dependencyType(?edge1, "SS") ->
sqwrl:select(?node1, ?edge1, ?node2)

ชื่อกฎ: กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-เริ่มต้นแบบที่ต้องกัน (SS_Consistency2)

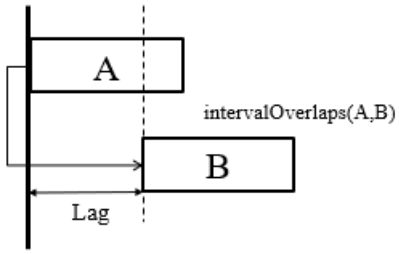
วัตถุประสงค์ของกฎ: เพื่อตรวจสอบความต้องกันของความสัมพันธ์ของกิจกรรมที่อยู่ก่อนหน้าและกิจกรรมที่ตามหลัง ที่มีการพึ่งพาแบบเริ่มต้น-เริ่มต้นที่ต้องกัน

คำอธิบาย: ถ้า (?node1) และ (?node2) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีความสัมพันธ์แบบลำดับก่อนหน้ากับกิจกรรม (?node2) และกิจกรรม (?node1) มีความสัมพันธ์แบบเท่ากันกับกิจกรรม (?node2) และ (?edge1) และ (?edge2) คือค่าของตัวแปรของเส้นเชื่อมใด ๆ ที่เป็นสมาชิกของคลาส Edge และกิจกรรม (?node1) มีเส้นเชื่อมที่วิ่งออกเป็น (?edge1) และกิจกรรม (?node2) มีเส้นเชื่อมที่วิ่งเข้าเป็น (?edge2) และเส้นเชื่อม (?edge1) และ (?edge2) เป็นเส้นเชื่อมเดียวกัน และเส้นเชื่อม (?edge1) มีประเภทการพึ่งพา มีค่าเป็น “SS” แล้วสรุปได้ว่าการพึ่งพาแบบเริ่มต้น-เริ่มต้นมีความต้องกัน

รูปที่ 3.45 กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-เริ่มต้นที่ต้องกัน (SS_Consistency2) ที่อธิบายด้วยภาษา SWRL

3.3.2.2.3 กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-เริ่มต้น แบบมีเวลารอคอยที่ต้องกัน (SS_Lag_Consistency)

จากการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าการพึ่งพาแบบเริ่มต้น-เริ่มต้นแบบมีเวลารอคอย จะตรงกับความสัมพันธ์แบบทับซ้อน (intervalOverlaps) ดังนั้นหากการพึ่งพาของกิจกรรมเป็นแบบเริ่มต้น-เริ่มต้น และมีความสัมพันธ์แบบทับซ้อน ก็จะสามารถสรุปได้ว่ามีความต้องกัน สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.46 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.47



รูปที่ 3.46 การพึ่งพาแบบเริ่มต้น-เริ่มต้นที่ต้องกัน (SS_Lag_Consistency)
ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็มและทฤษฎีช่วงเวลาของ Allen

Node(?node1) ^ Node(?node2) ^ nodeName(?node1, ?name1) ^ nodeName(?node2, ?name2) ^ isPredecessorOf(?node1, ?node2) ^ intervalOverlaps(?node1, ?node2) ^ Edge(?edge1) ^ Edge(?edge2) ^ hasOutputEdge(?node1, ?edge1) ^ hasInputEdge(?node2, ?edge2) ^ sameAs(?edge1, ?edge2) ^ dependencyType(?edge1, "SS") -> sqwrl:select(?node1, ?edge1, ?node2)

ชื่อกฎ: กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-เริ่มต้นแบบที่ต้องกัน (SS_Lag_Consistency)

วัตถุประสงค์ของกฎ: เพื่อตรวจสอบความต้องกันของความสัมพันธ์ของกิจกรรมที่อยู่ก่อนหน้าและกิจกรรมที่ตามหลัง ที่มีการพึ่งพาแบบเริ่มต้น-เริ่มต้นที่ต้องกัน

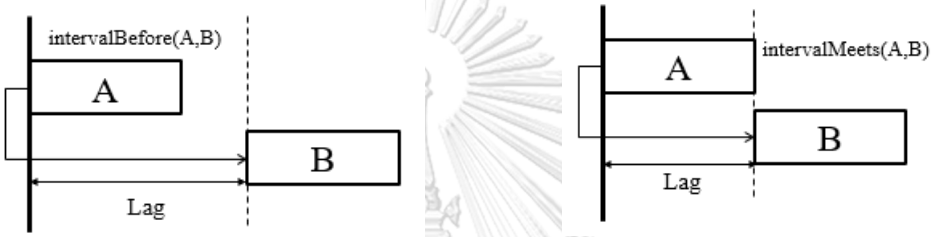
คำอธิบาย: ถ้า (?node1) และ (?node2) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีความสัมพันธ์แบบลำดับก่อนหน้ากับกิจกรรม (?node2) และกิจกรรม (?node1) มีความสัมพันธ์แบบซ้อนทับกับกิจกรรม (?node2)

และ (?edge1) และ (?edge2) คือค่าของตัวแปรของเส้นเชื่อมใด ๆ ที่เป็นสมาชิกของคลาส Edge และกิจกรรม (?node1) มีเส้นเชื่อมที่วิ่งออกเป็น (?edge1) และกิจกรรม (?node2) มีเส้นเชื่อมที่วิ่งเข้าเป็น (?edge2) และเส้นเชื่อม (?edge1) และ (?edge2) เป็นเส้นเชื่อมเดียวกัน และเส้นเชื่อม (?edge1) มีประเภทการพึ่งพามีค่าเป็น “SS” แล้วสรุปได้ว่าการพึ่งพาแบบเริ่มต้น-เริ่มต้นมีความต้องกัน

รูปที่ 3.47 กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-เริ่มต้นที่ต้องกัน (SS_Lag_Consistency)
ที่อธิบายด้วยภาษา SWRL

3.3.2.2.4 กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-เริ่มต้นที่ไม่ต้องกัน (SS_ Inconsistency)

จากการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าการพึ่งพาแบบเริ่มต้น-เริ่มต้นแบบทั่วไปจะตรงกับความสัมพันธ์แบบประชิด (intervalStarts) และความสัมพันธ์แบบเท่ากัน (intervalEquals) หรือหากมีเวลารอคอย (Lag) จะมีความสัมพันธ์แบบก่อนหน้า (intervalOverlaps) ดังนั้นหากการพึ่งพาของกิจกรรมแบบเริ่มต้น-เริ่มต้นมีความสัมพันธ์ของช่วงเวลาใน 3 รูปแบบข้างต้นก็จะสรุปได้ว่ามีความต้องกัน ในทางตรงกันข้ามหากกิจกรรมใด ๆ มีความสัมพันธ์ในรูปแบบอื่น ๆ นอกเหนือจากนี้ก็จะสรุปได้ว่าไม่ต้องกัน ซึ่งกรณีของการไม่ต้องกันสามารถเป็นได้ 2 รูปแบบตามรูปที่ 3.48



SS_ Inconsistency1 SS_ Inconsistency2

รูปที่ 3.48 การพึ่งพาแบบเริ่มต้น-เริ่มต้นที่ไม่ต้องกันในรูปแบบต่าง ๆ ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็มและทฤษฎีช่วงเวลาของ Allen

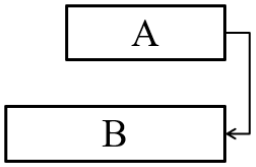
3.3.2.3 กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุด (FF)

รูปแบบทั่วไปของการพึ่งพาแบบสิ้นสุด-สิ้นสุด จะแสดงดังรูปที่ 3.49 คือเมื่อกิจกรรมก่อนหน้าและกิจกรรมที่ตามหลังสิ้นสุดพร้อมกัน ซึ่งเมื่อแปลงให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าตรงกับ ความสัมพันธ์แบบสิ้นสุด (intervalFinishes) และความสัมพันธ์แบบเท่ากัน (intervalEquals) ตามลำดับ

แต่ในทางปฏิบัติพบว่ากิจกรรมที่ตามหลังอาจจะไม่สามารถสิ้นสุดได้พร้อมกันกับกิจกรรมก่อนหน้า เรียกการพึ่งพาลักษณะนี้ว่าการสิ้นสุด-สิ้นสุดแบบมีเวลารอคอย (Finish-to-Finish with Lag) ซึ่งมักจะเกิดขึ้นในกรณีที่ไม่สามารถสิ้นสุดกิจกรรมได้ตามแผน การพึ่งพาแบบมีเวลารอคอยนั้นเมื่อทำการอนุมานด้วยกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าตรงกับความสัมพันธ์แบบทับซ้อน (intervalOverlaps) ความสัมพันธ์แบบเริ่มต้น (intervalStarts) และความสัมพันธ์แบบท่ามกลาง (intervalDuring)

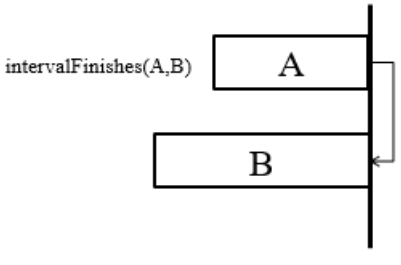
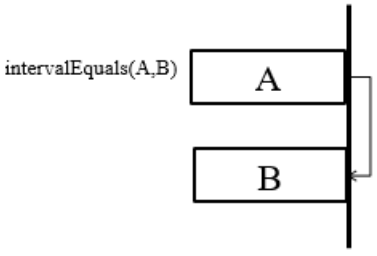
แต่หากมีการขยายเวลารอคอย (Lag) ในการสิ้นสุดของกิจกรรมที่ตามหลัง จนกิจกรรมก่อนหน้าแล้วเสร็จ ดังรูปลำดับที่ 6 และ 7 ในตารางที่ 3.5 ซึ่งเมื่อแปลงให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าตรงกับความสัมพันธ์แบบก่อนหน้า (intervalBefore) และความสัมพันธ์แบบประชิด (intervalMeets) ตามลำดับ จะพบว่ามีความไม่ต้องกันเชิงความหมายของการสิ้นสุดกิจกรรมพร้อมกัน ดังนั้นหากพบว่าความสัมพันธ์ของกิจกรรมคู่ใดที่มีการพึ่งพาแบบสิ้นสุด-สิ้นสุด (Finish-to-Finish) แต่มีความสัมพันธ์แบบก่อนหน้าและประชิด จะอนุมานได้ว่ามีความไม่ต้องกัน

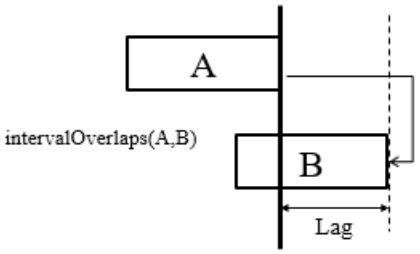
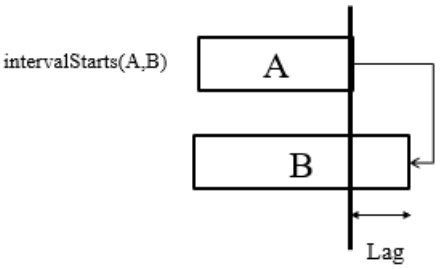
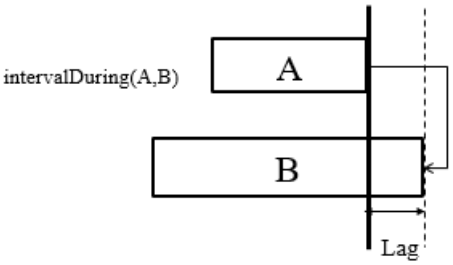
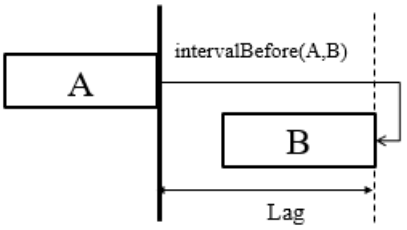
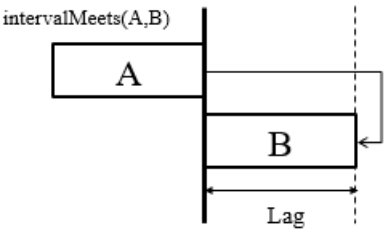
เนื่องจากการพึ่งพาแบบสิ้นสุด-สิ้นสุด ไม่มีการนิยามเวลานำ (Lead) ดังนั้นจึงไม่นำมาพิจารณาในการตรวจสอบความต้องกัน



รูปที่ 3.49 รูปแบบทั่วไปของการพึ่งพาแบบสิ้นสุด-สิ้นสุด

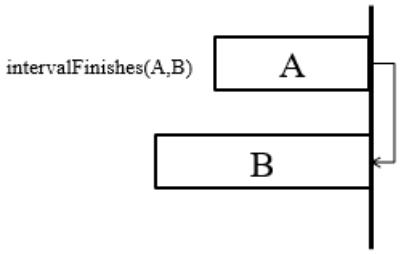
ตารางที่ 3.5 กฎการตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุด

ลำดับ	กฎ	ทฤษฎีช่วงเวลาของ Allen	มีความต้องกัน
1.	กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดแบบที่ต้องกัน (FF_Consistency1)		✓
2.	กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดแบบที่ต้องกัน (FF_Consistency2)		✓

ลำดับ	กฎ	ทฤษฎีช่วงเวลาของ Allen	มีความต้องกัน
3.	กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดแบบมีเวลารอคอยที่ต้องกัน (FF_Lag_Consistency1)		✓
4.	กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดแบบมีเวลารอคอยที่ต้องกัน (FF_Lag_Consistency2)		✓
5.	กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดแบบมีเวลารอคอยที่ต้องกัน (FF_Lag_Consistency3)		✓
6.	กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ไม่ต้องกันแบบที่ 1 (FF_ Inconsistency1)		✗
7.	กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ไม่ต้องกันแบบที่ 2 (FF_Inconsistency2)		✗

3.3.2.3.1 กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดแบบที่ต้องกัน (FF_Consistency1)

จากการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าการพึ่งพาแบบสิ้นสุด-สิ้นสุดแบบทั่วไปจะตรงกับความสัมพันธ์แบบสิ้นสุด (intervalFinishes) ดังนั้นหากการพึ่งพาของกิจกรรมเป็นแบบสิ้นสุด-สิ้นสุด และมีความสัมพันธ์แบบสิ้นสุด ก็จะสามารถสรุปได้ว่ามีความต้องกัน สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.50 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.51



รูปที่ 3.50 การพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ต้องกัน (FF_Consistency1)
ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็มและทฤษฎีช่วงเวลาของ Allen

Node(?node1) ^ Node(?node2) ^ nodeName(?node1, ?name1) ^ nodeName(?node2, ?name2) ^ isPredecessorOf(?node1, ?node2) ^ intervalFinishes(?node1, ?node2) ^ Edge(?edge1) ^ Edge(?edge2) ^ hasOutputEdge(?node1, ?edge1) ^ hasInputEdge(?node2, ?edge2) ^ sameAs(?edge1, ?edge2) ^ dependencyType(?edge1, "FF") -> sqwrl:select(?node1, ?edge1, ?node2)

ชื่อกฎ: กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ต้องกัน (FF_Consistency1)

วัตถุประสงค์ของกฎ: เพื่อตรวจสอบความต้องกันของความสัมพันธ์ของกิจกรรมที่อยู่ก่อนหน้าและกิจกรรมที่ตามหลัง ที่มีการพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ต้องกัน

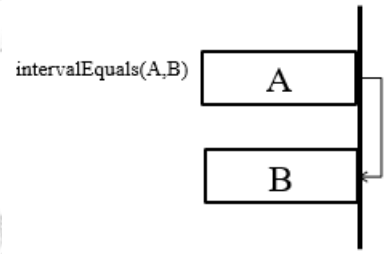
คำอธิบาย: ถ้า (?node1) และ (?node2) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีความสัมพันธ์แบบลำดับก่อนหน้ากับกิจกรรม (?node2)และกิจกรรม (?node1) มีความสัมพันธ์แบบสิ้นสุดกับกิจกรรม (?node2) และ (?edge1) และ (?edge2) คือค่าของตัวแปรของเส้นเชื่อมใด ๆ ที่เป็นสมาชิกของคลาส Edgeและกิจกรรม (?node1) มีเส้นเชื่อมที่วิ่งออกเป็น (?edge1) และกิจกรรม (?node2) มีเส้นเชื่อมที่วิ่งเข้าเป็น

(?edge2) และเส้นเชื่อม (?edge1) และ (?edge2) เป็นเส้นเชื่อมเดียวกันและเส้นเชื่อม (?edge1) มีประเภทการพึ่งพามีค่าเป็น “FF” แล้วสรุปได้ว่าการพึ่งพาแบบสิ้นสุด-สิ้นสุดมีความต้องกัน

รูปที่ 3.51 กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ต้องกัน (FF_Consistency1)
ที่อธิบายด้วยภาษา SWRL

3.3.2.3.2 กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดแบบที่ต้องกัน (FF_Consistency2)

จากการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าการพึ่งพาแบบสิ้นสุด-สิ้นสุดแบบทั่วไปจะตรงกับความสัมพันธ์แบบเท่ากัน (intervalEquals) ดังนั้นหากการพึ่งพาของกิจกรรมเป็นแบบสิ้นสุด-สิ้นสุด และมีความสัมพันธ์แบบเท่ากันก็จะสามารถสรุปได้ว่ามีความต้องกัน สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.52 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.53



รูปที่ 3.52 การพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ต้องกัน (FF_Consistency2)
ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็มและทฤษฎีช่วงเวลาของ Allen

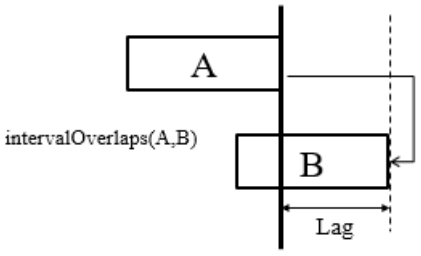
Node(?node1) ^ Node(?node2) ^ nodeName(?node1, ?name1) ^ nodeName(?node2, ?name2) ^ isPredecessorOf(?node1, ?node2) ^ intervalEquals(?node1, ?node2) ^ Edge(?edge1) ^ Edge(?edge2) ^ hasOutputEdge(?node1, ?edge1) ^ hasInputEdge(?node2, ?edge2) ^ sameAs(?edge1, ?edge2) ^ dependencyType(?edge1, "FF") -> sqwrl:select(?node1, ?edge1, ?node2) **ชื่อกฎ:** กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ต้องกัน (FF_Consistency2) **วัตถุประสงค์ของกฎ:** เพื่อตรวจสอบความต้องกันของความสัมพันธ์ของกิจกรรมที่อยู่ก่อนหน้าและกิจกรรมที่ตามหลัง ที่มีการพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ต้องกัน

คำอธิบาย: ถ้า (?node1) และ (?node2) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีความสัมพันธ์แบบลำดับก่อนหน้ากับกิจกรรม (?node2) และกิจกรรม (?node1) มีความสัมพันธ์แบบเท่ากันกับกิจกรรม (?node2) และ (?edge1) และ (?edge2) คือค่าของตัวแปรของเส้นเชื่อมใด ๆ ที่เป็นสมาชิกของคลาส Edge และกิจกรรม (?node1) มีเส้นเชื่อมที่วิ่งออกเป็น (?edge1) และกิจกรรม (?node2) มีเส้นเชื่อมที่วิ่งเข้าเป็น (?edge2) และเส้นเชื่อม (?edge1) และ (?edge2) เป็นเส้นเชื่อมเดียวกัน และเส้นเชื่อม (?edge1) มีประเภทการพึ่งพาเป็นค่าของตัวแปร (?depType1) และค่าตัวแปร (?depType1) มีค่าเป็น “FF” แล้วสรุปได้ว่าการพึ่งพาแบบสิ้นสุด-สิ้นสุดมีความต้องกัน

รูปที่ 3.53 กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ต้องกัน (FF_Consistency2) ที่อธิบายด้วยภาษา SWRL

3.3.2.3.3 กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดแบบมีเวลารอคอยที่ต้องกัน (FF_Lag_Consistency1)

จากการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าการพึ่งพาแบบสิ้นสุด-สิ้นสุดแบบมีเวลารอคอยจะตรงกับความสัมพันธ์แบบทับซ้อน (intervalOverlaps) ดังนั้นหากการพึ่งพาของกิจกรรมเป็นแบบสิ้นสุด-สิ้นสุด และมีความสัมพันธ์แบบทับซ้อนก็จะสามารถสรุปได้ว่ามีความต้องกัน สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.54 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.55



รูปที่ 3.54 การพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ต้องกัน (FF_Lag_Consistency1) ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็มและทฤษฎีช่วงเวลาของ Allen

Node(?node1) ^ Node(?node2) ^ nodeName(?node1, ?name1) ^
nodeName(?node2, ?name2) ^ isPredecessorOf(?node1, ?node2) ^

intervalOverlaps(?node1, ?node2) ^ Edge(?edge1) ^ Edge(?edge2) ^ hasOutputEdge(?node1, ?edge1) ^ hasInputEdge(?node2, ?edge2) ^ sameAs(?edge1, ?edge2) ^ dependencyType(?edge1, "FF") -> sqwrl:select(?node1, ?edge1, ?node2)

ชื่อกฎ: กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ต้องกัน (FF_Lag_Consistency1)

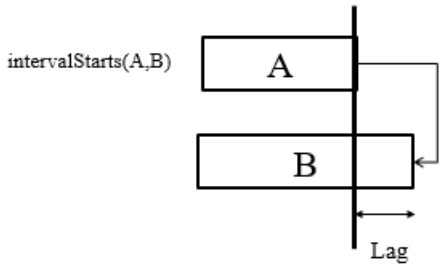
วัตถุประสงค์ของกฎ: เพื่อตรวจสอบความต้องกันของความสัมพันธ์ของกิจกรรมที่อยู่ก่อนหน้าและกิจกรรมที่ตามหลัง ที่มีการพึ่งพาแบบสิ้นสุด-สิ้นสุดแบบมีเวลารคอยที่ต้องกัน

คำอธิบาย: ถ้า (?node1) และ (?node2) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีความสัมพันธ์แบบลำดับก่อนหน้ากับกิจกรรม (?node2)และกิจกรรม (?node1) มีความสัมพันธ์แบบทับซ้อนกับกิจกรรม (?node2) และ (?edge1) และ (?edge2) คือค่าของตัวแปรของเส้นเชื่อมใด ๆ ที่เป็นสมาชิกของคลาส Edge และกิจกรรม (?node1) มีเส้นเชื่อมที่วิ่งออกเป็น (?edge1) และกิจกรรม (?node2) มีเส้นเชื่อมที่วิ่งเข้าเป็น (?edge2) และเส้นเชื่อม (?edge1) และ (?edge2) เป็นเส้นเชื่อมเดียวกัน และเส้นเชื่อม (?edge1) มีประเภทการพึ่งพา มีค่าเป็น “FF” แล้วสรุปได้ว่าการพึ่งพาแบบสิ้นสุด-สิ้นสุดมีความต้องกัน

รูปที่ 3.55 กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ต้องกัน (FF_Lag_Consistency1) ที่อธิบายด้วยภาษา SWRL

3.3.2.3.4 กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดแบบมีเวลารอคอยที่ต้องกัน (FF_Lag_Consistency2)

จากการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าการพึ่งพาแบบสิ้นสุด-สิ้นสุดแบบมีเวลารอคอยจะตรงกับความสัมพันธ์แบบเริ่มต้น (intervalStarts) ดังนั้นหากการพึ่งพาของกิจกรรมเป็นแบบสิ้นสุด-สิ้นสุด และมีความสัมพันธ์แบบเริ่มต้นก็จะสามารถสรุปได้ว่ามีความต้องกัน สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.56 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.57



รูปที่ 3.56 การพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ต้องกัน (FF_Lag_Consistency2)
ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็มและทฤษฎีช่วงเวลาของ Allen

Node(?node1) ^ Node(?node2) ^ nodeName(?node1, ?name1) ^ nodeName(?node2, ?name2) ^ isPredecessorOf(?node1, ?node2) ^ intervalStarts(?node1, ?node2) ^ Edge(?edge1) ^ Edge(?edge2) ^ hasOutputEdge(?node1, ?edge1) ^ hasInputEdge(?node2, ?edge2) ^ sameAs(?edge1, ?edge2) ^ dependencyType(?edge1, "FF") -> sqwrl:select(?node1, ?name1, ?edge1, ?node2, ?name2)

ชื่อกฎ: กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ต้องกัน (FF_Lag_Consistency2)

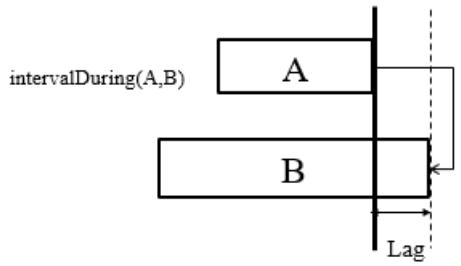
วัตถุประสงค์ของกฎ: เพื่อตรวจสอบความต้องกันของความสัมพันธ์ของกิจกรรมที่อยู่ก่อนหน้าและกิจกรรมที่ตามหลัง ที่มีการพึ่งพาแบบ**สิ้นสุด-สิ้นสุดแบบมีเวลารคอย**ที่ต้องกัน

คำอธิบาย: ถ้า (?node1) และ (?node2) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีความสัมพันธ์แบบลำดับก่อนหน้ากับกิจกรรม (?node2) และกิจกรรม (?node1) มีความสัมพันธ์แบบเริ่มต้นกับกิจกรรม (?node2) และ (?edge1) และ (?edge2) คือค่าของตัวแปรของเส้นเชื่อมใด ๆ ที่เป็นสมาชิกของคลาส Edge และกิจกรรม (?node1) มีเส้นเชื่อมที่วิ่งออกเป็น (?edge1) และกิจกรรม (?node2) มีเส้นเชื่อมที่วิ่งเข้าเป็น (?edge2) และเส้นเชื่อม (?edge1) และ (?edge2) เป็นเส้นเชื่อมเดียวกัน และเส้นเชื่อม (?edge1) มีประเภทการพึ่งพาเป็นค่าของตัวแปร (?depType1) และค่าตัวแปร (?depType1) มีค่าเป็น “FF” แล้วสรุปได้ว่าการพึ่งพาแบบสิ้นสุด-สิ้นสุดมีความต้องกัน

รูปที่ 3.57 กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ต้องกัน (FF_Lag_Consistency2)
ที่อธิบายด้วยภาษา SWRL

3.3.2.3.5 กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดแบบมีเวลารอคอยที่ต้องกัน (FF_Lag_Consistency3)

จากการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าการพึ่งพาแบบสิ้นสุด-สิ้นสุดแบบมีเวลารอคอยจะตรงกับความสัมพันธ์แบบท่ามกลาง (intervalDuring) ดังนั้นหากการพึ่งพาของกิจกรรมเป็นแบบสิ้นสุด-สิ้นสุด และมีความสัมพันธ์แบบท่ามกลางก็จะสามารถสรุปได้ว่ามีความต้องกัน สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.58 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.59



รูปที่ 3.58 การพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ต้องกัน (FF_Lag_Consistency3) ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็มและทฤษฎีช่วงเวลาของ Allen

Node(?node1) ^ Node(?node2) ^ nodeName(?node1, ?name1) ^ nodeName(?node2, ?name2) ^ isPredecessorOf(?node1, ?node2) ^ intervalDuring(?node1, ?node2) ^ Edge(?edge1) ^ Edge(?edge2) ^ hasOutputEdge(?node1, ?edge1) ^ hasInputEdge(?node2, ?edge2) ^ sameAs(?edge1, ?edge2) ^ dependencyType(?edge1, "FF") -> sqwrl:select(?node1,?edge1, ?node2)

ชื่อกฎ: กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ต้องกัน (FF_Lag_Consistency3)

วัตถุประสงค์ของกฎ: เพื่อตรวจสอบความต้องกันของความสัมพันธ์ของกิจกรรมที่อยู่ก่อนหน้าและกิจกรรมที่ตามหลัง ที่มีการพึ่งพาแบบสิ้นสุด-สิ้นสุดแบบมีเวลารคอยที่ต้องกัน

คำอธิบาย: ถ้า (?node1) และ (?node2) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีความสัมพันธ์แบบลำดับก่อนหน้ากับกิจกรรม (?node2) และกิจกรรม (?node1) มีความสัมพันธ์แบบท่ามกลางกับกิจกรรม (?node2) และ (?edge1) และ (?edge2) คือค่าของตัวแปรของเส้นเชื่อมใด ๆ ที่เป็นสมาชิกของคลาส Edge และกิจกรรม (?node1) มีเส้นเชื่อมที่วิ่งออกเป็น (?edge1) และกิจกรรม (?node2) มีเส้นเชื่อมที่วิ่งเข้าเป็น

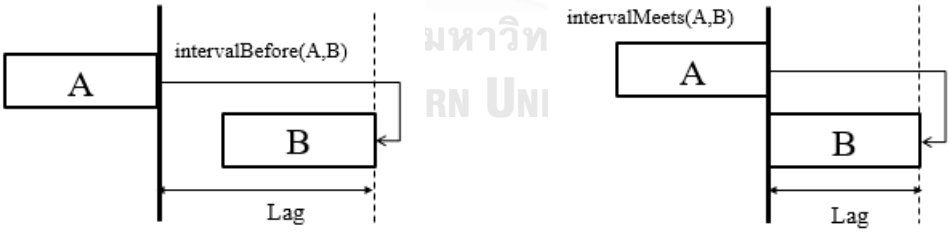
(?edge2) และเส้นเชื่อม (?edge1) และ (?edge2) เป็นเส้นเชื่อมเดียวกัน และเส้นเชื่อม (?edge1) มีประเภทการพึ่งพาเป็นค่าของตัวแปร (?depType1) และค่าตัวแปร (?depType1) มีค่าเป็น “FF” แล้วสรุปได้ว่าการพึ่งพาแบบสิ้นสุด-สิ้นสุดมีความต้องกัน

รูปที่ 3.59 กฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ต้องกัน (FF_Lag_Consistency3) ที่อธิบายด้วยภาษา SWRL

3.3.2.3.6 กฎตรวจสอบกฎตรวจสอบการพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ไม่ต้องกัน (FF_ Inconsistency)

จากการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าการพึ่งพาแบบสิ้นสุด-สิ้นสุดแบบทั่วไปจะตรงกับความสัมพันธ์แบบสิ้นสุด (intervalFinishes และความสัมพันธ์แบบเท่ากัน (intervalEquals) หรือหากมีเวลารอคอย (Lag) จะมีความสัมพันธ์แบบก่อนหน้า (intervalOverlaps) ความสัมพันธ์แบบเริ่มต้น (intervalStarts) และความสัมพันธ์แบบท่ามกลาง (intervalDuring)

ดังนั้นหากการพึ่งพาของกิจกรรมแบบสิ้นสุด-สิ้นสุดมีความสัมพันธ์ของช่วงเวลาใน 5 รูปแบบข้างต้นก็จะสรุปได้ว่ามีความต้องกัน ในทางตรงกันข้ามหากกิจกรรมใด ๆ มีความสัมพันธ์ในรูปแบบอื่น ๆ นอกเหนือจากนี้ก็จะสรุปได้ว่าไม่ต้องกัน ซึ่งกรณีของการไม่ต้องกันสามารถเป็นได้ 2 รูปแบบตามรูปที่ 3.60



FF_ Inconsistency1 FF_ Inconsistency2

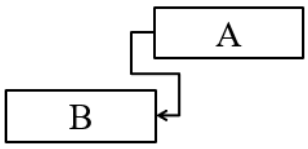
รูปที่ 3.60 การพึ่งพาแบบสิ้นสุด-สิ้นสุดที่ไม่ต้องกันในรูปแบบต่าง ๆ ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็มและทฤษฎีช่วงเวลาของ Allen

3.3.2.4 กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-สิ้นสุด (SF)

รูปแบบทั่วไปของการพึ่งพาแบบเริ่มต้น-สิ้นสุด จะแสดงดังรูปที่ 3.61 คือกิจกรรมที่ตามหลังจะสิ้นสุดทันทีเมื่อกิจกรรมก่อนหน้าเริ่มต้น ซึ่งเมื่อแปลงให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าตรงกับ ความสัมพันธ์แบบประชิดแบบผกผัน (intervalMetBy)

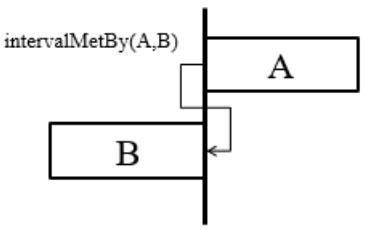
แต่ในทางปฏิบัติพบว่ากิจกรรมที่ตามหลังอาจจะไม่สามารถสิ้นสุดได้ทันที ก่อนที่จะมีการเริ่มต้นของกิจกรรมก่อนหน้า เรียกการพึ่งพาลักษณะนี้ว่าการเริ่มต้น-สิ้นสุดแบบมีเวลารอคอย (Start-to-Finish with Lag) การพึ่งพาแบบมีเวลารอคอยนั้น เมื่อทำการอนุมานด้วยกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าตรงกับความสัมพันธ์แบบถูกซ้อนทับหรือซ้อนทับแบบผกผัน (intervalOverlappedBy)

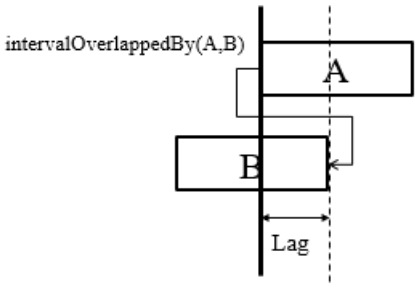
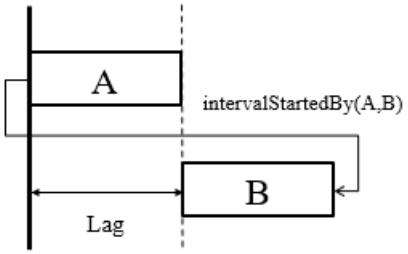
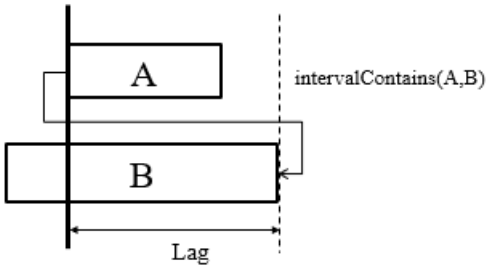
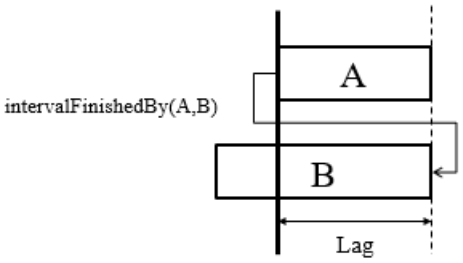
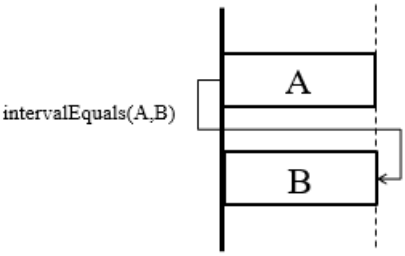
เนื่องจากการพึ่งพาแบบเริ่มต้น-สิ้นสุด ไม่มีการนิยามเวลานำ (Lead) ดังนั้นจึงไม่นำมาพิจารณาในการตรวจสอบความต้องกัน



รูปที่ 3.61 รูปแบบทั่วไปของการพึ่งพาแบบเริ่มต้น-สิ้นสุด

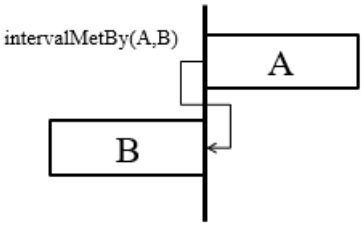
ตารางที่ 3.6 กฎการตรวจสอบการพึ่งพาแบบเริ่มต้น-สิ้นสุด

ลำดับ	กฎ	ทฤษฎีช่วงเวลาของ Allen	มีความต้องกัน
1.	กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-สิ้นสุดแบบที่ต้องกัน (SF_Consistency)		✓

ลำดับ	กฎ	ทฤษฎีช่วงเวลาของ Allen	มีความต้องกัน
2.	กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-สิ้นสุดแบบมีเวลารอคอยที่ต้องกัน (SF_Lag_Consistency)		✓
3.	กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-สิ้นสุดแบบที่ไม่ต้องกัน แบบที่ 1 (SF_Inconsistency1)		✗
4.	กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-สิ้นสุดแบบที่ไม่ต้องกัน แบบที่ 2 (SF_Inconsistency2)		✗
5.	กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-สิ้นสุดแบบที่ไม่ต้องกัน แบบที่ 3 (SF_Inconsistency3)		✗
6.	กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-สิ้นสุดแบบที่ไม่ต้องกัน แบบที่ 4 (SF_Inconsistency4)		✗

3.3.2.4.1 กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-สิ้นสุดแบบที่ต้องกัน (SF_Consistency)

จากการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าการพึ่งพาแบบเริ่มต้น-สิ้นสุดแบบทั่วไปจะตรงกับความสัมพันธ์แบบประชิดแบบผกผัน (intervalMetBy) ดังนั้นหากการพึ่งพาของกิจกรรมเป็นแบบเริ่มต้น-สิ้นสุด และมีความสัมพันธ์แบบประชิดแบบผกผันก็จะสามารถสรุปได้ว่ามีความต้องกัน สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.62 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.63



รูปที่ 3.62 การพึ่งพาแบบเริ่มต้น-สิ้นสุดที่ต้องกัน (SF_Consistency)
ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็มและทฤษฎีช่วงเวลาของ Allen

Node(?node1) ^ Node(?node2) ^ nodeName(?node1, ?name1) ^ nodeName(?node2, ?name2) ^ isPredecessorOf(?node1, ?node2) ^ intervalMetBy(?node1, ?node2) ^ Edge(?edge1) ^ Edge(?edge2) ^ hasOutputEdge(?node1, ?edge1) ^ hasInputEdge(?node2, ?edge2) ^ sameAs(?edge1, ?edge2) ^ dependencyType(?edge1, "SF") -> sqwrl:select(?node1, ?edge1, ?node2)

ชื่อกฎ: กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-สิ้นสุดที่ต้องกัน (SF_Consistency)

วัตถุประสงค์ของกฎ: เพื่อตรวจสอบความต้องกันของความสัมพันธ์ของกิจกรรมที่อยู่ก่อนหน้าและกิจกรรมที่ตามหลัง ที่มีการพึ่งพาแบบเริ่มต้น-สิ้นสุดที่ต้องกัน

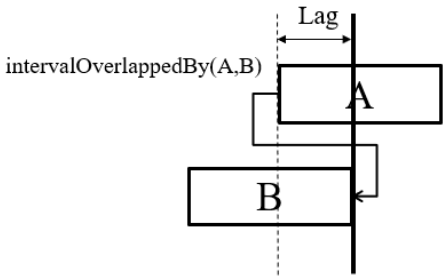
คำอธิบาย: ถ้า (?node1) และ (?node2) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีความสัมพันธ์แบบลำดับก่อนหน้ากับกิจกรรม (?node2) และกิจกรรม (?node1) มีความสัมพันธ์แบบประชิดแบบผกผันกับกิจกรรม (?node2) และ (?edge1) และ (?edge2) คือค่าของตัวแปรของเส้นเชื่อมใด ๆ ที่เป็นสมาชิกของคลาส Edge และกิจกรรม (?node1) มีเส้นเชื่อมที่วิ่งออกเป็น (?edge1) และกิจกรรม (?node2) มีเส้นเชื่อมที่วิ่งเข้าเป็น

(?edge2) และเส้นเชื่อม (?edge1) และ (?edge2) เป็นเส้นเชื่อมเดียวกัน และเส้นเชื่อม (?edge1) มีประเภทการพึ่งพามีค่าเป็น “SF” แล้วสรุปได้ว่าการพึ่งพาแบบเริ่มต้น-สิ้นสุดมีความต้องกัน

รูปที่ 3.63 กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-สิ้นสุดที่ต้องกัน (SF_Consistency) ที่อธิบายด้วยภาษา SWRL

3.3.2.4.2 กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-สิ้นสุดแบบมีเวลารอคอยที่ต้องกัน (SF_Lag_Consistency)

จากการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าการพึ่งพาแบบเริ่มต้น-สิ้นสุดแบบมีเวลารอคอย (Lag) จะตรงกับความสัมพันธ์แบบถูกซ้อนทับ (intervalOverlappedBy) ดังนั้นหากการพึ่งพาของกิจกรรมเป็นแบบเริ่มต้น-สิ้นสุด และมีความสัมพันธ์แบบถูกซ้อนทับก็จะสามารถสรุปได้ว่ามีความต้องกัน สามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.64 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.65



รูปที่ 3.64 การพึ่งพาแบบเริ่มต้น-สิ้นสุดที่ต้องกัน (SF_Lag_Consistency) ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็มและทฤษฎีช่วงเวลาของ Allen

Node(?node1) ^ Node(?node2) ^ nodeName(?node1, ?name1) ^ nodeName(?node2, ?name2) ^ isPredecessorOf(?node1, ?node2) ^ intervalOverlappedBy(?node1, ?node2) ^ Edge(?edge1) ^ Edge(?edge2) ^ hasOutputEdge(?node1, ?edge1) ^ hasInputEdge(?node2, ?edge2) ^ sameAs(?edge1, ?edge2) ^ dependencyType(?edge1, "SF") -> sqwrl:select(?node1, ?edge1, ?node2)

ชื่อกฎ: กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-สิ้นสุดที่ต้องกัน (SF_Lag_Consistency)

วัตถุประสงค์ของกฎ: เพื่อตรวจสอบความต้องกันของความสัมพันธ์ของกิจกรรมที่อยู่ก่อนหน้าและกิจกรรมที่ตามหลัง ที่มีการพึ่งพาแบบเริ่มต้น-สิ้นสุดแบบมีเวลารอคอยที่ต้องกัน

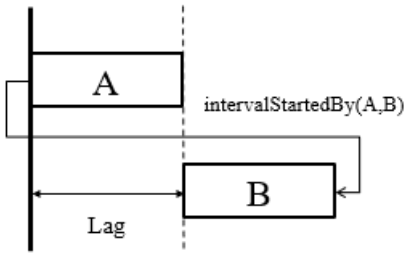
คำอธิบาย: ถ้า (?node1) และ (?node2) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีความสัมพันธ์แบบลำดับก่อนหน้ากับกิจกรรม (?node2) และกิจกรรม (?node1) มีความสัมพันธ์แบบถูกซ้อนทับ (?node2) และ (?edge1) และ (?edge2) คือค่าของตัวแปรของเส้นเชื่อมใด ๆ ที่เป็นสมาชิกของคลาส Edge และกิจกรรม (?node1) มีเส้นเชื่อมที่วิ่งออกเป็น (?edge1) และกิจกรรม (?node2) มีเส้นเชื่อมที่วิ่งเข้าเป็น (?edge2) และเส้นเชื่อม (?edge1) และ (?edge2) เป็นเส้นเชื่อมเดียวกัน และเส้นเชื่อม (?edge1) มีประเภทการพึ่งพามีค่าเป็น “SF” แล้วสรุปได้ว่าการพึ่งพาแบบเริ่มต้น-สิ้นสุดมีความต้องกัน

รูปที่ 3.65 กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-สิ้นสุดที่ต้องกัน (SF_Lag_Consistency) ที่อธิบายด้วยภาษา SWRL

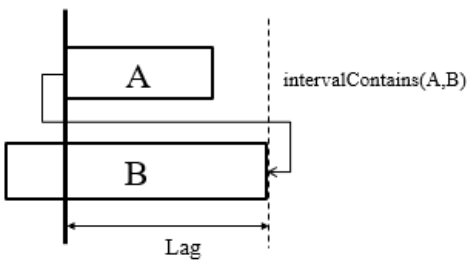
3.3.2.4.3 กฎตรวจสอบการพึ่งพาแบบเริ่มต้น-สิ้นสุดแบบที่ไม่ต้องกัน (SF_Inconsistency)

จากการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าการพึ่งพาแบบเริ่มต้น-สิ้นสุดแบบทั่วไปจะตรงกับความสัมพันธ์แบบถูกประชิด (intervalMetBy) หรือหากมีเวลารอคอย (Lag) จะมีความสัมพันธ์แบบถูกซ้อนทับ (intervalOverlapped)

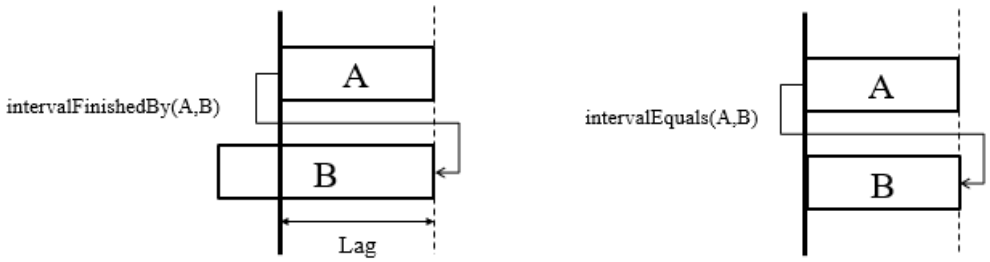
ดังนั้นหากการพึ่งพาของกิจกรรมแบบสิ้นสุด-สิ้นสุดมีความสัมพันธ์ของช่วงเวลาใน 2 รูปแบบข้างต้นก็จะสรุปได้ว่ามีความต้องกัน ในทางตรงกันข้ามหากกิจกรรมใด ๆ มีความสัมพันธ์ในรูปแบบอื่น ๆ นอกเหนือจากนี้ก็จะสรุปได้ว่าไม่ต้องกัน ซึ่งกรณีของการไม่ต้องกันสามารถเป็นได้ 4 รูปแบบตามรูปที่ 3.66



SF_ Inconsistency1



SF_ Inconsistency2



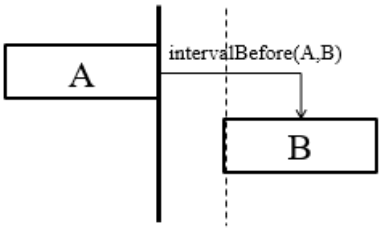
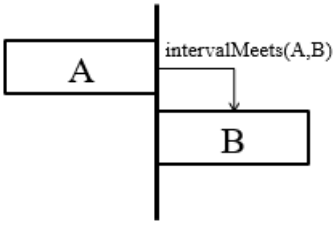
SF_Inconsistency3 SF_Inconsistency4

รูปที่ 3.66 การพึ่งพาแบบเริ่มต้น-สิ้นสุดที่ไม่ต้องกันในรูปแบบต่าง ๆ
ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็มและทฤษฎีช่วงเวลาของ Allen

3.3.3 กฎตรวจสอบความต้องกันของการจัดสรรทรัพยากร

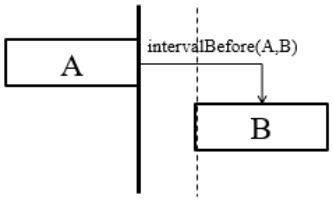
ตามหลักของการจัดสรรทรัพยากรที่ดีคือไม่ควรมอบหมายงานที่เกิดขึ้นในเวลาเดียวกัน (Multi-tasking) ให้กับทรัพยากรเดียวกัน [33] ดังนั้นกฎที่ใช้ในการตรวจสอบความต้องกันของการจัดการทรัพยากรจะทำการตรวจสอบกิจกรรมทั้งหมดที่บุคลากรนั้นได้รับมอบหมาย โดยที่ทุกกิจกรรมจะต้องอยู่ในรูปแบบของความสัมพันธ์แบบก่อนหน้า (intervalBefore) หรือความสัมพันธ์แบบประชิด (intervalMeets) เท่านั้น เพราะหากเป็นความสัมพันธ์ในรูปแบบอื่น ๆ จะมีการซ้อนทับหรือเหลื่อมกันของเวลา หากพบว่ามีความสัมพันธ์ในรูปแบบอื่น จะอนุมานได้ว่า แผนภาพข่ายงานแบบพีดีเอ็มนี้ไม่มีความต้องกันของการจัดการทรัพยากร

ตารางที่ 3.7 กฎการตรวจสอบการจัดสรรทรัพยากร

ลำดับ	กฎ	ทฤษฎีช่วงเวลาของ Allen	มีความต้องกัน
1.	กฎการตรวจสอบการจัดสรรทรัพยากรที่ต้องกัน ในรูปแบบของความสัมพันธ์แบบก่อนหน้า (Resource_Consistency1)		✓
2.	กฎการตรวจสอบการจัดสรรทรัพยากรที่ต้องกัน ในรูปแบบของความสัมพันธ์แบบประชิด (Resource_Consistency2)		✓

3.3.3.1 กฎการตรวจสอบการจัดสรรทรัพยากรที่ต้องกัน ในรูปแบบของความสัมพันธ์แบบก่อนหน้า (Resource_Consistency1)

จากการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าความสัมพันธ์แบบก่อนหน้า (intervalBefore) เป็นรูปแบบของความสัมพันธ์ที่ไม่มีการซ้อนทับกันของแต่ละกิจกรรม ดังนั้นหากมีการมอบหมายงานให้กับทรัพยากรที่เป็นคนคนเดียวกันให้กับกิจกรรมดังกล่าวก็จะสามารถกระทำได้ จึงสรุปได้ว่าการจัดสรรทรัพยากรมีความต้องกัน ซึ่งสามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.67 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.68



รูปที่ 3.67 ความสัมพันธ์แบบก่อนหน้าที่ต้องกันสำหรับกฎการจัดสรรทรัพยากร (Resource_Consistency1) ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็ม และทฤษฎีช่วงเวลาของ Allen

Node(?node1) ^ Node(?node2) ^ isPredecessorOf(?node1, ?node2) ^ intervalBefore(?node1, ?node2) ^ Edge(?edge1) ^ Edge(?edge2) ^ hasOutputEdge(?node1, ?edge1) ^ hasInputEdge(?node2, ?edge2) ^ sameAs(?edge1, ?edge2) ^ Resource(?resource1) ^ Resource(?resource2) ^ hasResource(?node1, ?resource1) ^ hasResource(?node2, ?resource2) ^ sameAs(?resource1, ?resource2) -> sqwrl:select(?resource1, ?node1, ?node2)

ชื่อกฎ: กฎการตรวจสอบการจัดสรรทรัพยากรที่ต้องกันในรูปแบบของความสัมพันธ์แบบก่อนหน้า (Resource_Consistency1)

วัตถุประสงค์ของกฎ: เพื่อตรวจสอบความต้องกันของการจัดสรรทรัพยากร ในรูปแบบของความสัมพันธ์แบบก่อนหน้า

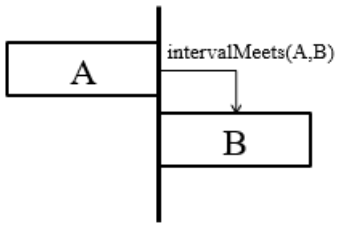
คำอธิบาย: ถ้า (?node1) และ (?node2) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีความสัมพันธ์แบบลำดับก่อนหน้ากับกิจกรรม (?node2) และกิจกรรม (?node1) มีความสัมพันธ์แบบก่อนหน้ากับกิจกรรม (?node2) และ (?edge1) และ

(?edge2) คือค่าของตัวแปรของเส้นเชื่อมใด ๆ ที่เป็นสมาชิกของคลาส Edge และกิจกรรม (?node1) มีเส้นเชื่อมที่วิ่งออกเป็น (?edge1) และกิจกรรม (?node2) มีเส้นเชื่อมที่วิ่งเข้าเป็น (?edge2) และเส้นเชื่อม (?edge1) และ (?edge2) เป็นเส้นเชื่อมเดียวกัน และ (?resource1) และทรัพยากร (?resource2) คือค่าตัวแปรของทรัพยากรใด ๆ ที่เป็นสมาชิกของคลาส Resource และกิจกรรม (?node1) มีทรัพยากรเป็น (?resource1) และกิจกรรม (?node2) มีทรัพยากรเป็น (?resource2) และ (?resource1) และ (?resource2) เป็นทรัพยากรเดียวกัน แล้วสรุปได้ว่าการจัดสรรทรัพยากรมีความต้องกัน

รูปที่ 3.68 กฎตรวจสอบความต้องกันของทรัพยากรที่ต้องกันในรูปแบบของความสัมพันธ์แบบก่อนหน้า (Resource_Consistency1) ที่อธิบายด้วยภาษา SWRL

3.3.3.2 กฎการตรวจสอบการจัดสรรทรัพยากรที่ต้องกันในรูปแบบของความสัมพันธ์แบบประชิด (Resource_Consistency2)

จากการอนุมานด้วยชุดกฎเพื่อหาความสัมพันธ์ที่ซ่อนเร้นให้อยู่ในรูปแบบทฤษฎีช่วงเวลาของ Allen จะพบว่าความสัมพันธ์แบบประชิด (intervalMeets) เป็นรูปแบบของความสัมพันธ์ที่ไม่มีการซ้อนทับกันของแต่ละกิจกรรม ดังนั้นหากมีการมอบหมายงานให้กับทรัพยากรที่เป็นคนคนเดียวกันให้กับกิจกรรมดังกล่าวก็จะสามารถกระทำได้ จึงสรุปได้ว่าการจัดสรรทรัพยากรมีความต้องกัน ซึ่งสามารถแสดงความสัมพันธ์ในรูปแบบของแผนภาพข่ายงานแบบพีดีเอ็มได้ดังรูปที่ 3.69 และสามารถอธิบายเป็นภาษากฎเอสดับเบิลยูอาร์แอลได้ดังรูปที่ 3.70



รูปที่ 3.69 ความสัมพันธ์แบบก่อนหน้าที่ต้องกันสำหรับกฎการจัดสรรทรัพยากรในรูปแบบของความสัมพันธ์แบบประชิด (Resource_Consistency2) ที่อธิบายด้วยแผนภาพข่ายงานแบบพีดีเอ็มและทฤษฎีช่วงเวลาของ Allen

Node(?node1) ^ Node(?node2) ^ isPredecessorOf(?node1, ?node2) ^ intervalMeets(?node1, ?node2) ^ Edge(?edge1) ^ Edge(?edge2) ^ hasOutputEdge(?node1, ?edge1) ^ hasInputEdge(?node2, ?edge2) ^ sameAs(?edge1,

?edge2) ^ Resource(?resource1) ^ Resource(?resource2) ^ hasResource(?node1, ?resource1) ^ hasResource(?node2, ?resource2) ^ sameAs(?resource1, ?resource2) -> sqwrl:select(?resource1, ?node1, ?node2)

ชื่อกฎ: กฎการตรวจสอบการจัดสรรทรัพยากรที่ต้องกัน ในรูปแบบของความสัมพันธ์แบบประชิด (Resource_Consistency2)

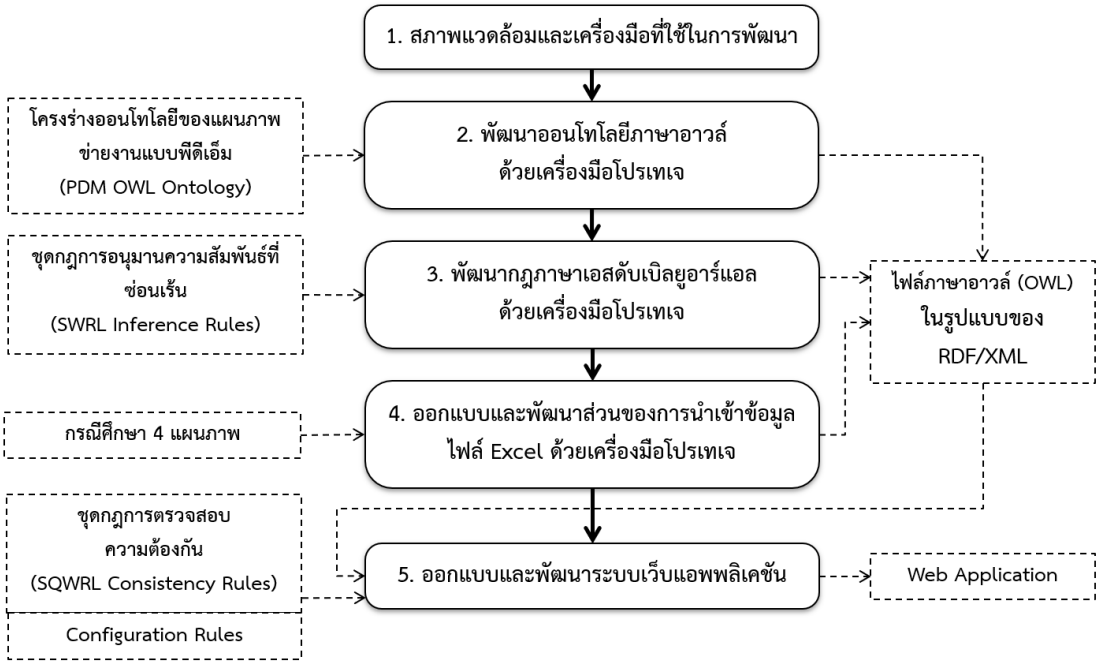
วัตถุประสงค์ของกฎ: เพื่อตรวจสอบความต้องกันของการจัดสรรทรัพยากร ในรูปแบบความสัมพันธ์แบบประชิด

คำอธิบาย: ถ้า (?node1) และ (?node2) คือค่าตัวแปรของกิจกรรมใด ๆ ที่เป็นสมาชิกของคลาส Node และกิจกรรม (?node1) มีความสัมพันธ์แบบลำดับก่อนหน้ากับกิจกรรม (?node2) และกิจกรรม (?node1) มีความสัมพันธ์แบบประชิดกับกิจกรรม (?node2) และ (?edge1) และ (?edge2) คือค่าของตัวแปรของเส้นเชื่อมใด ๆ ที่เป็นสมาชิกของคลาส Edge และกิจกรรม (?node1) มีเส้นเชื่อมที่วิ่งออกเป็น (?edge1) และกิจกรรม (?node2) มีเส้นเชื่อมที่วิ่งเข้าเป็น (?edge2) และเส้นเชื่อม (?edge1) และ (?edge2) เป็นเส้นเชื่อมเดียวกัน และ (?resource1) และทรัพยากร (?resource2) คือค่าตัวแปรของทรัพยากรใด ๆ ที่เป็นสมาชิกของคลาส Resource และกิจกรรม (?node1) มีทรัพยากรเป็น (?resource1) และกิจกรรม (?node2) มีทรัพยากรเป็น (?resource2) และ (?resource1) และ (?resource2) เป็นทรัพยากรเดียวกัน แล้วสรุปได้ว่าการจัดสรรทรัพยากรมีความต้องกัน

รูปที่ 3.70 กฎตรวจสอบความต้องกันของทรัพยากรที่ต้องกันในรูปแบบของความสัมพันธ์แบบประชิด (Resource_Consistency2) ที่อธิบายด้วยภาษา SWRL

บทที่ 4
การออกแบบและการพัฒนาระบบ

งานวิจัยนี้มีจุดมุ่งหมายเพื่อการออกแบบและพัฒนาออนโทโลยีพร้อมกับการสร้างกฎเพื่อตรวจสอบความต้องกันเชิงความหมายของแผนภาพข่ายงานแบบพีดีเอ็ม ซึ่งในบทที่ 3 ได้กล่าวถึงการออกแบบโครงสร้างของแบบจำลองออนโทโลยีและกฎต่าง ๆ เป็นที่เรียบร้อยแล้ว ดังนั้นในบทที่ 4 จะกล่าวถึงขั้นตอนในการนำโครงสร้างออนโทโลยีและกฎที่ได้ออกแบบไว้ มาพัฒนาเป็นออนโทโลยีด้วยภาษาอาวล์และกฎด้วยภาษาเอสดับเบิลยูอาร์แอลด้วยเครื่องมือโปรเทเจ จากนั้นจึงจะเข้าสู่ขั้นตอนการออกแบบเพื่อพัฒนาให้สามารถใช้งานได้ในรูปแบบเว็บแอพพลิเคชัน ซึ่งจะมีวิธีการดำเนินตามกระบวนการ ดังรูปที่ 4.1



รูปที่ 4.1 ภาพรวมการออกแบบและการพัฒนาระบบ

จากรูปที่ 4.1 อธิบายภาพรวมการออกแบบและการพัฒนาระบบ โดยเริ่มจากการจัดเตรียมสภาพแวดล้อมและเครื่องมือที่ใช้ในการพัฒนา จากนั้นจึงนำโครงร่างออนโทโลยีของแผนภาพข่ายงานแบบพีดีเอ็มที่ได้จากการออกแบบในบทที่ 3 มาพัฒนาให้เป็นออนโทโลยีภาษาอาวล์ด้วยเครื่องมือโปรเทเจ จากนั้นชุดกฎแรกที่ได้ทำการออกแบบหรือชุดกฎการอนุมานความสัมพันธ์ที่ซ่อนเร้นจะถูกนำเขียนลงใน SWRLTab ซึ่งเป็นส่วนเสริม (Plug-in) ที่ถูกติดตั้งมากับเครื่องมือโปรเทเจ โดย Tab SWRL จะมีเครื่องมือการให้เหตุผลที่ชื่อว่า Drools ที่สามารถประมวลผลกฎที่สร้างขึ้นมาได้โดย

อัตโนมัติ จากนั้นจึงทำการออกแบบและพัฒนาส่วนของการนำเข้าข้อมูลไฟล์ Excel ด้วยเครื่องมือโปรเทเจ โดยจะใช้กรณีศึกษา 4 กรณีที่เป็นไฟล์ Excel มาเป็นข้อมูลนำเข้า เมื่อทำการพัฒนาส่วนต่าง ๆ ในขั้นตอนที่ 2, 3 และ 4 เรียบร้อยแล้ว ก็จะสามารถส่งออกไฟล์อาวล์ในรูปแบบของไวยากรณ์ภาษา RDF/XML ซึ่งไฟล์อาวล์ที่ส่งออกนี้ จะเป็นข้อมูลนำเข้าสำหรับเว็บแอพพลิเคชันต่อไป และในขั้นตอนสุดท้ายจะเป็นการออกแบบและพัฒนาระบบเว็บแอพพลิเคชัน โดยจะนำชุดกฎการตรวจสอบความต้องกันที่ได้จากการออกแบบในบทที่ 3 ไปพัฒนาด้วยภาษาจาวาและทำการจัดเก็บในรูปแบบของ Configuration Rules เพื่อให้โปรแกรมจาวาสามารถเรียกใช้กฎดังกล่าวเพื่อนำไปประมวลผลกับข้อมูลนำเข้า เพื่อตรวจสอบความต้องกันของแผนภาพต่อไป สำหรับรายละเอียดในขั้นตอนทั้ง 5 ขั้นตอนนี้ สามารถอธิบายได้ดังต่อไปนี้

4.1 สภาพแวดล้อมและเครื่องมือที่ใช้ในการพัฒนา

เครื่องมือที่ใช้ในการพัฒนางานวิจัยมีรายละเอียดดังนี้

4.1.1 สภาพแวดล้อม

1. ระบบปฏิบัติการ Window 10 แบบ 64 bit
2. หน่วยประมวลผล Processor Core i7, CpU 2.40 GHz
3. หน่วยความจำ 8 กิกะไบต์ (RAM 8 GB)

4.1.2 เครื่องมือที่ใช้ในการพัฒนา

1. โปรแกรมโปรเทเจ เวอร์ชัน 5.2 (Protégé 5.2.0)
2. ส่วนเชื่อมต่อภาษาอาวล์ (OWL API)
3. ส่วนเชื่อมต่อภาษาเอสดับเบิลยูอาร์แอล (SWRL API)
4. แพลตฟอร์มจาวา (Java Platform JDK 10, Java SE Development Kit 10.0.1)
5. เครื่องมือพัฒนาโปรแกรมภาษาจาวา Eclipse Jee Oxygen
6. เครื่องแม่ข่าย (Server) Apache Tomcat 8.5.31
7. เครื่องลูกข่าย (Client) Apache PHP

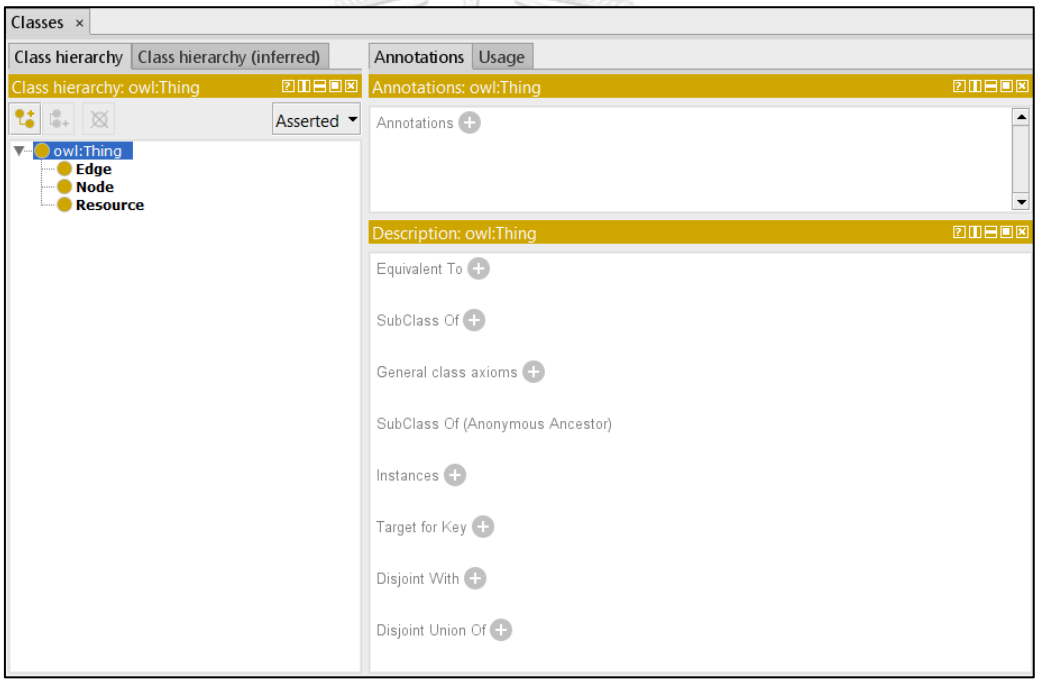
4.2 พัฒนาออนโทโลยีภาษาอาวล์ด้วยเครื่องมือโปรเทเจ

ในขั้นตอนนี้จะเป็นการสร้างแฟ้มข้อมูลภาษาอาวล์ (OWL file) ด้วยเครื่องมือโปรเทเจ ตามแบบจำลองในรูปที่ 3.20 ที่ได้ทำการออกแบบไว้ในบทที่ 3 ซึ่งแบบจำลองดังกล่าวมีองค์ประกอบตามโครงสร้างของภาษาอาวล์ที่แบ่งออกเป็น 3 ส่วนดังนี้

1) คลาส (Class) เป็นกลุ่มของแนวคิดที่มีคุณสมบัติเดียวกันภายใต้ขอบเขตที่สนใจ ซึ่งคลาสที่ได้จากการออกแบบประกอบไปด้วย 3 คลาสหลักดังนี้

- คลาสโหนดกิจกรรม (Node) เป็นคลาสที่มีการอธิบายรายละเอียดเกี่ยวกับเวลาที่ใช้ในการดำเนินแต่ละกิจกรรมในรูปแบบต่าง ๆ
- คลาสเส้นเชื่อมระหว่างกิจกรรม (Edge) เป็นคลาสที่อธิบายความสัมพันธ์ (Dependency) ระหว่างกิจกรรมที่อยู่ก่อนหน้า (Predecessor) และกิจกรรมที่ตามหลัง (Successor)
- คลาสทรัพยากรบุคคล (Resource) เป็นคลาสที่ใช้อธิบายรายละเอียดต่าง ๆ ที่เกี่ยวข้องกับทรัพยากรบุคคล

เมื่อนำคลาสต่าง ๆ มาสร้างด้วยเครื่องมือโปรเทเจ จะได้ผลลัพธ์ดังรูปที่ 4.2



รูปที่ 4.2 การสร้างคลาส (Class) ด้วยเครื่องมือโปรเทเจ

2) คุณลักษณะ (Property) เป็นการอธิบายความสัมพันธ์ระหว่างคลาสต่าง ๆ ภายใต้ขอบเขตที่สนใจ ซึ่งคุณลักษณะที่ได้จากการออกแบบจะแบ่งออกเป็น 2 ส่วนคือ

2.1 คุณลักษณะของอ็อบเจ็ค (Object property) ใช้ในการอธิบายความสัมพันธ์ระหว่างคลาส ซึ่งประกอบไปด้วยความสัมพันธ์ในรูปแบบต่าง ๆ ดังนี้

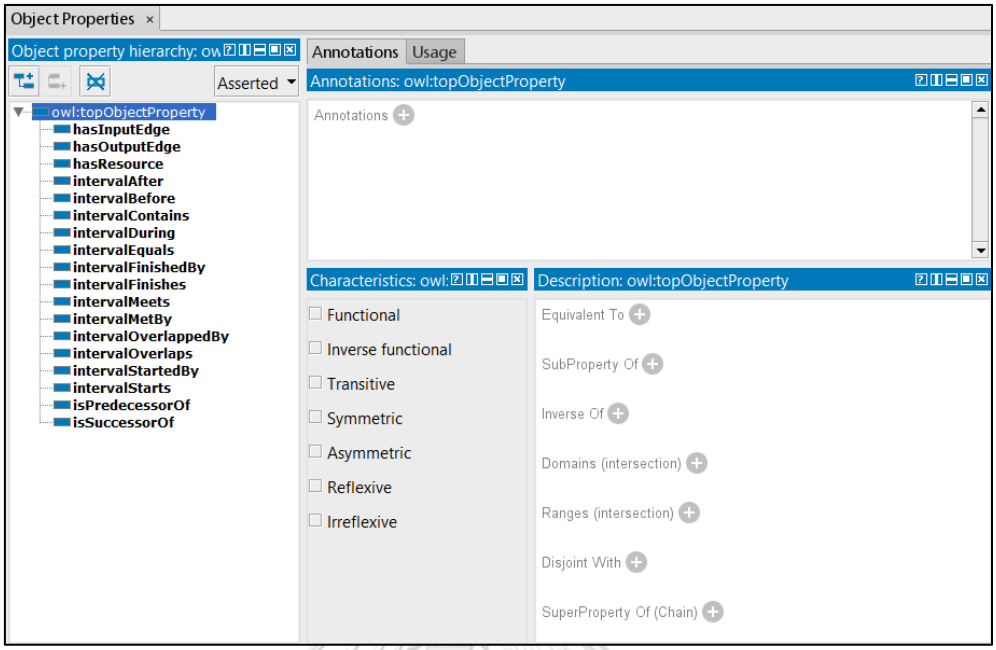
- hasInputEdge ทำหน้าที่อธิบายความสัมพันธ์แบบเส้นเชื่อมที่วิ่งเข้าระหว่างคลาส Node และคลาส Edge

- hasOutputEdge ทำหน้าที่อธิบายความสัมพันธ์แบบเส้นเชื่อมที่วิ่งออกระหว่างคลาส Node และคลาส Edge
- hasResource ทำหน้าที่อธิบายความสัมพันธ์ในการมอบหมายกิจกรรมให้แก่ทรัพยากรบุคคลระหว่างคลาส Node และ คลาส Resource
- isPredecessorOf ซึ่งมีความสัมพันธ์แบบผกผันกับ isSuccessorOf ทำหน้าที่อธิบายความสัมพันธ์ของโหนดก่อนหน้าที่เชื่อมด้วย edge เดียวกันกับโหนดที่ตามหลังในคลาส Node
- intervalBefore ซึ่งมีความสัมพันธ์แบบผกผันกับ intervalAfter ทำหน้าที่อธิบายความสัมพันธ์แบบก่อนหน้าระหว่างโหนดที่เชื่อมกันในคลาส Node
- intervalDuring ซึ่งมีความสัมพันธ์แบบผกผันกับ intervalContains ทำหน้าที่อธิบายความสัมพันธ์แบบท่ามกลางระหว่างโหนดที่เชื่อมกันในคลาส Node
- intervalEquals ซึ่งมีความสัมพันธ์แบบสมมาตร ทำหน้าที่อธิบายความสัมพันธ์แบบเท่ากันระหว่างโหนดที่เชื่อมกันในคลาส Node
- intervalFinishes ซึ่งมีความสัมพันธ์แบบผกผันกับ intervalFinishedBy ทำหน้าที่อธิบายความสัมพันธ์แบบสิ้นสุดระหว่างโหนดที่เชื่อมกันในคลาส Node
- intervalMeets ซึ่งมีความสัมพันธ์แบบผกผันกับ intervalMetBy ทำหน้าที่อธิบายความสัมพันธ์แบบประชิดระหว่างโหนดที่เชื่อมกันในคลาส Node
- intervalOverlaps ซึ่งมีความสัมพันธ์แบบผกผันกับ intervalOverlappedBy ทำหน้าที่อธิบายความสัมพันธ์แบบซ้อนทับกันระหว่างโหนดที่เชื่อมกันในคลาส Node
- intervalStarts ซึ่งมีความสัมพันธ์แบบผกผันกับ intervalStartedBy ทำหน้าที่อธิบายความสัมพันธ์แบบเริ่มต้นระหว่างโหนดที่เชื่อมกันในคลาส Node

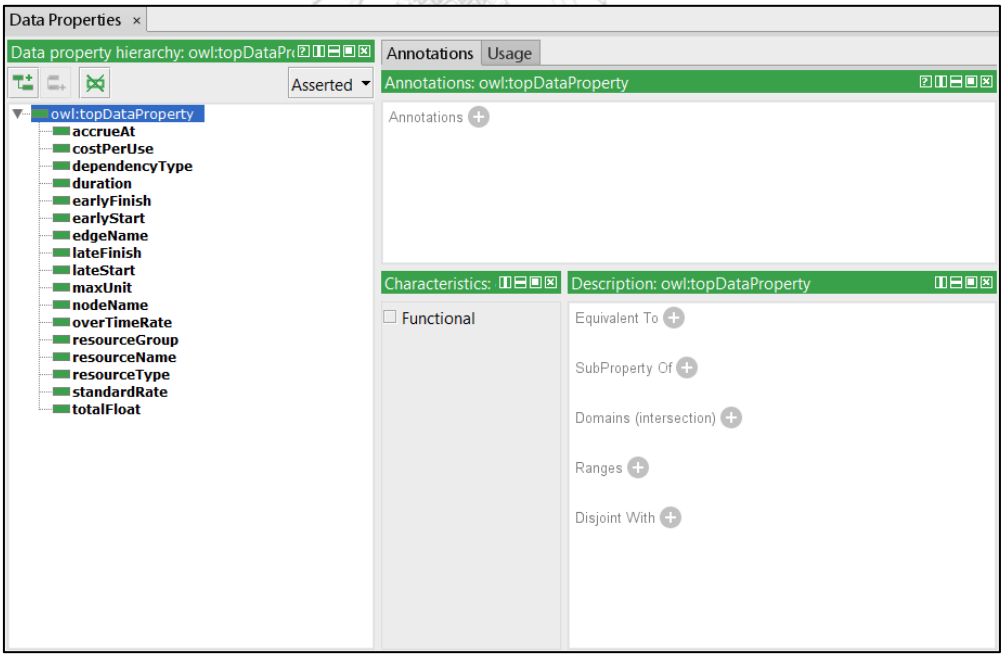
2.2 คุณลักษณะของข้อมูล (Data property) ใช้ในการอธิบายคุณลักษณะเฉพาะตัวของคลาส ซึ่งประกอบไปด้วยข้อมูลต่าง ๆ ที่ใช้อธิบายแต่ละคลาส ดังนี้

- คุณลักษณะของข้อมูลที่ใช้ในการอธิบายคลาส Node ประกอบด้วย nodeName, duration, earlyStart, earlyFinish, lateStart, lateFinish, totalFloat,
- คุณลักษณะของข้อมูลที่ใช้ในการอธิบายคลาส Edge ประกอบด้วย edgeName, DependencyType
- คุณลักษณะของข้อมูลที่ใช้ในการอธิบายคลาส Resource ประกอบด้วย resourceName, resourceType, resourceGroup, maxUnit, standardRate, overtimeRate, costPerRate, accrueAt

เมื่อนำคุณลักษณะต่าง ๆ มาสร้างด้วยเครื่องมือโปรเทเจ จะได้ผลลัพธ์ดังรูปที่ 4.3 และ 4.4



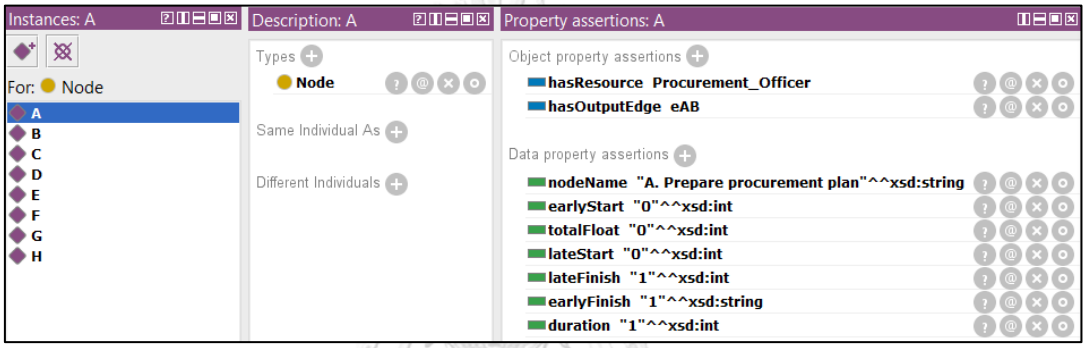
รูปที่ 4.3 การสร้างคุณลักษณะของอ็อบเจค (Property) ด้วยเครื่องมือโปรเทเจ



รูปที่ 4.4 การสร้างคุณลักษณะของข้อมูล (Data property) ด้วยเครื่องมือโปรเทเจ

3) อินสแตนซ์หรืออินดิวิดวล (Instance/Individual) เป็นข้อมูลเกี่ยวกับแผนภาพข่ายงานแบบพีดีเอ็มของงานโครงการต่าง ๆ ที่แสดงโหนดกิจกรรมและความสัมพันธ์ของแต่ละ

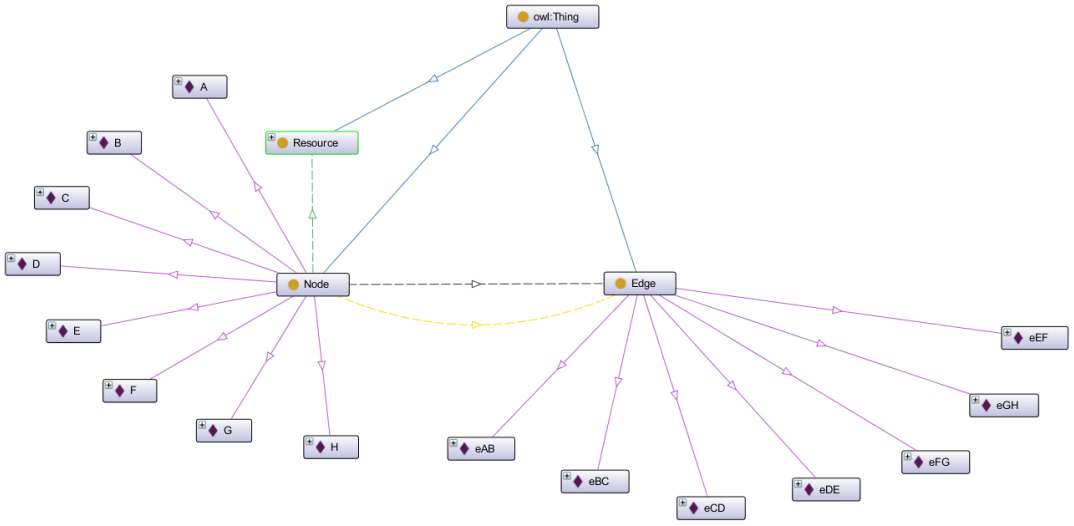
กิจกรรม ซึ่งจะแตกต่างกันไปในแต่ละโครงการ โดยจะมีการนำเข้าข้อมูลจากภายนอก ซึ่งจะกล่าวโดยละเอียดในหัวข้อ 4.4 การออกแบบและพัฒนาส่วนของการนำเข้าข้อมูลไฟล์ Excel ด้วยเครื่องมือโปรเทเจ ดังนั้นในส่วนนี้จะทำการสร้างอินสแตนซ์ด้วยวิธีเพิ่มข้อมูลอินสแตนซ์หรืออินดิวดวลเข้าไปโดยตรง โดยจะทำการเพิ่มโหนดกิจกรรม A, B, C, D, E, F, G, และ H เพื่อให้ได้ไฟล์ในภาษาอาวล์ที่สมบูรณ์แบบ และสามารถนำไปใช้ในการประมวลผลกฎด้วยเครื่องมือ SWRLTab ซึ่งเป็นเครื่องมือที่ติดตั้งมากับโปรแกรมโปรเทเจอยู่แล้ว



รูปที่ 4.5 ตัวอย่างการสร้างอินสแตนซ์ (Instance) ของกิจกรรม A ที่สร้างด้วยเครื่องมือโปรเทเจ

ในรูปที่ 4.5 เป็นตัวอย่างการสร้างอินแสตนซ์ของกิจกรรม A โดยที่ A เป็นสมาชิกที่อยู่ภายใต้คลาส Node และมีคุณลักษณะหรือความสัมพันธ์ hasResource กับคลาส Resource โดยมีค่าเป็น “Procurement_Officer” และมีความสัมพันธ์ hasOutputEdge กับคลาส Node โดยมีค่าเป็น “eAB” นอกจากนี้อินสแตนซ์ A ยังมีคุณสมบัติของข้อมูลต่าง ๆ ได้แก่ nodeName มีค่าเป็น “A. Prepare procurement plan” โดยที่มี earlyStart, totalFloat, และ lateStart มีค่าเป็น “0” และมีค่า lateFinish, earlyFinish, และ duration เป็น “1” ซึ่งจะทำการกำหนดคุณสมบัติต่าง ๆ เหล่านี้ให้กับอินสแตนซ์อื่น ๆ เช่นเดียวกัน โดยที่จะมีการกำหนดค่าที่แตกต่างกันไปของแต่ละโหนด

เมื่อทำการสร้างอินสแตนซ์ของทุกคลาสเรียบร้อยแล้ว จากนั้นนำมาแสดงผลด้วยโปรแกรม OntoGraf ที่เป็นส่วนเสริม (Plug-in) ของเครื่องมือโปรเทเจ จะแสดงผลผลลัพธ์ดังรูปที่ 4.6



รูปที่ 4.6 ตัวอย่างคลาส คุณลักษณะและอินสแตนซ์ ที่สร้างด้วยด้วยเครื่องมือโปรเทเจ แสดงผลโดยใช้ OntoGraf

หลังจากที่ทำการสร้าง Classes, Properties และ Individuals/Instances ในระบบเสร็จเรียบร้อยแล้ว ก็จะสามารถส่งออกข้อมูลดังกล่าวให้อยู่ในรูปของแฟ้มข้อมูลภาษา OWL ในรูปแบบของ RDF/XML ดังรูปที่ 4.7 ได้เพื่อนำไปใช้งานต่อไป

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://www.semanticweb.org/natty/ontologies/PDM#"
     xml:base="http://www.semanticweb.org/natty/ontologies/PDM"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
     xmlns:swrl="http://www.w3.org/2003/11/swrl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:swrla="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#">
    <owl:Ontology
rdf:about="http://www.semanticweb.org/natty/ontologies/PDM"/>


<!-- ///////////////////////////////////////////////////////////////////
//
// Classes
//
///////////////////////////////////////////////////////////////////-->

<!-- http://www.semanticweb.org/natty/ontologies/PDM#Edge -->
<owl:Class
rdf:about="http://www.semanticweb.org/natty/ontologies/PDM#Edge"/>
```

```
<!-- http://www.semanticweb.org/natty/ontologies/PDM#Node -->
<owl:Class
rdf:about="http://www.semanticweb.org/natty/ontologies/PDM#Node"/>


<!-- http://www.semanticweb.org/natty/ontologies/PDM#Resource -->
<owl:Class
rdf:about="http://www.semanticweb.org/natty/ontologies/PDM#Resource"/>


<!-- ///////////////////////////////////////////////////////////////
//
// Object Properties
//
///////////////////////////////////////////////////////////////-->

<!-- http://www.semanticweb.org/natty/ontologies/PDM#hasInputEdge -->
<owl:ObjectProperty
rdf:about="http://www.semanticweb.org/natty/ontologies/PDM#hasInputEdge">
<rdfs:domain
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#Node"/>
<rdfs:range
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#Edge"/>
</owl:ObjectProperty>


<!-- http://www.semanticweb.org/natty/ontologies/PDM#hasOutputEdge -->
<owl:ObjectProperty
rdf:about="http://www.semanticweb.org/natty/ontologies/PDM#hasOutputEdge"
>
<rdfs:domain
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#Node"/>
<rdfs:range
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#Edge"/>
</owl:ObjectProperty>


<!-- http://www.semanticweb.org/natty/ontologies/PDM#intervalAfter -->
<owl:ObjectProperty
rdf:about="http://www.semanticweb.org/natty/ontologies/PDM#intervalAfter"
>
<owl:inverseOf
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#intervalBef
ore"/>
<rdfs:domain
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#Node"/>
<rdfs:range
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#Node"/>
</owl:ObjectProperty>


<!-- http://www.semanticweb.org/natty/ontologies/PDM#intervalBefore -->
<owl:ObjectProperty
rdf:about="http://www.semanticweb.org/natty/ontologies/PDM#intervalBefore
">
```

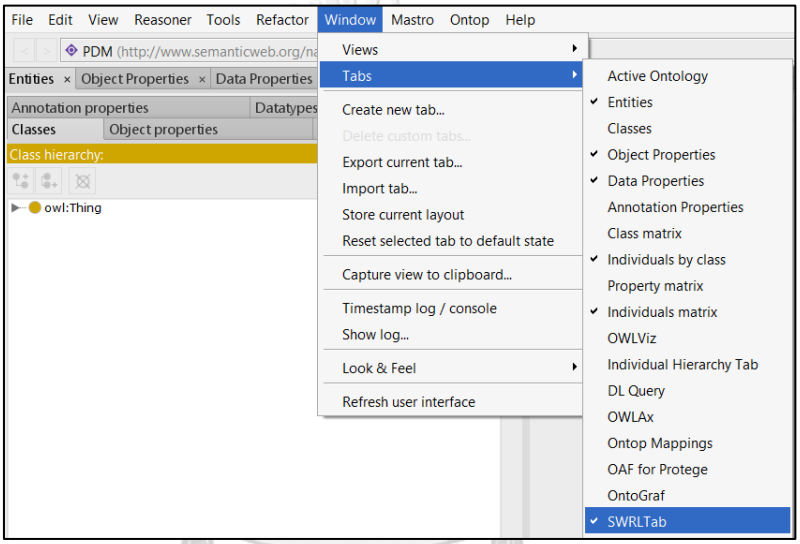
```
<rdfs:domain
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#Node"/>
<rdfs:range
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#Node"/>
</owl:ObjectProperty>

…

<!-- ///////////////////////////////////////////////////////////////////
//
// Data properties
//
///////////////////////////////////////////////////////////////////////-->

<!-- http://www.semanticweb.org/natty/ontologies/PDM#dependencyType -->
<owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/natty/ontologies/PDM#dependencyType
">
<rdfs:domain
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#Edge"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>


<!-- http://www.semanticweb.org/natty/ontologies/PDM#duration -->
<owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/natty/ontologies/PDM#duration">
<rdfs:domain
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#Node"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty>


<!-- http://www.semanticweb.org/natty/ontologies/PDM#earlyFinish -->
<owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/natty/ontologies/PDM#earlyFinish">
<rdfs:domain
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#Node"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty>


<!-- http://www.semanticweb.org/natty/ontologies/PDM#earlyStart -->
<owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/natty/ontologies/PDM#earlyStart">
<rdfs:domain
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#Node"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty>

…

<!-
//////////////////////////////////////////////////////////////////////
//
// Individuals
//
```

```
//////////////////////////////////////////////////////////////////////-->

<!-- http://www.semanticweb.org/natty/ontologies/PDM#A -->
<owl:NamedIndividual
rdf:about="http://www.semanticweb.org/natty/ontologies/PDM#A">
<rdf:type
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#Node"/>
<hasOutputEdge
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#eAB"/>
<hasOutputEdge
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#eAC"/>
<hasResource
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#John"/>
<intervalBefore
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#B"/>
<intervalBefore
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#D"/>
<intervalBefore
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#G"/>
<intervalBefore
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#H"/>
<intervalEquals
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#A"/>
<intervalOverlaps
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#E"/>
<intervalOverlaps
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#F"/>
<intervalStarts
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#C"/>
<isPredecessorOf
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#B"/>
<isPredecessorOf
rdf:resource="http://www.semanticweb.org/natty/ontologies/PDM#C"/>
<duration
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2</duration>
<earlyFinish
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2</earlyFinish>
<earlyStart
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">0</earlyStart>
<lateFinish
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">2</lateFinish>
<lateStart
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">0</lateStart>
<nodeName
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">AAA</nodeName>
<totalFloat
rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">0</totalFloat>
</owl:NamedIndividual>

...
```

รูปที่ 4.7 แฟ้มข้อมูลภาษา OWL ในรูปแบบ RDF/XML Syntax ที่ส่งออกจากเครื่องมือโปรเทเจ

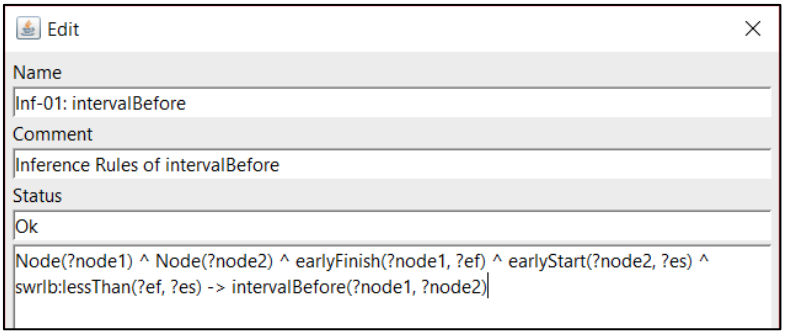
4.3 พัฒนากฎภาษาเอสดับเบิลยูอาร์แอลด้วยเครื่องมือโปรเทเจ

จากบทที่ 3 ได้มีการออกแบบกฎซึ่งแบ่งออกเป็น 2 ชุด คือ 1) ชุดกฎเพื่ออนุมานความสัมพันธ์ที่ซ่อนเร้นของแผนภาพข่ายงานแบบพีดีเอ็ม (Inference Rules) และ 2) ชุดกฎเพื่อตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม (Consistency Rules) ด้วยภาษาเอสดับเบิลยูอาร์แอล (SWRL) ในขั้นตอนต่อไปจะนำชุดกฎดังกล่าวที่ได้ออกแบบไว้มาพัฒนาให้อยู่ในรูปแบบของภาษา SWRL ตามลำดับ โดยใช้เครื่องมือโปรเทเจ ซึ่งโปรเทเจเวอร์ชัน 5.2 จะมีการติดตั้งส่วนเสริมของการพัฒนากฎด้วยภาษา SWRL มาให้ให้งานอยู่แล้ว โดยสามารถทำการเปิด SWRLTab ขึ้นมาก็จะสามารถสร้างกฎได้ทันที



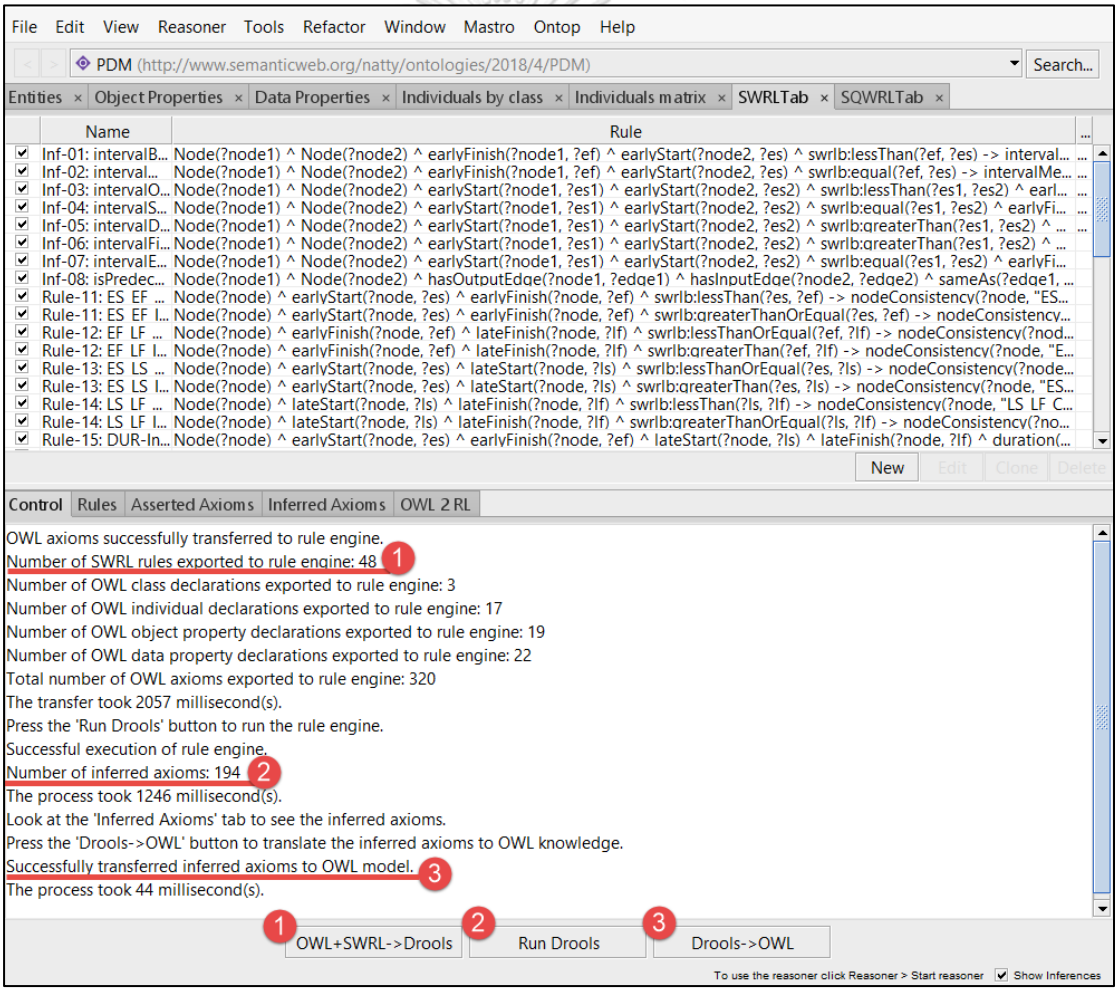
รูปที่ 4.8 การเปิดใช้งาน SWRLTab เพื่อสร้างกฎภาษา SWRL

จากนั้นจึงทำการสร้างกฎภาษา SWRL ตามที่ได้ออกแบบไว้ในบทที่ 3 ตามรูปที่ 4.9 เป็นการสร้างกฎเพื่อหาความสัมพันธ์แบบก่อนหน้า (intervalBefore) ซึ่งเป็นกฎที่ใช้ในการอนุมานความสัมพันธ์ที่ซ่อนเร้นของแผนภาพข่ายงานแบบพีดีเอ็ม (Inference Rules)



รูปที่ 4.9 ตัวอย่างการสร้างกฎภาษา SWRL ด้วยส่วนเสริม SWRLTab ในโปรแกรมโปรเทเจ

เมื่อทำการสร้างกฎจนครบทุกข้อตามที่ได้ออกแบบ จะได้กฎทั้งหมดดังรูปที่ 4.10 จากนั้นทดสอบการใช้งานกฎด้วยการกดปุ่ม OWL+SWRL->Drools จะพบมีกฎที่ถูกส่งออกไปยังเครื่องมือ Drools ซึ่งเป็นเครื่องมือที่ใช้ในการประมวลผลกฎทั้งหมด 48 ข้อ จากนั้นจึงกดปุ่ม Run Drools เพื่อทำการประมวลผลกฎแต่ละข้อที่สร้างขึ้น จะพบว่ามีจำนวนสัจพจน์ (Axioms) ที่เกิดขึ้นทั้งหมด 194 สัจพจน์ สุดท้ายกดปุ่ม Drools->OWL เพื่อส่งผลการอนุมานความสัมพันธ์ที่ได้จาก Drools Engine กลับไปปรับปรุงแบบจำลองภาษาอาวล์ (OWL Model) ก็จะได้แบบจำลองภาษาอาวล์ที่สามารถนำไปสืบค้นเพื่อตรวจสอบความต้องกันได้ ซึ่งกฎต่าง ๆ เหล่านี้จะถูกนำไปใช้งานในเว็บแอพพลิเคชันในส่วนของ Configuration Rules ซึ่งจะถูกใช้งานผ่านส่วนเชื่อมต่อของภาษาเอสดับยูอาร์แอล (SWRLAPI) เพื่อมาเรียก Drools Engine อีกครั้งหนึ่ง ซึ่งจะกล่าวในหัวข้อถัดไป

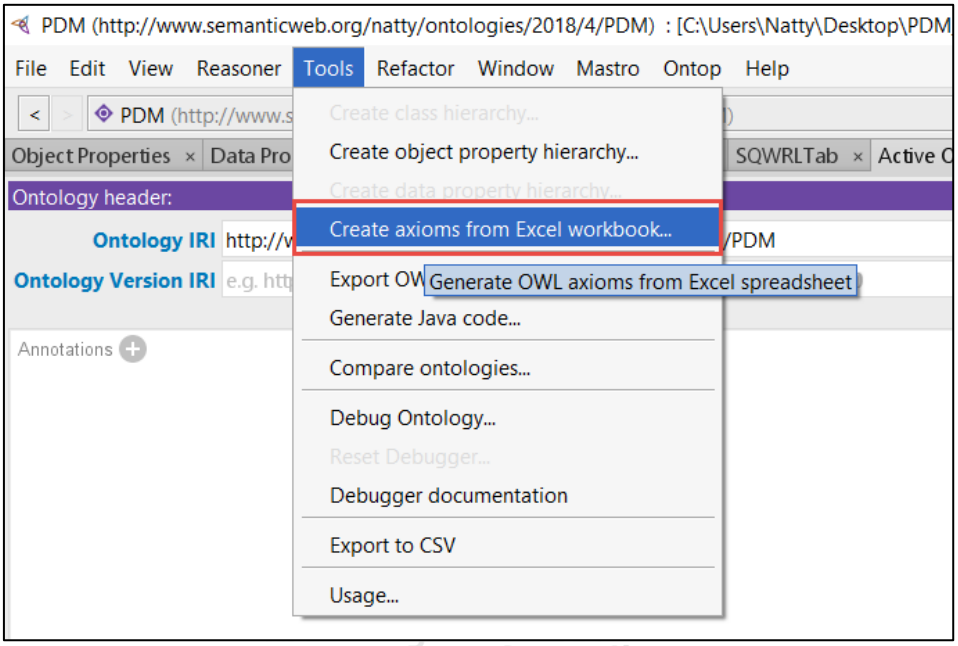


รูปที่ 4.10 การทดสอบประมวลผลกฎ SWRL ด้วยเครื่องมือ Drools ในโปรแกรมโปรเทเจ

กฎที่เขียนด้วยไวยากรณ์ภาษา SWRL และ SQWRL ต่าง ๆ เหล่านี้สามารถส่งออกให้อยู่ในรูปของแฟ้มข้อมูลภาษา OWL ในรูปแบบของ RDF/XML เพื่อนำไปใช้งานในโปรแกรมจาวาต่อไป

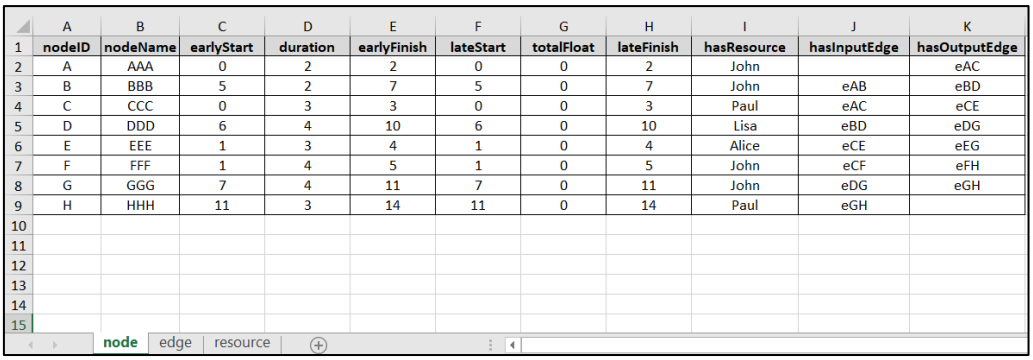
4.4 ออกแบบและพัฒนาส่วนของการนำเข้าข้อมูลไฟล์ Excel ด้วยเครื่องมือโปรเทเจ

เนื่องจากเครื่องมือโปรเทเจ เวอร์ชัน 5.2 ได้ถูกออกแบบและพัฒนาขึ้นมาให้สามารถรองรับการนำเข้าข้อมูลในรูปแบบของแฟ้มข้อมูล Excel ได้ โดยมีการติดตั้งส่วนเสริม (Plug-in) อยู่ในส่วนเรียกว่า Cellfie Plugin UI ซึ่งจะอยู่ภายใต้เมนูเครื่องมือ (Tools) >> “Create axioms from Excel workbook...” ในโปรแกรมโปรเทเจ ซึ่งจะช่วยอำนวยความสะดวกให้แก่ผู้พัฒนาสามารถสร้างแฟ้มข้อมูลภาษาอาวล์ โดยการนำเข้าข้อมูลจาก Excel ได้อย่างง่ายดาย



รูปที่ 4.11 ส่วนเสริม (Plug-in) ในการนำเข้าแฟ้มข้อมูล Excel ของเครื่องมือโปรเทเจ

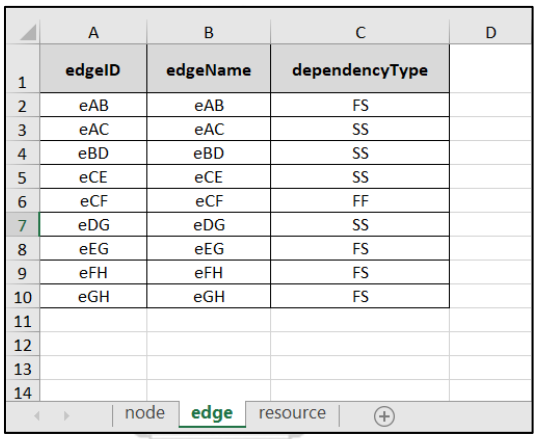
อย่างไรก็ตามในการสร้างออนโทโลยีโดยการนำเข้าข้อมูลจาก Excel จำเป็นต้องใช้ความรู้เรื่องการเขียนกฎภาษาอาวล์ด้วยรูปแบบของไวยากรณ์แมนเชสเตอร์ (Manchester Syntax) [34] เพื่อทำการแปลงข้อมูลในแต่ละคอลัมน์ใน Excel ให้เป็นสัจพจน์ (Axiom) ในภาษาอาวล์ ก่อนที่จะทำเข้าข้อมูลจำเป็นต้องทำการออกแบบโครงร่างของไฟล์ที่จะนำเข้า ซึ่งในงานวิจัยนี้จะนำเข้าเฉพาะข้อมูลส่วนที่เป็นอินแสตนซ์หรืออินดิวิดวล ที่ได้ทำการออกแบบไว้ ดังรูปที่ 4.12 – 4.14



	A	B	C	D	E	F	G	H	I	J	K
1	nodeID	nodeName	earlyStart	duration	earlyFinish	lateStart	totalFloat	lateFinish	hasResource	hasInputEdge	hasOutputEdge
2	A	AAA	0	2	2	0	0	2	John		eAC
3	B	BBB	5	2	7	5	0	7	John	eAB	eBD
4	C	CCC	0	3	3	0	0	3	Paul	eAC	eCE
5	D	DDD	6	4	10	6	0	10	Lisa	eBD	eDG
6	E	EEE	1	3	4	1	0	4	Alice	eCE	eEG
7	F	FFF	1	4	5	1	0	5	John	eCF	eFH
8	G	GGG	7	4	11	7	0	11	John	eDG	eGH
9	H	HHH	11	3	14	11	0	14	Paul	eGH	

node edge resource

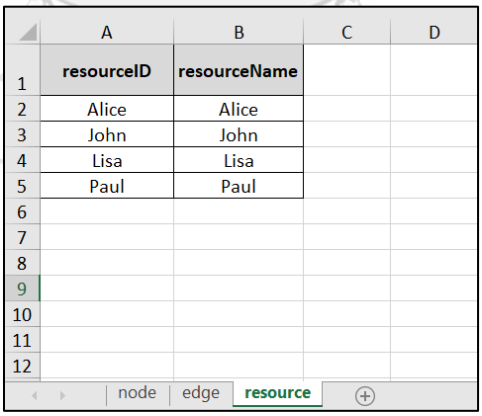
รูปที่ 4.12 รูปแบบข้อมูล Excel เพื่อนำเข้าข้อมูลโหนดกิจกรรม



	A	B	C	D
1	edgeID	edgeName	dependencyType	
2	eAB	eAB	FS	
3	eAC	eAC	SS	
4	eBD	eBD	SS	
5	eCE	eCE	SS	
6	eCF	eCF	FF	
7	eDG	eDG	SS	
8	eEG	eEG	FS	
9	eFH	eFH	FS	
10	eGH	eGH	FS	

node edge resource

รูปที่ 4.13 รูปแบบข้อมูล Excel เพื่อนำเข้าข้อมูลการพึ่งพาแต่ละกิจกรรม



	A	B	C	D
1	resourceID	resourceName		
2	Alice	Alice		
3	John	John		
4	Lisa	Lisa		
5	Paul	Paul		

node edge resource

รูปที่ 4.14 รูปแบบข้อมูล Excel เพื่อนำเข้าข้อมูลทรัพยากรบุคคล

ข้อมูลที่จะนำเข้านั้นจะต้องประกอบไปด้วยข้อมูลโหนดกิจกรรม (Node) ข้อมูลการพึ่งพาหรือเส้นเชื่อมระหว่างกิจกรรม (Edge) และข้อมูลเกี่ยวกับทรัพยากร (Resource) ซึ่งจะต้องสอดคล้องกับการออกแบบโคร่างร่างของออนโทโลยี ที่ประกอบไปด้วย 3 คลาส ได้แก่ คลาสโหนดกิจกรรม

คลาสเส้นเชื่อมระหว่างกิจกรรม และคลาสทรัพยากร ในแต่ละคลาสก็จะประกอบไปด้วยคุณลักษณะของข้อมูล (Data Properties) ต่าง ๆ ซึ่งคุณลักษณะเหล่านี้ ก็จะถูกนำเข้ามาจากแฟ้มข้อมูล Excel ตามรูปที่ 4.12 – 4.14 นั่นเอง

- nodeID ให้ระบุชื่ออินสแตนซ์ของแต่ละโหนดกิจกรรม ซึ่งจะต้องไม่ซ้ำกัน
- nodeName ให้ระบุชื่อกิจกรรมของแต่ละโหนดกิจกรรม
- earlyStart ให้ระบุเวลาเริ่มต้นที่เร็วที่สุดของกิจกรรม
- duration ให้ระบุระยะเวลาในการดำเนินงานของกิจกรรม
- earlyFinish ให้ระบุเวลาสิ้นสุดที่เร็วที่สุดของกิจกรรม
- lateStart ให้ระบุเวลาเริ่มต้นที่ช้าที่สุดของกิจกรรม
- totalFloat ให้ระบุเวลาลอยรวมของกิจกรรม
- lateFinish ให้ระบุเวลาสิ้นสุดที่เร็วที่สุดของกิจกรรม
- hasResource ให้ระบุชื่อของทรัพยากรที่ได้รับมอบหมายให้ทำกิจกรรมนั้น ๆ
- inputEdge ให้ระบุชื่อเส้นเชื่อมที่วิ่งเข้าหาโหนดกิจกรรม
- outputEdge ให้ระบุชื่อเส้นเชื่อมที่วิ่งออกจากโหนดกิจกรรม

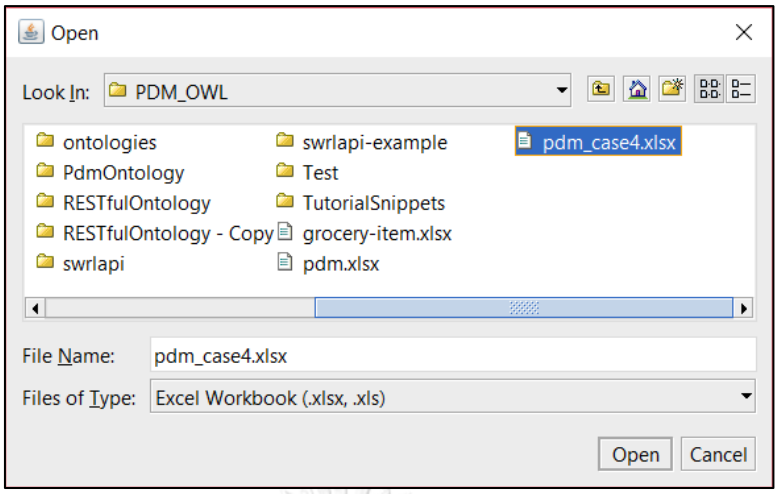
สำหรับการนำเข้าข้อมูลการพึ่งพาแต่ละกิจกรรมของแฟ้มข้อมูล Excel ดังรูปที่ 4.13 มีรายละเอียด ดังนี้

- edgeID ให้ระบุชื่ออินสแตนซ์ของการพึ่งพา ซึ่งจะต้องไม่ซ้ำกัน
- edgeName ให้ระบุเชื่อของเส้นการพึ่งพา เช่น หากเป็นเส้นเชื่อมระหว่างโหนด A ไปโหนด B อาจจะตั้งชื่อว่า eAB เป็นต้น
- dependencyType ให้ระบุประเภทของการพึ่งพาของเส้นเชื่อมที่วิ่งเข้าหาโหนดกิจกรรม

และสุดท้ายคือรูปแบบของการนำเข้าข้อมูลทรัพยากรบุคคลจากแฟ้มข้อมูล Excel ดังรูปที่ 4.14 มีรายละเอียด ดังนี้

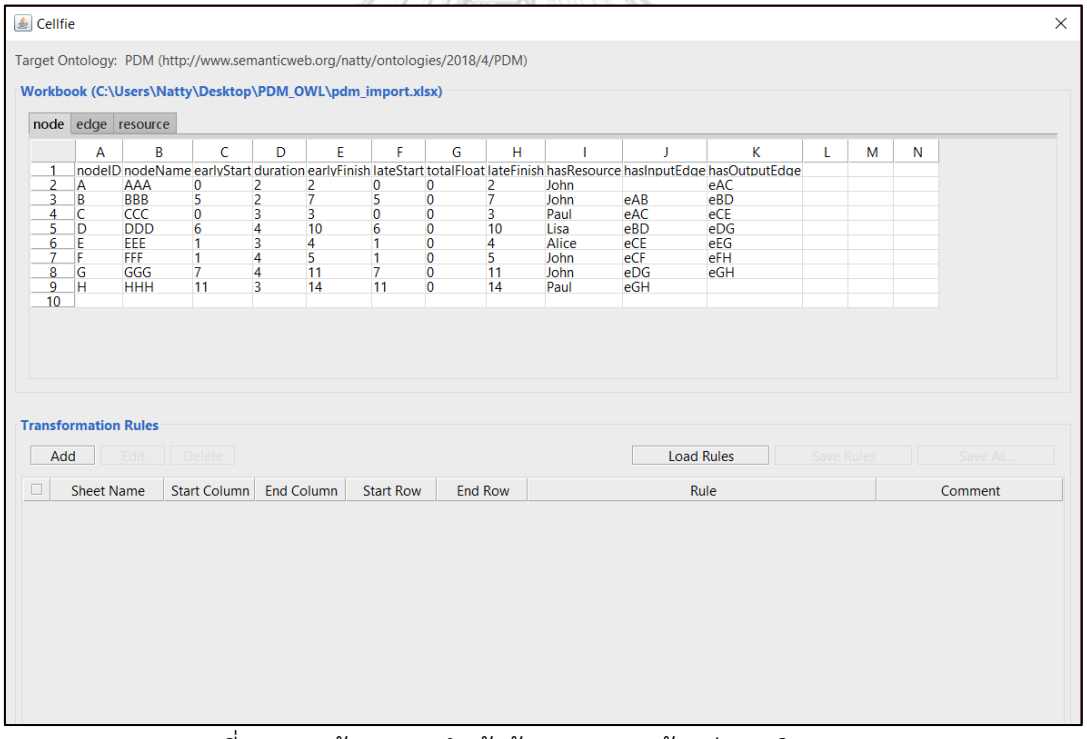
- resourceID ให้ระบุชื่ออินสแตนซ์ของทรัพยากรบุคคล ซึ่งจะต้องไม่ซ้ำกัน
- resourceName ให้ระบุชื่อผู้รับผิดชอบของโหนดกิจกรรม

หลังจากที่มีการจัดเตรียมแฟ้มข้อมูล Excel เรียบร้อยแล้ว ขั้นตอนต่อมาจะเป็นการเรียกใช้งานเครื่องมือโปรเทเจ โดยการคลิกที่เมนู Tool และเลือก Create axioms from Excel workbook... ดังรูปที่ 4.11 จากนั้นโปรแกรมจะแสดงหน้าต่างขึ้นมาให้เราเลือกไฟล์ที่ต้องการจะนำเข้า ให้เลือกไฟล์ที่ได้จัดเตรียมไว้ จากนั้นกดปุ่ม Open ดังรูปที่ 4.15



รูปที่ 4.15 หน้าจอเลือกไฟล์ Excel ที่จะนำเข้า

เมื่อคลิกที่ปุ่ม Open แล้ว โปรแกรมจะแสดงหน้าต่าง Cellfie ซึ่งเป็นส่วนเสริมที่ติดตั้งมาพร้อมกับโปรแกรมโปรเทเจ 5.2 ดังรูปที่ 4.16



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	nodeID	nodeName	earlyStart	duration	earlyFinish	lateStart	totalFloat	lateFinish	hasResource	hasInputEdge	hasOutputEdge			
2	A	AAA	0	2	2	0	0	2	John		eAC			
3	B	BBB	5	2	7	5	0	7	John	eAB	eBD			
4	C	CCC	0	3	3	0	0	3	Paul	eAC	eCE			
5	D	DDD	6	4	10	6	0	10	Lisa	eBD	eDG			
6	E	EEE	1	3	4	1	0	4	Alice	eCE	eEG			
7	F	FFF	1	4	5	1	0	5	John	eCF	eFH			
8	G	GGG	7	4	11	7	0	11	John	eDG	eGH			
9	H	HHH	11	3	14	11	0	14	Paul	eGH				
10														

รูปที่ 4.16 หน้าจอการนำเข้าข้อมูล Excel ด้วยส่วนเสริม Cellfie

ในหน้าจอ Cellfie จะแสดงข้อมูลเป็นแท็บและตารางคล้ายกับข้อมูลใน Excel ซึ่งด้านล่างจะมีส่วนของ Transformation Rules ให้ผู้ใช้งานทำการเขียนกฎเพื่อแปลงข้อมูลในแต่ละคอลัมน์ใน

Excel ให้เป็นอินดิวิดวล/อินแสตนซ์ในภาษาอาวล์ ในการสร้างกฎให้คลิกปุ่ม ‘Add’ ในส่วนของ Transformation Rules ซึ่งผู้วิจัยได้ทำการออกแบบและสร้างกฎดังกล่าว โดยแบ่งเป็น 3 กฎดังนี้

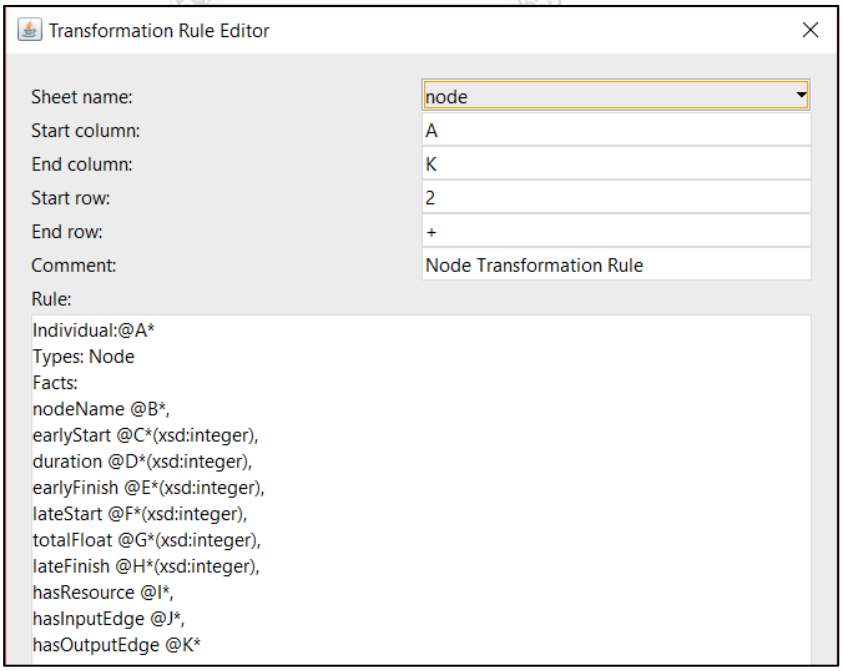
1) กฎการแปลงโหนดกิจกรรม (Node Transformation Rule)

2) กฎการแปลงเส้นเชื่อมระหว่างกิจกรรม (Edge Transformation Rule)

3) กฎการแปลงทรัพยากรบุคคล (Resource Transformation Rule)

ซึ่งทั้ง 3 กฎนี้ จะมีส่วนที่มีการกำหนดค่าเริ่มต้นคล้ายกันดังนี้

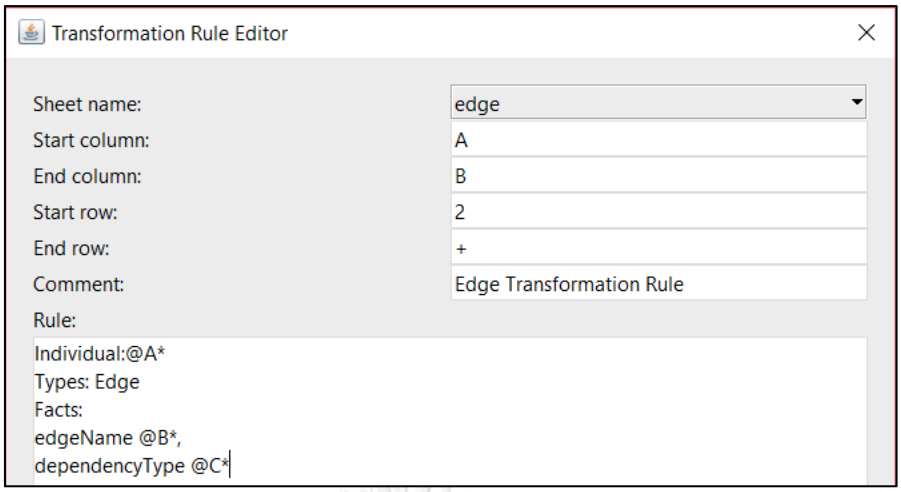
- Sheet name คือ ชีทในไฟล์ Excel ที่ต้องการจะนำข้อมูลเข้ามาแปลงกับกฎ
- Start column คือ คอลัมน์เริ่มต้นที่จะทำการแปลง
- End column คือ คอมลัมน์สุดท้ายที่จะทำการแปลง
- Start row คือ บรรทัดเริ่มต้นที่จะทำการแปลง โดยจะเริ่มต้นที่บรรทัดที่ 2 เนื่องจากบรรทัดแรกเป็นชื่อคอลัมน์
- End row คือ บรรทัดสุดท้ายที่จะทำการแปลง ในที่นี้ให้กำหนดเป็นค่า + หมายถึงค่าในบรรทัดสุดของของคอลัมน์ใน Excel
- Comment คือ คำอธิบายเพิ่มเติมเกี่ยวกับกฎ
- Rule คือ ส่วนของกฎที่เขียนอยู่ในรูปแบบไวยากรณ์แมนเชสเตอร์ ซึ่งแต่ละกฎจะมีการเขียนที่แตกต่างกันไป



รูปที่ 4.17 กฎการแปลงโหนดกิจกรรม (Node Transformation Rule)

จากกฎการแปลงโหนดกิจกรรม (Node Transformation Rule) ในรูปที่ 4.17 สามารถอธิบายได้ดังนี้

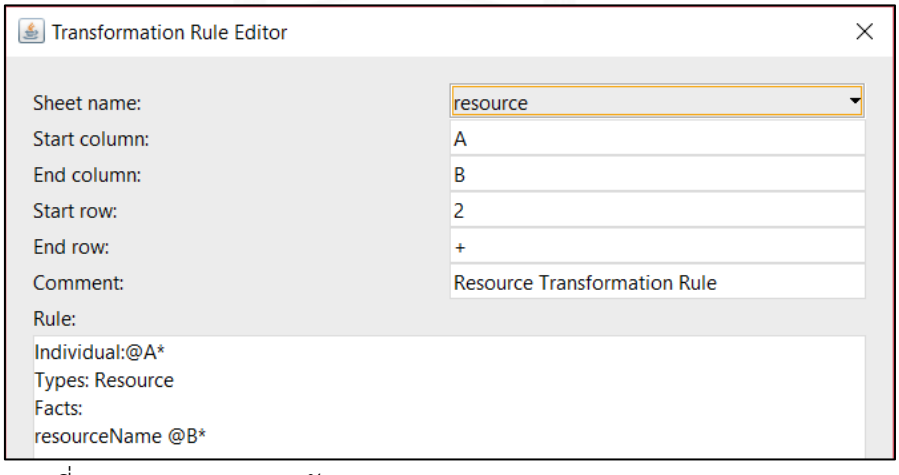
- Individual:@A*	หมายถึง	การเพิ่มข้อมูลอินดิวิดวลตั้งแต่คอลัมน์ A จนถึง คอลัมน์สุดท้าย
- Types: Node	หมายถึง	คลาสที่ต้องการจะเพิ่มอินดิวิดวล ในกฎนี้คือ คลาสโหนด
- Facts:	หมายถึง	รายละเอียดที่จะทำการแปลง
- nodeName @B*,	หมายถึง	แปลงข้อมูลในคอลัมน์ B ไปเป็นอินดิวิดวลของคลาสโหนดที่มีคุณลักษณะข้อมูล (Data property) ที่ชื่อว่า ‘nodeName’
- earlyStart @C*(xsd:integer),	หมายถึง	แปลงข้อมูลในคอลัมน์ C ไปเป็นอินดิวิดวลของคลาสโหนดที่มีคุณลักษณะข้อมูล (Data property) ที่ชื่อว่า ‘earlyStart’
- duration @D*(xsd:integer),	หมายถึง	แปลงข้อมูลในคอลัมน์ D ไปเป็นอินดิวิดวลของคลาสโหนดที่มีคุณลักษณะข้อมูล (Data property) ที่ชื่อว่า ‘duration’ ซึ่งมีรูปแบบเป็นข้อมูลเป็นเลขจำนวนเต็มบวก
- earlyFinish @E*(xsd:integer),	หมายถึง	แปลงข้อมูลในคอลัมน์ E ไปเป็นอินดิวิดวลของคลาสโหนดที่มีคุณลักษณะข้อมูล (Data property) ที่ชื่อว่า ‘earlyFinish’ ซึ่งมีรูปแบบเป็นข้อมูลเป็นเลขจำนวนเต็มบวก
- lateStart @F*(xsd:integer),	หมายถึง	แปลงข้อมูลในคอลัมน์ F ไปเป็นอินดิวิดวลของคลาสโหนดที่มีคุณลักษณะข้อมูล (Data property) ที่ชื่อว่า ‘lateStart’ ซึ่งมีรูปแบบเป็นข้อมูลเป็นเลขจำนวนเต็มบวก
- totalFloat @G*(xsd:integer),	หมายถึง	แปลงข้อมูลในคอลัมน์ G ไปเป็นอินดิวิดวลของคลาสโหนดที่มีคุณลักษณะข้อมูล (Data property) ที่ชื่อว่า ‘totalFloat’ ซึ่งมีรูปแบบเป็นข้อมูลเป็นเลขจำนวนเต็มบวก
- lateFinish @H*(xsd:integer),	หมายถึง	แปลงข้อมูลในคอลัมน์ H ไปเป็นอินดิวิดวลของคลาสโหนดที่มีคุณลักษณะข้อมูล (Data property) ที่ชื่อว่า ‘lateFinish’ ซึ่งมีรูปแบบเป็นข้อมูลเป็นเลขจำนวนเต็มบวก
- hasResource @I*,	หมายถึง	แปลงข้อมูลในคอลัมน์ I ไปเป็นอินดิวิดวลของคลาสโหนดที่มีคุณลักษณะข้อมูล (Data property) ที่ชื่อว่า ‘hasResource’
- hasInputEdge @J*,	หมายถึง	แปลงข้อมูลในคอลัมน์ J ไปเป็นอินดิวิดวลของคลาสโหนดที่มีคุณลักษณะข้อมูล (Data property) ที่ชื่อว่า ‘hasInputEdge’
- hasOutputEdge @K*,	หมายถึง	แปลงข้อมูลในคอลัมน์ K ไปเป็นอินดิวิดวลของคลาสโหนดที่มีคุณลักษณะข้อมูล (Data property) ที่ชื่อว่า ‘hasOutputEdge



รูปที่ 4.18 กฎการแปลงเส้นเชื่อมระหว่างกิจกรรม (Edge Transformation Rule)

จากกฎการแปลงเส้นเชื่อม (Edge Transformation Rule) ในรูปที่ 4.18 สามารถอธิบายได้ดังนี้

- Individual:@A* หมายถึง การเพิ่มข้อมูลอินดิวิดวลตั้งแต่คอลัมน์ A จนถึง คอลัมน์สุดท้าย
- Types: Edge หมายถึง คลาสที่ต้องการจะเพิ่มอินดิวิดวล ในกฎนี้คือ คลาสเส้นเชื่อม
- Facts: หมายถึง รายละเอียดที่จะทำการแปลง
- edgeName @B*, หมายถึง แปลงข้อมูลในคอลัมน์ B ไปเป็นอินดิวิดวลของคลาสเส้นเชื่อมที่มีคุณลักษณะข้อมูล (Data property) ที่ชื่อว่า ‘edgeName’
-dependencyType @C*, หมายถึง แปลงข้อมูลในคอลัมน์ C ไปเป็นอินดิวิดวลของคลาสเส้นเชื่อมที่มีคุณลักษณะข้อมูล (Data property) ที่ชื่อว่า ‘dependencyType’

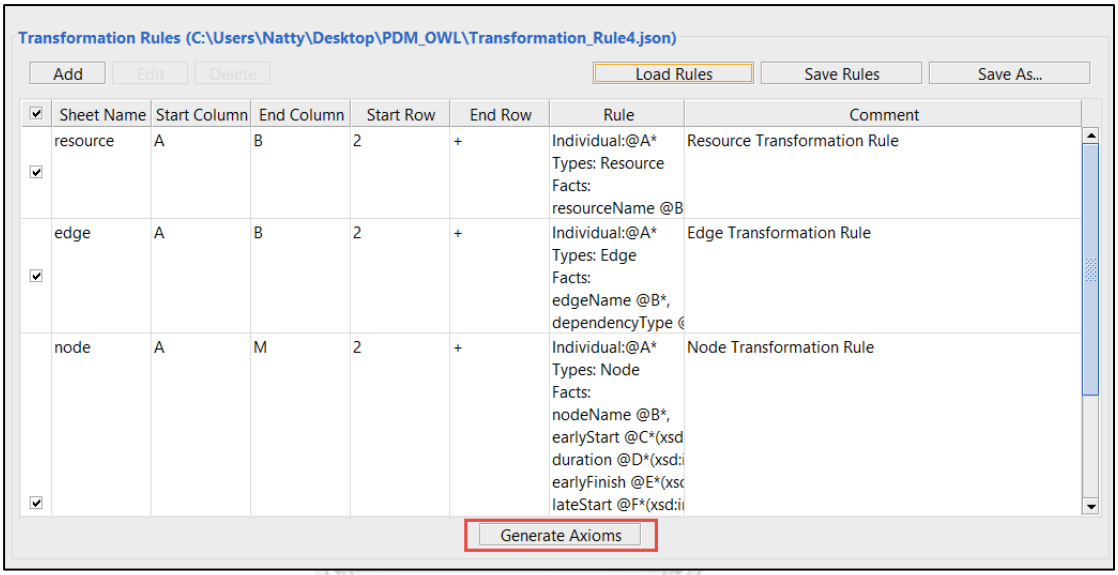


รูปที่ 4.19 กฎการแปลงทรัพยากรบุคคล (Resource Transformation Rule)

จากกฎการแปลงทรัพยากรบุคคล (Resource Transformation Rule) ในรูปที่ 4.19 สามารถอธิบายได้ดังนี้

- Individual:@A*	หมายถึง	การเพิ่มข้อมูลอินดิวิดวลตั้งแต่คอลัมน์ A จนถึง คอลัมน์สุดท้าย
- Types: Edge	หมายถึง	คลาสที่ต้องการจะเพิ่มอินดิวิดวล ในกฎนี้คือ คลาสทรัพยากรบุคคล
- Facts:	หมายถึง	รายละเอียดที่จะทำการแปลง
- resourceName @B*,	หมายถึง	แปลงข้อมูลในคอลัมน์ B ไปเป็นอินดิวิดวลของคลาสทรัพยากรที่มีคุณลักษณะข้อมูล (Data property) ที่ชื่อว่า ‘resourceName

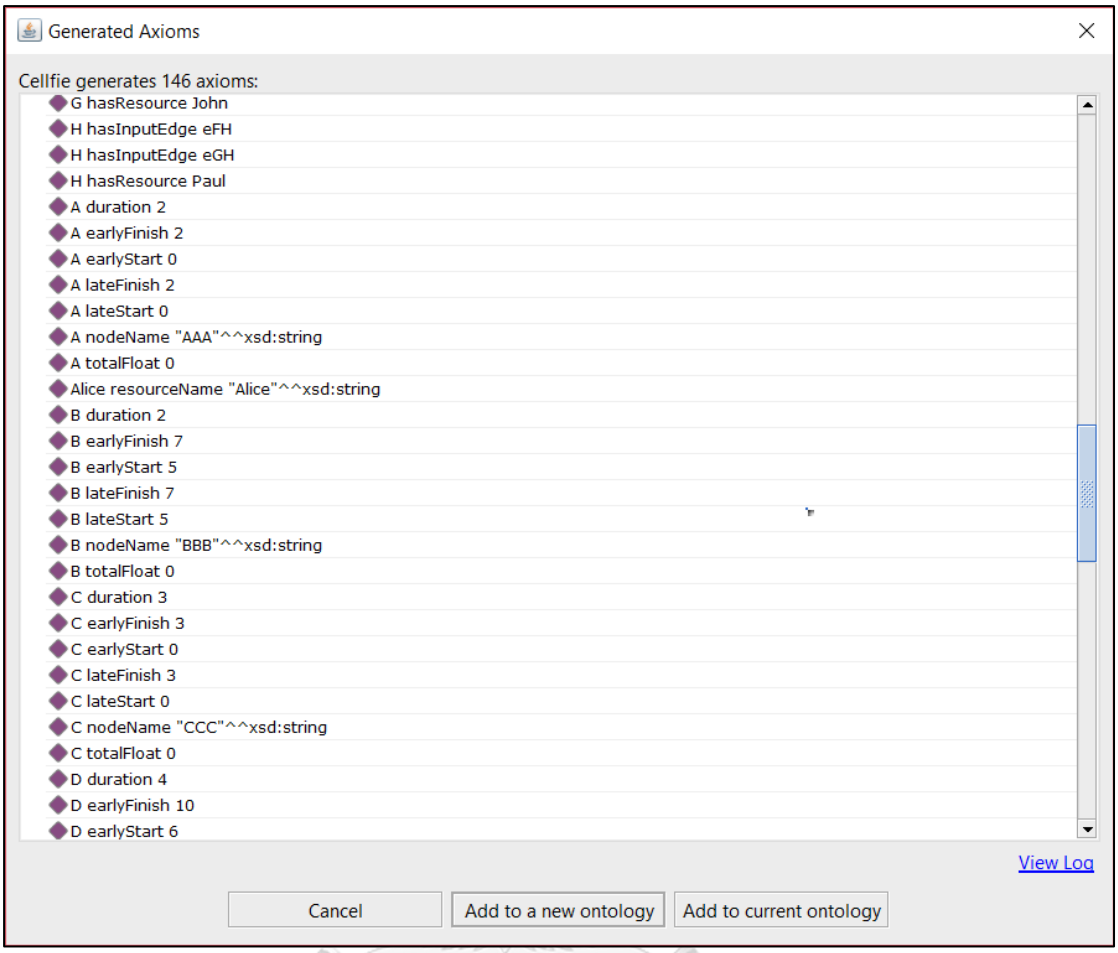
เมื่อเพิ่มกฎการแปลงข้อมูล Excel ไปเป็นอินดิวิดวลในออนโทโลยีเรียบร้อยแล้ว ก็จะสามารถเพิ่มข้อมูลดังกล่าวเข้าสู่ออนโทโลยีได้ด้วยการกดปุ่ม ‘Generate Axioms’ ตามรูปที่ 4.20



รูปที่ 4.20 การสร้างสัจพจน์เพื่อนำข้อมูลเข้าออนโทโลยี

จากนั้นโปรแกรมจะทำการสร้างข้อความสัจพจน์ให้โดยอัตโนมัติ ดังรูปที่ 4.21 พร้อมกับให้ผู้ใช้งานเลือกว่าจะทำการเพิ่มข้อมูลเหล่านี้ในออนโทโลยีใหม่ (Add to a new ontology) หรือจะทำการเพิ่มต่อจากออนโทโลยีที่กำลังเปิดอยู่ (Add to current ontology) ในที่นี้ให้เลือกเพิ่มต่อจากออนโทโลยีที่กำลังเปิดอยู่

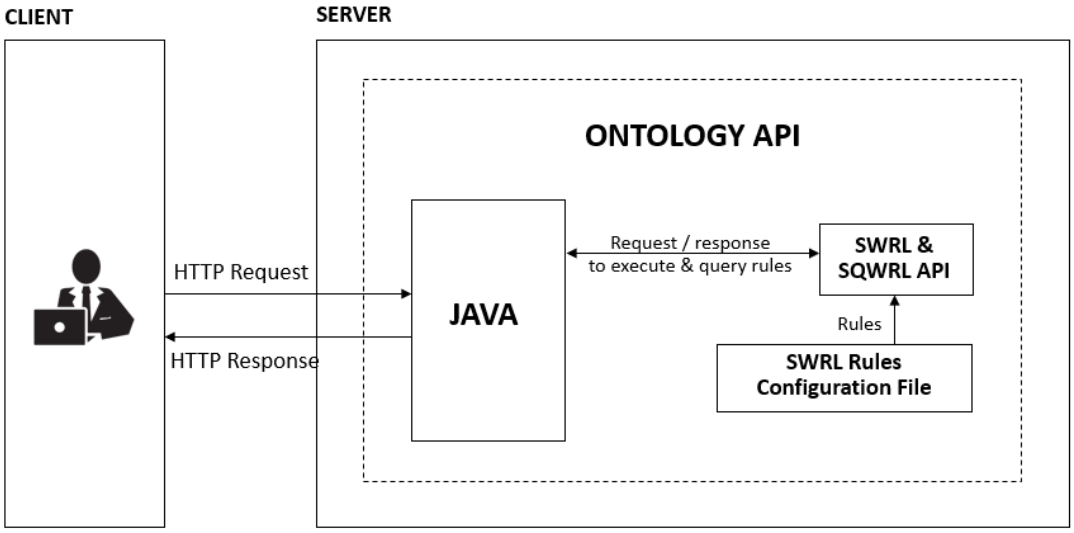
เมื่อทำการนำเข้าข้อมูลจากไฟล์ Excel เข้าสู่ออนโทโลยีเรียบร้อยแล้ว เครื่องมือโปรเทเจจะสร้างไฟล์ข้อมูลภาษาอาวล์ ที่อยู่ในรูปไวยากรณ์ภาษา RDF/XML ให้โดยอัตโนมัติ ซึ่งผู้ใช้งานสามารถคัดลอกไฟล์ดังกล่าว แล้วนำไป Import เข้าสู่เว็บแอพพลิเคชัน ในหน้าจอของผู้ใช้งาน (Client) ต่อไปได้ ซึ่งจะกล่าวในหัวข้อถัดไป



รูปที่ 4.21 โปรแกรมสร้างข้อความสัจพจน์ให้โดยอัตโนมัติ

4.5 ออกแบบและพัฒนาระบบเว็บแอพพลิเคชัน

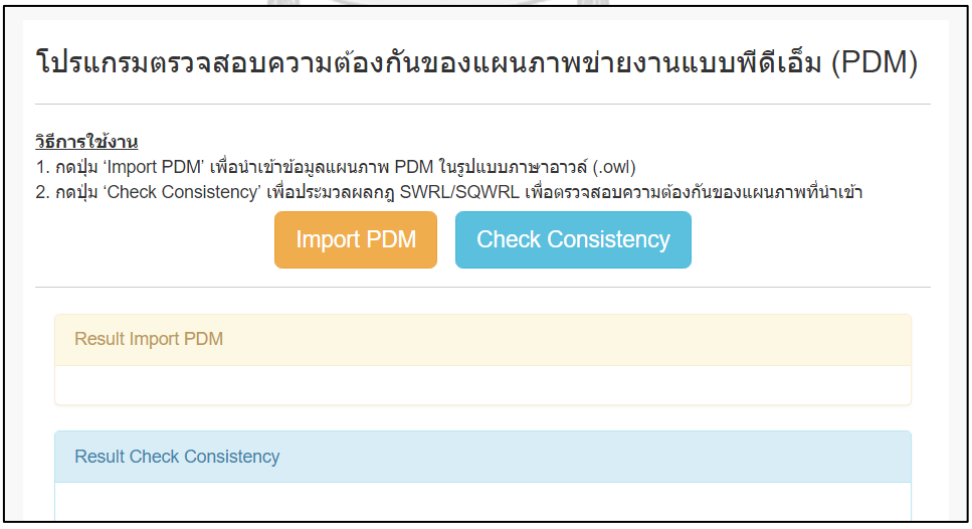
การออกแบบสถาปัตยกรรมระบบตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม ด้วยออนโทโลยีประยุกต์ใช้เทคโนโลยีเว็บแอพพลิเคชัน โดยแบ่งการทำงานออกเป็น 2 ส่วนหลัก ๆ คือ ส่วนผู้ใช้งาน (Client) และส่วนของเครื่องแม่ข่าย (Server) ดังรูปที่ 4.22



รูปที่ 4.22 ภาพรวมของสถาปัตยกรรมระบบ

4.5.1 ส่วนผู้ใช้งาน (Client)

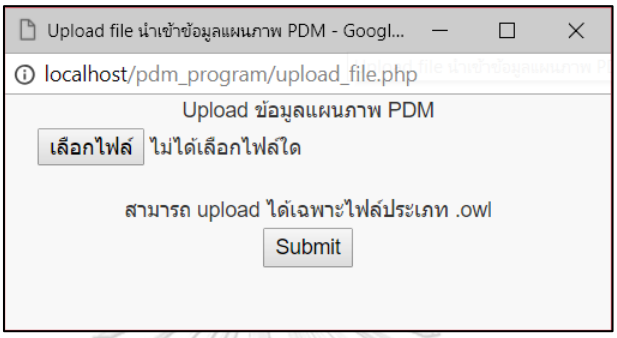
ในส่วนของผู้ใช้งาน ผู้วิจัยได้ออกแบบให้สามารถเรียกใช้งานผ่านหน้าเว็บแอพพลิเคชัน โดยมีฟังก์ชันการทำงานหลัก 2 ส่วนคือ 1) ส่วนที่นำเข้าข้อมูล (Import PDM) โดยมีการออกแบบให้สามารถนำเข้าข้อมูลในรูปแบบ OWL file ที่ได้จากเครื่องมือโปรเทเจ และ 2) ส่วนตรวจสอบความต้องกัน (Check Consistency) ดังรูปที่ 4.23



รูปที่ 4.23 หน้าจอโปรแกรมตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม (PDM)

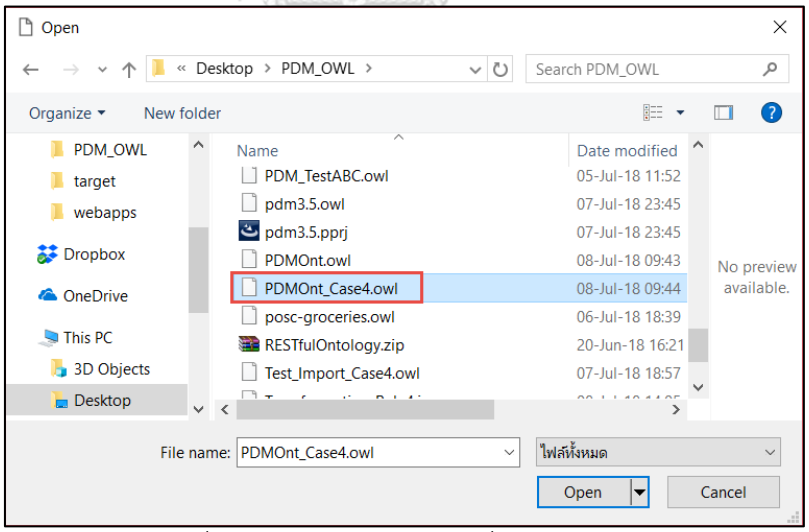
ในหน้าจอโปรแกรมตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม (PDM) จะอธิบายวิธีการใช้งานเบื้องต้น โดยมี 2 ขั้นตอนดังนี้

ขั้นตอนที่ 1. กดปุ่ม ‘Import PDM’ เพื่อนำเข้าข้อมูลแผนภาพ PDM ในรูปแบบ OWL file (.owl) ที่ได้มีการประมวลผลด้วยเครื่องมือโปรเทเจ โดยเมื่อทำการกดปุ่มดังกล่าวแล้ว ระบบจะแสดงหน้าต่างให้ Upload ข้อมูลเข้า



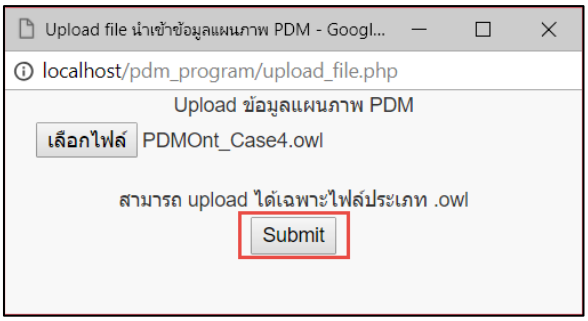
รูปที่ 4.24 หน้าจอ Upload ข้อมูลแผนภาพ PDM

จากนั้นให้ผู้ใช้งานกดปุ่ม ‘เลือกไฟล์’เพื่อเลือกไฟล์ที่ต้องการนำเข้า โดยที่ข้อมูลนำเข้าจะต้องอยู่ในรูปแบบที่กำหนดเท่านั้น ดังรูปที่ 4.13



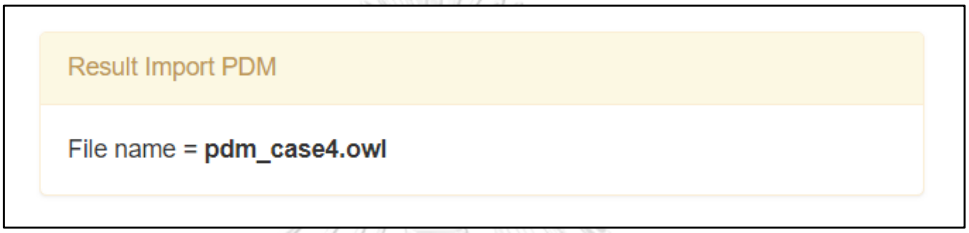
รูปที่ 4.25 หน้าจอเลือกไฟล์เพื่อนำข้อมูลเข้า

เมื่อเลือกข้อมูลนำเข้าที่ต้องการเรียบร้อยแล้วจึงกดปุ่ม ‘Submit เพื่อยืนยันการ Upload file



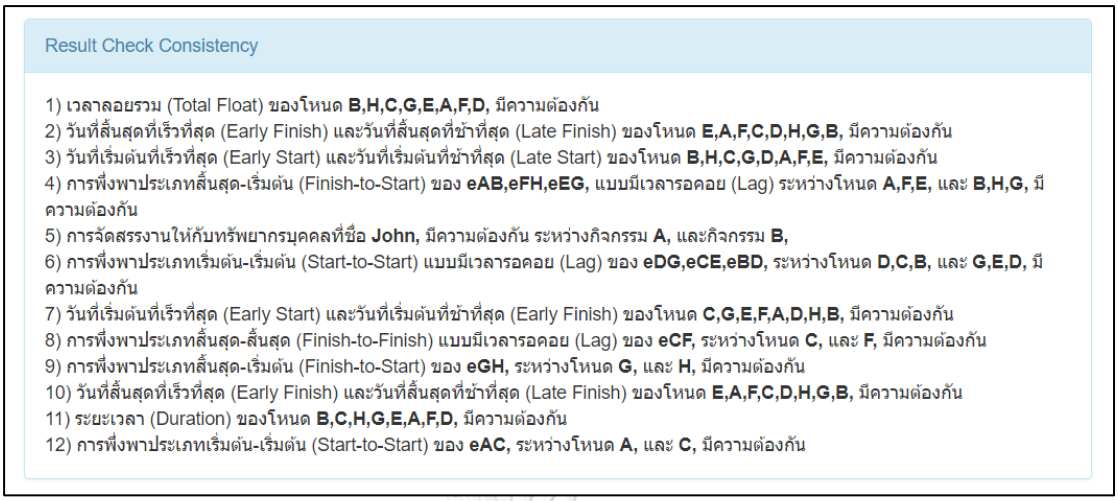
รูปที่ 4.26 หน้าจอยืนยันการเลือกข้อมูลนำเข้า

หากทำการ Upload สมบูรณ์ในช่องแสดงผล จะแสดงชื่อไฟล์ที่นำเข้า ดังรูปที่ 4.27



รูปที่ 4.27 หน้าจอแสดงการนำเข้าข้อมูลสำเร็จ

ขั้นตอนที่ 2 เป็นการตรวจสอบความต้องกันของแผนภาพ โดยผู้ใช้งานสามารถกดปุ่ม ‘Check Consistency’ เพื่อประมวลผลกฎ SWRL/SQWRL เพื่อตรวจสอบความต้องกันของแผนภาพที่นำเข้า เมื่อผู้ใช้งานกดปุ่มดังกล่าว จะเป็นการส่งข้อมูลในรูปแบบ HTTP Request ไปยังส่วนของเครื่องแม่ข่ายเพื่อรับข้อมูลเข้าและนำไปประมวลผลต่อไป ซึ่งจะกล่าวในรายละเอียดในหัวข้อ 4.4.2 ส่วนเครื่องแม่ข่าย เมื่อทางฝั่งเครื่องแม่ข่ายประมวลผลเสร็จสิ้นแล้ว จะส่งคืนผลลัพธ์การตรวจสอบกลับในรูปแบบขอ HTTP Response กลับมาแสดงผลให้ผู้ใช้งานทางหน้าจอในช่องแสดงผล ‘Result Check Consistency’ ดังตัวอย่างรูปที่ 4.28



รูปที่ 4.28 หน้าจอแสดงผลการตรวจสอบความต้องกันของข้อมูลนำเข้า

4.5.2 ส่วนเครื่องแม่ข่าย (Server)

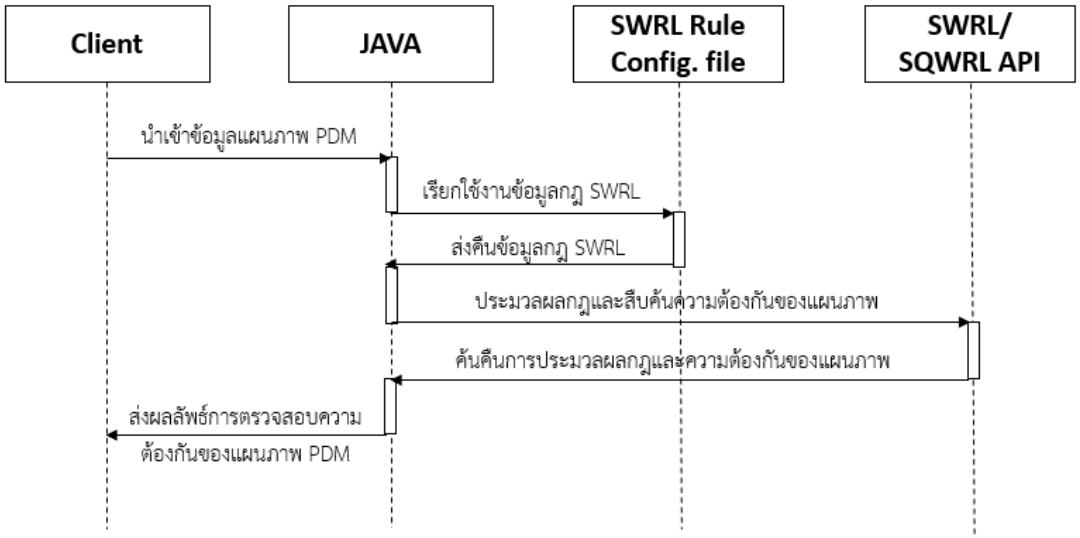
ในส่วนของเครื่องแม่ข่ายจะเป็นการพัฒนาส่วนที่รับข้อมูลนำเข้ามาจากส่วนผู้ใช้งาน (Client) โดยใช้ใช้ส่วนเชื่อมต่อของโปรแกรมภาษาจาวา (Java API) เพื่อเชื่อมต่อกับส่วนเชื่อมต่ออื่น ๆ ที่จำเป็นต่อการสร้างออนโทโลยีและการประมวลผลกฎ ซึ่งประกอบด้วย 3 ส่วนย่อย ดังนี้

1) จาวา (Java) [35] เป็นส่วนสำคัญที่ทำหน้าที่เป็นตัวกลางในการเชื่อมต่อกับส่วนเชื่อมต่อ (API) ต่าง ๆ ที่จำเป็นต้องเรียกใช้งาน โดยจะทำการรับข้อมูลนำเข้าจากส่วนผู้ใช้งาน (Client) และทำหน้าที่ในการนำข้อมูลกฎที่ถูกจัดเก็บในส่วนจัดเก็บการตั้งค่าของกฎ (SWRL Rules Configuration File) มาจัดทำให้อยู่ในรูปแบบที่สามารถส่งข้อมูลไปประมวลผลต่อที่ SWRL API เพื่อประมวลผลกฎในการตรวจสอบความต้องกัน จากนั้นรับค่าผลลัพธ์มาจัดให้อยู่รูปแบบที่เหมาะสม เพื่อแสดงผลกลับไปให้ผู้ใช้งานทางหน้าจอ

2) ส่วนจัดเก็บการตั้งค่าของกฎ (SWRL Rules Configuration File) ทำหน้าที่จัดเก็บกฎภาษา SWRL ที่ใช้ในการตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม ซึ่งจัดเก็บอยู่ในรูปของ Configuration file เพื่อให้สะดวกต่อการบำรุงรักษาในอนาคต ในกรณีที่มีการเพิ่ม ลด หรือแก้ไข กฎต่าง ๆ

3) ส่วนเชื่อมต่อภาษาเอสดับเบิลยูอาร์แอล (SWRL API) [36] และส่วนเชื่อมต่อภาษาเอสคิวดับเบิลยูอาร์แอล (SQWRL API) [37] ทำหน้าที่ประมวลผลกฎตามที่ถูกจัดเก็บอยู่ใน SWRL Rules Configuration File อีกทั้งยังสามารถสืบค้นข้อมูลที่ได้จากการประมวลผลกฎ และส่งคืนค่ากลับมาให้จาวา เพื่อส่งกลับไปยังส่วนผู้ใช้งาน

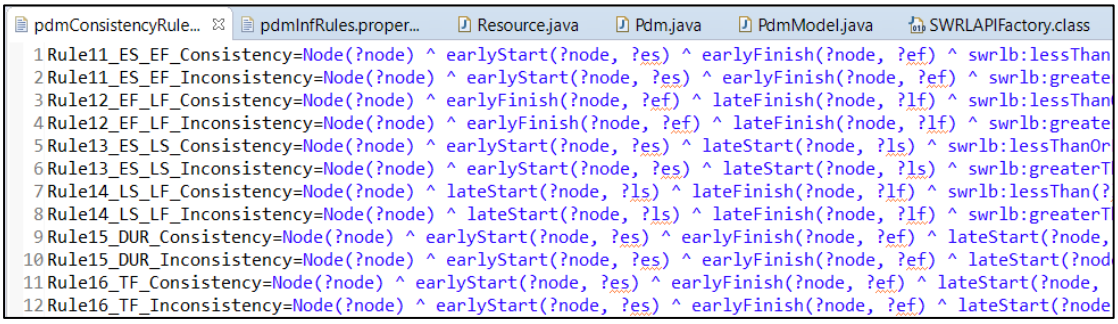
ในการออกแบบ Ontology API สามารถอธิบายขั้นตอนการรับส่งข้อมูลระหว่างส่วนเชื่อมต่อต่าง ๆ ด้วย Sequence diagram ดังรูปที่ 4.29



รูปที่ 4.29 Sequence diagram อธิบายการรับส่งข้อมูลระหว่างส่วนเชื่อมต่อต่าง ๆ ภายใน Ontology API

เริ่มต้นจากผู้ใช้งานนำเข้าข้อมูลแผนภาพ PDM ในรูปแบบ OWL file เข้าสู่ระบบผ่านทางส่วนผู้ใช้งาน (Client) ด้วย HTTP Request มายังฝั่งเครื่องแม่ข่าย (Server)

ในขั้นตอนถัดมา โปรแกรมจาวา จะทำหน้าที่เรียกใช้งานกฎจากส่วนจัดเก็บการตั้งค่าของกฎ (SWRL Rules Configuration File) เพื่อดึงค่ากฎมาเก็บไว้ก่อนที่จะส่งไปประมวลพร้อมกับออนโทโลยีที่นำเข้ามาในรูปแบบของภาษาอาวล์ ซึ่งกฎที่จะถูกนำมาประมวลผลนั้นถูกจัดเก็บให้อยู่ในรูปแบบของ Configuration File ดังรูปที่ 4.30



รูปที่ 4.30 ชุดกฎเพื่อตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม ที่ถูกจัดเก็บใน SWRL Rules Configuration File

```
public  ArrayList<String>  checkConsistency(String propName, String
source) {
       //System.out.println("getData");
       Map<String, String> map =  extractProperties(propName);
        //System.out.println(map);

        List keys = new ArrayList(map.keySet());
        List values = new ArrayList(map.values());
        ArrayList<String> consistencyList = new ArrayList<String>();
        ArrayList<String> resultArr = new ArrayList<String>();

        try {
             // Create an OWL Ontology using the OWLAPI
             String realPath = pdmContext.getRealPath("/WEB-INF/" +
source);
            OWLOntologyManager ontologyManager =
OWLManager.createOWLOntologyManager();
            OWLOntology ontology =
ontologyManager.loadOntologyFromOntologyDocument(new File(realPath));

            // Create SQWRL query engine using the SWRLAPI
            SQWRLQueryEngine queryEngine =
SWRLAPIFactory.createSQWRLQueryEngine(ontology);

            for (int i = 0; i < map.size(); i++) {
                String key =  keys.get(i).toString();
                String value = values.get(i).toString();

                queryEngine.createSQWRLQuery(key,value);
                SQWRLResult result = queryEngine.runSQWRLQuery(key,
value);

                if(result.next()){
                    consistencyList.add(key);
                }
            }

        }
```


รูปที่ 4.31 คำสั่งภาษาจาวาที่นำเข้าข้อมูลกฎจาก SWRL Rules Configuration File และส่งไปประมวลผลที่ SWRLAPI

จากรูปที่ 4.31 โปรแกรมจาวาจะส่งข้อมูลออนโทโลยีที่ผู้ใช้งานทำการ Import เข้ามา พร้อมกับชุดกฎไปประมวลผลด้วย SWRL API หลังจากการประมวลผลด้วย Rule Engine ที่อยู่ภายใต้ SWRL API แล้วระบบจะส่งคืนผลลัพธ์ของการตรวจสอบความต้องกันกลับมาที่โปรแกรมจาวา จากนั้นโปรแกรมจาวาจะนำมาจัดรูปแบบของข้อความที่จะส่งออก ไปยังหน้าจอของผู้ใช้งาน

บทที่ 5
การประเมินและการวัดผล

งานวิจัยการตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็มโดยใช้ออนโทโลยี มีวัตถุประสงค์ในการวิจัย ดังนี้

1. เพื่อนำเสนอกฎในการตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม
2. เพื่อพัฒนาเครื่องมือในการตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม

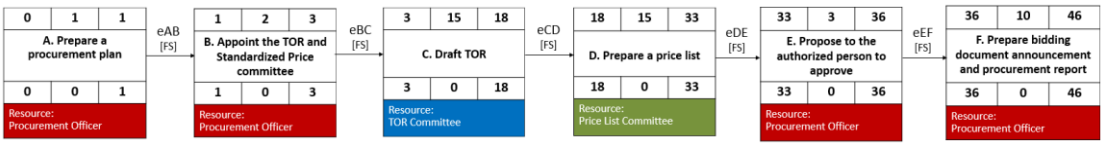
เพื่อประเมินและวัดผลกฎที่ได้นำเสนอ รวมถึงเครื่องมือที่พัฒนา ตามวัตถุประสงค์ข้างต้น ผู้วิจัยได้ใช้กรณีศึกษา 4 กรณี ซึ่งเป็นแผนภาพข่ายงานแบบพีดีเอ็ม 4 แผนภาพ โดยที่ 3 กรณีแรกจะเป็นแผนภาพแสดงกิจกรรมในการดำเนินการจัดซื้อจัดจ้างของหน่วยงานภาครัฐ ในกรณีต่าง ๆ ส่วนในกรณีสุดท้ายนั้น จะเป็นแผนภาพที่ผู้วิจัยได้จำลองขึ้นมาเพื่อทดสอบการทำงานของเครื่องมือในกรณีที่มีการกำหนดการพึ่งพาของกิจกรรมในทุกรูปแบบภายใต้แผนภาพเดียวกัน ซึ่งเป็นกรณีที่มักจะไม่เกิดขึ้นในการวางแผนงานโดยทั่วไป

โดยที่กรณีศึกษาที่ 1 และ 4 จะเป็นการแสดงผลลัพธ์ของการตรวจสอบความต้องกันของแผนภาพในกรณีที่ต้องกัน ส่วนกรณีศึกษาที่ 2 และ 3 จะเป็นการแสดงผลลัพธ์ของการตรวจสอบความต้องกันของแผนภาพในกรณีที่ไม่ต้องกัน

5.1 การตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม กรณีศึกษาที่ 1

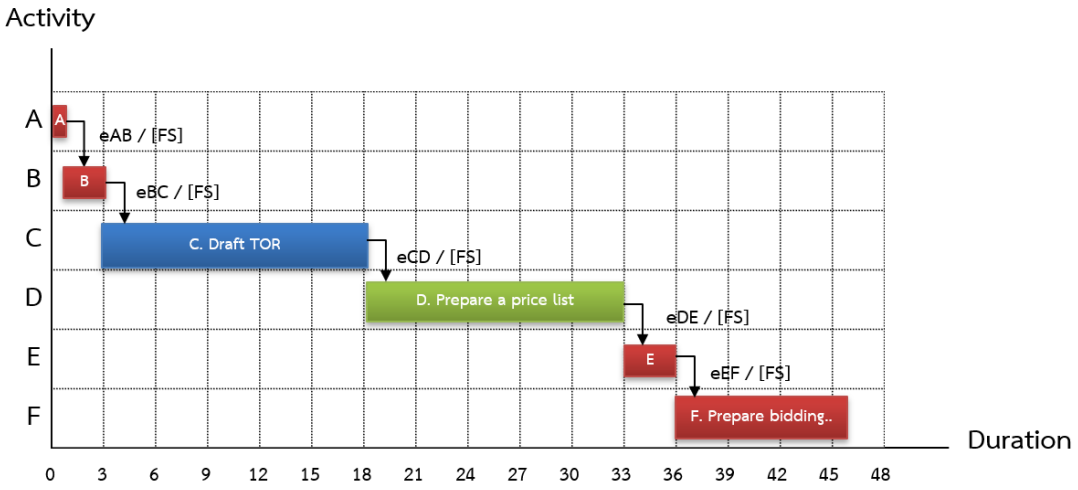
แผนภาพข่ายงานแบบพีดีเอ็มที่จะนำมาใช้เป็นกรณีศึกษาที่ 1 นั้น เป็นแผนภาพที่สามารถพบได้ทั่วไปในการบริหารงานโครงการ ซึ่งจะเป็นการจัดตารางกิจกรรมด้วยวิธีการพึ่งพาแบบสิ้นสุด-เริ่มต้น (Finish-to-Start) กล่าวคือ กิจกรรมถัดไปจะเริ่มดำเนินการได้ต่อเมื่อกิจกรรมก่อนหน้าดำเนินการเสร็จสิ้นแล้วเท่านั้น จากรูปที่ 5.1 จะเป็นการแสดงแผนภาพข่ายงานแบบพีดีเอ็มของขั้นตอนการจัดเตรียมเอกสารเพื่อการจัดหาพัสดุด้วยวิธีประกวดราคาอิเล็กทรอนิกส์ (e-bidding) ซึ่งประกอบไปด้วย 6 กิจกรรม เริ่มตั้งแต่การจัดเตรียมแผนการจัดซื้อจัดจ้างที่ใช้ระยะเวลาในการดำเนินงาน 1 วัน (กิจกรรม A) โดยเจ้าหน้าที่ทางพัสดุจะเป็นผู้จัดทำแผนนี้ จากนั้นเจ้าหน้าที่คนเดียวกันจะทำหน้าที่แต่งตั้งคณะกรรมการกำหนด TOR และ ราคากลาง ซึ่งใช้ระยะเวลาในการดำเนินการ 2 วัน (กิจกรรม B) จากนั้นผู้ที่ถูกแต่งตั้งให้เป็นคณะกรรมการกำหนด TOR จะดำเนินการร่าง TOR โดยใช้ระยะเวลา 15 วัน (กิจกรรม C) จากนั้นจึงส่งต่อร่าง TOR ให้คณะกรรมการราคากลางต่อไปเพื่อทำจัดทำราคากลาง ซึ่งใช้ระยะเวลาดำเนินการ 15 วัน (กิจกรรม D) จากนั้นจึงส่งต่อ

เอกสารทั้ง TOR และราคากลางกลับไปยังเจ้าหน้าที่พัสดุ เพื่อนำเอกสารดังกล่าวเสนอผู้มีอำนาจให้ความเห็นชอบ ซึ่งใช้ระยะเวลาในการดำเนินงาน 3 วัน (กิจกรรม E) และในขั้นตอนสุดท้ายของการจัดเตรียมเอกสารก็คือการจัดทำร่างเอกสารประกวดราคาฯ ร่างประกาศ และรายงานขอซื้อ ซึ่งจะดำเนินการโดยเจ้าหน้าที่พัสดุ โดยใช้ระยะเวลา 10 วัน รวมระยะเวลาในการดำเนินงานในขั้นตอนการจัดเตรียมเอกสารเพื่อการจัดหาพัสดุด้วยวิธีประกวดราคาอิเล็กทรอนิกส์ (e-bidding) ทั้งสิ้น 46 วัน



รูปที่ 5.1 แผนภาพข่ายงานพีดีเอ็ม กรณีศึกษาที่ 1

จากรูปที่ 5.1 เมื่อนำมาอธิบายให้อยู่ในรูปแบบของทฤษฎีช่วงเวลาของ Allen ตามกฎที่ออกแบบเพื่อหาความสัมพันธ์ที่ซ่อนเร้นจะแสดงได้ดังรูปที่ 5.2



รูปที่ 5.2 อธิบายความสัมพันธ์ของแผนภาพข่ายงานแบบพีดีเอ็ม
ในกรณีศึกษาที่ 1 ด้วยทฤษฎีช่วงเวลาของ Allen

จากรูป 5.2 แสดงให้เห็นว่าแต่ละกิจกรรมบนแผนภาพในกรณีศึกษาที่ 2 นั้นมีความต้องกันทุกประการทั้งโหนดกิจกรรม การพึ่งพาของกิจกรรม และการจัดสรรทรัพยากรบุคคลในแต่ละกิจกรรม

ในการตรวจสอบความต้องกันของโหนดกิจกรรมจะเห็นได้ว่ามีระยะเวลาการดำเนินการในการเริ่มต้นและสิ้นสุดที่สอดคล้องกัน เช่น กิจกรรม A มีการเริ่มต้นที่เร็วที่สุดเมื่อวันที่ 0 โดยมี

ระยะเวลาในการดำเนินงาน 1 วัน ดังนั้นก็จะไปเสร็จสิ้นที่วันที่ 1 เช่นเดียวกับกรณีที่เริ่มต้นช้าที่สุดคือกำหนดเป็นวันที่ 0 ใช้ระยะเวลาในการดำเนินงานเท่ากันคือ 1 วัน ดังนั้นก็จะไปเสร็จสิ้นวันที่ 1 เช่นกัน เนื่องจากเวลาลอยรวม (Total float) มีค่าเท่ากับ 0

สำหรับการตรวจสอบการพึ่งพาของกิจกรรมจะพบว่ากิจกรรมทั้งหมดมีการพึ่งพาในลักษณะของสิ้นสุด-เริ่มต้น (Finish-to-Start) ซึ่งเมื่ออธิบายด้วยทฤษฎีช่วงเวลาของ Allen จะพบว่าตรงกับกฎความสัมพันธ์แบบประชิด (intervalMeets) ซึ่งตรงกับกฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบทั่วไปที่ต้องกัน (FS_Consistency) ตามที่ได้ออกแบบ และจากแผนภาพจะพบว่าทุกกิจกรรมล้วนแล้วแต่มีความสัมพันธ์ที่ซ่อนเร้นแบบประชิดทั้งหมด ดังนั้นจึงกล่าวได้ว่าการพึ่งพาของกิจกรรมมีความต้องกัน

ในส่วนสุดท้ายคือการตรวจสอบความต้องกันของการจัดสรรทรัพยากร จะพบว่าไม่มีการดำเนินกิจกรรมที่ซ้ำซ้อนกันด้วยบุคลากรคนเดียวกัน หรือบุคคลากรไม่ได้ทำกิจกรรมหลายอย่างในเวลาเดียวกัน จึงกล่าวได้ว่าการจัดสรรทรัพยากรมีความต้องกัน

เมื่อนำแผนภาพข่ายงานแบบพีดีเอ็ม กรณีศึกษาที่ 1 ในรูปที่ 5.1 มาจัดเตรียมในรูปแบบของแฟ้มข้อมูล Excel เพื่อนำเข้าไปประมวลผลในโปรแกรม ดังรูปที่ 5.3

nodeID	nodeName	earlyStart	duration	earlyFinish	lateStart	totalFloat	lateFinish	hasResource	hasInputEdge	hasOutputEdge
A	A. Prepare a procurement	0	1	1	0	0	1	Procurement Officer		eAB
B	B. Appoint the TOR and Standardized Price committee	1	2	3	1	0	3	Procurement Officer	eAB	eBC
C	C. Draft TOR	3	15	18	3	0	18	TOR Committee	eBC	eCD
D	D. Prepare a price list	18	15	33	18	0	33	Price List Committee	eCD	eDE
E	E. Propose to the authorized person to approve	33	3	36	33	0	36	Procurement Officer	eDE	eEF
F	F. Prepare bidding document announcement and procurement report	36	10	46	36	0	46	Procurement Officer	eEF	

node | edge | resource

edgeID	edgeName	dependencyType
eAB	eAB	FS
eBC	eBC	FS
eCD	eCD	FS
eDE	eDE	FS
eEF	eEF	FS

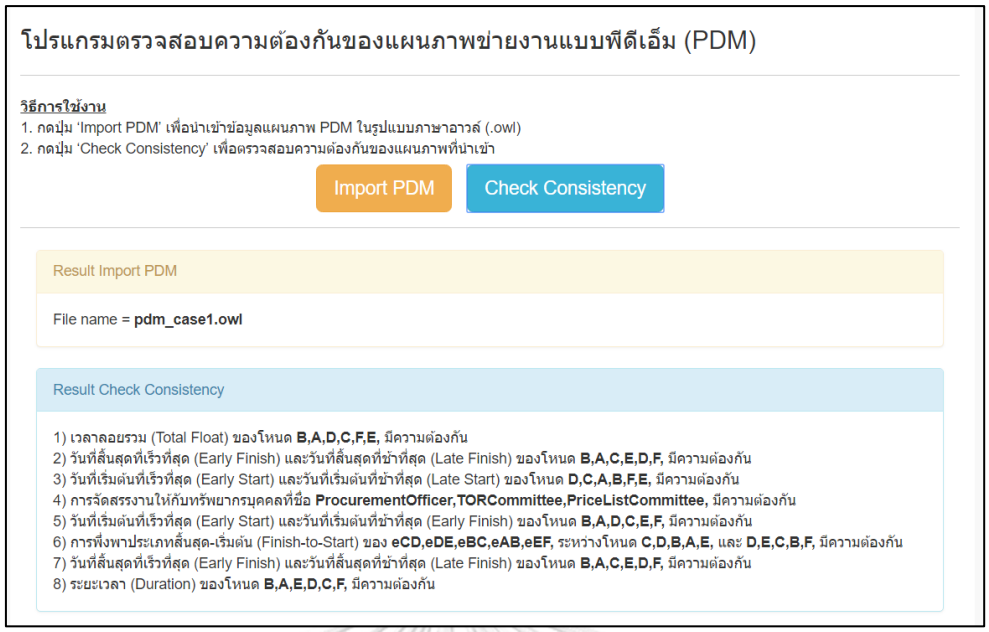
node | edge | resource

resourceID	resourceName
Procurement Officer	Procurement Officer
TOR Committee	TOR Committee
Price List Committee	Price List Committee

node | edge | resource

รูปที่ 5.3 การจัดเตรียมแฟ้มข้อมูล Excel เพื่อนำเข้าแผนภาพข่ายงานแบบพีดีเอ็ม กรณีศึกษาที่ 1

เมื่อนำเข้าแฟ้มข้อมูล Excel ตามรูปที่ 5.3 ด้วยโปรแกรมโปรเทเจ และทำการประมวลผลกฎที่ซ่อนเร้นเรียบร้อยแล้ว ก็สามารถส่งออกไฟล์ไฟล์อาวล์ เพื่อนำไปประมวลผลกฎเพื่อตรวจสอบความต้องกันด้วยโปรแกรมที่พัฒนาขึ้น และได้ผลลัพธ์ดังรูปที่ 5.4



รูปที่ 5.4 ผลลัพธ์จากการประมวลผลกฎ ด้วยโปรแกรมตรวจสอบความต้องกัน ของแผนภาพข่ายงานแบบพีดีเอ็ม

จากรูป 5.4 ผลลัพธ์จากการประมวลกฎ ด้วยโปรแกรมตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม พบว่า ระบบสามารถประมวลผลตามกฎที่ออกแบบไว้ อีกทั้งยังสามารถส่งคืนผลลัพธ์การตรวจสอบได้ถูกต้อง ซึ่งประกอบไปด้วยการตรวจสอบความต้องกันใน 3 ส่วน คือ

ส่วนที่ 1 คือ ตรวจสอบความต้องกันของโหนดกิจกรรม (Activity Node Consistency) ผลลัพธ์ที่ได้จากโปรแกรม เป็นดังนี้

1) เวลาลอยรวม (Total Float) ของโหนด B,A,D,C,F,E, มีความต้องกัน

2) วันที่สิ้นสุดที่เร็วที่สุด (Early Finish) และวันที่สิ้นสุดที่ช้าที่สุด (Late Finish) ของโหนด B,A,C,E,D,F, มีความต้องกัน

3) วันที่เริ่มต้นที่เร็วที่สุด (Early Start) และวันที่เริ่มต้นที่ช้าที่สุด (Late Start) ของโหนด D,C,A,B,F,E, มีความต้องกัน

5) วันที่เริ่มต้นที่เร็วที่สุด (Early Start) และวันที่เริ่มต้นที่ช้าที่สุด (Early Finish) ของโหนด B,A,D,C,E,F, มีความต้องกัน

7) วันที่สิ้นสุดที่เร็วที่สุด (Early Finish) และวันที่สิ้นสุดที่ช้าที่สุด (Late Finish) ของโหนด B,A,C,E,D,F, มีความต้องกัน

8) ระยะเวลา (Duration) ของโหนด B,A,E,D,C,F, มีความต้องกัน

ส่วนที่ 2 คือ ตรวจสอบความต้องกันของการพึ่งพาของกิจกรรม (Activity Node Consistency) ผลลัพธ์ที่ได้จากโปรแกรม เป็นดังนี้

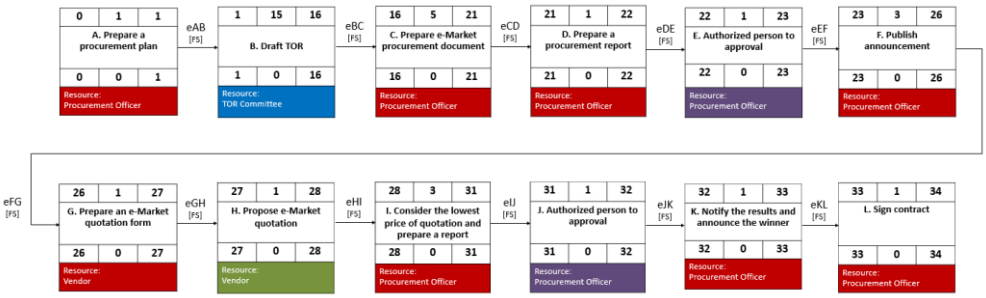
6) การพึ่งพาประเภทสิ้นสุด-เริ่มต้น (Finish-to-Start) ของ eCD,eDE,eBC,eAB,eEF, ระหว่างโหนด C,D,B,A,E, และ D,E,C,B,F, มีความต้องกัน

ส่วนที่ 3 คือ ตรวจสอบความต้องกันของการจัดสรรทรัพยากร (Resource Consistency)

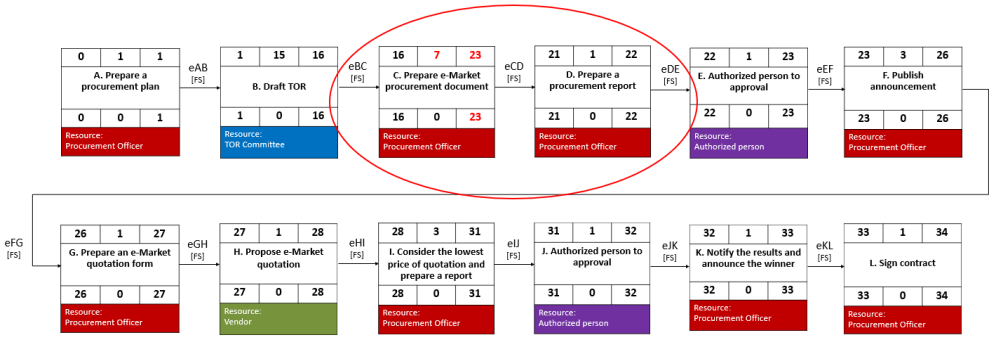
4) การจัดสรรงานให้กับทรัพยากรบุคคลที่ชื่อ ProcurementOfficer, TORCommittee, PriceListCommittee, มีความต้องกัน

5.2 การตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม กรณีศึกษาที่ 2

แผนภาพข่ายงานแบบพีดีเอ็มที่จะนำมาใช้เป็นกรณีศึกษาที่ 2 นั้น เป็นแผนภาพแสดงขั้นตอนการจัดซื้อจัดจ้างด้วยวิธีตลาดอิเล็กทรอนิกส์ (e-Market) ในกรณีที่วงเงินเกิน 5 แสนบาท แต่ไม่เกิน 5 ล้านบาท ซึ่งในแผนภาพกรณีศึกษาที่ 2 นี้จะนำเสนอในกรณีที่แผนภาพมีความไม่ต้องกัน ซึ่งประกอบไปด้วย 12 กิจกรรม เริ่มตั้งแต่ขั้นตอนการจัดเตรียมแผนการจัดซื้อจัดจ้าง โดยใช้ระยะเวลาในการดำเนินงาน 1 วัน (กิจกรรม A) โดยเจ้าหน้าที่ทางพัสดุจะเป็นผู้จัดทำแผนนี้ เมื่อกำหนดแผนเสร็จเรียบร้อยแล้ว ขั้นตอนต่อไปคณะกรรมการกำหนด TOR จะดำเนินการร่าง TOR โดยใช้ระยะเวลา 15 วัน (กิจกรรม B) จากนั้นเจ้าหน้าที่พัสดุจะดำเนินการจัดเตรียมเอกสารการจัดซื้อจัดจ้างกรณี e-Market ซึ่งโดยปกติจะใช้ระยะเวลา 5 วัน (กิจกรรม C) แต่เพื่อจำลองสถานการณ์กรณีที่เจ้าหน้าที่พัสดุทำงานล่าช้าจนต้องเพิ่มระยะเวลาในการทำงานจากเดิม 5 วัน เป็น 7 วัน ดังรูปที่ 5.6 ส่งผลให้กำหนดเดิมที่ควรจะเสร็จสิ้นภายในวันที่ 21 ทำให้ต้องเลื่อนระยะเวลาการสิ้นสุดของกิจกรรมเป็นวันที่ 23 ซึ่งจะส่งผลให้เกิดการทับซ้อนของงานระหว่างกิจกรรม C และกิจกรรมถัดไปอีก 2 กิจกรรม คือ การจัดเตรียมรายงานจัดซื้อจัดจ้าง (กิจกรรม D) และการส่งรายงานให้ผู้มีอำนาจเซ็นอนุมัติ (กิจกรรม E) ส่งผลให้การจัดสรรทรัพยากรบุคคลของกิจกรรม C และ D มีความไม่ต้องกันเนื่องจากบุคลากรคนเดียวกันทำกิจกรรม 2 อย่างในเวลาเดียวกัน แต่จะไม่ส่งผลกระทบกับกิจกรรม E เนื่องจาก เป็นเจ้าหน้าที่อีกท่านหนึ่งในการดำเนินงานจึงไม่มีประเด็นเรื่องความไม่ต้องกันของทรัพยากรบุคคล

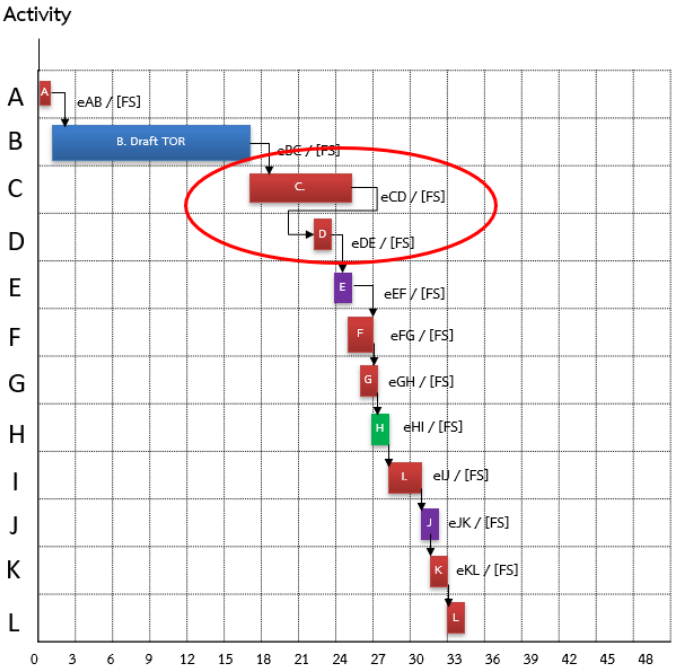


รูปที่ 5.5 แผนภาพข่ายงานพีดีเอ็ม กรณีศึกษาที่ 2 (กรณีที่ต้องกัน)



รูปที่ 5.6 แผนภาพข่ายงานพีดีเอ็ม กรณีศึกษาที่ 2 (กรณีที่ไม่ต้องกัน)

จากรูปที่ 5.5 เมื่อนำมาอธิบายให้อยู่ในรูปแบบของทฤษฎีช่วงเวลาของ Allen ตามกฎที่ออกแบบเพื่อหาความสัมพันธ์ที่ซ่อนเร้นจะแสดงได้ดังรูปที่ 5.6



รูปที่ 5.7 อธิบายความสัมพันธ์ของแผนภาพข่ายงานแบบพีดีเอ็ม
ในกรณีศึกษาที่ 2 (กรณีที่ไม่ต้องกัน) ด้วยทฤษฎีช่วงเวลาของ Allen

จากรูป 5.6 แสดงให้เห็นว่าแต่ละกิจกรรมบนแผนภาพในกรณีศึกษาที่ 2 นั้นมีความไม่ต้องกันของโหนดกิจกรรม และการจัดสรรทรัพยากรบุคคลในแต่ละกิจกรรม ซึ่งจะพบว่าจากแผนภาพที่แสดงด้วยช่วงเวลาของ Allen กิจกรรม C ซึ่งเป็นกิจกรรมก่อนหน้า (Predecessor) ของกิจกรรม D และมีการพึ่งพาแบบสิ้นสุด-เริ่มต้น (Finish-to-Start) แต่จากการเปลี่ยนแปลงระยะเวลาการดำเนินงาน

(Duration) จาก 5 วันเป็น 7 วัน ส่งผลให้กิจกรรม C มีการทำงานที่ล่าช้า แต่ในขณะเดียวกันกิจกรรม D ก็จำเป็นจะต้องเริ่มงานตามกำหนดเดิมส่งผลให้ความสัมพันธ์หรือการพึ่งพาของกิจกรรมเปลี่ยนไปจากสิ้นสุด-เริ่มต้น (Finish-to-Start) เป็นสิ้นสุด-เริ่มต้น แบบมีเวลานำ (Lead) ซึ่งตามนิยามของแผนภาพข่ายงานแบบพีดีเอ็มก็สามารถเกิดขึ้นได้ ดังนั้นในส่วนของการพึ่งพายังถือว่ามีความต้องกัน

ในการตรวจสอบความต้องกันของโหนดกิจกรรมจะเห็นได้ว่ามีระยะเวลาการดำเนินการในการเริ่มต้นและสิ้นสุดที่สอดคล้องกัน เช่น กิจกรรม A มีการเริ่มต้นที่เร็วที่สุดเมื่อวันที่ 0 โดยมีระยะเวลาในการดำเนินงาน 1 วัน ดังนั้นก็จะไปเสร็จสิ้นที่วันที่ 1 เช่นเดียวกับกรณีที่เริ่มต้นช้าที่สุดคือกำหนดเป็นวันที่ 0 ใช้ระยะเวลาในการดำเนินงานเท่ากันคือ 1 วัน ดังนั้นก็จะไปเสร็จสิ้นวันที่ 1 เช่นกัน เนื่องจากเวลาลอยรวม (Total float) มีค่าเท่ากับ 0

สำหรับการตรวจสอบการพึ่งพาของกิจกรรมจะพบว่ากิจกรรมทั้งหมดมีการพึ่งพาในลักษณะของสิ้นสุด-เริ่มต้น (Finish-to-Start) ซึ่งเมื่ออธิบายด้วยทฤษฎีช่วงเวลาของ Allen จะพบว่าตรงกับกฎความสัมพันธ์แบบประชิด (intervalMeets) ซึ่งตรงกับกฎตรวจสอบการพึ่งพาแบบสิ้นสุด-เริ่มต้นแบบทั่วไปที่ต้องกัน (FS_Consistency) ตามที่ได้ออกแบบ แต่ผลจากการปรับระยะเวลา (Duration) ในการดำเนินกิจกรรม C ส่งผลให้ ประเภทของการพึ่งพาเดิมที่เป็นสิ้นสุด-เริ่มต้น ที่มีความสัมพันธ์ซ่อนเร้นแบบประชิด กลายเป็น การพึ่งพาประเภทสิ้นสุด-เริ่มต้น แบบมีเวลารอคอย (Lag) โดยที่มีความสัมพันธ์แบบซ่อนเร้นแบบท่ามกลาง (intervalContains) ส่งผลให้เกิดความไม่ต้องกัน ของลักษณะความสัมพันธ์แบบ สิ้นสุด-เริ่มต้น

ในส่วนสุดท้ายคือการตรวจสอบความต้องกันของการจัดสรรทรัพยากร เป็นผลสืบเนื่องมาจากการปรับระยะเวลาในการดำเนินงานเช่นเดียวกัน จะพบว่าเมื่อขยายเวลาในการเดินกิจกรรม C จนทำให้เกิดการทับซ้อนกันระหว่างกิจกรรม C และกิจกรรม D ซึ่งใช้ทรัพยากรร่วมกัน ดังนั้นจึงส่งผลให้เกิดความไม่ต้องกันของการจัดสรรทรัพยากร

เมื่อนำแผนภาพข่ายงานแบบพีดีเอ็ม กรณีศึกษาที่ 2 ในรูปที่ 5.1 มาจัดเตรียมในรูปแบบของแฟ้มข้อมูล Excel เพื่อนำเข้าไปประมวลผลในโปรแกรม ดังรูปที่ 5.3

nodeName	earlyStart	duration	earlyFinish	lateStart	totalFloat	lateFinish	hasResource	hasInputEdge	hasOutputEdge
A. Prepare a procurement plan	0	1	1	0	0	1	Procurement Officer		eAB
B. Draft TOR	1	15	16	1	0	16	TOR Committee	eAB	eBC
C. Prepare e-Market procurement document	16	7	23	16	0	23	Procurement Officer	eBC	eCD
D. Prepare a procurement report	21	1	22	21	0	22	Procurement Officer	eCD	eDE
E. Authorized person to approval	22	1	23	22	0	23	Authorized person	eDE	eEF
F. Publish announcement	23	3	26	23	0	26	Procurement Officer	eEF	eFG
G. Prepare an e-Market quotation form	26	1	27	26	0	27	Procurement Officer	eFG	eGH
H. Propose e-Market quotation	27	1	28	27	0	28	Vendor	eGH	eHI
I. Consider the lowest price of quotation and pr	28	3	31	28	0	31	Procurement Officer	eHI	eIJ
J. Authorized person to approval	31	1	32	31	0	32	Authorized person	eIJ	eJK
K. Notify the results and announce the winner	32	1	33	32	0	33	Procurement Officer	eJK	eKL
L. Sign contract	33	1	34	33	0	34	Procurement Officer	eKL	

edgeID	edgeName	dependencyType
eAB	eAB	FS
eBC	eBC	FS
eCD	eCD	FS
eDE	eDE	FS
eEF	eEF	FS
eFG	eFG	FS
eGH	eGH	FS
eHI	eHI	FS
eIJ	eIJ	FS
eJK	eJK	FS
eKL	eKL	FS

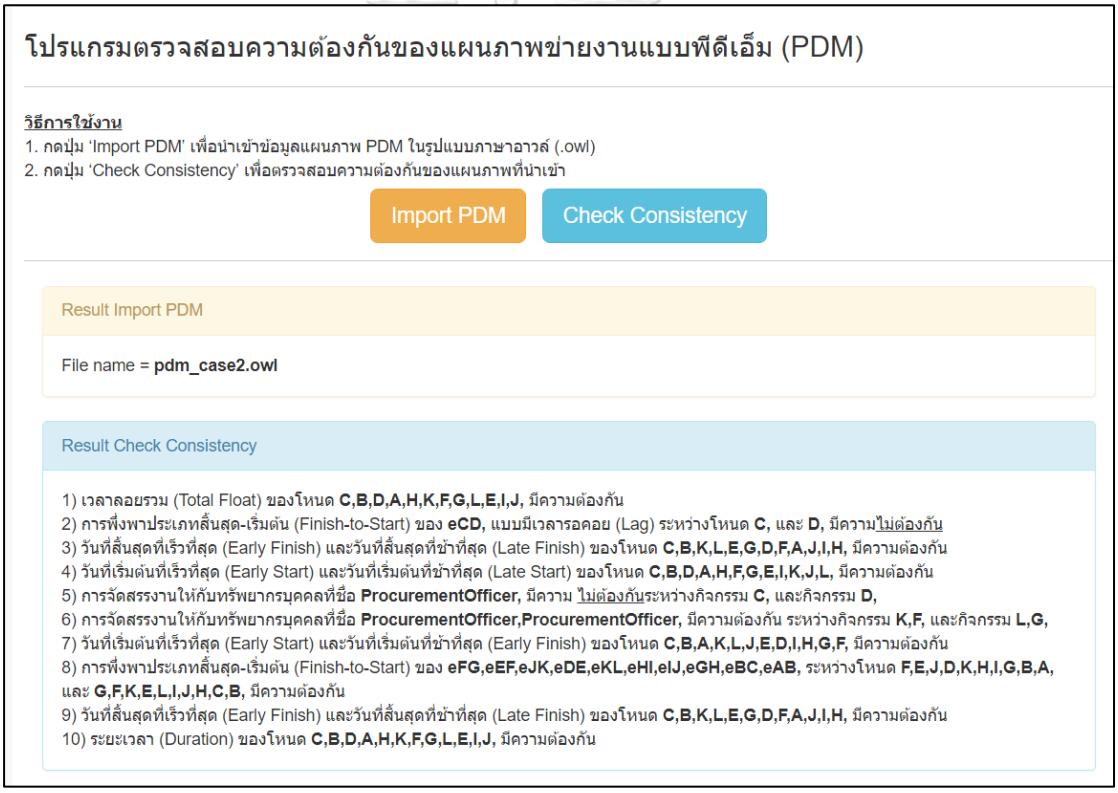
node | edge | resource

resourceID	resourceName
Procurement Officer	Procurement Officer
TOR Committee	TOR Committee
Authorized Person	Authorized Person
Vendor	Vendor

node | edge | resource

รูปที่ 5.8 การจัดเตรียมแฟ้มข้อมูล Excel เพื่อนำเข้าแผนภาพข่ายงานแบบพีดีเอ็ม กรณีศึกษาที่ 2

เมื่อนำเข้าแฟ้มข้อมูล Excel ตามรูปที่ 5.8 ด้วยโปรแกรมโปรเทเจ และทำการประมวลผลกฎที่ซ่อนเร้นเรียบร้อยแล้ว ก็สามารถส่งออกไฟล์ไฟล์อาวล์ เพื่อนำไปประมวลผลกฎเพื่อตรวจสอบความต้องกันด้วยโปรแกรมที่พัฒนาขึ้น และได้ผลลัพธ์ดังรูปที่ 5.9



รูปที่ 5.9 ผลลัพธ์จากการประมวลผลกฎ ด้วยโปรแกรมตรวจสอบความต้องกัน ของแผนภาพข่ายงานแบบพีดีเอ็ม (กรณีศึกษาที่ 2)

จากรูป 5.9 ผลลัพธ์จากการประมวลกฎ ด้วยโปรแกรมตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม พบว่า ระบบสามารถประมวลผลตามกฎที่ออกแบบไว้ อีกทั้งยังสามารถส่งคืนผลลัพธ์การตรวจสอบได้ถูกต้อง ซึ่งประกอบไปด้วยการตรวจสอบความต้องกันใน 3 ส่วน คือ

ส่วนที่ 1 คือ ตรวจสอบความต้องกันของโหนดกิจกรรม (Activity Node Consistency) ผลลัพธ์ที่ได้จากโปรแกรม เป็นดังนี้

1) เวลาลอยรวม (Total Float) ของโหนด C,B,D,A,H,K,F,G,L,E,I,J, มีความต้องกัน

3) วันที่สิ้นสุดที่เร็วที่สุด (Early Finish) และวันที่สิ้นสุดที่ช้าที่สุด (Late Finish) ของโหนด C,B,K,L,E,G,D,F,A,J,I,H, มีความต้องกัน

4) วันที่เริ่มต้นที่เร็วที่สุด (Early Start) และวันที่เริ่มต้นที่ช้าที่สุด (Late Start) ของโหนด C,B,D,A,H,F,G,E,I,K,J,L, มีความต้องกัน

7) วันที่เริ่มต้นที่เร็วที่สุด (Early Start) และวันที่เริ่มต้นที่ช้าที่สุด (Early Finish) ของโหนด C,B,A,K,L,J,E,D,I,H,G,F, มีความต้องกัน

9) วันที่สิ้นสุดที่เร็วที่สุด (Early Finish) และวันที่สิ้นสุดที่ช้าที่สุด (Late Finish) ของโหนด C,B,K,L,E,G,D,F,A,J,I,H, มีความต้องกัน

10) ระยะเวลา (Duration) ของโหนด C,B,D,A,H,K,F,G,L,E,I,J, มีความต้องกัน

ส่วนที่ 2 คือ ตรวจสอบความต้องกันของการพึ่งพาของกิจกรรม (Activity Node Consistency) ผลลัพธ์ที่ได้จากโปรแกรม เป็นดังนี้

8) การพึ่งพาประเภทสิ้นสุด-เริ่มต้น (Finish-to-Start) ของ eFG, eEF, eJK, eDE ,eKL, eHI, eIJ, eGH, eBC, eAB, ระหว่างโหนด F,E,J,D,K,H,I,G,B,A, และ G,F,K,E,L,I,J,H,C,B, มีความต้องกัน

2) การพึ่งพาประเภทสิ้นสุด-เริ่มต้น (Finish-to-Start) ของ eCD, แบบมีเวลารอคอย (Lag) ระหว่างโหนด C, และ D, **มีความไม่ต้องกัน**

ส่วนที่ 3 คือ ตรวจสอบความต้องกันของการจัดสรรทรัพยากร (Resource Consistency)

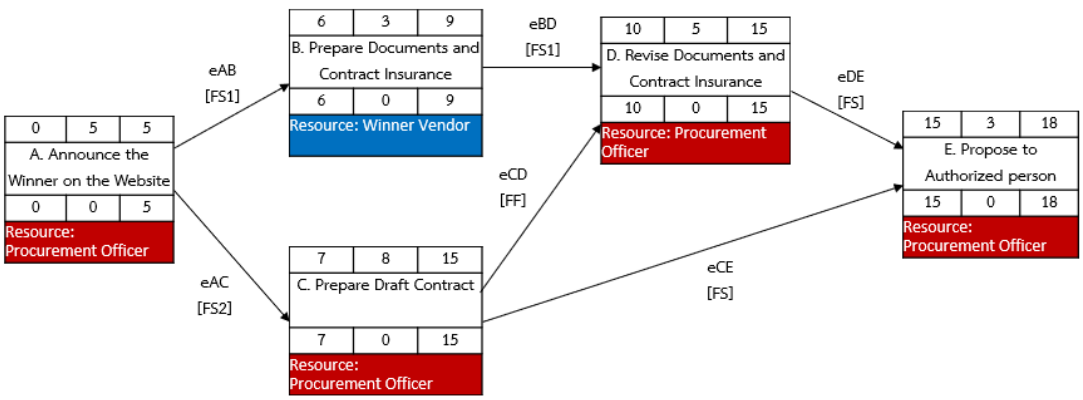
5) การจัดสรรงานให้กับทรัพยากรบุคคลที่ชื่อ ProcurementOfficer, **มีความไม่ต้องกัน** ระหว่างกิจกรรม C, และกิจกรรม D,

6) การจัดสรรงานให้กับทรัพยากรบุคคลที่ชื่อ ProcurementOfficer, มีความต้องกัน ระหว่างกิจกรรม K,F, และกิจกรรม L,G,

5.3 การตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม กรณีศึกษาที่ 3

แผนภาพข่ายงานแบบพีดีเอ็มที่จะนำมาใช้เป็นกรณีศึกษาที่ 3 นั้น เป็นแผนภาพที่ประกอบไปด้วยการพึ่งพาแบบสิ้นสุด-เริ่มต้น (Finish-to-Start) และการพึ่งพาแบบสิ้นสุด-สิ้นสุด (Fisnish-to-Finish) จากรูปที่ 5.10 จะเป็นการแสดงขั้นตอนการจัดทำสัญญาหลังจากได้ผู้ชนะในขั้นตอนการเลือกบริษัทที่เข้าร่วมการจัดซื้อจัดจ้างด้วยวิธีการต่าง ๆ โดยเริ่มต้นจากการประกาศเผยแพร่ผู้ชนะลงเว็บไซต์ของหน่วยงาน เป็นระยะเวลา 5 วัน (กิจกรรม A) ผู้ขายที่ชนะการแข่งขันจะต้องจัดเตรียม

เอกสารและหลักประกันสัญญามายื่นต่อเจ้าหน้าที่พัสดุ ภายในระยะเวลา 3 วัน หลังจากขึ้นประกาศทางหน้าเว็บไซต์ (กิจกรรม B) ซึ่งในระหว่างนี้อาจจะมีบริษัทอื่น ๆ ยื่นเรื่องอุทรณ์ ดังนั้นจึงต้องเผื่อเวลาให้เจ้าที่พัสดุก่อนที่จะทำการร่างสัญญา โดยการเผื่อระยะเวลาไว้ 2 วัน ก่อนที่จะเริ่มจัดเตรียมร่างสัญญา ซึ่งจะดำเนินการร่างเอกสารเป็นระยะเวลา 8 วัน (กิจกรรม C) ในขณะที่ผู้ขายมายื่นเอกสารเจ้าหน้าที่ก็จะต้องทำการตรวจเอกสารและหลักประกันสัญญาให้ครบถ้วน ถูกต้อง ตามเงื่อนไขสัญญา ซึ่งใช้ระยะเวลาในการตรวจสอบ 5 วัน (กิจกรรม D) เมื่อตรวจสอบเอกสารเรียบร้อยแล้วพร้อมกับร่างสัญญาเรียบร้อยแล้ว ซึ่ง 2 กระบวนการนี้ควรจะเสร็จสิ้นพร้อมกัน (กิจกรรม C และกิจกรรม D) เพื่อเตรียมส่งเรื่องไปยังผู้มีอำนาจในการอนุมัติและลงนามต่อไป ซึ่งจะใช้เวลาในการเดินเรื่องเอกสารเพื่อให้เซ็นลงนามอนุมัติเป็นลำดับขั้น โดยจะใช้เวลา 3 วัน (กิจกรรม E) ซึ่งรวมระยะเวลาทั้งหมดที่ใช้ในการจัดทำสัญญาหลังจากได้ผู้รับจ้างรวมทั้งสิ้น 18 วัน

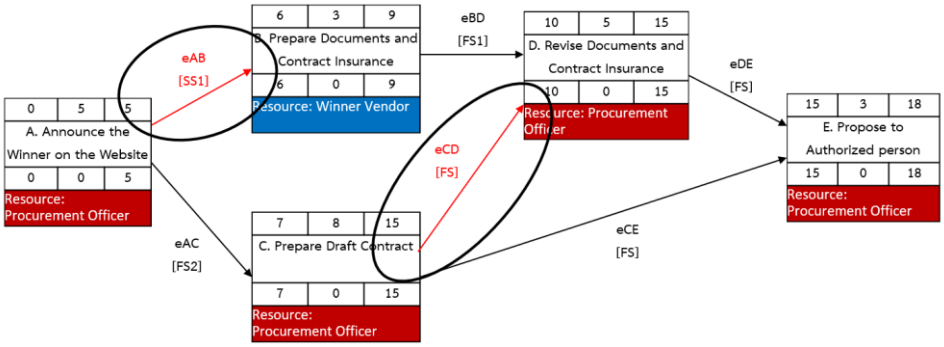


รูปที่ 5.10 แผนภาพข่ายงานแบบพีดีเอ็ม กรณีศึกษาที่ 3 (กรณีที่ต้องกัน)

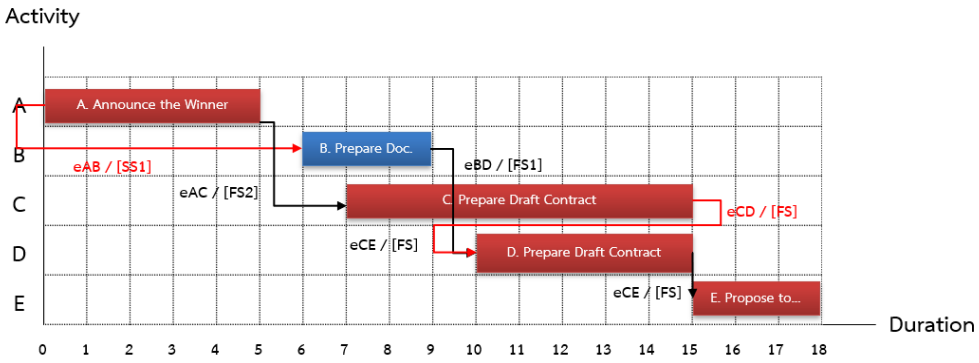
หากกิจกรรมดังกล่าวมีการดำเนินงานเป็นไปตามแผนที่กำหนด แผนภาพที่แสดงในรูป 5.10 ก็จะเป็นแผนภาพที่มีความต้องกัน แต่หากแผนงานดังกล่าวมีการเปลี่ยนแปลง การพึ่งพาของกิจกรรมจากเดิมที่ กิจกรม A และ B มีการพึ่งพาแบบสิ้นสุด-เริ่มต้น (Finish-to-Start) เปลี่ยนเป็น เริ่มต้น-เริ่มต้น (Start-to-Start) ก็จะส่งผลให้เกิดความไม่ต้องกัน เนื่องจากการพึ่งพาแบบเริ่มต้น-เริ่มต้น คือ การที่ 2 กิจกรรมจะต้องเริ่มต้นพร้อมกัน แต่จากรูปที่ 5.11 และ 5.12 แสดงให้เห็นว่ากิจกรรม A เริ่มต้นวันที่ 0 ส่วนกิจกรรม B เริ่มต้นวันที่ 6 ทำให้เกิดความไม่ต้องกันของการพึ่งพาแบบเริ่มต้น-เริ่มต้น

ในลักษณะเดียวกันหากมีการเปลี่ยนแปลงการพึ่งพาระหว่างกิจกรรม C และกิจกรรม D จาก สิ้นสุด-สิ้นสุด (Finish-to-Finish) เป็น สิ้นสุด-เริ่มต้น (Finish-to-Start) ก็จะส่งผลให้เกิดความไม่ต้องกัน เนื่องจากการพึ่งพาแบบสิ้นสุด-เริ่มต้น คือ การที่กิจกรรมแรกจะต้องสิ้นสุดก่อน กิจกรรมที่

ตามมาจึงจะเริ่มดำเนินการได้ แต่จากรูปที่ 5.11 และ 5.12 แสดงให้เห็นว่ากิจกรรม C สิ้นสุดวันที่ 15 แต่กิจกรรม D มีการเริ่มต้นของกิจกรรมไปล่วงหน้าก่อนที่กิจกรรม C จะแล้วเสร็จคือวันที่ 10 ทำให้เกิดความไม่ต้องกันของการพึ่งพาแบบสิ้นสุด-เริ่มต้น



รูปที่ 5.11 แผนภาพข่ายงานแบบพีดีเอ็ม กรณีศึกษาที่ 3 (กรณีที่ไม่ต้องกัน)



รูปที่ 5.12 อธิบายความสัมพันธ์ของแผนภาพข่ายงานแบบพีดีเอ็ม
ในกรณีศึกษาที่ 3 (กรณีที่ไม่ต้องกัน) ด้วยทฤษฎีช่วงเวลาของ Allen

ในส่วนสุดท้ายคือการตรวจสอบความต้องกันของการจัดสรรทรัพยากร จะพบว่ามีการดำเนินกิจกรรมที่ทับซ้อนกันด้วยบุคลากรคนเดียวกัน หรือบุคคลากรทำกิจกรรมหลายอย่างในเวลาเดียวกัน จากแผนภาที่ 5.12 จะเห็นได้ว่า Procurement Officer ซึ่งได้รับมอบหมายให้ทำกิจกรรม C และ D ในเวลาเดียวกัน จึงกล่าวได้ว่าการจัดสรรทรัพยากรมีความไม่ต้องกัน

เมื่อนำแผนภาพข่ายงานแบบพีดีเอ็ม กรณีศึกษาที่ 3 ในรูปที่ 5.11 มาจัดเตรียมในรูปแบบของแฟ้มข้อมูล Excel เพื่อนำเข้าไปประมวลผลในโปรแกรม ดังรูปที่ 5.13

earlyFinish	lateStart	totalFloat	lateFinish	hasResource	hasInputEdge	hasInputEdge	hasOutputEdge	hasOutputEdge
5	0	0	5	Procurement Officer			eAB	eAC
9	6	0	9	Winner Vendor	eAB		eBD	
15	7	0	15	Procurement Officer	eAC		eCD	eCE
15	10	0	15	Procurement Officer	eBD	eCD	eDE	
18	15	0	18	Procurement Officer	eCE	eDE		

node | edge | resource

edgeID	edgeName	dependencyType
eAB	eAB	SS
eAC	eAC	FS
eBD	eBD	FS
eCD	eCD	FS
eCE	eCE	FS
eDE	eDE	FS

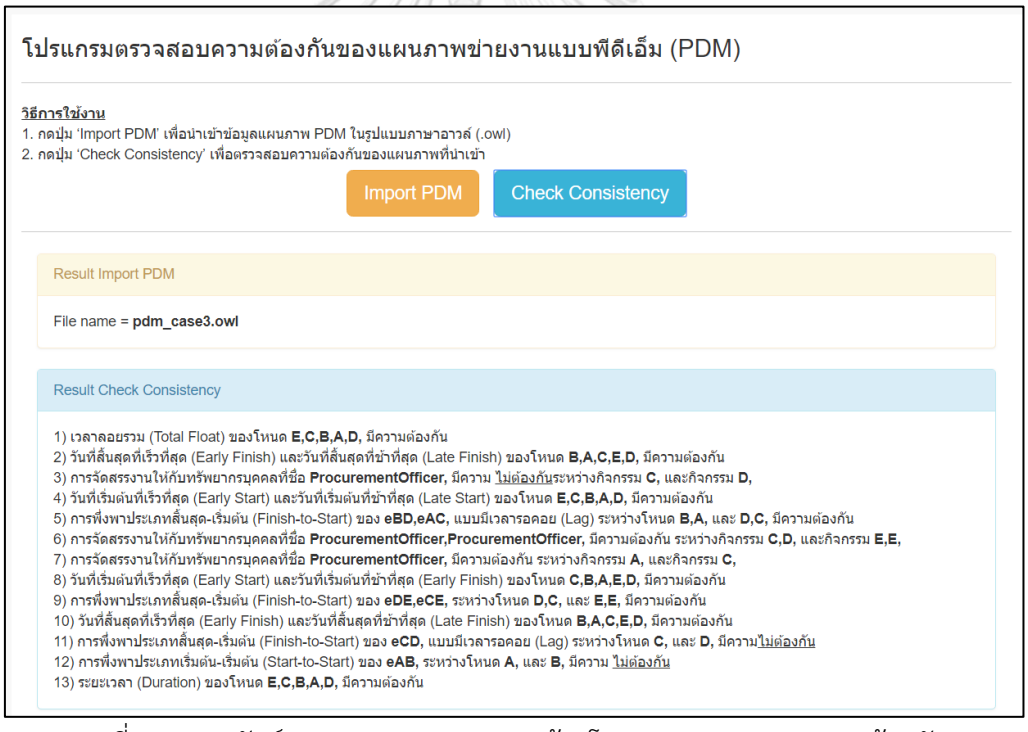
node | edge | resource

resourceID	resourceName
Procurement Officer	Procurement Officer
Winner Vendor	Winner Vendor

node | edge | resource

รูปที่ 5.13 การจัดเตรียมแฟ้มข้อมูล Excel เพื่อนำเข้าแผนภาพข่ายงานแบบพีดีเอ็ม กรณีศึกษาที่ 3

เมื่อนำเข้าแฟ้มข้อมูล Excel ตามรูปที่ 5.13 ด้วยโปรแกรมโปรเทเจ และทำการประมวลผลกฎที่ซ่อนเร้นเรียบร้อยแล้ว ก็สามารถส่งออกไฟล์ไฟล์อาวล์ เพื่อนำไปประมวลผลกฎเพื่อตรวจสอบความต้องกันด้วยโปรแกรมที่พัฒนาขึ้น และได้ผลลัพธ์ดังรูปที่ 5.14



รูปที่ 5.14 ผลลัพธ์จากการประมวลผลกฎ ด้วยโปรแกรมตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม กรณีศึกษาที่ 3

จากรูป 5.14 ผลลัพธ์จากการประมวลกฎ ด้วยโปรแกรมตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม พบว่า ระบบสามารถประมวลผลตามกฎที่ออกแบบไว้ อีกทั้งยังสามารถส่งคืนผลลัพธ์การตรวจสอบได้ถูกต้อง ซึ่งประกอบไปด้วยการตรวจสอบความต้องกันใน 3 ส่วน คือ

ส่วนที่ 1 คือ ตรวจสอบความต้องกันของโหนดกิจกรรม (Activity Node Consistency) ผลลัพธ์ที่ได้จากโปรแกรม เป็นดังนี้

1) เวลาลอยรวม (Total Float) ของโหนด E,C,B,A,D, มีความต้องกัน

2) วันที่สิ้นสุดที่เร็วที่สุด (Early Finish) และวันที่สิ้นสุดที่ช้าที่สุด (Late Finish) ของโหนด B,A,C,E,D, มีความต้องกัน

4) วันที่เริ่มต้นที่เร็วที่สุด (Early Start) และวันที่เริ่มต้นที่ช้าที่สุด (Late Start) ของโหนด E,C,B,A,D, มีความต้องกัน

8) วันที่เริ่มต้นที่เร็วที่สุด (Early Start) และวันที่เริ่มต้นที่ช้าที่สุด (Early Finish) ของโหนด C,B,A,E,D, มีความต้องกัน

10) วันที่สิ้นสุดที่เร็วที่สุด (Early Finish) และวันที่สิ้นสุดที่ช้าที่สุด (Late Finish) ของโหนด B,A,C,E,D, มีความต้องกัน

13) ระยะเวลา (Duration) ของโหนด E,C,B,A,D, มีความต้องกัน

ส่วนที่ 2 คือ ตรวจสอบความต้องกันของการพึ่งพาของกิจกรรม (Activity Node Consistency) ผลลัพธ์ที่ได้จากโปรแกรม เป็นดังนี้

5) การพึ่งพาประเภทสิ้นสุด-เริ่มต้น (Finish-to-Start) ของ eBD,eAC, แบบมีเวลารอคอย (Lag) ระหว่างโหนด B,A, และ D,C, มีความต้องกัน

9) การพึ่งพาประเภทสิ้นสุด-เริ่มต้น (Finish-to-Start) ของ eDE,eCE, ระหว่างโหนด D,C, และ E,E, มีความต้องกัน

11) การพึ่งพาประเภทสิ้นสุด-เริ่มต้น (Finish-to-Start) ของ eCD, แบบมีเวลารอคอย (Lag) ระหว่างโหนด C, และ D, มี**ความไม่ต้องกัน**

12) การพึ่งพาประเภทเริ่มต้น-เริ่มต้น (Start-to-Start) ของ eAB, ระหว่างโหนด A, และ B, มี**ความไม่ต้องกัน**

ส่วนที่ 3 คือ ตรวจสอบความต้องกันของการจัดสรรทรัพยากร (Resource Consistency)

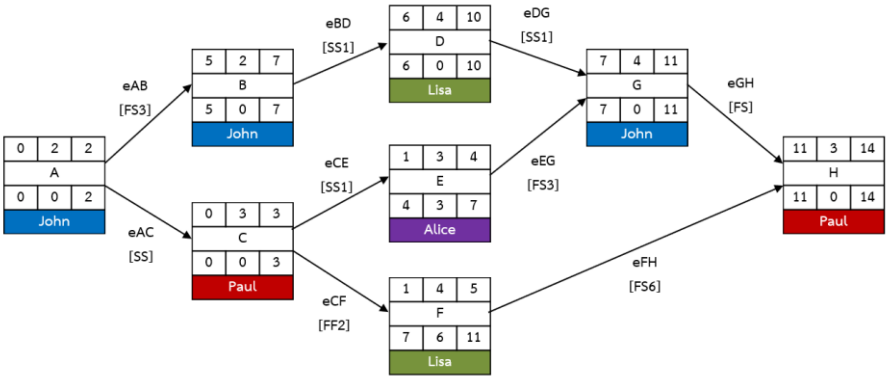
3) การจัดสรรงานให้กับทรัพยากรบุคคลที่ชื่อ ProcurementOfficer, มี**ความไม่ต้องกัน** ระหว่างกิจกรรม C, และกิจกรรม D,

6) การจัดสรรงานให้กับทรัพยากรบุคคลที่ชื่อ ProcurementOfficer, ProcurementOfficer, มีความต้องกัน ระหว่างกิจกรรม C,D, และกิจกรรม E,E,

7) การจัดสรรงานให้กับทรัพยากรบุคคลที่ชื่อ ProcurementOfficer, มีความต้องกัน ระหว่างกิจกรรม A, และกิจกรรม C,

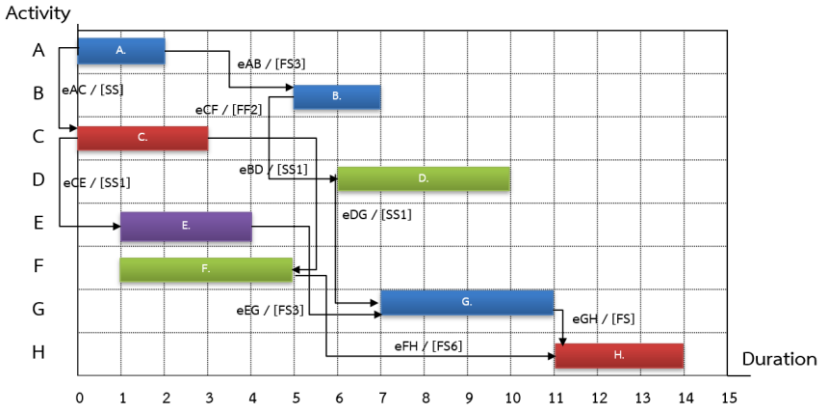
5.4 การตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม กรณีศึกษาที่ 4

แผนภาพสุดท้ายที่นำมาทดสอบ เป็นแผนภาพที่มีความซับซ้อนของโหนดกิจกรรม และการพึ่งพาในทุกรูปแบบ รวมถึงการที่มีระยะเวลารอคอย (Lag) ซึ่งปกติในการดำเนินงานโครงการจริงมักจะไม่ค่อยพบกรณีลักษณะนี้



รูปที่ 5.15 แผนภาพข่ายงานพีดีเอ็ม กรณีศึกษาที่ 4

จากรูปที่ 5.15 เมื่อนำมาอธิบายให้อยู่ในรูปแบบของทฤษฎีช่วงเวลาของ Allen ตามกฎที่ออกแบบเพื่อหาความสัมพันธ์ที่ซ่อนเร้นจะแสดงได้ดังรูปที่ 5.16



รูปที่ 5.16 อธิบายความสัมพันธ์ของแผนภาพข่ายงานแบบพีดีเอ็ม
ในกรณีศึกษาที่ 4 ด้วยทฤษฎีช่วงเวลาของ Allen

จากรูป 5.16 แสดงให้เห็นว่าแต่ละกิจกรรมบนแผนภาพในกรณีศึกษาที่ 4 นั้นมีความต้องกันทุกประการทั้งโหนดกิจกรรม การพึ่งพาของกิจกรรม และการจัดสรรทรัพยากรบุคคลในแต่ละกิจกรรม

ในการตรวจสอบความต้องกันของโหนดกิจกรรมจะเห็นได้ว่ามีระยะเวลาการดำเนินการในการเริ่มต้นและสิ้นสุดที่สอดคล้องกัน เช่น กิจกรรม A มีการเริ่มต้นที่เร็วที่สุดเมื่อวันที่ 0 โดยมี

ระยะเวลาในการดำเนินงาน 2 วัน ดังนั้นก็จะไปเสร็จสิ้นที่วันที่ 2 เช่นเดียวกับกรณีที่เริ่มต้นช้าที่สุดคือกำหนดเป็นวันที่ 0 ใช้ระยะเวลาในการดำเนินงานเท่ากันคือ 2 วัน ดังนั้นก็จะไปเสร็จสิ้นวันที่ 2 เช่นกัน เนื่องจากเวลาลอยรวม (Total float) มีค่าเท่ากับ 0 และเมื่อตรวจสอบกับทุก ๆ โหนดกิจกรรมในลักษณะเดียวกันก็พบว่ามีความต้องกัน

สำหรับการตรวจสอบการพึ่งพาของกิจกรรม ซึ่งประกอบไปด้วยการพึ่งพาของกิจกรรมที่หลากหลาย เช่น การพึ่งพาแบบเริ่มต้น-เริ่มต้น (Start-to-Start) ระหว่างกิจกรรม A และ กิจกรรม C จะพบว่ามีความต้องกัน เนื่องจากวันที่เริ่มต้นของทั้งสองกิจกรรมเป็นวันเดียวกัน เช่นเดียวกันกับการพึ่งพาสิ้นสุด-สิ้นสุด (Finish-to-Finish) แบบมีระยะเวลารอคอย (Lag) ระหว่างกิจกรรม C และกิจกรรม F กล่าวคือ กิจกรรม C สิ้นสุดวันที่ 3 จากนั้นรอคอยอีก 2 วัน กิจกรรม F จึงสิ้นสุดลงในวันที่ 5 ซึ่งก็แสดงให้เห็นถึงความต้องกัน หรือในกรณีของการพึ่งพาแบบสิ้นสุด-เริ่มต้น (Finish-to-Start) แบบมีระยะเวลารอคอย (Lag) ระหว่างกิจกรรม E และ กิจกรรม G กล่าวคือ กิจกรรม E สิ้นสุดวันที่ 4 จากนั้นรอคอยอีก 3 วัน กิจกรรม G จึงจะเริ่มดำเนินงาน

ในส่วนสุดท้ายคือการตรวจสอบความต้องกันของการจัดสรรทรัพยากร จะพบว่าไม่มีการดำเนินกิจกรรมที่ซ้ำซ้อนกันด้วยบุคลากรคนเดียวกัน หรือบุคคลากรไม่ได้ทำกิจกรรมหลายอย่างในเวลาเดียวกัน จึงกล่าวได้ว่าการจัดสรรทรัพยากรมีความต้องกัน

เมื่อนำแผนภาพข่ายงานแบบพีดีเอ็ม กรณีศึกษาที่ 4 ในรูปที่ 5.15 มาจัดเตรียมในรูปแบบของแฟ้มข้อมูล Excel เพื่อนำเข้าไปประมวลผลในโปรแกรม ดังรูปที่ 5.17

nodeID	nodeName	earlyStart	duration	earlyFinish	lateStart	totalFloat	lateFinish	hasResource	hasInputEdge	hasInputEdge	hasOutputEdge	hasOutputEdge
A	AAA	0	2	2	0	0	2	John			eAC	eAB
B	BBB	5	2	7	5	0	7	John	eAB		eBD	eBD
C	CCC	0	3	3	0	0	3	Paul	eAC		eCE	eCF
D	DDD	6	4	10	6	0	10	Lisa	eBD		eDG	
E	EEE	1	3	4	1	0	4	Alice	eCE		eEG	
F	FFF	1	4	5	1	0	5	John	eCF		eFH	
G	GGG	7	4	11	7	0	11	John	eDG	eEG	eGH	
H	HHH	11	3	14	11	0	14	Paul	eGH	eFH		

node | edge | resource

edgeID	edgeName	dependencyType
eAB	eAB	FS
eAC	eAC	SS
eBD	eBD	SS
eCE	eCE	SS
eCF	eCF	FF
eDG	eDG	SS
eEG	eEG	FS
eFH	eFH	FS
eGH	eGH	FS

node | edge | resource

resourceID	resourceName
Alice	Alice
John	John
Lisa	Lisa
Paul	Paul

node | edge | resource

รูปที่ 5.17 การจัดเตรียมแฟ้มข้อมูล Excel เพื่อนำเข้าแผนภาพข่ายงานแบบพีดีเอ็ม กรณีศึกษาที่ 4

เมื่อนำเข้าแฟ้มข้อมูล Excel ตามรูปที่ 5.17 ด้วยโปรแกรมโปรเทเจ และทำการประมวลผลกฎที่ซ่อนเร้นเรียบร้อยแล้ว ก็สามารถส่งออกไฟล์ไฟล์อาวล์ เพื่อนำไปประมวลผลกฎเพื่อตรวจสอบความต้องกันด้วยโปรแกรมที่พัฒนาขึ้น และได้ผลลัพธ์ดังรูปที่ 5.18

รูปที่ 5.18 ผลลัพธ์จากการประมวลผลกฎ ด้วยโปรแกรมตรวจสอบความต้องกัน

จากรูป 5.18 ผลลัพธ์จากการประมวลกฎ ด้วยโปรแกรมตรวจสอบความต้องกันของแผนภาพข่ายงานแบบพีดีเอ็ม พบว่า ระบบสามารถประมวลผลตามกฎที่ออกแบบไว้ อีกทั้งยังสามารถส่งคืนผลลัพธ์การตรวจสอบได้ถูกต้อง ซึ่งประกอบไปด้วยการตรวจสอบความต้องกันใน 3 ส่วน คือ

ส่วนที่ 1 คือ ตรวจสอบความต้องกันของโหนดกิจกรรม (Activity Node Consistency) ผลลัพธ์ที่ได้จากโปรแกรม เป็นดังนี้

1) เวลาลอยรวม (Total Float) ของโหนด B,H,C,G,E,A,F,D, มีความต้องกัน

2) วันที่สิ้นสุดที่เร็วที่สุด (Early Finish) และวันที่สิ้นสุดที่ช้าที่สุด (Late Finish) ของโหนด E,A,F,C,D,H,G,B, มีความต้องกัน

3) วันที่เริ่มต้นที่เร็วที่สุด (Early Start) และวันที่เริ่มต้นที่ช้าที่สุด (Late Start) ของโหนด B,H,C,G,D,A,F,E, มีความต้องกัน

7) วันที่เริ่มต้นที่เร็วที่สุด (Early Start) และวันที่เริ่มต้นที่ช้าที่สุด (Early Finish) ของโหนด C,G,E,F,A,D,H,B, มีความต้องกัน

10) วันที่สิ้นสุดที่เร็วที่สุด (Early Finish) และวันที่สิ้นสุดที่ช้าที่สุด (Late Finish) ของโหนด E,A,F,C,D,H,G,B, มีความต้องกัน

11) ระยะเวลา (Duration) ของโหนด B,C,H,G,E,A,F,D, มีความต้องกัน

ส่วนที่ 2 คือ ตรวจสอบความต้องกันของการพึ่งพาของกิจกรรม (Activity Node Consistency) ผลลัพธ์ที่ได้จากโปรแกรม เป็นดังนี้

4) การพึ่งพาประเภทสิ้นสุด-เริ่มต้น (Finish-to-Start) ของ eAB,eFH,eEG, แบบมีเวลารอคอย (Lag) ระหว่างโหนด A,F,E, และ B,H,G, มีความต้องกัน

6) การพึ่งพาประเภทเริ่มต้น-เริ่มต้น (Start-to-Start) แบบมีเวลารอคอย (Lag) ของ eDG,eCE,eBD, ระหว่างโหนด D,C,B, และ G,E,D, มีความต้องกัน

8) การพึ่งพาประเภทสิ้นสุด-สิ้นสุด (Finish-to-Finish) แบบมีเวลารอคอย (Lag) ของ eCF, ระหว่างโหนด C, และ F, มีความต้องกัน

9) การพึ่งพาประเภทสิ้นสุด-เริ่มต้น (Finish-to-Start) ของ eGH, ระหว่างโหนด G, และ H, มีความต้องกัน

12) การพึ่งพาประเภทเริ่มต้น-เริ่มต้น (Start-to-Start) ของ eAC, ระหว่างโหนด A, และ C, มีความต้องกัน

ส่วนที่ 3 คือ ตรวจสอบความต้องกันของการจัดสรรทรัพยากร (Resource Consistency)

5) การจัดสรรงานให้กับทรัพยากรบุคคลที่ชื่อ John, มีความต้องกัน ระหว่างกิจกรรม A, และกิจกรรม B,

บทที่ 6
สรุปผลการวิจัยและข้อเสนอแนะ

6.1 สรุปผลการวิจัย

งานวิจัยนี้ได้ทำการออกแบบองค์ความรู้เชิงความหมายของแผนภาพข่ายงานแบบพีดีเอ็ม ซึ่งเป็นแผนภาพที่ใช้ในการอธิบายกิจกรรม และระยะเวลาการดำเนินงานของแต่ละกิจกรรมภายใต้งานโครงการ โดยที่แต่ละกิจกรรมจะมีรูปแบบการพึ่งพา (Dependency) ที่แตกต่างกันไป ดังนั้นหากงานโครงการมีการปรับปรุงขั้นตอนหรือระยะเวลาในการดำเนินกิจกรรมอาจจะส่งผลให้รูปแบบของการพึ่งพาและเวลาที่ใช้ในกิจกรรมเกิดความคลาดเคลื่อนจนส่งผลให้เกิดความล่าช้า และไม่ถูกต้องในการดำเนินงานโครงการ ดังนั้นในงานวิจัยนี้จึงได้นำเทคนิคของออนโทโลยีมาช่วยในการอธิบายแผนภาพข่ายงานแบบพีดีเอ็มให้สามารถอธิบายเชิงความหมายได้มากขึ้น กล่าวคือคอมพิวเตอร์สามารถที่จะเข้าใจและประมวลผลได้จากข้อมูลนำเข้าของแผนภาพข่ายงานแบบพีดีเอ็ม

โดยเริ่มต้นจากการวิเคราะห์องค์ประกอบของแผนภาพและทำการออกแบบเมตาโมเดล จากนั้นจึงทำการแปลงเมตาโมเดลให้เป็นโครงร่างออนโทโลยี โดยใช้หลักการในการแปลงองค์ประกอบของยูเอ็มแอลเมตาโมเดล มาเป็นองค์ประกอบในภาษาอาวล์ จากนั้นจึงพัฒนาออนโทโลยีด้วยเครื่องมือโปรเทเจ เวอร์ชัน 5.2 โดยที่ข้อมูลโครงร่างออนโทโลยีของแผนภาพข่ายงานแบบพีดีเอ็ม ประกอบด้วย

- คลาส (Class) จำนวน 3 คลาส
- ความสัมพันธ์ระหว่างคลาส (Relation) จำนวน 19 ความสัมพันธ์

จากนั้นจึงทำการวิเคราะห์คุณสมบัติอื่น ๆ เพิ่มเติมบนแผนภาพ ที่ไม่สามารถอธิบายได้ด้วยโครงสร้างออนโทโลยีที่ออกแบบในขั้นแรก งานวิจัยนี้ได้นำทฤษฎีช่วงเวลาของ Alen มาช่วยในการอธิบายลักษณะของช่วงเวลาการเกิดขึ้นของกิจกรรม โดยทำการแปลงกฎทั้ง 13 ข้อของทฤษฎีดังกล่าว ให้เป็นภาษากฎเอสดับเบิลยูอาร์แอล (SWRL) เพื่อนำมาใช้อนุมานความรู้ที่ซ่อนเร้นของแผนภาพ โดยทำการพัฒนากฎที่ใช้อธิบายความสัมพันธ์ที่ซ่อนเร้นทั้งหมดจำนวน 8 กฎ

เมื่อได้องค์ความรู้ที่ครบถ้วนของแผนภาพข่ายงานแบบพีดีเอ็มแล้ว จึงทำการออกแบบกฎเพื่อตรวจสอบความต้องกันของแผนภาพ ด้วยภาษากฎเอสคิวดับเบิลยูอาร์แอล (SQWRL) ซึ่งมีการแบ่งกฎการตรวจสอบออกเป็น 3 ชุดย่อยตามส่วนประกอบของโครงสร้างออนโทโลยีที่ได้ทำการออกแบบ ประกอบด้วย

- กฎตรวจสอบความต้องกันของโหนดกิจกรรม จำนวน 6 กฎ
- กฎการตรวจสอบการพึ่งพาที่ต้องกัน จำนวน 13 กฎ
- กฎการตรวจสอบการพึ่งพาที่ไม่ต้องกัน จำนวน 12 กฎ
- กฎการตรวจสอบการจัดสรรทรัพยากร จำนวน 2 กฎ

รวมทั้งกฎที่ใช้ในการตรวจสอบความต้องกันทั้งสิ้น จำนวน 33 กฎ

เพื่อทำการทดสอบความถูกต้องของออนโทโลยีและกฎที่ได้ทำการออกแบบ จึงได้มีการนำกรณีศึกษา 4 กรณี ซึ่งเป็นขั้นตอนการดำเนินกิจกรรมการจัดซื้อจัดจ้างของหน่วยงานภาครัฐ มาใช้ในการทดสอบและประเมินประสิทธิภาพของออนโทโลยี ผลปรากฏว่าออนโทโลยีและกฎสามารถตรวจสอบความต้องกันของแผนภาพได้ถูกต้องตามที่ออกแบบไว้

6.2 ข้อจำกัด

- กฎที่ใช้ในการอนุมานความสัมพันธ์ที่ซ่อนเร้นในงานวิจัยนี้ ใช้หลักการของทฤษฎีช่วงเวลาของ Allen มาช่วยในการอธิบายลักษณะของช่วงเวลาการเกิดขึ้นของกิจกรรมเป็นหลัก ซึ่งอาจจะมีหลักการหรือทฤษฎีอื่น ๆ ที่สามารถอธิบายความสัมพันธ์ในที่เกี่ยวข้องได้

6.3 แนวทางการวิจัยในอนาคต

- ปรับปรุงหรือเพิ่มกฎการอนุมานเพื่ออธิบายความสัมพันธ์ที่ซ่อนเร้นด้วยหลักการอื่น ๆ ที่เกี่ยวข้องกับแผนภาพข่ายงานแบบพีดีเอ็ม หรืองานโครงการ

ปรับปรุงโครงสร้างองค์ความรู้ของแผนภาพข่ายงานแบบพีดีเอ็ม ให้มีความสามารถในการคำนวณการจัดตารางเวลาโซ่วิกฤต (Critical Chain Scheduling) ได้ ซึ่งเป็นเทคนิคที่ใช้ในการจัดสรรหรือมอบหมายหมายงานในกรณีที่มีทรัพยากรจำกัด

รายการอ้างอิง

1. Schwalbe, K., *Information technology project management*. 2015: Cengage Learning.
2. Callahan, M.T., D.G. Quackenbush, and J.E. Rowings, *Construction project scheduling*. 1992.
3. Wiest, J.D., *Precedence diagramming method: Some unusual characteristics and their implications for project managers.* Journal of Operations management, 1981. **1**(3): p. 121-130.
4. Institute, P.M. *Project Management Body of Knowledge (PMBOK)*. 1987. Project Management Institute.
5. Crandall, K.C. *Project planning with precedence lead/lag factors*. 1973. Project Management Institute.
6. Li, K. and R. Willis, *An iterative scheduling technique for resource-constrained project scheduling.* European Journal of Operational Research, 1992. **56**(3): p. 370-379.
7. Gruber, T.R., *A translation approach to portable ontology specifications.* Knowledge acquisition, 1993. **5**(2): p. 199-220.
8. Khan, A.H. and I. Porres, *Consistency of UML class, object and statechart diagrams using ontology reasoners.* Journal of Visual Languages & Computing, 2015. **26**: p. 42-65.
9. McGuinness, D.L., *Ontologies come of age.* Spinning the semantic web: bringing the World Wide Web to its full potential, 2002: p. 171-194.
10. Gómez-Pérez, A. and R. Benjamins. *Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods*. 1999. IJCAI and the Scandinavian AI Societies. CEUR Workshop Proceedings.
11. Noy, N.F. and C.D. Hafner, *The state of the art in ontology design: A survey and comparative review.* AI magazine, 1997. **18**(3): p. 53.
12. Hobbs, J.R. and F. Pan, *Time ontology in OWL.* W3C working draft, 2006. **27**: p. 133.

13. Allen, J.F., *Towards a general theory of action and time.* Artificial intelligence, 1984. **23**(2): p. 123-154.
14. Horrocks, I., et al., *SWRL: A semantic web rule language combining OWL and RuleML.* W3C Member submission, 2004. **21**: p. 79.
15. Eriksson, H., *Using jesstab to integrate protégé and jess.* IEEE Intelligent Systems, 2003. **18**(2): p. 43-50.
16. O'Connor, M.J. and A.K. Das. *A Pair of OWL 2 RL Reasoners*. in *OWLED*. 2012.
17. Van Emden, M.H. and R.A. Kowalski, *The semantics of predicate logic as a programming language.* Journal of the ACM (JACM), 1976. **23**(4): p. 733-742.
18. O'Connor, M. and A. Das. *SQWRL: a query language for OWL*. in *Proceedings of the 6th International Conference on OWL: Experiences and Directions-Volume 529*. 2009. CEUR-WS. org.
19. Date, C.J. and H. Darwen, *A guide to the SQL Standard: a user's guide to the standard relational language SQL*. 1989: Addison-Wesley.
20. Protegewiki.stanford.edu. *Protege Wiki*. 2017 [cited 2017 MARCH 12].
21. Kabilan, V. *Contract workflow model patterns using BPMN*. in *Proceedings of the 10th International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD 05), Caise*. 2005.
22. Arévalo Maldonado, C., I. Ramos Román, and M.J. Escalona Cuaresma. *Discovering Business Models for Software Process Management-An Approach for Integrating Time and Resource Perspectives from Legacy Information Systems*. in *ICEIS 2015: 17th International Conference on Enterprise Information Systems (2015), p 353-359*. 2015. ScitePress Digital Library.
23. Chung, S.-H., et al. *A Semantic Mapping Representation and Generation Tool Using UML for System Engineers*. in *Semantic Computing (ICSC), 2014 IEEE International Conference on*. 2014. IEEE.
24. Pham, T.A. and N. Le Thanh. *Ontology-based workflow validation*. in *Computing & Communication Technologies-Research, Innovation, and Vision for the Future (RIVF), 2015 IEEE RIVF International Conference on*. 2015. IEEE.
25. Booch, G., I. Jacobson, and J. Rumbaugh, *The unified modeling language reference manual.* 1999.

26. Dumas, M. and A.H. Ter Hofstede. *UML activity diagrams as a workflow specification language*. in *UML*. 2001. Springer.
27. Eshuis, R. and R. Wieringa, *A formal semantics for UML Activity Diagrams-Formalising workflow models.* 2001.
28. Guelfi, N. and A. Mammar. *A formal semantics of timed activity diagrams and its promela translation*. in *Software Engineering Conference, 2005. APSEC'05. 12th Asia-Pacific*. 2005. IEEE.
29. Miksa, T. and A. Rauber, *Using ontologies for verification and validation of workflow-based experiments.* Web Semantics: Science, Services and Agents on the World Wide Web, 2017. **43**: p. 25-45.
30. Ouardani, A., et al. *A Meta-modeling Approach for Sequence Diagrams to Petri Nets Transformation within the requirements validation process*. in *Proceedings of the European Simulation and Modeling Conference*. 2006.
31. Pătraşcu, A., *Comparative Analysis between OWL Modelling and UML Modelling.* Petroleum-Gas University of Ploiesti Bulletin, Technical Series, 2015. **67**(2).
32. Dienes, Z. and J. Perner, *A theory of implicit and explicit knowledge.* Behavioral and brain sciences, 1999. **22**(5): p. 735-808.
33. Kerzner, H. and H.R. Kerzner, *Project management: a systems approach to planning, scheduling, and controlling*. 2017: John Wiley & Sons.
34. Horridge, M., et al. *The Manchester OWL syntax*. in *OWLed*. 2006.
35. Arnold, K., J. Gosling, and D. Holmes, *The Java programming language*. 2005: Addison Wesley Professional.
36. O'Connor, M.J., et al. *The SWRLAPI: A Development Environment for Working with SWRL Rules*. in *OWLED*. 2008.
37. O'Connor, M.J. and A.K. Das. *SQWRL: A Query Language for OWL*. in *OWLED*. 2009.

ภาคผนวก

ประวัติผู้เขียนวิทยานิพนธ์

นางสาววิสาข์รัตน์ ศรีสูงเนิน เกิดเมื่อวันที่ 11 พฤษภาคม 2530 ที่กรุงเทพมหานคร สำเร็จการศึกษาระดับปริญญาตรีหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศและการสื่อสาร (หลักสูตรนานาชาติ) มหาวิทยาลัยมหิดล เมื่อปีการศึกษา 2553 และเข้าศึกษาในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัยในปีการศึกษา 2558