วิธีการตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์ด้วยแกรมมาของกราฟ

นายทรงพล ทองคำ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2556

AN APPROACH OF SOFTWARE ARCHITECTURAL STYLE DETECTION USING GRAPH GRAMMAR

Mr. Songpon Thongkum

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Software Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2013

หัวข้อวิทยานิพนธ์	วิธีการตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์ด้วยแกรมมาของกราฟ
โดย	นายทรงพล ทองคำ
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.บัณฑิต เอื้ออาภรณ์)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์)

.....กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร.ทรงศักดิ์ รองวิริยะพานิช)

ทรงพล ทองคำ : วิธีการตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์ด้วยแกรมมาของกราฟ. (AN APPROACH OF SOFTWARE ARCHITECTURAL STYLE DETECTION USING GRAPH GRAMMAR) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : รศ.ดร.วิวัฒน์ วัฒนาวุฒิ, 160 หน้า.

วิทยานิพนธ์นี้นำเสนอทางเลือกวิธีการตรวจจับรูปแบบสถาปัตยกรรมโดยใช้ขั้นตอนการลดรูปของกราฟแกรมมา คำนิยามของไวยากรณ์พึ่งบริบทของกราฟแกรมมาได้ถูกกำหนดไว้และขั้นตอนการคืนรูปและการลดรูปได้ถูกปรับขยายและนำเสนอให้ใช้งานได้กับคอมโพเนนท์ อินเตอร์เฟส และลิ้งก์ที่มีอยู่ในแบบจำลองสถาปัตยกรรมซอฟต์แวร์ที่เขียนด้วยภาษาเอกซ์เอดีแอล โดยภาษาเอกซ์เอดีแอลเป็นหนึ่งในภาษาอธิบายสถาปัตยกรรมที่ได้รับความนิยม กรณีศึกษาของวิธีการตรวจจับรูปแบบสถาปัตยกรรม 4 รูปแบบหลัก คือ รูปแบบรวมศูนย์กลาง รูปแบบไปป์และฟิลเตอร์ รูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้น และ รูปแบบลำดับชั้นได้ถูกแสดงไว้

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก .....

ปีการศึกษา 2556 .....

5272418123 : MAJOR COMPUTER ENGINEERING

KEYWORDS : SOFTWARE ARCHITECTURE / XADL / SOFTWARE ARCHITECTURALSTYLE DETECTION / GRAPH GRAMMAR

SONGPON THONGKUM : AN APPROACH OF SOFTWARE ARCHITECTURAL STYLE DETECTION USING GRAPH GRAMMAR. ADVISOR : ASSOC. PROF. WIWAT VATANAWOOD, PH.D., 160 pp.

In this thesis, we propose an alternative scheme of the architectural styles detection using the reduction steps of a graph grammar. The definition of the context sensitive graph grammar and its derivation and reduction steps are extended and proposed to represent the typical components, interfaces, and links in the software architectural model written in xADL. The xADL is one of the popular architectural description languages. The architectural style case studies of 4 selected software architectural styles - repository , pipe and filter , event base , and layer - are demonstrated.

Department : Computer Engineering Student's Signature .....

Field of Study : Software Engineering Advisor's Signature .....

Academic Year : 2013

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลือจากรองศาสตราจารย์ ดร. วิวัฒน์ วัฒนาวุฒิ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ขอกราบขอบพระคุณอาจารย์ที่ให้คำแนะนำ และข้อเสนอแนะต่างๆ ตลอดระยะเวลาของการทำวิทยานิพนธ์ของข้าพเจ้า จนสำเร็จลุล่วงได้ด้วยดี

ขอขอบพระคุณ รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์ ประธานกรรมการสอบ ผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์ และผู้ช่วยศาสตราจารย์ ดร.ทรงศักดิ์ รองวิริยะพานิช กรรมการสอบวิทยานิพนธ์ ที่ได้กรุณาเสียสละเวลา เพื่อให้คำแนะนำ แนวทางในการทำงานวิจัยให้วิทยานิพนธ์ฉบับนี้สำเร็จเสร็จสิ้นเป็นไปอย่างมีคุณภาพ

ท้ายที่สุดนี้ ขอกราบขอบพระคุณบิดา มารดา พี่ และขอบคุณเพื่อนๆ ทุกคนที่เป็นกำลังใจและให้ความสนับสนุนมาโดยตลอด

สารบัญ หน้า

หน้า

สารบัญตาราง หน้า

สารบัญรูป

หน้า

หน้า

หน้า

หน้า

หน้า

หน้า

หน้า

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันบริษัทที่เป็นองค์กรขนาดใหญ่มักจะมีโครงสร้างของระบบสารสนเทศที่มีขนาดใหญ่เป็นไปตามการเจริญเติบโตของบริษัทซึ่งจะทำให้เกิดปัญหาความซับซ้อนในโครงสร้างของระบบโดยสามารถวัดค่าความซับซ้อนได้หลายวิธีด้วยกัน เช่น วิธีของ McCabe's cyclomatic complexity เป็นต้น และเนื่องจากบริษัทต้องการข้อมูลที่มากขึ้นทำให้ต้องมีการขยายโครงสร้างของระบบออกไปจนทำให้เกิดปัญหาความซับซ้อนในโครงสร้างของสถาปัตยกรรมซึ่งทำให้การตรวจสอบและแก้ไขเป็นไปได้ยาก

ปัญหาดังกล่าวสามารถใช้แนวคิดเกี่ยวกับการออกแบบสถาปัตยกรรมในการแก้ไขปัญหา เนื่องจากการออกแบบสถาปัตยกรรมเป็นการกำหนดส่วนประกอบของซอฟต์แวร์แสดงความสัมพันธ์ระหว่างระบบย่อยของระบบซึ่งจะให้สามารถกำหนดคุณลักษณะต่างๆของระบบผ่านทางการออกแบบสถาปัตยกรรม [1] โดยการออกแบบทำได้โดยการใช้แผนภาพ เพื่อนำเสนอแบบจำลองการออกแบบ (Design Model) ของระบบได้ แต่เนื่องจากการแบบโดยใช้แผนภาพยังมีข้อด้อยในหลายส่วน ดังนั้นจึงมีการคิดค้นภาษาที่ใช้ในการออกแบบสถาปัตยกรรม ซึ่งมีไว้สำหรับลดจุดด้อยในการออกแบบสถาปัตยกรรมโดยใช้แผนภาพขึ้น [2]

โดยในปัจจุบันนั้น ภาษาที่ใช้ในการออกแบบสถาปัตยกรรมมีหลายรูปแบบ [3] ซึ่งแต่ละแบบจะมีข้อดีและข้อเสียที่แตกต่างกันไป โดยหนึ่งในนั้นมีชื่อว่า ภาษาที่ใช้สำหรับระบุลักษณะรูปแบบของสถาปัตยกรรมซอฟต์แวร์ (Architecture Description Language) [3,5] ที่สามารถนำมาใช้อธิบายถึงการทำงานของระบบ และในปัจจุบันได้มีการพัฒนาภาษาที่ใช้สำหรับระบุลักษณะรูปแบบของสถาปัตยกรรมซอฟต์แวร์ขึ้นหลายภาษา

เพื่อให้เหมาะสมกับลักษณะของงานที่แตกต่างกันไป เช่นภาษา Rapide, Darwin, Mae, Koala และ C2SADEL เป็นต้น [4] ดังแสดงในตารางที่ 1.1 จากการที่ได้มีการพัฒนาภาษาที่ใช้สำหรับระบุลักษณะรูปแบบของสถาปัตยกรรมซอฟต์แวร์ ขึ้นหลายภาษาดังกล่าวสามารถช่วยให้ทำงานได้ครอบคลุมมากขึ้น และภาษาที่ใช้สำหรับระบุลักษณะรูปแบบของสถาปัตยกรรมซอฟต์แวร์ในแต่ละรูปแบบยังมีข้อดี คือ สามารถใช้งานได้แค่งานลักษณะใดลักษณะหนึ่งได้อย่างมีประสิทธิภาพสูงสุด เช่น ภาษา Mae สามารถนำมาใช้อธิบายลักษณะสถาปัตยกรรม การจัดการคอนฟิคกูเรชั่นของสถาปัตยกรรม (Configuration management) ต่อมาได้มีการพัฒนาภาษาที่ใช้สำหรับระบุลักษณะรูปแบบของสถาปัตยกรรมซอฟต์แวร์ ชนิดใหม่ที่มีความสามารถในการใช้งานหลายๆลักษณะงานพร้อมกันซึ่งเรียกว่า ภาษาเอกซ์เอดีแอล (xADL) ซึ่งในปัจจุบันภาษาเอกซ์เอดีแอลไปใช้ยังไม่

แพร่หลายเท่าที่ควร และการนำเอาคุณลักษณะของรูปแบบสถาปัตยกรรมซอฟต์แวร์ไปปรับใช้กับองค์กรขนาดใหญ่นั้น จะทำให้สามารถมองเห็นคุณลักษณะความต้องการของระบบงานได้อย่างมีประสิทธิภาพสูงสุด เช่น ในการใช้ลักษณะสถาปัตยกรรมแบบเครื่องแม่ข่ายกับเครื่องลูกข่าย ระบบจะมีความปลอดภัยสูง และระบบมีความน่าเชื่อถือ หรือในการใช้ลักษณะสถาปัตยกรรมในรูปแบบรวมศูนย์กลาง ระบบจะมีความมั่นคงสูงและข้อมูลมีความน่าเชื่อถือ เป็นต้น

ตารางที่ 1.1 ลักษณะการใช้งานของภาษาที่ใช้สำหรับระบุลักษณะรูปแบบของสถาปัตยกรรมซอฟต์แวร์ [4]

ADL	Supporting Domain
Darwin, C2SADL	Dynamic systems
Mae	Configuration management
Koala	Product-line Families
Rapide	Event oriented system
Wright	Behavioral properties
ACME	Interchange Language

ในสถาปัตยกรรมที่มีขนาดใหญ่และมีความซับซ้อนสูงนั้น การจะตรวจสอบสถาปัตยกรรมในระบบมักจะกระทำด้วยมือซึ่งเป็นงานที่ค่อนข้างยุ่งยากแต่ก็ยังมีงานวิจัยบางส่วนที่หาวิธีการตรวจสอบสถาปัตกรรมอย่างอัตโนมัติ ซึ่งทฤษฎีกราฟแกรมมาเป็นทฤษฎีหนึ่งที่ได้รับความนิยมสำหรับการสร้างกราฟ โดยจะมีการกำหนดกฎเกณฑ์ขึ้นเพื่อเป็นพื้นฐานสำหรับการสร้างกราฟ

เพื่อเป็นการสนับสนุนการออกแบบสถาปัตยกรรมโดยใช้รูปแบบสถาปัตยกรรมให้สะดวกขึ้น ทั้งยังช่วยตรวจสอบการนำรูปแบบต่างๆ ไปใช้งานจริงและยังเป็นทางเลือกสำหรับวิธีการตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์ วิทยานิพนธ์ฉบับนี้ได้นำเสนอวิธีการในการตรวจจับว่ามีรูปแบบของสถาปัตยกรรมใดในสถาปัตยกรรมที่เขียนด้วยภาษาเอกซ์เอดีแอลโดยใช้ทฤษฎีกราฟแกรมมามาประยุกต์ใช้

1.2 วัตถุประสงค์ของการวิจัย

1) เพื่อพัฒนากราฟแกรมมาให้สามารถแสดงรายละเอียดของสถาปัตยกรรมที่มีอินเตอร์เฟสประกอบอยู่ได้
2) เพื่อพัฒนาเครื่องมือสำหรับตรวจจับรูปแบบสถาปัตยกรรมในภาษาเอกซ์เอดีแอล

1.3 ขอบเขตงานวิจัย

1) ข้อมูลของภาษาเอกซ์เอดีแอล ที่เป็นข้อมูลนำเข้าต้องเป็นข้อมูลที่สร้างโดยใช้ ArchStudio4 เท่านั้น โดยจะใช้หลักไวยากรณ์เวอร์ชัน 2.0
2) การตรวจสอบรูปแบบของสถาปัตยกรรม (Styles) จะตรวจสอบเฉพาะรูปแบบดังต่อไปนี้
    - สถาปัตยกรรมไปป์และฟิลเตอร์ (Pipe and Filter)
    - สถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้น (EventBase)
    - สถาปัตยกรรมในรูปแบบรวมศูนย์กลาง (Repository)
    - สถาปัตยกรรมในรูปแบบลำดับชั้น (Layer)
3) ในสถาปัตยกรรมหนึ่งใดจะประกอบด้วย คอมโพเนนท์ สำหรับแทนส่วนโปรแกรม ลิ้งก์ สำหรับแทนเส้นเชื่อมโยง และอินเตอร์เฟสแทนส่วนต่อประสาน
4) คอมโพเนนท์ ที่ปรากฏในสถาปัตยกรรม สามารถเป็น Composition ได้ กล่าวคือ คอมโพเนนท์ นั้นสามารถมีคอมโพเนนท์ ภายในอีกระดับหนึ่ง
5) นำแนวคิดเกี่ยวกับกราฟแกรมมา มาประยุกต์ใช้ในการตรวจสอบสถาปัตยกรรม
6) สถาปัตยกรรมของระบบนำเข้าประกอบด้วย คอมโพเนนท์ ที่เชื่อมต่อกันด้วยลิ้งก์ (Link) โดยไม่ผ่านคอนเนคเตอร์ (Connector)
7) สถาปัตยกรรมหนึ่ง สามารถมีรูปแบบสถาปัตยกรรมที่ผสมผสานรวมมากกว่า 1 รูปแบบได้ กล่าวคือ มีสถาปัตยกรรมมากกว่า 1 แบบได้

1.4 วิธีดำเนินงานวิจัย

1) ศึกษาและนิยามทฤษฎีที่เกี่ยวข้อง
2) วิเคราะห์และออกแบบข้อมูลนำเข้า
3) การออกแบบวิธีการตรวจจับสถาปัตยกรรม
4) ออกแบบข้อกำหนดที่เกี่ยวข้องกับวิธีตรวจจับสถาปัตยกรรมที่ได้ออกแบบไว้
5) พัฒนาเครื่องมือและส่วนต่อประสานสำหรับผู้ใช้เพื่อทดสอบการออกแบบ
6) ทดสอบและประเมินผลวิธีการตรวจจับสถาปัตยกรรม
7) สรุปและเรียบเรียงรูปเล่มวิทยานิพนธ์

1.5 บทความวิจัยที่ได้รับการตีพิมพ์

ในงานวิจัยนี้ มีบทความวิจัย คือ “วิธีการตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์ด้วยกราฟแกรมมา (An Approach of Software Architectural Styles Detection Using Graph Grammar)“ จัดทำโดยนายทรงพล ทองคำ และรองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ ซึ่งได้รับการคัดเลือกนำเสนอและตีพิมพ์ในงาน “งานประชุมวิชาการนานาชาติของวิศวกรรมและวิทยาการคอมพิวเตอร์ (The 3rd World Conference on Science and Engineering 2013 : WCSE 2013)” ระหว่างวันที่ 24-25 สิงหาคม 2556 ณ โรงแรม ควอลิตี้โฮเทล ประเทศสิงคโปร์

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

ในงานวิจัยมีทฤษฎีที่เกี่ยวข้อง ได้แก่ สถาปัตยกรรมซอฟต์แวร์ ภาษาเอกซ์เอดีแอล ทฤษฎีกราฟแกรมมา โดยมีรายละเอียดดังต่อไปนี้

2.1.1 รูปแบบสถาปัตยกรรม (Software Architectural Style)

สถาปัตยกรรมซอฟต์แวร์นั้น ได้ถูกนิยามในหลากหลายความหมายโดย Perry และ Wolf [1] ได้นำเสนอไว้ว่า สถาปัตยกรรมซอฟต์แวร์ เป็นกลุ่มของการออกแบบซึ่งมีรูปแบบแน่นอน ซึ่งประกอบด้วยสามส่วน คือ กระบวนการ (process) ข้อมูล (data) และจุดเชื่อมต่อ (connection)

Shaw และ Garlan [1,7] ได้นำเสนอความหมายของสถาปัตยกรรมซอฟต์แวร์ไว้ว่า ถ้ามองโปรแกรมที่พัฒนาให้อยู่ในระดับโมดูล จะเห็นได้ว่าโมดูลจะมีความสัมพันธ์กันจนอยู่ในรูปแบบของสถาปัตยกรรมหนึ่งใดๆ ซึ่งโปรแกรมเมอร์มีหน้าที่ในการจัดการการติดต่อสื่อสารกันระหว่างโมดูลต่างๆนั้น การติดต่อกันระหว่างโมดูลนั้น จะมีรูปแบบหลายๆ รูปแบบ ตั้งแต่อดีตจนถึงปัจจุบัน ซึ่งแต่ละรูปแบบของสถาปัตยกรรม จะนำไปแก้ปัญหาที่แตกต่างกันไป ซึ่ง Shaw และ Garlan ได้นำเสนอกลุ่มของสถาปัตยกรรม ไว้เป็นกลุ่มๆ ดังตารางที่ 2.1

งานวิจัยนี้จะใช้นิยามของรูปแบบสถาปัตยกรรมซอฟต์แวร์ของ Shaw และ Garlan [6] เท่านั้น โดยเฉพาะ สถาปัตยกรรมในรูปแบบรวมศูนย์กลาง สถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้นสถาปัตยกรรมไปป์และฟิลเตอร์ และ สถาปัตยกรรมในรูปแบบลำดับชั้น

ตารางที่ 2.1 กลุ่มของสถาปัตยกรรมซอฟต์แวร์ที่ Shaw และ Garlan เสนอ [1]

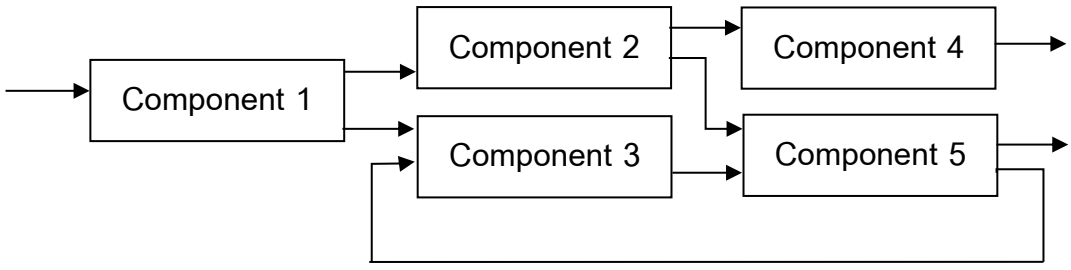
Category	Style
Data Centered	-Repository -Blackboard
Data Flow	-Pipe-and-Filter -Batch Sequential
Virtual Machine	-Rule-Based System -Interpreter

ตารางที่ 2.1 กลุ่มของสถาปัตยกรรมซอฟต์แวร์ที่ Shaw และ Garlan เสนอ [1] (ต่อ)

Category	Style
Call and Return	-Main program and Subroutine -Objected Oriented -Layered
Independent Components	-Communicating Process -Client-Server -Event Systems

2.1.1.1 รูปแบบสถาปัตยกรรมไปป์และฟิลเตอร์ (Pipe and Filter) [6]

โดยลักษณะของสถาปัตยกรรมไปป์และฟิลเตอร์ประกอบด้วยหลาย “คอมโพเนนท์” แต่ละคอมโพเนนท์จะมีหน้าที่แตกต่างกันโดยจะประกอบด้วยกลุ่มของ “สายข้อมูลนำเข้า” และกลุ่มของ "สายข้อมูลนำออก” ซึ่งคอมโพเนนท์จะทำการอ่านข้อมูลที่เข้ามาทางสายข้อมูลนำเข้าและทำการประมวลผล แล้วส่งออกไปยังสายข้อมูลนำออก และยังสามารถเรียกคอมโพเนนท์ต่างๆ ได้อีกชื่อหนึ่งว่า "ฟิลเตอร์ (Filters)" เนื่องจากคอมโพเนนท์ต่างๆ มีหน้าที่ในการกรองข้อมูลที่ได้จนได้ผลลัพธ์ที่ต้องการ สำหรับสายเชื่อมต่อมีหน้าที่เป็นท่อส่งสายข้อมูล สำหรับส่งผลลัพธ์จาก ตัวกรองหนึ่งไปสู่ตัวกรองถัดๆไป ซึ่งสายเชื่อมต่อจะเรียกว่า "ไปป์ (Pipe)" ซึ่งลักษณะของสถาปัตยกรรมไปป์และฟิลเตอร์มีลักษณะดังรูปที่ 2.1

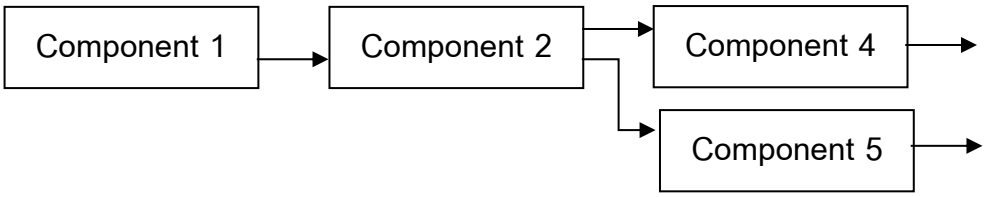


รูปที่ 2.1 สถาปัตยกรรมไปป์และฟิลเตอร์ [6]

สถาปัตยกรรมไปป์และฟิลเตอร์นั้นสามารถแบ่งเป็น 3 ลักษณะ

ก) สถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์

เป็นลักษณะของสถาปัตยกรรมที่มีประกอบด้วยฟิลเตอร์หนึ่งใดมีไปป์เชื่อมฟิลเตอร์ถัดไปอยู่ในรูปแบบลำดับเชิงเส้น (linear sequences) ซึ่งไม่มีข้อจำกัดอื่นๆ แสดงตัวอย่างของไปป์ไลน์ในรูปที่ 2.2



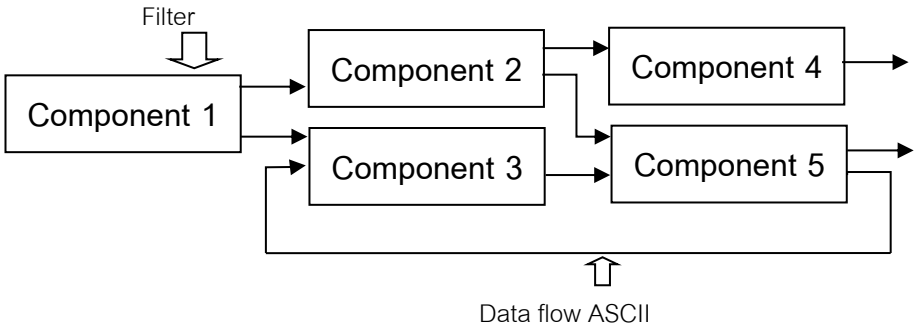
รูปที่ 2.2 สถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ [6]

ข) สถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะบาวไปป์ (Bound pipes)

เป็นลักษณะของสถาปัตยกรรมที่มีประกอบด้วยฟิลเตอร์หนึ่งใดมีไปป์เชื่อมฟิลเตอร์ถัดไป ซึ่งมีข้อแตกต่างจากสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ คือ จำนวนปริมาณของข้อมูลที่จะผ่านไปบนไปป์ได้

ค) สถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ (Typed pipes)

เป็นลักษณะของสถาปัตยกรรมที่มีประกอบด้วยฟิลเตอร์หนึ่งใดมีไปป์เชื่อมฟิลเตอร์ถัดไป ซึ่งมีข้อแตกต่างจากสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะอื่นๆ คือ ชนิดของข้อมูลที่รับระหว่างฟิลเตอร์จะต้องเหมือนกัน แสดงตัวอย่างของไทป์ไปป์ในรูปที่ 2.3



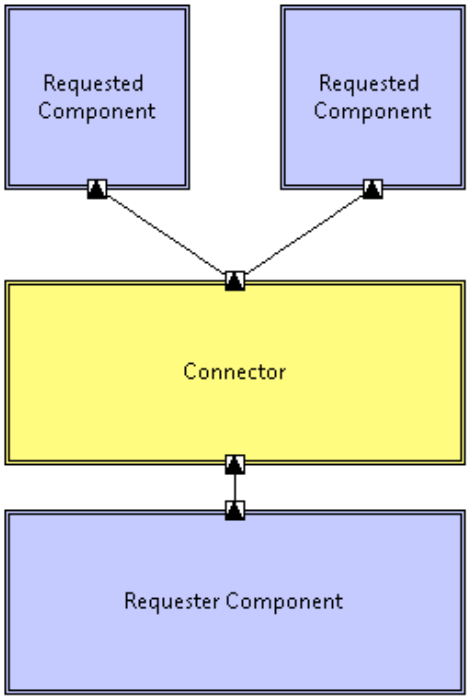
รูปที่ 2.3 สถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ [6]

2.1.1.2 รูปแบบสถาปัตยกรรมที่ทำงานตามเหตุการณ์ที่เกิดขึ้น (EventBase) [6]

สถาปัตยกรรมในรูปแบบทำงานตามเหตุการณ์ที่เกิดขึ้น นั้นมีพื้นฐานมาจากสถาปัตยกรรมที่ใช้ผู้กระทำเป็นหลัก โดยสถาปัตยกรรมนี้มีแนวคิดในการเรียกใช้บริการต่างๆ โดยตรง ซึ่งมีขั้นตอนการทำงาน ดังนี้

1) คอมโพเนนท์ต่างๆ ทำการประกาศว่า คอมโพเนนท์ดังกล่าวสามารถรองรับเหตุการณ์ ใดจากผู้กระทำได้บ้าง
2) คอมโพเนนท์อื่นๆ ที่เกี่ยวข้องในระบบสามารถทำการลงทะเบียนได้ว่ามีความสนใจในเหตุการณ์ใด โดยการเชื่อมโยงบริการกับเหตุการณ์ที่เกี่ยวข้อง
3) เมื่อผู้ใช้ได้กระทำเหตุการณ์หนึ่งเหตุการณ์ใด ระบบจะไปทำการเรียกทุกๆ บริการที่ได้ลงทะเบียนไว้

แสดงตัวอย่างของสถาปัตยกรรมที่ทำงานตามเหตุการณ์ที่เกิดขึ้นในรูปที่ 2.4



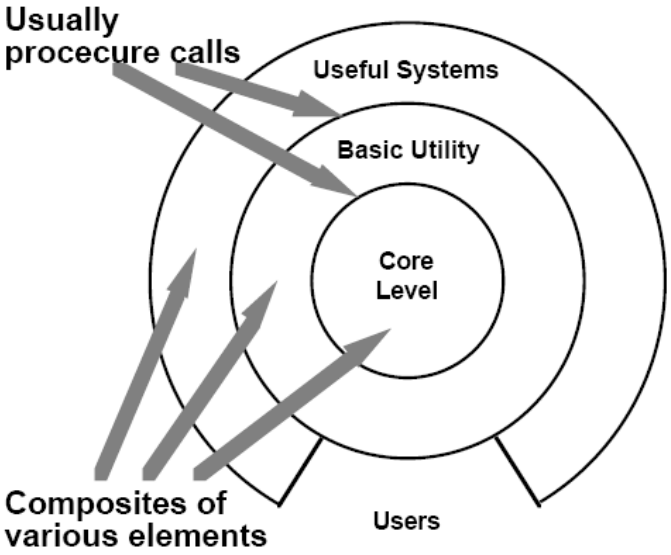
รูปที่ 2.4 สถาปัตยกรรมที่ทำงานตามเหตุการณ์ที่เกิดขึ้น [8]

จากรูปที่ 2.4 แสดงตัวอย่างของสถาปัตยกรรมที่ทำงานตามเหตุการณ์ที่เกิดขึ้น โดยเมื่อคอมโพเนนท์ที่เป็นตัวสั่งงานซึ่งเป็นผู้กระทำได้สั่งงานไปให้คอมโพเนนท์ที่มีคุณลักษณะสถาปัตยกรรมที่ทำงานตามเหตุการณ์ที่เกิดขึ้นทำงานตามเหตุการณ์ ซึ่งในที่นี้หมายถึง คอมโพเนนท์ที่มีชื่อว่า Connector คอมโพเนนท์ดังกล่าวก็จะทำการกระจายคำสั่งต่อไปยัง คอมโพเนนท์ที่ได้ลงทะเบียนไว้ตามเหตุการณ์ที่สนใจเพื่อให้คอมโพเนนท์ดังกล่าวตัดสินใจว่าจะทำงานหรือไม่ต่อไป

2.1.1.3 รูปแบบสถาปัตยกรรมในรูปแบบลำดับชั้น (Layered Systems) [6]

เป็นลักษณะสถาปัตยกรรมที่จัดการข้อมูลแบบลำดับชั้นการทำงาน ซึ่งแต่ละลำดับชั้นจะเตรียมบริการสำหรับให้ลำดับชั้นที่มีการทำงานใกล้กับผู้ใช้มากขึ้น โดยที่บางครั้งในแต่ละลำดับชั้นจะปกปิดข้อมูลจากชั้นที่อยู่ติดกัน ยกเว้นสำหรับบางบริการ เพื่อใช้ในการงานเฉพาะอย่าง ในบางครั้งส่วนประกอบที่อยู่ในลำดับชั้นต่างๆ อาจจะอยู่ในรูปแบบระบบเสมือน สำหรับเรียกใช้งาน ซึ่งเชื่อมต่อกันโดยใช้กฎที่มีข้อกำหนดในการติดต่อลำดับชั้นที่อยู่ติดกัน ซึ่งมีลักษณะดังรูปที่ 2.5

จากรูปที่ 2.5 แสดงตัวอย่างของสถาปัตยกรรมในรูปแบบลำดับชั้น โดยลำดับชั้น Basic Utility จะทำงานให้บริการแก่ลำดับชั้น Useful Systems และ ลำดับชั้น Core Level จะทำงานให้บริการแก่ลำดับชั้น Basic Utility โดยลำดับชั้น Useful System จะไม่สามารถติดต่อโดยตรงกับลำดับชั้น Core Level ได้



รูปที่ 2.5 สถาปัตยกรรมในรูปแบบลำดับชั้น [6]

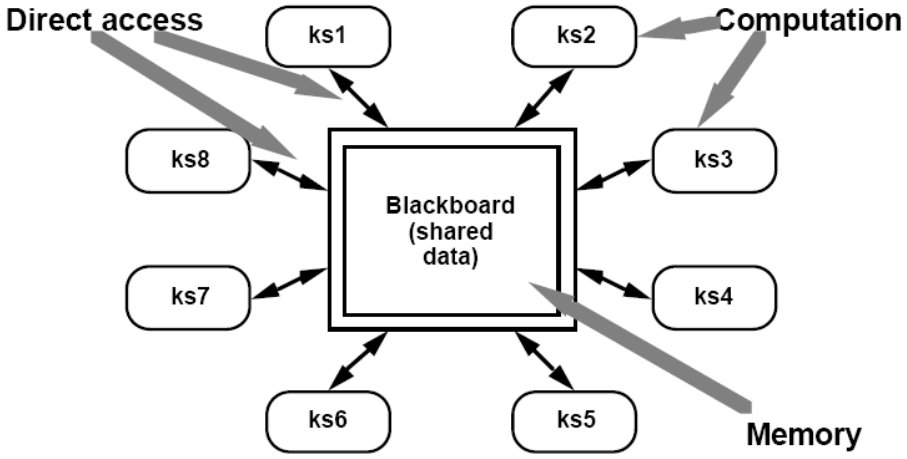
2.1.1.4 รูปแบบสถาปัตยกรรมรูปแบบรวมศูนย์กลาง (Repository)

ในสถาปัตยกรรมแบบรูปแบบรวมศูนย์กลางนั้นแบ่งส่วนประกอบ ออกเป็น 2 ชนิด

1. ส่วนโครงสร้างข้อมูลรวมศูนย์กลาง (Central data structure) อธิบายถึงสถานะปัจจุบัน

2. ชุดของส่วนประกอบที่มีการทำงานร่วมกับส่วนโครงสร้างข้อมูลรวมศูนย์กลาง

โดยที่การปฏิสัมพันธ์ระหว่างรูปแบบสถาปัตยกรรมรูปแบบรวมศูนย์กลาง กับส่วนประกอบอื่นๆที่เกี่ยวข้องนั้นจะขึ้นอยู่กับระบบที่มาติดต่อด้วย ซึ่งถ้ามีรายการออกมาจากข้างนอกสถาปัตยกรรมรูปแบบรวมศูนย์กลาง ก็จะทำตัวการไปสืบค้นข้อมูลจากฐานข้อมูลโดยตรง ดังรูปที่ 2.6



รูปที่ 2.6 สถาปัตยกรรมสถาปัตยกรรมรูปแบบรวมศูนย์กลาง [6]

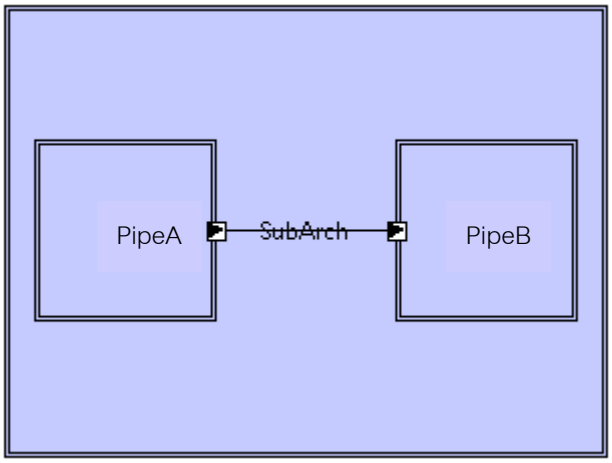
2.1.2 ภาษาเอกซ์เอดีแอล (xADL) [4]

ภาษาอธิบายสถาปัตยกรรมที่ขยายความได้ (Extensible Architecture Description Language: xADL) เป็นภาษาที่ได้รับการพัฒนามาจากภาษาที่ใช้สำหรับระบุลักษณะรูปแบบของสถาปัตยกรรมซอฟต์แวร์ โดยมีการกำหนดรูปแบบโครงแบบให้อยู่ในรูปของ เอ็กเอ็มแอลโดยภาษาเอกซ์เอดีแอลนั้น มีเป้าหมาย

เพื่อเพิ่มความสามารถใหม่โดยไม่ต้องเปลี่ยนแปลงโครงสร้างของภาษาที่มีอยู่เดิม โดย ภาษาเอกซ์เอดีแอลนั้นมีคุณสมบัติหลัก 3 ประการ คือ

1) ภาษาเอกซ์เอดีแอลสามารถที่จะแยกโครงสร้างระหว่าง “การออกแบบ” และ “การทำงาน” ได้

ในอดีตภาษาที่ถูกพัฒนาขึ้นมาโดยมีพื้นฐานมาจากภาษาที่ใช้สำหรับระบุลักษณะรูปแบบของสถาปัตยกรรมซอฟต์แวร์ นั้นจะสามารถพัฒนาโครงสร้างได้เฉพาะด้านการออกแบบเท่านั้น ดังนั้นจึงเกิดปัญหาสำหรับระบบที่มีการเปลี่ยนสถาปัตยกรรมระหว่างการทำงานขึ้น ซึ่งจากที่กล่าวมาจึงมีความต้องการที่จะแยกโครงสร้างด้านการออกแบบและด้าน การทำงานออกจากกัน โดยโครงสร้างของด้านการทำงานนั้นจะอธิบายถึงสถานะของระบบ เช่น ทำงาน ปิดกั้น เป็นต้น และโครงสร้างของด้านการออกแบบจะอธิบายถึงข้อมูลเกี่ยวกับส่วนประกอบต่างๆ เช่น ผู้แต่ง วันที่สร้าง เป็นต้น โดยที่โครงสร้างด้านการออกแบบนั้นจะอธิบายถึงข้อจำกัดต่างๆบนระบบ และงานที่คาดหวังว่าจะได้จากระบบ และในด้านการออกแบบยังสามารถระบุลักษณะคอมโพเนนท์ที่เป็นแบบสถาปัตยกรรมซ้อนสถาปัตยกรรมได้โดยมีลักษณะดังรูปที่ 2.7



รูปที่ 2.7 รูปแบบสถาปัตยกรรมที่มีสถาปัตยกรรมซ้อนอยู่ด้านใน

2) ภาษาเอกซ์เอดีแอลสามารถที่จะจับคู่ระหว่างการพัฒนาได้

อีกความต้องการหนึ่งที่ภาษาที่พัฒนามาจากภาษาที่ใช้สำหรับระบุลักษณะรูปแบบของสถาปัตยกรรมซอฟต์แวร์ต้องการ คือ การทำให้การเขียนโปรแกรมสามารถทำได้สะดวกมากยิ่งขึ้น แต่เป็นการยากที่จะกำหนดรูปแบบที่แน่นอนเนื่องจากผู้วิจัยไม่สามารถทราบได้ว่า

ในการใช้งานจริงจะใช้บนระบบแบบใด ดังนั้นในภาษาเอกซ์เอดีแอล รุ่นที่ 2.0 ได้มีการเพิ่มเติมเค้าร่าง (Schema) ของ Structure และ Type เพื่อเพิ่มความสามารถในการเชื่อมโยงจากส่วนประกอบต่างๆ ไปยังส่วนของการเขียนโปรแกรม

3) การสร้างรูปแบบสถาปัตยกรรมในรูปแบบสายการผลิต (Product line)

การพัฒนาสถาปัตยกรรมในรูปแบบสายการผลิต นั้นเป็นแนวคิดที่ใหม่และเป็นสิ่งที่มีการศึกษาค้นคว้าอย่างต่อเนื่อง โดยที่ภาษาเอกซ์เอดีแอลได้พัฒนารูปแบบเค้าร่างไว้ใช้สำหรับสถาปัตยกรรมในรูปแบบสายการผลิต ซึ่งมีส่วนประกอบดังนี้

1) การเก็บบันทึกรุ่น (Versions)

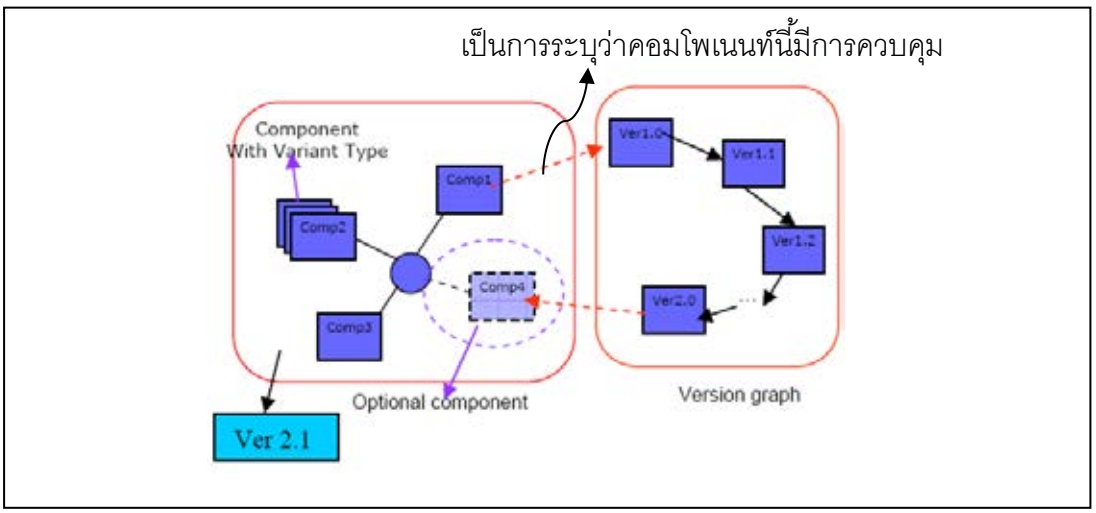
ภาษาเอกซ์เอดีแอล สามารถที่จะทำการจัดเก็บข้อมูลเกี่ยวกับวิวัฒนาการของสถาปัตยกรรมและโครงสร้างต่างๆได้โดยผ่านเค้าร่างที่ชื่อเวอร์ชัน ซึ่งข้อมูลต่างๆที่ถูกเก็บไว้สามารถนำมาแสดงให้อยู่ในรูปกราฟของรุ่นสถาปัตยกรรม (Version Graph) ได้

2) การเปลี่ยนแปลงรูปแบบตามสถาปัตยกรรมที่เปลี่ยนไป

ภาษาเอกซ์เอดีแอล อนุญาตให้มีการเปลี่ยนชนิดโครงสร้างในขั้นตอนการออกแบบให้เป็นไปตามลักษณะของสถาปัตยกรรมที่เปลี่ยนไปโดยผ่านเค้าร่างที่ชื่อ VARIANTS

3) การเพิ่มทางเลือกสำหรับเค้าร่าง

ภาษาเอกซ์เอดีแอล ได้เพิ่มเค้าร่าง OPTIONS เพื่ออธิบายส่วนประกอบเพิ่มเติมในตัวโปรแกรมซึ่งลักษณะทั้งหมดที่ได้กล่าวไปสามารถอธิบายได้ดังรูปที่ 2.8

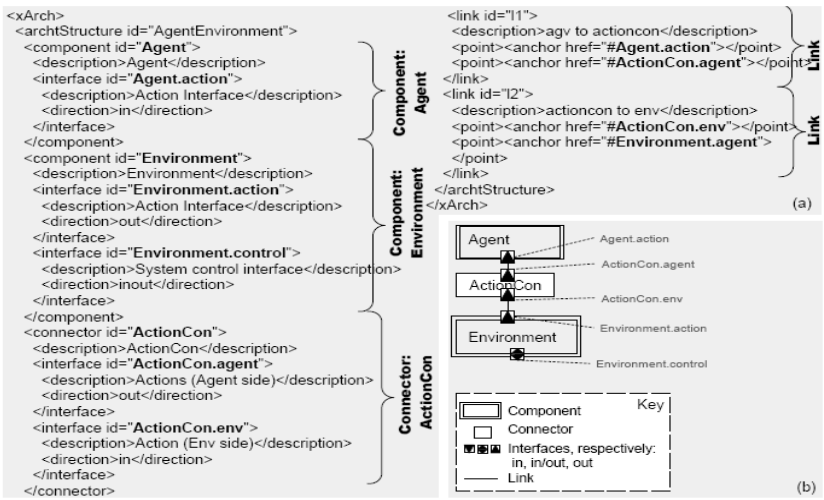


รูปที่ 2.8 การสร้างรูปแบบสถาปัตยกรรมในรูปแบบสายการผลิต [4]

ในภาษาเอกซ์เอดีแอล ประกอบด้วยส่วนประกอบพื้นฐานทั้งหมด 4 ส่วน ดังนี้

- **คอมโพเนนท์** ใช้เป็นสัญลักษณ์การแสดงถึงส่วนประกอบในระบบ ตัวอย่าง เช่น เครื่องลูกข่าย เครื่องแม่ข่าย และฐานข้อมูล เป็นต้น ซึ่งจะต้องมีอินเตอร์เฟส ไว้สำหรับติดต่อสื่อสาร
- **คอนเนคเตอร์** (Connector) ใช้เป็นสัญลักษณ์ที่มีไว้เพื่อบอกเกี่ยวกับการสื่อสาร ตัวอย่างเช่น Http Snmp และ Ftp เป็นต้น ซึ่งจะต้องมีอินเตอร์เฟส ไว้สำหรับติดต่อสื่อสาร เช่นเดียวกับ คอมโพเนนท์
- **อินเตอร์เฟส** เป็นช่องทางให้คอมโพเนนท์ และ คอนเนคเตอร์ เชื่อมต่อโลกภายนอก ซึ่งจะเรียกว่า พอร์ต (Port) สำหรับ คอมโพเนนท์ และ บทบาท (Role) สำหรับ คอนเนคเตอร์ในภาษาที่ใช้สำหรับระบุลักษณะรูปแบบของสถาปัตยกรรมซอฟต์แวร์แบบอื่นๆ เช่น ภาษา ACME ภายในอินเตอร์เฟส จะต้องมีการระบุทิศทางของอินเตอร์เฟสด้วย
- **ลิ้งก์ (Link)** เป็นช่องทางในการเชื่อมต่อระหว่างอินเตอร์เฟสในระบบ

รูปที่ 2.9 แสดงตัวอย่างโครงสร้างของสถาปัตยกรรมที่ประกอบด้วย คอมโพเนนท์ เชื่อมต่อกันโดยมีตัวกลางมีลักษณะเป็นคอนเนคเตอร์ และ ลิ้งก์สำหรับเชื่อมต่อโครงสร้างต่างๆเข้าด้วยกัน คอมโพเนนท์ ที่ชื่อ Environment นั้นเป็น คอมโพเนนท์ ที่มีการให้บริการกับบุคคลอื่น ซึ่งจะต้องมีอินเตอร์เฟส ไว้สำหรับเป็นช่องทางสำหรับการเชื่อมต่อกับโลกภายนอกโดยที่ ช่องทางนั้นจะต้องมีค่าเป็นรหัสที่ไม่ซ้ำกัน และจะมีการใช้รหัสนี้ในการอ้างอิงระหว่างอินเตอร์เฟส และ ลิ้งก์อีกด้วย



รูปที่ 2.9 ตัวอย่างโครงสร้างของภาษาเอกซ์เอดีแอล [4]

2.1.3 ทฤษฎีไวยากรณ์พึ่งบริบท (Context-Sensitive Grammar) [9]

ทฤษฎีไวยากรณ์พึ่งบริบท เป็นส่วนหนึ่งของไวยากรณ์ที่มีรูปแบบแน่นอน โดยในกฎการผลิตจะประกอบไปด้วย สัญลักษณ์ที่แทนเทอร์มินัล (Terminal) และนอนเทอร์มินัล (Nonterminal) ซึ่งจะมีรูปแบบในการเขียนในลักษณะของกฎการผลิต โดยจะมีด้านขวา (Right-hand sides) และด้านซ้าย (Left-hand sides) ในทุกกฎการผลิต

โดยทฤษฎีไวยากรณ์พึ่งบริบทถูกคิดค้นโดย Noam Chomsky ในปี 1950 [8] เพื่อใช้อธิบายถึงลักษณะของภาษาธรรมชาติ ที่นำมาเป็นทางเลือกในการตัดสินใจและลดความกำกวมในการตัดสินใจลง ซึ่งต่อมาได้มีการสร้างภาษาที่ใช้ทฤษฎีไวยากรณ์พึ่งบริบทมาประยุกต์ เรียกว่า ภาษาทางการ (Formal Language) โดยได้มีการนิยามลักษณะของคอนเท็กซ์เซ็นซิทีปแกรมมาไว้ดังนี้

$$G = (N, \Sigma, P, S)$$

โดยที่ N = ชุดของสัญลักษณ์ของนอนเทอร์มินัล

Σ = ชุดของสัญลักษณ์ของเทอร์มินัล

P = ชุดของกฎการผลิต

S = ชุดของจุดเริ่มต้น

โดยกราฟที่สร้างขึ้นมาดังกล่าวจะมีลักษณะของไวยากรณ์พึ่งบริบทอยู่ก็ต่อเมื่อ P อยู่ในรูปแบบดังนี้

$$\alpha A \beta \rightarrow \alpha \gamma \beta$$

A = ชุดของสัญลักษณ์ของนอนเทอร์มินัล

γ = ชุดของสัญลักษณ์ของนอนเทอร์มินัล หรือเทอร์มินัล

α, β = ชุดของสัญลักษณ์ของนอนเทอร์มินัล หรือเทอร์มินัล

ซึ่งลักษณะของทฤษฎีไวยากรณ์พึ่งบริบทจะอธิบายถึงการแทนที่ว่า A สามารถแทนที่ได้ด้วย γ หรือไม่ ซึ่งต่างจากไวยากรณ์ไม่พึ่งบริบท ซึ่งจะไม่ยอมให้ A แทนที่ด้วย γ

2.2 งานวิจัยที่เกี่ยวข้อง

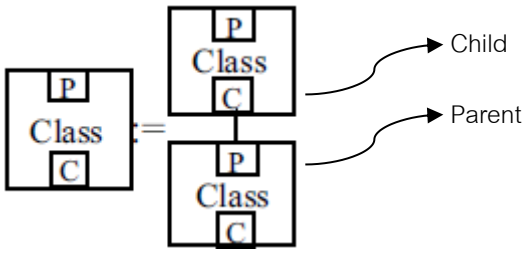
2.2.1 การตรวจสอบสถาปัตยกรรมและโครงสร้างโดยใช้กราฟแกรมมา [7]

การตรวจสอบสถาปัตยกรรมและโครงสร้างโดยใช้กราฟแกรมมาโดย Jing Dong Guanglei Song และ J. K. Kang Zhang ปี ค.ศ. 2003 ได้นำเสนอวิธีการตรวจสอบโครงสร้างของซอฟต์แวร์โดยการใช้ ลักษณะของกราฟที่เรียกว่า กราฟแกรมมา ซึ่งจะมีความเฉพาะเจาะจงมากกว่าการใช้แผนภาพยูเอ็มแอล (UML Diagram) ที่มีความกำกวมและไม่สามารถใช้เครื่องมืออัตโนมัติในการตรวจสอบได้

งานวิจัยนำเสนอกราฟแกรมมา และใช้กราฟแกรมมาเพื่อลดรูปของกราฟโดยอาศัยหลักการ Reverse Graph Grammar ซึ่งประกอบด้วยส่วนประกอบดังต่อไปนี้

1) **กฎการผลิต (Production Rule)** กฎการผลิตที่ประกอบด้วยกราฟด้านซ้ายและกราฟด้านขวาโดยมีเครื่องหมาย ::= คั่น แสดงถึงการแทนที่กราฟด้านซ้ายด้วยกราฟข้างขวา และในทางกลับกันผู้วิจัยสามารถ แทนที่กราฟด้านขวาด้วยกราฟด้านซ้ายได้

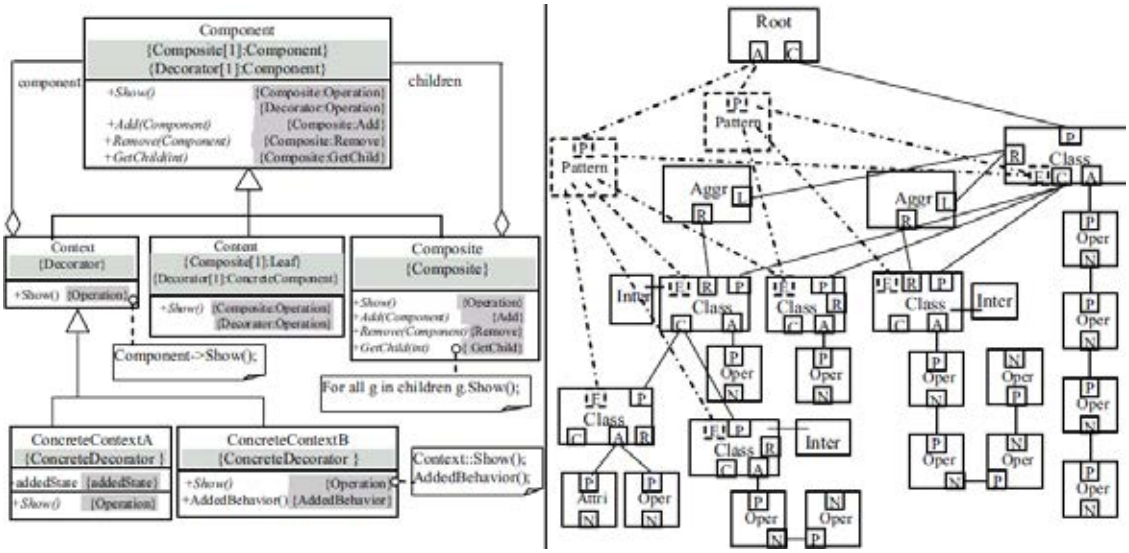
2) **สัญลักษณ์ที่ใช้ในกราฟแกรมมา** สัญลักษณ์ที่ใช้ในกราฟแกรมมานั้นจะประกอบด้วยกล่องสี่เหลี่ยมที่ใหญ่ที่สุด จะถูกเรียกว่า Super Vertex และภายในจะประกอบไปด้วยกล่องสี่เหลี่ยมขนาดเล็ก ที่เรียกว่า Vertex หรือ Vertices โดยแต่ละกล่องจะมีความเป็นเอกลักษณ์เป็นของตัวเอง และในกราฟแกรมมาที่อธิบายในงานวิจัยนี้จะประกอบด้วย Edge ที่มีหน้าที่สำหรับเชื่อมต่อระหว่าง Vertex หรือ Super Vertex ดังรูปที่ 2.10



รูปที่ 2.10 ลักษณะของ Super Vertex และ Vertex [7]

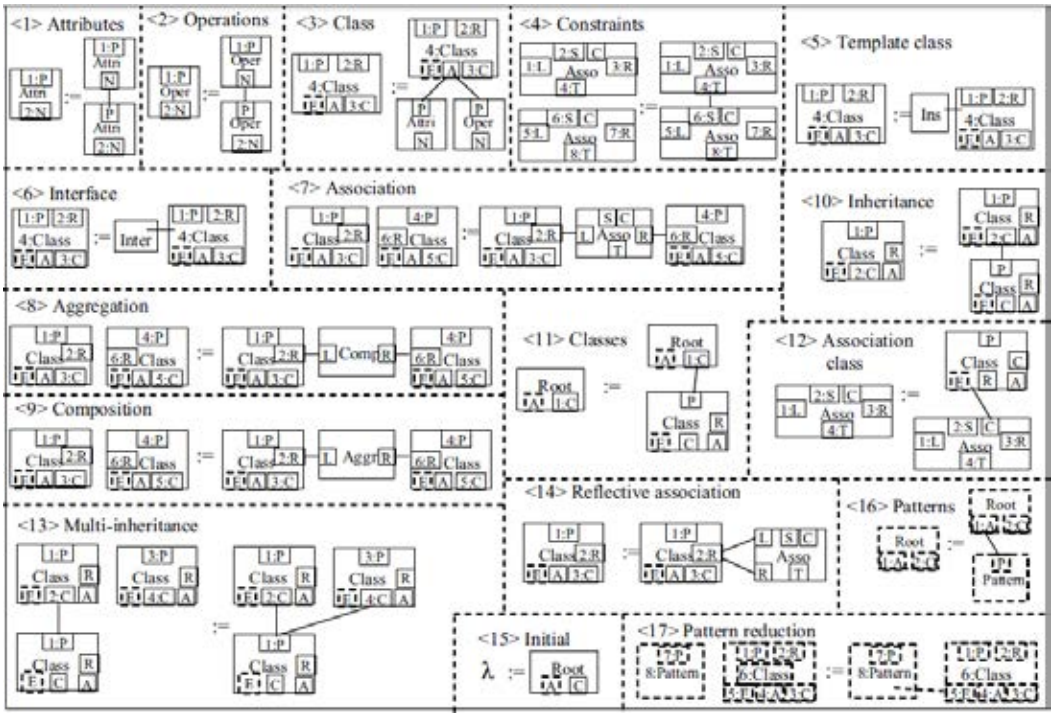
ในงานวิจัยได้นำเสนอการนำกราฟแกรมมาที่อธิบายข้างต้นมาประยุกต์ใช้กับแผนภาพของคลาสในรูปแบบยูเอ็มแอล (UML Class Diagram) โดยผู้วิจัยสามารถที่จะเขียนแผนภาพคลาสในรูปแบบยูเอ็มแอลด้วยกราฟแกรมมาได้ ซึ่งจะได้ผลลัพธ์ดังรูปที่ 2.11 โดยมีการกำหนดชื่อของบัพให้

สอดคล้องกับลักษณะของแผนภาพของคลาส และจะมีการกำหนด ชื่อของ Vertex หรือ Super Vertex เพิ่มเติมเพื่อให้สอดคล้องกับคุณสมบัติต่างๆที่แผนภาพคลาสในรูปแบบยูเอ็มแอลมี ดังตัวอย่าง เช่น Associations , Oper , Class , Attri หรือ Asso เป็นต้น



รูปที่ 2.11 ตัวอย่างกราฟแกรมมาที่แปลงมาจากแผนภาพคลาส [7]

และในงานวิจัยดังกล่าวได้มีการกำหนดรูปแบบของกฎการผลิต สำหรับการทำการแปลงรูปร่างเพื่อใช้ในการตรวจสอบสถาปัตยกรรมหรือ แบบรูปการออกแบบ (Design Pattern) โดยมีการกำหนดดังรูปที่ 2.12 โดยสำหรับงานของผู้วิจัยจะเอาเทคนิคเดียวกันนี้เป็นส่วนหนึ่งในการตรวจจับรูปแบบสถาปัตยกรรมเนื่องจากกราฟแกรมมา มีความสามารถในการเพิ่มเติมส่วนประกอบให้สอดคล้องกับโครงสร้างของแต่ละรูปแบบภาษา



รูปที่ 2.12 ตัวอย่างกราฟแกรมมาที่แปลงมาจากแผนภาพคลาส [7]

2.2.2 ทฤษฎีกราฟแกรมมาสำหรับไวยากรณ์พึ่งบริบทแบบเอ็นซีอี (A Context-Sensitive NCE Graph Grammar) [9]

ทฤษฎีกราฟแกรมมาสำหรับไวยากรณ์พึ่งบริบทแบบเอ็นซีอีเป็นลักษณะของกราฟแกรมมาที่ถูกพัฒนาขึ้นโดย Yoshihiro Adachi และ Yuichi Nakajima ซึ่งเป็นส่วนเพิ่มเติมของ NCE Graph Grammar ซึ่งมีความยืดหยุ่นสำหรับที่จะเป็นกฎที่ไว้ใช้กับไวยากรณ์ของกราฟ ซึ่งในกราฟลักษณะนี้มีคุณสมบัติที่สามารถกระจาย หรือความสามารถที่จะค้นหาที่มาของกราฟโดยการใช้คุณสมบัติ Reverse application ของแกรมมาที่ระบุไว้ได้ โดยคุณสมบัติ Reverse application คือ การความสามารถในการย้อนกลับกฎที่นำมาใช้กับไวยากรณ์ของกราฟได้ และในทฤษฎียังได้ระบุถึงเงื่อนไขสำหรับการตรวจสอบว่าแกรมมาที่เขียนขึ้นมานั้นมีคุณสมบัติตามทฤษฎีกราฟแกรมมาสำหรับไวยากรณ์พึ่งบริบทแบบเอ็นซีอีหรือไม่ ซึ่งมีการอธิบายถึงลักษณะของทฤษฎีกราฟแกรมมาสำหรับไวยากรณ์พึ่งบริบทแบบเอ็นซีอีไว้ในนิยามดังต่อไปนี้

1) นิยามที่ 1 กราฟ (Graph)

ให้ Σ เป็นตัวอักษรสำหรับอธิบายบัพ และให้ Γ เป็นตัวอักษรสำหรับอธิบายเส้น ซึ่งกราฟที่อยู่ภายใต้ Σ และ Γ ถูกเขียนโดยกฎการผลิตดังต่อไปนี้

$$H = (V, E, \lambda)$$

โดยที่

V = ชุดของบัพ

E = ชุดของด้าน

λ = เซ็ตของบัพฟังก์ชั่น

โดยผู้ที่นำเสนองานวิจัยใช้ตัวอักษร “d” เพื่อแสดงว่าทฤษฎีกราฟแกรมมาสำหรับไวยากรณ์พึ่งบริบทแบบเอ็นซีอี ได้สร้างกราฟที่มีลักษณะกราฟระบุทิศทาง (Directed graph) และใช้ตัวอักษร “e” เพื่อแสดงว่าทฤษฎีกราฟแกรมมาสำหรับไวยากรณ์พึ่งบริบทแบบเอ็นซีอี ได้สร้างกราฟที่มีลักษณะคำอธิบายเส้นซึ่งจากที่กล่าวไปข้างต้นจะทำให้มีทฤษฎีกราฟแกรมมาสำหรับไวยากรณ์พึ่งบริบทแบบเอ็นซีอีสามลักษณะ คือ dCS-NCE graph grammar , eCS-NCE graph grammar และ edCS-NCE graph grammar และในทฤษฎีจะแทนกราฟที่อยู่ภายใต้ Σ และ Γ โดยใช้สัญลักษณ์ $GR_{\Sigma,T}$

2) นิยาม 2 กราฟแกรมมาสำหรับไวยากรณ์พึ่งบริบทแบบเอ็นซีอีในรูปแบบย่อยอีดี (edCS-NCE graph grammar)

ลักษณะของ edCS-NCE Graph Grammar นั้นสามารถเขียนได้โดยใช้กฎการผลิตดังต่อไปนี้

$$G = (\Sigma N, \Sigma T, \Gamma N, \Gamma T, s, P)$$

โดยที่

ΣN = เซ็ตของสัญลักษณ์ของนอนเทอร์มินัล (Nonterminal node)

ΣT = เซ็ตของสัญลักษณ์ของเทอร์มินัล (Terminal node)

$\Sigma = \Sigma N \cup \Sigma T$

ΓN = เซ็ตของด้านที่นับไม่ได้ (Nonfinal Edge)

ΓT = เซ็ตของด้านที่นับได้ (Final Edge)

S = จุดเริ่มต้นของกราฟ โดยที่จะต้องอยู่ในชุดของสัญลักษณ์นอนเทอร์มินัล ซึ่ง H_S ที่เป็นกราฟเริ่มต้นจะประกอบไปด้วยบัพ ที่เป็นจุดเริ่มต้น S บัพเดียวโดยไม่มีเส้นเชื่อมต่อ

P = เป็นลักษณะของกฎการผลิตที่ถูกเขียนขึ้นมาให้อยู่ในรูป

P= (A::= B,C)

โดย

A = ด้านซ้ายของกฎการผลิตเขียนแทนด้วย lhs(p)

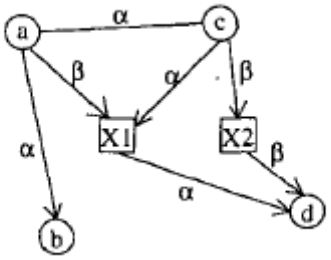
B = ด้านขวาของกฎการผลิตเขียนแทนด้วย rhs(p)

C = แทนข้อกำหนดของจุดเชื่อมต่อ

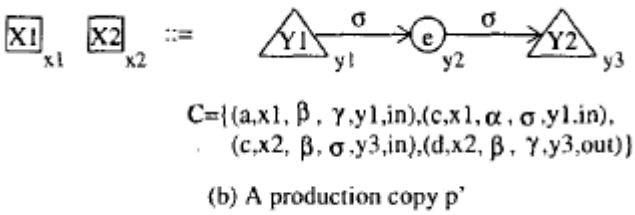
3) นิยาม 3 การคืนรูป (Derivation)

กำหนดให้ $G = (\Sigma_N, \Sigma_T, \Gamma_N, \Gamma_T, s, P)$ เป็นกราฟชนิด edCS-NCE graph grammar และกำหนดให้ H เป็นกราฟที่อยู่ใน $GR_{\Sigma,T}$ ถ้ากราฟ H เปลี่ยนรูปร่างเป็นกราฟ H' โดยใช้ลักษณะของด้านขวาของกฎการผลิต จะเขียนแทนด้วย $H \xrightarrow[p']{} H'$ ซึ่งจะเรียกว่า *Derivation step* และขั้นตอนการทำ Derivation step จะเรียกว่า Derivation ดังตัวอย่างต่อไปนี้กำหนดให้

รูปที่ 2.13 เป็นกราฟเริ่มต้น และรูปที่ 2.14 เป็นกฎที่เขียนขึ้นมาในรูป A::= B,C

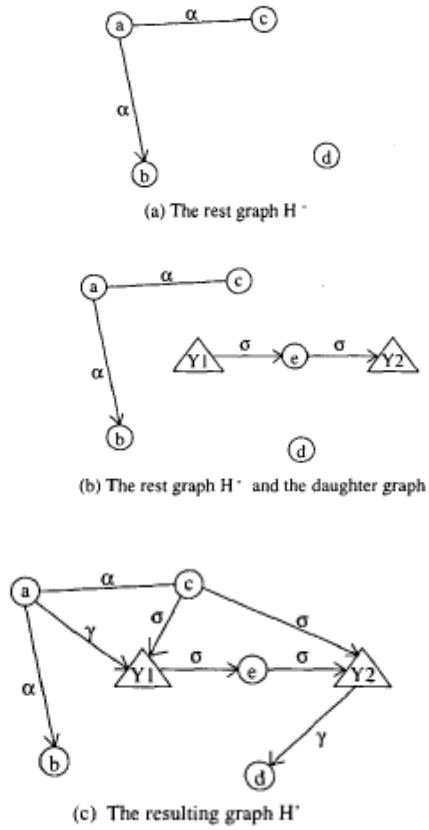


รูปที่ 2.13 ตัวอย่างกราฟเริ่มต้น [9]



รูปที่ 2.14 กฎที่เขียนขึ้นมาในรูป A::= B,C [9]

ซึ่งในทฤษฎีสามารถเขียน Derivation step ได้ดังรูปที่ 2.15 โดยผลลัพธ์ของการทำ Derivation จะอยู่ในรูปที่ 2.15C

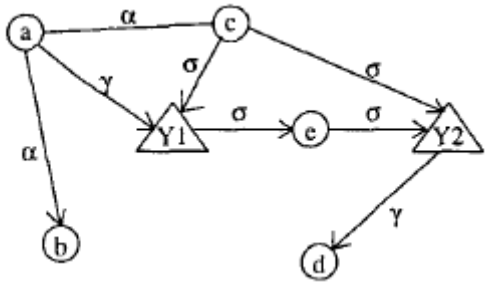


รูปที่ 2.15 ขั้นตอนการทำ Derivation step [9]

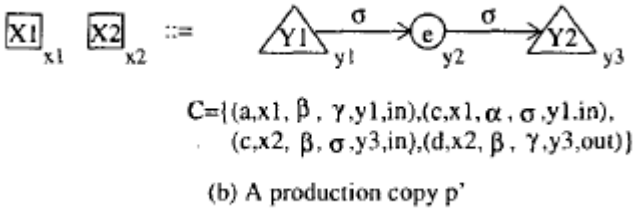
4) นิยาม 4 การลดรูป (Reduction)

กำหนดให้ $G = (\Sigma N, \Sigma T, \Gamma N, \Gamma T, s, P)$ เป็นกราฟชนิด edCS-NCE graph grammar และกำหนดให้ H เป็นกราฟที่อยู่ใน $GR_{\Sigma,T}$ ถ้ากราฟ H เปลี่ยนรูปร่างเป็นกราฟ H' โดยใช้ลักษณะของด้านซ้ายของกฎการผลิต จะเขียนแทนด้วย $H \xrightarrow[p]{} H'$ ซึ่งจะเรียกว่า Reduction *step* และขั้นตอนการทำ Reduction step จะเรียกว่า Reduction ดังตัวอย่างต่อไปนี้

กำหนดให้รูปที่ 2.16 เป็นกราฟเริ่มต้น และรูปที่ 2.17เป็นกฎที่เขียนขึ้นมาในรูป A::= B,C

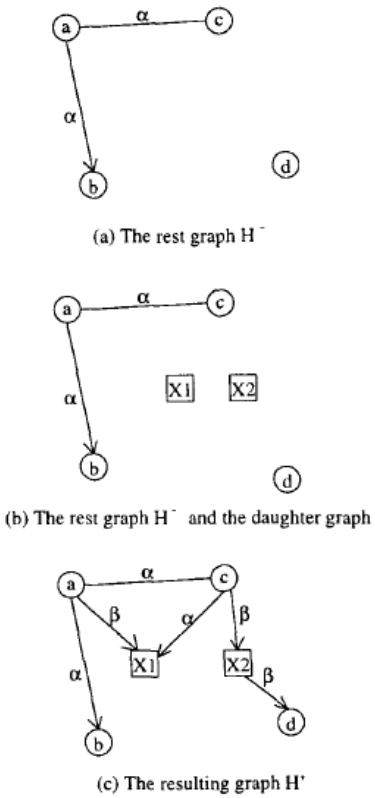


รูปที่ 2.16 ตัวอย่างกราฟเริ่มต้น [9]



รูปที่ 2.17 กฎที่เขียนขึ้นมาในรูป A::= B,C [9]

ซึ่งผู้วิจัยสามารถเขียน Reduction step ได้ดังรูปที่ 2.18 โดยผลลัพธ์ของการทำ Reduction จะอยู่ในรูปที่ 2.18C



รูปที่ 2.18 ขั้นตอนการทำ Reduction step [9]

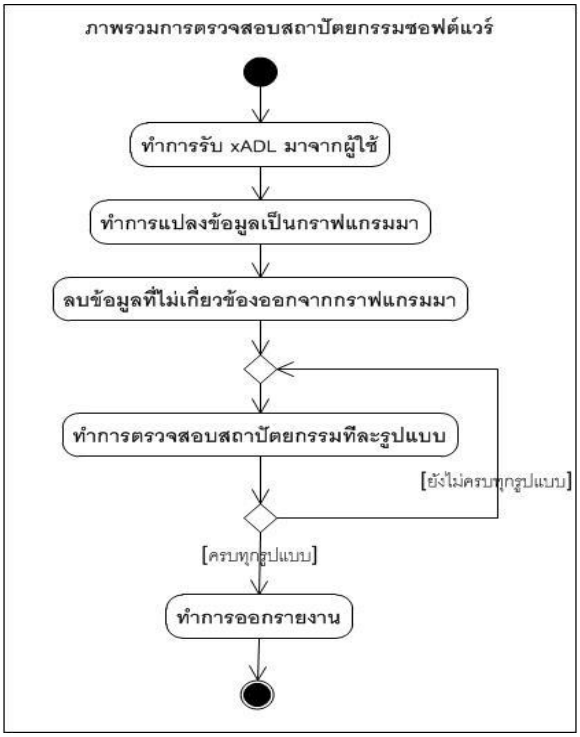
โดยสำหรับงานของผู้วิจัยจะเอาเทคนิคเดียวกันนี้เป็นส่วนหนึ่งมาสร้างกฎสำหรับการตรวจจับรูปแบบสถาปัตยกรรมแทน

บทที่ 3

การออกแบบกราฟแกรมมาเพื่อการตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์

ผู้วิจัยได้อาศัยแนวคิดพื้นฐานจากทฤษฎีกราฟแกรมมาสำหรับไวยากรณ์พึ่งบริบทแบบเอ็นซีอี [10] นี้มาพัฒนาเพิ่มเติมให้สามารถทำการตรวจสอบสถาปัตยกรรมซอฟต์แวร์จากกฎได้ และงานวิจัยนี้จะนำทฤษฎี กราฟแกรมมามาประยุกต์ใช้เพื่อเพิ่มความสามารถทำให้เห็นรายละเอียดของโครงสร้างของระบบได้มากยิ่งขึ้น

โดยผู้วิจัยได้ทำการออกแบบกราฟที่มีคุณลักษณะของไวยากรณ์พึ่งบริบท ขึ้นสำหรับตรวจสอบสถาปัตยกรรม เรียกว่า กราฟแกรมมาสำหรับตรวจสอบสถาปัตยกรรม (Architectural Style Based Graph Grammar) โดยจะใช้ทฤษฎีกราฟแกรมมาสำหรับไวยากรณ์พึ่งบริบทแบบเอ็นซีอี [10] มาประยุกต์ โดยขั้นตอนการตรวจสอบสถาปัตยกรรมทีละรูปแบบจะใช้วิธีการลดรูป เพื่อให้เหลือรูปสถาปัตยกรรมตั้งต้นของแต่ละรูปแบบสถาปัตยกรรม เนื่องจากกราฟแกรมมาที่นำเข้ามาจะยังไม่ทราบว่ามีส่วนประกอบส่วนใดที่มีคุณสมบัติสถาปัตยกรรมอยู่ ซึ่งสามารถแสดงภาพรวมของการตรวจสอบสถาปัตยกรรมดังแผนภาพกิจกรรมรูปที่ 3.1



รูปที่ 3.1 ภาพรวมการตรวจสอบสถาปัตยกรรมซอฟต์แวร์

จากรูปที่ 3.1 แสดงภาพรวมของการตรวจสอบสถาปัตยกรรมโดยจะรับข้อมูลเอกซ์เอดีแอลมาจากผู้ใช้ หลังจากนั้นทำการแปลงข้อมูลให้อยู่ในรูปของกราฟแกรมมา และทำการลบข้อมูลที่ไม่เกี่ยวข้องออกจากกราฟแกรมมาแล้วทำการตรวจสอบสถาปัตยกรรมทีละรูปแบบโดยใช้วิธีการลดรูปเพื่อให้ได้รูปแบบสถาปัตยกรรมที่กำลังตรวจสอบโดยถ้าไม่สามารถใช้กฎที่สร้างขึ้นมาเพื่อลดรูปให้ได้รูปสถาปัตยกรรมที่กำลังตรวจสอบได้ ก็จะไม่ถือว่ามีสถาปัตยกรรมนั้นอยู่ในรูปแบบสถาปัตยกรรมหลัก

3.1 การกำหนดนิยามกราฟแกรมมาสำหรับตรวจสอบสถาปัตยกรรม

คำนิยามที่ 1 นิยามคำศัพท์

กำหนดให้

- C แทนเงื่อนไขก่อนการใช้กฎ ซึ่งถ้ารูปแบบสถาปัตยกรรมไม่เข้าเงื่อนไข จะไม่สามารถใช้กฎนั้นได้
- Non-terminal symbol แทนลักษณะของสิ่งของที่นับไม่ได้ โดยใช้อักษรตัวพิมพ์ใหญ่เป็นหลัก
- Terminal symbol แทนลักษณะของสิ่งของที่นับได้ โดยใช้อักษรตัวพิมพ์เล็กเป็นหลัก
- P แทนเซตของกฎที่ใช้ในการตรวจจับสถาปัตยกรรม
- lhs(P) แทนด้านซ้ายของกฎ P เช่น P แทน L ::= R ถ้า lhs(P) จะได้ผลลัพธ์เป็น L
- rhs(P) แทนด้านขวาของกฎ P เช่น P แทน L ::= R ถ้า rhs(P) จะได้ผลลัพธ์เป็น R
- บัพ แทนส่วนประกอบต่างๆ ที่มีอยู่ในสถาปัตยกรรม

คำนิยามที่ 2 รูปแบบของสถาปัตยกรรม

กำหนดให้ AS แทนเซตของชื่อรูปแบบสถาปัตยกรรมเป้าหมายที่ต้องการตรวจจับ ชื่อรูปแบบสถาปัตยกรรมเหล่านี้จะใช้เป็นสัญลักษณ์นอนเทอร์มินัลแรกเริ่ม (start non-terminal symbol) ในไวยากรณ์ที่ผู้วิจัยใช้ในการตรวจจับรูปแบบ ผู้วิจัยสามารถเขียนเซต AS = {REPOSITORY, PIPEANDFILTER, EVENTBASE, LAYER} โดย REPOSITORY, PIPEANDFILTER, EVENTBASE, LAYER เป็นชื่อรูปแบบสถาปัตยกรรมเป้าหมายที่ต้องการตรวจจับ

คำนิยามที่ 3 รูปแบบของกราฟ

กำหนดให้กราฟ เขียนแทนด้วยกฎการผลิต G = (V, E) โดยที่ V แทนชุดของบัพ และ E แทน ชุดของด้าน

คำนิยามที่ 4 สัญลักษณ์ที่ใช้ในสถาปัตยกรรม

กำหนดให้ AE แทนเซตของสัญลักษณ์เทอร์มินัลซึ่งเป็นสัญลักษณ์ที่ใช้ในสถาปัตยกรรม (software architectural model) ในไวยากรณ์ที่ผู้วิจัยใช้ในการตรวจจับรูปแบบ ผู้วิจัยสามารถเขียนเซต AE = { compo, intf , intf_in , intf_out} โดย compo เป็นคอมโพเนนท์ที่ใช้ในสถาปัตยกรรม intf เป็นอินเตอร์เฟสที่ไม่ระบุทิศทางที่ใช้ในสถาปัตยกรรม intf_in และ intf_out เป็นอินเตอร์เฟสที่ระบุทิศทางเข้าและออกที่ใช้ในสถาปัตยกรรม

คำนิยามที่ 5 กราฟแกรมมาสำหรับตรวจสอบรูปแบบสถาปัตยกรรม (Architectural Style Based Graph Grammar)

กราฟแกรมมาสำหรับตรวจสอบสถาปัตยกรรม ASGG เขียนแทนด้วย

ASGG = (Ψ , Σ, Γ, $\mathbf{P}$)

โดยที่

- $\Psi \in$ AS
- $\Sigma = \Sigma_N \cup \Sigma_T$ และ $\Sigma_N \cap \Sigma_T = \phi$
- Σ_N เป็นเซตของสัญลักษณ์ของนอนเทอร์มินัล (non-terminal node)
- Σ_T เป็นเซตของสัญลักษณ์ของเทอร์มินัล (terminal node) ซึ่งได้มีการกำหนดไว้แทนด้วยสัญลักษณ์ AE
- Γ แทนชุดของด้านในเซต $\Sigma \times \Sigma$
- **P** แทนด้วยเซตของกฎการผลิต

คำนิยามที่ 6 กฎการผลิต (Production Rule)

กำหนดให้ P แทนเซตของกฎการผลิตโดยแต่ละกฎการผลิตเขียนอยู่ในรูปของ

L ::= R , C

โดยที่

- L แทนด้วยสัญลักษณ์เริ่มต้นที่แทนสถาปัตยกรรมที่ผู้วิจัยต้องการค้นหาΨ หรือเขียนแทนด้วยกราฟ ซึ่งสามารถเขียนแทนด้วย lhs(P)
- R แทนด้วยกราฟ ซึ่งสามารถเขียนแทนด้วย rhs(P)
- C แทนคุณสมบัติเบื้องต้นก่อนที่จะสามารถใช้กฎการผลิตได้ซึ่งถ้าไม่มีการประกาศ C ในกฎการผลิตแสดงว่ากฎการผลิตนั้นไม่ต้องมีคุณสมบัติเบื้องต้น

คำนิยามที่ 7 การคืนรูป (Derivation)

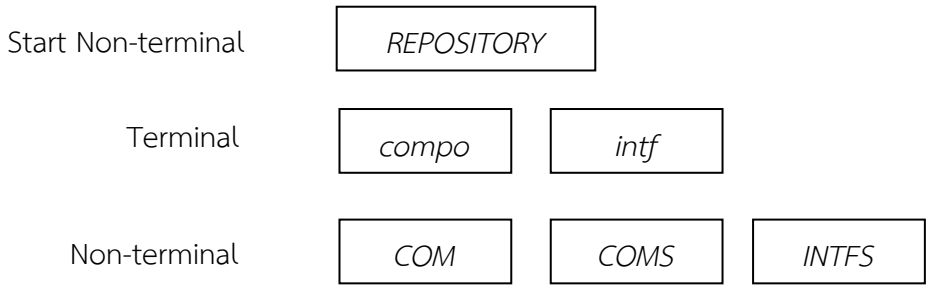
กำหนดให้วิธีการสำหรับการเปลี่ยนแปลงรูปกราฟจากกราฟเริ่มต้น Ψ ไปเป็นกราฟใหม่ โดยกราฟใหม่ถูกเขียนแทนด้วย lhs(P) ซึ่งจะถูกเรียกว่า การเขียนกราฟใหม่โดยใช้ด้านซ้ายของกฎการผลิต P เป็นหลัก โดยสามารถเขียนแทนสัญลักษณ์เป็น $G \underset{P}{\Rightarrow} G'$ โดย P เป็นกฎการผลิตจากซ้ายไปขวา ซึ่งจากสัญลักษณ์ข้างต้นจะหมายความว่า สามารถทำการเขียนกราฟ G’ แทนกราฟ G โดยอ้างอิงกฎ P

คำนิยามที่ 8 การลดรูป (Reduction)

กำหนดให้วิธีการสำหรับการเปลี่ยนแปลงรูปกราฟจากกราฟ G' ไปเป็นกราฟเริ่มต้น Ψ หรือกราฟ G โดยขั้นตอนนี้จะถูกเรียกว่า การเขียนกราฟใหม่โดยกราฟใหม่ถูกเขียนแทนด้วย rhs(P) เป็นหลัก สามารถเขียนแทนสัญลักษณ์เป็น $G' \underset{P}{\Rightarrow} G$ โดยกฎ P ถ้ามีเครื่องหมาย ´ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายแทน

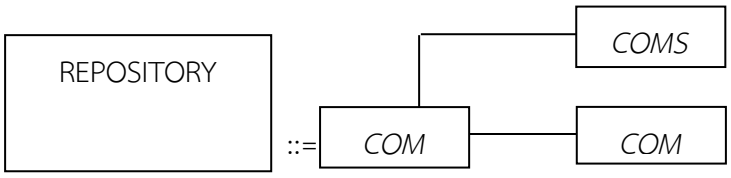
คำนิยามที่ 9 การตรวจสอบสถาปัตยกรรมในรูปแบบรวมศูนย์กลาง

ผู้วิจัยได้สร้างกราฟแกรมมาสำหรับตรวจสอบสถาปัตยกรรมในรูปแบบรวมศูนย์กลาง โดยมีพื้นฐานมาจากนิยามข้างต้น โดยมีชุดของสัญลักษณ์ของนอนเทอร์มินัล และ ชุดของสัญลักษณ์ของเทอร์มินัล ซึ่งแสดงดังรูปที่ 3.2



รูปที่ 3.2 สัญลักษณ์เริ่มต้น ชุดของสัญลักษณ์ของนอนเทอร์มินัล (non-terminal) และ ชุดของสัญลักษณ์ของเทอร์มินัล (terminal node)

จากรูปที่ 3.2 จะประกอบประกอบไปด้วยบัพที่มีลักษณะเทอร์มินัลและนอนเทอร์มินัล ซึ่งประกอบด้วย REPOSITORY แทนรูปแบบสถาปัตยกรรมซอฟต์แวร์ในรูปแบบรวมศูนย์กลางที่กำลังค้นหา COM แทนสัญลักษณ์ที่มีไว้สำหรับรวม compo และ intf หรือ INTFS เข้าไว้ด้วยกัน COMS แทนสัญลักษณ์ที่มีไว้สำหรับรวม COM หลายแห่งเข้าด้วยกัน และ INTFS แทนสัญลักษณ์ที่มีไว้สำหรับรวม intf หลายแห่งเข้าด้วยกัน โดยผู้วิจัยได้กำหนดชุดของกฎการผลิต P1-P7 และกฎการผลิต P1´- P7´ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้าย โดยกฎเหล่านี้มีไว้สำหรับการตรวจสอบสถาปัตยกรรมซอฟต์แวร์ในรูปแบบรวมศูนย์กลาง โดยแต่ละกฎการผลิตเขียนอยู่ในรูปของ L ::= R,C ไว้สำหรับตรวจสอบสถาปัตยกรรมดังรูปที่ 3.3

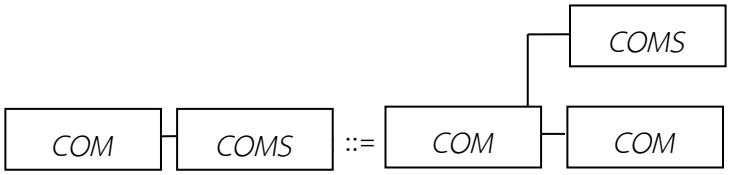


(P1)

c= { number of COM= 1}

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพที่เป็นบัพที่แทนคุณลักษณะสถาปัตยกรรมรูปศูนย์กลางจะประกอบไปด้วยบัพศูนย์กลาง COM ที่เชื่อมต่ออยู่กับบัพ COMS และบัพ COM โดยในทางกลับกันกฎข้อนี้ก็จะ

บ่งบอกว่าบัพศูนย์กลาง COM ที่เชื่อมต่ออยู่กับบัพ COMS และบัพ COM สามารถแทนคุณลักษณะสถาปัตยกรรมรวมศูนย์กลางได้



(P2)

c= { number of COM > 1}

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพ COM ที่เชื่อมต่ออยู่กับบัพ COMS สามารถแยกบัพ COMS ออกมาเป็นบัพศูนย์กลาง COM เชื่อมอยู่กับบัพ COMS และบัพ COM โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าสามารถใช้บัพ COMS รวมบัพ COM ที่เชื่อมอยู่กับบัพศูนย์กลางไปเป็นบัพ COMS ได้

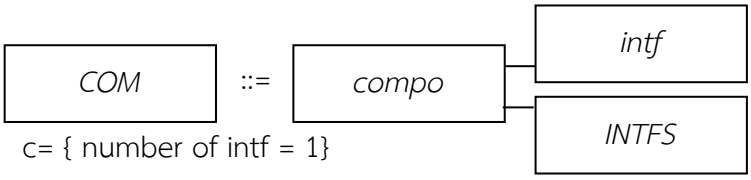
(P3)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าสามารถรวมบัพ COMS ไว้ในบัพ COM ได้ โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าสามารถสร้างบัพ COMS ขึ้นมาจากบัพ COM ไว้สำหรับทำการรวมบัพ COM ไปอยู่ในบัพ COMS ได้

(P4)

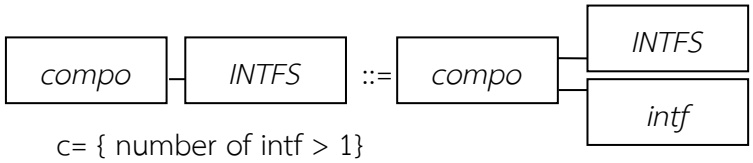
กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพ COM เชื่อมต่อกัน สามารถแยกออกมาเป็นบัพ COM เชื่อมต่อกับบัพ intf ที่เชื่อมต่ออยู่กับบัพ compo

โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าสามารถรวมบัพ COM ที่ต่อกับบัพ compo โดยผ่าน Intf เป็นบัพ COM เชื่อมต่อกับบัพ COM ได้



(P5)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพ COM จะประกอบไปด้วยบัพ compo intf และ INTFS ที่เชื่อมต่อกัน โดยในทางกลับกันกฎข้อนี้สามารถรวมบัพ compo INTFS และ intf ที่เชื่อมต่อกันเป็นบัพ COM



(P6)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพ compo ที่เชื่อมต่ออยู่กับบัพ INTFS สามารถแยกบัพ INTFS ออกมาเป็นบัพ ศูนย์กลาง compo เชื่อมอยู่กับบัพ intf และบัพ INTFS โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าสามารถใช้บัพ INTFS รวมบัพ intf ที่เชื่อมอยู่กับบัพศูนย์กลางไปเป็นบัพ INTFS ได้

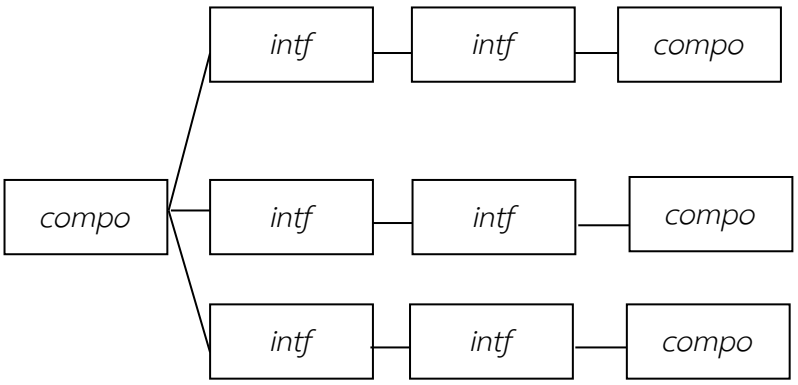
(P7)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพ compo ที่เชื่อมต่อกับบัพ INTFS สามารถรวมกันเป็นบัพ compo ได้ โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าสามารถสร้างบัพ INTFS ขึ้นมาจากบัพ compo ไว้สำหรับทำการรวมบัพ intf ไปอยู่ในบัพ INTFS ได้

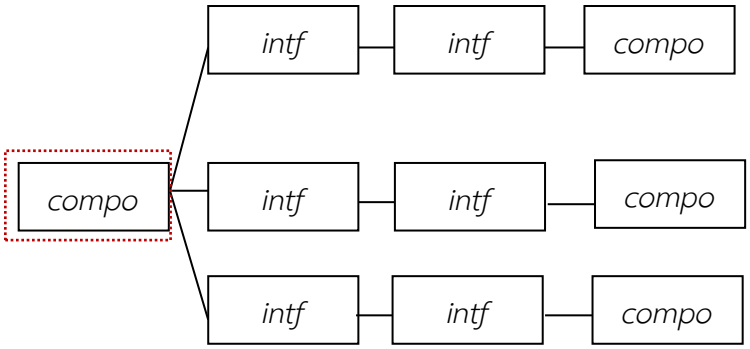
รูปที่ 3.3 กฎการผลิต P1-P7 โดยเขียนอยู่ในรูปของ L ::= R,C เพื่อตรวจสอบสถาปัตยกรรมในรูปแบบรวมศูนย์กลาง

จากรูปที่ 3.3 แสดงถึงกฎการผลิตสำหรับการตรวจสอบสถาปัตยกรรมในรูปแบบรวมศูนย์กลางที่ใช้กฎการผลิตที่มีอยู่นี้ในการลดรูปเพื่อให้เหลือรูปเริ่มต้นซึ่งในที่นี้จะใช้กฎข้อ P1 ในการลดรูปโดยจะแสดงในตัวอย่างต่อไปนี้

จากรูปที่ 3.4 เป็นตัวอย่างของรูปกราฟแกรมมาเริ่มต้นสำหรับการใช้กฎการลดรูป ซึ่งจะใช้กฎการผลิต P1´- P7´ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายในการลดรูป เพื่อตรวจสอบรูปว่ามีคุณลักษณะของสถาปัตยกรรมในรูปแบบรวมศูนย์กลางอยู่หรือไม่ โดยขั้นตอนแรกผู้วิจัยทำการเลือกบัพเริ่มต้น โดยดูจากคุณสมบัติของสถาปัตยกรรมในรูปแบบรวมศูนย์กลางเป็นหลัก ซึ่งลักษณะของสถาปัตยกรรมในรูปแบบรวมศูนย์กลางจะมีการเชื่อมต่อจากบัพรอบข้างไปบัพศูนย์กลางบัพใดบัพหนึ่ง [6] ซึ่งหลักการเลือกบัพที่เป็นบัพเริ่มต้นโดยดูจากบัพที่มีการเชื่อมต่อมามากที่สุด ซึ่งถ้ามีบัพที่เข้าเกณฑ์มากกว่า 1 บัพให้ใช้ตัวที่ตรวจพบก่อนเป็นหลัก จากการเลือกบัพเริ่มต้นผู้วิจัยจะเลือกบัพเริ่มต้นดังรูปที่ 3.5

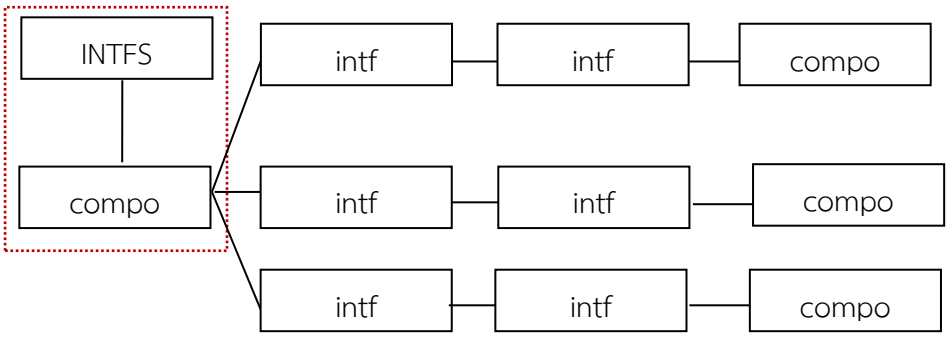


รูปที่ 3.4 รูปของกราฟเริ่มต้นที่จะทำการลดรูป



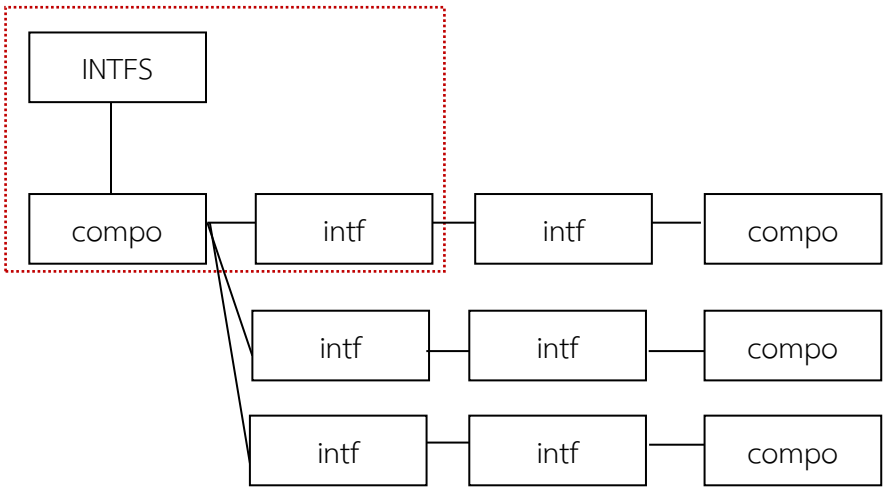
รูปที่ 3.5 รูปของกราฟเริ่มต้นหลังจากทำการเลือกบัพเริ่มต้นแล้ว

หลังจากทำการเลือกบัพเริ่มต้นแล้ว ใช้กฎข้อ P7’ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P7 ในการเปลี่ยนรูป ซึ่งจะได้ผลดังรูปที่ 3.6

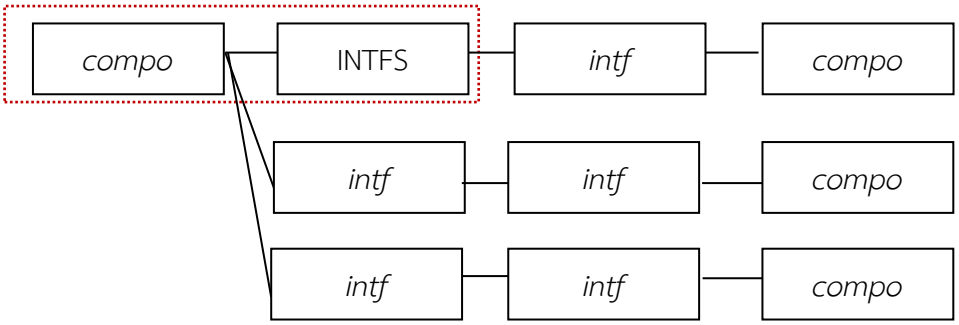


รูปที่ 3.6 รูปของกราฟหลังจากทำการใช้กฎ P7’

ผลลัพธ์ในรูปที่ 3.7 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P6' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P6 แสดงในรูปที่ 3.8

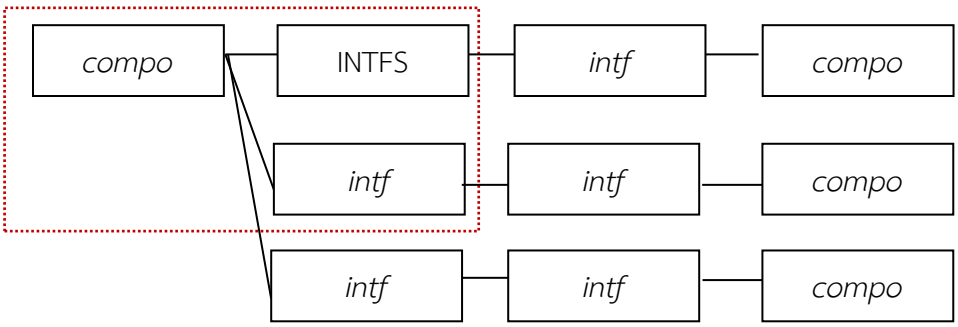


รูปที่ 3.7 รูปของกราฟก่อนทำการใช้กฎ P6’

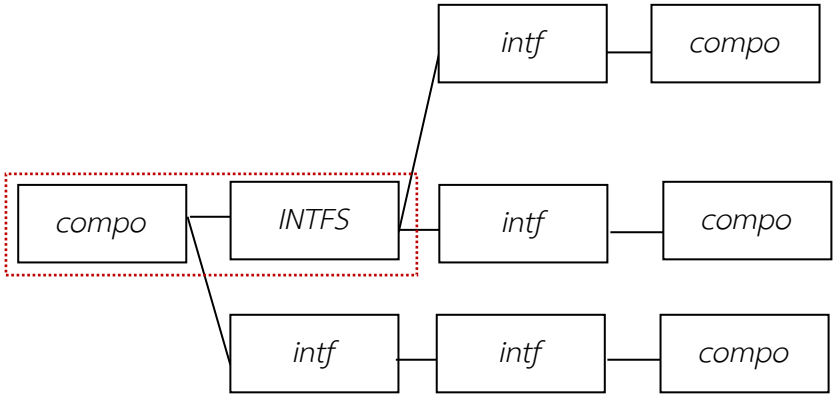


รูปที่ 3.8 รูปของกราฟหลังทำการใช้กฎ P6’

ผลลัพธ์ในรูปที่ 3.9 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P6' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P6 แสดงในรูปที่ 3.10

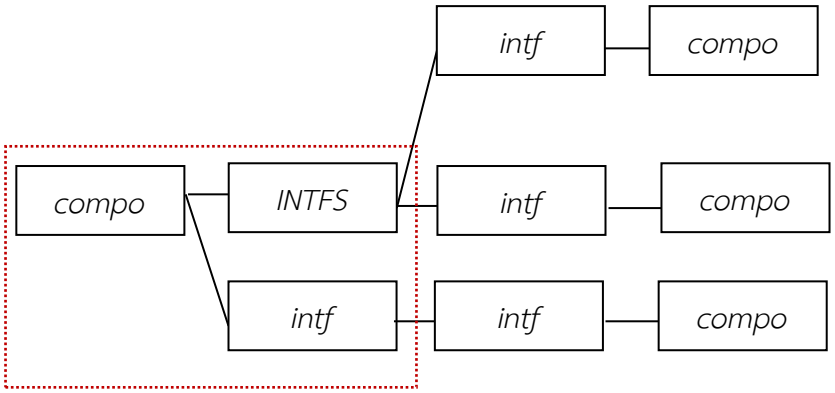


รูปที่ 3.9 รูปของกราฟก่อนทำการใช้กฎ P6’ รอบที่ 2

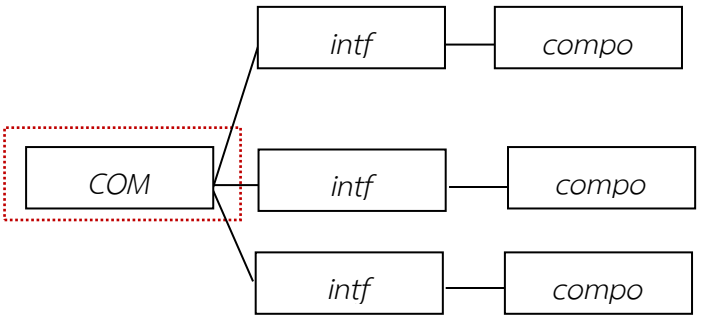


รูปที่ 3.10 รูปของกราฟหลังทำการใช้กฎ P6’ รอบที่ 2

ผลลัพธ์ในรูปที่ 3.11 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P5' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P5 แสดงในรูปที่ 3.12

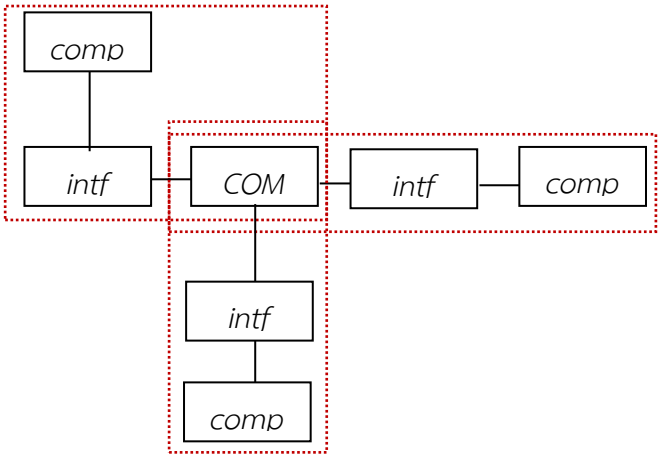


รูปที่ 3.11 รูปของกราฟก่อนทำการใช้กฎ P5’

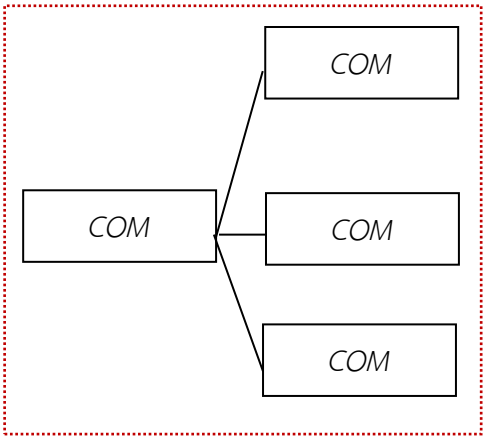


รูปที่ 3.12 รูปของกราฟหลังทำการใช้กฎ P5’

ผลลัพธ์ในรูปที่ 3.13 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P4' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P4 แสดงในรูปที่ 3.14

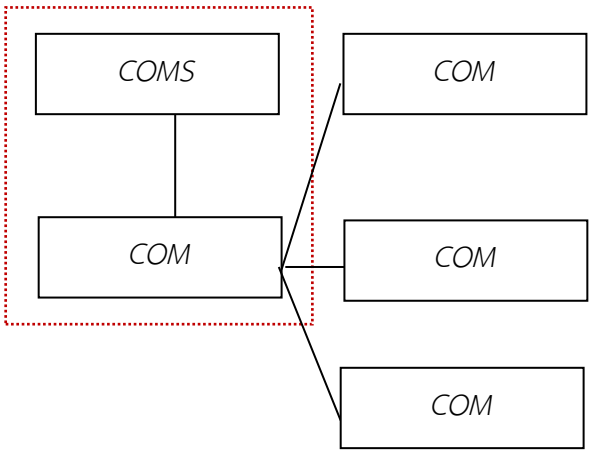


รูปที่ 3.13 รูปของกราฟก่อนทำการใช้กฎ P4’



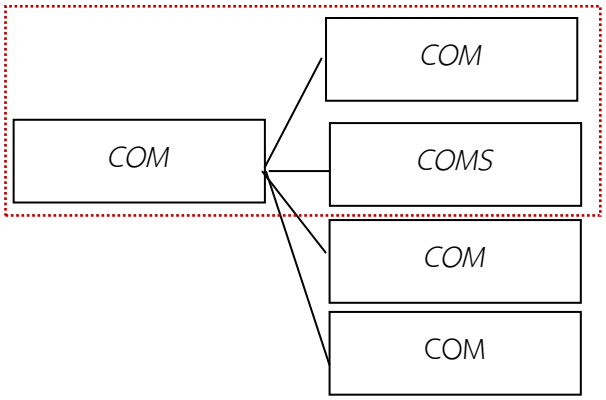
รูปที่ 3.14 รูปของกราฟหลังทำการใช้กฎ P4’

ผลลัพธ์ในรูปที่ 3.14 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P3' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P3 แสดงในรูปที่ 3.15

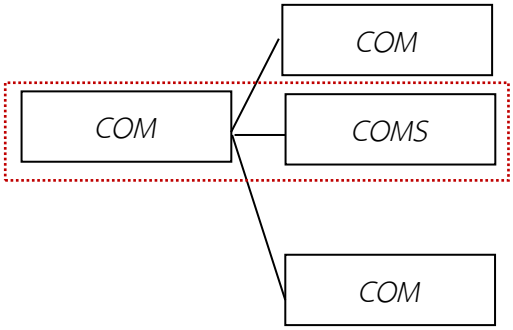


รูปที่ 3.15 รูปของกราฟหลังทำการใช้กฎ P3’

ผลลัพธ์ในรูปที่ 3.16 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P2' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P2 แสดงในรูปที่ 3.17

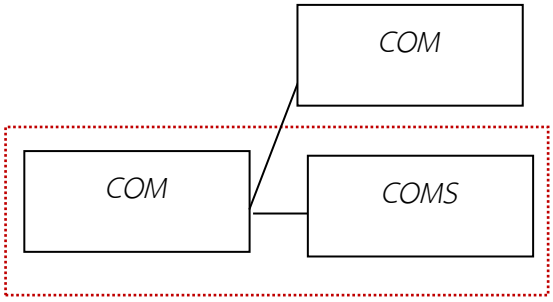


รูปที่ 3.16 รูปของกราฟก่อนทำการใช้กฎ P2’



รูปที่ 3.17 รูปของกราฟหลังทำการใช้กฎ P2’

ผลลัพธ์ในรูปที่ 3.17 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P2' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P2 แสดงในรูปที่ 3.18



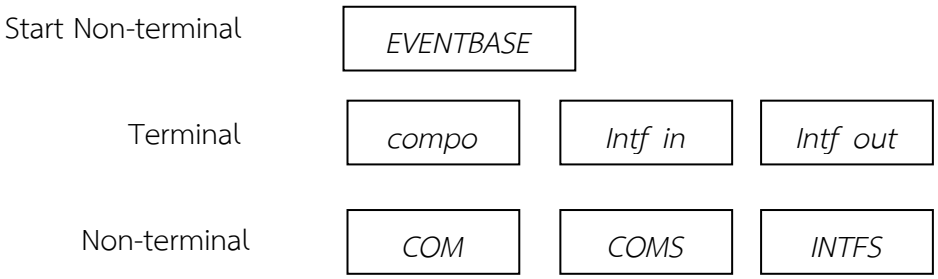
รูปที่ 3.18 รูปของกราฟหลังทำการใช้กฎ P2’ อีกครั้ง

จากรูปที่ 3.18 จะเห็นได้ว่าผู้วิจัยสามารถใช้กฎ P1’ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P1 ในการสร้างจุดเริ่มต้น (Start Symbol) ซึ่งหมายความว่าในกราฟหลักสามารถค้นพบคุณสมบัติของสถาปัตยกรรมในรูปแบบรวมศูนย์กลางเพราะสามารถใช้กฎของการลดรูปลดรูปจนได้รูปแบบสถาปัตยกรรมเริ่มต้นได้โดยจะได้ผลลัพธ์ดังรูปที่ 3.19

รูปที่ 3.19 รูปของกราฟที่ได้จุดเริ่มต้น (Start Symbol) ที่มีชื่อว่า REPOSITORY

คำนิยามที่ 10 **การตรวจสอบสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้น**

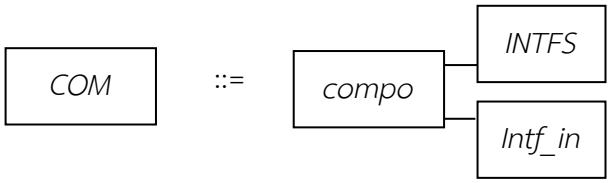
ผู้วิจัยได้สร้างกราฟแกรมมาสำหรับตรวจสอบสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้น โดยมีพื้นฐานมาจากนิยามข้างต้น โดยมีชุดของสัญลักษณ์ของนอนเทอร์มินัล และ ชุดของสัญลักษณ์ของเทอร์มินัล ซึ่งแสดงดังรูปที่ 3. 20



รูปที่ 3. 20 สัญลักษณ์เริ่มต้น ชุดของสัญลักษณ์ของนอนเทอร์มินัล (non-terminal) และ ชุดของสัญลักษณ์ของเทอร์มินัล (terminal node)

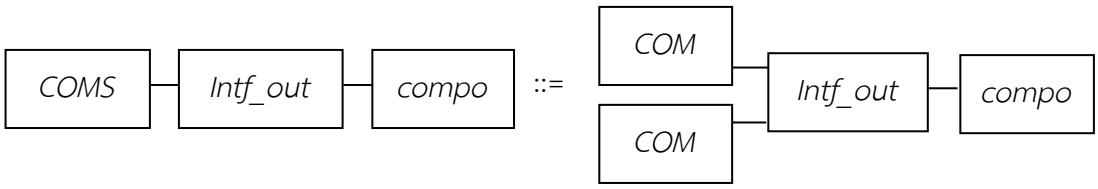
จากรูปที่ 3. **20** จะประกอบประกอบไปด้วยบัพที่มีลักษณะเทอร์มินัลและนอนเทอร์มินัล ซึ่งประกอบด้วย EVENTBASE แทนรูปแบบสถาปัตยกรรมซอฟต์แวร์ในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้นที่กำลังค้นหา COM แทนสัญลักษณ์ที่มีไว้สำหรับรวม compo และ intf หรือ INTFS เข้าไว้ด้วยกัน COMS แทนสัญลักษณ์ที่มีไว้สำหรับรวม COM หลายแห่งเข้าด้วยกัน และ INTFS แทนสัญลักษณ์ที่มีไว้สำหรับรวม intf หลายแห่งเข้าด้วยกัน โดยผู้วิจัยได้กำหนดชุดของกฎการผลิต P8-P11 และ กฎการผลิต P8´- P11´ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวา

มาทางด้านซ้าย โดยกฎเหล่านี้มีไว้สำหรับการตรวจสอบสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้น โดยแต่ละกฎการผลิตเขียนอยู่ในรูปของ L ::= R,C ไว้สำหรับตรวจสอบสถาปัตยกรรมดังรูปที่ 3.21



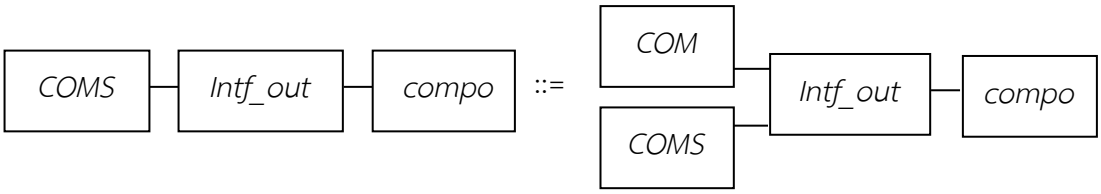
(P8)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพ COM จะประกอบไปด้วยบัพ compo intf_in และ INTFS ที่เชื่อมต่อกัน โดยในทางกลับกันกฎข้อนี้สามารถรวมบัพ compo INTFS และ intf_in ที่เชื่อมต่อกันเป็นบัพ COM ได้



(P9)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพ COMS ที่เชื่อมต่ออยู่กับ Intf_out และบัพ compo สามารถแยกบัพ COMS ออกมาเป็นบัพ COM สองบัพเชื่อมต่ออยู่กับบัพ Intf_out และบัพ compo ได้ โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าสามารถรวมบัพ COM สองบัพที่เชื่อมต่ออยู่กับบัพ Intf_out เป็นบัพ COMS หนึ่งบัพได้



(P10)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพ COMS ที่เชื่อมต่ออยู่กับ Intf_out และบัพ compo สามารถแยกบัพ COMS ออกมาเป็นบัพ COMS และบัพ COM สองบัพ เชื่อมต่ออยู่กับบัพ Intf_out และบัพ compo ได้ โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าสามารถรวมบัพ COM และบัพ COMS ที่เชื่อมต่ออยู่กับบัพ Intf_out เป็นบัพ COMS หนึ่งบัพได้

(P11)

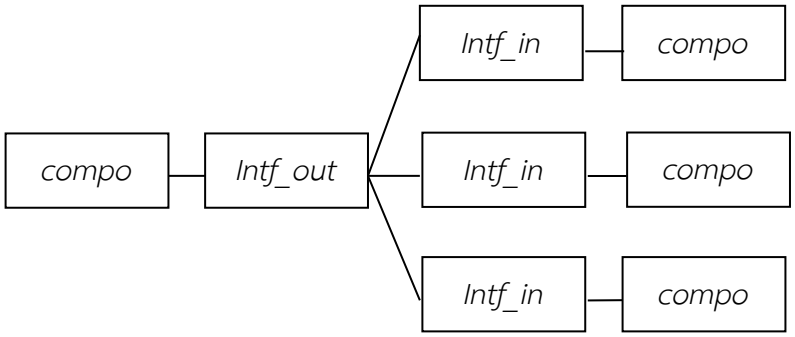
กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพที่เป็นบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้นจะประกอบไปด้วยบัพ compo ที่เชื่อมต่อกับอินเตอร์เฟส Intf_out และอินเตอร์เฟสเดียวกันนี้ยังเชื่อมต่ออยู่กับ COMS ด้วย โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าบัพ compo ที่เชื่อมต่อกับอินเตอร์เฟส Intf_out และอินเตอร์เฟสเดียวกันนี้ยังเชื่อมต่ออยู่กับ COMS สามารถแทนคุณลักษณะสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้น

รูปที่ 3.21 กฎการผลิต P8-P11 โดยเขียนอยู่ในรูปของ L ::= R,C เพื่อตรวจสอบสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้น

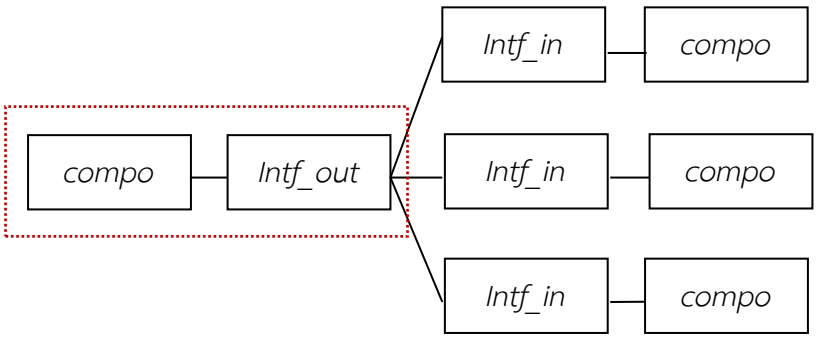
จากรูปที่ 3.21 แสดงถึงกฎการผลิตสำหรับการตรวจสอบสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้นที่ใช้กฎการผลิตที่มีอยู่นี้ในการลดรูปเพื่อให้เหลือรูปเริ่มต้น ซึ่งในที่นี้จะใช้กฎข้อ P11 ในการลดรูปโดยจะแสดงในตัวอย่างต่อไปนี้

จากรูปที่ 3.22 เป็นตัวอย่างของรูปเริ่มต้นสำหรับการใช้กฎการลดรูปซึ่งจะใช้กฎการผลิต P8´- P11´ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายในการลดรูป

เพื่อตรวจสอบรูปว่ามีคุณลักษณะของสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้นอยู่หรือไม่ โดยขั้นตอนแรกผู้วิจัยทำการเลือกบัพเริ่มต้น โดยดูจากคุณสมบัติของสถาปัตยกรรมสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้น ซึ่งมีลักษณะที่สามารถส่งข้อมูลจากช่องทางหนึ่งๆ ไปให้กับบัพปลายทางได้หลายบัพ [6] เป็นหลัก ซึ่งถ้ามีบัพที่เข้าเกณฑ์มากกว่า 1 บัพให้ใช้ตัวที่ตรวจพบก่อน ซึ่งจากการเลือกบัพเริ่มต้นผู้วิจัยจะเลือกบัพเริ่มต้นดังรูปที่ 3.23

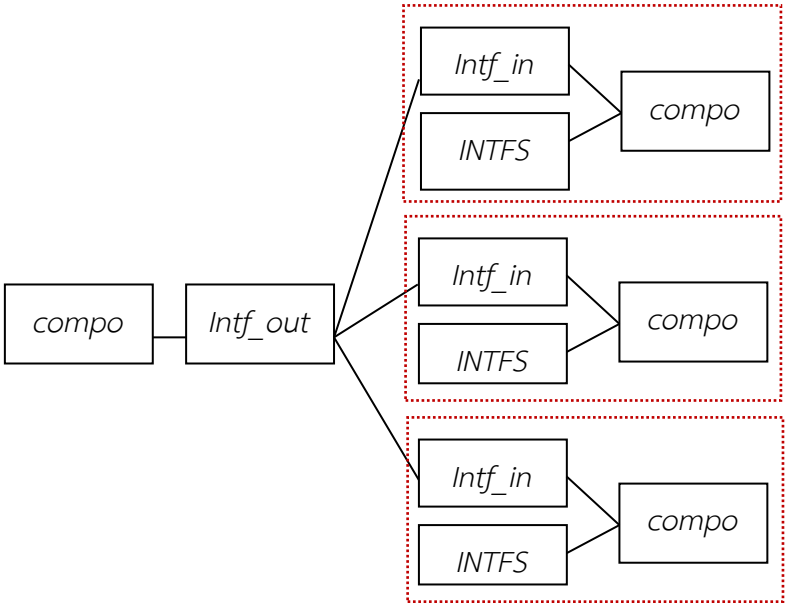


รูปที่ 3.22 รูปของกราฟเริ่มต้นที่จะทำการใช้กฎการลดรูป

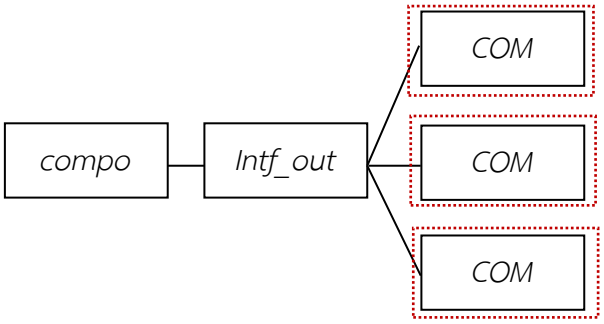


รูปที่ 3.23 รูปของกราฟเริ่มต้นหลังจากทำการเลือกบัพเริ่มต้นแล้ว

จากรูปที่ 3. 24 ผู้วิจัยสามารถใช้กฎข้อ P8’ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P8 เพื่อที่จะเปลี่ยนรูปร่างของกราฟ ซึ่งได้ผลลัพธ์ดังรูปที่ 3.25

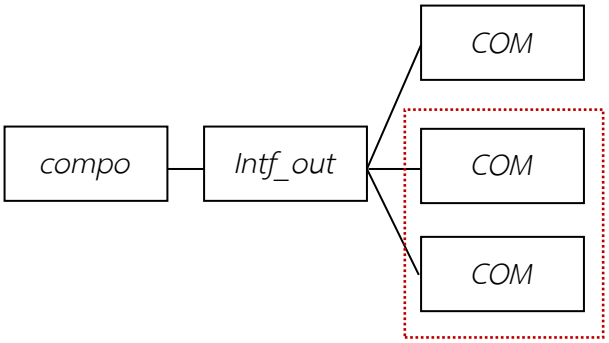


รูปที่ 3. 24 รูปของกราฟก่อนจากทำการใช้กฎ P8’

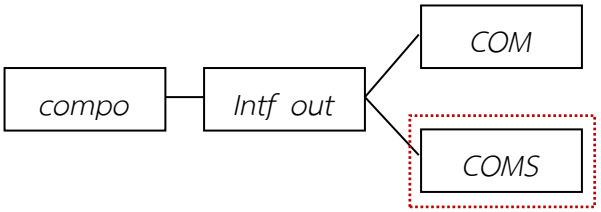


รูปที่ 3.25 รูปของกราฟหลังจากทำการใช้กฎ P8’

ผลลัพธ์ในรูปที่ 3.26 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P9' แสดงในรูปที่ 3.27

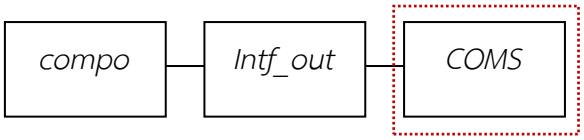


รูปที่ 3.26 รูปของกราฟก่อนทำการใช้กฎ P9’



รูปที่ 3.27 รูปของกราฟหลังทำการใช้กฎ P9’

ผลลัพธ์ในรูปที่ 3.27 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P10' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P10 แสดงในรูปที่ 3.28



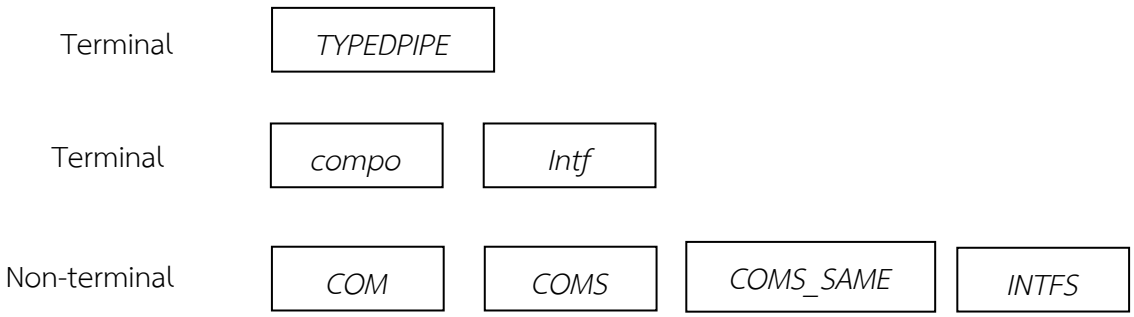
รูปที่ 3.28 รูปของกราฟหลังทำการใช้กฎ P10’

จากรูปที่ 3.28 จะเห็นได้ว่าผู้วิจัยสามารถใช้กฎ P11’ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P11 ในการสร้างจุดเริ่มต้น (Start Symbol) ซึ่งหมายความว่าในกราฟหลักสามารถค้นพบคุณสมบัติของสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้นเพราะสามารถใช้กฎของการลดรูปลดรูปจนได้รูปแบบสถาปัตยกรรมเริ่มต้นได้โดยจะได้ผลลัพธ์ดังรูปที่ 3.29

รูปที่ 3.29 รูปของกราฟหลังทำการใช้กฎ P11’

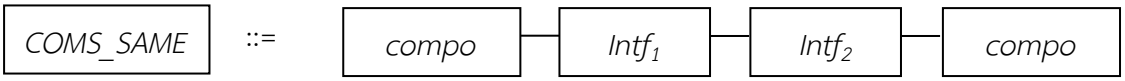
คำนิยามที่ 11 การตรวจสอบสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไทป์ไปป์

ผู้วิจัยได้สร้างกราฟแกรมมาสำหรับตรวจสอบสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ โดยมีพื้นฐานมาจากนิยามข้างต้น โดยมีชุดของสัญลักษณ์ของนอนเทอร์มินัล และ ชุดของสัญลักษณ์ของเทอร์มินัล ซึ่งแสดงดังรูปที่ 3.30



รูปที่ 3.30 สัญลักษณ์เริ่มต้น ชุดของสัญลักษณ์ของนอนเทอร์มินัล (non-terminal) และ ชุดของสัญลักษณ์ของเทอร์มินัล (terminal node)

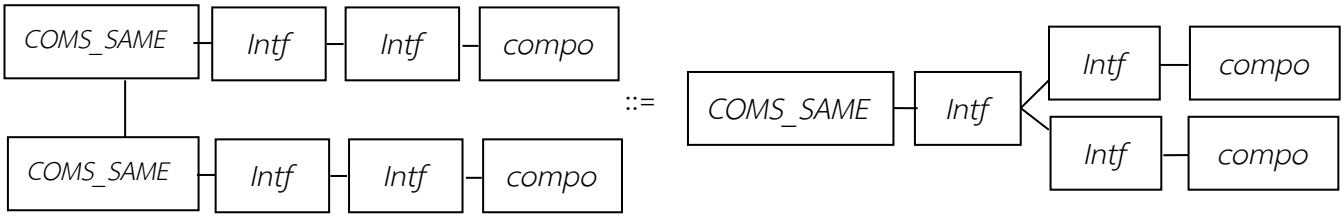
จากรูปที่ 3.30 จะประกอบประกอบไปด้วยบัพที่มีลักษณะเทอร์มินัลและนอนเทอร์มินัล ซึ่งประกอบด้วย *TYPEDPIPE* แทนรูปแบบสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ที่กำลังค้นหา COM แทนสัญลักษณ์ที่มีไว้สำหรับรวม compo และ intf หรือ INTFS เข้าไว้ด้วยกัน COMS แทนสัญลักษณ์ที่มีไว้สำหรับรวม COM หลายแห่งเข้าด้วยกัน COM_SAME แทนสัญลักษณ์ที่มีไว้สำหรับรวม compo และ intf ที่มีลักษณะเหมือนกันเข้าไว้ด้วยกัน และ INTFS แทนสัญลักษณ์ที่มีไว้สำหรับรวม intf หลายแห่งเข้าด้วยกัน โดยผู้วิจัยได้กำหนดชุดของกฎการผลิต P12-P18 และกฎการผลิต P12´- P18´ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้าย โดยกฎเหล่านี้มีไว้สำหรับการตรวจสอบสถาปัตยกรรมซอฟต์แวร์ในรูปแบบไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ โดยแต่ละกฎการผลิตเขียนอยู่ในรูปของ L ::= R,C ไว้สำหรับตรวจสอบสถาปัตยกรรมดังรูปที่ 3. 31



$C = \{ Type\ of\ INTF_1 = INTF_2 \}$

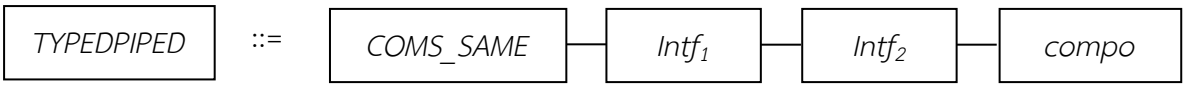
(P12)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพ COMS_SAME ประกอบไปด้วยบัพ compo ที่เชื่อมต่อกันสองบัพโดยผ่าน Intf ของแต่ละบัพเป็นตัวเชื่อมต่อและ Intf ทั้งสองบัพจะต้องเป็นชนิดเดียวกัน โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าบัพ compo ที่เชื่อมต่อกันสองบัพโดยผ่าน Intf ของแต่ละบัพเป็นตัวเชื่อมต่อและ Intf ทั้งสองบัพจะต้องเป็นชนิดเดียวกันนั้นสามารถแทนด้วยบัพ COMS_SAME



(P13)

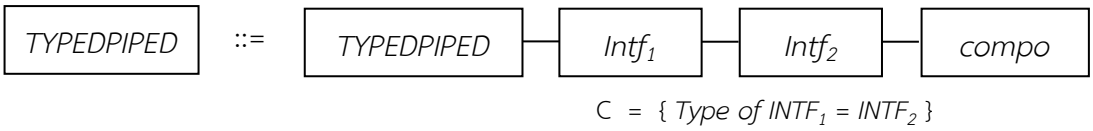
กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพ COMS_SAME สองบัพที่แยกกันโดยมีเส้นเชื่อมโดยแต่ละบัพก็ยังไปเชื่อมต่อกับบัพอื่นผ่านทาง intf สามารถทำการรวมบัพ COMS_SAME และ Intf ที่เชื่อมต่อกับบัพ COMS_SAME เป็นบัพเดียวได้ โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าบัพ COMS_SAME ที่เชื่อมต่อกับบัพ intf และบัพเดียวกันนี้ก็แยกช่องทางการเชื่อมต่อออกมาอีกเป็นสองช่องทาง สามารถทำการแยกบัพ COMS_SAME เป็นสองบัพที่แยกกันโดยมีเส้นเชื่อมถึงกันโดยแต่ละบัพก็ยังไปเชื่อมต่อกับบัพอื่นผ่านทาง intf ได้



C = { *Type of* $INTF_1 = INTF_2$ }

(P14)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ประกอบไปด้วยบัพ COMS_SAME ที่เชื่อมต่อบัพ compo โดยเชื่อมต่อผ่านบัพ Intf และ Intf ทั้งสองบัพจะต้องเป็นชนิดเดียวกัน โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าบัพ COMS_SAME ที่เชื่อมต่อบัพ compo โดยเชื่อมต่อผ่านบัพ Intf และ Intf ทั้งสองบัพจะต้องเป็นชนิดเดียวกันนั้นสามารถแทนด้วยบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไทป์ไปป์

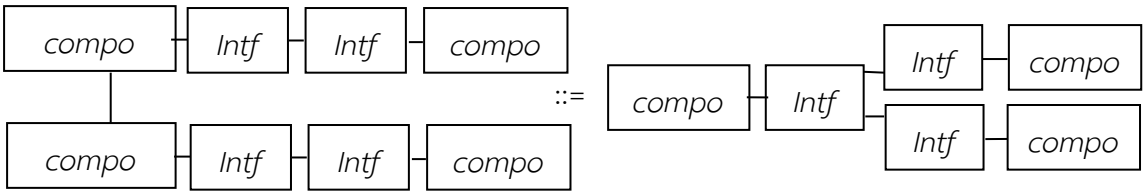


(P15)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ประกอบไปด้วยบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ที่เชื่อมต่อบัพ compo โดยเชื่อมต่อผ่านบัพ Intf และ Intf ทั้งสองบัพจะต้องเป็นชนิดเดียวกัน โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ที่เชื่อมต่อบัพ compo โดยเชื่อมต่อผ่านบัพ Intf และ Intf ทั้งสองบัพจะต้องเป็นชนิดเดียวกันนั้นสามารถแทนด้วยบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไทป์ไปป์

(P16)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ประกอบไปด้วยบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ที่เชื่อมต่อกันสองบัพ โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ที่เชื่อมต่อกันสองบัพสามารถแทนด้วยบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไทป์ไปป์



(P17)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพ compo สองบัพที่แยกกันโดยมีเส้นเชื่อมโดยแต่ละบัพก็ยังไปเชื่อมต่อกับบัพอื่นผ่านทาง intf สามารถทำการรวมบัพ compo และ Intf ที่เชื่อมต่อกับบัพ compo เป็นบัพเดียวได้ โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าบัพ compo ที่เชื่อมต่อกับบัพ intf และบัพเดียวกันนี้ก็แยกช่องทางการเชื่อมต่อออกมาอีกเป็นสองช่องทาง สามารถทำการแยกบัพ compo เป็นสองบัพที่แยกกันโดยมีเส้นเชื่อมถึงกันโดยแต่ละบัพก็ยังไปเชื่อมต่อกับบัพอื่นผ่านทาง intf ได้

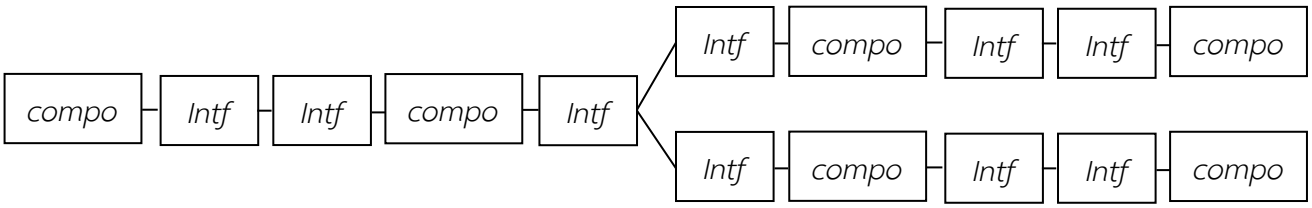
(P18)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพ COMS-SAME ประกอบไปด้วยบัพ COMS-SAME ที่เชื่อมต่อกันสองบัพ โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าบัพ COMS-SAME ที่เชื่อมต่อกันสองบัพสามารถแทนด้วยบัพ COMS-SAME

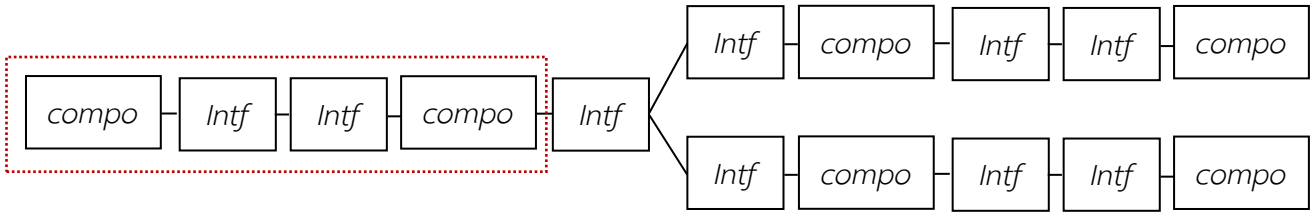
รูปที่ 3. 31 กฎการผลิต P12-P18 โดยเขียนอยู่ในรูปของ L ::= R,C เพื่อตรวจสอบสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไทป์ไปป์

จากรูปที่ 3. 31 แสดงถึงกฎการผลิตสำหรับการตรวจสอบสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ที่เกิดขึ้นที่ใช้กฎการผลิตที่มีอยู่นี้ในการลดรูปเพื่อให้เหลือรูปเริ่มต้นซึ่งในที่นี้จะใช้กฎข้อ P14 P15 หรือ P16 ในการลดรูปโดยจะแสดงในตัวอย่างต่อไปนี้

จากรูปที่ 3.32 เป็นตัวอย่างของรูปเริ่มต้นสำหรับการใช้กฎการลดรูปซึ่งจะใช้กฎการผลิต P12´- P18´ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายในการลดรูป เพื่อตรวจสอบรูปว่ามีคุณลักษณะของ สถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไทป์ไปป์อยู่หรือไม่ โดยขั้นตอนแรกผู้วิจัยทำการเลือกบัพเริ่มต้น โดยดูจากคุณสมบัติของสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไทป์ไปป์เป็นหลักซึ่งจะมีลักษณะชนิดของช่องทางเชื่อมต่อแบบเดียวกัน ซึ่งจะเลือกคอมโพเนนท์ที่มีอินเตอร์เฟสเดียวเป็นหลักก่อน ซึ่งถ้ามีบัพที่เข้าเกณมากกว่า 1 บัพให้ใช้ตัวที่ตรวจพบก่อน โดยจากการเลือกบัพเริ่มต้นผู้วิจัยจะเลือกบัพเริ่มต้นดังรูปที่ 3.33

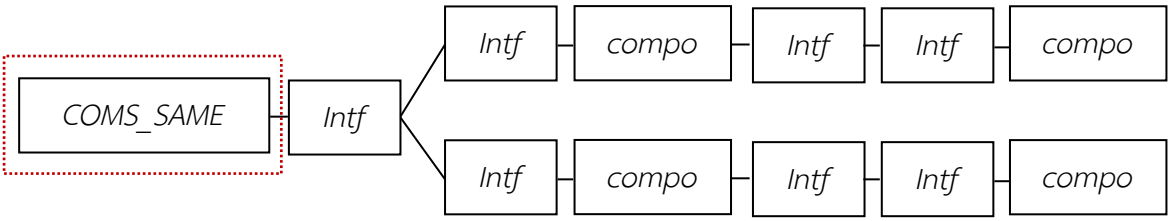


รูปที่ 3.32 รูปของกราฟเริ่มต้นที่จะทำการใช้กฎการลดรูป



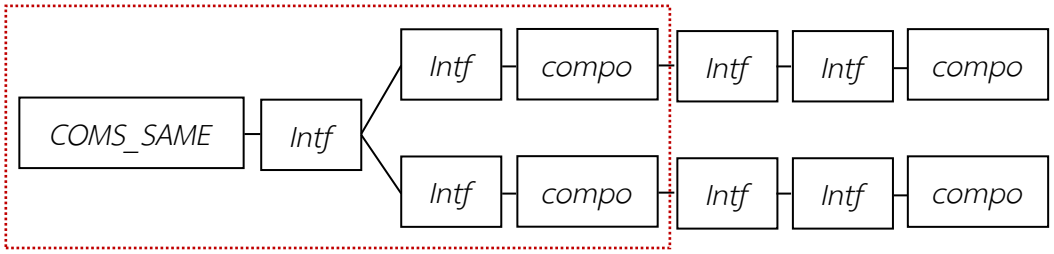
รูปที่ 3.33 รูปของกราฟเริ่มต้นหลังจากทำการใช้กฎการลดรูป

ผู้วิจัยสามารถใช้กฎข้อ P12’ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P12 เพื่อที่จะเปลี่ยนรูปร่างของกราฟ ซึ่งได้ผลลัพธ์ดังรูปที่ 3.34

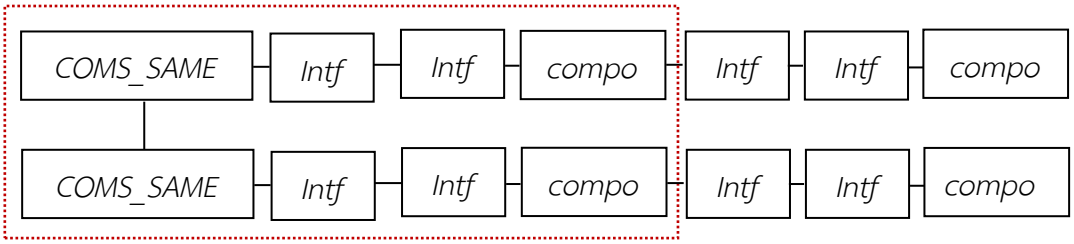


รูปที่ 3.34 รูปของกราฟหลังจากทำการใช้กฎ P12’

ผลลัพธ์ในรูปที่ 3.35 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P13' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P13 แสดงในรูปที่ 3.36

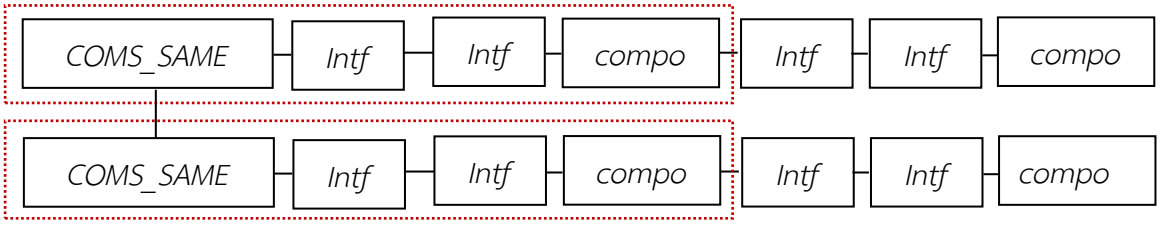


รูปที่ 3.35 รูปของกราฟก่อนจากทำการใช้กฎ P13’

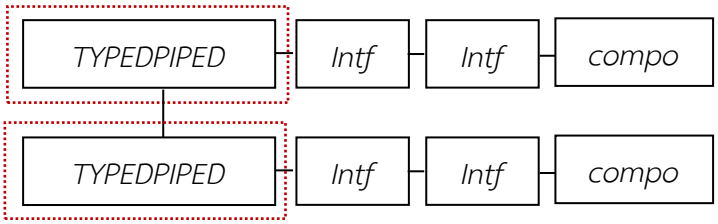


รูปที่ 3.36 รูปของกราฟหลังจากทำการใช้กฎ P13’

ผลลัพธ์ในรูปที่ 3.37 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P14' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P14 แสดงในรูปที่ 3.38

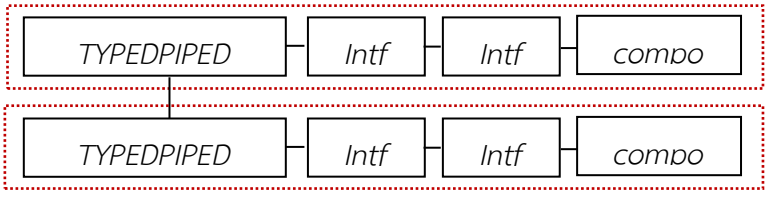


รูปที่ 3.37 รูปของกราฟก่อนจากทำการใช้กฎ P14’

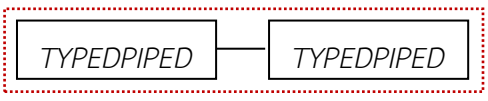


รูปที่ 3.38 รูปของกราฟหลังจากทำการใช้กฎ P14’

ผลลัพธ์ในรูปที่ 3.39 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P15' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P15 แสดงในรูปที่ 3.40



รูปที่ 3.39 รูปของกราฟก่อนจากทำการใช้กฎ P15’



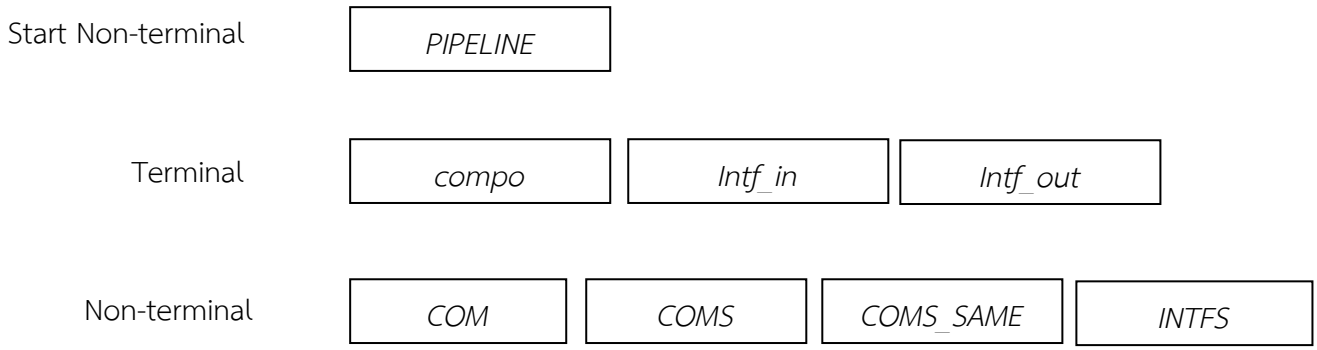
รูปที่ 3.40 รูปของกราฟหลังจากทำการใช้กฎ P15’

จากรูปที่ 3.40 จะเห็นได้ว่าผู้วิจัยสามารถใช้กฎ P16’ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P16 ในการสร้างจุดเริ่มต้น (Start Symbol) ซึ่งหมายความว่าในกราฟหลักสามารถค้นพบคุณสมบัติของสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไทป์ไปป์เพราะสามารถใช้กฎของการลดรูปลดรูปจนได้รูปแบบสถาปัตยกรรมเริ่มต้นได้โดยจะได้ผลลัพธ์ดังรูปที่ 3.41

รูปที่ 3.41 รูปของกราฟหลังจากทำการใช้กฎ P16’

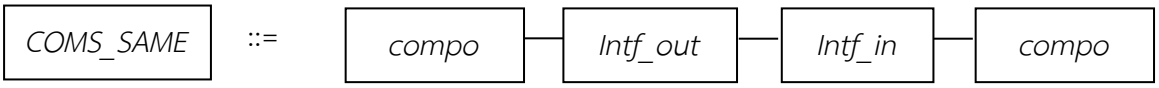
คำนิยามที่ 12 การตรวจสอบสถาปัตยกรรมในรูปไปป์และฟิลเตอร์ลักษณะไปป์ไลน์

ผู้วิจัยได้สร้างกราฟแกรมมาสำหรับตรวจสอบสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ โดยมีพื้นฐานมาจากนิยามข้างต้น โดยมีชุดของสัญลักษณ์ของนอนเทอร์มินัล และ ชุดของสัญลักษณ์ของเทอร์มินัล ซึ่งแสดงดังรูปที่ 3.42



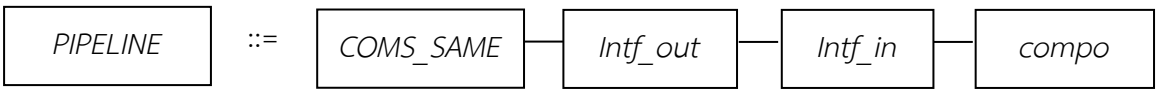
รูปที่ 3.42 สัญลักษณ์เริ่มต้น ชุดของสัญลักษณ์ของนอนเทอร์มินัล (non-terminal) และ ชุดของสัญลักษณ์ของเทอร์มินัล (terminal node)

จากรูปที่ 3.42 จะประกอบประกอบไปด้วยบัพที่มีลักษณะเทอร์มินัลและนอนเทอร์มินัล ซึ่งประกอบด้วย *PIPELINE* แทนสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ที่กำลังค้นหา COM แทนสัญลักษณ์ที่มีไว้สำหรับรวม compo และ intf หรือ INTFS เข้าไว้ด้วยกัน COMS แทนสัญลักษณ์ที่มีไว้สำหรับรวม COM หลายแห่งเข้าด้วยกัน และ INTFS แทนสัญลักษณ์ที่มีไว้สำหรับรวม intf หลายแห่งเข้าด้วยกัน โดยผู้วิจัยได้กำหนดชุดของกฎการผลิต P19-P23 และกฎการผลิต P19´- P23´ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้าย โดยกฎเหล่านี้มีไว้สำหรับการตรวจสอบสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ โดยแต่ละกฎการผลิตเขียนอยู่ในรูปของ L ::= R,C ไว้สำหรับตรวจสอบสถาปัตยกรรมดังรูปที่ 3.43



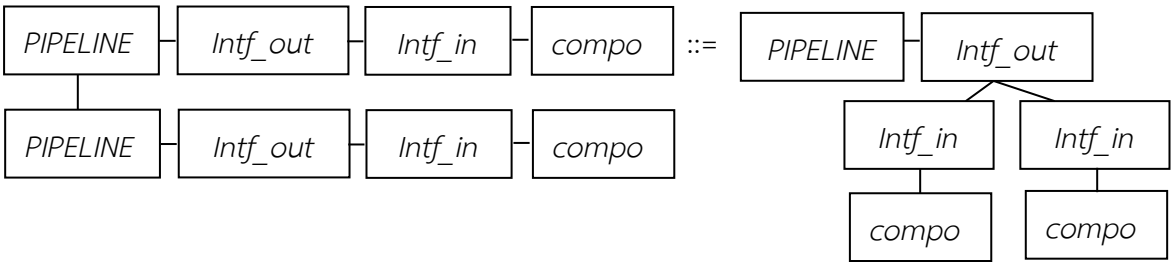
(P19)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพ COMS_SAME ประกอบไปด้วยบัพ compo ที่เชื่อมต่อกันสองบัพโดยผ่าน Intf_in และ Intf_out ของแต่ละบัพเป็นตัวเชื่อมต่อและ Intf ทั้งสองบัพจะต้องไปในทิศทางเดียวกัน โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าบัพ compo ที่เชื่อมต่อกันสองบัพโดยผ่าน Intf_in และ Intf_out ของแต่ละบัพเป็นตัวเชื่อมต่อและ Intf ทั้งสองบัพจะต้องไปในทิศทางเดียวกันนั้นสามารถแทนด้วยบัพ COMS_SAME



(P20)

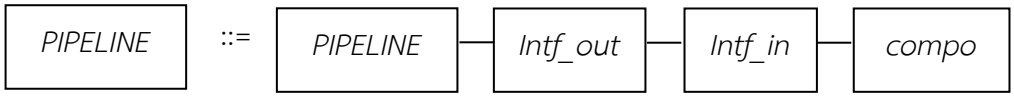
กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ประกอบไปด้วยบัพ COMS_SAME ที่เชื่อมต่อบัพ compo โดยเชื่อมต่อผ่านบัพ Intf_in และ Intf_out ซึ่งทั้งสองบัพจะต้องไปในทิศทางเดียวกัน โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าบัพ COMS_SAME ที่เชื่อมต่อบัพ compo โดยเชื่อมต่อผ่านบัพ Intf_in และ Intf_out ซึ่งทั้งสองบัพจะต้องไปในทิศทางเดียวกันนั้นสามารถแทนด้วยบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไปป์ไลน์



(P21)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไปป์ไลน์สองบัพที่แยกกันโดยมีเส้นเชื่อมโดยแต่ละบัพก็ยังไปเชื่อมต่อกับบัพอื่นผ่านทาง Intf_in และ Intf_out สามารถทำการรวมบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไปป์ไลน์และ Intf_out ที่เชื่อมต่อกับบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไปป์ไลน์เป็นบัพเดียวได้ โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ที่

เชื่อมต่อกับบัพ Intf_out และบัพเดียวกันนี้ก็แยกช่องทางการเชื่อมต่อออกมาอีกเป็นสองช่องทาง สามารถทำการแยกว่าบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไปป์ไลน์เป็นสองบัพที่แยกกันโดยมีเส้นเชื่อมถึงกันโดยแต่ละบัพก็ยังไปเชื่อมต่อกับบัพอื่นผ่านทาง Intf_in และ Intf_out



(P22)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ประกอบไปด้วยบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ที่เชื่อมต่อบัพ compo โดยเชื่อมต่อผ่านบัพ Intf_in และ Intf_out ทั้งสองบัพจะต้องไปในทิศทางเดียวกัน โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ที่เชื่อมต่อบัพ compo โดยเชื่อมต่อผ่านบัพ Intf_in และ Intf_out ทั้งสองบัพจะต้องไปในทิศทางเดียวกันนั้นสามารถแทนด้วยบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไทป์ไปป์

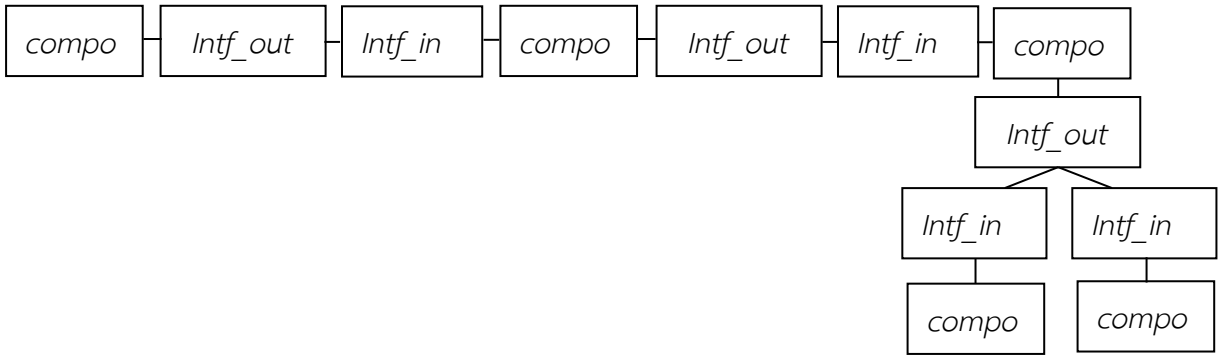
(P23)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ประกอบไปด้วยบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ที่เชื่อมต่อกันสองบัพ โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ที่เชื่อมต่อกันสองบัพสามารถแทนด้วยบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไปป์ไลน์

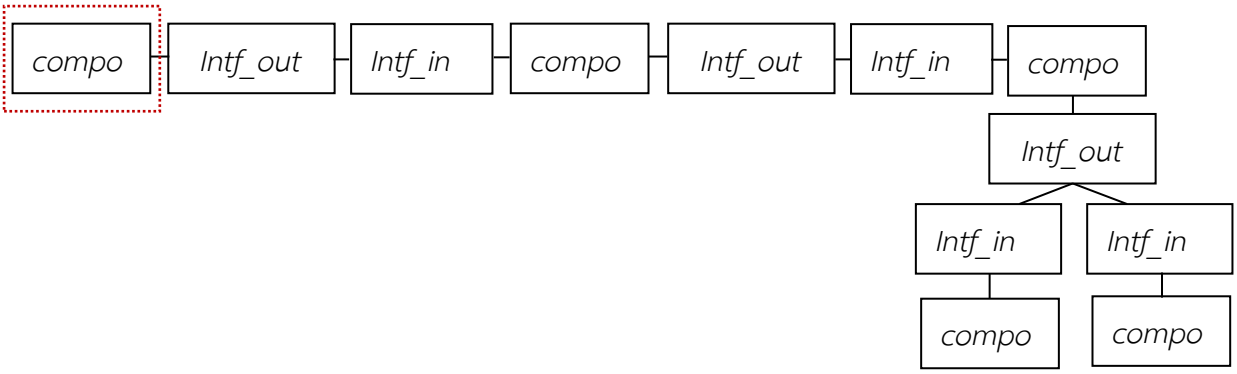
รูปที่ 3.43 กฎการผลิต P19-P23 โดยเขียนอยู่ในรูปของ L ::= R,C เพื่อตรวจสอบสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไปป์ไลน์

จากรูปที่ 3.43 แสดงถึงกฎการผลิตสำหรับการตรวจสอบสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ที่เกิดขึ้นที่ใช้กฎการผลิตที่มีอยู่นี้ในการลดรูปเพื่อให้เหลือรูปเริ่มต้นซึ่งในที่นี้จะใช้กฎข้อ P20 P22 หรือ P23 ในการลดรูปโดยจะแสดงในตัวอย่างต่อไปนี้

จากรูปที่ 3.44 เป็นตัวอย่างของรูปเริ่มต้นสำหรับการใช้กฎการลดรูปซึ่งจะใช้กฎการผลิต P19´- P23´ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายในการลดรูปเพื่อตรวจสอบรูปว่ามีสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ อยู่หรือไม่ โดยขั้นตอนแรกผู้วิจัยทำการเลือกบัพเริ่มต้น โดยดูจากคุณสมบัติของสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์เป็นหลัก ซึ่งคุณลักษณะของสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์นั้น คือ บัพเดินทางไปในทางเดียวกัน ซึ่งแต่ละบัพจะมีเส้นทางเข้าทางเดียวและทางออกทางเดียว โดยจะเลือกคอมโพเนนท์ที่มีอินเตอร์เฟสออกเดียวเป็นหลัก ซึ่งถ้ามีบัพที่เข้าเกณฑ์มากกว่า 1 บัพให้ใช้ตัวที่ตรวจพบก่อน โดยจากการเลือกบัพเริ่มต้นผู้วิจัยจะเลือกบัพเริ่มต้นดังรูปที่ 3.45

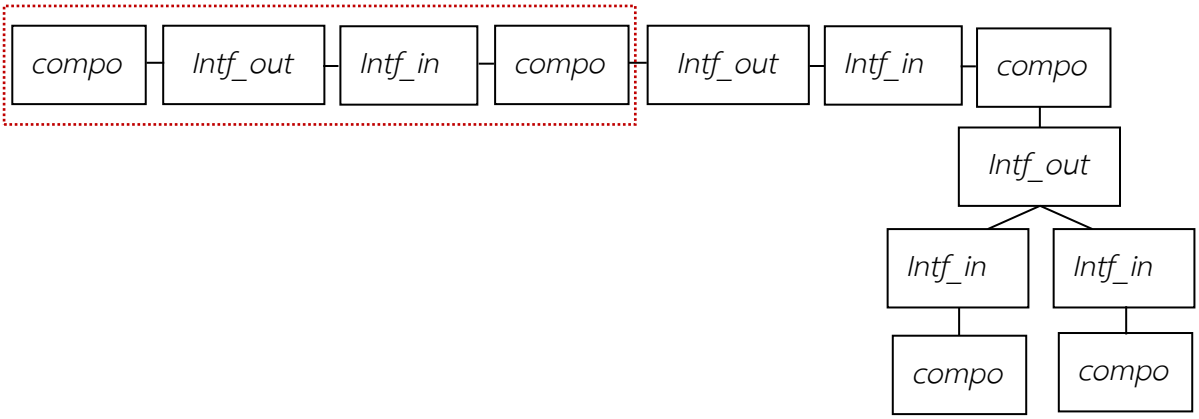


รูปที่ 3.44 รูปของกราฟเริ่มต้นที่จะทำการใช้กฎการลดรูป

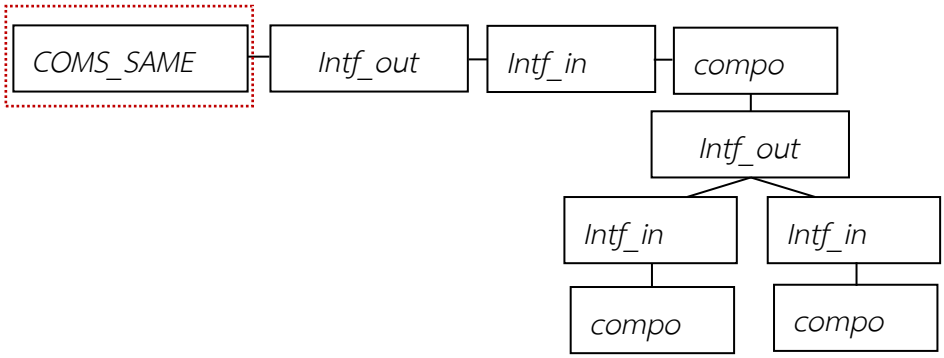


รูปที่ 3.45 รูปของกราฟเริ่มต้นที่จะทำการใช้กฎการลดรูป โดยมีการเลือกบัพเริ่มต้นแล้ว

จากรูปที่ 3.46 ผู้วิจัยสามารถใช้กฎข้อ P19’ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P19 เพื่อที่จะเปลี่ยนรูปร่างของกราฟ ซึ่งได้ผลลัพธ์ดังรูปที่ 3.47

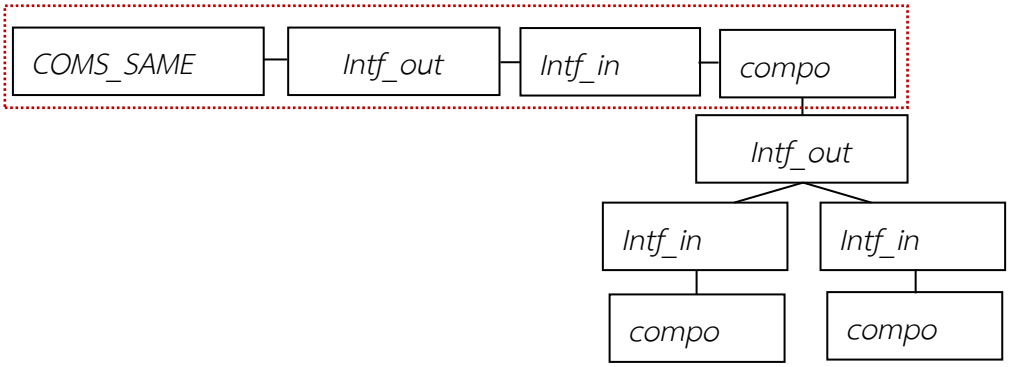


รูปที่ 3.46 รูปของกราฟก่อนทำการใช้กฎ P19’

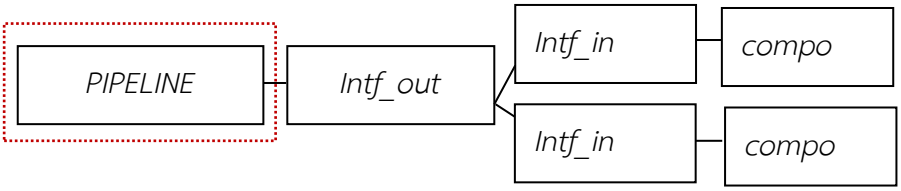


รูปที่ 3.47 รูปของกราฟหลังจากทำการใช้กฎ P19’

ผลลัพธ์ในรูปที่ 3.48 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P20' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P20 แสดงในรูปที่ 3.49

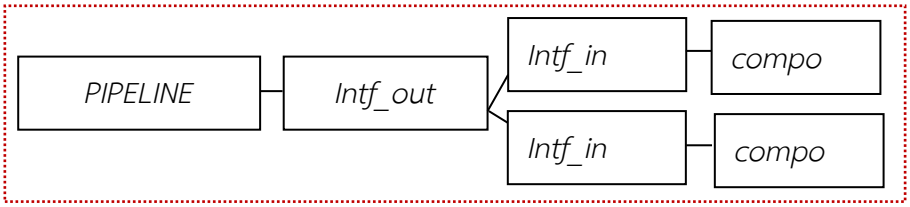


รูปที่ 3.48 รูปของกราฟก่อนทำการใช้กฎ P20’

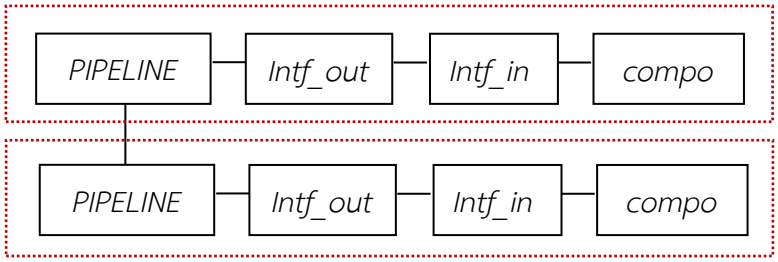


รูปที่ 3.49 รูปของกราฟหลังจากทำการใช้กฎ P20’

ผลลัพธ์ในรูปที่ 3.50 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P21' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P21 แสดงในรูปที่ 3.51

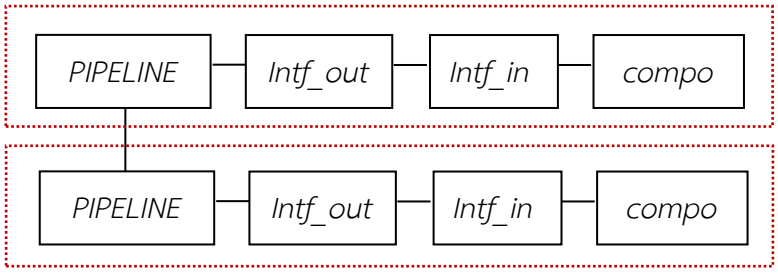


รูปที่ 3.50 รูปของกราฟก่อนทำการใช้กฎ P21’

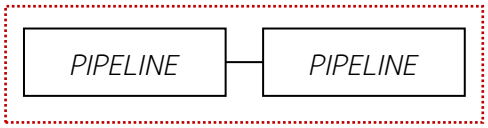


รูปที่ 3.51 รูปของกราฟหลังจากทำการใช้กฎ P21’

ผลลัพธ์ในรูปที่ 3.52 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P22' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P22 แสดงในรูปที่ 3.53



รูปที่ 3.52 รูปของกราฟก่อนทำการใช้กฎ P22’



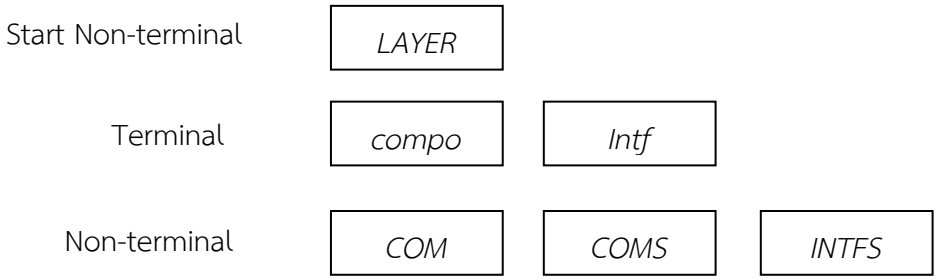
รูปที่ 3.53 รูปของกราฟหลังจากทำการใช้กฎ P22’

จากรูปที่ 3.53 จะเห็นได้ว่าผู้วิจัยสามารถใช้กฎ P23’ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P23 ในการสร้างจุดเริ่มต้น (Start Symbol) ซึ่งหมายความว่าในกราฟหลักสามารถค้นพบคุณสมบัติของสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์เพราะสามารถใช้กฎของการลดรูปลดรูปจนได้รูปแบบสถาปัตยกรรมเริ่มต้นได้โดยจะได้ผลลัพธ์ดังรูปที่ 3.54

รูปที่ 3.54 รูปของกราฟหลังจากทำการใช้กฎ P23’

คำนิยามที่ 13 การตรวจสอบสถาปัตยกรรมในรูปแบบลำดับชั้น

ผู้วิจัยได้สร้างกราฟแกรมมาสำหรับตรวจสอบสถาปัตยกรรมสำหรับตรวจสอบสถาปัตยกรรมในรูปแบบลำดับชั้น โดยมีพื้นฐานมาจากนิยามข้างต้น โดยมีชุดของสัญลักษณ์ของนอนเทอร์มินัล และ ชุดของสัญลักษณ์ของเทอร์มินัล ซึ่งแสดงดังรูปที่ 3.55



รูปที่ 3.55 สัญลักษณ์เริ่มต้น ชุดของสัญลักษณ์ของนอนเทอร์มินัล (non-terminal) และ ชุดของสัญลักษณ์ของเทอร์มินัล (terminal node)

จากรูปที่ 3.55 จะประกอบประกอบไปด้วยบัพที่มีลักษณะเทอร์มินัลและนอนเทอร์มินัล ซึ่งประกอบด้วย LAYER แทนรูปแบบสถาปัตยกรรมซอฟต์แวร์ในรูปแบบลำดับชั้นที่กำลังค้นหา COM แทนสัญลักษณ์ที่มีไว้สำหรับรวม compo และ intf หรือ INTFS เข้าไว้ด้วยกัน COMS แทนสัญลักษณ์ที่มีไว้สำหรับรวม COM หลายแห่งเข้าด้วยกัน และ INTFS แทนสัญลักษณ์ที่มีไว้สำหรับรวม intf หลายแห่งเข้าด้วยกัน โดยผู้วิจัยได้กำหนดชุดของกฎการผลิต P24-P27 และกฎการผลิต P24´- P27´ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้าย โดยกฎเหล่านี้มีไว้สำหรับการตรวจสอบสถาปัตยกรรมในรูปแบบลำดับชั้น โดยแต่ละกฎการผลิตเขียนอยู่ในรูปของ L ::= R,C ไว้สำหรับตรวจสอบสถาปัตยกรรมดัง

(P24)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบลำดับชั้นประกอบไปด้วยบัพ COMS ที่เชื่อมต่อบัพ COM โดยทั้งสองอยู่ภายใต้ขอบเขตลำดับชั้น โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าบัพ COMS ที่เชื่อมต่อบัพ COM โดยทั้งสองอยู่ภายใต้ขอบเขตลำดับชั้นนั้นสามารถแทนด้วยบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบลำดับชั้น

(P25)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบลำดับชั้นที่เชื่อมต่อกับบัพ COM สามารถทำการแยกบัพ intf ออกมาจากบัพ COM ได้ โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบรูปแบบลำดับชั้นที่เชื่อมต่อกับบัพ COM สามารถทำการรวมบัพ intf มาไว้ในบัพ COM ได้

(P26)

กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบลำดับชั้นประกอบไปด้วยบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบลำดับชั้นเชื่อมต่อกับบัพ COMS ที่เชื่อมต่อบัพ COM โดยทั้งหมดอยู่ภายใต้ขอบเขตลำดับ

ชั้นเดียวกัน โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบลำดับชั้นประกอบไปด้วยบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบลำดับชั้นเชื่อมต่อกับบัพ COMS ที่เชื่อมต่อบัพ COM นั้นสามารถแทนด้วยบัพที่แทนคุณลักษณะสถาปัตยกรรมในรูปแบบลำดับชั้น

(P27)

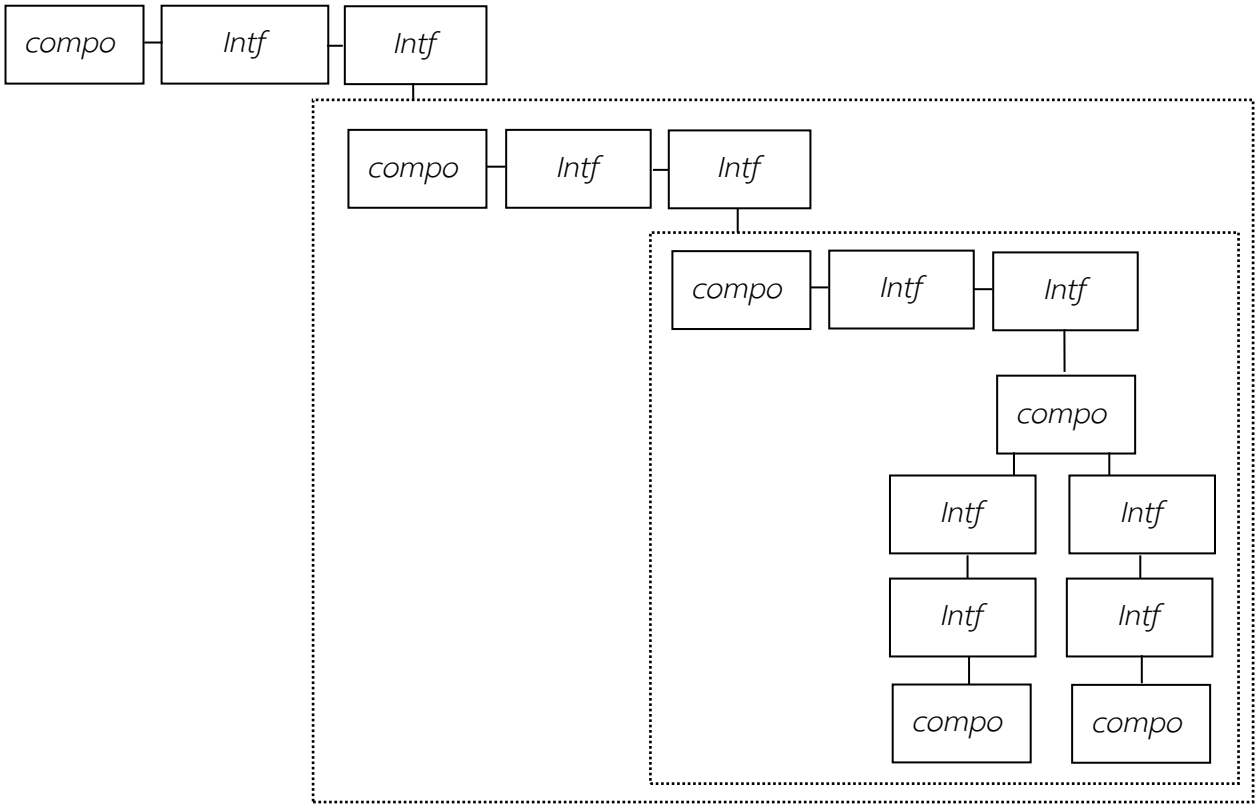
กฎข้อนี้มีไว้สำหรับในการบ่งบอกว่าบัพ COM ที่เชื่อมต่อกับบัพ COM สามารถทำการแยกบัพ intf ออกมาจากบัพ COM ได้ โดยในทางกลับกันกฎข้อนี้ก็จะบ่งบอกว่าบัพ COM ที่เชื่อมต่อกับบัพ COM สามารถทำการรวมบัพ intf มาไว้ในบัพ COM ได้

รูปที่ 3. 56 กฎการผลิต P24-P27 โดยเขียนอยู่ในรูปของ L ::= R,C เพื่อตรวจสอบสถาปัตยกรรมในรูปแบบลำดับชั้น

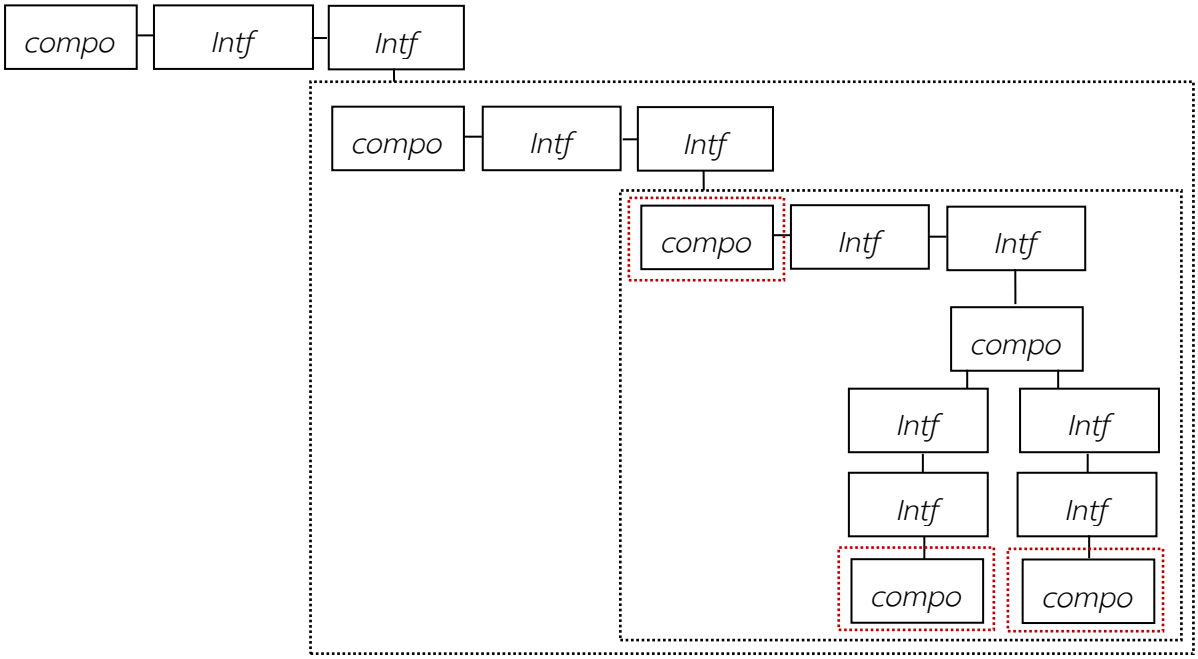
จากรูปที่ 3. 56 แสดงถึงกฎการผลิตสำหรับการตรวจสอบสถาปัตยกรรมในรูปแบบไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ที่เกิดขึ้นที่ใช้กฎการผลิตที่มีอยู่นี้ในการลดรูปเพื่อให้เหลือรูปเริ่มต้นซึ่งในที่นี้จะใช้กฎข้อ P24 หรือ P26 ในการลดรูปโดยจะแสดงในตัวอย่างต่อไปนี้

จากรูปที่ 3.57 เป็นตัวอย่างของรูปเริ่มต้นสำหรับการใช้กฎการลดรูปซึ่งจะใช้กฎการผลิต P24´- P27´ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายในการลดรูป เพื่อตรวจสอบรูปว่ามีคุณลักษณะของสถาปัตยกรรมในรูปแบบลำดับชั้นอยู่หรือไม่ โดยขั้นตอนแรกผู้วิจัยทำการเลือกบัพเริ่มต้น โดยดูจากคุณสมบัติของสถาปัตยกรรมในรูปแบบลำดับชั้น ซึ่งจะทำการเลือกบัพที่อยู่ชั้นในสุดและคอมโพเนนท์ที่มีอินเตอร์เฟสเพียงอันเดียวเป็นหลัก ซึ่งถ้ามีบัพที่เข้าเกณ

มากกว่า 1 บัพให้ใช้ตัวที่ตรวจพบก่อน โดยการเลือกบัพเริ่มต้นผู้วิจัยจะเลือกบัพเริ่มต้นดังรูปที่ 3.58

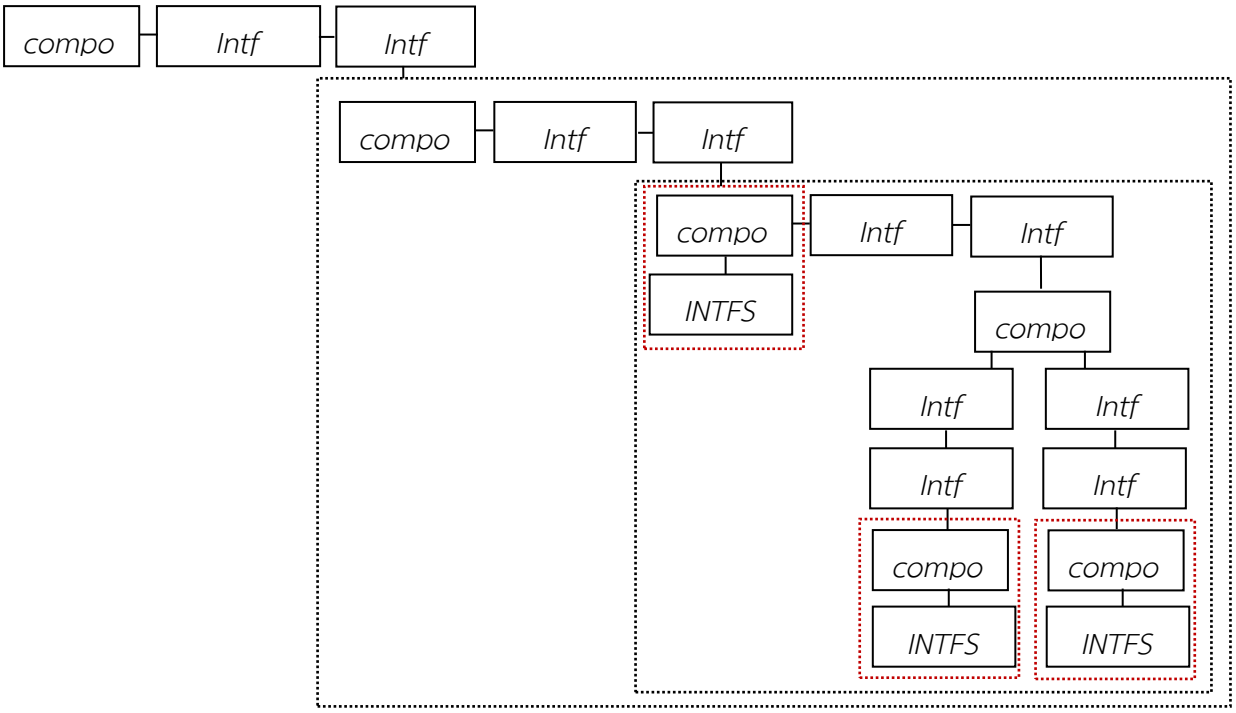


รูปที่ 3.57 รูปของกราฟเริ่มต้นที่จะทำการใช้กฎการลดรูป



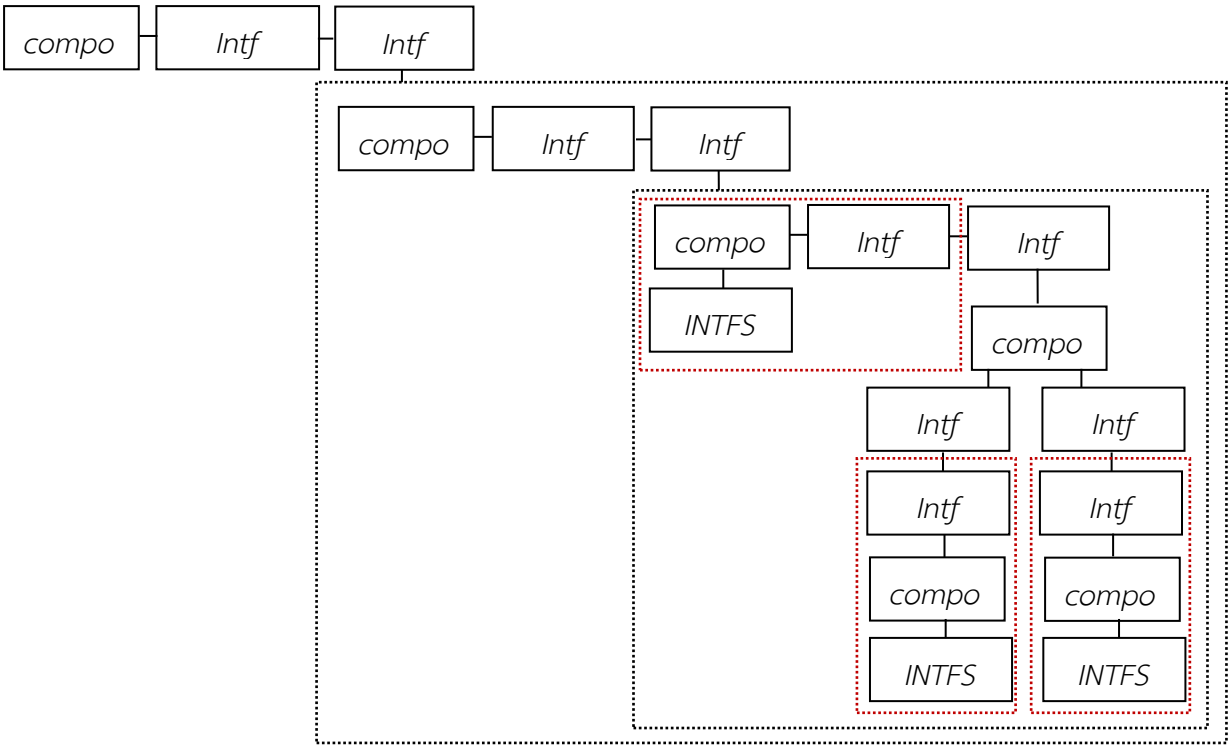
รูปที่ 3.58 รูปของกราฟเริ่มต้นที่จะทำการใช้กฎการลดรูป

หลังจากทำการเลือกบัพเริ่มต้นแล้ว เป้าหมายของผู้วิจัยคือการทำให้บัพในแต่ละชั้นอยู่ในรูปที่จะใช้กฎข้อ P24 ได้ ดังนั้นผู้วิจัยสามารถใช้กฎข้อ P7’ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P7 เพื่อที่จะเปลี่ยนรูปร่างของกราฟ ซึ่งจะได้ผลดังรูปที่ 3.59

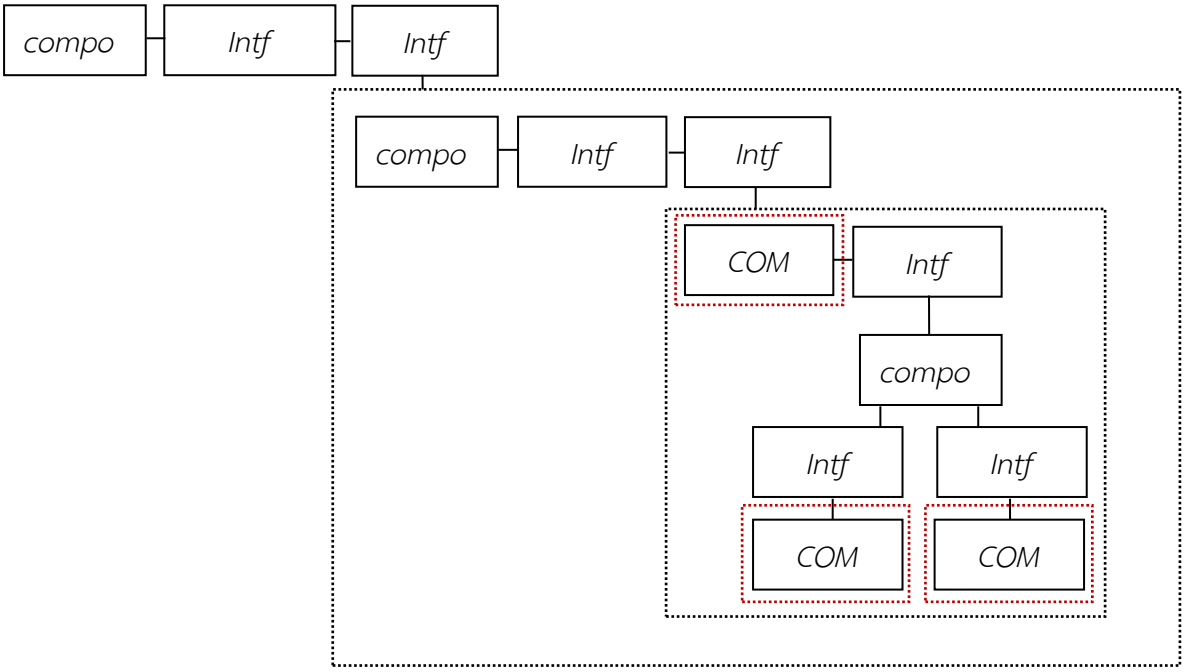


รูปที่ 3.59 รูปของกราฟหลังจากทำการใช้กฎ P7’

ผลลัพธ์ในรูปที่ 3.60 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P5' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P5 แสดงในรูปที่ 3.61

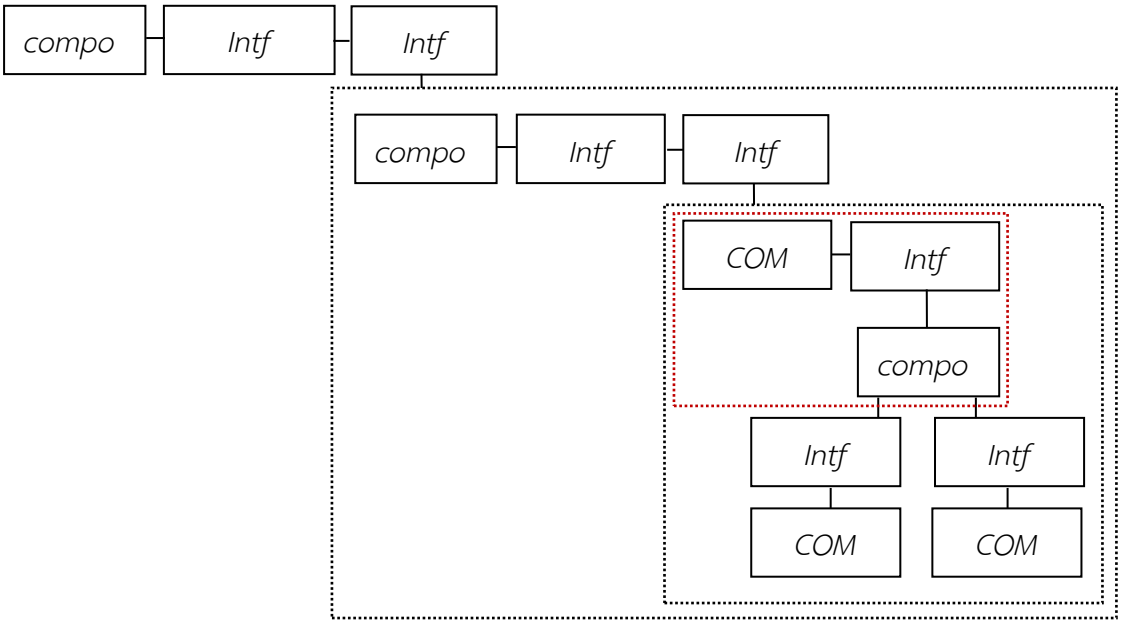


รูปที่ 3.60 รูปของกราฟก่อนทำการใช้กฎ P5’

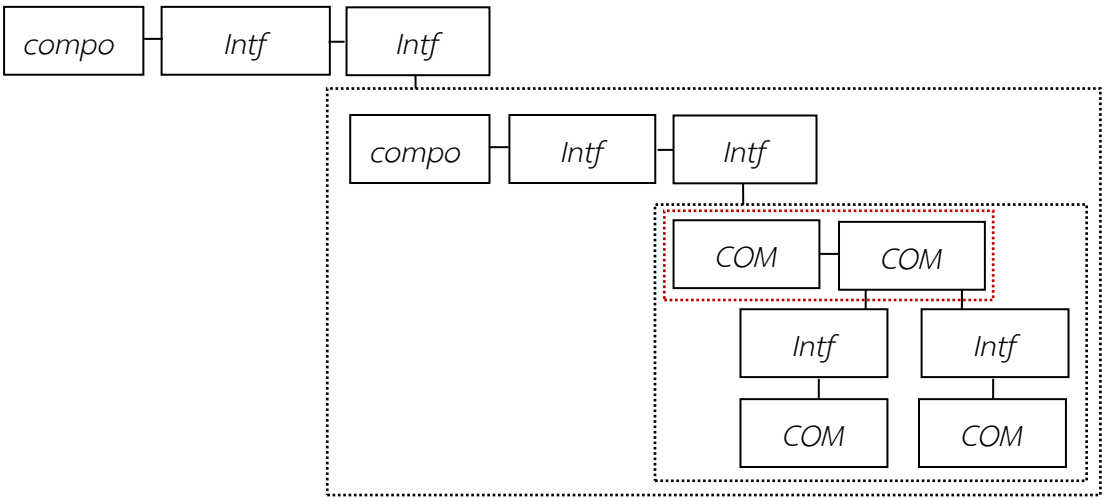


รูปที่ 3.61 รูปของกราฟหลังจากทำการใช้กฎ P5’

ผลลัพธ์ในรูปที่ 3.62 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P4' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P4 แสดงในรูปที่ 3.63

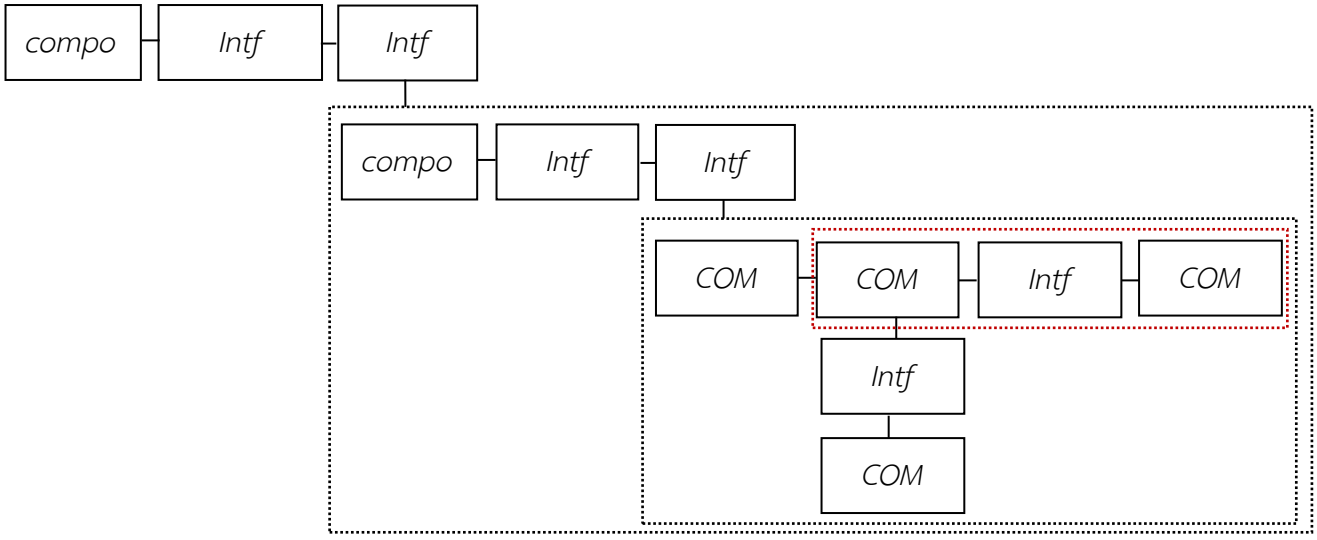


รูปที่ 3.62 รูปของกราฟก่อนทำการใช้กฎ P4’

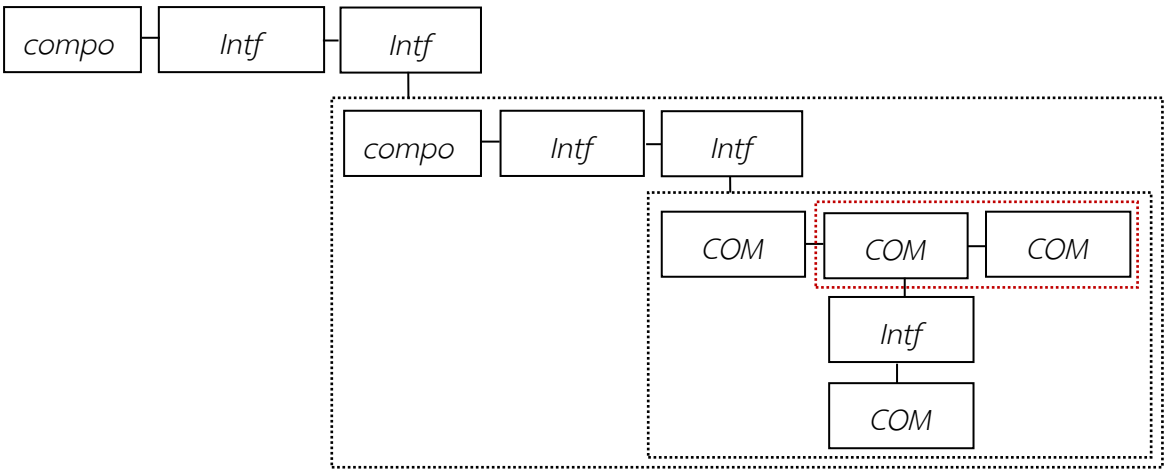


รูปที่ 3.63 รูปของกราฟหลังจากทำการใช้กฎ P4’

ผลลัพธ์ในรูปที่ 3.64 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P27' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P27 แสดงในรูปที่ 3.65

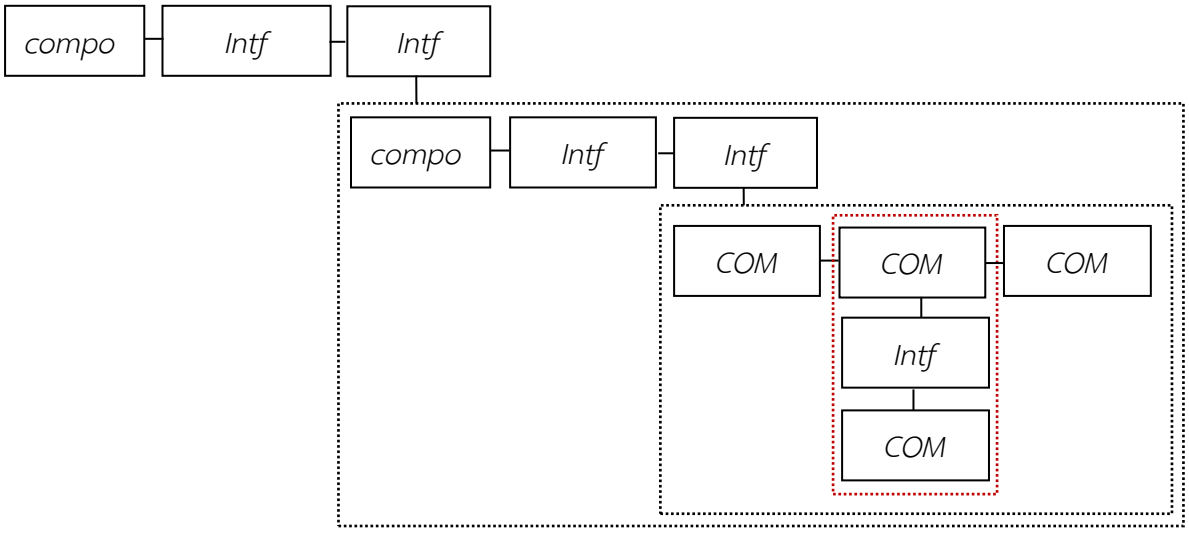


รูปที่ 3.64 รูปของกราฟก่อนทำการใช้กฎ P27’

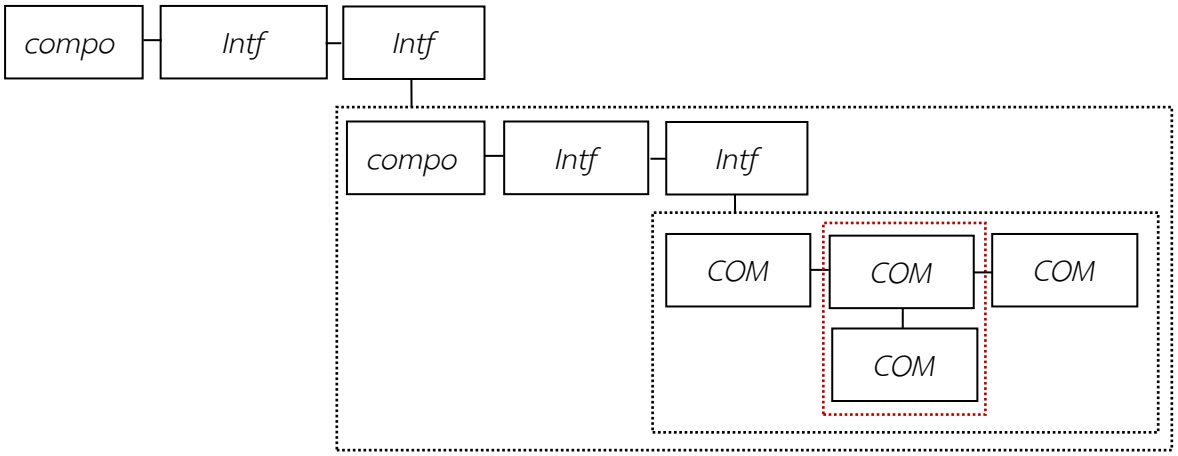


รูปที่ 3.65 รูปของกราฟหลังจากทำการใช้กฎ P27’

ผลลัพธ์ในรูปที่ 3.66 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P27' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P27 แสดงในรูปที่ 3.67

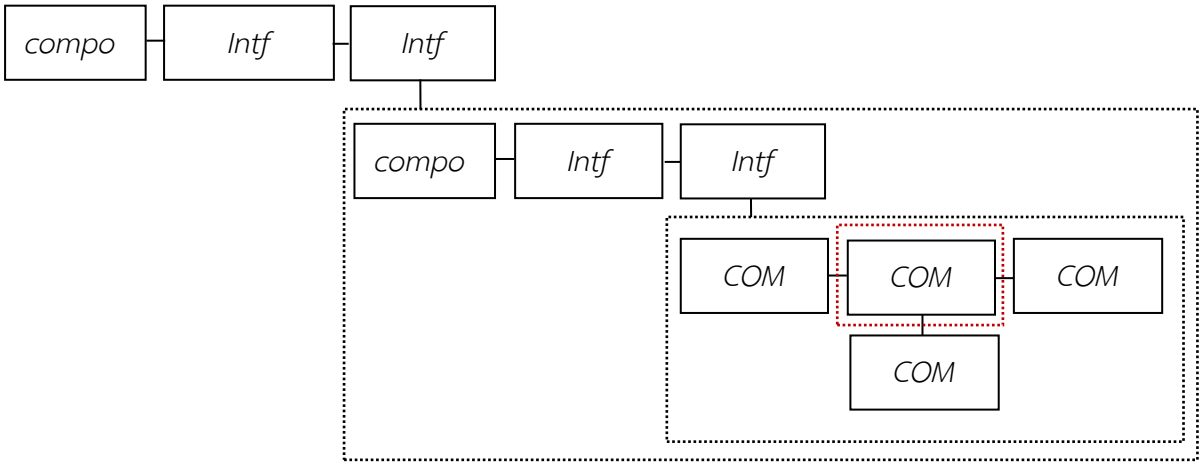


รูปที่ 3.66 รูปของกราฟก่อนทำการใช้กฎ P27’

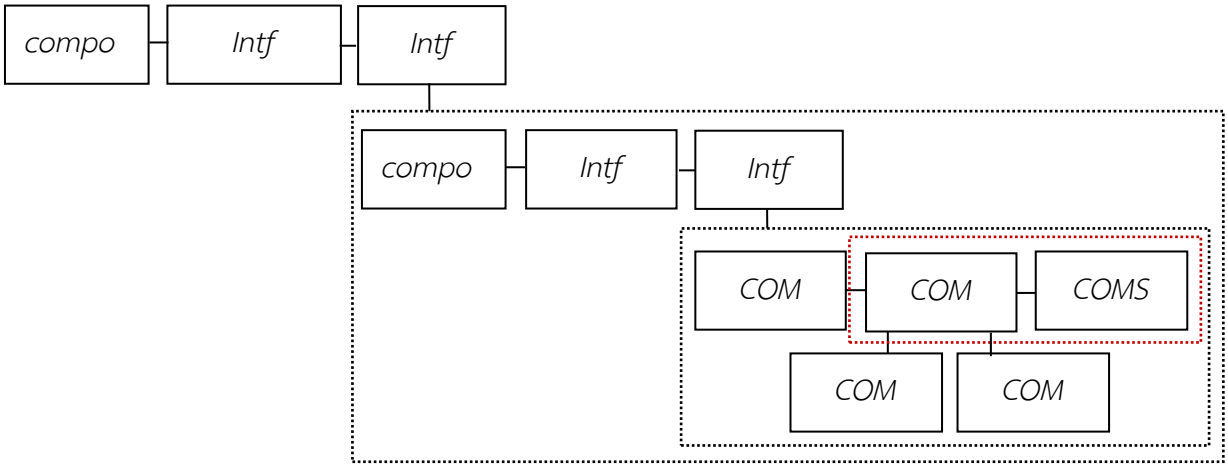


รูปที่ 3.67 รูปของกราฟหลังจากทำการใช้กฎ P27’

ผลลัพธ์ในรูปที่ 3.68 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P3' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P3 แสดงในรูปที่ 3.69

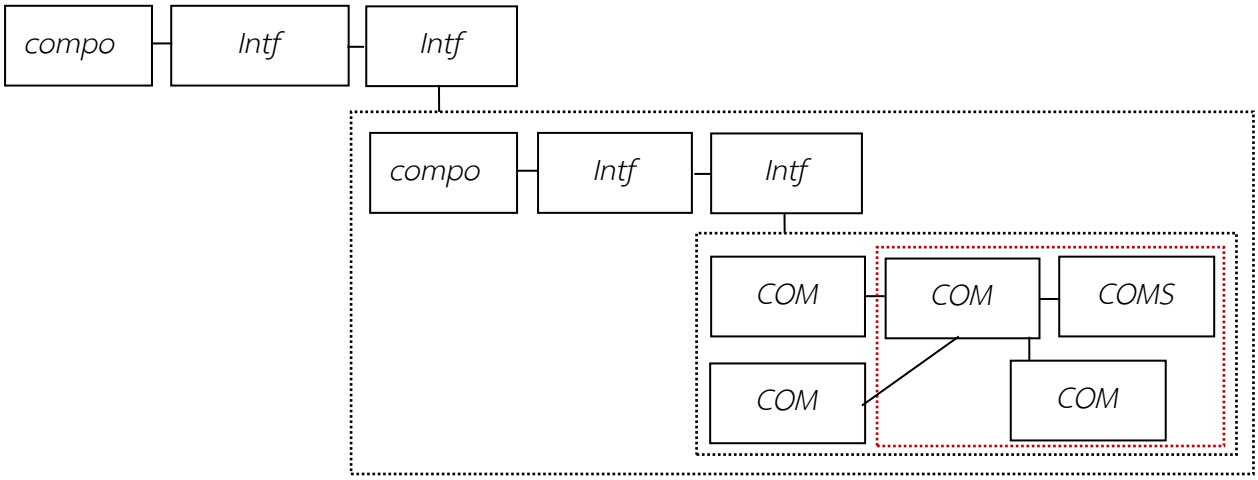


รูปที่ 3.68 รูปของกราฟก่อนทำการใช้กฎ P3’

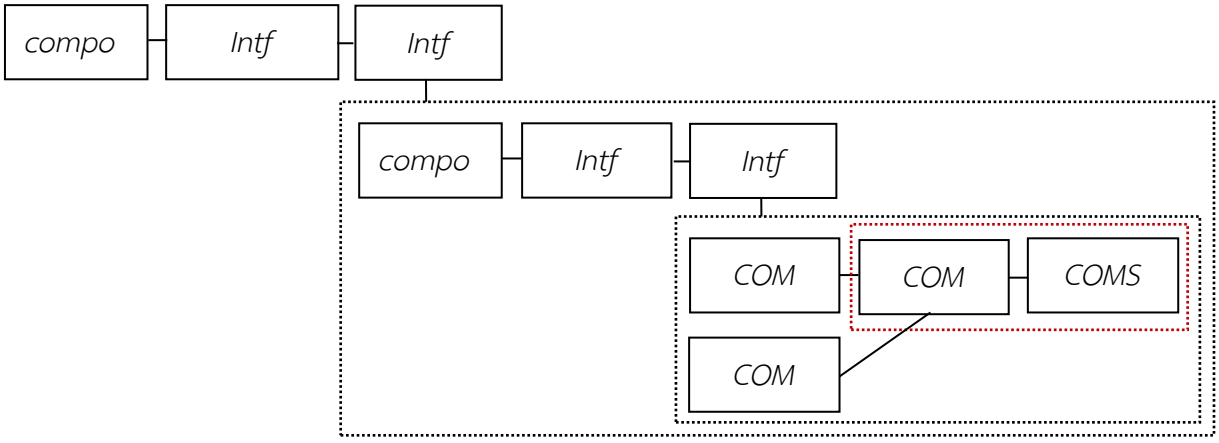


รูปที่ 3.69 รูปของกราฟหลังจากทำการใช้กฎ P3’

ผลลัพธ์ในรูปที่ 3.70 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P2' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P2 แสดงในรูปที่ 3.71

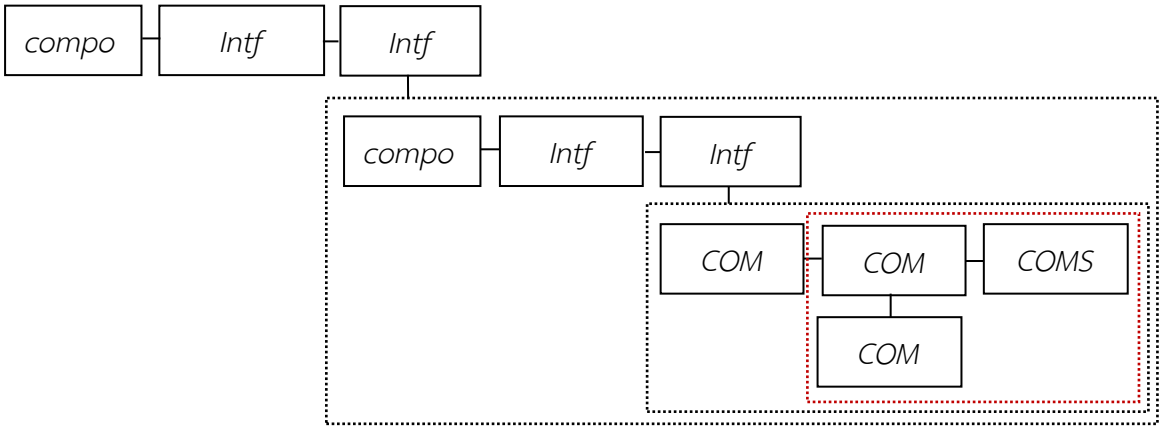


รูปที่ 3.70 รูปของกราฟก่อนทำการใช้กฎ P2’

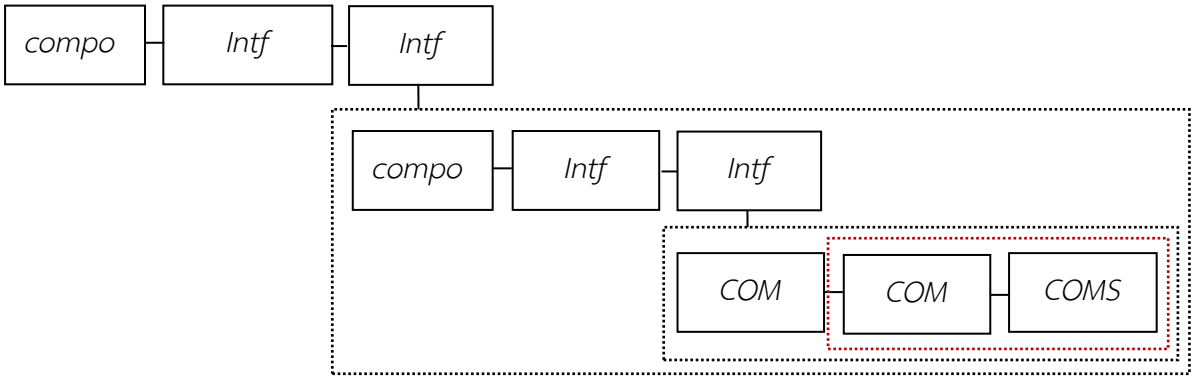


รูปที่ 3.71 รูปของกราฟหลังจากทำการใช้กฎ P2’

ผลลัพธ์ในรูปที่ 3.72 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P2' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P2 แสดงในรูปที่ 3.73

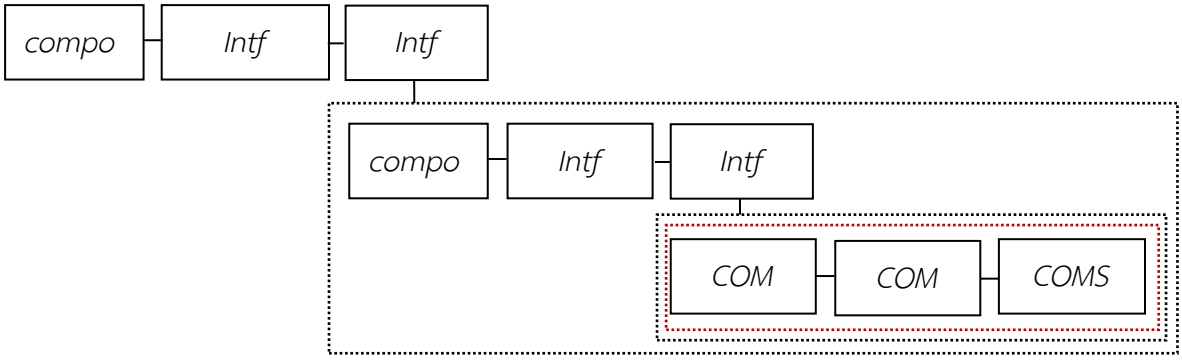


รูปที่ 3.72 รูปของกราฟก่อนทำการใช้กฎ P2’

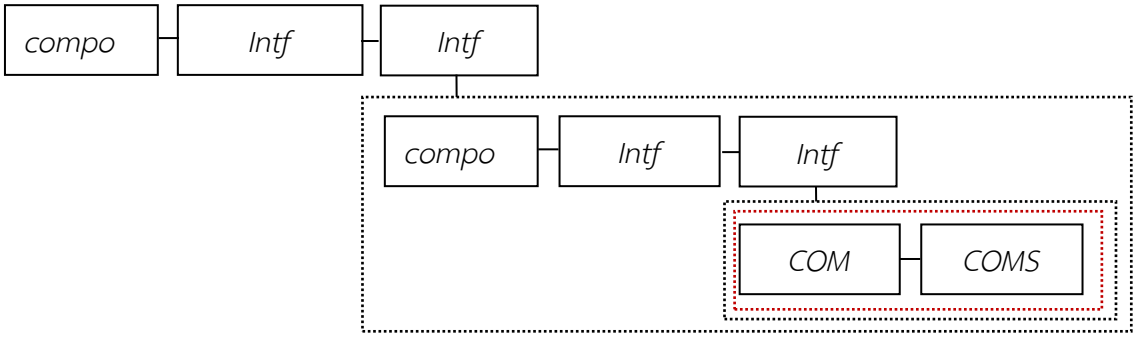


รูปที่ 3.73 รูปของกราฟหลังจากทำการใช้กฎ P2’

ผลลัพธ์ในรูปที่ 3.74 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P2' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P2 แสดงในรูปที่ 3.75

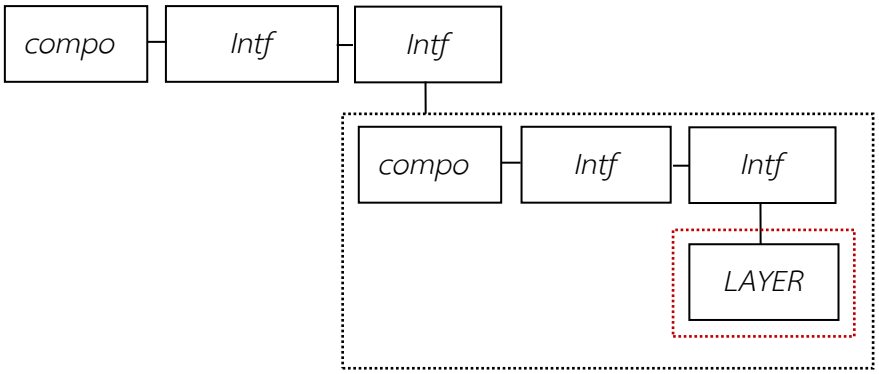


รูปที่ 3.74 รูปของกราฟก่อนทำการใช้กฎ P2’



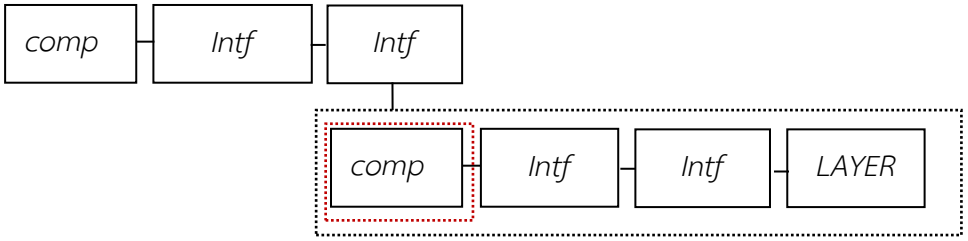
รูปที่ 3.75 รูปของกราฟหลังจากทำการใช้กฎ P2’

ผลลัพธ์ในรูปที่ 3.75 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P24' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P24 แสดงในรูปที่ 3.76

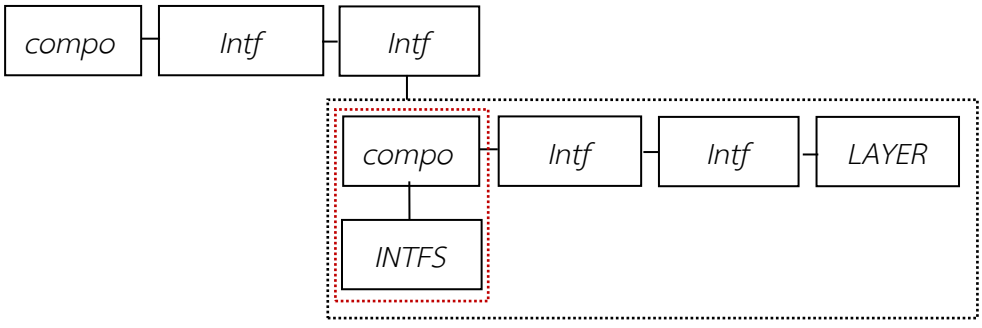


รูปที่ 3.76 รูปของกราฟหลังจากทำการใช้กฎ P24’

ผลลัพธ์ในรูปที่ 3.77 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P7' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P7 แสดงในรูปที่ 3. 78

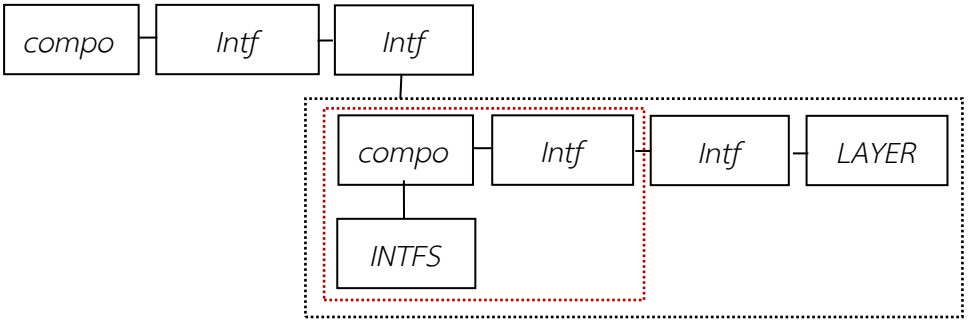


รูปที่ 3.77 รูปของกราฟก่อนจากทำการใช้กฎ P7’

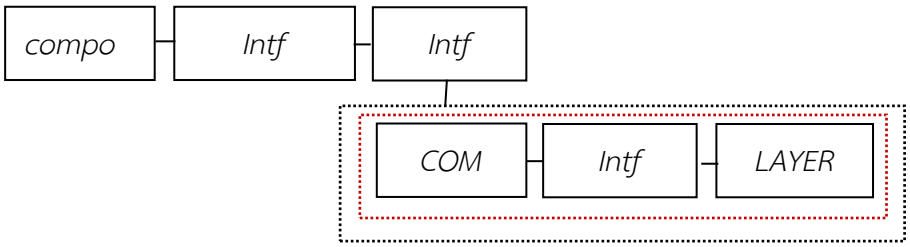


รูปที่ 3. 78 รูปของกราฟหลังจากทำการใช้กฎ P7’

ผลลัพธ์ในรูปที่ 3.79 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P5' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P5 แสดงในรูปที่ 3.80

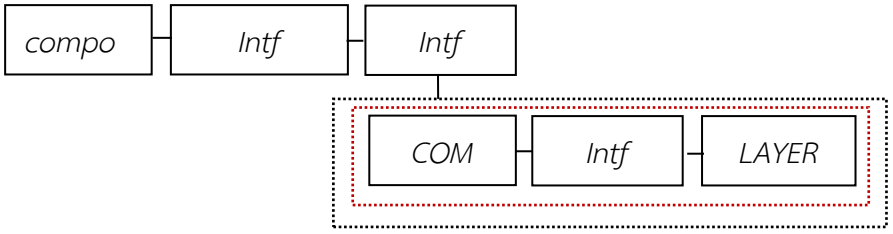


รูปที่ 3.79 รูปของกราฟก่อนทำการใช้กฎ P5’

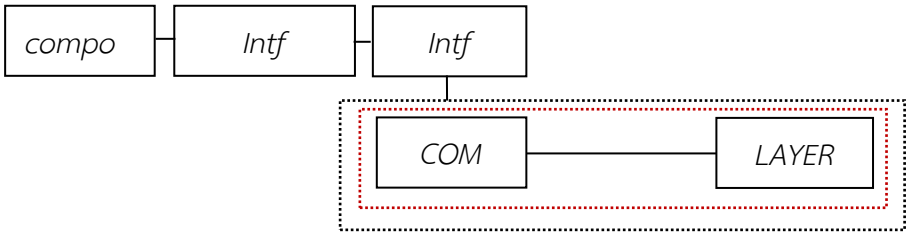


รูปที่ 3.80 รูปของกราฟหลังจากทำการใช้กฎ P5’

ผลลัพธ์ในรูปที่ 3.81 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P25' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P25 แสดงในรูปที่ 3. 82

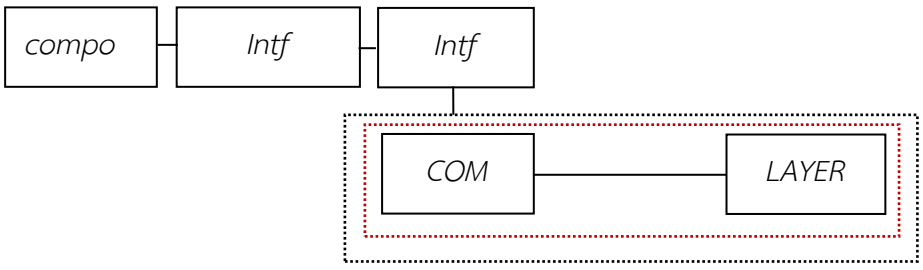


รูปที่ 3.81 รูปของกราฟก่อนทำการใช้กฎ P25’

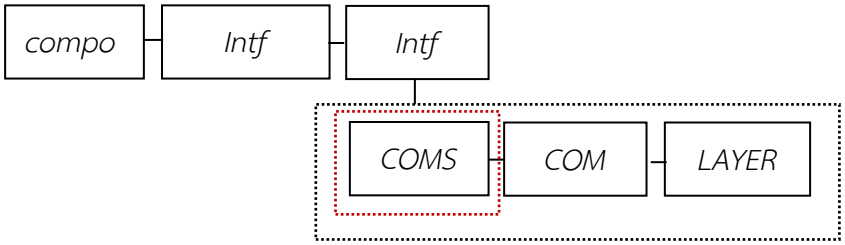


รูปที่ 3. 82 รูปของกราฟหลังจากทำการใช้กฎ P25’

ผลลัพธ์ในรูปที่ 3.83 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P3' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P3 แสดงในรูปที่ 3.84

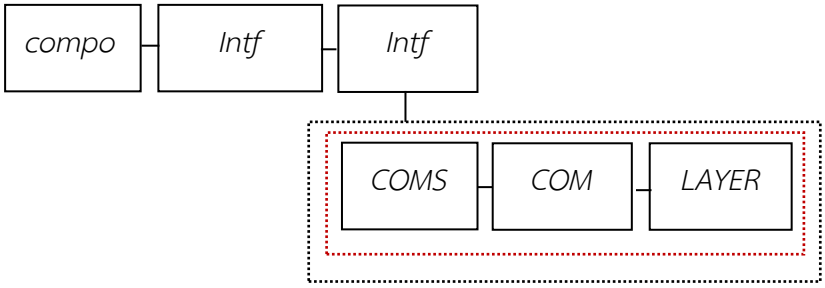


รูปที่ 3.83 รูปของกราฟก่อนทำการใช้กฎ P3’

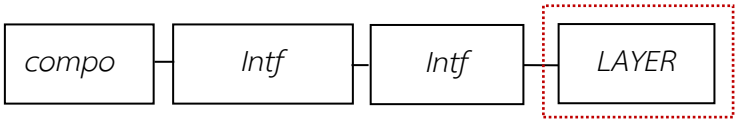


รูปที่ 3.84 รูปของกราฟหลังจากทำการใช้กฎ P3’

ผลลัพธ์ในรูปที่ 3.85 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P26' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P26 แสดงในรูปที่ 3.86

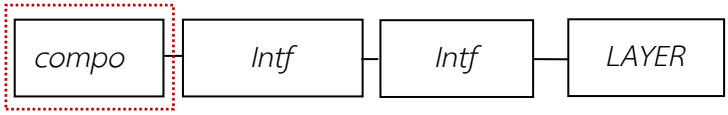


รูปที่ 3.85 รูปของกราฟก่อนทำการใช้กฎ P26’

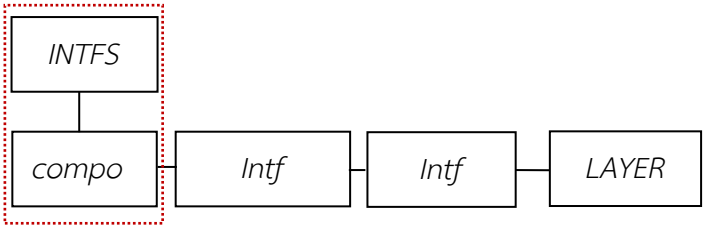


รูปที่ 3.86 รูปของกราฟหลังจากทำการใช้กฎ P26’

ผลลัพธ์ในรูปที่ 3.87 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P7' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P7 แสดงในรูปที่ 3.88

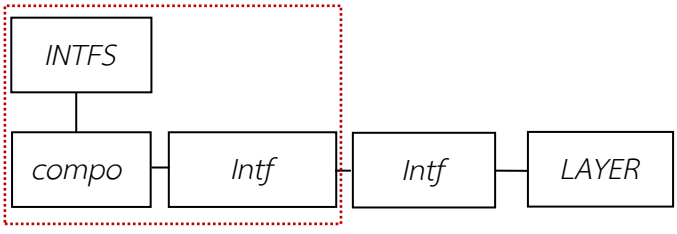


รูปที่ 3.87 รูปของกราฟก่อนทำการใช้กฎ P7’

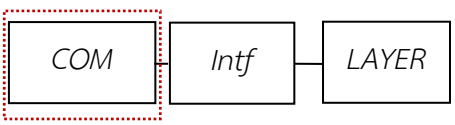


รูปที่ 3.88 รูปของกราฟหลังจากทำการใช้กฎ P7’

ผลลัพธ์ในรูปที่ 3.89 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P5' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P5 แสดงในรูปที่ 3.90

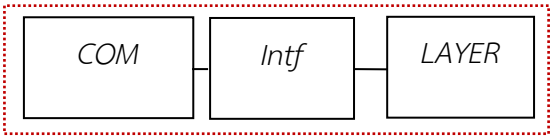


รูปที่ 3.89 รูปของกราฟก่อนทำการใช้กฎ P5’

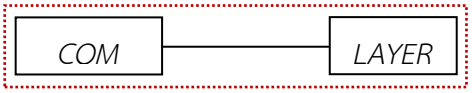


รูปที่ 3.90 รูปของกราฟหลังจากทำการใช้กฎ P5’

ผลลัพธ์ในรูปที่ 3.91 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P25' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P25 แสดงในรูปที่ 3.92

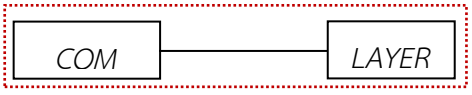


รูปที่ 3.91 รูปของกราฟก่อนทำการใช้กฎ P25’



รูปที่ 3.92 รูปของกราฟหลังจากทำการใช้กฎ P25’

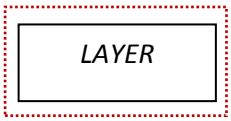
ผลลัพธ์ในรูปที่ 3.93 ผู้วิจัยนำมาพิจารณาลดรูปต่อไปโดยพบว่ามีส่วนกราฟที่จะได้รับการลดรูปโดยใช้กฎ P3' จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P3 แสดงในรูปที่ 3.94



รูปที่ 3.93 รูปของกราฟก่อนทำการใช้กฎ P3’

รูปที่ 3.94 รูปของกราฟก่อนทำการใช้กฎ P3’

จากรูปที่ 3.94 จะเห็นได้ว่าผู้วิจัยสามารถใช้กฎ P26’ จะเป็นการพิจารณากฎโดยย้อนจากด้านขวามาทางด้านซ้ายของกฎข้อ P26 ในการสร้างจุดเริ่มต้น (Start Symbol) ซึ่งหมายความว่าในกราฟหลักสามารถค้นพบคุณสมบัติของสถาปัตยกรรมในรูปแบบลำดับชั้นเพราะสามารถใช้กฎของการลดรูปลดรูปจนได้รูปแบบสถาปัตยกรรมเริ่มต้นได้โดยจะได้ผลลัพธ์ดังรูปที่ 3.95



รูปที่ 3.95 รูปของกราฟที่ได้จุดเริ่มต้น (Start Symbol) ที่มีชื่อว่า LAYER

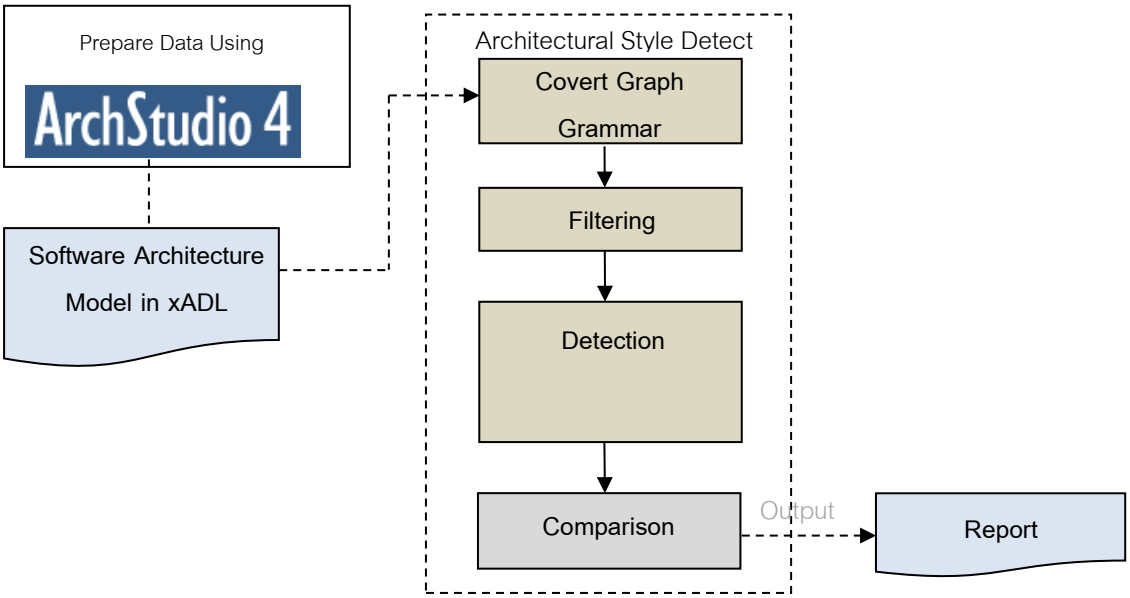
ซึ่งขั้นตอนทั้งหมดที่ได้กล่าวมาข้างต้นเป็นวิธีการตรวจสอบสถาปัตยกรรมของแต่ละสถาปัตยกรรมที่ทางผู้วิจัยได้ทำการวิจัย ซึ่งมีสถาปัตยกรรมในรูปแบบรวมศูนย์กลาง สถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้น สถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ สถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ และ สถาปัตยกรรมในรูปแบบลำดับชั้น

บทที่ 4
การออกแบบและการพัฒนาเครื่องมือ

สำหรับการออกแบบและพัฒนาเครื่องมือตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์ ซึ่งอธิบายแยกเป็นสองส่วนโดยส่วนแรกกล่าวถึงการออกแบบเครื่องมือ ซึ่งอธิบายภาพรวมการดำเนินงานเครื่องมือตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์และขั้นตอนการดำเนินงาน โดยนำเสนอด้วยแผนภาพกิจกรรม แผนภาพยูสเคสและแผนภาพคลาสตามลำดับ และส่วนที่สองกล่าวถึงการพัฒนาเครื่องมือตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์ กล่าวถึงสภาพแวดล้อมสำหรับการพัฒนาเครื่องมือและส่วนต่อประสานกับผู้ใช้เครื่องมือ ซึ่งมีรายละเอียด ดังนี้

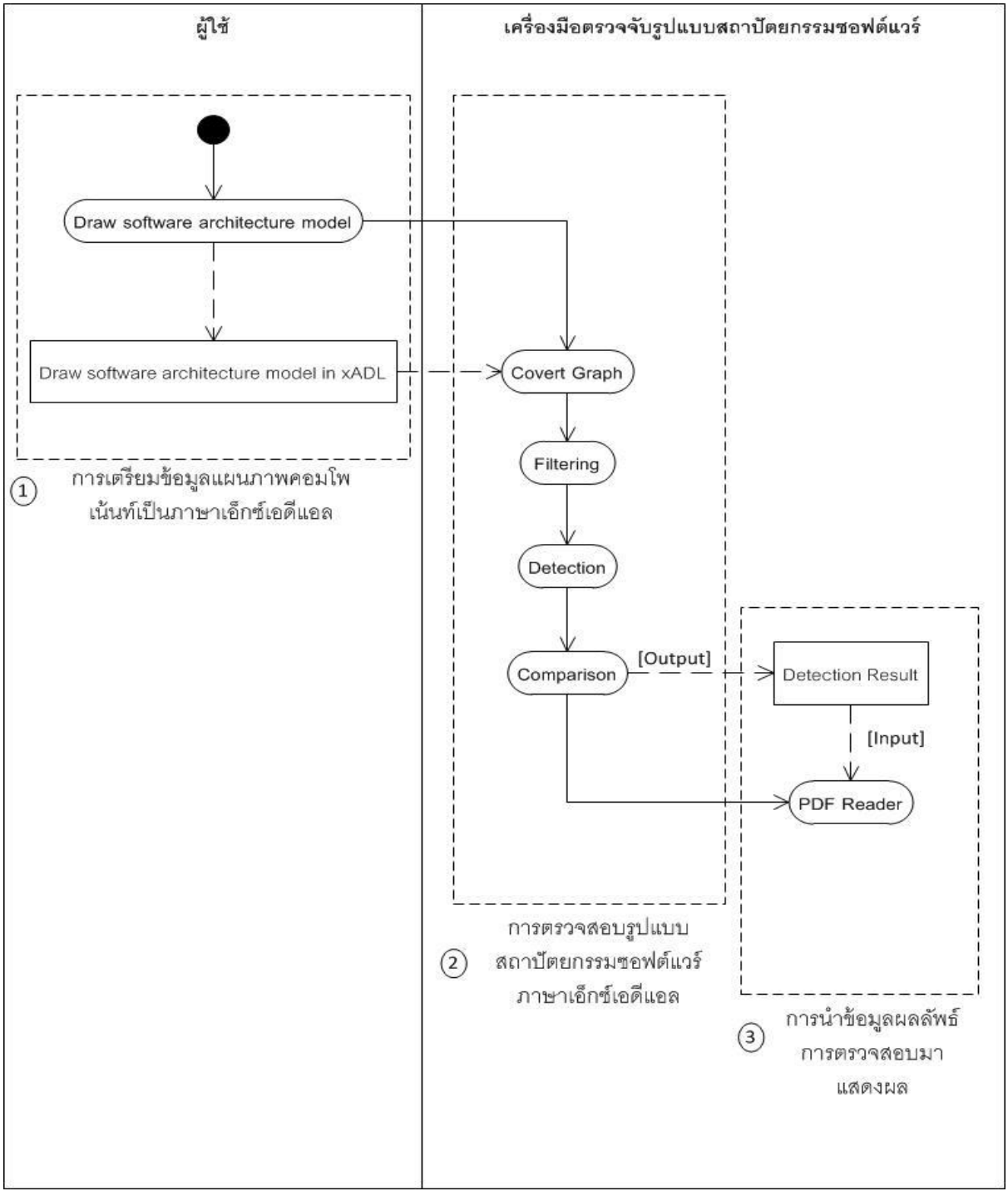
4.1 ภาพรวมการดำเนินงานเครื่องมือตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์

ในภาพรวมการดำเนินงานเครื่องมือตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์ ซึ่งข้อมูลแผนภาพถูกเขียนขึ้นโดยผู้ใช้ผ่านเครื่องมือทางภาษาเอกซ์เอดีแอล ตัวอย่างเช่น ArchStudio4 เป็นต้น จะถูกจัดเก็บในรูปแบบภาษาเอกซ์เอดีแอล โดยการวิเคราะห์จากลักษณะของข้อมูลในภาษาเอกซ์เอดีแอลที่ถูกแปลงเป็นกราฟแกรมมา เพื่อตรวจสอบถึงรูปแบบสถาปัตยกรรมและนำข้อมูลผลลัพธ์ที่ได้ไปใช้งานต่อไป โดยแสดงภาพรวมการดำเนินงานแสดงดังรูปที่ 4.1



รูปที่ 4.1 ภาพรวมการดำเนินงานเครื่องมือตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์

จากรูปที่ 4.1 แสดงแผนภาพกิจกรรมภาพรวมการดำเนินงานเครื่องมือตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์ดังรูปที่ 4.2

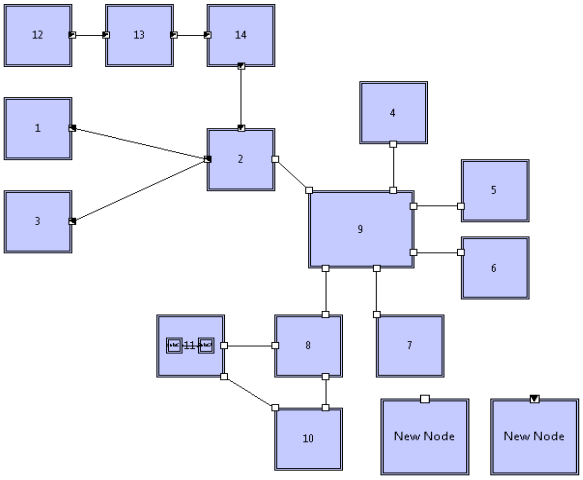


รูปที่ 4.2 แผนภาพกิจกรรมภาพรวมการดำเนินงานเครื่องมือตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์

จากรูปที่ 4.2 สามารถแสดงรายละเอียดขั้นตอนการดำเนินงานเครื่องมือตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์ ซึ่งแยกออกเป็นสามกลุ่มกิจกรรมดังนี้

4.1.1 กลุ่มกิจกรรมการเตรียมข้อมูลแผนภาพคอมโพเนนท์เป็นภาษาเอกซ์เอดีแอล

โดยข้อมูลดังกล่าวได้นำออกมาจากเครื่องมือ ArchStudio รุ่นที่ 4.0 เพื่อใช้เป็นข้อมูลนำเข้าสำหรับเครื่องมือตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์ ดังตัวอย่างการเตรียมข้อมูลแผนภาพที่เขียนขึ้นโดยใช้เครื่องมือ ArchStudio รุ่นที่ 4.0 เป็นภาษาเอกซ์เอดีแอลดังรูปที่ 4.3 โดยข้อมูลจะเชื่อมต่อกันระหว่าคอมโพเน้นท์ไม่ต้องผ่านคอนเนกเตอร์ เนื่องจากเอกซ์เอดีแอลสามารถเชื่อมต่อโดยใช้อินเตอร์เฟสได้และอินเตอร์เฟสเป็นการระบุช่องทางสำหรับการเชื่อมต่อสำหรับแต่ละคอมโพเนนท์



(ลักษณะของสถาปัตยกรรมที่ถูกสร้างโดยใช้เครื่องมือ ArchStudio4)

```
<types:component types:id="componentffffff8e-19b00853-2a59958f-e0bd0027" xsi:type="types:Component">
    <types:description xsi:type="instance:Description">Com2</types:description>
    <types:interface types:id="interfaceffffff8e-19b33c74-4f3f92e0-e0bd0473" xsi:type="types:Interface">
        <types:description xsi:type="instance:Description">[New Interface]</types:description>
        <types:direction xsi:type="instance:Direction">in</types:direction>
    </types:interface>
    <types:interface types:id="interfaceffffff8e-19b3dbbc-58177301-e0bd0517" xsi:type="types:Interface">
        <types:description xsi:type="instance:Description">[New Interface]</types:description>
        <types:direction xsi:type="instance:Direction">out</types:direction>
    </types:interface>
</types:component>
<types:component types:id="componentffffff8e-19b03d11-78735d83-e0bd0094" xsi:type="types:Component">
    <types:description xsi:type="instance:Description">Com3</types:description>
    <types:interface types:id="interfaceffffff8e-19b3423f-4358979b-e0bd0492" xsi:type="types:Interface">
        <types:description xsi:type="instance:Description">[New Interface]</types:description>
        <types:direction xsi:type="instance:Direction">in</types:direction>
    </types:interface>
    <types:interface types:id="interfaceffffff8e-19b354d8-dbc7e88e-e0bd0498" xsi:type="types:Interface">
        <types:description xsi:type="instance:Description">[New Interface]</types:description>
        <types:direction xsi:type="instance:Direction">out</types:direction>
    </types:interface>
```

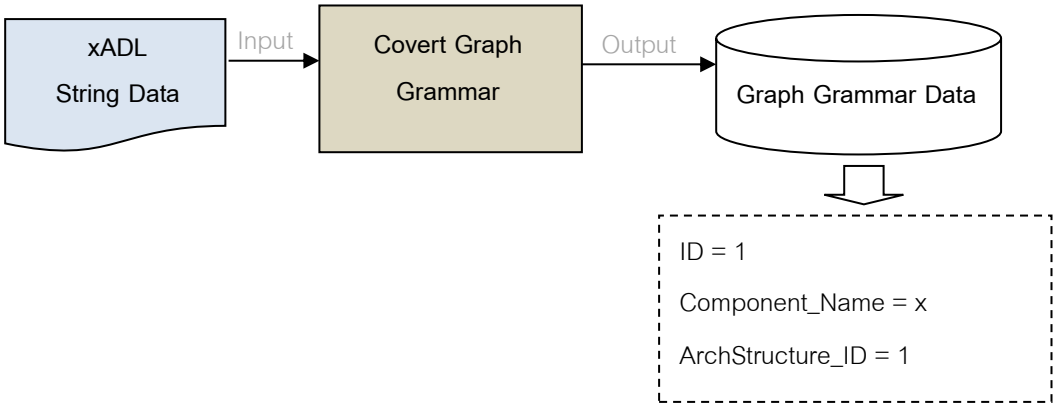
(ข้อมูลที่ได้หลังจากการบันทึกสถาปัตยกรรมที่เขียนลงในแฟ้มข้อมูล)

รูปที่ 4.3 ข้อมูลนำเข้าภาษาเอกซ์เอดีแอลแปลงจากแผนภาพสร้างโดยใช้เครื่องมือ ArchStudio4

4.1.2 กลุ่มกิจกรรมการตรวจสอบรูปแบบสถาปัตยกรรมซอฟต์แวร์ภาษาเอกซ์เอดีแอล

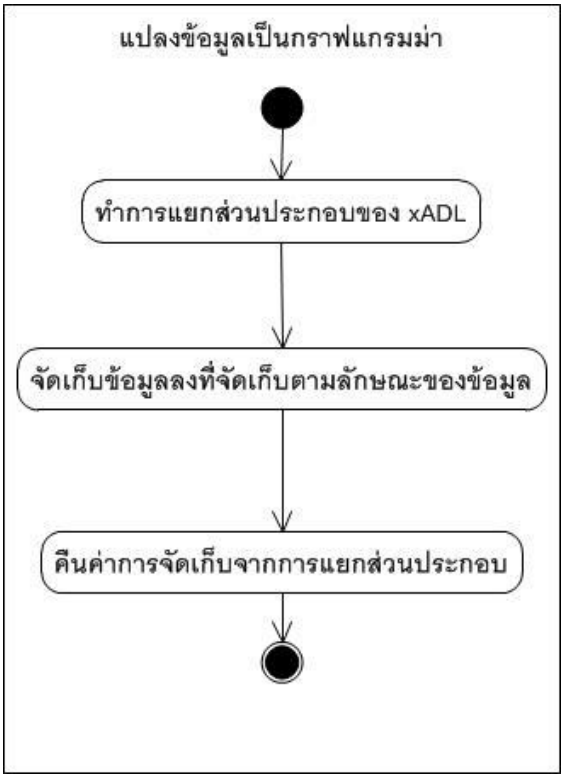
ซึ่งเป็นกลุ่มกิจกรรมสำคัญที่สุดสำหรับการดำเนินการวิจัย แบ่งเป็นกระบวนการย่อยสี่กระบวนการดังนี้

4.1.2.1 กระบวนการรับข้อมูลนำเข้ามาแปลงเป็นกราฟแกรมมา (Convert Graph Grammar) ในกระบวนการแรกหลังจากเริ่มต้นกระบวนการโดยการรับข้อมูลนำเข้าที่เป็นข้อมูลของภาษาเอกซ์เอดีแอล ที่เขียนโดยเขียนผ่านเครื่องมือ ArchStudio4 ให้อยู่ในรูปแบบของกราฟแกรมมา กล่าวคือจะมีการทำการแม็บปิ้งข้อมูลที่อยู่ในรูปภาษาเอกซ์เอดีแอลให้อยู่ในรูปของกราฟแกรมมาโดยอ้างอิงจากทฤษฎีของบทที่ 3 ซึ่งสามารถยกตัวอย่าง เช่น แท็ก types:archStructure ในภาษาเอกซ์เอดีแอลสามารถจับคู่ได้กับแผนภาพในกราฟแกรมมา และแท็ก types:component สามารถจับคู่ได้กับคอมโพเนนท์ในกราฟแกรมมา โดยกราฟแกรมมาที่ได้จะเป็นไปตามนิยามข้อ3 ในบทที่ 3 และผลลัพธ์จะถูกเก็บไว้ในตารางข้อมูลส่วนประกอบของกราฟแกรมมา เพื่อนำมาใช้ในขั้นตอนการการกรองข้อมูลต่อไป แสดงดังรูปที่ 4.4



รูปที่ 4.4 การรับข้อมูลนำเข้ามาแปลงเป็นกราฟแกรมมา

จากรูปที่ 4.4 สามารถแสดงแผนภาพกิจกรรมดังรูปที่ 4.5

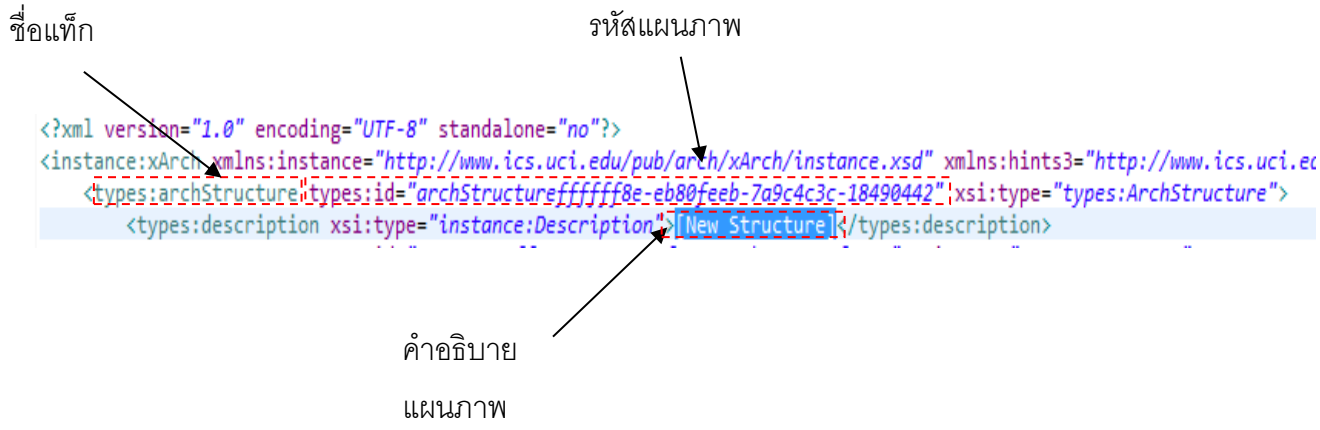


รูปที่ 4.5 แผนภาพกิจกรรมการแปลงข้อมูลเป็นกราฟแกรมมา

จากกระบวนการดังกล่าว สามารถแสดงตัวอย่างการรับข้อมูลนำเข้าแล้วทำการแยกส่วนประกอบ และเก็บข้อมูลลงตารางส่วนประกอบของกราฟแกรมมา ดังนี้

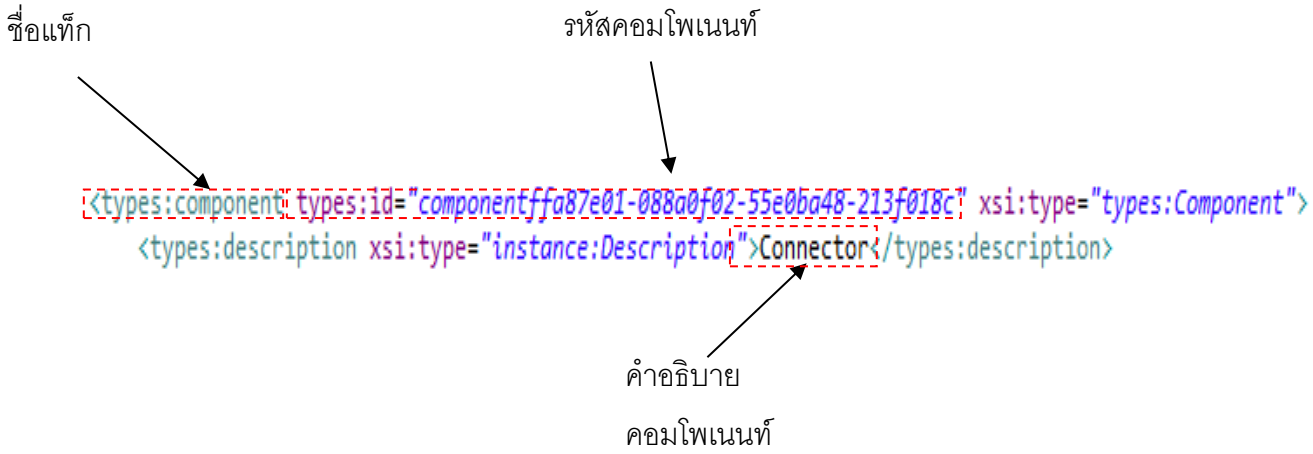
1) ทำการรับข้อมูลนำเข้าแล้วแยกออกมาเป็นส่วนประกอบหลัก ดังต่อไปนี้

ก) ส่วนแผนภาพ วิเคราะห์ส่วนของเอกสารเอกซ์เอดีแอลในส่วนของแผนภาพผ่านแท็ก types:archStructure ที่ซึ่งประกอบด้วยรหัสแผนภาพ "archStructureffffff8e-eb80feeb-7a9c4c3c-18490442" และคำอธิบายซึ่งอยู่ในแท็กลูกของ types:archStructure ซึ่งมีค่า "[New Structure]" ตามลำดับ แสดงดังรูปที่ 4.6



รูปที่ 4.6 ตัวอย่างภาษาเอกซ์เอดีแอลที่นำมาวิเคราะห์ในส่วนของแผนภาพ

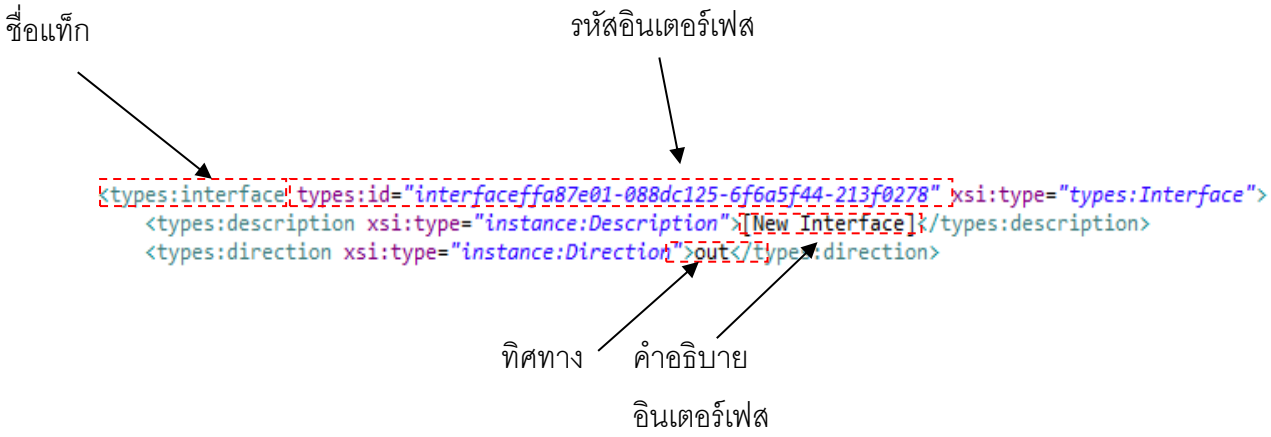
ข) ส่วนคอมโพเนนท์ วิเคราะห์ส่วนของเอกสารเอกซ์เอดีแอลในส่วนของคอมโพเนนท์ ผ่านแท็ก types:component ที่ซึ่งประกอบด้วยรหัสคอมโพเนนท์ "componentffa87e01-088a0f02-55e0ba48-213f018c" และคำอธิบายซึ่งอยู่ในแท็กลูกของ types:component ซึ่งมีค่า "Connector" ตามลำดับ แสดงดังรูปที่ 4.7 โดยในบางกรณีคอมโพเนนท์สามารถกำหนดชนิดของคอมโพเนนท์ตามที่ได้สร้างไว้ได้



รูปที่ 4.7 ตัวอย่างภาษาเอกซ์เอดีแอลที่นำมาวิเคราะห์ในส่วนของคอมโพเนนท์

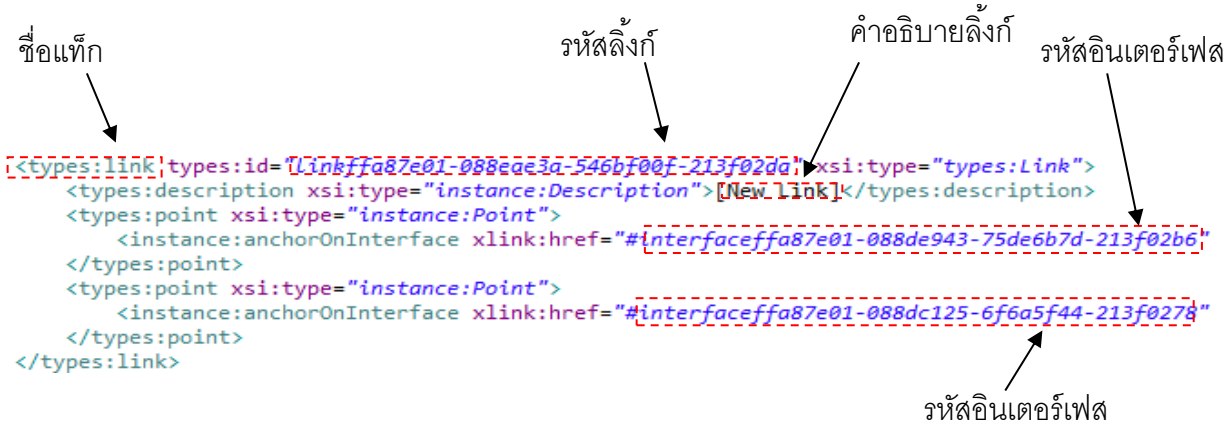
ค) ส่วนอินเตอร์เฟส วิเคราะห์ส่วนของเอกสารเอกซ์เอดีแอลในส่วนของอินเตอร์เฟส ผ่านแท็ก types:interface ที่ซึ่งประกอบด้วยรหัสอินเตอร์เฟส "interfaceffa87e01-088dc125-6f6a5f44-213f0278" คำอธิบายซึ่งอยู่ในแท็กลูกของ types:interface ซึ่งมีค่า " [New Interface]" และทิศทางซึ่งอยู่

ในแท็กลูกของ types:interface ซึ่งมีค่า "out" ตามลำดับ แสดงดังรูปที่ 4.8 โดยในบางกรณีอินเตอร์เฟสสามารถกำหนดชนิดของอินเตอร์เฟสตามที่ได้สร้างไว้ได้



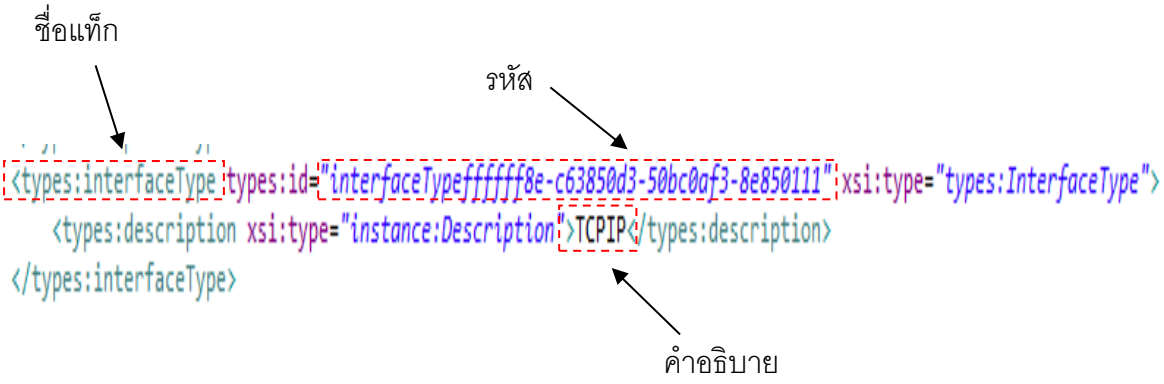
รูปที่ 4.8 ตัวอย่างภาษาเอกซ์เอดีแอลที่นำมาวิเคราะห์ในส่วนของอินเตอร์เฟส

ง) ส่วนลิ้งก์ วิเคราะห์ส่วนของเอกสารเอกซ์เอดีแอลในส่วนของลิ้งก์ ผ่านแท็ก types:link ที่ซึ่งประกอบด้วยรหัสลิ้งก์ "linkffa87e01-088ed1ee-250b1ce0-213f02e8" คำอธิบายซึ่งอยู่ในแท็กลูกของ types:link ซึ่งมีค่า " [New Link] " และรหัสของอินเตอร์เฟสหัวและท้าย คือ "interfaceffa87e01-088dd510-94c1a654-213f0297" และ "interfaceffa87e01-088dc125-6f6a5f44-213f0278" ตามลำดับ แสดงดังรูปที่ 4.9



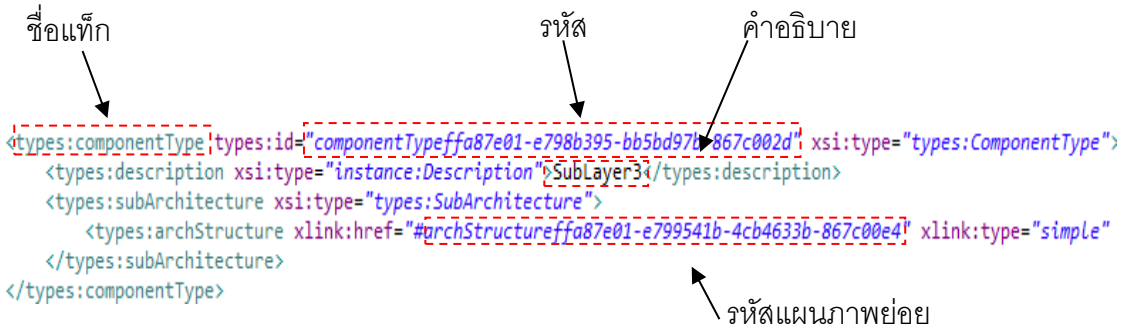
รูปที่ 4.9 ตัวอย่างภาษาเอกซ์เอดีแอลที่นำมาวิเคราะห์ในส่วนของลิ้งก์

จ) ส่วนอินเตอร์เฟสไทป์ วิเคราะห์ส่วนของเอกสารเอกซ์เอดีแอลในส่วนของอินเตอร์เฟสไทป์ ผ่านแท็ก types:interfaceType ที่ซึ่งประกอบด้วยรหัสอินเตอร์เฟสไทป์ "interfaceTypefffffff8e-c63850d3-50bc0af3-8e850111" คำอธิบายซึ่งอยู่ในแท็กลูกของ types:interfaceType ซึ่งมีค่า " TCPIP " ตามลำดับ แสดงดังรูปที่ 4.10



รูปที่ 4.10 ตัวอย่างภาษาเอกซ์เอดีแอลที่นำมาวิเคราะห์ในส่วนของอินเตอร์เฟสไทป์

ฉ) ส่วนคอมโพเนนท์ไทป์ วิเคราะห์ส่วนของเอกสารเอกซ์เอดีแอลในส่วนของคอมโพเนนท์ไทป์ ผ่านแท็ก types:componentType ที่ซึ่งประกอบด้วยรหัสคอมโพเนนท์ไทป์ "componentTypeffa87e01-e798b395-bb5bd97b-867c002d" คำอธิบายซึ่งอยู่ในแท็กลูกของ types:componentType ซึ่งมีค่า "SubLayer2" และยังประกอบด้วย Subarchitecture ผ่านแท็ก types:subArchitecture ซึ่งจะบอกถึงสถาปัตยกรรมที่ซ้อนอยู่โดยจะเชื่อมโยงรหัสแผนภาพตามลำดับ แสดงดังรูปที่ 4.11



รูปที่ 4.11 ตัวอย่างภาษาเอกซ์เอดีแอลที่นำมาวิเคราะห์ในส่วนของคอมโพเนนท์ไทป์

2) จัดเก็บส่วนประกอบหลักลงในตารางที่แยกตามชื่อเพื่อนำมาทำการตรวจสอบสถาปัตยกรรมต่อไป ซึ่งจะมีตัวอย่างข้อมูลดังตารางที่ 4.1 - ตารางที่ 4.5

ตารางที่ 4.1 ข้อมูลตัวอย่างที่เก็บลงในตาราง Component

คุณลักษณะ	รายละเอียด	ค่าข้อมูลตัวอย่าง
Component_ID	ค่าคีย์ของคลาส Component	001
Description	คำอธิบาย	Connector
ArchStructure_ID	ค่าคีย์ของคลาส Architecture	001
SubArchitect_ID	ค่าคีย์ของคลาส Architecture สำหรับระบุว่า Component นี้มีคุณสมบัติ Sub Architecture	
Component_Name	ชื่อของ Component	Connector
selected	สถานะของ Component	

ตารางที่ 4.2 ข้อมูลตัวอย่างที่เก็บลงในตาราง Interface

คุณลักษณะ	รายละเอียด	ค่าข้อมูลตัวอย่าง
Interface_ID	ค่าคีย์ของคลาส Interface	001
Description	คำอธิบาย	[New Interface]
direction	ทิศทางของ Interface	out
Component_ID	ค่าคีย์ของคลาส Component	001
interfacename	ชื่อของ Interface	[New Interface]
ArchType_ID	ค่าคีย์ของคลาส ArchType	

ตารางที่ 4.3 ข้อมูลตัวอย่างที่เก็บลงในตาราง Link

คุณลักษณะ	รายละเอียด	ค่าข้อมูลตัวอย่าง
Link_ID	ค่าคีย์ของคลาส Link	001
EndPoint_ID	ค่าคีย์ของคลาส Interface ปลายทาง	001
StartPoint_ID	ค่าคีย์ของคลาส Interface ต้นทาง	002
Description	คำอธิบาย	[New Link]
ArchStructure_ID	ค่าคีย์ของคลาส Architecture	001
LinkName	ชื่อของ Link	[New Link]

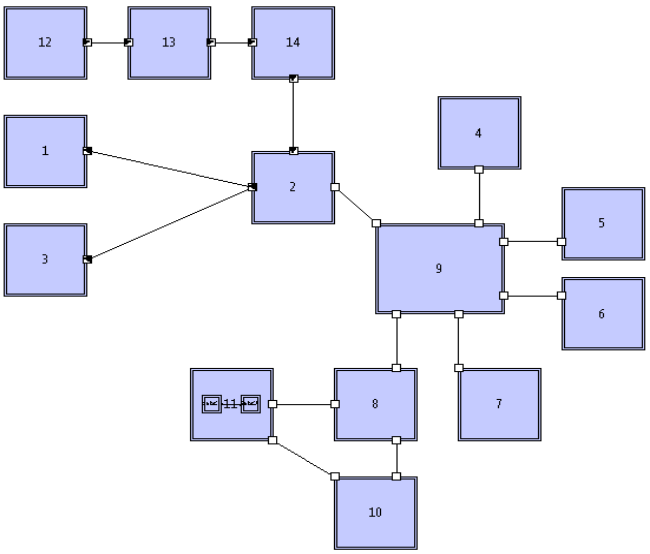
ตารางที่ 4.4 ข้อมูลตัวอย่างที่เก็บลงในตาราง Architecture

คุณลักษณะ	รายละเอียด	ค่าข้อมูลตัวอย่าง
Architecture_ID	ค่าคีย์ของคลาส Architecture	001
Description	คำอธิบาย	[New Structure]
Structure_ID	ค่า Structure	archStructureffffff8e-eb80feeb-7a9c4c3c-18490442

ตารางที่ 4.5 ข้อมูลตัวอย่างที่เก็บลงในตาราง ArchType

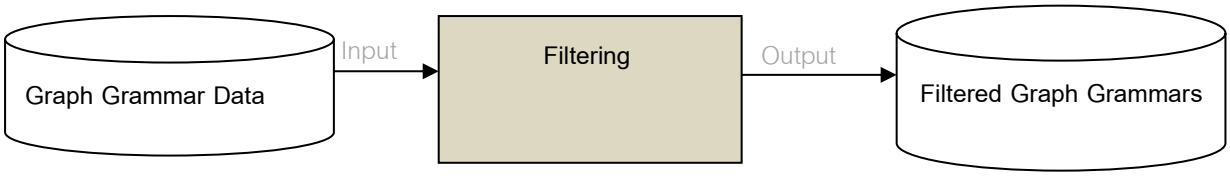
คุณลักษณะ	รายละเอียด	ค่าข้อมูลตัวอย่าง
ArchType_ID	ค่าคีย์ของคลาส ArchType	001
ArchType_Description	คำอธิบาย	TCPIP
Type	ชนิดของ Type	interfaceType
SubstructureType	ค่าคีย์ของคลาส Architecture ถ้า Type มีคุณสมบัติ Sub Structure	
ArchType_Name	ชื่อ ArchType	TCPIP

4.1.2.2 กระบวนการกรองข้อมูลที่ไม่เกี่ยวข้องออกจากกราฟแกรมมา (Filtering) ในกระบวนการที่สองเป็นการกรองข้อมูลที่ไม่เกี่ยวข้องออกจากกราฟแกรมมาโดยดูจากความสัมพันธ์ของกราฟ โดยถ้ากราฟใดไม่มีความสัมพันธ์กับระบบหลัก หรือทิศทางการเชื่อมต่อผิดเพี้ยน จะถูกกรองออกจากกราฟแกรมมาดังรูปที่ 4.12



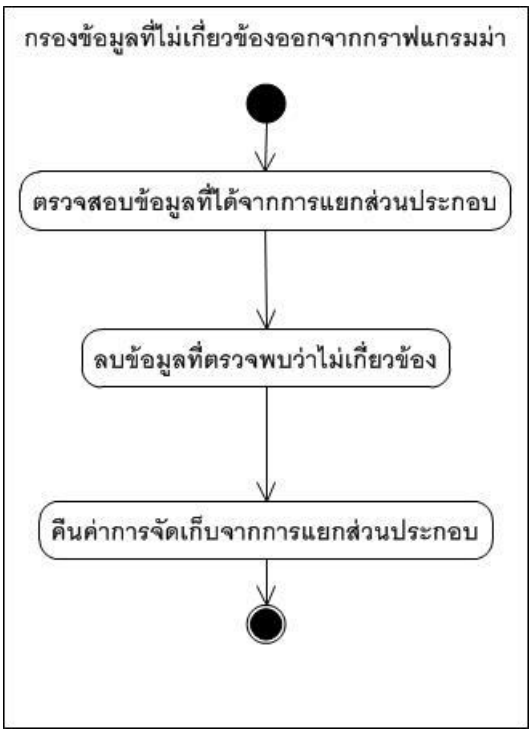
รูปที่ 4.12 ตัวอย่างผลลัพธ์จากการกรองข้อมูลออกจากกราฟแกรมมา

วิธีการกรองข้อมูลนั้นจะทำการกรองข้อมูลโดยดูจากตารางที่ได้จากกระบวนการแปลงภาษาเอกซ์เอดีแอลเป็นกราฟแกรมมา ซึ่งหลักการในการกรองข้อมูลจะทำโดยการค้นหาอินเตอร์เฟสที่ไม่มีลิ้งก์เชื่อมถึงในตารางลิ้งก์และตารางอินเตอร์เฟส เมื่อตรวจพบก็จะมีการตรวจสอบอีกว่าในคอมโพเนนท์นั้นมีอินเตอร์เฟสที่เชื่อมต่อกับส่วนอื่นอีกหรือไม่ ถ้าตรวจไม่พบก็จะทำการลบข้อมูลคอมโพเนนท์และอินเตอร์เฟสทั้งหมดที่เกี่ยวข้องกับคอมโพเนนท์นั้นออกจากตารางคอมโพเนนท์และตารางอินเตอร์เฟส แสดงดังรูปที่ 4.13



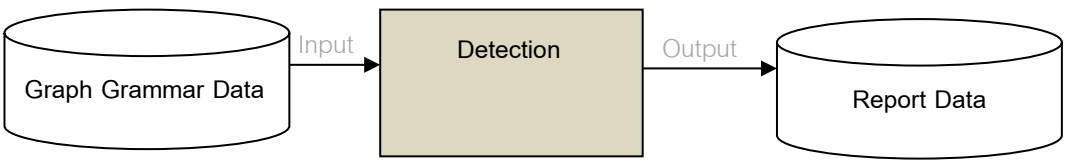
รูปที่ 4.13 การรับข้อมูลตารางที่เกี่ยวข้องเพื่อทำการกรองข้อมูล

จากรูปที่ 4.13 สามารถแสดงแผนภาพกิจกรรมดังรูปที่ 4.14



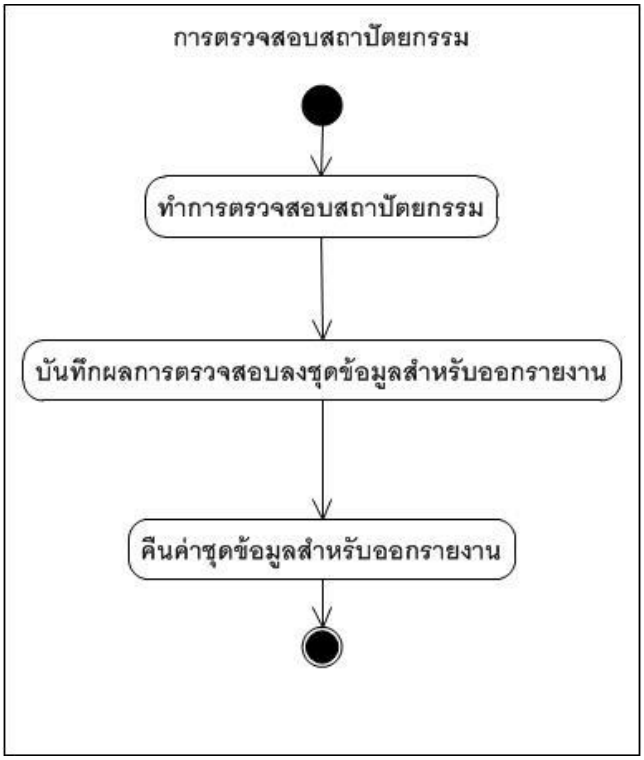
รูปที่ 4.14 แผนภาพกิจกรรมการกรองข้อมูลที่ไม่เกี่ยวข้องออกจากกราฟแกรมมา

4.1.2.3 กระบวนการตรวจสอบสถาปัตยกรรม (Detection) ในกระบวนการที่สามเป็นการตรวจสอบสถาปัตยกรรมโดยการตรวจจับจะเริ่มจากรูปแบบสถาปัตยกรรมที่ 1 - 4 โดยจะมีการรับตารางข้อมูลส่วนประกอบของกราฟแกรมมาที่ผ่านการกรองข้อมูลเรียบร้อยแล้ว และผลลัพธ์ของการตรวจสอบสถาปัตยกรรมจะถูกเก็บลงในตารางรายงานเพื่อที่จะนำไปใช้ในการออกรายงานในขั้นตอนการเปรียบเทียบต่อไป แสดงดังรูปที่ 4.15



รูปที่ 4.15 การรับข้อมูลตารางที่เกี่ยวข้องเพื่อทำการตรวจจับสถาปัตยกรรมซอฟต์แวร์

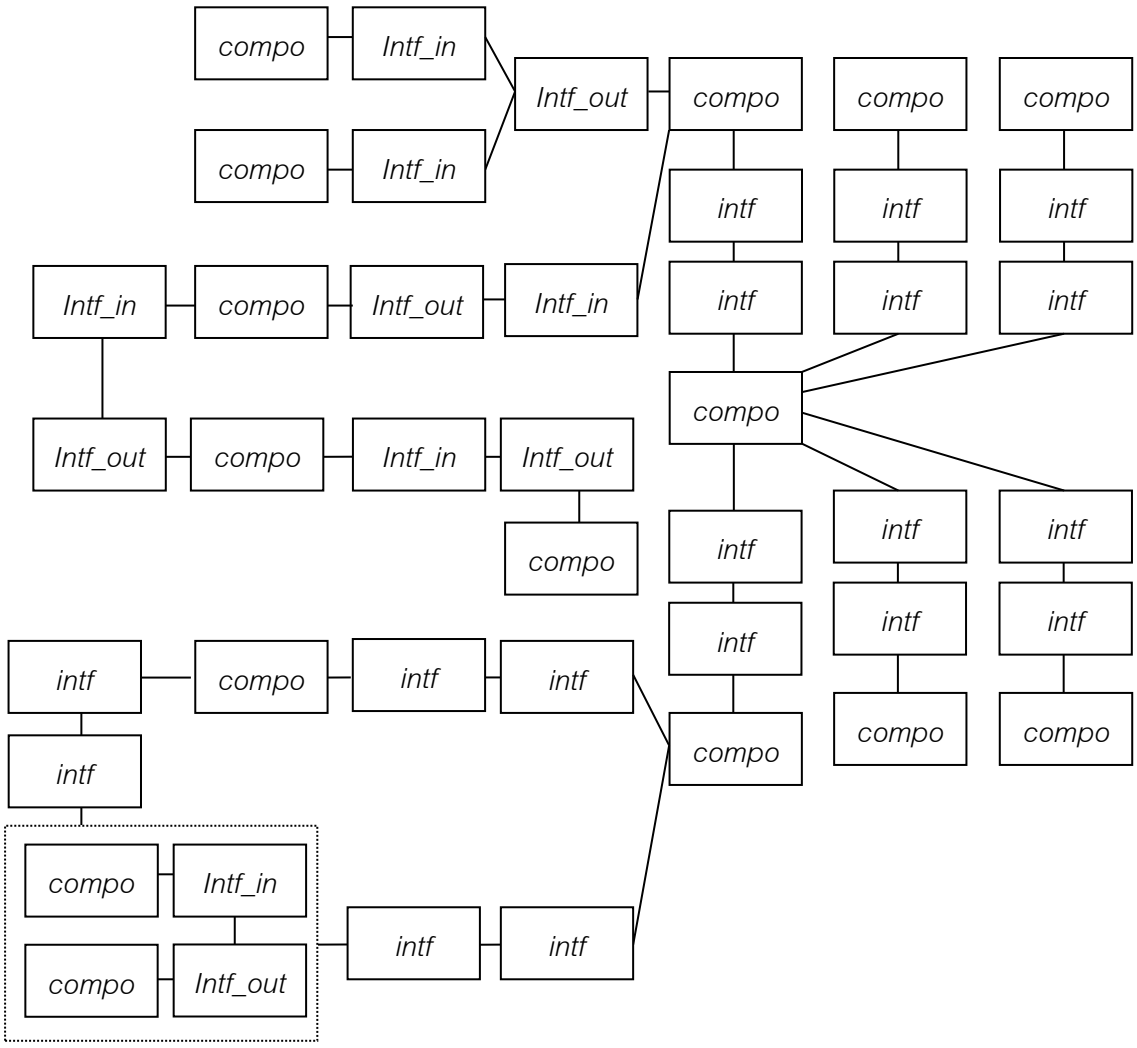
จากรูปที่ 4.15 สามารถแสดงแผนภาพกิจกรรมดังรูปที่ 4.16



รูปที่ 4.16 แผนภาพกิจกรรมการตรวจสอบสถาปัตยกรรม

จากกระบวนการดังกล่าว สามารถแสดงตัวอย่างการตรวจสอบสถาปัตยกรรม และเก็บข้อมูลลงตารางสำหรับออกรายงาน จากรูปที่ 4.12 ดังนี้

1) จากข้อมูลในตารางข้อมูลส่วนประกอบของกราฟแกรมมา สามารถนำความสัมพันธ์ของแต่ละตารางในส่วนประกอบของกราฟแกรมมามาสร้างเป็นรูปโดยอ้างอิงจากนิยามข้อ 3 ในบทที่ 3 ได้ โดยจากตัวอย่างในรูปที่ 4.12 สามารถนำมาสร้างเป็นรูปโดยอ้างอิงจากนิยามข้อ 3 ในบทที่ 3 ได้ดังรูปที่ 4.17

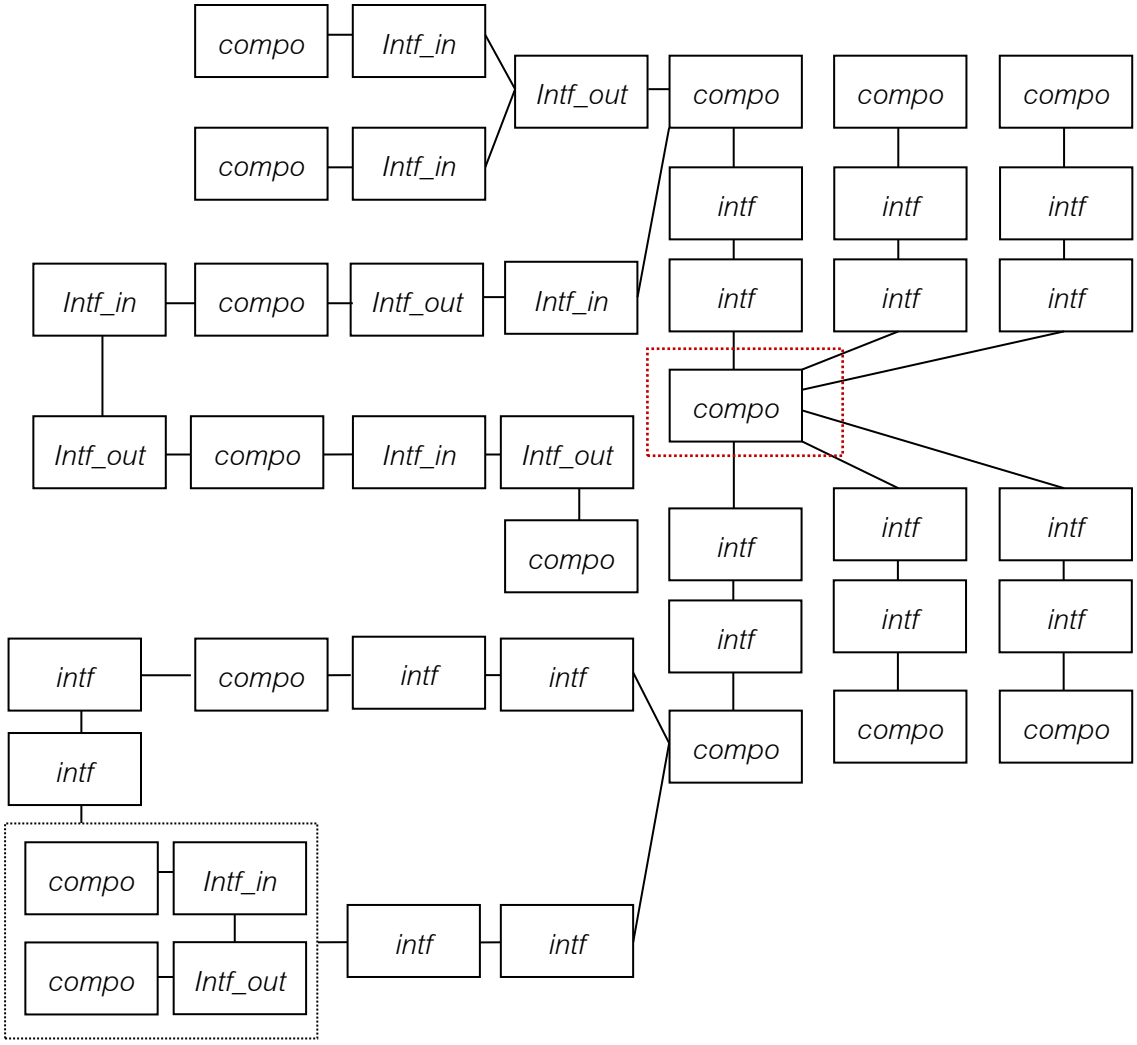


รูปที่ 4.17 ตัวอย่างกราฟแกรมมาที่ได้จากการแปลงต้นฉบับในรูปที่ 4.12

2) จากความสัมพันธ์ของแต่ละตารางในส่วนประกอบของกราฟแกรมมาสามารถสร้างเป็นรูปโดยอ้างอิงจากนิยามข้อ 3 ในบทที่ 3 ได้ ซึ่งจากรูปเริ่มต้นในรูปที่ 4.17 สามารถจำลองการตรวจสอบสถาปัตยกรรมในแต่ละรูปแบบได้ดังนี้

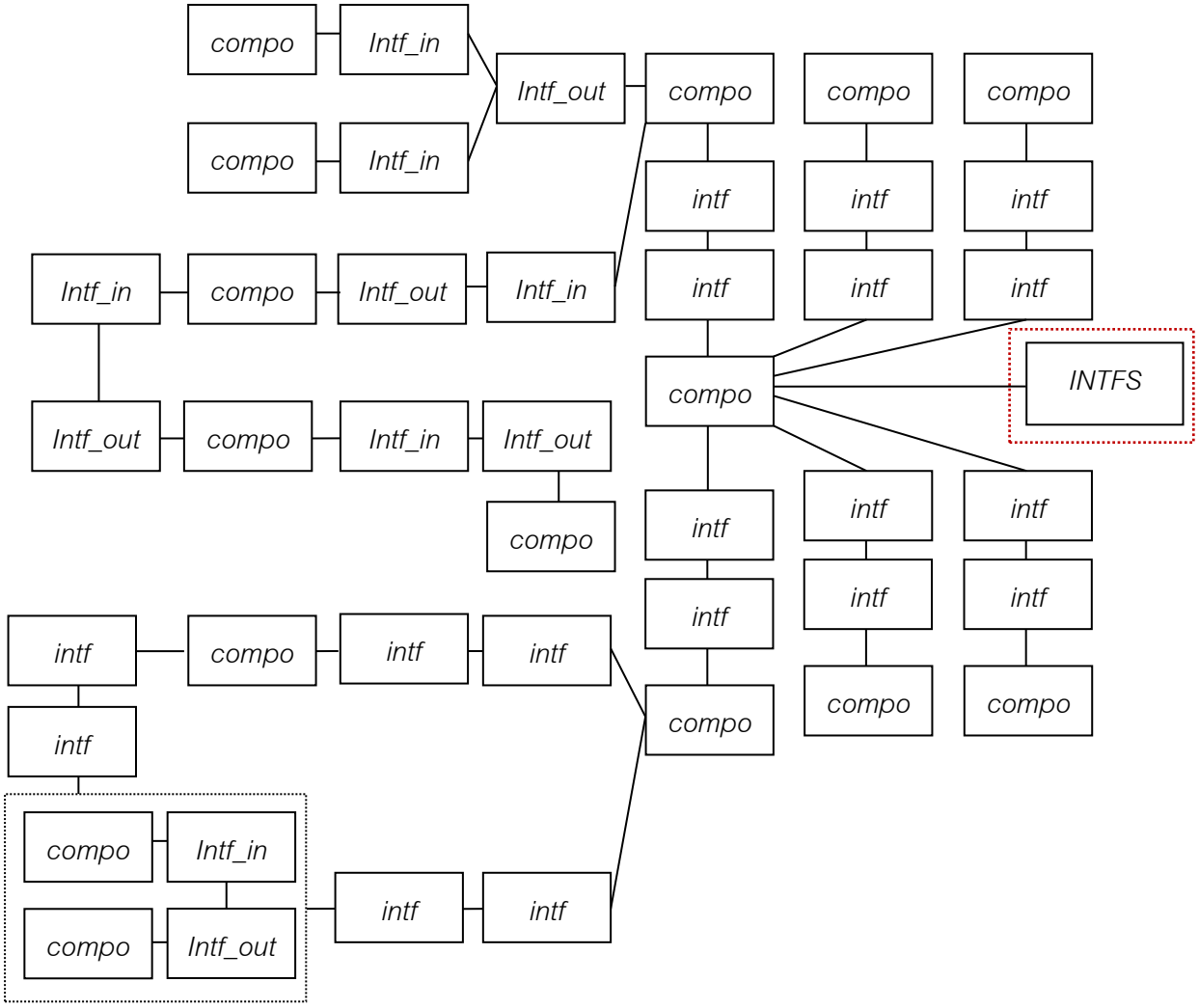
ก) การตรวจสอบสถาปัตยกรรมในรูปแบบรวมศูนย์กลาง

การตรวจจับรูปแบบสถาปัตยกรรมในรูปแบบรวมศูนย์กลางจะเริ่มจากนิยามที่ 7 ในบทที่ 3 เพื่อหาตำแหน่งเริ่มต้นและส่วนประกอบที่ค้นพบ ซึ่งจะได้รูปบัพเริ่มต้นดังรูปที่ 4.18



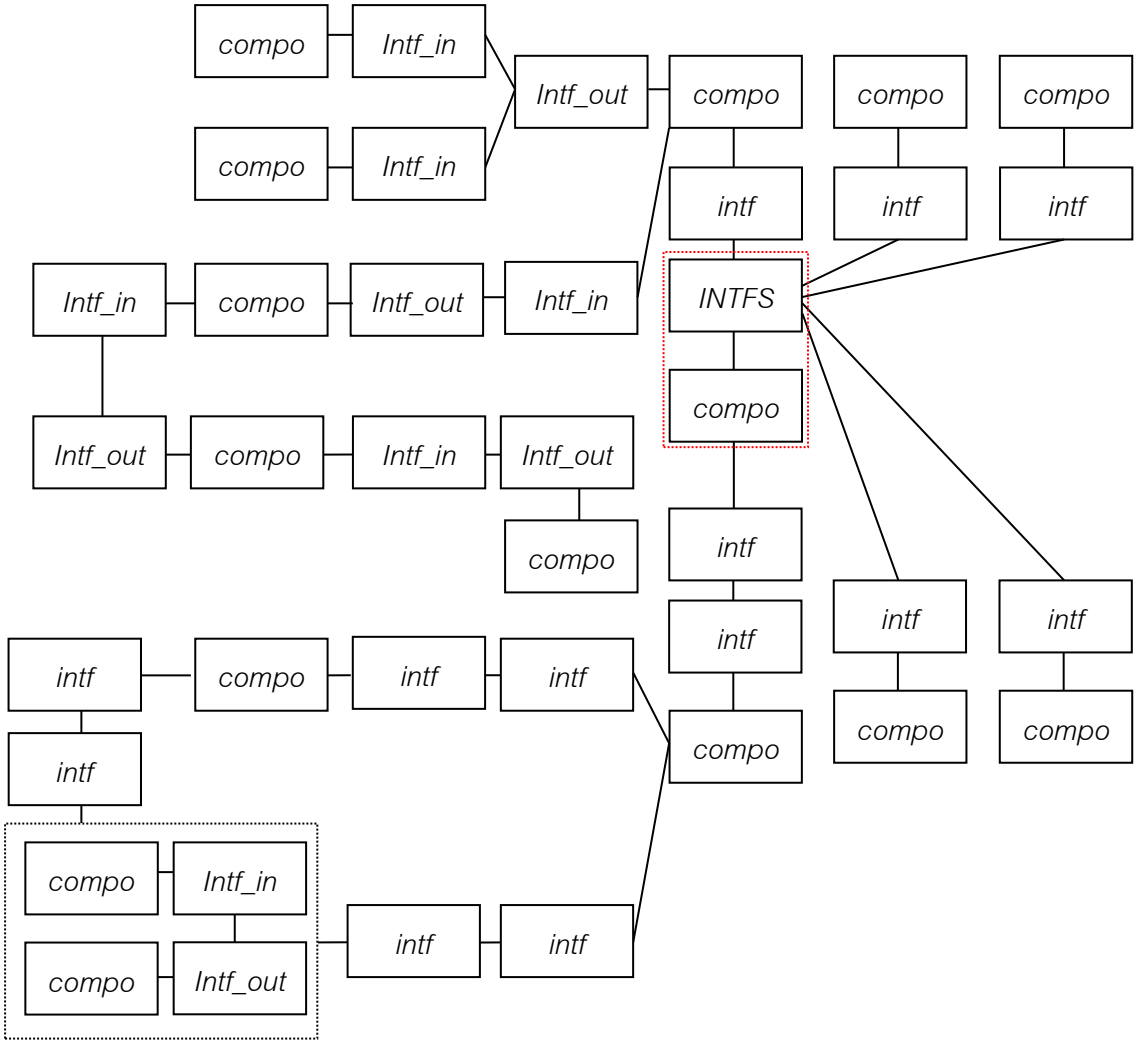
รูปที่ 4.18 ตัวอย่างข้อมูลนำเข้าในรูปที่ 4.12 ที่แปลงเป็นกราฟและทำการเลือกบัพเริ่มต้นแล้ว

จากนิยามข้อ 7 ในบทที่ 3 หลังจากทำการเลือกบัพเริ่มต้นแล้ว ใช้กฎข้อ P7 แต่เนื่องจากเราใช้วิธีการตรวจสอบโดยใช้กฎการลดรูป ดังนั้นกฎที่จะเอามาใช้จะเขียนแทนด้วย P7’ ซึ่งจะใช้ในการเปลี่ยนรูป ซึ่งจะได้ผลดังรูปที่ 4.19



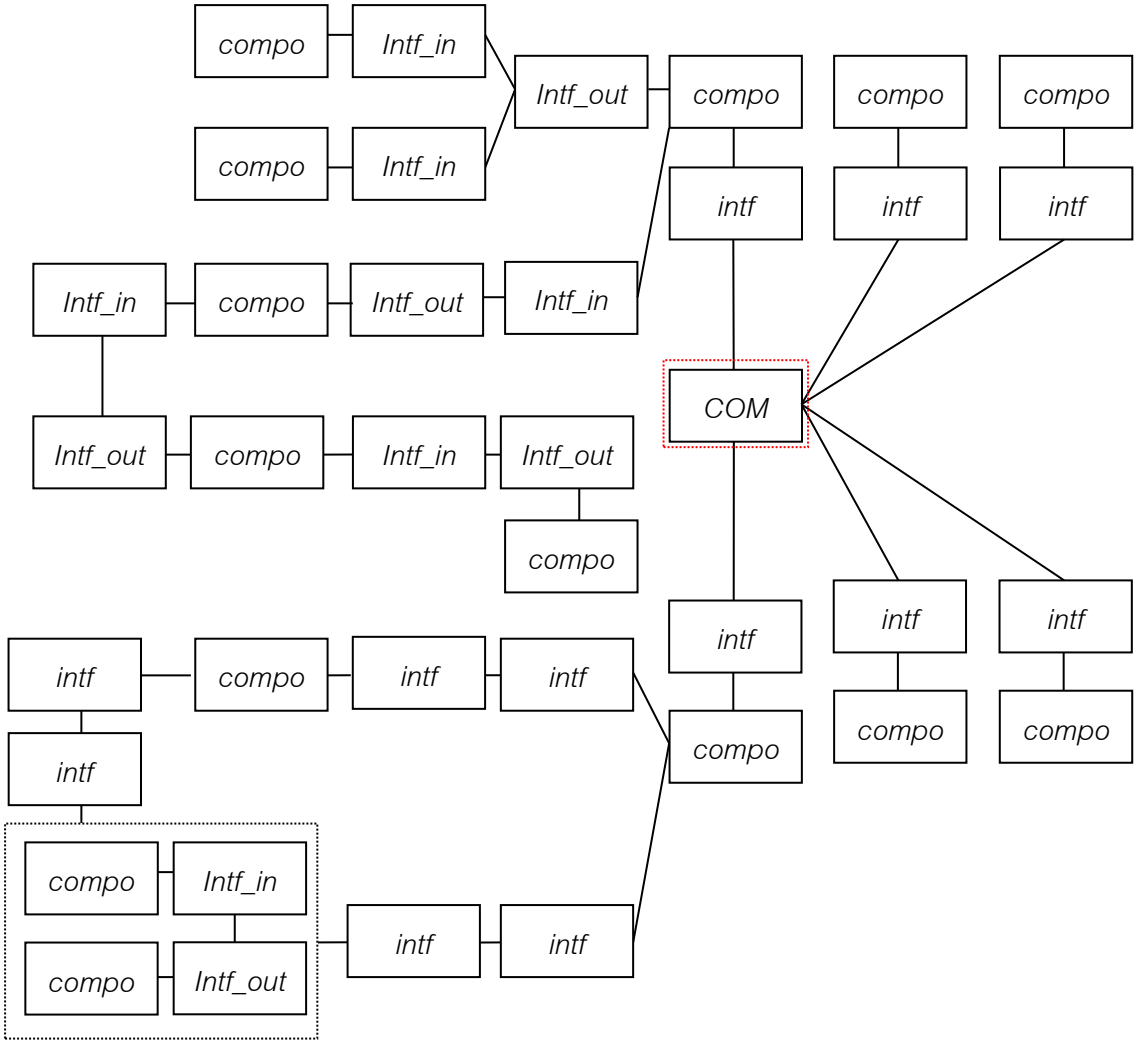
รูปที่ 4.19 ตัวอย่างข้อมูลนำเข้าในรูปที่ 4.12 ที่แปลงเป็นกราฟแล้วและทำการใช้กฎ P7' แล้ว

จากนั้นใช้นิยามข้อ 7 ในบทที่ 3 เราสามารถใช้กฎข้อ P6’ สี่ครั้งเพื่อที่จะเปลี่ยนรูปร่างของกราฟเพื่อทำการรวม intf เข้ากับ INTFS โดยจะมีลักษณะคล้ายกับตัวอย่างที่อยู่ในนิยามข้อ 7 ซึ่งได้ผลลัพธ์ดังรูปที่ 4.20



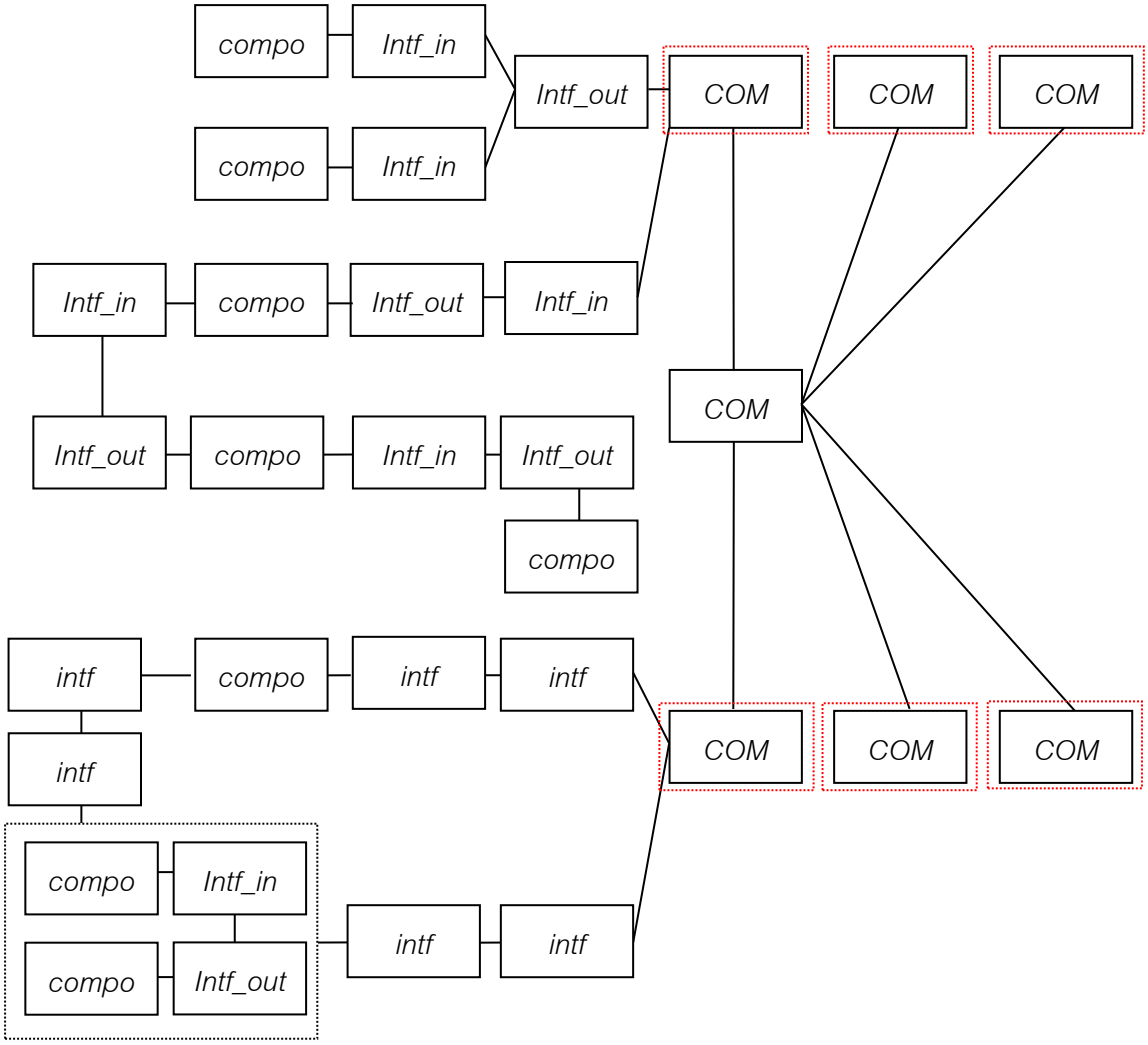
รูปที่ 4.20 ตัวอย่างข้อมูลนำเข้าที่แปลงเป็นกราฟแล้วและทำการใช้กฎ P6' แล้ว

จากนั้นใช้นิยามข้อ 7 ในบทที่ 3 เราสามารถใช้กฎ P5’ เพื่อที่จะเปลี่ยนรูปร่างของกราฟโดยการสร้าง COM ขึ้นมา ซึ่งได้ผลลัพธ์ดังรูปที่ 4.21



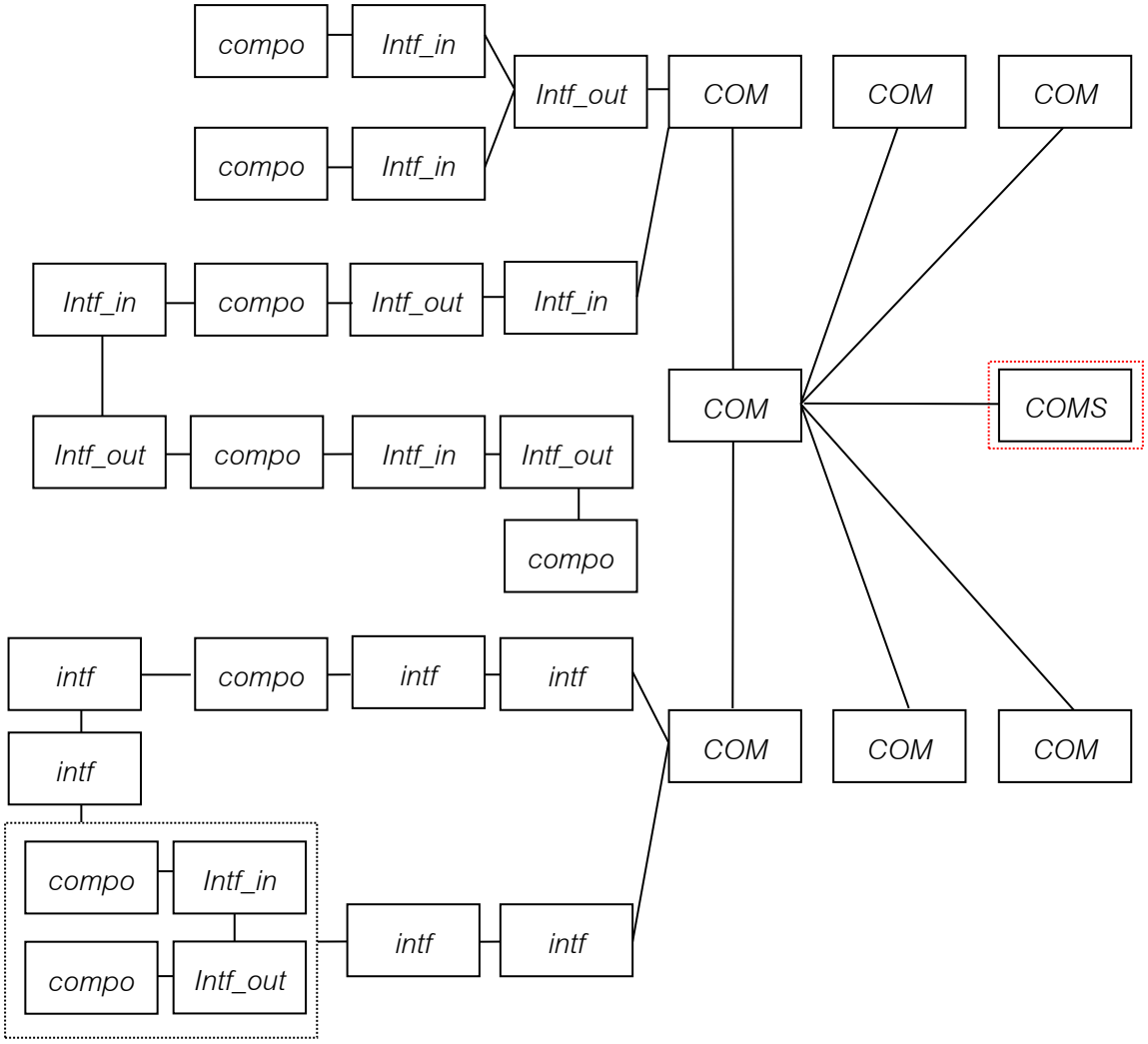
รูปที่ 4.21 ตัวอย่างข้อมูลนำเข้าที่แปลงเป็นกราฟแล้วและทำการใช้กฎ P5' แล้ว

จากนั้นใช้นิยามข้อ 7 ในบทที่ 3 เราสามารถใช้กฎ P4’ เพื่อที่จะเปลี่ยนรูปร่างของกราฟโดยการเปลี่ยน compo ที่เชื่อมโยงกับ COM ให้เป็น COM ทั้งหมด ซึ่งจะได้ผลลัพธ์ดังรูปที่ 4.22



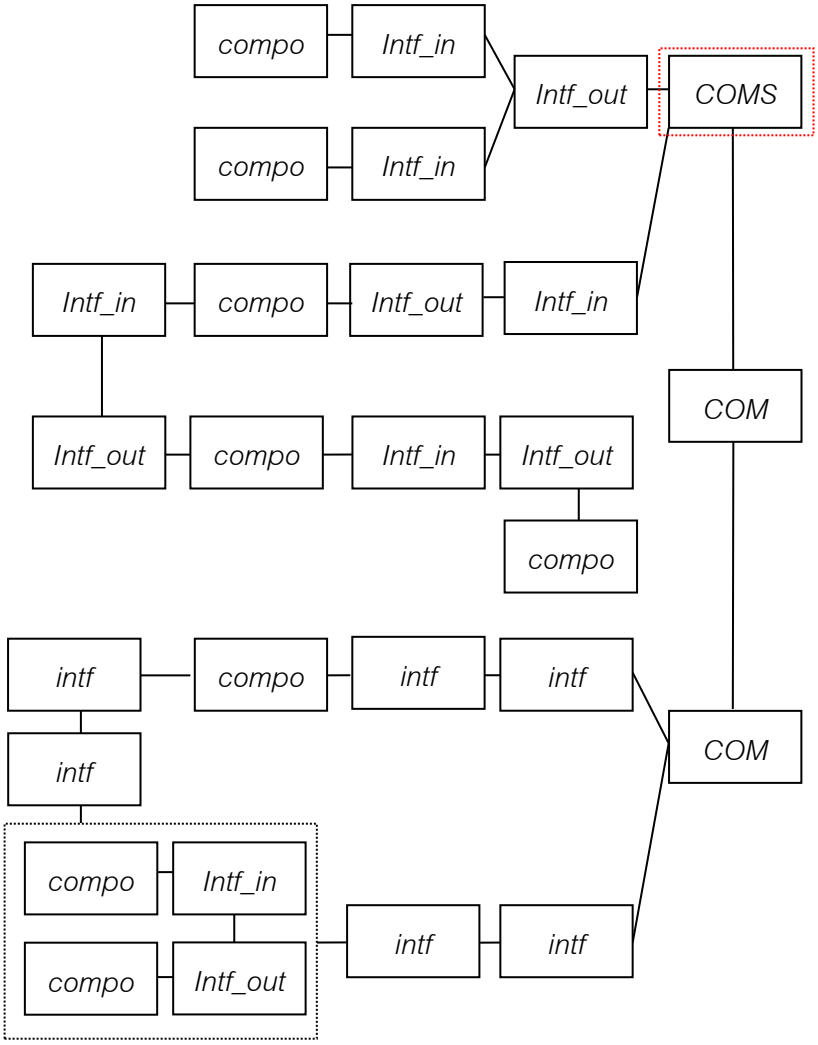
รูปที่ 4.22 ตัวอย่างข้อมูลนำเข้าที่แปลงเป็นกราฟแล้วและทำการใช้กฎ P4' แล้ว

จากนั้นใช้นิยามข้อ 7 ในบทที่ 3 เราสามารถใช้กฎ P3’ เพื่อที่จะเปลี่ยนรูปร่างของกราฟเพื่อสร้าง COMS ที่ใช้ในการรวม COM ที่ไม่ได้เชื่อมต่อกันเองมาไว้ใน COM เนื่องจาก COM ดังกล่าวมีคุณลักษณะของรูปแบบสถาปัตยกรรมรวมศูนย์กลาง ดังรูปที่ 4.23



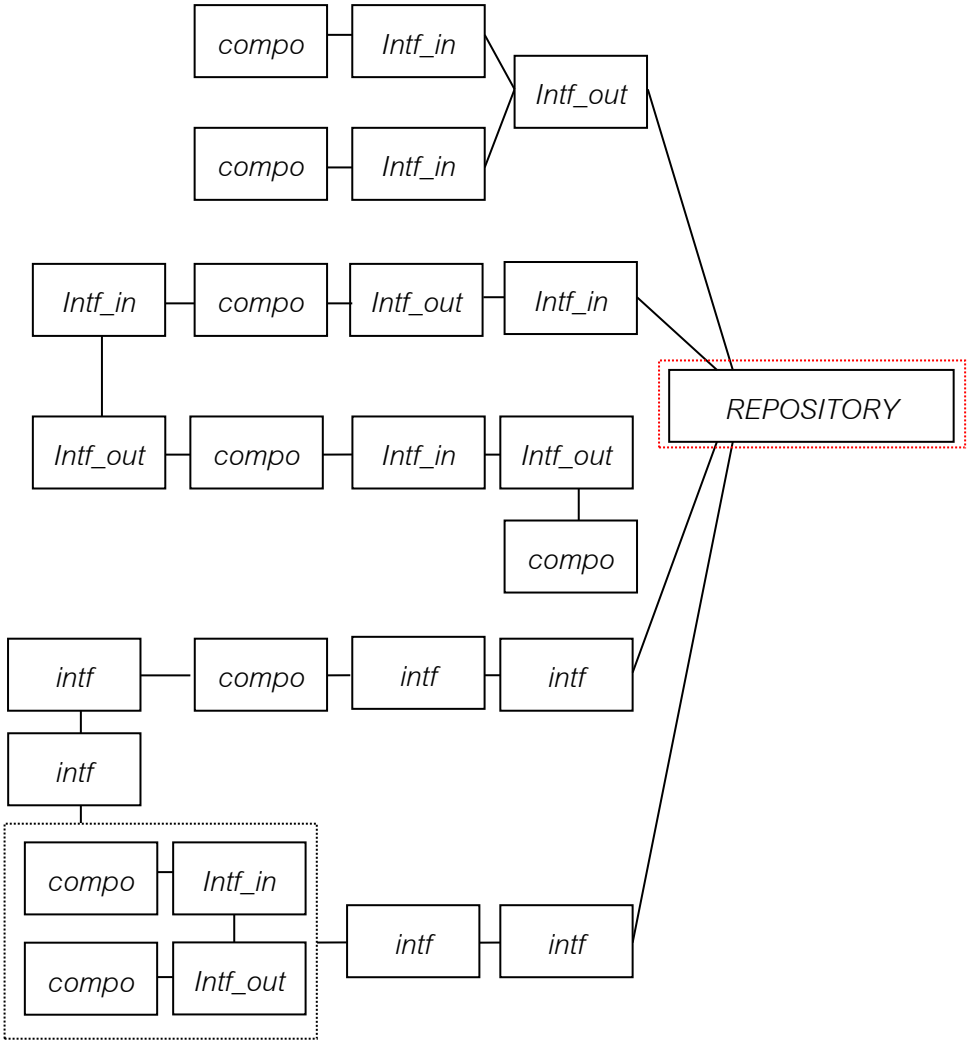
รูปที่ 4.23 ตัวอย่างข้อมูลนำเข้าในรูปที่ 4.12 ที่แปลงเป็นกราฟและทำการใช้กฎ P3' แล้ว

จากนั้นใช้นิยามข้อ 7 ในบทที่ 3 เราสามารถใช้กฎ P2’ เพื่อที่จะเปลี่ยนรูปร่างของกราฟเพื่อรวม COM ที่มีคุณลักษณะของรูปแบบสถาปัตยกรรมรวมศูนย์กลางไว้ใน COMS ดังรูปที่ 4.24



รูปที่ 4.24 ตัวอย่างข้อมูลนำเข้าในรูปที่ 4.12 ที่แปลงเป็นกราฟแล้วและทำการใช้กฎ P2' แล้ว

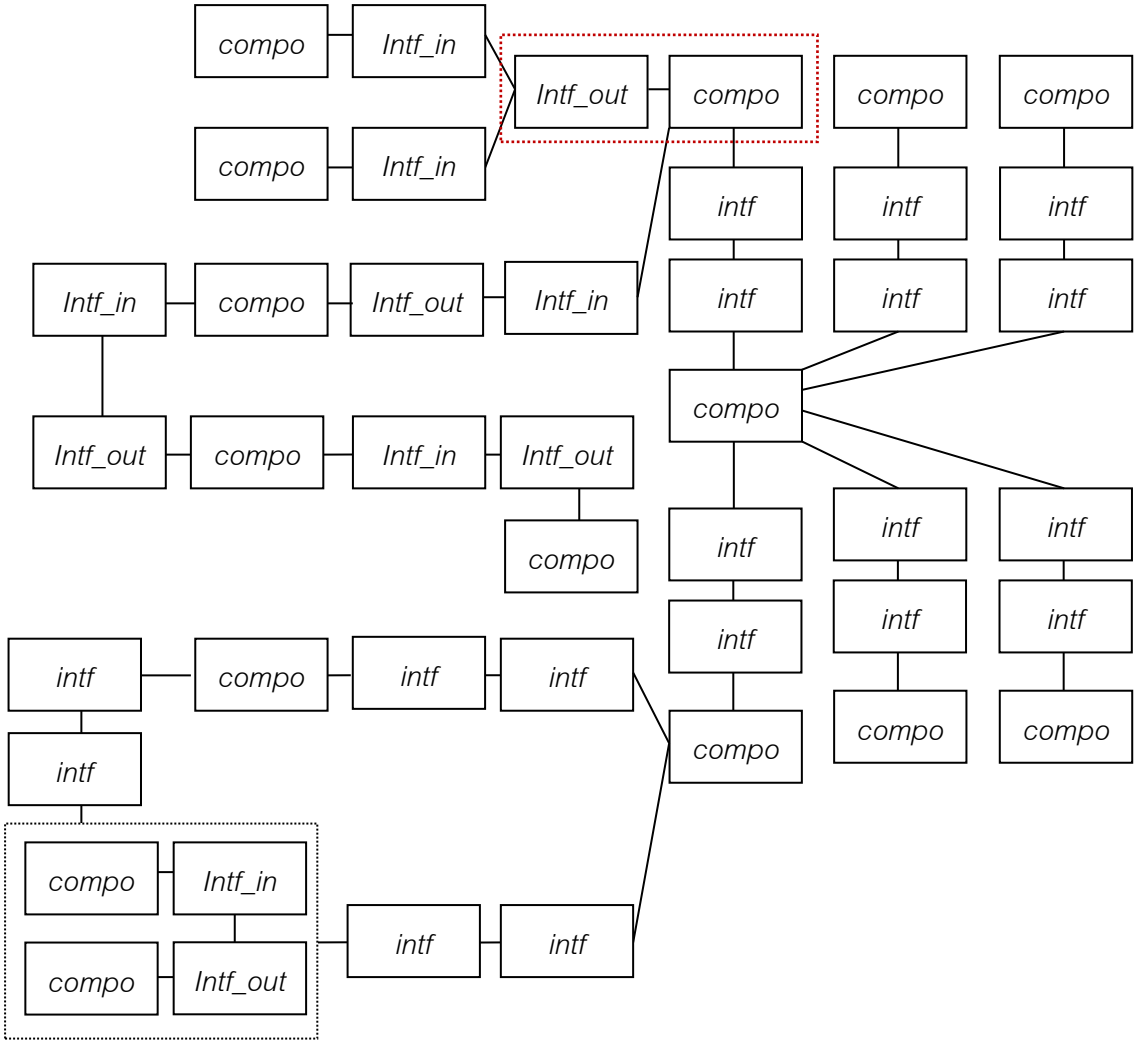
จากนั้นใช้นิยามข้อ 7 ในบทที่ 3 เราสามารถใช้กฎ P1’ เพื่อที่จะยืนยันว่าเราสามารถค้นพบคุณสมบัติของสถาปัตยกรรมในรูปแบบรวมศูนย์กลางได้ดังรูปที่ 4.25



รูปที่ 4.25 ตัวอย่างข้อมูลนำเข้าในรูปที่ 4.12 ที่แปลงเป็นกราฟแล้วและทำการใช้กฎ P1' แล้ว

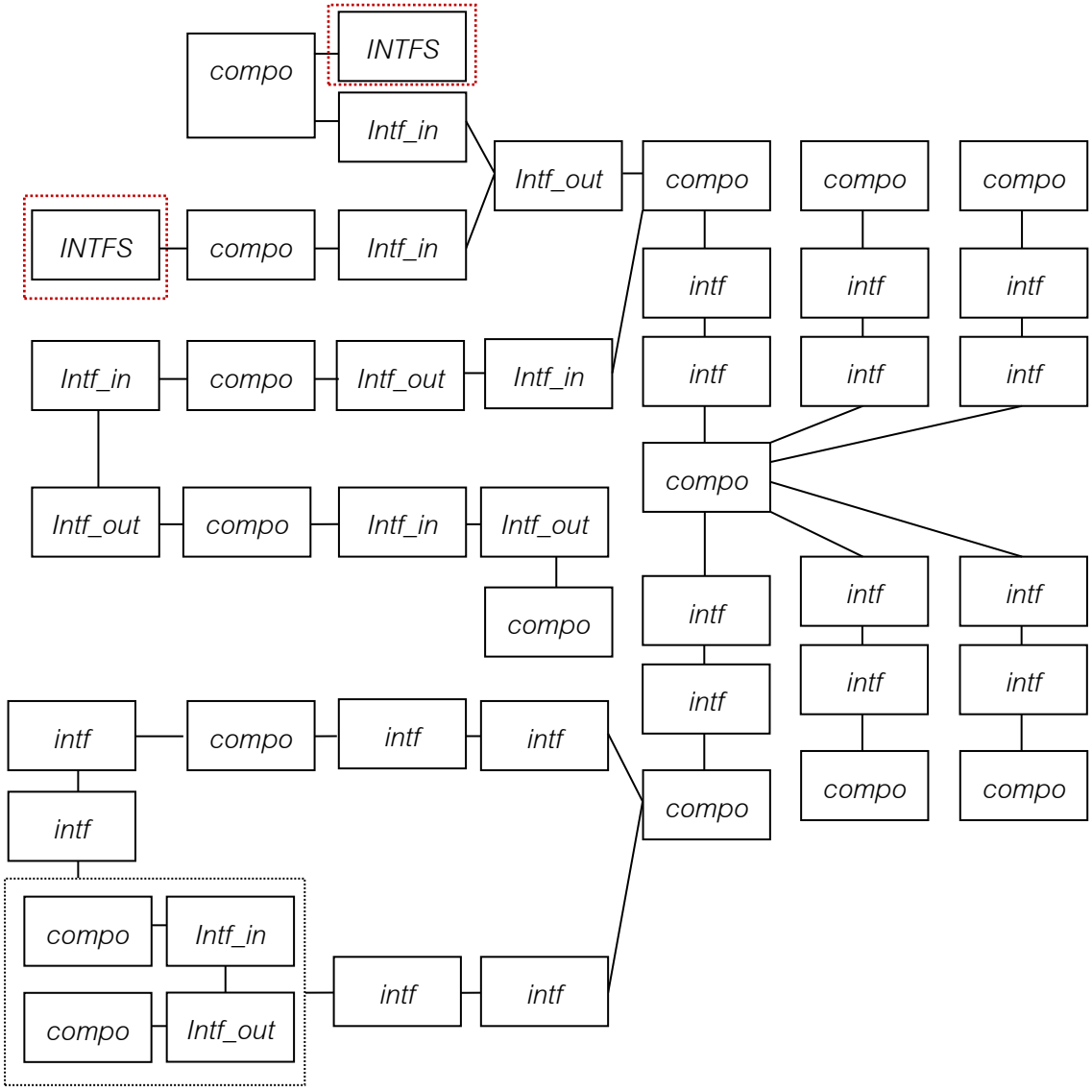
ข) การตรวจสอบสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้น

จากนิยามข้อ 8 ในบทที่ 3 หลังจากทำการเลือกบัพเริ่มต้นแล้ว ซึ่งจะได้บัพเริ่มต้นดังรูปที่ 4.26



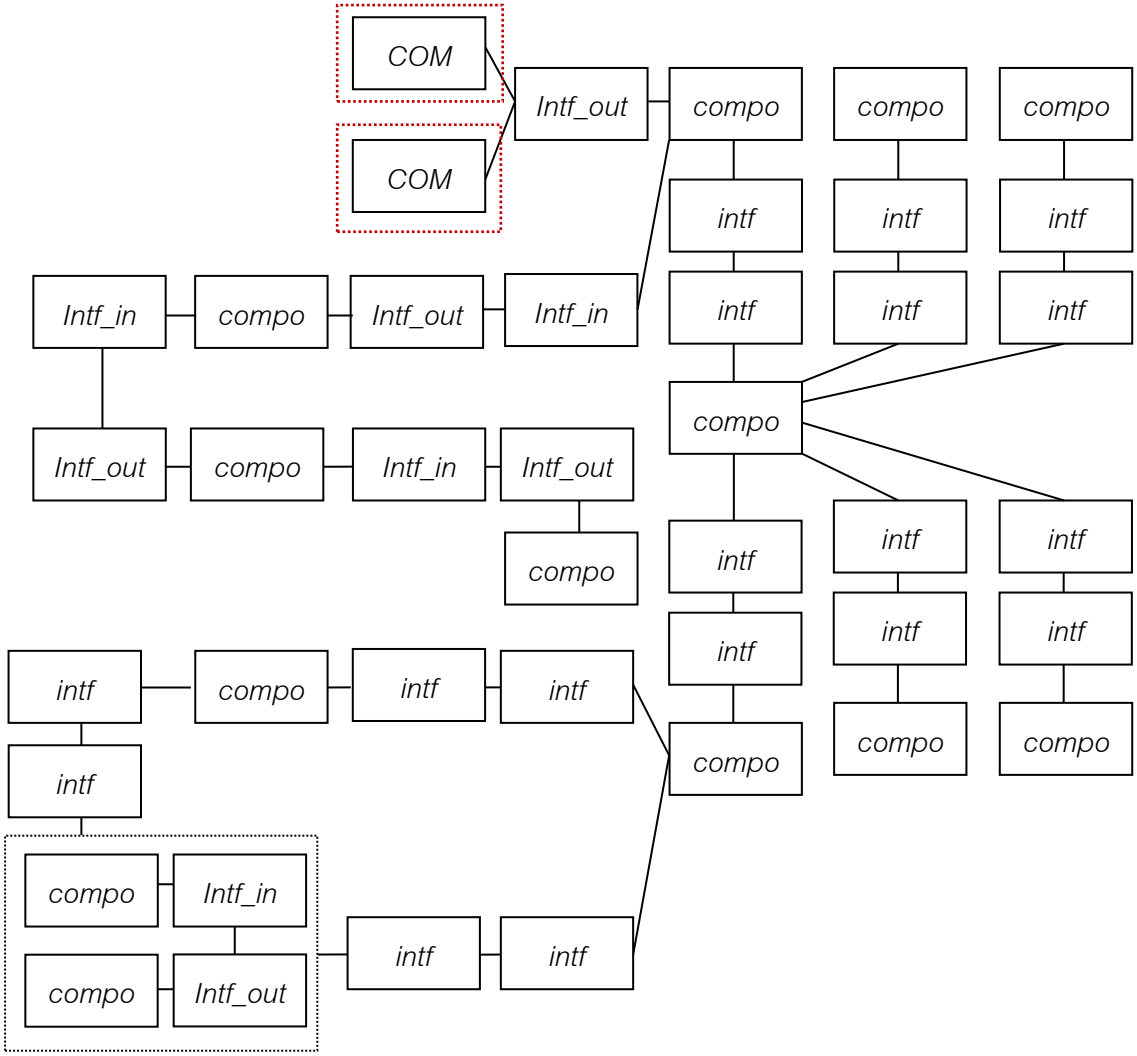
รูปที่ 4.26 ตัวอย่างข้อมูลนำเข้าที่แปลงเป็นกราฟและทำการเลือกบัพเริ่มต้นแล้วสำหรับตรวจสอบคุณสมบัติของสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้น

จากนั้นใช้นิยามข้ออื่นที่เกี่ยวข้องและนิยามข้อ 8 ในบทที่ 3 หลังจากทำการเลือกบัพเริ่มต้นแล้ว ใช้กฎข้อ P7 สองครั้งในการเปลี่ยนรูปแต่เนื่องจากเราใช้วิธีการตรวจสอบโดยใช้กฎการลดรูป ดังนั้นกฎที่จะเอามาใช้จะเขียนแทนด้วย P7’ ซึ่งจะใช้ในการเปลี่ยนรูปโดยการสร้าง INTFS ขึ้นมาบนคอมโพเนนท์ที่อินเตอร์เฟสที่มีทิศทางออกของคอมโพเนนท์เริ่มต้นเชื่อมต่ออยู่ ซึ่งจะได้ผลดังรูปที่ 4.27



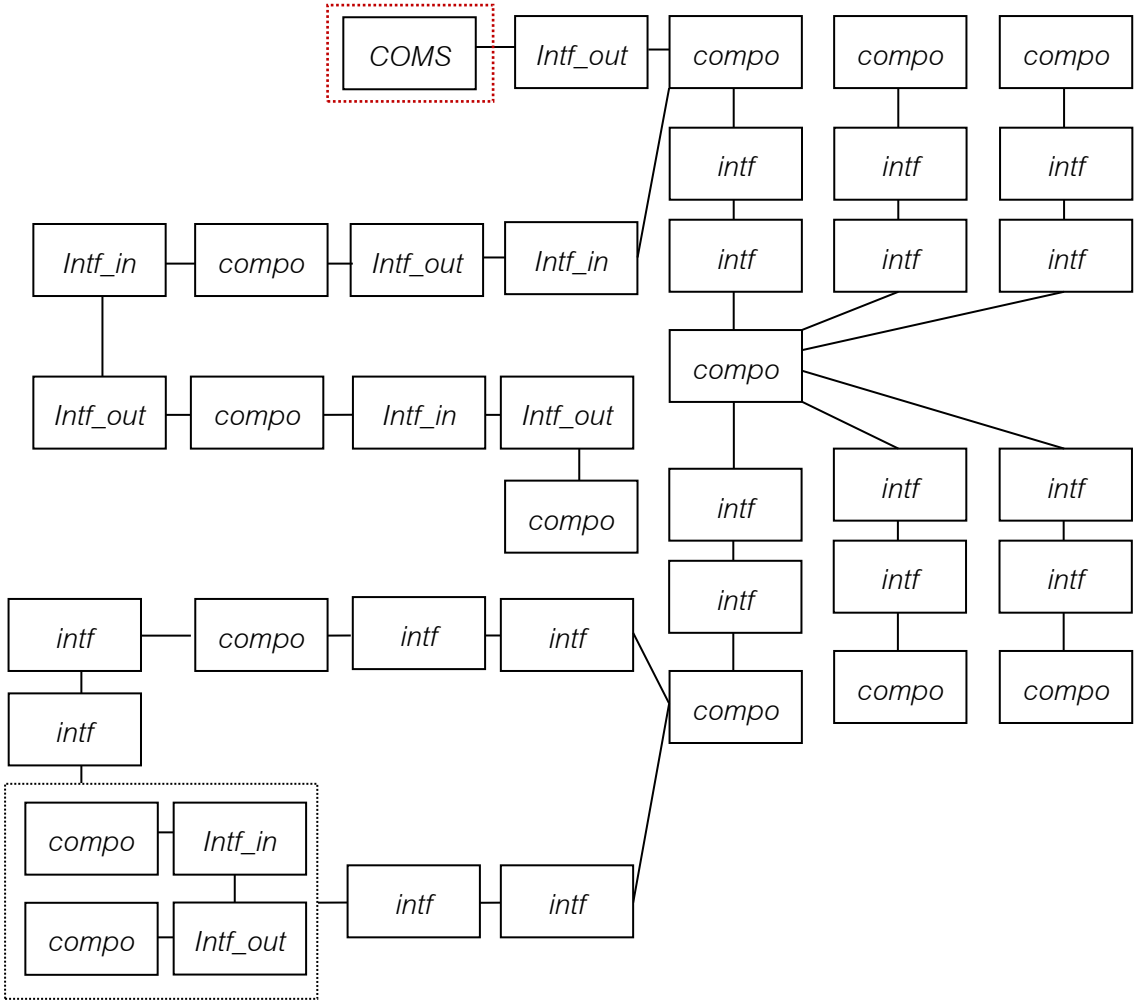
รูปที่ 4.27 ตัวอย่างข้อมูลนำเข้าที่แปลงเป็นกราฟและทำการเลือกบัพเริ่มต้นแล้ว

จากนั้นใช้นิยามข้ออื่นที่เกี่ยวข้องและนิยามข้อ 8 ในบทที่ 3 เราสามารถใช้กฎ P8’ เพื่อที่จะเปลี่ยนรูปร่างของกราฟเพื่อสร้าง COM ดังรูปที่ 4.28



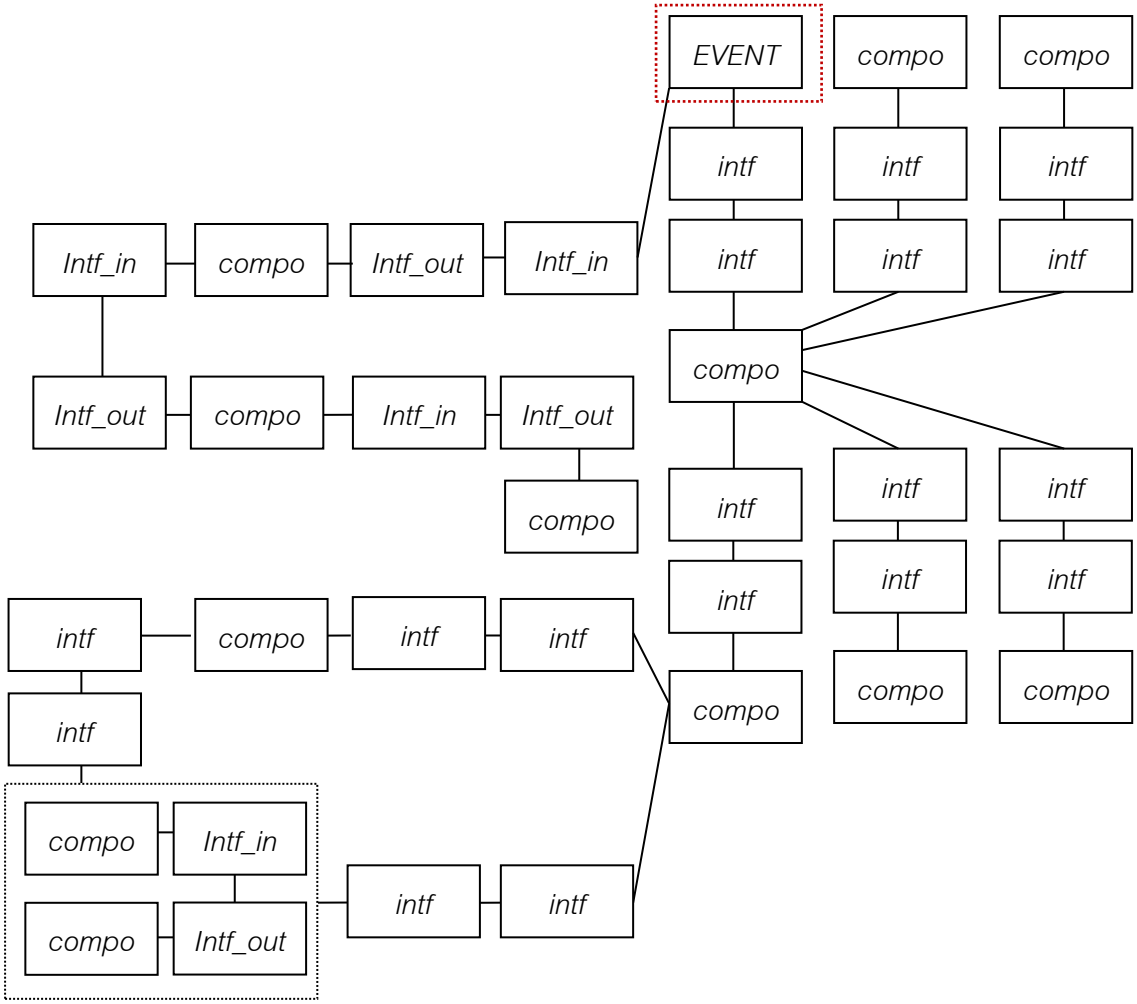
รูปที่ 4.28 ตัวอย่างข้อมูลนำเข้าที่แปลงเป็นกราฟแล้วและทำการใช้กฎ P8' แล้ว

จากนั้นใช้นิยามข้ออื่นที่เกี่ยวข้องและนิยามข้อ 8 ในบทที่ 3 เราสามารถใช้กฎ P9’ เพื่อที่จะเปลี่ยนรูปร่างของกราฟโดยการรวม COM ที่มีมากกว่าหนึ่งและเชื่อมต่อกับอินเตอร์เฟสที่มีทิศทางออกให้เป็น COMS เพื่อที่จะใช้กฎ P11’ ต่อไป ซึ่งได้ผลลัพธ์ดังรูปที่ 4.29



รูปที่ 4.29 ตัวอย่างข้อมูลนำเข้าที่แปลงเป็นกราฟแล้วและทำการใช้กฎ P9' แล้ว

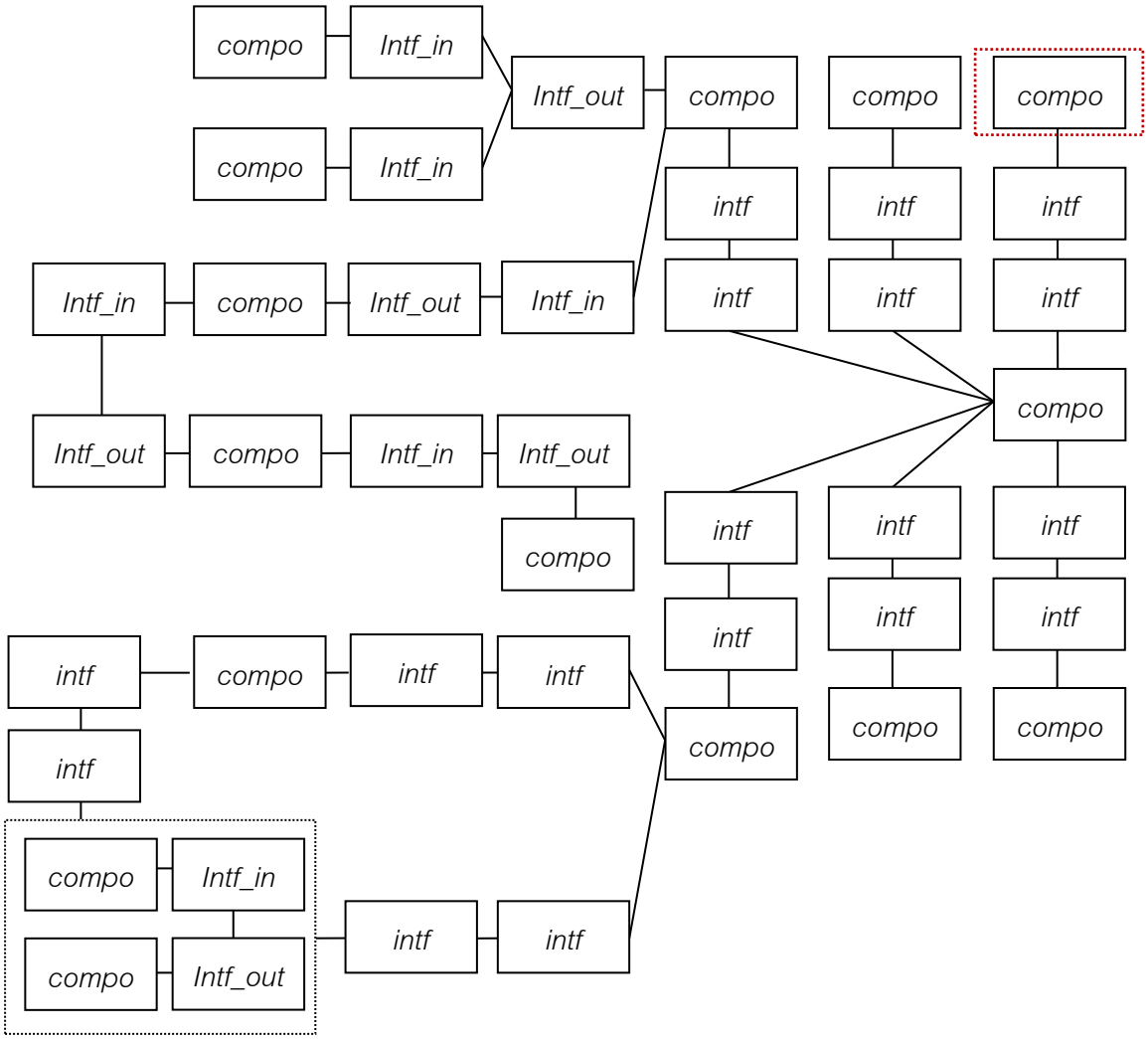
จากนิยามข้ออื่นที่เกี่ยวข้องและ 8 ในบทที่ 3 เราสามารถใช้กฎ P11’ เพื่อยืนยันว่าสามารถตรวจพบสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้นพบ ซึ่งได้ผลลัพธ์ดังรูปที่ 4.30



รูปที่ 4.30 ตัวอย่างข้อมูลนำเข้าที่แปลงเป็นกราฟแล้วและทำการใช้กฎ P11' แล้ว

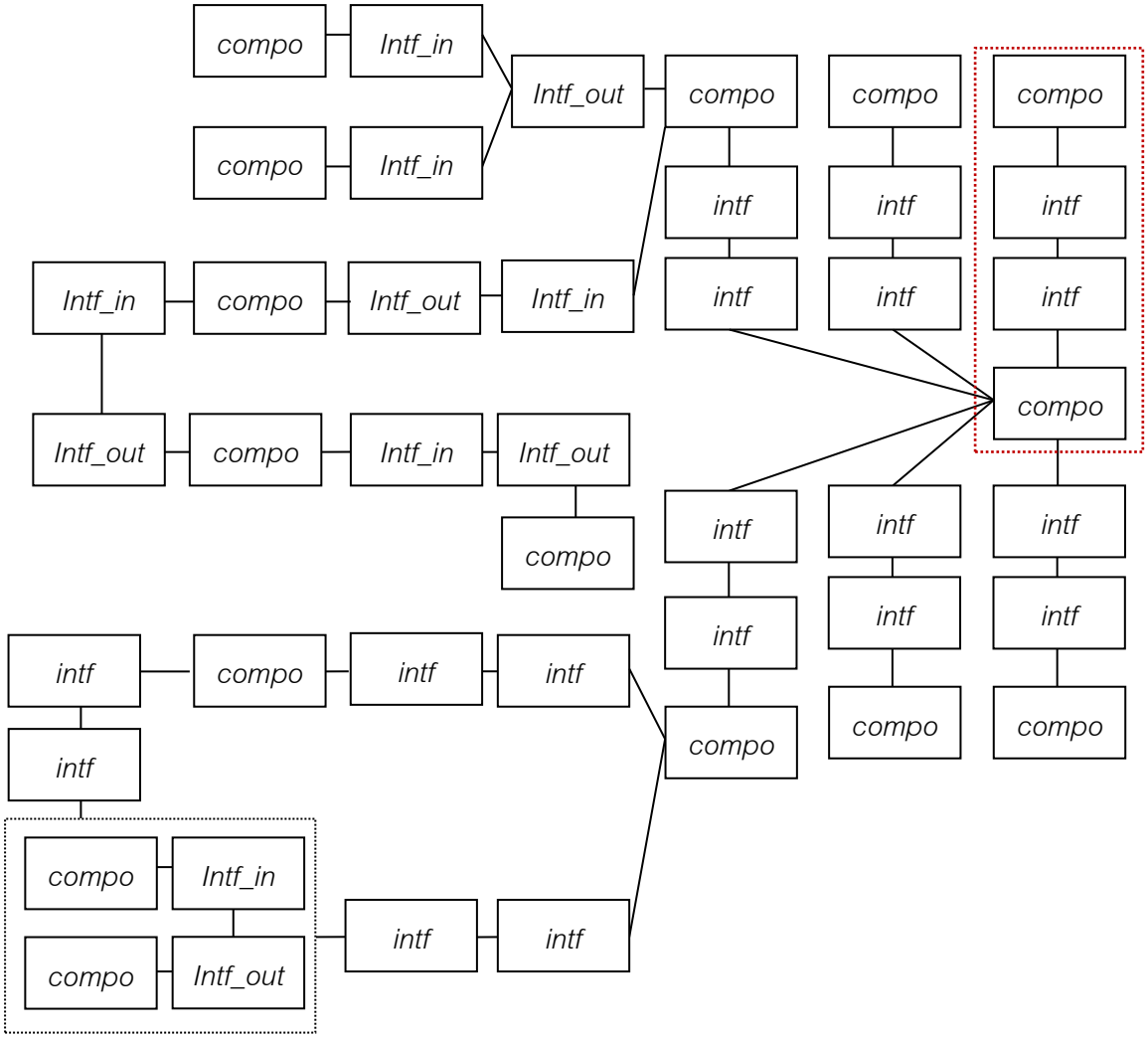
ค) การตรวจสอบสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไทป์ไปป์

จากนิยามข้ออื่นที่เกี่ยวข้องและนิยามข้อ 9 ในบทที่ 3 หลังจากทำการเลือกบัพเริ่มต้นดังรูปที่ 4.31

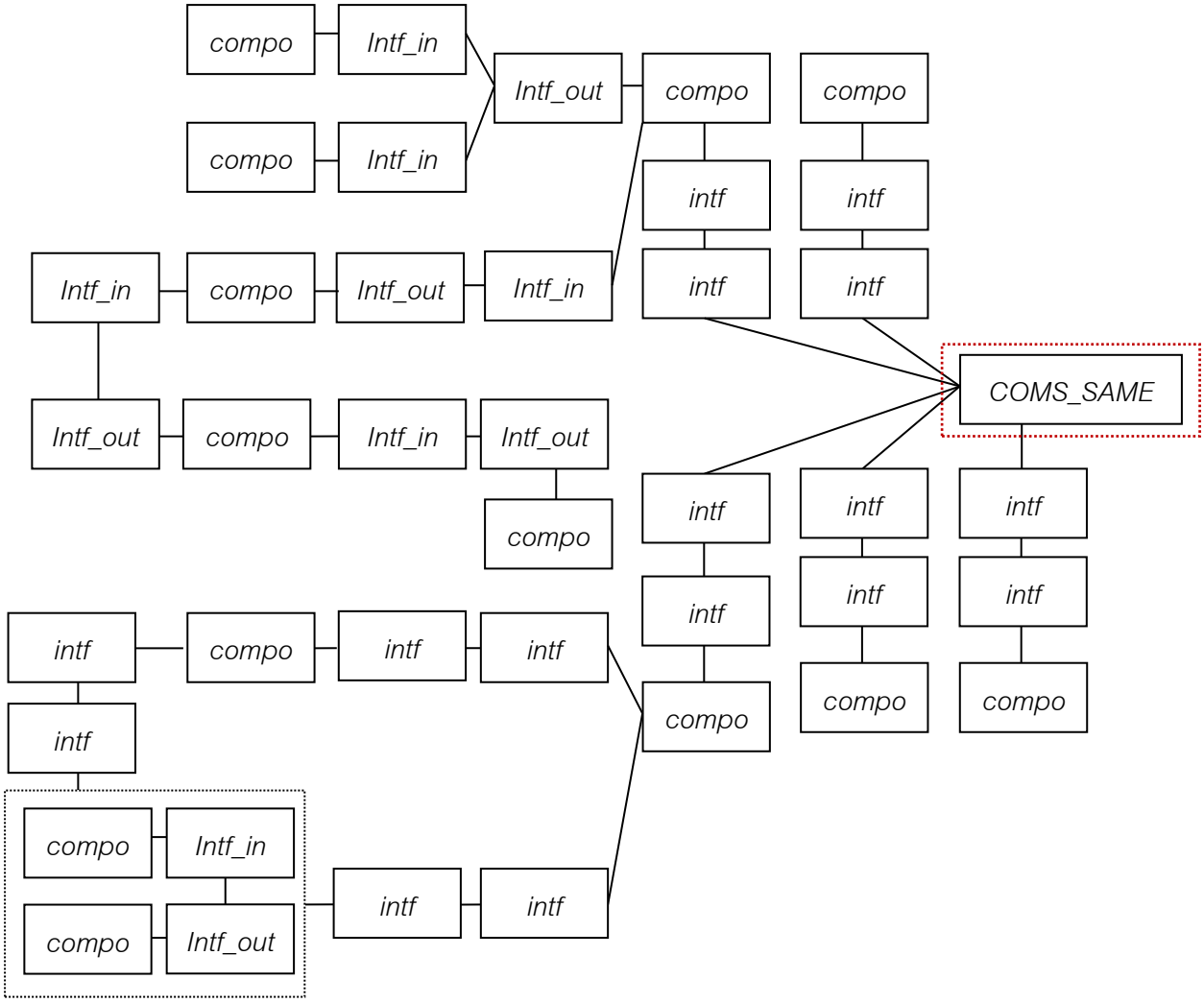


รูปที่ 4.31 ตัวอย่างข้อมูลนำเข้าที่แปลงเป็นกราฟและทำการเลือกบัพเริ่มต้นแล้ว

จากนิยามข้ออื่นที่เกี่ยวข้องและนิยามข้อ 9 ในรูปที่ 4.32 หลังจากทำการเลือกบัพเริ่มต้นแล้ว ใช้กฎข้อ P12 สองครั้งในการเปลี่ยนรูปแต่เนื่องจากเราใช้วิธีการตรวจสอบโดยใช้กฎการลดรูป ดังนั้นกฎที่จะเอามาใช้จะเขียนแทนด้วย P12’ ซึ่งจะใช้ในการเปลี่ยนรูปโดยการรวมคอมโพเนนท์ที่มีชนิดเดียวกันให้อยู่ในคอมโพเนนท์ COMS_SAM ซึ่งจะได้ผลดังดังรูปที่ 4.33

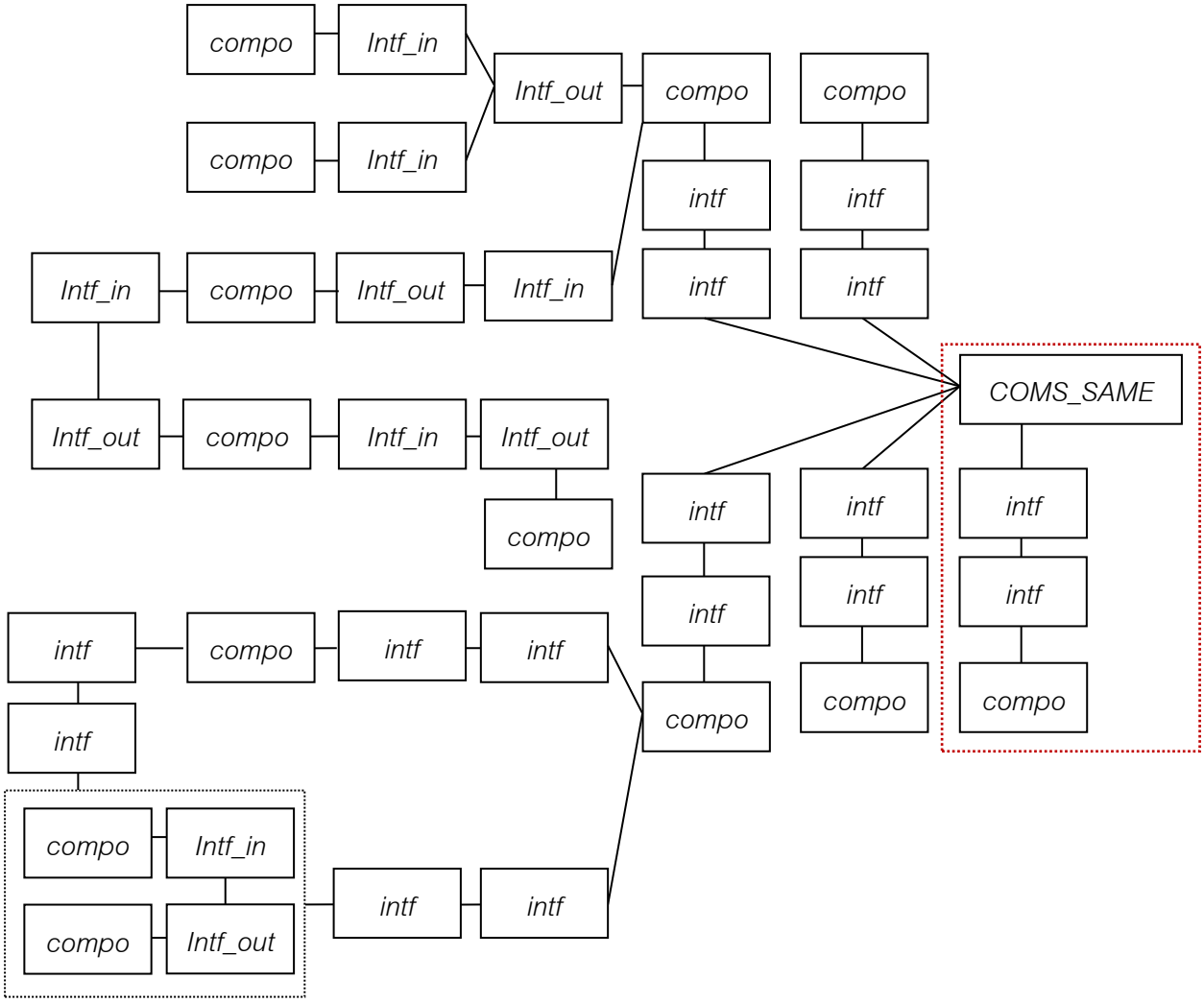


รูปที่ 4.32 ตัวอย่างข้อมูลนำเข้าที่แปลงเป็นกราฟก่อนทำการใช้กฎ P12’

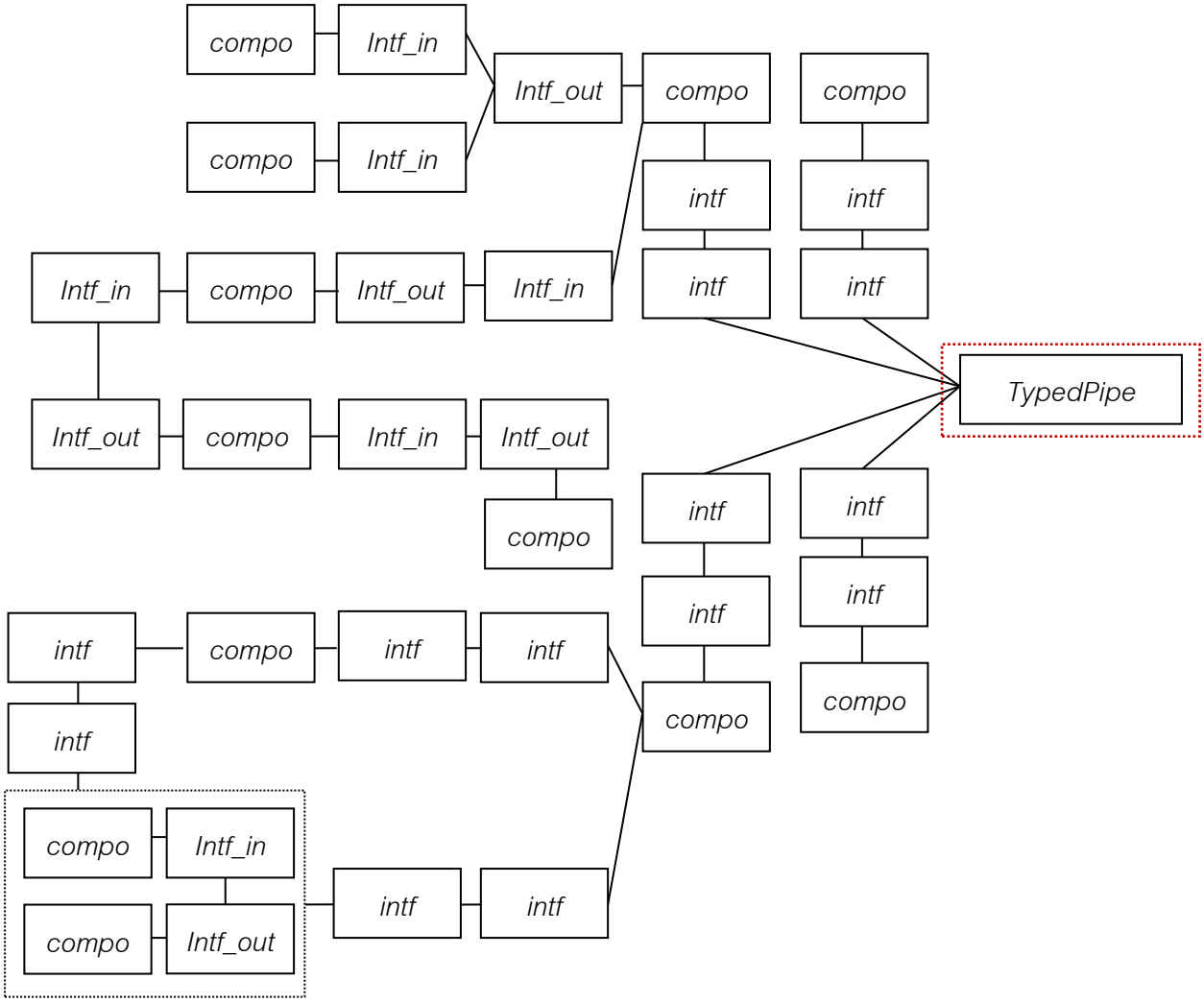


รูปที่ 4.33 ตัวอย่างข้อมูลนำเข้าที่แปลงเป็นกราฟหลังทำการใช้กฎ P12’

จากนั้นใช้นิยามข้ออื่นที่เกี่ยวข้องและนิยามข้อ 9 ในรูปที่ 4.34 ทำการใช้กฎ P14’ เพื่อที่จะเปลี่ยนรูปร่างของกราฟโดยการรวมคอมโพเนนท์ที่มีชนิดของอินเตอร์เฟสเดียวกันไว้ด้วยกันโดยเก็บไว้ในรูปของคอมโพเนนท์ PIPETYPE และเป็นการยืนยันว่าได้ตรวจพบสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไทป์ไปป์แล้วดังรูปที่ 4.35 โดยจากตัวอย่างยังสามารถใช้กฎไวยกรณ์ในนิยามที่เกี่ยวข้องและนิยามข้อ 9 ในการหาบัพที่มีคุณสมบัติสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ได้เพิ่มเติมอีกได้



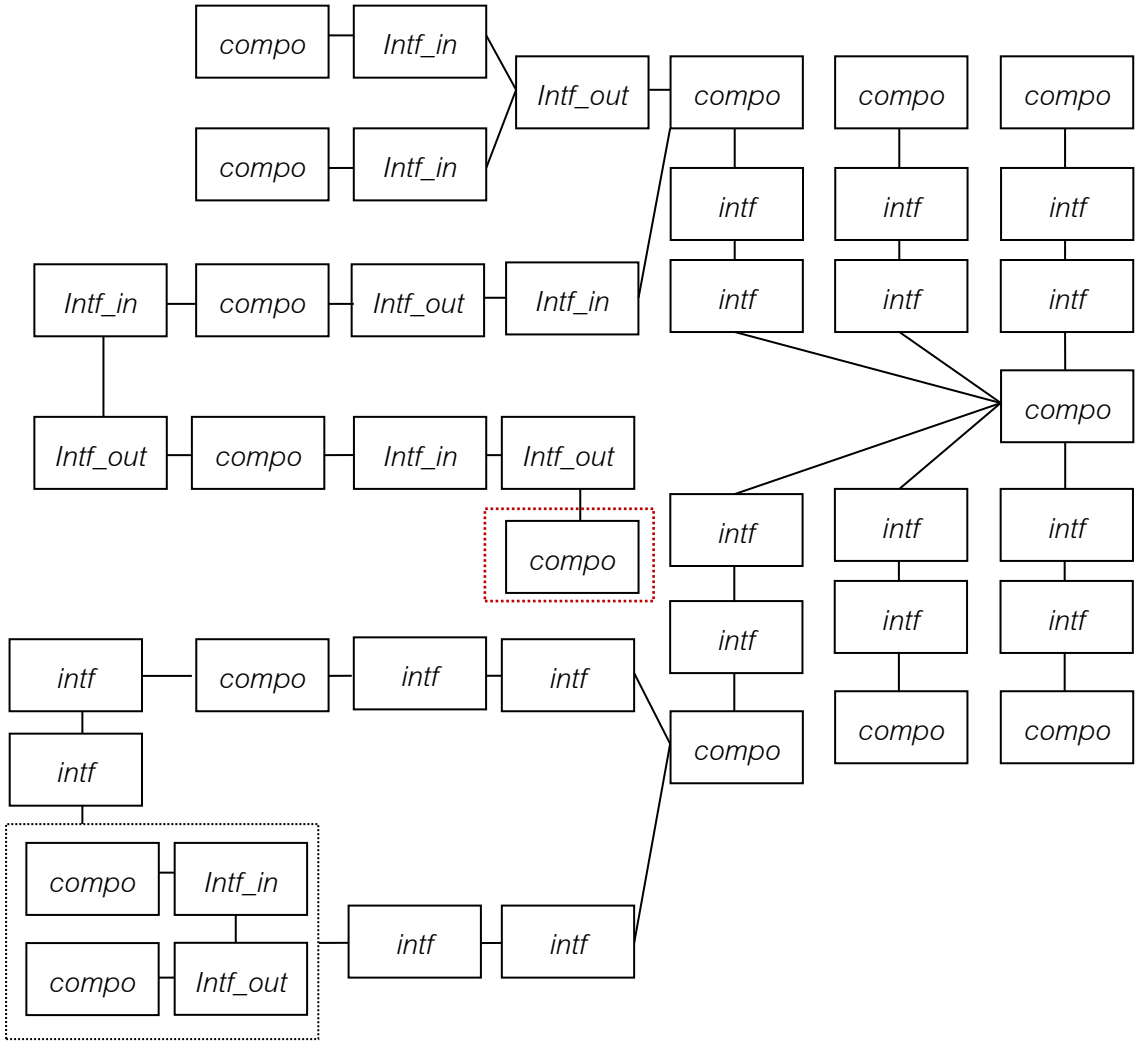
รูปที่ 4.34 ตัวอย่างข้อมูลนำเข้าที่แปลงเป็นกราฟก่อนทำการใช้กฎ P14’



รูปที่ 4.35 ตัวอย่างข้อมูลนำเข้าที่แปลงเป็นกราฟแล้วและทำการใช้กฎ P14' แล้ว

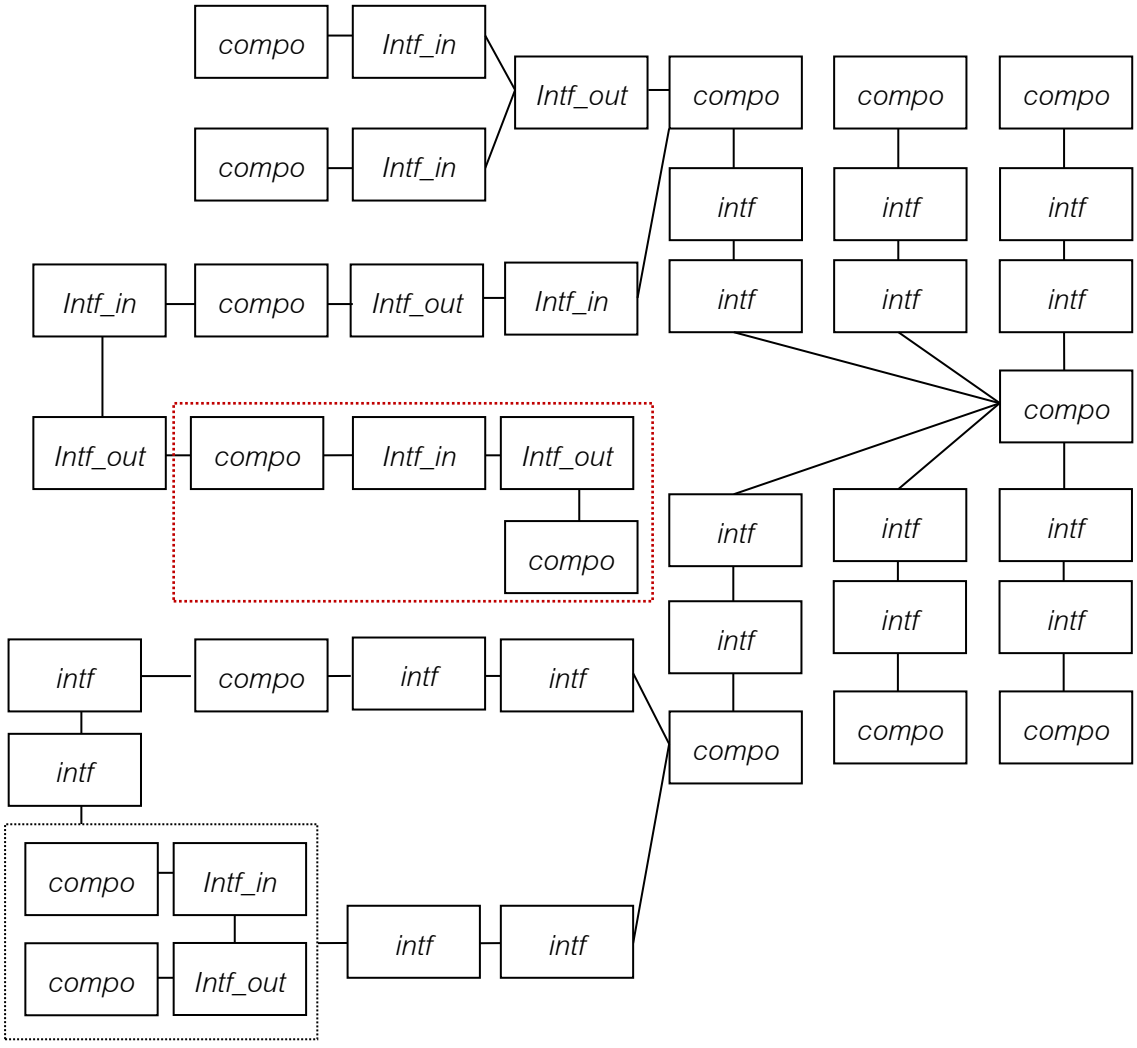
ง) การตรวจสอบสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์

จากนิยามข้ออื่นที่เกี่ยวข้องและนิยามข้อ 10 ในบทที่ 3 หลังจากทำการเลือกบัพเริ่มต้นดังรูปที่ 4.36

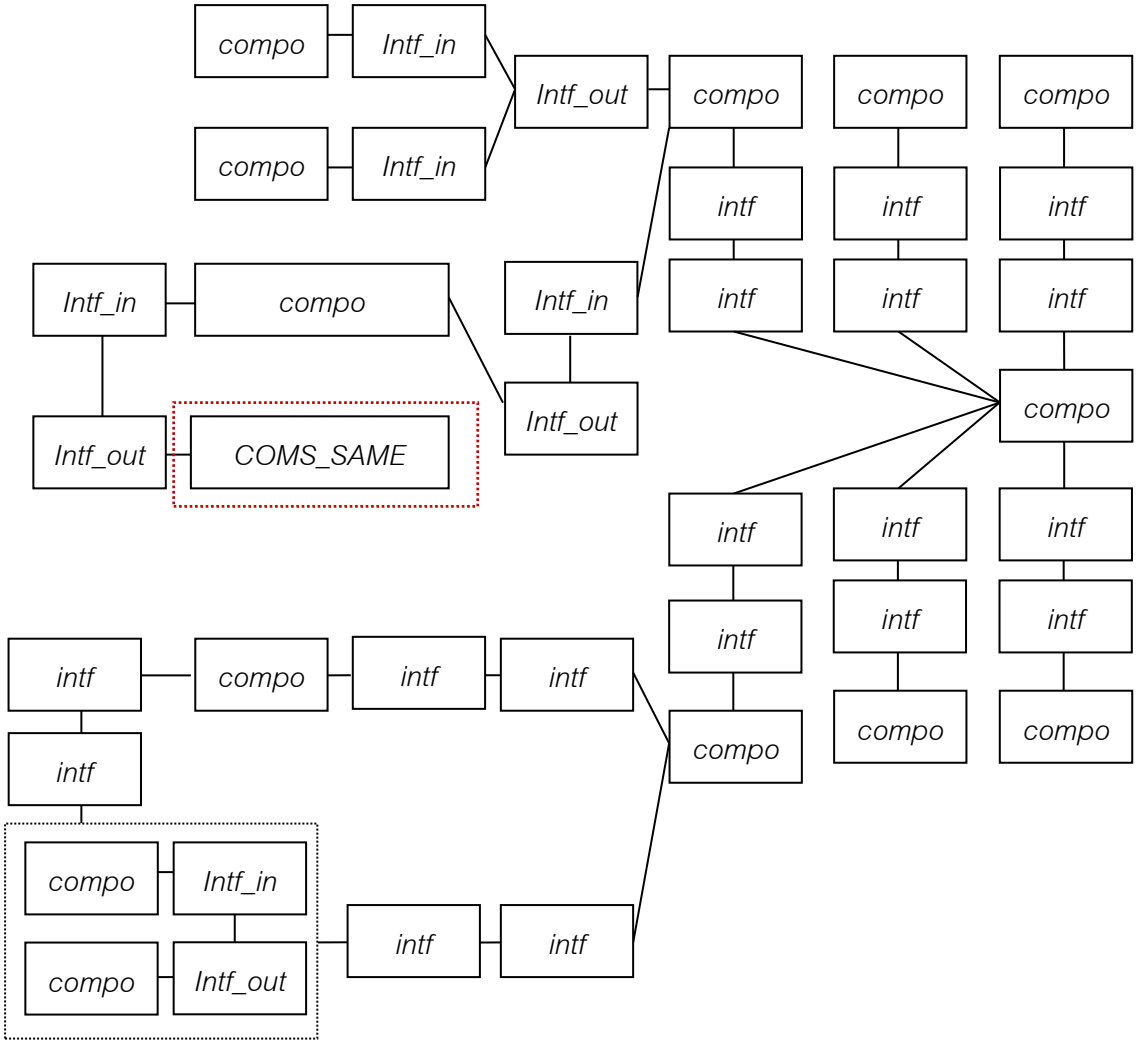


รูปที่ 4.36 ตัวอย่างข้อมูลนำเข้าที่แปลงเป็นกราฟและทำการเลือกบัพเริ่มต้นแล้ว

จากนิยามข้ออื่นที่เกี่ยวข้องและนิยามข้อ 10 ในรูปที่ 4.37 หลังจากทำการเลือกบัพเริ่มต้นแล้ว ใช้กฎข้อ P19 สองครั้งในการเปลี่ยนรูปแต่เนื่องจากเราใช้วิธีการตรวจสอบโดยใช้กฎการลดรูป ดังนั้นกฎที่จะเอามาใช้จะเขียนแทนด้วย P19’ ซึ่งจะใช้ในการเปลี่ยนรูปโดยการรวมคอมโพเนนท์ที่มีทิศทางไปในทางเดียวกันให้อยู่ในคอมโพเนนท์ COMS_SAM ซึ่งจะได้ผลดังรูปที่ 4.38

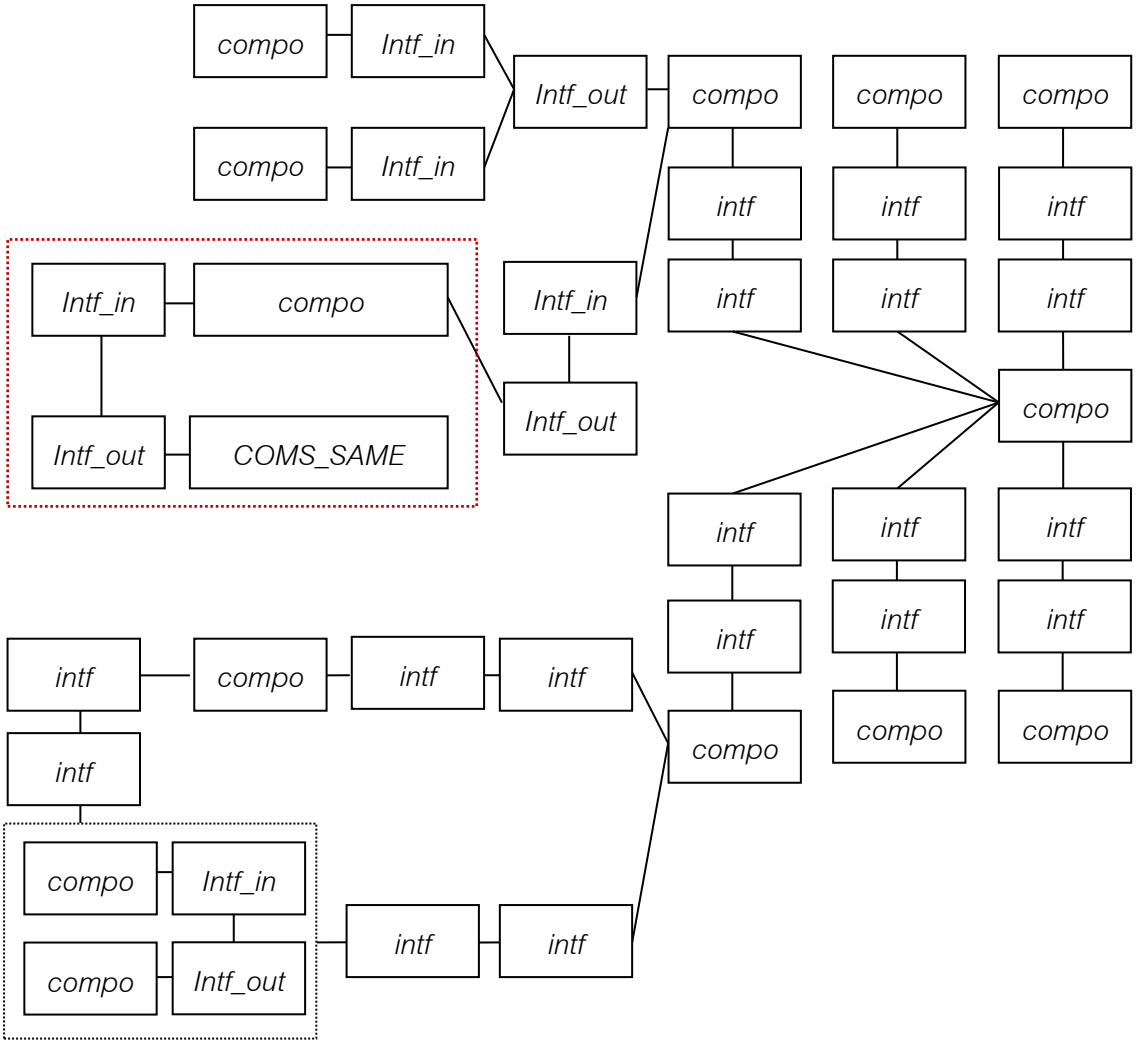


รูปที่ 4.37 ตัวอย่างข้อมูลที่แปลงเป็นกราฟแล้วก่อนทำการใช้กฎ P19'

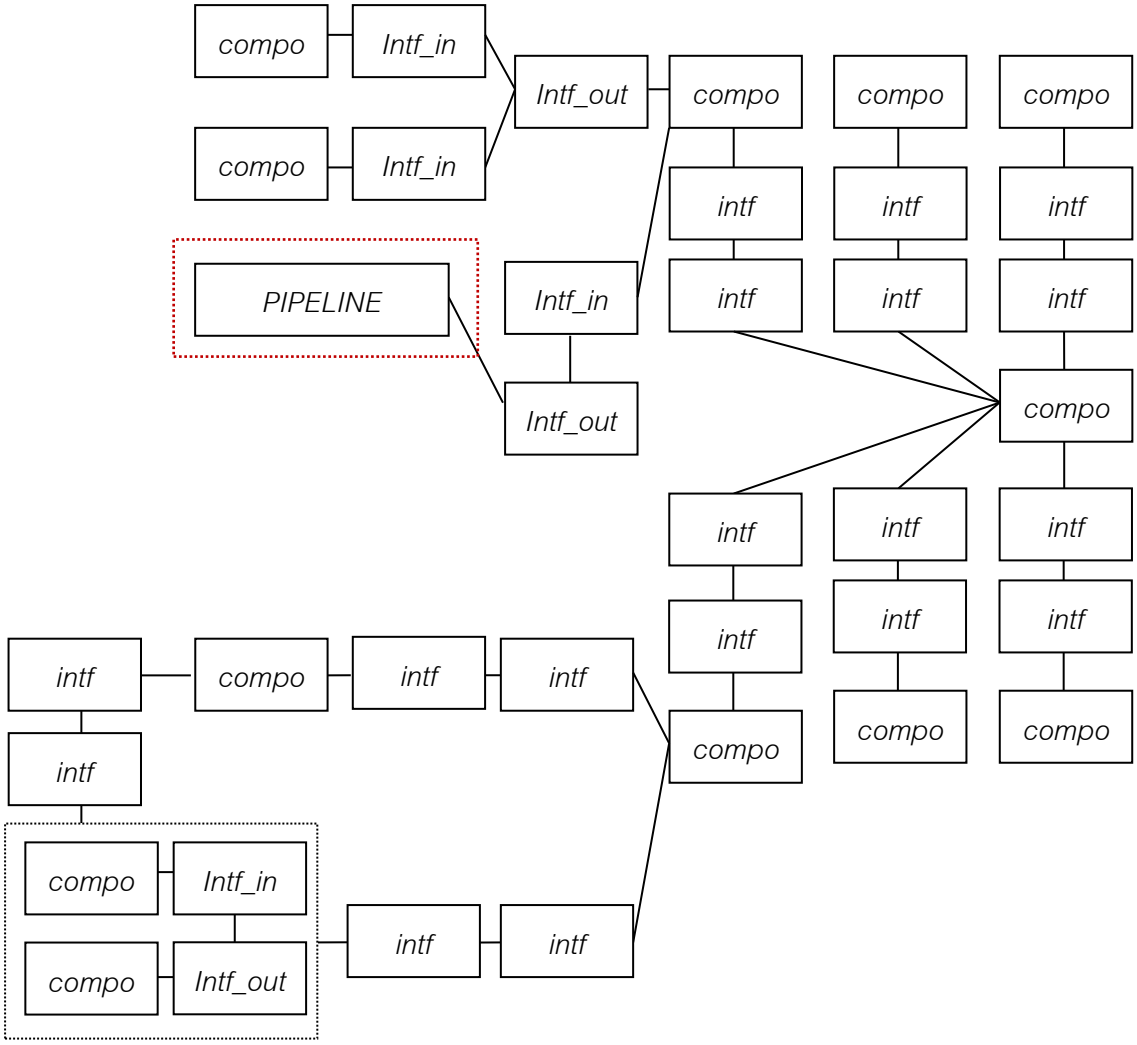


รูปที่ 4.38 ตัวอย่างข้อมูลที่แปลงเป็นกราฟแล้วหลังจากทำการใช้กฎ P19'

จากนิยามข้ออื่นที่เกี่ยวข้องและนิยามข้อ 10 ในบทที่ 3 จากรูปที่ 4.39 ทำการใช้กฎ P20’ เพื่อที่จะเปลี่ยนรูปร่างของกราฟเพื่อเป็นการรวมคอมโพเนนท์ที่เดินทางไปในทางเดียวกันมารวมกันไว้ใน COMS_SAME เป็นแบบที่เราต้องการแต่เนื่องจากเงื่อนไขของสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ว่าต้องมีคอมโพเนนท์ที่เดินทางไปในทิศทางเดียวกันอย่างน้อยสามคอมโพเนนท์ดังนั้นเมื่อรวมคอมโพเนนท์ที่เดินทางไปในทิศทางเดียวกันมาเพิ่มก็จะครบตามเงื่อนไขของสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ ดังรูปที่ 4.40



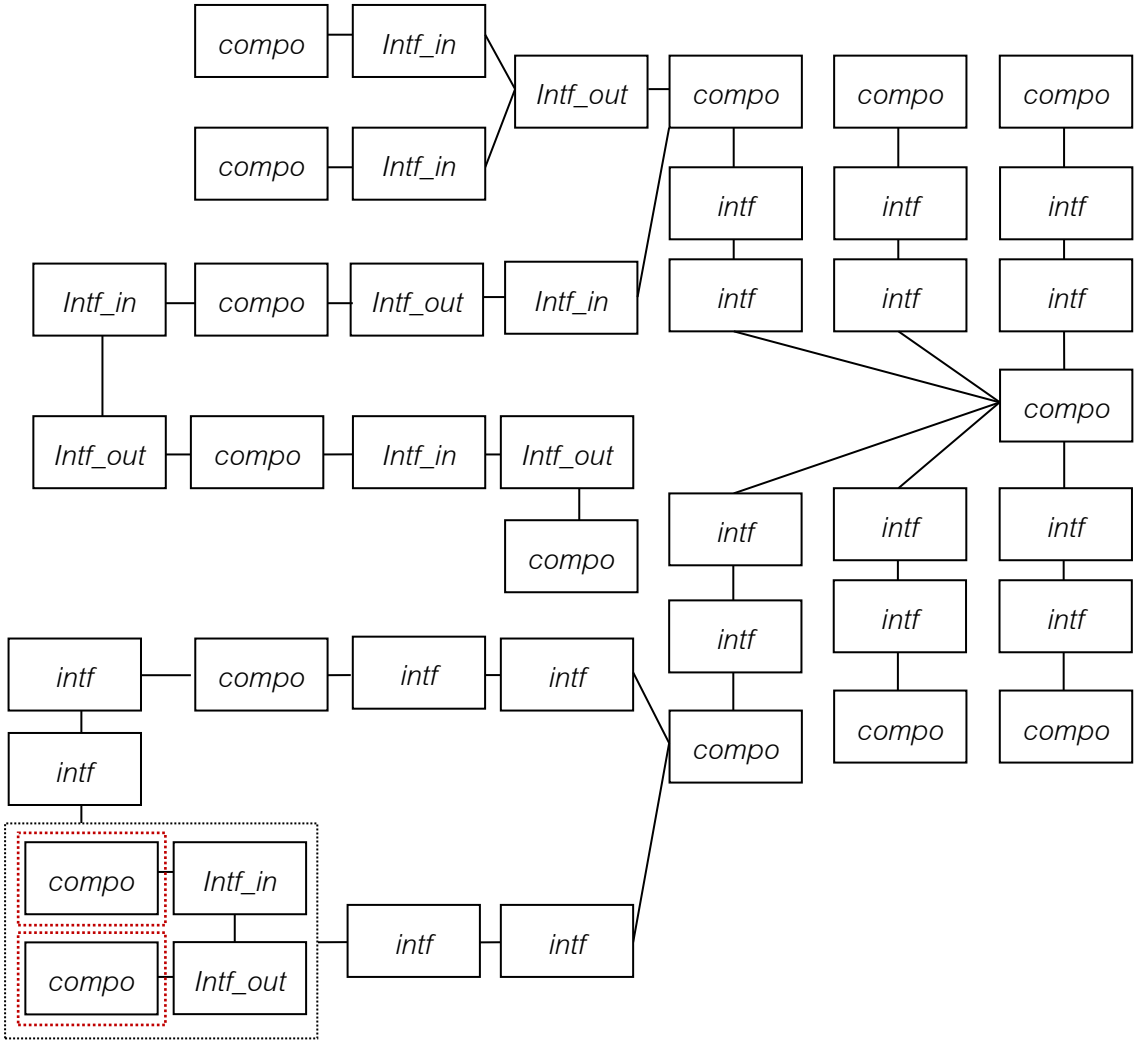
รูปที่ 4.39 ตัวอย่างข้อมูลที่แปลงเป็นกราฟแล้วก่อนทำการใช้กฎ P20'



รูปที่ 4.40 ตัวอย่างข้อมูลนำเข้าที่แปลงเป็นกราฟแล้วและทำการใช้กฎ P20' แล้ว

จ) การตรวจสอบสถาปัตยกรรมในรูปแบบลำดับชั้น (Layer)

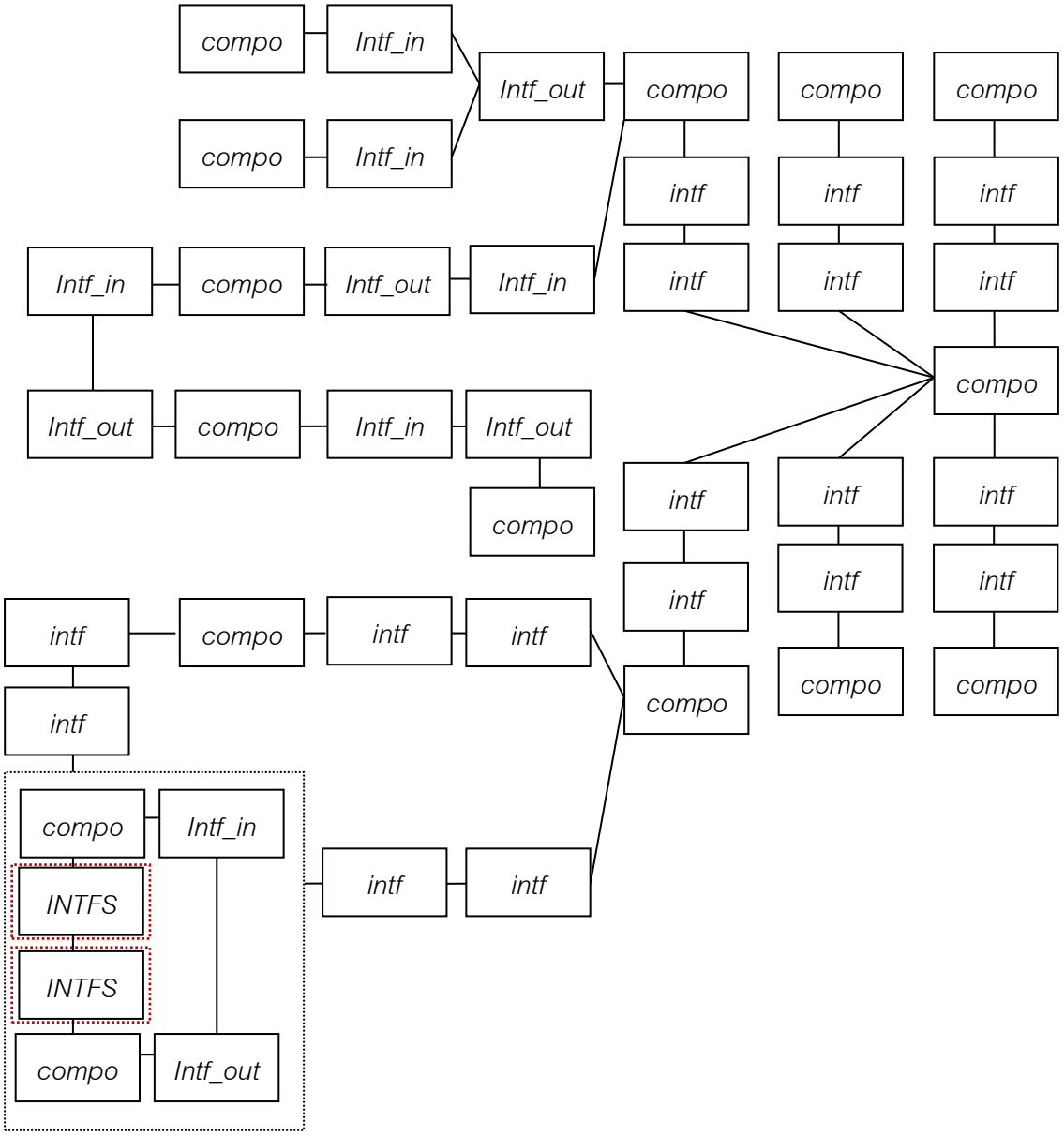
จากนิยามข้ออื่นที่เกี่ยวข้องและนิยามข้อ 11 ในบทที่ 3 หลังจากทำการเลือกบัพเริ่มต้นดังรูปที่ 4.41



รูปที่ 4.41 ตัวอย่างข้อมูลนำเข้าที่แปลงเป็นกราฟและทำการเลือกบัพเริ่มต้นแล้วสำหรับตรวจสอบคุณสมบัติของสถาปัตยกรรมในรูปแบบลำดับชั้น

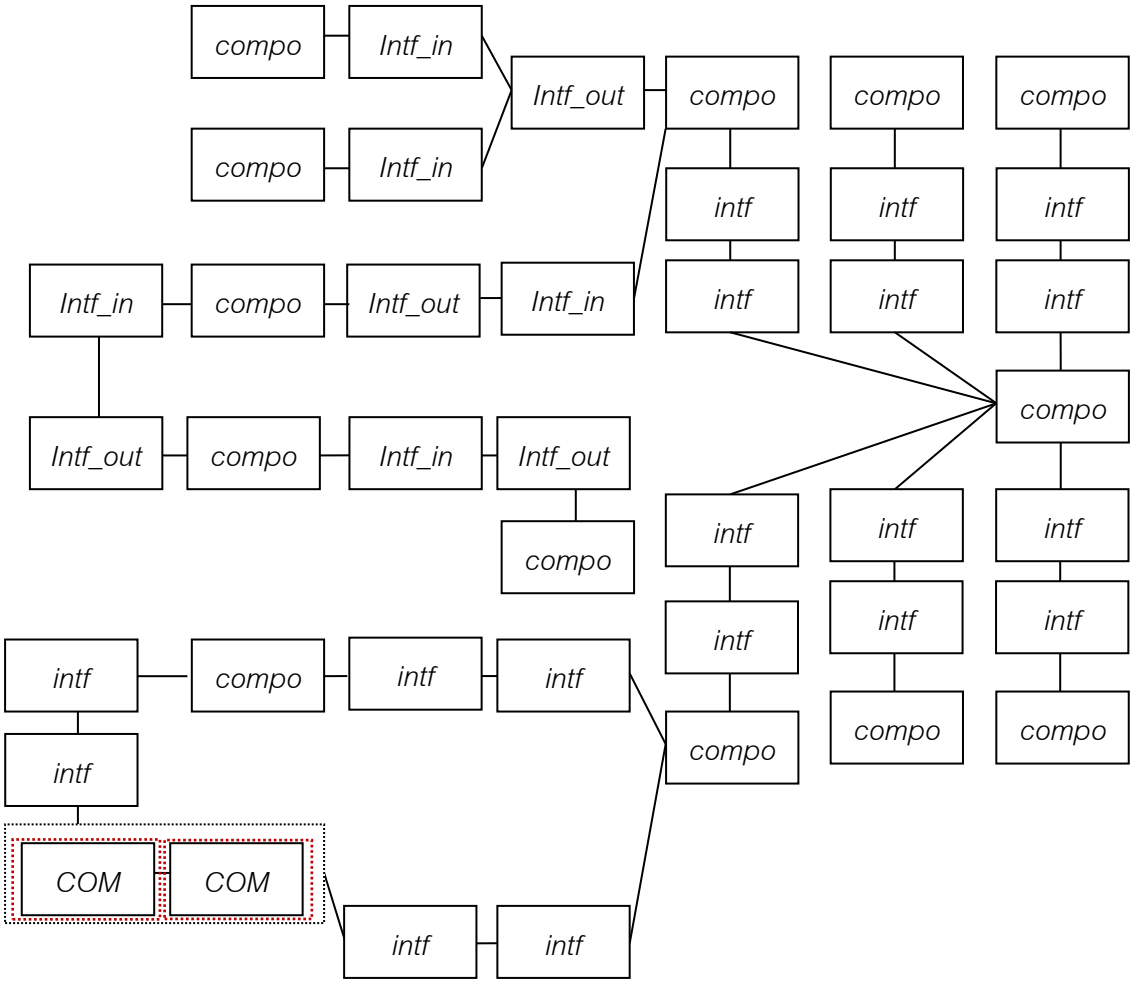
หลังจากทำการเลือกบัพเริ่มต้นแล้ว เป้าหมายของเราคือการทำให้บัพในแต่ละชั้นอยู่ในรูปที่จะใช้กฎข้อ P24 ได้ ดังนั้นเราสามารถใช้กฎข้อ P7 แต่เนื่องจากเราใช้วิธีการตรวจสอบโดยใช้กฎการลดรูป ดังนั้นกฎที่จะเอามาใช้

จะเขียนแทนด้วย P7’ เพื่อที่จะเปลี่ยนรูปร่างของกราฟ ในการเปลี่ยนรูป ซึ่งจะได้ผลดังรูปที่ 4.42



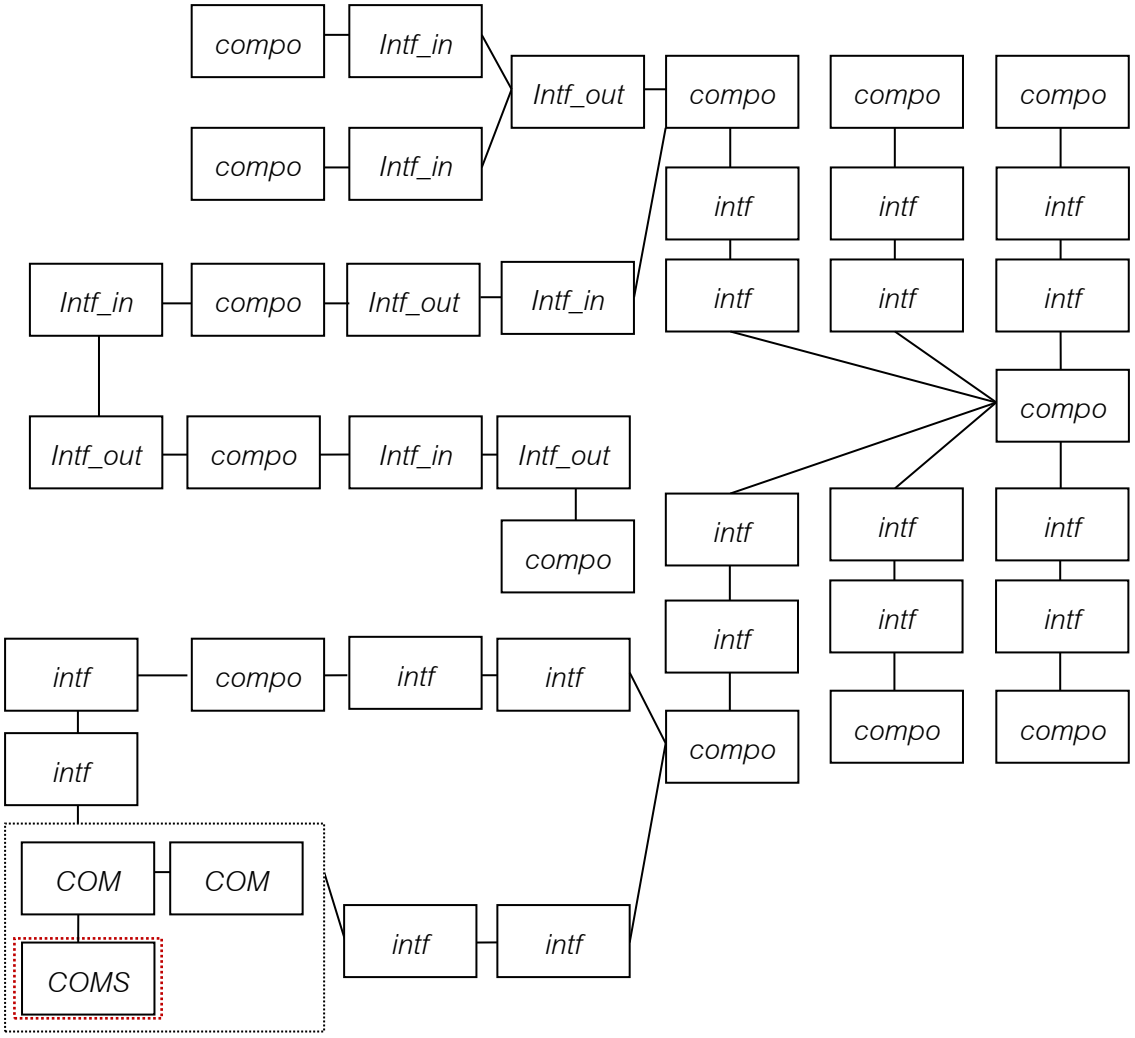
รูปที่ 4.42 ตัวอย่างข้อมูลที่แปลงเป็นกราฟแล้วหลังจากทำการใช้กฎ P7'

จากนิยามข้ออื่นที่เกี่ยวข้องและนิยามข้อ 11 ในบทที่ 3 จากรูปที่ 4.42ทำการใช้กฎ P8’ และ P28’ เพื่อที่จะเปลี่ยนรูปร่างของกราฟเพื่อสร้าง COM ดังรูปที่ 4.43



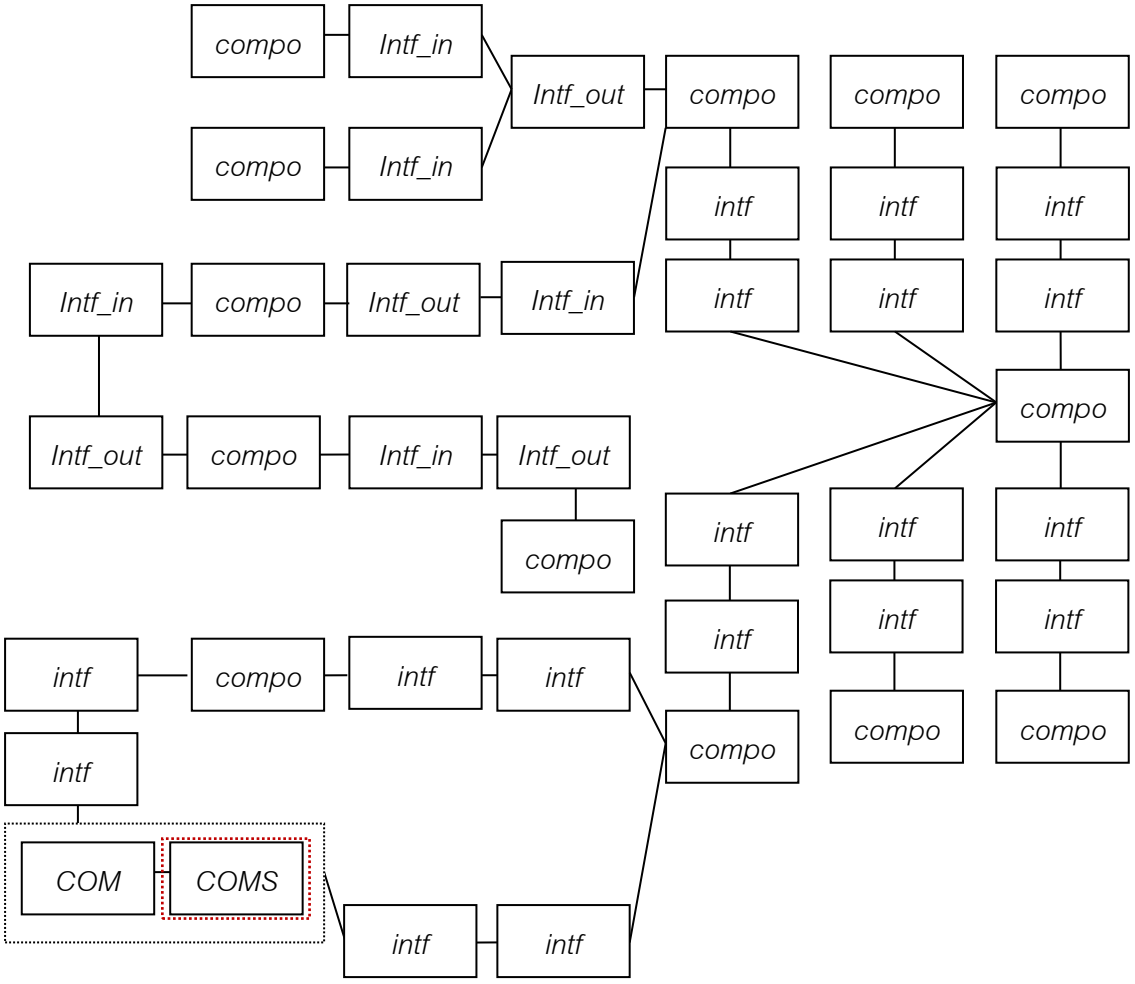
รูปที่ 4.43 ตัวอย่างข้อมูลที่แปลงเป็นกราฟแล้วหลังจากทำการใช้กฎ P8’ และ P28’

จากนิยามข้ออื่นที่เกี่ยวข้องและนิยามข้อ 11 ในบทที่ 3 จากรูปที่ 4.43 ทำการใช้กฎ P3’ เพื่อที่จะเปลี่ยนรูปร่างของกราฟเพื่อสร้าง COMS ดังรูปที่ 4.44



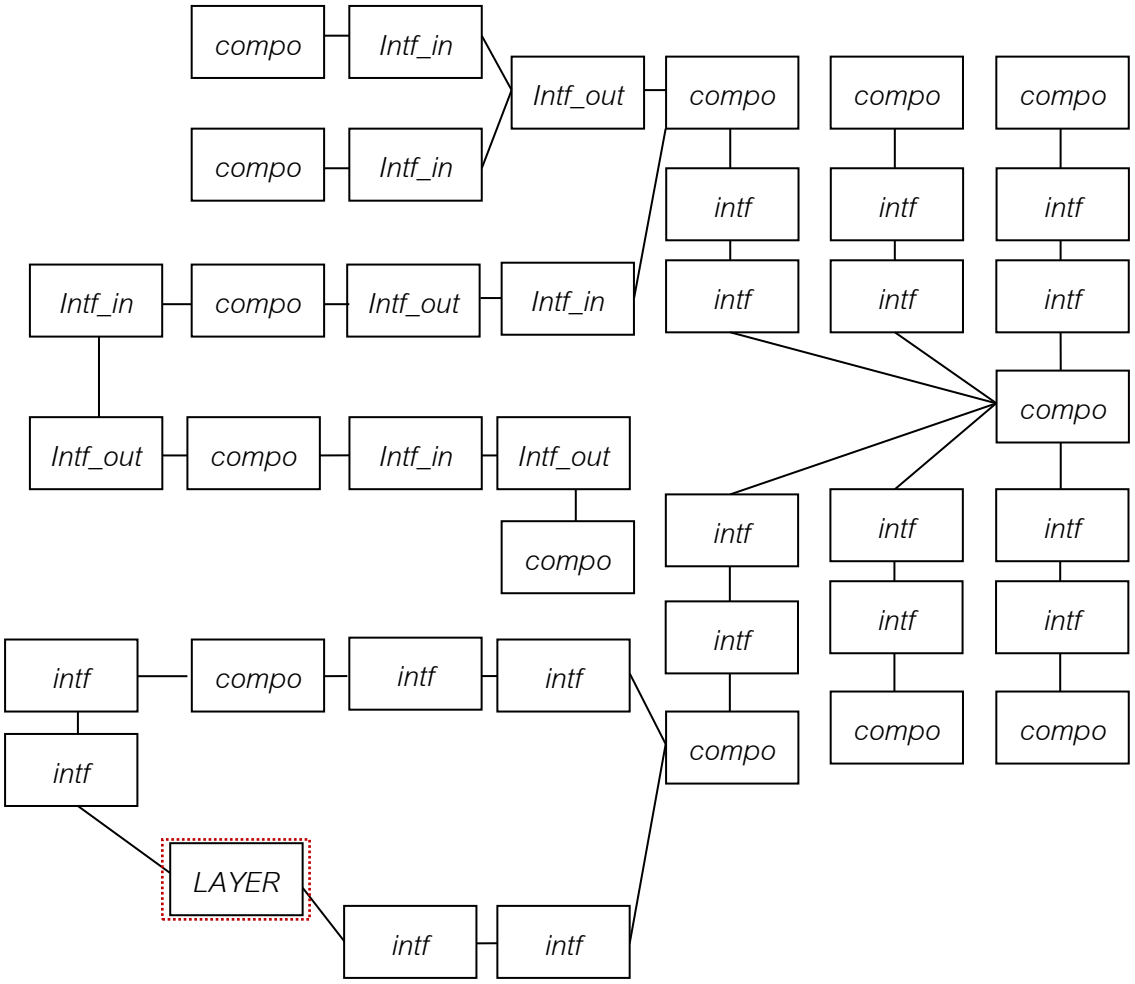
รูปที่ 4.44 ตัวอย่างข้อมูลที่แปลงเป็นกราฟแล้วหลังจากทำการใช้กฎ P3’

จากนิยามข้ออื่นที่เกี่ยวข้องและนิยามข้อ 11 ในบทที่ 3 จากรูปที่ 4.44 ทำการใช้กฎ P2’ เพื่อที่จะเปลี่ยนรูปร่างของกราฟเพื่อรวม COM เข้าไปอยู่ใน COMS ดังรูปที่ 4.45



รูปที่ 4.45 ตัวอย่างข้อมูลที่แปลงเป็นกราฟแล้วหลังจากทำการใช้กฎ P2’

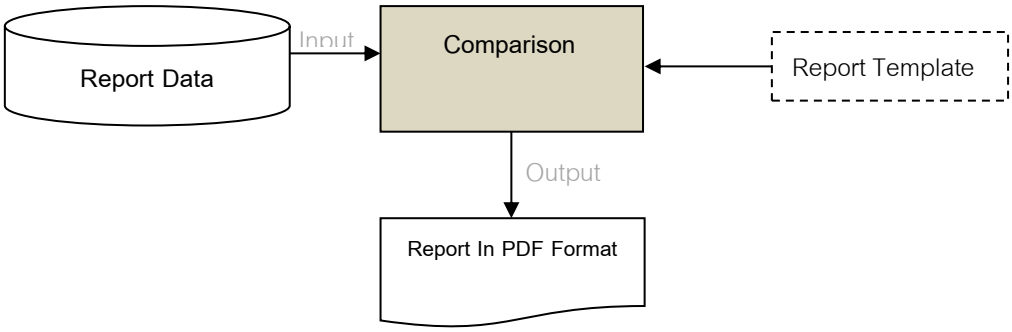
จากนิยามข้ออื่นที่เกี่ยวข้องและนิยามข้อ 11 ในบทที่ 3 จากรูปที่ 4.45 ทำการใช้กฎ P24’ เพื่อที่จะเปลี่ยนรูปร่างของกราฟเพื่อสร้าง Layer และเป็นการยืนยันว่าได้ตรวจพบสถาปัตยกรรมในรูปแบบลำดับชั้นดังรูปที่ 4.46



รูปที่ 4.46 ตัวอย่างข้อมูลนำเข้าที่แปลงเป็นกราฟแล้วและทำการใช้กฎ P24' แล้ว

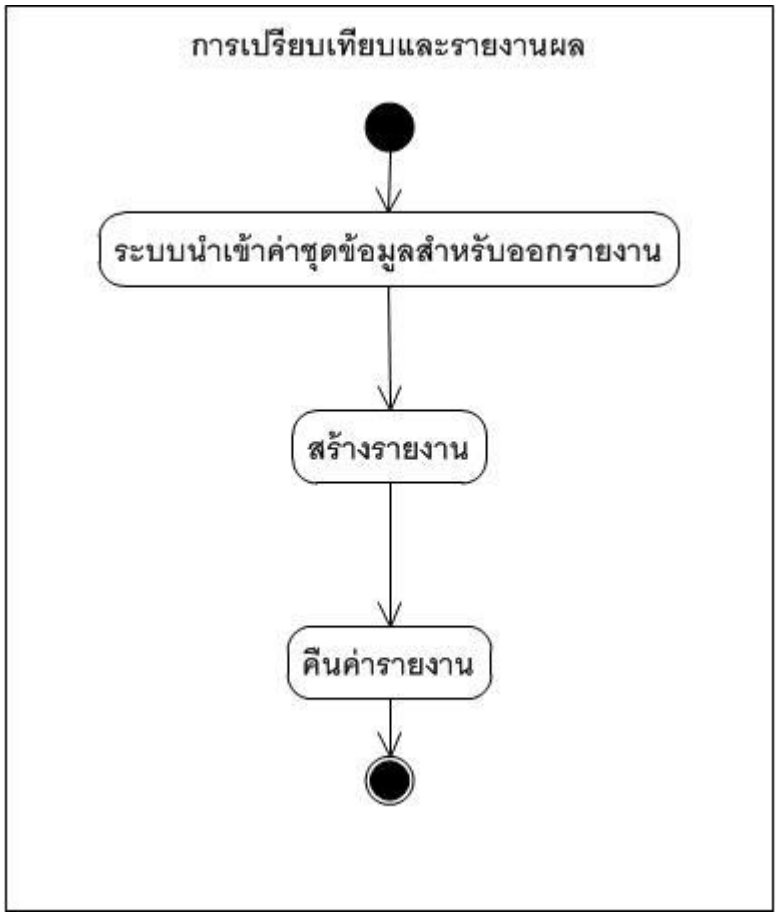
4.1.2.4 กระบวนการเปรียบเทียบและรายงานผล (Comparison)

ในกระบวนการที่สี่เป็นการนำผลลัพธ์ที่ถูกเก็บอยู่ในตารางรายงานมาแสดงให้ผู้ใช้ได้เห็นผลลัพธ์ของการทำงานต่อไป แสดงดังรูปที่ 4.47



รูปที่ 4.47 การเปรียบเทียบและรายงานผล

จากรูปที่ 4.47 สามารถแสดงแผนภาพกิจกรรมดังรูปที่ 4.48



รูปที่ 4.48 แผนภาพกิจกรรมการเปรียบเทียบและรายงานผล

4.1.3 กลุ่มกิจกรรมนำข้อมูลผลลัพธ์การตรวจสอบรูปแบบสถาปัตยกรรมซอฟต์แวร์มาแสดงผล

การแสดงผลข้อมูลผลลัพธ์จากเครื่องมือตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์ ด้วยเครื่องมือ PDF Reader แสดงดังรูปที่ 4.49

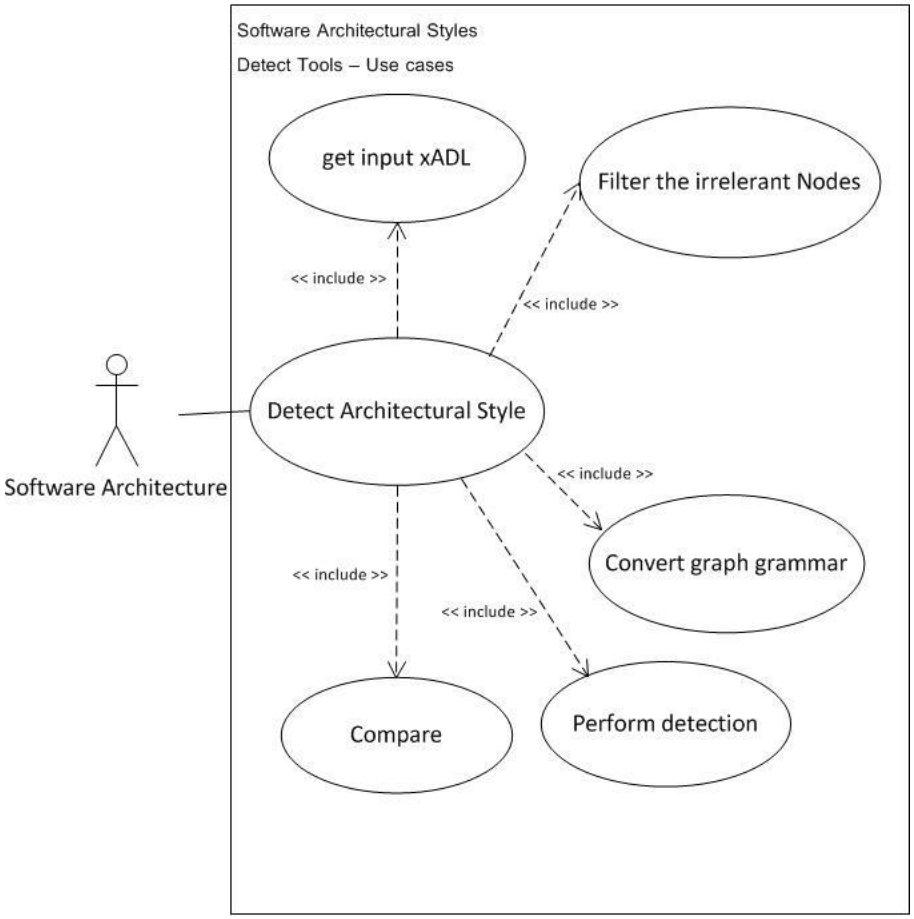
รูปที่ 4.49 ตัวอย่างรายงานผลการตรวจสอบสถาปัตยกรรม

4.2 การวิเคราะห์และออกแบบเครื่องมือตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์

จากขั้นตอนการตรวจจับรูปแบบของสถาปัตยกรรมซอฟต์แวร์ดังกล่าวข้างต้น นำไปพัฒนาเครื่องมือตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์ สำหรับการออกแบบและพัฒนาเครื่องมือได้นำเสนอผ่านยูสเคส แผนภาพคลาส อธิบายรายละเอียดดังนี้

1.2.1 แผนภาพยูสเคส

เครื่องมือการตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์ได้ถูกอธิบายถึงพื้นฐานของเครื่องมือในมุมมองของการใช้งานด้วยแผนภาพยูสเคสดังรูปที่ 4.50



รูปที่ 4.50 แผนภาพยูสเคสของเครื่องมือการตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์ด้วยกราฟแกรมมา

เครื่องมือสำหรับตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์ด้วยกราฟแกรมมาประกอบได้ด้วย 6 ยูสเคส ได้แก่ ยูสเคสการตรวจสอบสถาปัตยกรรม,ยูสเคสนำเข้าข้อมูลสถาปัตยกรรม,แปลงข้อมูลเป็นกราฟแกรมมา,กรองข้อมูลที่ไม่เกี่ยวข้องออกจากกราฟแกรมมา,ยูสเคสการตรวจสอบสถาปัตยกรรมและยูสเคสการเปรียบเทียบและรายงานผล โดยแสดงดังตาราง

ตารางที่ 4.6 คำอธิบายยูสเคสการตรวจสอบสถาปัตยกรรม (Detect architectural style)

ชื่อยูสเคส	ยูสเคสการตรวจสอบสถาปัตยกรรมซอฟต์แวร์
ผู้กระทำ	สถาปนิกซอฟต์แวร์
รายละเอียดยูสเคส	เป็นยูสเคสสำหรับการตรวจสอบสถาปัตยกรรมซอฟต์แวร์ โดยการนำเข้าสถาปัตยกรรมซอฟต์แวร์ผ่านยูสเคสนำเข้าข้อมูลสถาปัตยกรรม และแสดงผลลัพธ์ของการตรวจสอบสถาปัตยกรรมผ่านทางยูสเคสการเปรียบเทียบและรายงานผล
เงื่อนไขก่อนหน้า	แฟ้มข้อมูลนาเข้าที่ใช้ตรวจจับต้องมีรูปแบบเป็นภาษาเอกซ์เอดีแอลเวอร์ชันสอง
ขั้นตอน	1. ผู้ใช้เข้าสู่หน้าจอตรวจสอบสถาปัตยกรรม 2. ผู้ใช้เลือกแฟ้มข้อมูลสถาปัตยกรรมซอฟต์แวร์ที่จะทำการตรวจสอบ 3. กดปุ่มยืนยันการนำเข้าข้อมูล
เงื่อนไขภายหลัง	ในกรณีที่เครื่องมือเจอลักษณะข้อมูลไม่ตรงตามที่รูปแบบเป็นภาษาเอกซ์เอดีแอล เวอร์ชันสองระบบจะทำการแจ้งเตือน

ตารางที่ 4.7 คำอธิบายยูสเคสนำเข้าข้อมูลสถาปัตยกรรม (Get Input xADL)

ชื่อยูสเคส	ยูสเคสนำเข้าข้อมูลสถาปัตยกรรม
ผู้กระทำ	ไม่ระบุ
รายละเอียดยูสเคส	เป็นยูสเคสสำหรับการนำเข้าแฟ้มข้อมูลสถาปัตยกรรมซอฟต์แวร์ที่เก็บอยู่ในรูปของเอกซ์เอดีแอล ซึ่งผู้ใช้จะทำการเลือกข้อมูลสถาปัตยกรรมซอฟต์แวร์ที่ได้ออกแบบ และทำการแปลงข้อมูลให้อยู่ในรูปของส่วนประกอบต่างๆ ของกราฟแกรมมาเพื่อนำไปตรวจสอบสถาปัตยกรรมต่อไป
เงื่อนไขก่อนหน้า	แฟ้มข้อมูลนาเข้าที่ใช้ตรวจจับต้องมีรูปแบบเป็นภาษาเอกซ์เอดีแอลเวอร์ชันสอง
ขั้นตอน	1. ผู้ใช้เข้าสู่หน้าจอตรวจสอบสถาปัตยกรรม 2. ผู้ใช้เลือกแฟ้มข้อมูลสถาปัตยกรรมซอฟต์แวร์ที่จะทำการตรวจสอบ 3. กดปุ่มยืนยันการนำเข้าข้อมูล

ตารางที่ 4.7 คำอธิบายยูสเคสนำเข้าข้อมูลสถาปัตยกรรม (Get Input xADL) (ต่อ)

เงื่อนไขภายหลัง	ในกรณีที่เครื่องมือเจอลักษณะข้อมูลไม่ตรงตามที่รูปแบบเป็นภาษาเอกซ์เอดีแอลเวอร์ชันสองระบบจะทำการแจ้งเตือน

ตารางที่ 4.8 แปลงข้อมูลเป็นกราฟแกรมมา (Convert Graph Grammar)

ชื่อยูสเคส	ยูสเคสแปลงข้อมูลเป็นกราฟแกรมมา
ผู้กระทำ	ไม่ระบุ
รายละเอียดยูสเคส	เป็นยูสเคสสำหรับการนำเข้าแฟ้มข้อมูลสถาปัตยกรรมซอฟต์แวร์ที่จัดเก็บอยู่ในรูปแบบภาษาเอกซ์เอดีแอล สำหรับกระบวนการแปลงข้อมูลเป็นกราฟแกรมมา ซึ่งระบบจะทำการแปลงข้อมูลให้อยู่ในรูปของส่วนประกอบต่างๆ ของกราฟแกรมมาเพื่อนำไปตรวจสอบสถาปัตยกรรมต่อไป
เงื่อนไขก่อนหน้า	ข้อมูลนำเข้าผ่านกระบวนการนำเข้าข้อมูลสถาปัตยกรรม เพื่อตรวจสอบลักษณะข้อมูลก่อน
ขั้นตอน	1. ทำการแยกส่วนประกอบของเอกซ์เอดีแอล 2. จัดเก็บข้อมูลที่ได้จากการแยกส่วนประกอบของเอกซ์เอดีแอล ลงที่จัดเก็บตามลักษณะของข้อมูล 3. คืนค่าการจัดเก็บจากการแยกส่วนประกอบ
เงื่อนไขภายหลัง	คืนค่าข้อมูลที่ได้จากแยกส่วนประกอบ

ตารางที่ 4. 9 กรองข้อมูลที่ไม่เกี่ยวข้องออกจากกราฟแกรมมา (Filter the irrelevant Nodes)

ชื่อยูสเคส	ยูสเคสกรองข้อมูลที่ไม่เกี่ยวข้องออกจากกราฟแกรมมา
ผู้กระทำ	ไม่ระบุ
รายละเอียดยูสเคส	เป็นยูสเคสสำหรับการนำเข้าส่วนประกอบต่างๆ ของกราฟแกรมมา สำหรับกระบวนการกรองข้อมูลที่ไม่เกี่ยวข้องออกจากกราฟแกรมมา ซึ่งระบบจะทำการตรวจสอบข้อมูลและลบข้อมูลที่ไม่เกี่ยวข้องออกเพื่อนำไปตรวจสอบสถาปัตยกรรมต่อไป
เงื่อนไขก่อนหน้า	ข้อมูลนำเข้าผ่านกระบวนการแปลงข้อมูลเป็นกราฟแกรมมาก่อน

ตารางที่ 4.9 กรองข้อมูลที่ไม่เกี่ยวข้องออกจากกราฟแกรมมา (Filter the irrelevant Nodes) (ต่อ)

ขั้นตอน	1. ตรวจสอบข้อมูลที่ได้จากการแยกส่วนประกอบ ว่ามีส่วนประกอบใดบ้างไม่เชื่อมต่อกับส่วนประกอบอื่นๆ 2. ลบข้อมูลที่ตรวจพบว่าไม่มีการเชื่อมต่อกับส่วนประกอบอื่นๆ 3. คืนค่าการจัดเก็บจากการแยกส่วนประกอบ
เงื่อนไขภายหลัง	คืนค่าข้อมูลที่ได้จากแยกส่วนประกอบและมีการตรวจสอบข้อมูลที่ไม่เกี่ยวข้องแล้ว

ตารางที่ 4.10 คำอธิบายยูสเคสการตรวจสอบสถาปัตยกรรม (Perform detection)

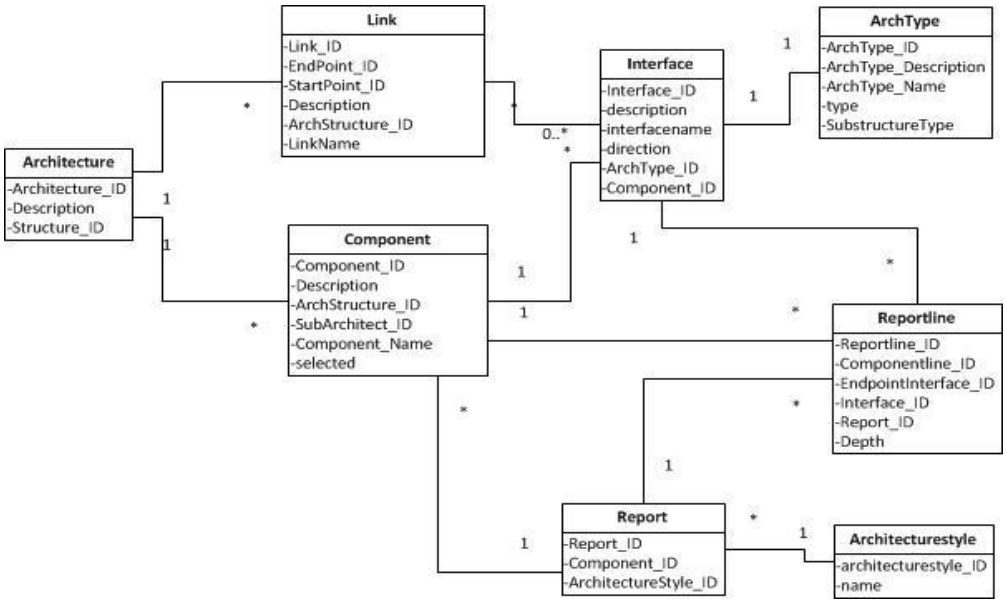
ชื่อยูสเคส	ยูสเคสการตรวจสอบสถาปัตยกรรม
ผู้กระทำ	ไม่ระบุ
รายละเอียดยูสเคส	เป็นยูสเคสสำหรับการนำเข้าส่วนประกอบต่างๆ ของกราฟแกรมมา เพื่อใช้ทำการตรวจสอบสถาปัตยกรรม ซึ่งระบบจะทำการตรวจสอบสถาปัตยกรรมที่เกี่ยวข้อง และคืนผลลัพธ์ของผลการตรวจสอบสถาปัตยกรรมเพื่อนำไปใช้เปรียบเทียบและออกรายงานต่อไป
เงื่อนไขก่อนหน้า	ข้อมูลนำเข้าผ่านกระบวนการกรองข้อมูลที่ไม่เกี่ยวข้องออกจากกราฟแกรมมาก่อน
ขั้นตอน	1. ระบบทำการตรวจสอบสถาปัตยกรรมตามสถาปัตยกรรมที่ผู้ใช้ให้ทำการตรวจสอบ 2. ทำการบันทึกผลการตรวจสอบลงชุดข้อมูลสำหรับออกรายงาน 3. คืนค่าชุดข้อมูลสำหรับออกรายงาน
เงื่อนไขภายหลัง	มีการคืนค่าชุดข้อมูลสำหรับออกรายงาน เพื่อใช้สำหรับออกรายงาน

ตารางที่ 4.11 คำอธิบายยูสเคสการเปรียบเทียบและรายงานผล (Compare)

ชื่อยูสเคส	ยูสเคสการเปรียบเทียบและรายงานผล
ผู้กระทำ	สถาปนิกซอฟต์แวร์
รายละเอียดยูสเคส	เป็นยูสเคสสำหรับการนำเข้าค่าชุดข้อมูลสำหรับออกรายงานเพื่อใช้ในการออกรายงาน ซึ่งระบบจะทำออกรายงานให้กับสถาปนิกซอฟต์แวร์
เงื่อนไขก่อนหน้า	ข้อมูลนำเข้าผ่านกระบวนการตรวจสอบสถาปัตยกรรมมาก่อน
ขั้นตอน	1. ระบบนำเข้าค่าชุดข้อมูลสำหรับออกรายงานไปใช้สำหรับออกรายงาน 2. ทำการสร้างรายงาน 3. คืนค่ารายงาน
เงื่อนไขภายหลัง	แสดงผลลัพธ์เป็นรายงานของรูปแบบสถาปัตยกรรม

1.2.2 แผนภาพคลาส

เครื่องมือสำหรับตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์ มีองค์ประกอบและความสัมพันธ์ของข้อมูลที่เกี่ยวข้องกัน โดยแสดงเป็นแผนภาพคลาสได้ดังรูปที่ 4.51



รูปที่ 4.51 แผนภาพคลาสของเครื่องมือการตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์ด้วยกราฟแกรมมา

แผนภาพคลาสในรูปที่ 4.51 มีรายละเอียดดังต่อไปนี้

1) คลาส Component คือ คลาสที่ใช้จัดเก็บโครงสร้างของสถาปัตยกรรม
2) คลาส Interface คือ คลาสที่ใช้จัดเก็บจุดเชื่อมต่อของโครงสร้างของสถาปัตยกรรม
3) คลาส Link คือ คลาสที่ใช้จัดเก็บข้อมูลเส้นเชื่อมของโครงสร้างของสถาปัตยกรรม
4) คลาส Architecture คือ คลาสที่ใช้จัดเก็บข้อมูลเบื้องต้นของโครงสร้างของสถาปัตยกรรม
5) คลาส ArchType คือ คลาสที่ใช้จัดเก็บข้อมูล ชนิด ที่ใช้กับจุดเชื่อมต่อ หรือ โครงสร้างของสถาปัตยกรรม
6) คลาส Architecturalstyle คือ คลาสที่ใช้จัดเก็บข้อมูลชนิดของสถาปัตยกรรมที่ใช้ในการตรวจสอบ
7) คลาส Report คือ คลาสที่ใช้จัดเก็บข้อมูลผลลัพธ์เบื่องต้นจากการค้นหาสถาปัตยกรรม
8) คลาส Reportline คือ คลาสที่ใช้จัดเก็บข้อมูลผลลัพธ์ที่อ้างอิงจากข้อมูลในคลาส Report

ตารางที่ 4.12 รายละเอียดตารางความสัมพันธ์ระหว่างคุณลักษณะกับแผนภาพคลาสออบเจ็กต์

ชื่อคลาส	รายละเอียด
Component	จัดเก็บโครงสร้างของสถาปัตยกรรม
Interface	จัดเก็บจุดเชื่อมต่อของโครงสร้างของสถาปัตยกรรม
Link	จัดเก็บข้อมูลเส้นเชื่อมของโครงสร้างของสถาปัตยกรรม
Architecture	จัดเก็บข้อมูลเบื่องต้นของโครงสร้างของสถาปัตยกรรม
ArchType	จัดเก็บข้อมูล ชนิด ที่ใช้กับจุดเชื่อมต่อ หรือโครงสร้างของสถาปัตยกรรม
Architecturalstyle	จัดเก็บข้อมูลชนิดของสถาปัตยกรรมที่ใช้ในการตรวจสอบ

ตารางที่ 4.12 รายละเอียดตารางความสัมพันธ์ระหว่างคุณลักษณะกับแผนภาพคลาสออบเจ็กต์ (ต่อ)

ชื่อคลาส	รายละเอียด
Report	จัดเก็บข้อมูลผลลัพธ์เบื่องต้นจากการค้นหาสถาปัตยกรรม
Reportline	เก็บข้อมูลผลลัพธ์ที่อ้างอิงจากข้อมูลในคลาส Report

4.3 การพัฒนาเครื่องมือตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์

การพัฒนาเครื่องมือตัวแปลภาษาอธิบายแยกเป็นสองส่วน โดยส่วนแรกกล่าวถึงสภาพแวดล้อมที่ใช้ในการพัฒนาเครื่องมือและส่วนที่สองกล่าวถึงส่วนต่อประสานผู้ใช้งานเครื่องมือ ซึ่งมีรายละเอียดดังนี้

4.3.1 สภาพแวดล้อมที่ใช้ในการพัฒนาเครื่องมือ

สภาพแวดล้อมที่ใช้ในการพัฒนาเครื่องมือ มีรายละเอียดแสดงได้ดังต่อไปนี้

1) ฮาร์ดแวร์ (Hardware)

- เครื่องคอมพิวเตอร์แบบโน๊ตบุ๊ก (Notebook) หน่วยประมวลผลอินเทลคอร์ไอไฟวความเร็วสัญญาณนาฬิกา 2.30 กิกะเฮิร์ท (Intel CORE i5 2.30 GHz)
- หน่วยความจำสำรอง (RAM) ขนาด 4 กิกะไบต์ (4 GB)
- ฮาร์ดดิสก์ (Hard disk) ขนาด 300 กิกะไบต์ (300 GB)

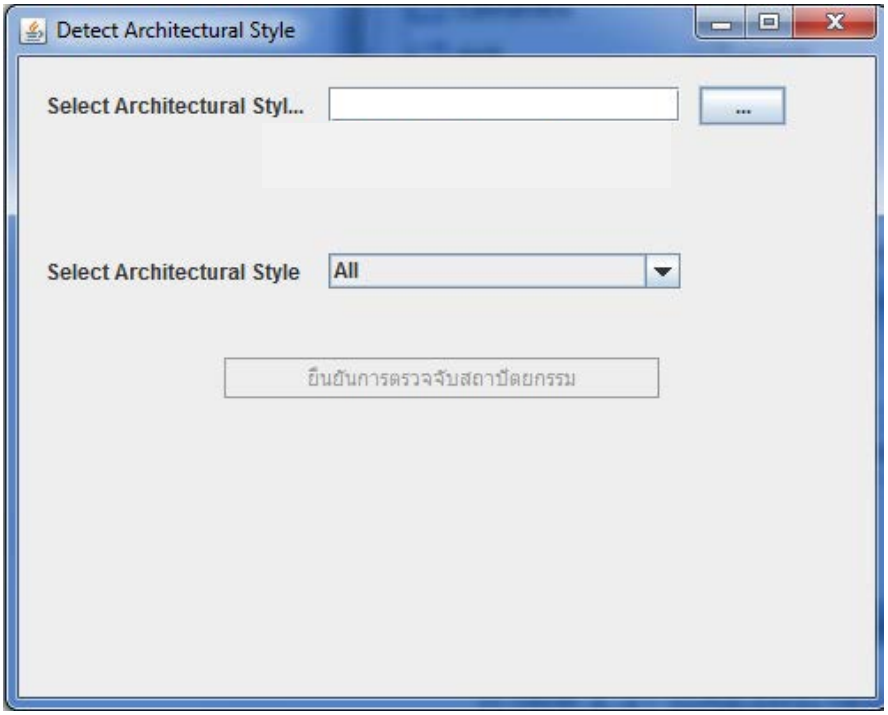
2) ซอฟต์แวร์ (Software)

- ระบบปฏิบัติการ (Operation System) ไมโครซอฟต์วินโดวส์ 7 (MS Windows 7)
- โปรแกรมอาร์คสตูดิโอ เวอร์ชัน 4 (ArchStudio 4)
- โปรแกรมอีคลิปส์ จูโน่ ไอดีอี (Eclipse Juno IDE)
- โปรแกรมอีดิทพลัส เวอร์ชัน 6.3 (EditPlus 6.3)
- โปรแกรมแจสเปอร์ รีพอร์ต เวอร์ชัน 4.8.0 (JasperReport 4.8.0)

4.3.2 ส่วนต่อประสานกับผู้ใช้เครื่องมือ

โครงสร้างส่วนต่อประสานกับผู้ใช้เครื่องมือในการตรวจจับรูปแบบสถาปัตยกรรมซอฟต์แวร์ด้วยกราฟแกรมมา อธิบายด้วยแผนภาพส่วนประกอบ (Component Diagram) ซึ่งเป็นแผนภาพที่ใช้ในการแสดงความสัมพันธ์ระหว่างส่วนประกอบต่างๆ ของเครื่องมือที่ได้ทำการ พัฒนาขึ้น จากการเพิ่มเติมส่วนต่อขยายของเครื่องมือ ArchStudio 4 โดยสามารถแบ่งการทำงานหลักคือ ส่วนการตรวจจับสถาปัตยกรรม โดยหน้าจอการทำงานจะประกอบไปด้วย

1) ส่วนการตรวจจับสถาปัตยกรรม ผู้ใช้งานสามารถทำการตรวจจับสถาปัตยกรรมซอฟต์แวร์ ผ่านหน้าจอสำหรับการตรวจจับสถาปัตยกรรมซอฟต์แวร์ โดยผู้ใช้งานจะต้องทำการเลือกแฟ้มข้อมูลสถาปัตยกรรมซอฟต์แวร์ที่ได้ออกแบบไว้ ในรูปแบบภาษาเอกซ์เอดีแอล จากนั้นทำการเลือกยืนยันการนำเข้าข้อมูล หลังจากการนำเข้าข้อมูลสถาปัตยกรรมซอฟต์แวร์แล้ว ก็ทำการเลือกสถาปัตยกรรมที่ต้องการตรวจสอบหรือตรวจสอบทั้งหมดโดยการเลือก ทั้งหมด (All) จากนั้นเลือกยืนยันการตรวจจับ หลังจากเครื่องมือทำงานเสร็จ เครื่องมือจะทำการแสดงผลลัพธ์ของการตรวจจับโดยการเปิดแฟ้มข้อมูล PDF ที่เครื่องมือได้สร้างขึ้นมา เพื่อดูผลการตรวจจับ ซึ่งแสดงดังรูปที่ 4.52 - รูปที่ 4.53



รูปที่ 4.52 ตัวอย่างส่วนต่อประสานกับผู้ใช้งานสำหรับการตรวจจับสถาปัตยกรรม

รูปที่ 4.53 ตัวอย่างผลลัพธ์การตรวจจับสถาปัตยกรรมในรูปแบบรายงาน

จากตัวอย่างรายงานแสดงสถาปัตยกรรมที่ตรวจจับพบ โดยสามารถอธิบายได้ว่า มีการตรวจพบสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ โดยคอมโพเนนท์ที่เริ่มต้นตรวจพบ คือ Start และ คอมโพเนนท์ที่มีลักษณะสถาปัตยกรรมไปป์และ

ฟิลเตอร์ลักษณะไทป์ไปป์ ที่เชื่อมต่อกับคอมโพเนนท์ Start คือ คอมโพเนนท์ Set1 และคอมโพเนนท์ Set3 เป็นต้น

บทที่ 5
การทดสอบเครื่องมือ

การทดสอบเครื่องมือเป็นการทดสอบเพื่อตรวจสอบความถูกต้องของเครื่องมือ ซึ่งมีความสามารถในการตรวจจับสถาปัตยกรรมซอฟต์แวร์ โดยสามารถดูรายละเอียดวิธีการใช้งานเครื่องมือดัง ภาคผนวก ก ตลอดจนสรุปผลการค้นหาสถาปัตยกรรมซอฟต์แวร์ โดยกล่าวถึงรายละเอียดของการทดสอบและผลของการทดสอบ มีรายละเอียดดังต่อไปนี้

5.1 การทดสอบเครื่องมือ

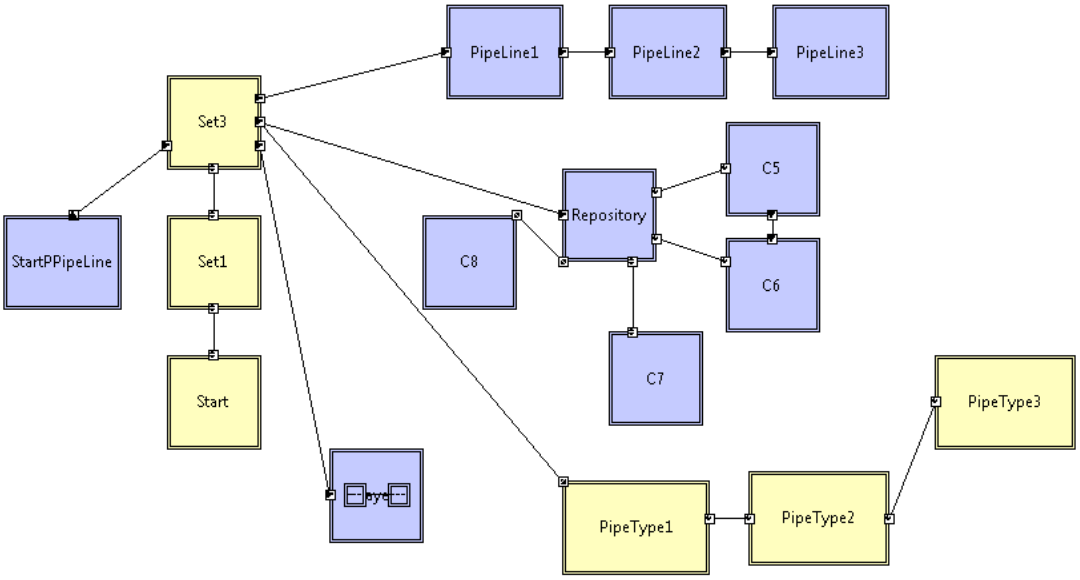
ผู้วิจัยได้ทำการแบ่งการทดสอบตามกรณีทดสอบ ทั้งหมดจำนวน 2 กรณีทดสอบ ซึ่งแยกตามลักษณะกรณีทดสอบดังนี้

1. ตัวอย่างการตรวจจับสถาปัตยกรรมซอฟต์แวร์ ซึ่งมีลักษณะของสถาปัตยกรรมที่กำหนดไว้ในการทดสอบครบทุกเงื่อนไขและทุกส่วนประกอบของสถาปัตยกรรมสามารถตรวจพบสถาปัตยกรรมซอฟต์แวร์ได้เกิน 1 รูปแบบและมีจำนวนชั้นของสถาปัตยกรรมในรูปแบบลำดับชั้นไม่เกิน 1 ชั้นจำนวน 1 กรณีทดสอบ
2. ตัวอย่างการตรวจจับสถาปัตยกรรมซอฟต์แวร์ที่มาจากระบบงานจริงของระบบโทรคมนาคม

โดยหลังจากทำการทดสอบด้วยข้อมูลตามกรณีต่างๆที่ได้กล่าวถึงข้างต้นแล้ว สามารถแสดงตัวอย่างผลการทดสอบแยกตามกรณีทดสอบได้ดังนี้

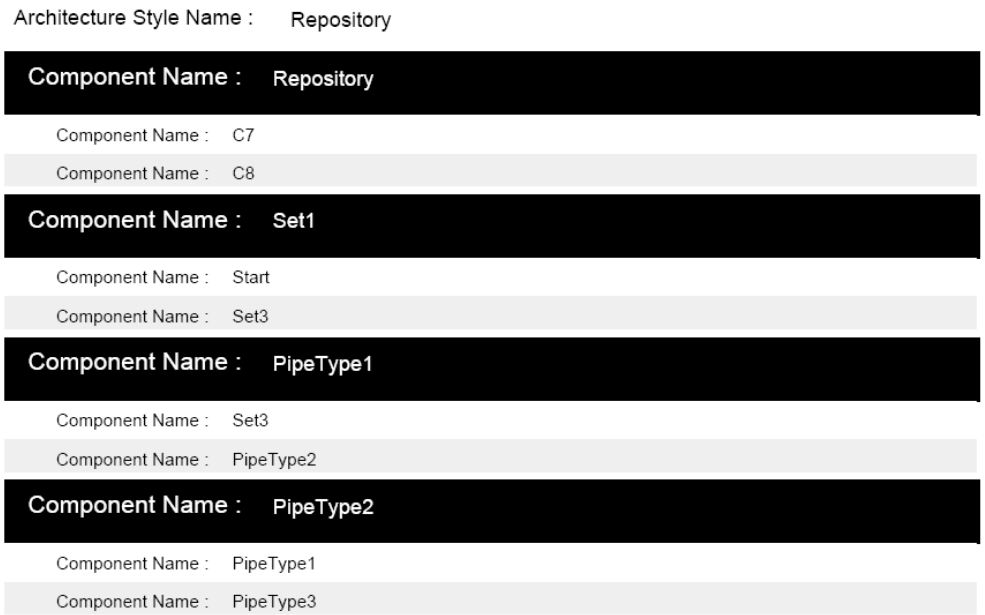
5.1.1 ตัวอย่างการตรวจจับสถาปัตยกรรมซอฟต์แวร์แบบที่หนึ่ง

กรณีทดสอบที่หนึ่งมีการนำเข้าโครงสร้างของสถาปัตยกรรมต้นแบบ ซึ่งมีลักษณะของสถาปัตยกรรมที่กำหนดไว้ในการทดสอบครบทุกเงื่อนไขและทุกส่วนประกอบของสถาปัตยกรรมสามารถตรวจพบสถาปัตยกรรมซอฟต์แวร์ได้เกิน 1 รูปแบบ เช่น จุด A ตรวจพบสถาปัตยกรรมในรูปแบบรวมศูนย์กลางแล้ว จุด A อาจตรวจพบสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้นได้อีก และมีจำนวนชั้นของสถาปัตยกรรมสถาปัตยกรรมในรูปแบบลำดับชั้นไม่เกิน 1 ชั้นจำนวน 1 กรณีทดสอบ ซึ่งสามารถแสดงลักษณะของสถาปัตยกรรมที่ได้กล่าวไปข้างต้นได้ดังรูปที่ 5.1



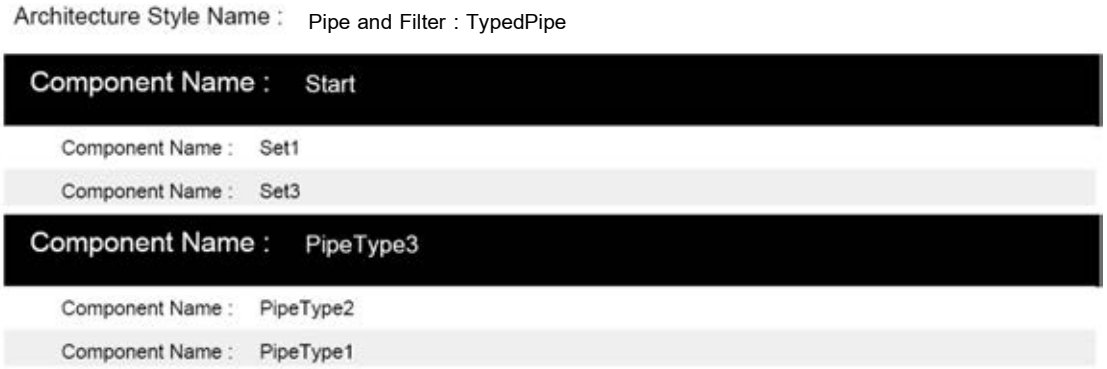
รูปที่ 5.1 ภาพรวมของสถาปัตยกรรมในกรณีทดสอบที่หนึ่ง

หลังจากสถาปนิกซอฟต์แวร์ได้ใช้เครื่องมือที่พัฒนาขึ้นในการตรวจจับรูปแบบของสถาปัตยกรรม โดยตัวอย่างผลลัพธ์การทดสอบเครื่องมือสามารถแสดงได้ดัง



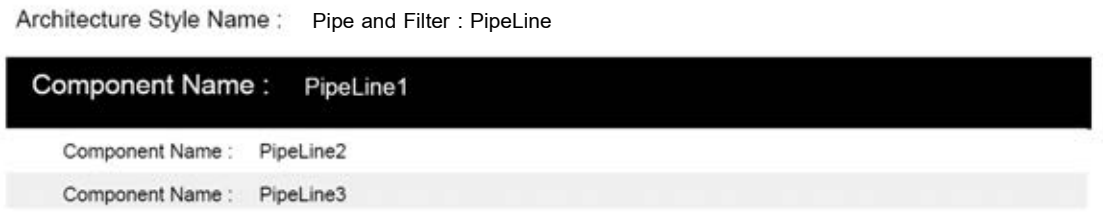
รูปที่ 5.2 รูปผลลัพธ์ของการตรวจจับสถาปัตยกรรมรายงานการพบสถาปัตยกรรมสถาปัตยกรรมในรูปแบบรวมศูนย์กลาง

จะสามารถสรุปผลการตรวจจับสถาปัตยกรรมได้ว่า ตรวจพบสถาปัตยกรรมรูปแบบ Repository ซึ่งมีคอมโพเนนท์ที่ชื่อ Repository เป็นคอมโพเนนท์หลักและมีคอมโพเนนท์ที่เป็นส่วนประกอบของสถาปัตยกรรมในรูปแบบรวมศูนย์กลาง มีชื่อว่า C7 และ C8 ซึ่งที่เครื่องมือไม่ได้ตรวจจับพบคอมโพเนนท์ C6 และ C5 ด้วยนั้น เนื่องจากตามทฤษฎีของ David garland และ Mary Shaw [6] ได้กล่าวไว้ว่าสถาปัตยกรรมในรูปแบบรวมศูนย์กลาง จะเชื่อมต่อแลกข้อมูลไปกลับกับคอมโพเนนท์ที่เป็นศูนย์กลางเท่านั้น จะไม่สามารถแลกเปลี่ยนข้อมูลระหว่างคอมโพเนนท์ได้ และสามารถตรวจพบสถาปัตยกรรมในรูปแบบรวมศูนย์กลาง ซึ่งมีคอมโพเนนท์ที่ชื่อ Set1 เป็นคอมโพเนนท์หลักและมีคอมโพเนนท์ที่เป็นส่วนประกอบของสถาปัตยกรรมในรูปแบบรวมศูนย์กลาง มีชื่อว่า Start และ Set3 แล้วยังสามารถตรวจพบสถาปัตยกรรมในรูปแบบรวมศูนย์กลาง ซึ่งมีคอมโพเนนท์ที่ชื่อ PipeType1 และ PipeType3 เป็นคอมโพเนนท์หลักและมีคอมโพเนนท์ที่เป็นส่วนประกอบของสถาปัตยกรรมในรูปแบบรวมศูนย์กลาง มีชื่อว่า Set3 , PipeType1,PipeType2 และ PipeType3 อีกด้วย



รูปที่ 5.3 รูปผลลัพธ์ของการตรวจจับสถาปัตยกรรมรายงานการพบสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไทป์ไปป์

จากรูปที่ 5.3 จะสามารถสรุปผลการตรวจจับสถาปัตยกรรมได้ว่า ตรวจพบสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ ซึ่งมีคอมโพเนนท์ที่ชื่อ Start เป็นคอมโพเนนท์เริ่มต้นและมีคอมโพเนนท์ที่เป็นส่วนประกอบของสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ มีชื่อว่า set1 และ set3 ซึ่งมีชนิดของอินเตอร์เฟสที่เหมือนกัน คือ TCPIP และยังมีการตรวจจบสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ ซึ่งมีคอมโพเนนท์ที่ชื่อ PipeType3 เป็นคอมโพเนนท์เริ่มต้นและมีคอมโพเนนท์ที่เป็นส่วนประกอบของสถาปัตยกรรมที่ชื่อ PipeType2 และ PipeType1



รูปที่ 5.4 รูปผลลัพธ์ของการตรวจจับสถาปัตยกรรมรายงานการพบสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์

จากรูปที่ 5.4 จะสามารถสรุปผลการตรวจจับสถาปัตยกรรมได้ว่า ตรวจพบสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ ซึ่งมีการตรวจพบสถาปัตยกรรมรูปแบบนี้สองจุด คือจุดแรกเริ่มต้นจากคอมโพเนนท์ PipeLine1 และมีคอมโพเนนท์ที่มีลักษณะของสถาปัตยกรรมภายใต้ PipeLine1 คือ คอมโพเนนท์ PipeLine2 และ PipeLine3 โดยมีคอมโพเนนท์ PipeLine1 เป็นหลัก และจากรูปที่ 5.4 ชุดของคอมโพเนนท์ Start , Set1 และ Set3 จะไม่ถูกนับเป็นสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ด้วย เนื่องจากลักษณะของสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์เป็นลักษณะที่มี คอมโพเนนท์ต่อกันเป็นสายเชื่อมต่อไปในทิศทางเดียว และมีช่องทางเข้าและออกเพียงช่องทางเดียว ดังนั้นคอมโพเนนท์ Set3 จึงไม่ถือว่าเป็นมีคุณสมบัติของสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ เพราะมีช่องทางเชื่อมต่อออกไปจากตัวคอมโพเนนท์มากกว่าหนึ่งเส้นทาง และเมื่อ Set3 ไม่ถือว่ามีคุณสมบัติของสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ แล้วดังนั้นคอมโพเนนท์ที่เหลือ คือ Set1 และ Start จึงไม่ถือว่ามีคุณสมบัติของสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ ไปด้วยเนื่องจากมี คอมโพเนนท์ไม่ถึงจำนวนขั้นต่ำของคุณสมบัติของสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ ที่จะต้องต่อกันอย่างน้อยสามคอมโพเนนท์

รูปที่ 5.5 รูปผลลัพธ์ของการตรวจจับสถาปัตยกรรมรายงานการพบสถาปัตยกรรมสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้น

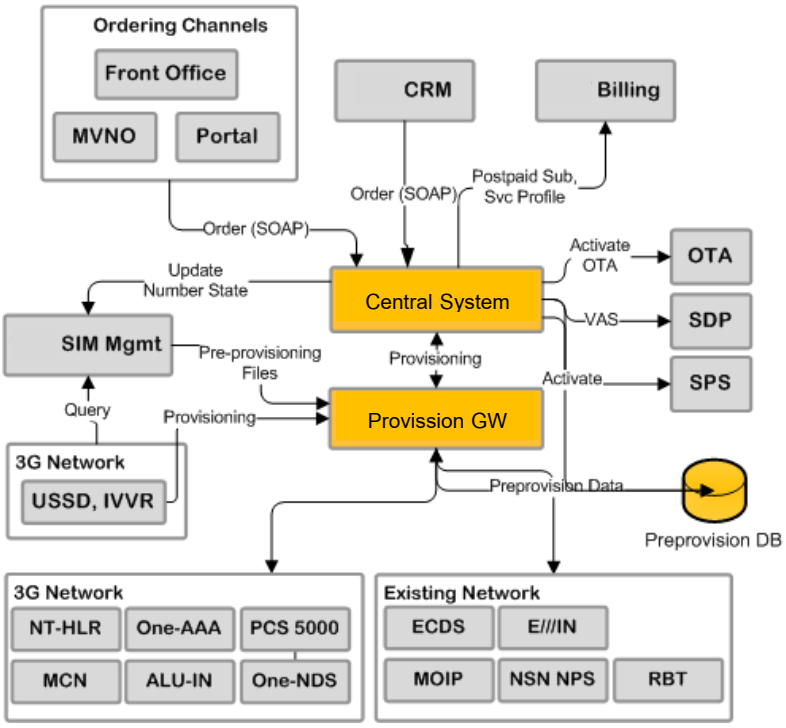
จากรูปที่ 5.5 จะสามารถสรุปผลการตรวจจับสถาปัตยกรรมได้ว่า ตรวจพบสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้น ซึ่งมีคอมโพเนนท์ที่ชื่อ Set3 เป็นคอมโพเนนท์หลักและมีคอมโพเนนท์ที่เป็นส่วนประกอบของสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้นมีชื่อว่า PipeType1 และ Repository ซึ่งอยู่ภายใต้อินเตอร์เฟส ชื่อว่า [New Interface] ของ คอมโพเนนท์ที่ชื่อ Set3

รูปที่ 5.6 รูปผลลัพธ์ของการตรวจจับสถาปัตยกรรมรายงานการพบสถาปัตยกรรมในรูปแบบลำดับชั้น

จากรูปที่ 5.6 จะสามารถสรุปผลการตรวจจับสถาปัตยกรรมได้ว่า ตรวจพบสถาปัตยกรรมในรูปแบบลำดับชั้น ซึ่งมีคอมโพเนนท์ที่ชื่อ Layer เป็นคอมโพเนนท์ที่มีคุณลักษณะของสถาปัตยกรรมในรูปแบบลำดับชั้นอยู่ ซึ่งภายใต้คอมโพเนนท์ Layer นั้นจะประกอบไปด้วย คอมโพเนนท์ ClientSubLayer และ ServerSubLayer ซึ่งถือว่าอยู่ความลึกระดับหนึ่งถ้านับจากคอมโพเนนท์ Layer ที่อยู่ในระดับความลึก 0 หรือ ไม่มีความลึก

5.1.2 ตัวอย่างการตรวจจับสถาปัตยกรรมซอฟต์แวร์ที่มาจากระบบงานจริงของระบบโทรคมนาคม

ในกรณีทดสอบที่สองนั้นได้นำลักษณะของสถาปัตยกรรมซอฟต์แวร์ของระบบเปิดใช้บริการโทรศัพท์มือถือ 3G ซึ่งมีลักษณะของสถาปัตยกรรมซอฟต์แวร์แสดงดังรูปที่ 5.7



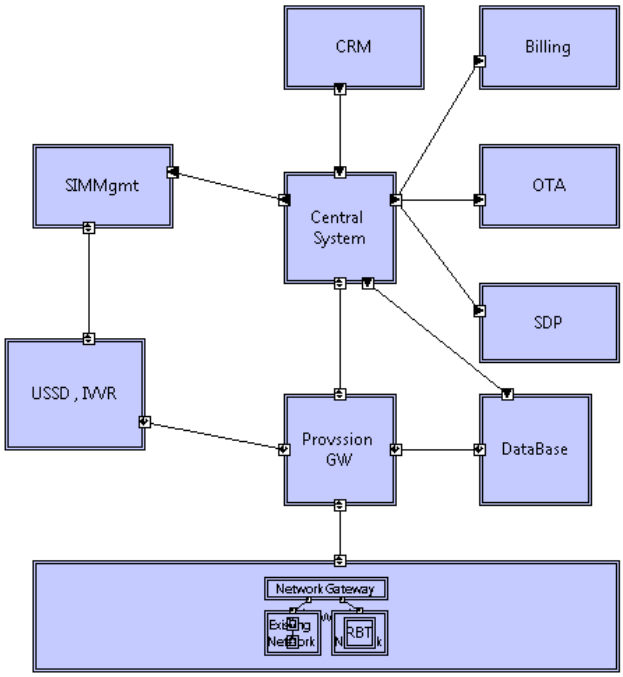
รูปที่ 5.7 ภาพรวมของสถาปัตยกรรมในกรณีทดสอบที่สอง

ซึ่งจากรูปที่ 5.7 โครงสร้างของระบบการเปิดใช้บริการของโทรศัพท์มือถือ 3G จะประกอบไปด้วยส่วนต่างที่สำคัญดังต่อไปนี้

- ระบบซีอาร์เอ็ม (CRM) เป็นระบบศูนย์บริการที่สามารถทำการเปิดบริการ, ยกเลิกบริการ หรือทำการปลดการระงับการใช้ ให้กับผู้ใช้ได้
- ระบบออกบิล (Billing) เป็นระบบสมัครใช้บริการของผู้ใช้รายเดือน ซึ่งจะมีการเก็บข้อมูลรายละเอียดของลูกค้าไว้
- ช่องทางภายนอก (Open front channels) เป็นระบบจากภายนอกที่จะใช้บริการ
- ระบบ โอทีเอ (OTA) เป็นระบบที่ส่งข้อมูลการติดตั้งระบบไปให้กับเครื่องปลายทาง

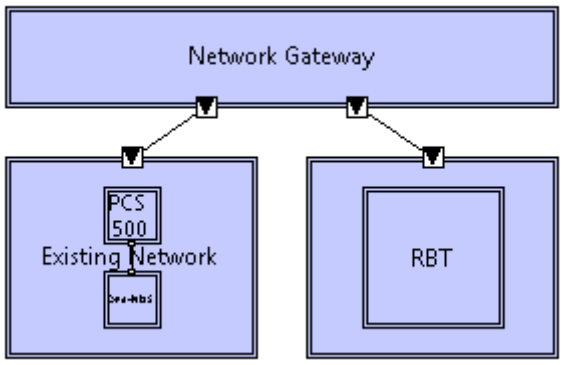
- ระบบ เอสดีพี (SDP) เป็นระบบที่ไว้สำหรับทำการหักเงินค่าใช้จ่ายต่างๆ
- ระบบ เอสพีเอส (SPS) เป็นระบบที่ไว้รวมบริการต่างๆ โดยผู้ใช้สามารถทำรายการผ่านโปรแกรมที่อยู่บนมือถือแล้วระบบเอสพีเอส จะเป็นตัวแยกว่าจะไปทำงานใดต่อไป
- ระบบ ยูเอสเอสดี (USSD) เป็นระบบที่ใช้สำหรับโต้ตอบด้วยข้อความ
- ระบบ ไอวีวีอาร์ (IVVR) เป็นระบบที่ใช้สำหรับโต้ตอบด้วยข้อความเสียง
- ระบบ จัดการซิมการ์ด (SIMMgmt) เป็นระบบที่ใช้สำหรับเก็บข้อมูลของเบอร์ และค่าซิมการ์ด

ซึ่งจากโครงสร้างของระบบการเปิดใช้บริการของโทรศัพท์มือถือ ที่ได้นำเสนอในรูปที่ 5.7 นั้น ผู้เชี่ยวชาญด้านวิศวกรรมซอฟต์แวร์ได้ทำการเขียนโครงร่างดังกล่าวให้อยู่ในรูปของภาษาที่ใช้อธิบายสถาปัตยกรรม โดยในที่นี้ผู้เชี่ยวชาญด้านวิศกรรมซอฟต์แวร์ได้ใช้ภาษาเอ็กซ์เอดีแอลเป็นภาษาที่ไว้ใช้อธิบายโครงร่างของสถาปัตยกรรมดังรูปที่ 5.8



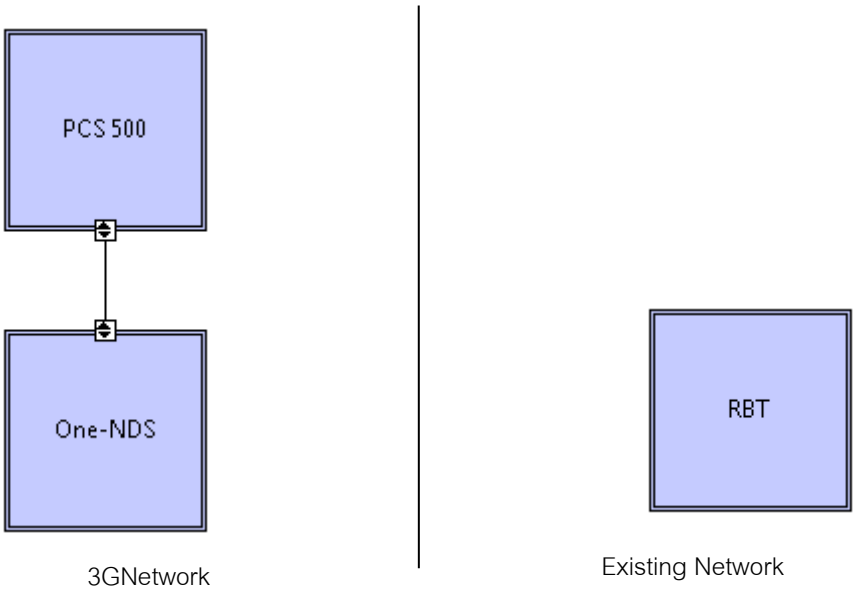
รูปที่ 5.8 ภาพรวมของสถาปัตยกรรมในกรณีทดสอบที่สองโดยสร้างจาก ArchStudio4

โดยภายในคอมโพเนนท์ Network จะประกอบด้วยโครงสร้างภายในดังรูปที่ 5.9



รูปที่ 5.9 ภาพของสถาปัตยกรรมภายในคอมโพเนนท์ Network ในกรณีทดสอบที่สองโดยสร้างจาก ArchStudio4

และภายในคอมโพเนนท์ Existing Network และคอมโพเนนท์ 3GNetwork จะประกอบด้วยโครงสร้างภายในดังรูปที่ 5.10

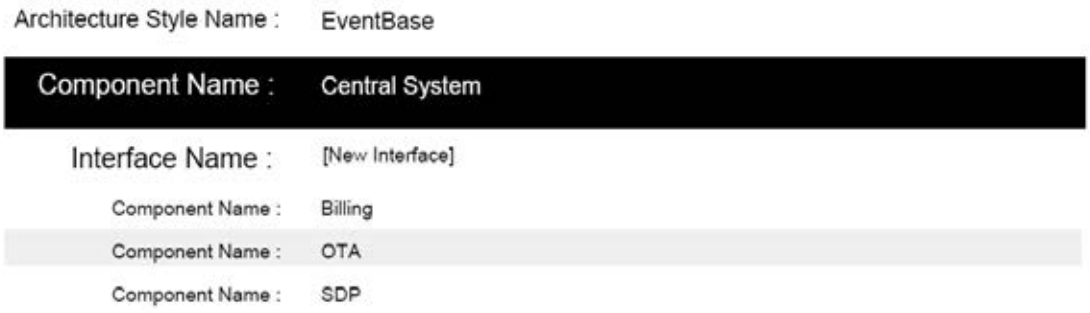


รูปที่ 5.10 ภาพของสถาปัตยกรรมภายในคอมโพเนนท์ Existing Network และ 3GNETWORK ในกรณีทดสอบที่สองโดยสร้างจาก ArchStudio4

หลังจากสถาปนิกซอฟต์แวร์ได้ใช้เครื่องมือที่พัฒนาขึ้นในการตรวจจับรูปแบบของสถาปัตยกรรมเพื่อตรวจสอบนำรูปแบบสถาปัตยกรรมที่ตรวจพบมาหาจุดบกพร่องเพื่อแก้ไขให้ระบบทำงานอย่างมีประสิทธิภาพมากที่สุด โดยตัวอย่างผลลัพธ์การทดสอบเครื่องมือสามารถแสดงได้ดังรูปที่ 5.11

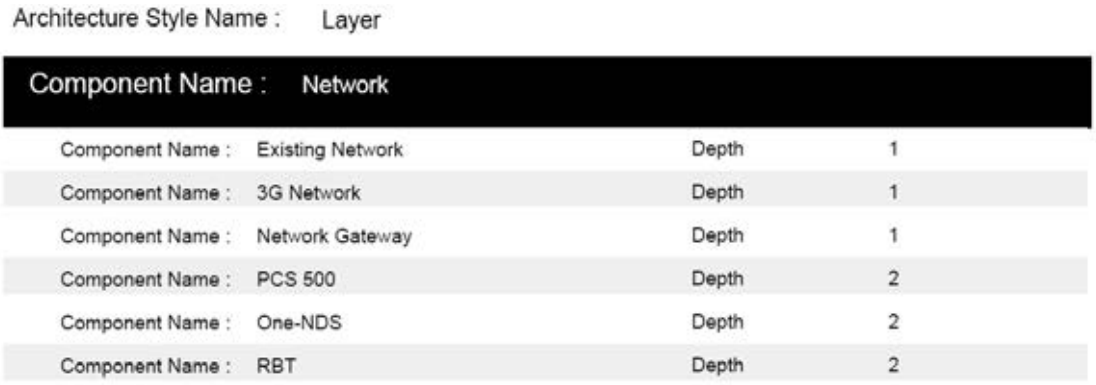
รูปที่ 5.11 รูปผลลัพธ์ของการตรวจจับสถาปัตยกรรมรายงานการพบสถาปัตยกรรมในรูปแบบรวมศูนย์กลางในกรณีทดสอบที่สอง

จากรูปที่ 5.11 จะสามารถสรุปผลการตรวจจับสถาปัตยกรรมได้ว่า ตรวจพบสถาปัตยกรรมในรูปแบบรวมศูนย์กลาง ซึ่งมีคอมโพเนนท์ที่ชื่อ Provssion GW เป็นคอมโพเนนท์หลักและมีคอมโพเนนท์ที่เป็นส่วนประกอบของสถาปัตยกรรมในรูปแบบรวมศูนย์กลางมีชื่อว่า Network และ USSI,IVVR เป็นส่วนประกอบและจากรายงานยังได้ตรวจพบสถาปัตยกรรมในรูปแบบรวมศูนย์กลางซึ่งมีคอมโพเนนท์ที่ชื่อ USSD,IVVR เป็นคอมโพเนนท์หลักและมีคอมโพเนนท์ที่เป็นส่วนประกอบของสถาปัตยกรรมในรูปแบบรวมศูนย์กลาง มีชื่อว่า SIMMgmt และ ProvssionGW เป็นส่วนประกอบ



รูปที่ 5.12 รูปผลลัพธ์ของการตรวจจับสถาปัตยกรรมรายงานการพบสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้นในกรณีทดสอบที่สอง

จากรูปที่ 5.12 จะสามารถสรุปผลการตรวจจับสถาปัตยกรรมได้ว่า ตรวจพบสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้น ซึ่งมีคอมโพเนนท์ที่ชื่อ Central System เป็นคอมโพเนนท์หลักและมีคอมโพเนนท์ที่เป็นส่วนประกอบของสถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้นมีชื่อว่า Billing , OTA และ SDP ซึ่งอยู่ภายใต้อินเตอร์เฟสที่ชื่อว่า [New Interface] ของคอมโพเนนท์ที่ชื่อ Central System



รูปที่ 5.13 รูปผลลัพธ์ของการตรวจจับสถาปัตยกรรมรายงานการพบสถาปัตยกรรมลำดับชั้นในกรณีทดสอบที่สอง

จากรูปที่ 5.13จะสามารถสรุปผลการตรวจจับสถาปัตยกรรมได้ว่า ตรวจพบสถาปัตยกรรมในรูปแบบลำดับชั้น ซึ่งมีคอมโพเนนท์ที่ชื่อ Network เป็นคอมโพเนนท์ที่มีคุณลักษณะของสถาปัตยกรรมรูปแบบลำดับชั้นอยู่ ซึ่งภายใต้คอมโพเนนท์ Layer นั้นจะประกอบไปด้วยคอมโพเนนท์ Existing Network และ 3G Network ซึ่งถือว่าอยู่ความลึกระดับหนึ่ง และในExisting Network ยัง

มีคุณสมบัติสถาปัตยกรรมรูปแบบลำดับชั้นอีกซึ่งภายใต้ Existing Network นั้นจะประกอบด้วยคอมโพเนนท์ PCS500 และ One-NDS อยู่ซึ่งถือว่าอยู่ในความลึกระดับสอง และใน3G Network ยังมีคุณสมบัติของสถาปัตยกรรมในรูปแบบลำดับชั้น อีกซึ่งภายใต้ 3G Network นั้นจะประกอบด้วยคอมโพเนนท์ RBT ซึ่งถือว่าอยู่ในความลึกระดับสองเช่นกัน

บทที่ 6
สรุปผลวิทยานิพนธ์

ในบทนี้กล่าวถึงผลสรุปวิทยานิพนธ์ ประโยชน์ที่ได้รับจากวิทยานิพนธ์ ข้อจำกัดของเครื่องมือ และแนวทางในการพัฒนาวิทยานิพนธ์ต่อไปในอนาคต ซึ่งมีรายละเอียดดังต่อไปนี้

6.1 สรุปผลวิทยานิพนธ์

วิทยานิพนธ์นี้ได้นำเสนอวิธีการตรวจจับสถาปัตยกรรมซอฟต์แวร์ที่ใช้เป็นทางเลือกสำหรับการตรวจสอบสถาปัตยกรรมซอฟต์แวร์ที่ใช้ในปัจจุบัน วิธีการตรวจสอบดังกล่าวได้รับการสนับสนุนการทำงานโดยเครื่องมือที่พัฒนาขึ้น ซึ่งมีชื่อว่า "xADL : Software Architectural Style Detection Tools - xADLArchDetectTools" ซึ่งการตรวจจับสถาปัตยกรรมทำให้สามารถทำการปรับปรุงลักษณะของสถาปัตยกรรมซอฟต์แวร์ได้ง่ายขึ้น โดยใช้เทคนิคกราฟแกรมมาเข้ามาประยุกเพิ่มเติม โดยงานวิจัยนี้สามารถตรวจจับรูปแบบสถาปัตยกรรมได้ทั้งหมด 4 รูปแบบ คือ สถาปัตยกรรมในรูปแบบรวมศูนย์กลาง สถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้น สถาปัตยกรรมไปป์และฟิลเตอร์ และ สถาปัตยกรรมในรูปแบบลำดับชั้น ซึ่งจากงานวิจัยสามารถสร้างคำนิยามได้ทั้งหมด 13 คำนิยาม และได้กฎในการตรวจจับสถาปัตยกรรมซอฟต์แวร์ทั้งหมด 27 กฎ โดยการทดสอบเครื่องมือที่ใช้ในการตรวจจับสถาปัตยกรรมซอฟต์แวร์ กระทำโดยการกำหนดกรณีทดสอบทั้งหมด 2 กรณีทดสอบ ที่ได้ผลการทดสอบโดยการตรวจพบทุกสถาปัตยกรรมที่ปรากฏในกรณีทดสอบแต่วิธีการตรวจสอบยังมีข้อจำกัดอยู่หลายประการตามที่จะนำเสนอต่อไป

6.2 ประโยชน์ที่ได้รับจากวิทยานิพนธ์

1) สามารถนำผลการตรวจจับสถาปัตยกรรมที่ได้จากการนำเครื่องมือ มาช่วยผู้พัฒนาสถาปัตยกรรมซอฟต์แวร์วิเคราะห์ถึงผลดีและผลเสียของรูปแบบสถาปัตยกรรมที่ได้พัฒนาเพื่อประโยชน์ในการควบคุมคุณลักษณะของระบบ

2) สามารถเพิ่มแนวทางในการแก้ไขปัญหาสำหรับการตรวจสอบสถาปัตยกรรมที่จัดเก็บในรูปแบบของภาษาเอ็กซ์เอดีแอลให้มากยิ่งขึ้น โดยเน้นในด้านการนำไปต่อยอดเพื่อความสมบูรณ์มากยิ่งขึ้น

6.3 ข้อจำกัดของงานวิจัย

1) ข้อมูลของภาษาเอ็กซ์เอดีแอลที่เป็นข้อมูลนำเข้าต้องเป็นข้อมูลที่สร้างโดยใช้ ArchStudio4 เท่านั้น โดยจะใช้หลักไวยากรณ์เวอร์ชัน 2.0
2) การตรวจสอบรูปแบบของสถาปัตยกรรม จะตรวจสอบเฉพาะรูปแบบดังต่อไปนี้
   - สถาปัตยกรรมในรูปแบบรวมศูนย์กลาง
   - สถาปัตยกรรมในรูปแบบการทำงานตามเหตุการณ์ที่เกิดขึ้น
   - สถาปัตยกรรมไปป์และฟิลเตอร์
   - สถาปัตยกรรมในรูปแบบลำดับชั้น
3) แฟ้มข้อมูลสถาปัตยกรรมซอฟต์แวร์นำเข้าประกอบด้วย คอมโพเนนท์ ที่เชื่อมต่อกันด้วยลิ้งก์ โดยไม่ผ่านคอนเนคเตอร์
4) สถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์จะต้องมีจุดเริ่มต้นที่มีคอมโพเนนท์เป็นลักษณะที่มีอินเตอร์เฟส มีทิศทาง out อย่างเดียวอย่างน้อยหนึ่ง Interface และไม่มีทิศทาง In หรือ Inout ในคอมโพเนนท์นั้น
5) สถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ จะไม่สนใจอินเตอร์เฟสที่มีลักษณะ Inout
6) สถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ จะตรวจสอบเฉพาะคอมโพเนนท์ที่มีอินเตอร์เฟสที่มีชนิดเป็น Inout เท่านั้น
7) สถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไทป์ไปป์ จะต้องมีจุดเริ่มต้นที่มีคอมโพเนนท์เป็นลักษณะที่มีอินเตอร์เฟส มีทิศทาง inout อย่างเดียวอย่างน้อยหนึ่งอินเตอร์เฟสและไม่มีทิศทาง In หรือ Inout ในคอมโพเนนท์นั้น
8) การตรวจจับสถาปัตยกรรมสามารถตรวจจับได้เฉพาะในระดับโครงสร้างของสถาปัตยกรรม
9) การตรวจจับสถาปัตยกรรมในรูปแบบลำดับชั้นไม่สามารถตรวจสอบทิศทางการเรียกข้อมูลของแต่ละลำดับชั้นได้

6.4 แนวทางในการพัฒนางานวิจัยต่อ

1) พัฒนาให้เครื่องมือมีความสามารถในการตรวจจับกับภาษาเอ็กซ์เอดีแอลในเวอร์ชัน 3.0
2) พัฒนาความสามารถของเครื่องมือเพิ่มเติม เพื่อให้ตรวจสอบสถาปัตยกรรมอื่นๆ เพิ่มเติมได้

3) พัฒนาให้การตรวจจับสถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ ครอบคลุมคอมโพเนนท์เป็นลักษณะที่มีอินเตอร์เฟสมีทิศทาง out อย่างเดียวอย่างน้อยหนึ่งอินเตอร์เฟสและไม่มีทิศทาง In หรือ Inout ในคอมโพเนนท์นั้น
4) สถาปัตยกรรมไปป์และฟิลเตอร์ลักษณะไปป์ไลน์ จะไม่สนใจอินเตอร์เฟสที่มีลักษณะ Inout

รายการอ้างอิง

[1] E. M. Mahdieh Alemi, and Hassan Rashidi. Software Architecture: A Survey and Classification. In Second International Conference on Communication Software and Networks, 454-460. Singapore : , 2010.

[2] Frederic D. McKenzie, Mikel D. Petty and Qingwen Xu. Usefulness of Software Architecture Description Languages for Modeling and Analysis of Federates and Federation Architectures. In Society for Computer Simulation International , 559 – 576. San Diego, CA, USA, 2004

[3] Frederic D. McKenzie, Mikel D. Petty and Qingwen Xu. Usefulness of Software Architecture Description Languages for Modeling and Analysis of Federates and Federation Architectures. In Society for Computer Simulation International San Diego, CA, USA, 559 – 576. USA : , 2004

[4] Surya Prakash Kotha, Mtech, CSE, and IIT Kanpur. xADL – A Better way to Describe Architecture, 1-9. ,2009.

[5] Mouhamadou Lamine BA, Talel Abdessalem, and Pierre Senellart. Towards a Version Control Model with Uncertain Data. In CIKM Conference on Information and Knowledge Management, 43-50. USA : ACM New York, 2011

[6] David Garlan, and Mary Shaw. An Introduction to Software Architecture. 1st ed. USA : Carnegie Mellon University, 1994.

[7] Jun Kong, Kang Zhang, Jing Dong, and Guanglei Song. A Graph Grammar Approach to Software Architecture Verification and Transformation. In COMPSAC '03 Proceedings of the 27th Annual International Conference on Computer Software and Applications, 492. USA : IEEE Computer Society, 2003.

[8] “ArchStudio and Myx”. [Online]. Available: http://www.isr.uci.edu/projects/archstudio/myx.html 2012.

[9] Martin, John C. Introduction to Languages and the Theory of Computation. 4th ed. USA : McGraw-Hill , 2012.

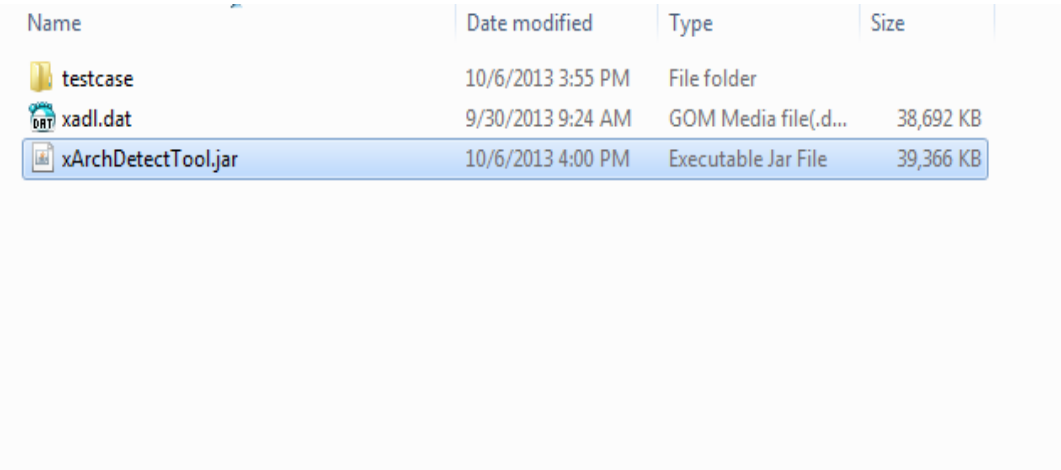
[10] Yoshihiro Adachi and Yuichi Nakajima. A Context-Sensitive NCE Graph Grammar and its Parsability. In Proceedings 2000 IEEE international symposium on visual languages , 111 - 118.USA : IEEE Computer Society, 2000.

ภาคผนวก

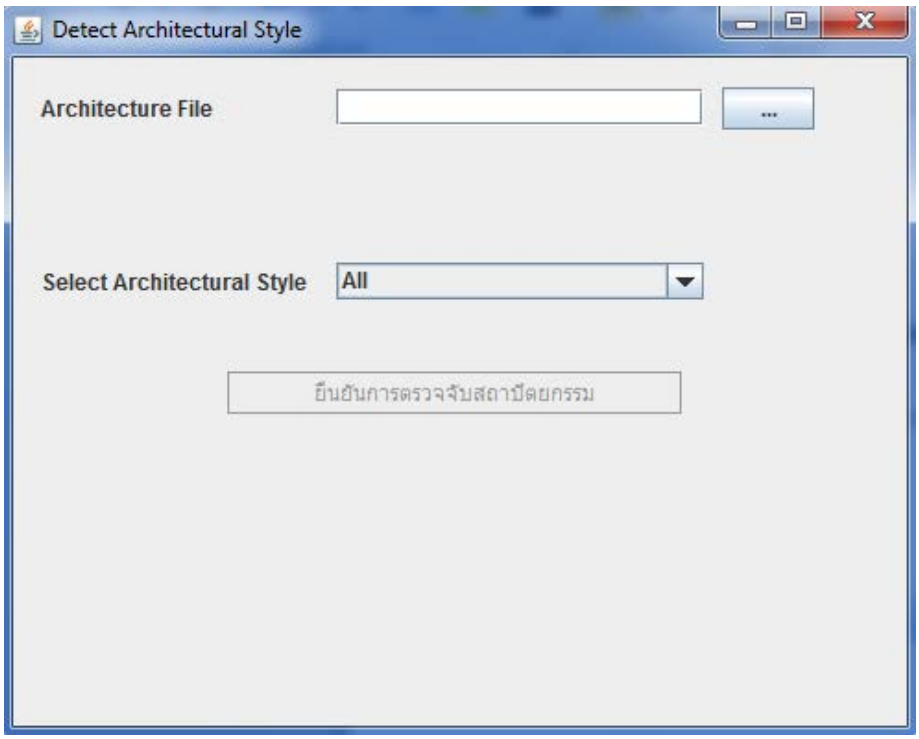
ภาคผนวก ก

วิธีการใช้เครื่องมือในการตรวจจับสถาปัตยกรรม

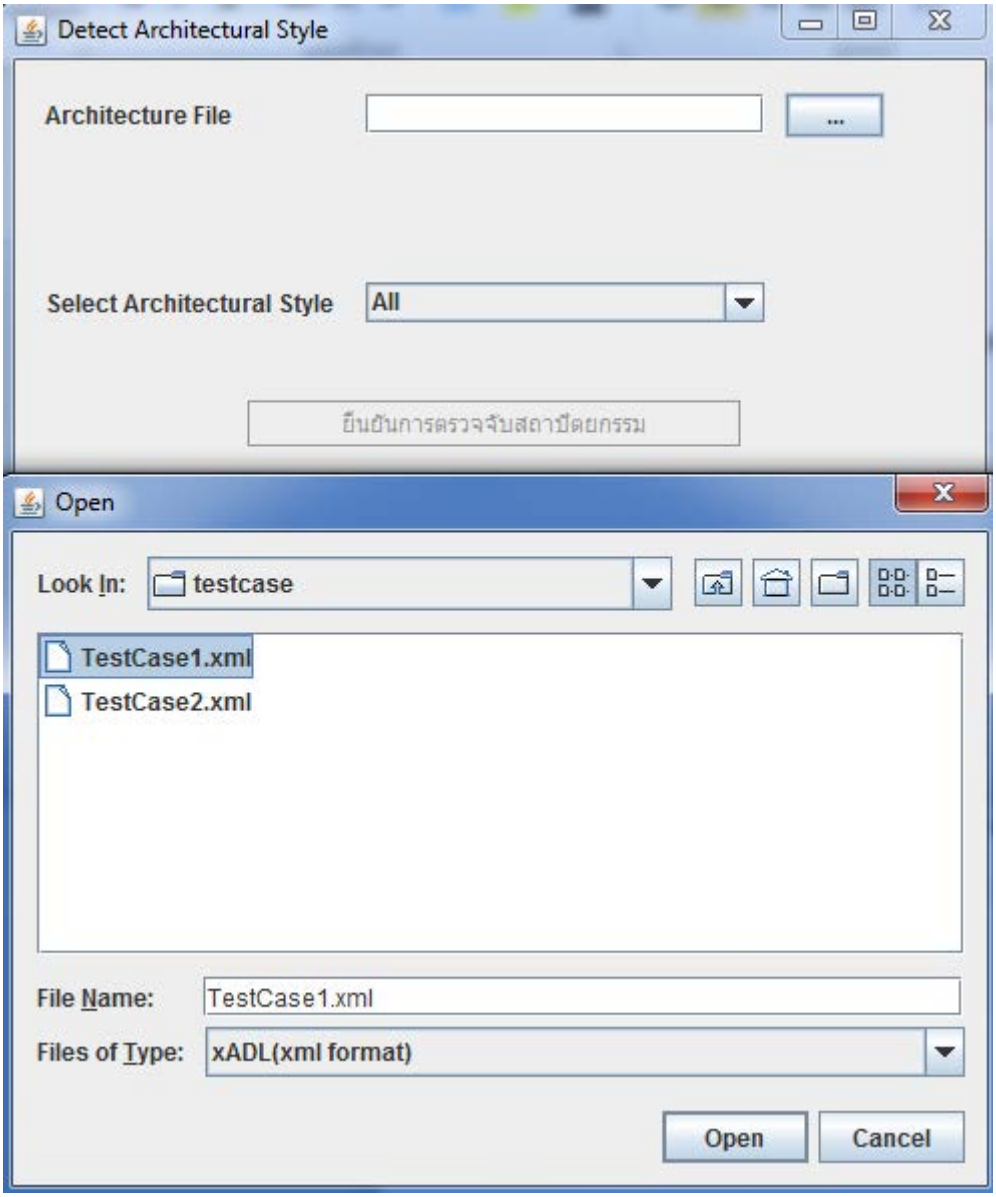
สำหรับวิธีการใช้เครื่องมือในการตรวจจับสถาปัตยกรรม แสดงดังภาพที่ ก-1



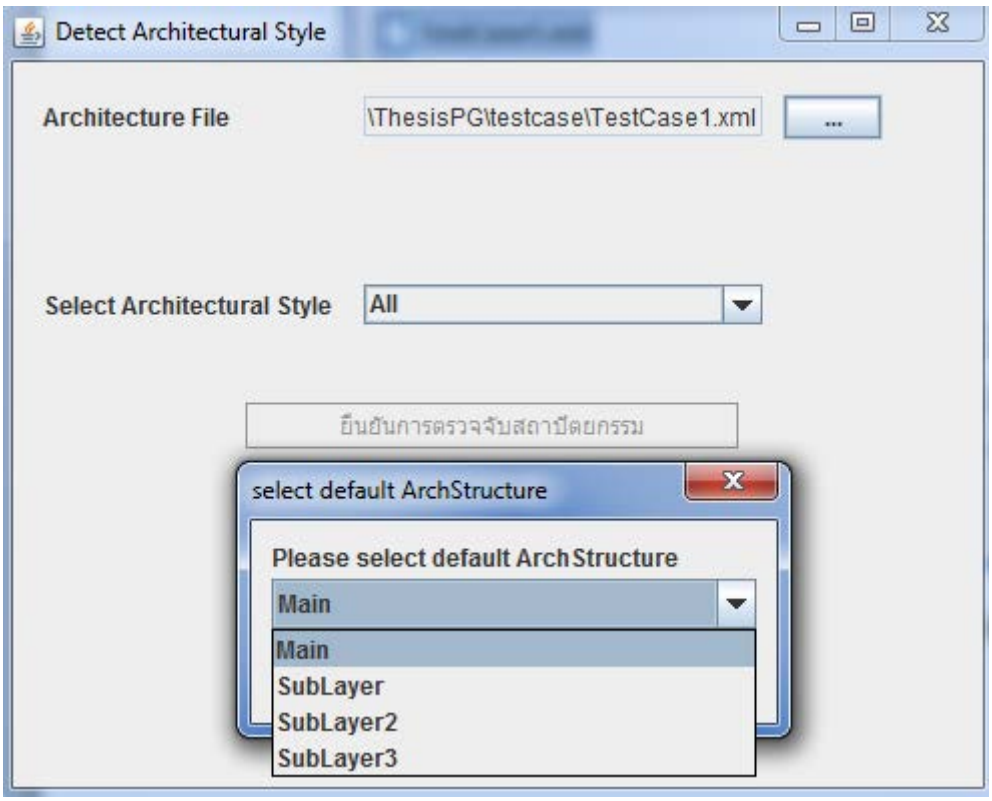
รูปที ก- 1 ภาพที่ทำการเปิดโปรแกรมผ่านทาง xArchDetectTool.jar เพื่อเปิดโปรแกรม



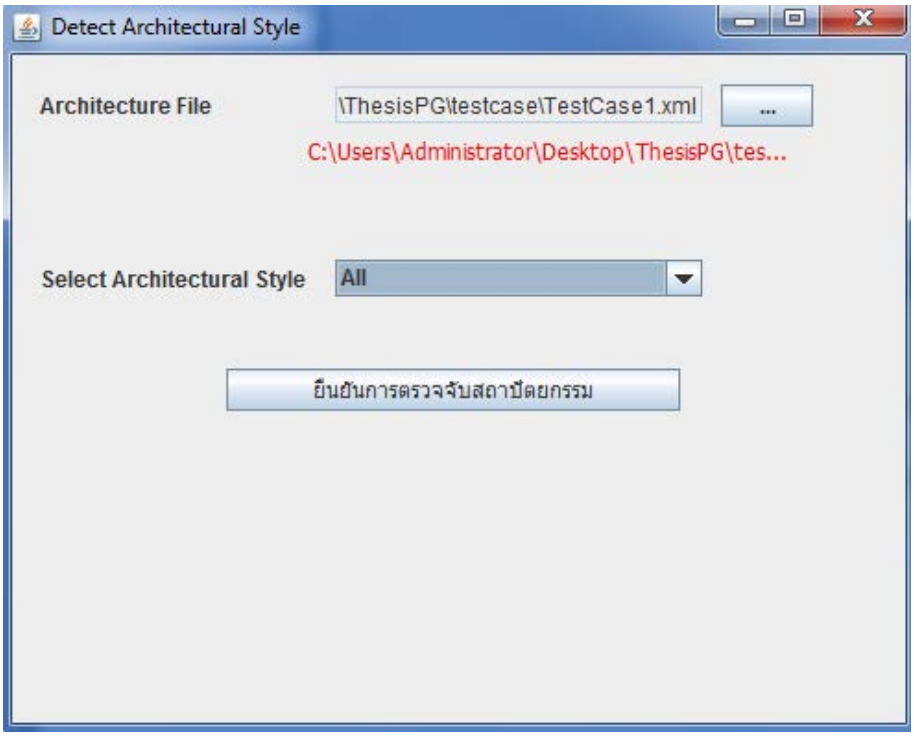
รูปที ก- 2 รูปที่ได้หลังจากการเปิดโปรแกรม



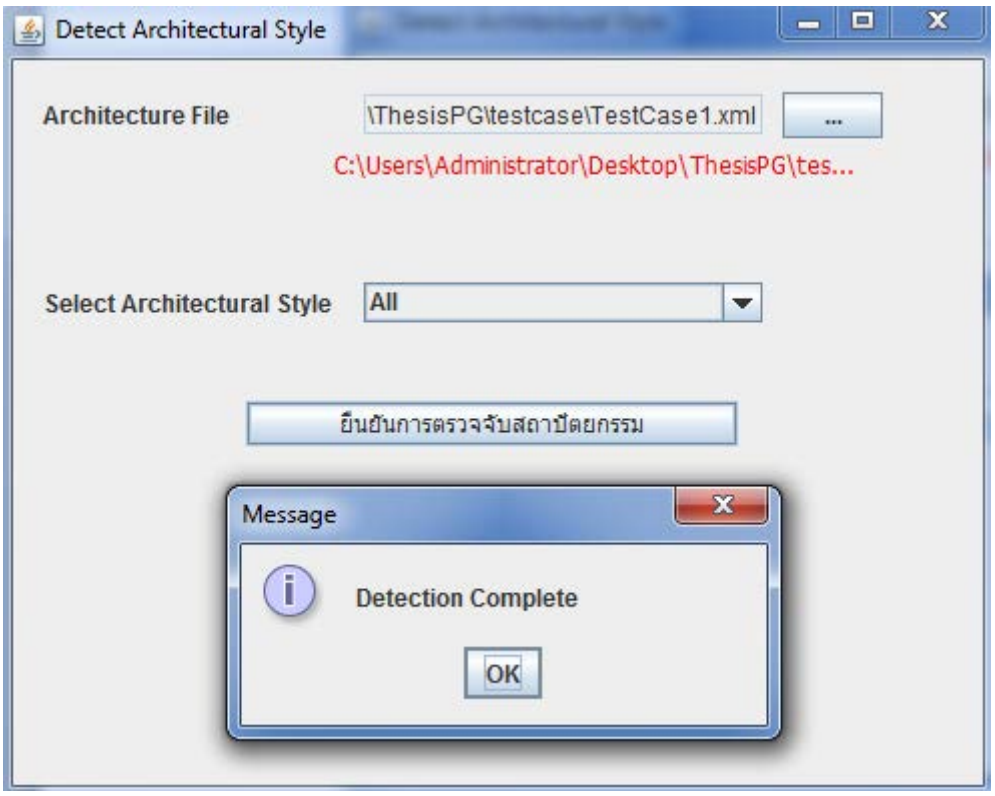
รูปที ก- 3 ทำการเลือกสถาปัตยกรรมที่ต้องการตรวจจับ



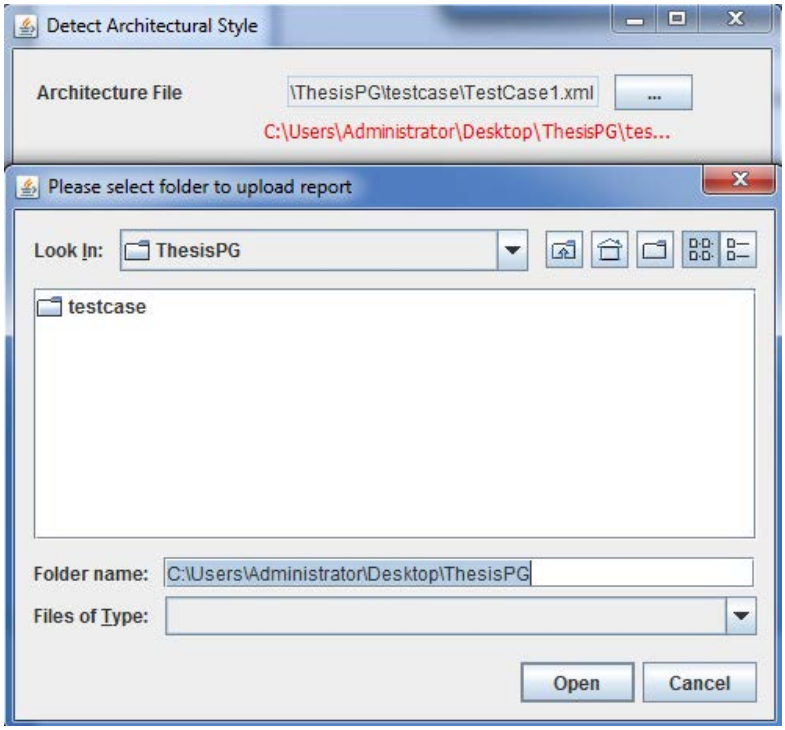
รูปที ก- 4 ทำการเลือกสถาปัตยกรรมตั้งต้น



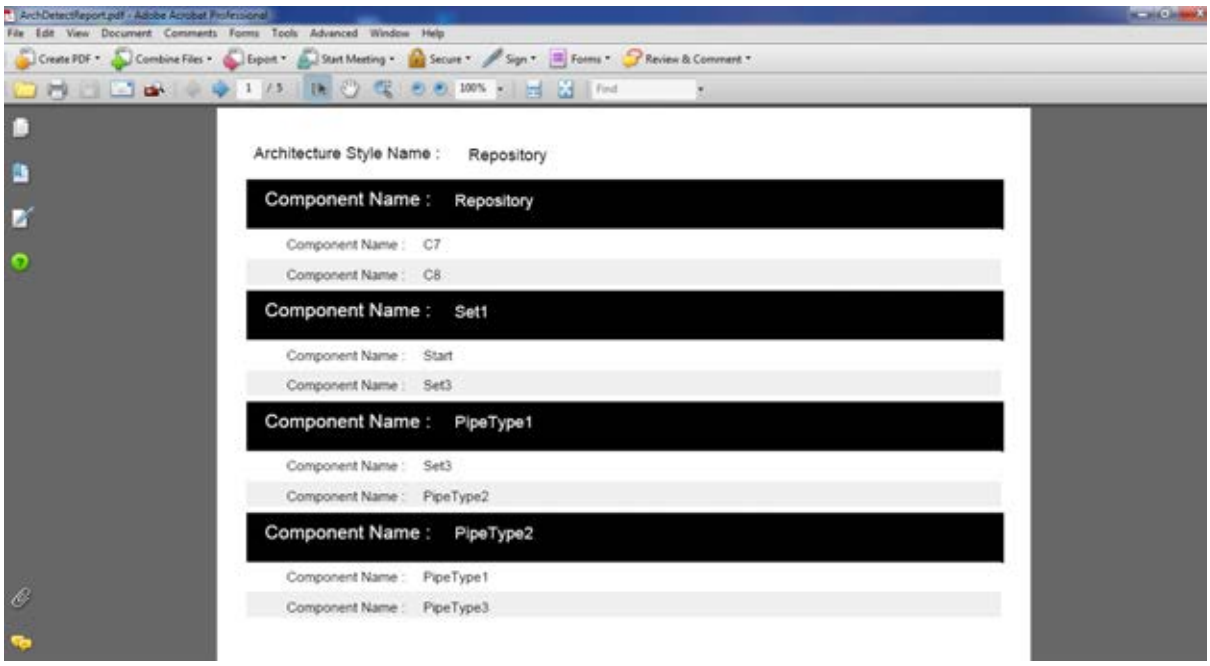
รูปที ก- 5 รูปหลังทำการเลือกสถาปัตยกรรมที่ต้องการทำการตรวจจับเรียบร้อยแล้ว



รูปที ก- 6 ทำการตรวจจับสถาปัตยกรรมโดยการกดปุ่ม “ยืนยันการตรวจจับสถาปัตยกรรม” จะขึ้นข้อความดังภาพ



รูปที ก- 7 ทำการเลือกที่เก็บผลลัพธ์ของการตรวจจับสถาปัตยกรรม



รูปที ก- 8 ระบบจะทำการเปิดโปรแกรม PDF เพื่อแสดงผลลัพธ์จากการตรวจสอบ

ประวัติผู้เขียนวิทยานิพนธ์

นายทรงพล ทองคำ เกิดเมื่อวันที่ 11 พฤษจิกายน พุทธศักราช 2529 ที่จังหวัดยะลา สำเร็จการศึกษาในระดับปริญญาวิทยาศาสตร์บัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ จากคณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์ และได้เข้าศึกษาต่อในระดับปริญญาวิทยาศาสตร์มหาบัณฑิต สาขาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย