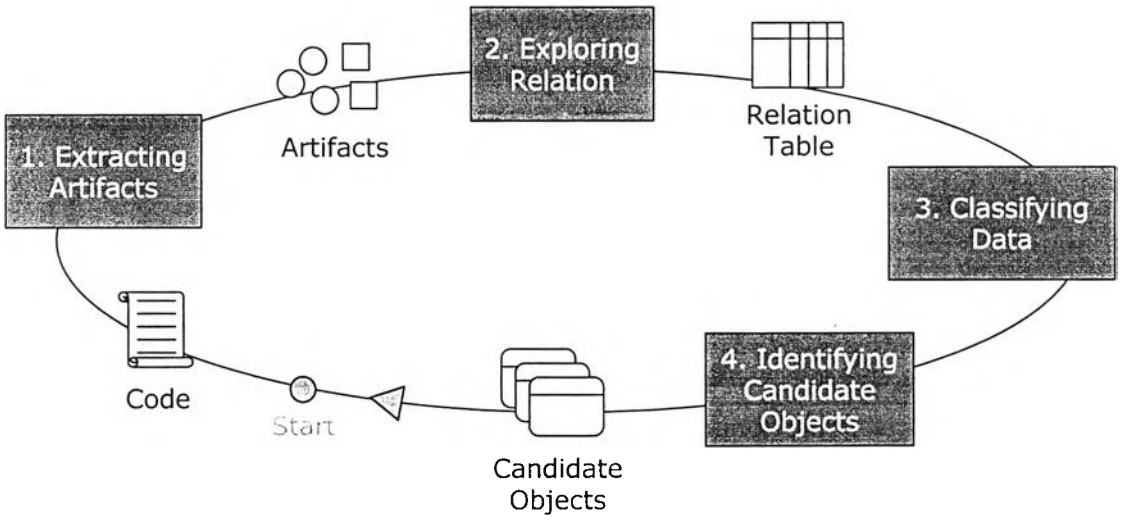
บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 การระบุวัตถุซอฟต์แวร์ที่เป็นไปได้

การระบุวัตถุซอฟต์แวร์ที่เป็นไปได้ หมายถึง กระบวนการค้นหาและดัดแปลงส่วนประกอบของโปรแกรมเชิงโครงสร้างให้สามารถใช้ในโปรแกรมเชิงวัตถุได้ การระบุวัตถุซอฟต์แวร์ที่เป็นไปได้ประกอบด้วยขั้นตอน 4 ขั้น คือ (1) การคัดแยกส่วนประกอบเดิม (2) การค้นหาความสัมพันธ์ระหว่างส่วนประกอบเดิม (3) การจัดเรียงส่วนประกอบเดิม และ (4) การระบุวัตถุซอฟต์แวร์ที่เป็นไปได้ ดังแสดงในรูปที่ 2.1



รูปที่ 2.1 ขั้นตอนการระบุวัตถุซอฟต์แวร์ที่เป็นไปได้

1) การคัดแยกส่วนประกอบเดิม (Extracting Artifacts)

การคัดแยกส่วนประกอบเดิม หมายถึง การนำโปรแกรมต้นฉบับมาวิเคราะห์เพื่อคัดแยกส่วนประกอบเดิมที่ต้องการออกมา สามารถทำได้โดยสร้างโปรแกรมประเภทตัวแจง (Parser) ขึ้นมาช่วยทำงาน ส่วนประกอบเดิมที่ถูกคัดแยกแบ่งออกเป็น 2 ประเภท ได้แก่ ส่วนข้อมูล (Data Artifact) และส่วนคำสั่ง (Operation Artifact) ซึ่งมีรายละเอียดดังต่อไปนี้

a. ส่วนข้อมูล คือ ส่วนประกอบที่มีหน้าที่เก็บข้อมูลในโปรแกรมเชิงโครงสร้าง อาจเทียบได้กับสเตทในโปรแกรมเชิงวัตถุ รูปแบบของส่วนข้อมูลมีได้ต่างๆ กัน เช่น สตรักท์ (Struct), ตัวแปรโกลบอล (Global Variable), เรคคอร์ด (Record) ฯลฯ

b. ส่วนคำสั่ง คือ ส่วนประกอบที่มีหน้าที่ทำงานตามคำสั่งของผู้เขียนโปรแกรม รูปแบบของส่วนคำสั่งมีได้ต่างๆ กัน เช่น ฟังก์ชัน (Function), โพรซีเจอร์ (Procedure), สับรูทีน (Subroutine) ฯลฯ

2) การค้นหาความสัมพันธ์ระหว่างส่วนประกอบเดิม (Exploring Relation)

การค้นหาความสัมพันธ์ระหว่างส่วนประกอบเดิม หมายถึง การวิเคราะห์ส่วนประกอบเดิมเพื่อตรวจสอบความเกี่ยวเนื่องที่ส่วนประกอบเหล่านั้นมีต่อกัน ซึ่งสามารถทำได้โดยใช้วิธีการอย่างเช่น การทำโค้ดสไลซ์ซิ่ง (Code Slicing) , การสร้างกราฟอ้างอิง (Reference Graph) ฯลฯ ช่วยในการทำงาน

3) การจัดเรียงส่วนประกอบเดิม

การจัดเรียงส่วนประกอบเดิม หมายถึง การนำส่วนประกอบเดิมที่มีความสัมพันธ์กันมาจัดเรียงให้เป็นกลุ่ม โดยใช้วิธีการวิเคราะห์ข้อมูลอย่างเช่น วิธีการวิเคราะห์คอนเซ็ปต์, วิธีการจัดกลุ่มข้อมูลแบบลำดับชั้น ฯลฯ ช่วยในการทำงาน ทั้งนี้เพื่อทำให้การระบุวัตถุวัตถุซอฟต์แวร์ที่เป็นไปได้มีความสะดวกรวดเร็วมากขึ้น

4) การระบุวัตถุซอฟต์แวร์ที่เป็นไปได้

การระบุวัตถุซอฟต์แวร์ที่เป็นไปได้ หมายถึง การเลือกกลุ่มส่วนคำสั่งที่มีความหมาย (Semantic) มารวมเข้ากับส่วนข้อมูลที่เกี่ยวข้องกันเพื่อประกอบเป็นวัตถุซอฟต์แวร์ที่เป็นไปได้ (อนึ่ง มีข้อสังเกตว่า การทำงานในขั้นตอนนี้เป็นการทำงานที่ขึ้นอยู่กับวิจารณญาณของแต่ละบุคคล (Subjective) ผู้ปฏิบัติงานบางท่านอาจเลือกระบุวัตถุซอฟต์แวร์ที่เป็นไปได้แบบนี้ แต่ผู้ปฏิบัติงานท่านอื่นอาจเลือกแบบที่ต่างออกไปก็เป็นได้ เนื่องจากประเด็นดังกล่าวเป็นปัญหาในขั้นการออกแบบ (Design Level) ซึ่งไม่สามารถชี้ถูกผิดอย่างชัดเจนได้ งานวิจัยที่นำเสนอส่วนใหญ่จึงไม่ค่อยอธิบายรายละเอียดของการทำงานในขั้นตอนนี้โดยมากกล่าวแต่เพียงว่า ให้เลือกกลุ่มส่วนประกอบเดิมที่มีความหมายมาประกอบกัน แล้วสร้างเป็นวัตถุซอฟต์แวร์ที่เป็นไปได้

2.1.2 วิธีการจัดกลุ่มข้อมูลแบบลำดับชั้น

วิธีการจัดกลุ่มข้อมูลแบบลำดับชั้น[2] เป็นวิธีการวิเคราะห์ข้อมูลวิธีการหนึ่งที่ได้รับความเชื่อถือมานานและถูกนำไปประยุกต์ใช้ในงานวิจัยด้านต่างๆ อย่างกว้างขวาง ไม่ว่าจะเป็นงานวิจัยทางเศรษฐศาสตร์ ดาราศาสตร์ และชีววิทยา ฯลฯ การทำงานของวิธีการจัดกลุ่มข้อมูลฯ แบ่งออกเป็น 3 ขั้น ได้แก่ (1) ขั้นเตรียมข้อมูล (2) ขั้นประมวลผล และ (3) ขั้นแสดงผล ซึ่งมีรายละเอียดดังต่อไปนี้

---

[2] นับตั้งแต่นี้ไปจะขอเรียกวิธีการนี้โดยใช้คำว่า วิธีการจัดกลุ่มข้อมูลฯ แทนการเรียกชื่อเต็ม

1) ขั้นเตรียมข้อมูล

การเตรียมข้อมูล หมายถึง การบันทึกรายละเอียดของข้อมูลที่ต้องการจัดกลุ่มลงในเมตริกซ์ (Matrix) เพื่อให้สามารถคำนวณตามกระบวนการของวิธีการจัดกลุ่มข้อมูลฯ ได้ โดยทั่วไป ข้อมูลที่จะนำมาจัดกลุ่มได้ต้องมีคุณสมบัติ (Attribute) อย่างใดอย่างหนึ่งเสมอ ซึ่งอาจจะเป็นอายุ, ความสามารถทางภาษา ฯลฯ ทั้งนี้เพื่อให้แยกแยะได้ว่า ข้อมูลแต่ละอันมีคุณลักษณะ (Characteristic) เหมือนหรือต่างกันอย่างไร

ในการเตรียมข้อมูล จะบันทึกชื่อ (Label) และคุณสมบัติของข้อมูลไว้ในเมตริกซ์ที่เรียกว่า เมตริกซ์ข้อมูล (Data Matrix) ซึ่งเป็นตารางที่แสดงชื่อในคอลัมน์แรก และแสดงค่าคุณสมบัติในคอลัมน์ต่อๆ ไป ดังตัวอย่างในตารางที่ 2.1

ตารางที่ 2.1 ตัวอย่างเมตริกซ์ข้อมูล

Object 1	1.0	0.0	1.0	0.0
Object 2	0.0	1.0	1.0	0.0
Object 3	0.0	1.0	1.0	0.0
Object 4	0.0	0.0	0.0	1.0

ขั้นตอนต่อไปของการเตรียมข้อมูล คือ การนำค่าคุณสมบัติที่แสดงไว้ในเมตริกซ์ข้อมูลมาคำนวณเพื่อหาค่าระยะทาง (Distance) ค่าระยะทาง หมายถึง ตัวเลขที่บอกปริมาณความแตกต่างระหว่างข้อมูลคู่หนึ่งๆ การคำนวณค่าระยะทางสามารถทำได้สองแบบ ได้แก่ การคำนวณโดยปฏิบัติ (Treat) ต่อคุณสมบัติทุกอย่างเท่าเทียมกัน (Unweighting) และการคำนวณโดยปฏิบัติต่อคุณสมบัติแตกต่างกัน (Weighting) การพิจารณาว่าการคำนวณแบบใดมีความเหมาะสมมากกว่า ต้องคำนึงถึงทั้งลักษณะของคุณสมบัติและจุดมุ่งหมายของการจัดกลุ่มข้อมูล ยกตัวอย่างเช่น ถ้าเป็นการจัดกลุ่มเมืองหลวงของประเทศต่างๆ อาจกำหนดให้ปฏิบัติต่อพิกัดรุ้งและพิกัดแวงอย่างเท่าเทียมกัน แต่ถ้าเป็นการจัดกลุ่มนักบาสเก็ตบอล ก็อาจกำหนดให้ส่วนสูงมีความสำคัญมากกว่าคุณสมบัติอื่นๆ

การคำนวณค่าระยะทางโดยปฏิบัติต่อคุณสมบัติแตกต่างกัน ทำได้โดยเลือกใช้สูตรคำนวณค่าระยะทางที่อนุญาตให้ใส่ค่าน้ำหนัก วิทยานิพนธ์นี้เลือกใช้สูตรที่เรียกว่า สูตรยูคลิเดียนแบบให้ค่าน้ำหนัก (Weighted Euclidean Distance Method) เพราะเป็นสูตรเดียวที่สามารถคำนวณได้ทั้งแบบให้ค่าน้ำหนัก (โดยกำหนดให้น้ำหนักของคุณสมบัติแต่ละอย่างมีค่าแตกต่างกัน) และแบบไม่ให้ค่าน้ำหนัก (โดยกำหนดให้น้ำหนักของคุณสมบัติทั้งหมดมีค่าเท่ากับ 1.0) เพื่อช่วยอำนวยความสะดวกการพัฒนาโปรแกรม รายละเอียดของสูตรยูคลิเดียนแบบให้ค่าน้ำหนักเป็นดังนี้

$$d(i, j) = \left(\sum_{k=1}^{p} w_k |x_{ik} - x_{jk}|^2\right)^{1/2} \quad (1)$$

โดยที่

$d(i, j)$ คือ ค่าระยะทางระหว่างข้อมูลลำดับที่ i กับข้อมูลลำดับที่ j

x_{ik} คือ ค่าคุณสมบัติที่ k ของข้อมูลในลำดับที่ i

x_{jk} คือ ค่าคุณสมบัติที่ k ของข้อมูลในลำดับที่ j

w_k คือ ค่าน้ำหนักที่ให้แก่คุณสมบัติที่ k

p คือ จำนวนคุณสมบัติข้อมูล

ตัวอย่างเช่น เมื่อกำหนดให้ค่า $w_k = 1.0$ ทุกค่า k จะสามารถคำนวณค่าระยะทางระหว่าง Object1 กับ Object2 ได้ดังนี้

$$d(1, 2) = \left(\sum_{k=1}^{4} w_k |x_{1k} - x_{2k}|^2\right)^{1/2}$$

$$= (1.0*(1.0-0.0)^2 + 1.0*(0.0-1.0)^2 + 1.0*(1.0-1.0)^2 + 1.0*(0.0-0.0)^2)^{1/2}$$

$$= (1.0 + 1.0 + 0.0 + 0.0)^{1/2}$$

$$= 1.414$$

ภายหลังการคำนวณ จะบันทึกค่าระยะทางที่ได้ลงในเมตริกซ์ความแตกต่าง (Dissimilarity Matrix) เมตริกซ์ชนิดนี้มีลักษณะน่าที่สนใจ คือ มีขนาดเท่ากับ n x n เสมอ (เมื่อ n คือจำนวนข้อมูล) และแสดงข้อมูลไว้เพียงครึ่งเดียว ดังตัวอย่างในตารางที่ 2.2

ตารางที่ 2.2 ตัวอย่างเมตริกซ์ความแตกต่าง

	Object 1	**Object 2**	**Object 3**	**Object 4**
Object 1	X	1.414	1.414	1.732
Object 2		X	0.000	1.732
Object 3			X	1.732
Object 4				X

2) ขั้นประมวลผล

วิธีการจัดกลุ่มข้อมูลฯ จัดเรียงข้อมูลโดยมีลำดับการทำงานดังนี้

a. เริ่มต้นโดยนำข้อมูลไปแยกบรรจุไว้ในคลัสเตอร์ว่าง หนึ่งคลัสเตอร์ต่อหนึ่งข้อมูล
b. ค้นหาคู่ข้อมูลที่มีความแตกต่างกันน้อยที่สุด โดยพิจารณาจากค่าระยะทางที่แสดงไว้ในเมตริกซ์ความแตกต่าง เช่น คู่ Object 2 และ Object 3 ในตารางที่ 2.2 เป็นคู่ที่มีความแตกต่างกันน้อยที่สุด เพราะมีค่าระยะทางน้อยที่สุด คือ 0.000
c. นำคลัสเตอร์ของข้อมูลคู่นั้นมารวมกัน สร้างเป็นคลัสเตอร์อันใหม่ เช่น นำคลัสเตอร์ที่บรรจุ Object 1 มารวมเข้ากับคลัสเตอร์ที่บรรจุ Object 2 สร้างเป็นคลัสเตอร์ Object 2+3
d. ปรับค่าระยะทางระหว่างคลัสเตอร์ใหม่กับคลัสเตอร์อื่นๆ ที่มีอยู่เดิมโดยใช้กฎการปรับค่า (Updating Rule)
e. กลับไปทำขั้นตอนที่ b อีกครั้ง จนกระทั่งข้อมูลทั้งหมดถูกรวมกันไว้ในคลัสเตอร์อันเดียว จึงหยุดการทำงาน

กฎการปรับค่า คือ เกณฑ์สำหรับเลือกค่าระยะทางของสมาชิกในคลัสเตอร์เก่า (ที่ถูกรวมเข้าด้วยกัน) มาใช้เป็นค่าระยะทางของสมาชิกในคลัสเตอร์ใหม่ วิทยานิพนธ์นี้เลือกใช้กฎปรับค่าที่เรียกว่า กฎ

เลือกค่ามากสุด (Maximum Distance Rule) เพราะต้องการให้ข้อมูลมีความแตกต่างอย่างเด่นชัด รายละเอียดของกฎเลือกค่ามากสุดเป็นดังนี้

$$d_{max}(C_i, C_j) = \max_{p \in C_i,\, p' \in C_j} |p-p'| \quad (2)$$

โดยที่

$d_{max}(C_i, C_j)$ คือ ค่าระยะทางระหว่างคลัสเตอร์ C_i กับคลัสเตอร์ C_j

$|p-p'|$ คือ ค่าระยะทางระหว่าง p (สมาชิกของคลัสเตอร์ใหม่) กับ p' (คลัสเตอร์อื่นที่มีอยู่เดิม)

ตัวอย่างเช่น การปรับค่าระยะทางระหว่างคลัสเตอร์ Object 2+3 (ใหม่) กับคลัสเตอร์ Object 1 (เก่า) สามารถทำได้ดังนี้

$$d_{max}(C_{2+3}, C_1) = \max_{p \in C2+3,\, p' \in C1} |p-p'|$$
$$= max(1.41, 1.41)$$
$$= 1.41$$

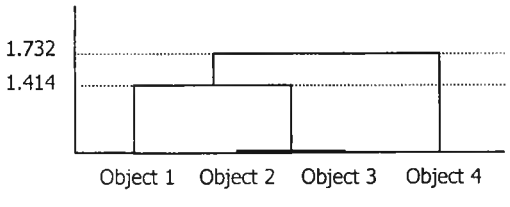
หลังการคำนวณ จะบันทึกค่าระยะทางอันใหม่ไว้ในเมตริกซ์ความแตกต่างอันใหม่ ดังตารางที่ 2.3

ตารางที่ 2.3 ตัวอย่างเมตริกซ์ความแตกต่างหลังการรวมกลุ่มครั้งที่หนึ่ง

	Object 1	**Object 2+3**	**Object 4**
Object 1	X	1.414	1.732
Object 2+3		X	1.732
Object 3			X

3) ขั้นแสดงผล

วิธีการจัดกลุ่มข้อมูลฯ แสดงผลลัพธ์ที่ได้ในแผนภาพต้นไม้ของคลัสเตอร์ (Tree of Cluster) ซึ่งเป็นรูปแบบที่ช่วยให้ทราบได้ง่ายว่า คลัสเตอร์ต่างๆ ถูกสร้างขึ้นและรวมเข้าด้วยกันอย่างไร มีค่าระยะทางระหว่างกันเป็นเท่าไหร่ แผนภาพดังกล่าวมีชื่อเรียกว่า เดนโดแกรม (Dendogram) ดังแสดงในรูปที่ 2.2



รูปที่ 2.2 ตัวอย่างเดนโดแกรม

2.1.3 วิธีการวิเคราะห์คอนเซ็ปต์

วิธีการวิเคราะห์คอนเซ็ปต์เป็นวิธีการที่มีรากฐานมาจากทฤษฎีทางด้านปัญญาประดิษฐ์ ใช้สำหรับจัดเรียงวัตถุที่มีคุณสมบัติตรงกันให้อยู่ในกลุ่มเดียวกัน วิธีการวิเคราะห์คอนเซ็ปต์ทำงานโดยเกี่ยวข้องกับคำนิยามต่างๆ ดังนี้

1) วัตถุ (Object) คือ สิ่งที่ถูกนำมาจัดเรียงเข้าด้วยกัน

2) คุณสมบัติ (Attribute) คือ ลักษณะด้านต่างๆ ของวัตถุที่นำมาจัดเรียง

3) คอนเท็กต์ (Context) หมายถึง ความสัมพันธ์ $C = (O, A, R)$ โดยที่ O คือ เซ็ตวัตถุ, A คือ เซ็ตคุณสมบัติ และ R คือ ความสัมพันธ์ไบนารี (Binary Relation) ระหว่างเซ็ตวัตถุกับเซ็ตคุณสมบัติ โดยทั่วไปจะแสดงคอนเท็กต์ในรูปแบบตาราง ดังตัวอย่างในตารางที่ 2.4 คอลัมน์แรกของตารางแสดงสมาชิกในเซ็ตวัตถุ ส่วนคอลัมน์ที่ 2 ถึงคอลัมน์ที่ 4 แสดงสมาชิกในเซ็ตคุณสมบัติ และสัญลักษณ์ T คือ สิ่งที่บอกให้ทราบว่าวัตถุมีคุณสมบัติตามที่ระบุไว้ในคอลัมน์นั้นๆ

ตารางที่ 2.4 ตัวอย่างคอนเท็กต์

	Attribute 1	**Attribute 2**	**Attribute 3**	**Attribute 4**
Object 1	T		T	
Object 2		T	T	
Object 3		T	T	
Object 4				T

4) $\sigma(X)$ หมายถึง ฟังก์ชันสำหรับค้นหา 'เซ็ตคุณสมบัติ' ของวัตถุ X เมื่อกำหนดให้ $X \subseteq O$ จะสามารถเขียนเป็นสมการของ $\sigma(X)$ ได้ดังนี้

$$\sigma(X) = \{a \in A \mid \forall o \in X : (o, a) \in R\}$$

ตัวอย่างเช่น

$$\sigma(\text{Object 2}) = \{\text{Attribute 1, Attribute2}\}$$

5) $\tau(Y)$ หมายถึง ฟังก์ชันสำหรับค้นหา 'เซ็ตวัตถุ' ของคุณสมบัติ Y เมื่อกำหนดให้ $Y \subseteq A$ จะสามารถเขียนเป็นสมการของ $\tau(Y)$ ได้ดังนี้

$$\tau(Y) = \{o \in O \mid \forall a \in Y : (o, a) \in R\}$$

ตัวอย่างเช่น

$$\tau(\text{Attribute 2}) = \{\text{Object 2, Object 3}\}$$

6) คอนเซ็ปต์ (Concept) หมายถึง คู่ลำดับของเซ็ตวัตถุกับเซ็ตคุณสมบัติ (X, Y) โดยที่ $Y = \sigma(X)$ และ $X = \tau(Y)$ หรือในอีกความหมายหนึ่ง คอนเซ็ปต์ก็คือ กลุ่มใหญ่ที่สุดของวัตถุ (Maximal Collection of Objects) ที่มีคุณสมบัติร่วมกัน

ตัวอย่างเช่น

กลุ่ม ({Object 1}, {Attribute 1, Attribute 3}) เป็นคอนเซ็ปต์ เพราะเป็นกลุ่มใหญ่ที่สุดของวัตถุที่มีคุณสมบัติ Attribute 1 และ Attribute 3

แต่กลุ่ม ({Object 2}, {Attribute 2, Attribute 3}) ไม่ใช่คอนเซ็ปต์ เพราะไม่ได้รวม Object 3 (ซึ่งเป็นวัตถุที่มีคุณสมบัติ Attribute 2 และ Attribute 3 ด้วย) ไว้ในกลุ่มเดียวกัน

7) สับคอนเซ็ปต์ (Subconcept) หมายถึง คอนเซ็ปต์ที่มีลักษณะเป็นกลุ่มย่อยของคอนเซ็ปต์อื่นๆ คอนเซ็ปต์ (X0, Y0) จะเป็นสับคอนเซ็ปต์ของ (X1, Y1) ก็ต่อเมื่อ $X0 \subseteq X1$ หรือ $Y1 \subseteq Y0$

ตัวอย่างเช่น

คอนเซ็ปต์ ({Object 1}, {Attribute 1, Attribute 3}) เป็นสับคอนเซ็ปต์ของ ({Object 1, Object 2, Object 3}, {Attribute 3}) เพราะว่า {Object 1} $\subseteq$ {Object 1, Object 2, Object 3}

8) ซุปเปอร์คอนเซ็ปต์ (Superconcept) หมายถึง คอนเซ็ปต์ที่มีลักษณะเป็นกลุ่มรวมของคอนเซ็ปต์อื่นๆ คอนเซ็ปต์ (X1, Y1) จะเป็นซุปเปอร์คอนเซ็ปต์ของ (X0, Y0) ก็ต่อเมื่อ X0 $\subseteq$ X1 หรือ Y1 $\subseteq$ Y0

ตัวอย่างเช่น

คอนเซ็ปต์ ของ ({Object 1, Object 2, Object 3}, {Attribute 3}) เป็นซุปเปอร์คอนเซ็ปต์ของ ({Object 1}, {Attribute 1, Attribute 3}) เพราะว่า {Object 1} $\subseteq$ {Object 1, Object 2, Object 3}

9) อะตอมมิกคอนเซ็ปต์ (Atomic Concept) คือ คอนเซ็ปต์ที่ได้จากการคำนวณโดยใช้เซ็ตวัตถุที่มีสมาชิกเพียงหนึ่งสมาชิก หรือ ($\tau(\sigma(\{x\})), \sigma(\{x\})$) เมื่อ x คือ วัตถุที่นำมาจัดเรียง

ตัวอย่างเช่น

$(\tau(\sigma(\{Object\ 2\})), \sigma(\{Object\ 2\})) = (\tau(\{Attribute\ 2, Attribute\ 3\}), \sigma(\{Object\ 2\}))$

$= (\{Object\ 2, Object\ 3\}, \{Attribute\ 2, Attribute\ 3\})$

10) บ็อททอมคอนเซ็ปต์ (Bottom Concept) คือ คอนเซ็ปต์ที่แสดงเซ็ตวัตถุที่มีคุณสมบัติครบหมดทุกอย่าง หรือ ($\tau(A), A$) เมื่อ A คือ เซ็ตของคุณสมบัติทั้งหมดในโดเมน

ตัวอย่างเช่น

$(\tau(A), A) = (\tau(\{Attribute\ 1, Attribute\ 2, Attribute\ 3, Attribute\ 4\}), \{Attribute\ 1, Attribute\ 2, Attribute\ 3, Attribute\ 4\})$

$= (\varnothing, \{Attribute\ 1, Attribute\ 2, Attribute\ 3, Attribute4\})$

11) ท็อปคอนเซ็ปต์ (Top Concept) คือ คอนเซ็ปต์ที่แสดงเซ็ตคุณสมบัติที่วัตถุทั้งหมดมีร่วมกัน หรือ ($O, \sigma(O)$) เมื่อ O คือ เซ็ตของคุณสมบัติทั้งหมดในโดเมน

ตัวอย่างเช่น

$(O, \sigma(O)) = (\{Object\ 1, Object\ 2, Object\ 3, Object\ 4\}), \sigma(\{Object\ 1, Object\ 2, Object\ 3, Object\ 4\})$

$= (\{Object\ 1, Object\ 2, Object\ 3, Object\ 4\}, \varnothing)$

12) การจอยน์ (Join) คือ การนำคู่คอนเซ็ปต์มาสร้างเป็นซุปเปอร์คอนเซ็ปต์ โดยวิธีการดังนี้คือ นำเซ็ตคุณสมบัติของคู่ของคอนเซ็ปต์มาทำอินเทอร์เซ็กชัน (Intersection) เพื่อหาคุณสมบัติร่วมของคู่คอนเซ็ปต์ ถ้าผลการอินเทอร์เซ็กชันไม่เป็นเซ็ตว่าง (การจอยน์ประสบความสำเร็จ) ให้นำคุณสมบัติร่วมอันนั้นมาสร้างซุปเปอร์คอนเซ็ปต์ แต่ถ้าผลการอินเทอร์เซ็กชันเป็นเซ็ตว่าง (การจอยน์ล้มเหลว) ก็แสดงว่าไม่สามารถสร้างซุปเปอร์คอนเซ็ปต์จากคู่คอนเซ็ปต์นั้นๆ ได้

ตัวอย่างเช่น

$c1 = (\{Object\ 1\}, \{Attribute\ 1, Attribute\ 3\})$

c2 = ({Object 2, Object 3}, {Attribute 2, Attribute 3})

c1 ∨ c2 = (τ(Attribute 3), {Attribute 3})

= ({Object 1, Object 2, Object 3}, {Attribute 3}) = c4

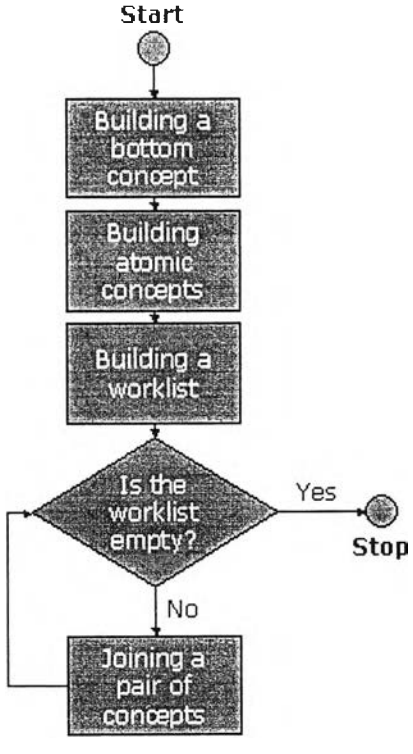
13) เวิร์กลิสต์ (Worklist) หมายถึง กลุ่มของคู่ลำดับคอนเซ็ปต์ (c, c') โดยที่ c ไม่เป็นสับคอนเซ็ปต์ของ c' และ c' ไม่เป็นสับคอนเซ็ปต์ของ c

ตัวอย่างเช่น

Worklist = [(c1, c2), (c1, c3), (c2, c3)]

14) โครงข่ายคอนเซ็ปต์ (Concept Lattice) หมายถึง แผนภาพต้นไม้ที่แสดงความสัมพันธ์ระหว่างคอนเซ็ปต์ทุกคอนเซ็ปต์ในโดเมน

การสร้างคอนเซ็ปต์จากข้อมูลชุดหนึ่งๆ สามารถทำได้โดยปฏิบัติตามขั้นตอน 5 ขั้น (ดังแสดงในรูปที่ 2.3) ซึ่งมีรายละเอียดดังต่อไปนี้



รูปที่ 2.3 ขั้นตอนการสร้างคอนเซ็ปต์

1) สร้างบ็อททอมคอนเซ็ปต์

ตัวอย่างเช่น

BOT = (∅, {Attribute 1, Attribute 2, Attribute 3, Attribute4})

2) สร้างอะตอมมิกคอนเซ็ปต์

ตัวอย่างเช่น

c1 = ({Object 1}, {Attribute 1, Attribute 3})

c2 = ({Object 2, Object 3}, {Attribute 2, Attribute 3})

c3 = ({Object 4}, {Attribute 4})

3) สร้างเวิร์กลิสต์

ตัวอย่างเช่น

Worklist = [(c1, c2), (c1, c3), (c2, c3)]

4) นำคู่คอนเซ็ปต์ออกมาจากเวิร์กลิสต์แล้วจอยน์เข้าด้วยกัน ถ้าการจอยน์ประสบความสำเร็จ ให้เพิ่มคอนเซ็ปต์อันใหม่เข้าไปในเวิร์กลิสต์ ถ้าการจอยน์ไม่ประสบความสำเร็จ ให้ระบุด้วยสัญลักษณ์ T

ตัวอย่างเช่น

c1 ∨ c2 = (π(Attribute 3), {Attribute 3})

= ({Object 1, Object 2, Object 3}, {Attribute 3}) = c4

Worklist = [(c1, c3), (c2, c3), (c3, c4)]

5) ทำข้อ 4) ซ้ำจนกระทั่งเวิร์กลิสต์ว่างเปล่า

c1 ∨ c3 = T

Worklist = [(c2, c3), (c3, c4)]

c2 ∨ c3 = T

Worklist = [(c3, c4)]

c3 ∨ c4 = T

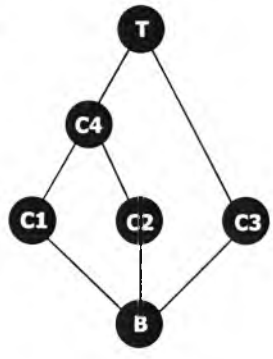
Worklist = [ ]

จากนั้นนำผลที่ได้มาแสดงในตารางคอนเซ็ปต์ ดังตัวอย่างในตารางที่ 2.5

ตารางที่ 2.5 ตัวอย่างตารางคอนเซ็ปต์

TOP	({Object 1, Object 2, Object 3, Object 4}, Ø)
C4	({Object 1, Object 2, Object 3}, {Attribute 3})
C3	({Object 4}, {Attribute 4})
C2	({Object 2, Object 3}, {Attribute 2, Attribute 3})
C1	({Object 1}, {Attribute 1, Attribute 3})
BOT	(Ø, {Attribute 1, Attribute 2, Attribute 3, Attribute4})

และผลลัพธ์เป็นโครงข่ายคอนเซ็ปต์ ดังตัวอย่างในรูปที่ 2.4



รูปที่ 2.4 ตัวอย่างโครงข่ายคอนซ็ปต์

2.2 งานวิจัยที่เกี่ยวข้อง

วิธีการระบุวัตถุซอฟต์แวร์ที่เป็นไปได้แบ่งออกเป็น 2 กลุ่มใหญ่ ได้แก่ กลุ่มที่ไม่ได้ประยุกต์ใช้วิธีการวิเคราะห์ข้อมูล กับกลุ่มที่ประยุกต์ใช้วิธีการวิเคราะห์ข้อมูล รายละเอียดของวิธีการทั้งสองกลุ่มมีดังนี้

2.2.1 กลุ่มที่ไม่ได้ประยุกต์ใช้วิธีการวิเคราะห์ข้อมูล

วิธีการระบุวัตถุซอฟต์แวร์ที่เป็นไปได้ที่อยู่ในกลุ่มนี้ได้แก่ วิธีการในระยะเริ่มแรกซึ่งส่วนใหญ่แล้วมีลักษณะการทำงานคล้ายคลึงกัน กล่าวคือ มีกระบวนการทำงานไม่ซับซ้อน ใช้สติปัญญาของมนุษย์เป็นหลักในการค้นหาความสัมพันธ์และการจัดเรียงข้อมูล ยกตัวอย่างเช่น วิธีการของ Liu กับ Wilde [11] ที่วิเคราะห์หาความสัมพันธ์ระหว่างโครงสร้างข้อมูลแบบกำหนดเอง (User-defined Data Structure) กับสับรูทีน (Subroutine) วิธีการของ Yeh กับคณะ [2] ที่วิเคราะห์หาความสัมพันธ์ระหว่างสับรูทีน กับโครงสร้างข้อมูลแบบกำหนดเองและตัวแปรโกลบอล และวิธีการของ Gall กับ Klosh [12] ซึ่งวิเคราะห์หาความสัมพันธ์ระหว่างสับรูทีน กับเรคคอร์ดที่ถูกเรียกใช้ผ่านทางไฟล์

2.2.2 กลุ่มที่ประยุกต์ใช้วิธีการวิเคราะห์ข้อมูล

วิธีการระบุวัตถุซอฟต์แวร์ที่เป็นไปได้ที่จัดอยู่ในกลุ่มนี้ได้แก่ วิธีการซึ่งนำเอาวิธีการวิเคราะห์ข้อมูลชนิดต่างๆ มาประยุกต์ใช้แทนการทำงานของมนุษย์ เพื่อช่วยให้การจัดเรียงข้อมูลเป็นไปอย่างถูกต้องรวดเร็วมากขึ้น วิธีการที่กล่าวถึงสามารถจำแนกตามชนิดของวิธีการวิเคราะห์ข้อมูลที่นำมาประยุกต์ใช้ได้ดังนี้

1) วิธีการวิเคราะห์กราฟ (Graph Analysis) [13] เป็นวิธีการที่นำเอาอัลกอริทึมวิเคราะห์กราฟเข้ามาช่วยงาน โดยมีขั้นตอนเริ่มจากแปลงโปรแกรมต้นฉบับให้เป็นกราฟชนิดพิเศษที่เรียกว่า StDG (Statements Dependence Graph) นำกราฟที่สร้างขึ้นไปลดรูปเพื่อตัดข้อมูลที่ไม่เกี่ยวข้องออกจนกระทั่งมีคุณสมบัติตรงตามเกณฑ์ที่ตั้งไว้ จึงนำผลลัพธ์ที่ได้แปลงกลับไปเป็นวัตถุซอฟต์แวร์ ข้อดีของวิธีการนี้คือ ค้นหาความสัมพันธ์ในหน่วยย่อยที่สุดของโปรแกรม คือ รหัสคำสั่ง จึงทำให้ได้ผลลัพธ์ที่มีรายละเอียดกว่าวิธีการอื่นๆ อย่างไรก็ตาม วิธีการนี้ก็มีข้อเสียอยู่บ้าง ซึ่งได้แก่ขั้นตอนการแปลงกราฟและการลดรูปกราฟที่มีความยุ่งยากซับซ้อนและใช้เวลานาน
2) วิธีการวิเคราะห์หลักฐาน (Evidence-driven Analysis) [14] เป็นวิธีการที่นำเอาการวัดค่าชนิดต่างๆ

เช่น การวัดจำนวนข้อมูลเรียกเข้าส่งออก (Fan-in, Fan-out), การวัดค่าฟังก์ชันพอยต์ (Function Point) ฯลฯ มาใช้ เพื่อช่วยประเมินคุณภาพของวัตถุซอฟต์แวร์ที่สร้างขึ้น ข้อดีของวิธีการนี้คือ สามารถวัดคุณภาพของวัตถุซอฟต์แวร์ฯ ได้ โดยพิจารณาจากตัวเลขซึ่งถูกคำนวณออกมา อย่างไรก็ตาม วิธีการนี้มีข้อเสียคือ เกณฑ์การให้คะแนนที่ใช้ยังไม่เป็นที่ยอมรับ จึงไม่อาจพิสูจน์ได้ว่าค่าที่คำนวณได้จะเป็นค่าที่ถูกต้องตรงกับความเป็นจริง

3) วิธีการวิเคราะห์คอนเซ็ปต์ [7, 15, 16] เป็นวิธีการที่นำเอาอัลกอริทึมวิเคราะห์คอนเซ็ปต์เข้ามาช่วยทำงาน โดยใช้เป็นเครื่องมือสำหรับจัดเรียงส่วนคำสั่งเป็นกลุ่มๆ เพื่อให้การเลือกส่วนประกอบมาสร้างวัตถุซอฟต์แวร์ฯ ทำได้สะดวกรวดเร็วมากยิ่งขึ้น ข้อดีของวิธีการนี้ คือ มีรูปแบบการแสดงผลลัพธ์ที่บอกรายละเอียดต่างๆ อย่างชัดเจน สำหรับประเด็นข้อบกพร่อง พบว่าวิธีการนี้ไม่สามารถกลั่นกรองผลการทำงานได้ คอนเซ็ปต์ที่ได้ทับซ้อนกัน และทำงานผิดพลาดเมื่อโปรแกรมมีโครงสร้างแบบผูกติด

4) วิธีการแบบผสม (Eclectic Approach) [17] เป็นวิธีการที่ปฏิบัติตามสมมติฐานที่ว่า โปรแกรมแต่ละโปรแกรมมีลักษณะแตกต่างกัน และยังไม่มีวิธีการใดที่ทำงานให้ผลดีในโปรแกรมทุกลักษณะได้ จึงควรนำเอาวิธีการหลายๆ ชนิด เช่น วิธีการวิเคราะห์กราฟ วิธีการวัดค่าคุณสมบัติ และวิธีการวิเคราะห์คอนเซ็ปต์ มาทำงานร่วมกัน ข้อดีของวิธีการนี้คือ สามารถใช้ประโยชน์จากข้อดีของวิธีการทั้งสามชนิดได้ แต่มีข้อเสียคือ ผลลัพธ์ที่ได้จากวิธีการทั้งสามอาจขัดแย้งกัน การทำงานซ้ำซ้อน จึงเสียเวลาและค่าใช้จ่ายมากกว่าวิธีการอื่นๆ

5) วิธีการจัดกลุ่มข้อมูล เป็นวิธีการที่นำเอาอัลกอริทึมจัดกลุ่มข้อมูลฯ เข้ามาช่วยทำงาน โดยใช้เป็นเครื่องมือสำหรับจัดเรียงส่วนคำสั่งเป็นกลุ่มๆ เพื่อให้การเลือกส่วนประกอบมาสร้างวัตถุซอฟต์แวร์ฯ ทำได้สะดวกรวดเร็วมากยิ่งขึ้น วิธีการนี้มีจุดเด่นอยู่ที่ความยืดหยุ่นในการทำงาน เพราะสามารถปรับเปลี่ยนสูตรที่เกี่ยวข้องเพื่อให้ผลลัพธ์ออกมาถูกต้องตรงกับความเป็นจริงมากขึ้นได้ อย่างไรก็ตาม วิธีการนี้มีข้อเสียอยู่ที่ มีความละเอียดอ่อนมาก หากออกแบบวิธีการไม่ดีพอก็อาจทำให้เกิดความผิดพลาดในการทำงานได้ อย่างเช่น งาน Kuipers กับ van Deursen [8] ซึ่งได้นำวิธีการนี้มาช่วยจัดเรียงเร็คคอร์ดในโปรแกรมภาษาโคบอล โดยใช้ความสัมพันธ์ระหว่างเร็คคอร์ดกับโปรแกรมมเป็นคุณสมบัติข้อมูล ผลปรากฏว่าเร็คคอร์ดเกาะกลุ่มกันมาก เพราะคุณสมบัติข้อมูลที่กำหนดให้มีจำนวนน้อยเกินไป จนวิธีการไม่สามารถนำไปใช้แยกแยะข้อมูลได้ ตรงกันข้ามกับงาน Phattarasukol กับ Muenchaisri [10] ซึ่งนำวิธีการนี้มาช่วยจัดเรียงฟังก์ชันในโปรแกรมภาษาซี โดยใช้ความสัมพันธ์ระหว่างฟังก์ชันกับส่วนข้อมูลหลายๆ แบบเป็นคุณสมบัติ ผลปรากฏว่าฟังก์ชันถูกแบ่งเป็นคลัสเตอร์ต่างๆ ได้อย่างถูกต้อง