บทที่ 2
หลักการและทฤษฎี

ในชีวิตประจำวันในปัจจุบัน ไม่ว่าจะดำเนินงานใด ๆ มนุษย์จะต้องเกี่ยวข้องกับข้อมูลอย่าง ใดอย่างหนึ่งเสมอ เช่น การเป็นนักศึกษาที่ต้องมีการลงทะเบียน เพื่อเก็บรายละเอียดเกี่ยวกับวิชาที่เรียน เป็นต้น เมื่อเทคโนโลยีของโลกของได้พัฒนาขึ้น จนกระทั้งปัจจุบันที่มีการใช้คอมพิวเตอร์อย่างกว้างขวาง ข้อมูลต่าง ๆ ซึ่งในอดีตจัดเก็บอยู่บนกระดาษ ซึ่งนับวันจะมีจำนวนเพิ่มมากขึ้น โดยปัจจุบัน ได้ถูกนำมาจัดเก็บไว้ในคอมพิวเตอร์แทน ซึ่งก็คือ ระบบฐานข้อมูล

2.1 ระบบฐานข้อมูล

ฐานข้อมูล (Database) คือ การนำเอาข้อมูลต่าง ๆ ที่มีความสัมพันธ์กันมาเก็บรวบรวมกันไว้ภายในฐานข้อมูลเดียว และสามารถใช้ข้อมูลร่วมกันได้

ระบบฐานข้อมูล (Database system) คือ การจัดข้อมูลอย่างเป็นระบบซึ่งผู้ใช้สามารถเรียกใช้ข้อมูลดังกล่าวได้ในลักษณะต่างๆเช่น การเพิ่มข้อมูล (Add Data) การเรียกใช้ข้อมูล (Retrieve Data) การแก้ไขและลบข้อมูล (Update & Delete) ตลอดจนการเคลื่อนย้ายข้อมูล (Move Data) ไปตามกำหนด หรือระบบฐานข้อมูล อาจหมายถึง การจัดทำฐานข้อมูลขึ้นเพื่อสนับสนุนการดำเนินงานหรือกิจกรรมอย่างใดอย่างหนึ่ง

2.2 โครงสร้างของระบบฐานข้อมูล (Structure of Database)

ระบบฐานข้อมูลสามารถแบ่งออก ตามลักษณะโครงสร้าง ซึ่งประกอบไปด้วยโครงสร้างหลัก 2 ส่วน ได้แก่ ส่วน Front End และ Back End

2.2.1 Front End

เป็นโปรแกรมประยุกต์ (Application) ที่อาจจะสร้างจากภาษาต่างๆ ส่วนนี้โดยปกติ จะรองรับการทำงานของผู้ใช้ (End User) เพื่อทำหน้าที่ติดต่อกับระบบ

2.2.2 Back End

เป็นส่วนที่ทำหน้าในการจัดการกับระบบฐานข้อมูลทั้งหมด ในแง่ของ การจัดเก็บ และเรียกใช้ข้อมูลจากแหล่งข้อมูลจริง ได้แก่ การปฏิบัติการต่างๆกับข้อมูล, การจัดทำ Backup, การควบคุมความถูกต้องในการใช้ข้อมูลพร้อมกัน รวมไปถึงการควบคุมความปลอดภัยของระบบ เป็นต้น

2.3 การรวบความความต้องการ

ในการวิเคราะห์ระบบนั้นจะต้องมีการเข้าไปค้นหาความต้องการของผู้ใช้ ซึ่งขั้นตอนการค้นหาความต้องการและการจดบันทึกความต้องการนั่นเป็นไปได้ยากต้องเข้าไปพบกับบุคคลที่ไม่รู้จักมาก่อนแล้วต้องทำการสอบถามว่า มีความต้องการอะไรบ้างที่จะให้มีในระบบใหม่ ผู้ใช้ระบบบางคนอาจไม่มีความรู้ในเรื่องคอมพิวเตอร์เลย หรืออาจไม่ทราบว่าความต้องการของตัวเองในการใช้ระบบนั่นคืออะไร ดังนั้นต้องทำการเก็บรวบรวมข้อมูลให้ได้ตามรูปแบบมาตรฐาน

หลักการที่ใช้ในการวิเคราะห์ระบบนั้นต้องค้นหาความต้องการให้มีความชัดเจนและต้องมีแนวทางในการจัดการกับบุคคลที่เกี่ยวข้องซึ่งประกอบด้วย ใคร อะไร เมื่อไร่ ที่ไหน ทำไม และอย่างไร

Who ในสถานการณ์ที่กำลังวิเคราะห์อยู่นี้ มีใครเกี่ยวข้องบ้าง? บทบาทของแต่ละคนนั้นคืออะไร? ใครเป็นบุคคลแท้จริงที่ร้องขอเพื่อพัฒนาระบบใหม่?

What ในสถานการณ์นี้ อะไรคือสิ่งที่ทำให้เกิดปัญหา? ระบบที่ต้องการหรือระบบที่อยากได้คือระบบอะไร? มีฟังก์ชั่นในการทำงานอะไรบ้าง?

When ระบบติดตั้งได้เมื่อไหร่? ผู้สนับสนุนเงินทุนพร้อมที่จะสนับสนุนเมื่อไหร่? ทดสอบระบบใหม่เมื่อไหร่?

Why ทำไมต้องเสาะแสวงหาระบบใหม่? ทำไมผู้ใช้จึงเชื่อว่าระบบใหม่สามารถแก้ไขปัญหาให้กับเขาได้

How ระบบใหม่จะทำงานได้อย่างไร? มีข้อจำกัดอย่างไร

2.3.1 **ชนิดความต้องการ** (Types of Requirements)

ปกติแล้ว ความต้องการสามารถแบ่งออกเป็น 2 ชนิดด้วยกัน คือ ความต้องการที่เป็น**ฟังก์ชั่นการทำงานและความต้องการที่ไม่เป็นฟังก์ชันการทำงาน โดยทั้งสองจะมีความแตกต่างกันดังรายละเอียดต่อไปนี้**

ความต้องการที่เป็นฟังก์ชันการทำงาน (Functional Requirements)

คือกิจกรรมที่ระบบต้องปฏิบัติงาน กล่าวคือ เป็นขั้นตอนการทำงานที่ประกอบไปด้วยกิจกรรมต่างๆ ที่ข้องเกี่ยวกับผู้ปฏิบัติงานโดยแต่ละกิจกรรมจะก่อให้เกิดผลการดำเนินงานออกมาและโดยปกติ ความต้องการที่เป็นฟังก์ชันการทำงานมันเขียนอยู่ในรูปแบบของกริยา

ความต้องการที่เป็นฟังก์ชันการทำงานนั้น ตั้งอยู่บนพื้นฐานของขั้นตอนการทำงานและกฎเกณฑ์ขององค์กรที่ใช้สำหรับในการดำเนินธุรกิจเป็นสำคัญ ด้วยการอธิบายว่าระบบจะต้องทำอะไร ดังนั้นความต้องการที่เป็นฟังก์ชันการทำงานจึงเกี่ยวข้องกับ

1. มีอะไรบ้างที่ต้องอินพุตเข้าไปในระบบ
2. มีเอาต์พุตอะไรบ้าง ที่ระบบต้องดำเนินการ
3. มีข้อมูลอะไรบ้าง ที่ระบบต้องจัดเก็บ เพื่อให้ระบบอื่นๆ ที่เกี่ยวข้อง นำข้อมูลไปใช้งานได้
4. การคำนวณอะไร ที่ระบบต้องดำเนินการ
5. สิ่งที่กล่าวข้างต้น จะต้องประสานการทำงานหรือซิงโครไนซ์กันอย่างมีระบบ

ความต้องการที่ไม่ได้เป็นฟังก์ชั่นการทำงาน (Non-Functional Requirements)

เป็นความต้องการที่เกี่ยวข้องกับการกำหนดคุณภาพในการทำงานของซอฟต์แวร์โดยเป็นการปฏิบัติการเพื่อให้บรรลุจุดประสงค์ในทุกๆ ด้านที่เกี่ยวข้องกับสภาพแวดล้อม ฮาร์ดแวร์ และซอฟต์แวร์ขององค์กร กล่าวคือเป็นคุณสมบัติที่ระบบซอฟต์แวร์ควรมี

ความต้องการของผู้ใช้ (User Requirements)

ความต้องการของผู้ใช้ จะบ่งบอกถึงทั้งความต้องการที่เป็นฟังก์ชันการทำงานและความต้องการที่ไม่เป็นฟังก์ชันการทำงาน และโดยส่วนใหญ่ ผู้ใช้ระบบมักเป็นบุคคลที่ไม่มีความรู้ทางด้านเทคนิค แต่ผู้ใช้ระบบสามารถระบุรายละเอียดเกี่ยวกับความต้องการที่ข้องเกี่ยวกับงานที่ดำเนินการอยู่ โดยความต้องการของผู้ใช้ระบบจะเป็นข้อความอธิบาย ที่อาจบรรยายลงในแบบฟอร์มที่กำหนด

อย่างไรก็ตาม ความต้องการของผู้ใช้ที่เขียนอยู่ในรูปของภาษาธรรมชาติ หรือภาษาทั่วไปที่ถ่ายทอดออกมาเป็นตัวหนังสือนั้น ก็อาจส่งผลกระทบในด้าน

1. ยากต่อการทำความเข้าใจ (Lack of clarity)

เป็นที่เข้าใจว่า คำพูดที่แต่ละคนถ่ายทอดออกมาเป็นตัวหนังสือนั้นย่อมแตกต่างกันดังนั้นบางครั้งจึงเป็นการยากที่จะใช้ภาษาพูดที่ถ่ายทอดลงไปเป็นตัวหนังสือที่ทำให้ผู้อื่นอ่านแล้วเข้าใจตรงกัน ซึ่งอาจส่งผลให้ผู้อ่านๆ แล้วไม่เข้าใจ กำกวม ถ้อยคำต่างๆ อ่านแล้วเข้าใจยากเป็นต้น

2. มีความสับสน (Requirements confusion)

เป็นไปได้ที่ผู้ใช้อาจสับสนระหว่างความต้องการที่เป็นฟังก์ชันการทำงานและความต้องการที่ไม่ได้เป็นฟังก์ชันการทำงาน ซึ่งอาจทำให้จุดประสงค์ของระบบและการออกแบบขาดความชัดเจนได้

3. **ความต้องการผสมรวมกัน** (Requirements amalgamation)

ความต้องการต่างๆ ที่หลากหลายอาจผสมอยู่รวมกัน ดังนั้น ควรทำการรวบรวมความต้องการที่เป็นสิ่งเดียวกันหรือข้องเกี่ยวกันให้เหลือเพียงหนึ่งความต้องการ3.2 การวิเคราะห์ระบบ

2.4 การวิเคราะห์ความต้องการ (Requirements Analysis)

หลังจากที่รวบรวมความต้องการจากผู้ใช้ได้แล้ว ข้อมูลทั้งหมดที่ได้นั้นไม่สามารถนำมาใช้งานได้ทันทีหรือนำมาใช้ได้ทั้งหมด จะต้องมีการนำความต้องการที่ได้มารวบรวมผ่านกระบวนการวิเคราะห์ความต้องการ(Requirements Analysis) เพื่อให้ได้มาซึ่งข้อกำหนดความต้องการที่สมบูรณ์ที่บันทึกอยู่ในรูปแบบของเอกสาร เพื่อใช้ประโยชน์ต่อไปในขั้นตอนของการพัฒนาซอฟต์แวร์

2.4.1 ข้อกำหนดความต้องการของระบบใหม่

โดยการวิเคราะห์ความต้องการ เพื่อที่จะได้มาซึ่งข้อกำหนดความต้องการของระบบใหม่จะประกอบไปด้วย 3 ขั้นตอนดังต่อไปนี้

วิเคราะห์ข้อเท็จจริงในข้อมูล (Analysis of Factual Data)

ข้อมูลความต้องการที่ได้รวบรวมมาจากวิธีการสืบเสาะข้อเท็จจริง จะต้องนำมาพิจารณาว่า ระบบจะต้องดำเนินงานอย่างไร เพื่อให้ตรงกับวัตถุประสงค์ตามที่ต้องการ

กำหนดสาระสำคัญของความต้องการ (Identification of Essential Requirements)

คือคุณสมบัติ หรือสาระสำคัญที่ระบบใหม่พึงมี โดยรายละเอียดการปฏิบัติงานจะต้องได้รับการกำหนดขึ้นมา

คัดเลือกความต้องการที่ตรงกับวัตถุประสงค์ (Select of Requirements Fulfillment)

การนำถ้อยแถลงที่ระบุอยู่ในความต้องการที่ผ่านการคัดเลือก มาใช้ให้เกิดผลสำเร็จ โดยความต้องการที่ผ่านการคัดเลือกนี้ จะใช้เป็นพื้นฐานสำหรับการออกแบบระบบต่อไป

2.4.2 หลักในการค้นหาความต้องการที่ดี

1. **ค้นหาความต้องการกับบุคคลที่เกี่ยวข้องโดยตรง และให้ตรงวัตถุประสงค์มากที่สุด**
2. **ควรระบุความต้องการต่างๆ ลงในรูปของเอกสารและมีการทำข้อตกลงร่วมกันทั้งสองฝ่ายซึ่งในบางครั้งอาจจะมีการเซ็นกำกับร่วมกันก็ได้ แต่ในกรณีนี้อาจสร้างความกดดัน**

ให้กับผู้ใช้ได้ แต่ก็ถือเป็นผลดีต่อผู้พัฒนาระบบ ในกรณีที่อาจมีการปรับแก้ในภายภาคหน้า

3. Requirement ที่ดีต้องตกลงร่วมกันทั้งสองฝ่าย อย่าคิด วิเคราะห์ หรือออกแบบด้วยตนเองทั้งหมดซึ่งเป็นการเข้าข้างตัวเอง และมีโอกาสก่อให้เกิดผลเสียตามมา

2.5 แบบจำลองกระบวนการ

2.5.1 ชนิดของแบบจำลอง (Types of Models)

ในการพัฒนาระบบสารสนเทศ นักวิเคราะห์ระบบสามารถนำแบบจำลองชนิดต่างๆมาประยุกต์ใช้กับงานพัฒนาระบบ โดยแบบจำลองแต่ละชนิดสามารถนำมาใช้งานได้อย่างเหมาะสมเพื่อแก้ไขปัญหาตามส่วนงานนั้นๆ เช่น งานบางชนิดสามารถใช้แบบจำลองที่เป็นแค่เพียงถ้อยคำอธิบายก็สามารถนำไปใช้งานได้แล้วในขณะที่งานบางชนิด เพียงแค่ถ้อยคำอาจยังไม่สามารถนำไปใช้ประโยชน์ได้อย่างพอเพียง ดังนั้น จึงต้อง พัฒนาแบบจำลองที่เป็นแผนภาพหรือไดอะแกรมซึ่งมีความเหมาะสมกว่า เพราะสามารถเห็นภาพรวมของระบบได้ทั้งหมดบนกระดาษแผ่นเดียว

ชนิดของแบบจำลอง สามารถแบ่งออกได้เป็น 3 ชนิดด้วยกัน คือ

1. แบบจำลองทางคณิตศาสตร์ (Mathematical Models)
2. แบบจำลองที่เป็นถ้อยคำอธิบาย (Descriptive Models)
3. แบบจำลองแผนภาพ (Graphical Models)

แบบจำลองทางคณิตศาสตร์ (Mathematical Models)

แบบจำลองทางคณิตศาสตร์ คือ กลุ่มของสูตรคำนวณที่ใช้อธิบายกฎเกณฑ์ทางเทคนิคของระบบ โดยมักใช้กับงานทางด้านวิทยาศาสตร์และวิศวกรรม หรืองานด้านการคำนวณสัญลักษณ์ทางคณิตศาสตร์ส่วนใหญ่มักนำเสนอในรูปแบบของฟังก์ชัน แต่อย่างไรก็ตามสัญลักษณ์และสูตรทางคณิตศาสตร์ต่างๆ บางครั้งก็สามารถนำมาใช้ได้เป็นอย่างดีสำหรับระบบงานทางธุรกิจ ตัวอย่างเช่น ระบบเงินเดือนที่จำเป็นต้องมีสูตรคณิตศาสตร์เพื่อใช้เป็นแบบจำลองในการคำนวณเงินสุทธิ ภาษี และค่าประกันต่างๆต่างๆหรือสูตรการคิดภาษีเงินได้บุคคลธรรมดา ซึ่งสูตรคำนวณภาษีดังกล่าวทำให้ผู้มีเงินได้สามารถใช้คำนวณภาษีเงินได้ดังกล่าวทำให้ผู้มีเงินได้ด้วยตนเอง เป็นต้น

แบบจำลองที่เป็นถ้อยคำอธิบาย (Descriptive Models)

ความต้องการทั้งหมดคงไม่สามารถกำหนดขึ้นได้แบบจำลองที่เป็นการกล่าวนำถึงเรื่องราว เช่น ถ้อยคำอธิบาย รายละเอียด รายงาน หรือรายการต่างๆสามารถนำมาใช้ประโยชน์ได้ดีทีเดียว นอกจากนี้ แบบจำลองที่เป็นถ้อยคำอธิบายยังสามารถเขียนให้อยู่ในรูปแบบของกระบวนการ หรือ

ขั้นตอนวิธีที่อยู่ในรูปแบบของรหัสจำลองหรือประโยคสร้างภาษาอังกฤษที่โปรแกรมเมอร์มักนำแบบจำลองชนิดนี้ไปใช้เพื่อการออกแบบโปรแกรม

แบบจำลองแผนภาพ (Graphical Models)

แบบจำลองแผนภาพหรือไดอะแกรม จัดเป็นแบบจำลองที่มีประโยชน์มากที่สุดที่พัฒนาขึ้นโดยนักวิเคราะห์ระบบ โดยไดอะแกรมที่เขียนขึ้นนั้นจะทำให้สามารถเข้าใจถึงความสัมพันธ์ของสิ่งต่างที่มีอยู่ในระบบ ตัวอย่างแบบจำลองดังกล่าวเช่น แผนภาพกระแสข้อมูล แผนภาพอีอาร์ เป็นต้น ซึ่งแผนภาพเพียงแผนภาพหนึ่ง สามารถบรรยายภาพรวมของระบบได้เป็นอย่างดี ซึ่งนับเป็นข้อดีของแบบจำลองชนิดนี้

แบบจำลองกระบวนการ (Process Model)

แบบจำลองกระบวนการ จะอธิบายถึงกระบวนการทางธุรกิจ ด้วยการนำเสนอให้เห็นภาพรวมถึงการปฏิบัติอย่างไรในระบบธุรกิจในลักษณะของแผนภาพหรือไดอะแกรม โดยความสำคัญอยู่ที่ว่า นักวิเคราะห์ระบบจะต้องดำเนินการวิเคราะห์เพื่อให้ได้มาซึ่งกระบวนการหลักๆของธุรกิจให้ได้ และอาจนำบางกระบวนการมาปรับปรุงให้ดียิ่งขึ้น หรือที่เรียกว่า การออกแบบกระบวนการธุรกิจใหม่ (Business Process Redesign : BPR) ซึ่งเป็นการศึกษา และวิเคราะห์ถึงพื้นฐานกระบวนการทางธุรกิจ เพื่อสร้างมูลค่าเพิ่มให้กับธุรกิจ รวมถึงการลดต้นทุน โดยกระบวนการทางธุรกิจ (Business Process) จะเป็นกลุ่มของกระบวนการทำงานที่ตอบสนองต่อเหตุการณ์ทางธุรกิจ กระบวนการทางธุรกิจเป็นงานที่ถูกปฏิบัติการโดยระบบ และการที่นักวิเคราะห์ระบบได้ทราบรายละเอียดเกี่ยวกับกระบวนการทางธุรกิจ จะทำให้เข้าใจถึงการไหลของข้อมูลในระบบว่าแต่ละงานนั้น มีทิศทางการทำงานอย่างไร เกี่ยวข้องกับผู้ใด เกี่ยวข้องกับผู้ใด และสื่อสารกันอย่างไร

ในการสร้างแบบจำลองกระบวนการ สามารถสร้างด้วยเทคนิคที่แตกต่างกันตามแต่ละเทคโนโลยี เช่น แบบจำลองเชิงโครงสร้าง กันแบบจำลองเชิงวัตถุ (Data Flow Diagram) ซึ่งแผนภาพดังกล่าวจะแสดงถึงกระบวนการหรือกิจกรรมที่ปฏิบัติการ รวมถึงการแสดงความเคลื่อนไหวของข้อมูลในระบบ โดยแผนภาพกระแสข้อมูลสามารถนำมาประยุกต์ใช้กับระบบงานเดิมหรือระบบงานใหม่ก็ได้

2.5.2 แผนภาพกระแสข้อมูล (Data Flow Diagram: DFD)

หากเปรียบเทียบกับพัฒนาซอฟแวร์กับการสร้างบ้านแล้ว จะเห็นได้ว่าการสร้างบ้านขึ้นมาสักหลังหนึ่ง เจ้าของบ้านกับสถาปนิกจะต้องมีการตกลงถึงความเข้าใจระหว่างกันในเบื้องต้นว่า ตนจะสร้างบ้านบนพื้นที่นี้ในรูปแบบอย่างไร แบบบ้านทรงไหนมีกี่ชั้น กี่ห้อง มีการจัดสดส่วนห้องต่าง

ไว้อย่างไร ซึ่งสิ่งเหล่านี้ก็คือความต้องการหรือ Requirements นั่นเอง จากนั้นสถาปนิกก็จะรวบรวมความต้องการของลูกค้าในเบื้องต้นเพื่อนำไปเขียนเป็นแบบแปลนหรือแบบพิมพ์เขียว (Blueprint) ขึ้นมา ซึ่งแบบพิมพ์เขียวนี่เองที่จะนำมาใช้เป็นข้อตกลงระหว่างกันว่า บ้านที่ต้องการให้สร้างนั้นเป็นไปตามแบบที่ต้องการหรือไม่ โดยสถาปนิกอาจจำเป็นต้องมีการปรับแบบแปลนบางส่วน ตามที่ลูกค้าต้องการ จนกระทั่งได้แบบแปลนที่สมบูรณ์ จากนั้นแบบแปลนหรือแบบพิมพ์เขียวดังกล่าว ก็จะนำไปให้วิศวกรควบคุมงาน เพื่อให้งานก่อสร้างนั้นเป็นไปตามแผนงานที่วางไว้

แผนภาพกระแสข้อมูล ก็เปรียบเสมือนกับแบบพิมพ์เขียวนั่นเอง สถาปนิกก็เปรียบเสมือนกับนักวิเคราะห์ระบบ ที่จะต้องนำความต้องการของลูกค้าไปวิเคราะห์ เพื่อสร้างแบบแปลนบ้านตามความต้องการของลูกค้าโดยวิศวกรก็เปรียบเสมือนกับนักวิศวกรรมซอฟแวร์ หรือผู้จัดการโครงการที่ควบคุมคนงานก่อสร้าง ให้สร้างบ้านตามแบบบนงบประมาณที่วางไว้ และส่งมอบบ้านตามเวลาที่กำหนดไว้ในสัญญา ส่วนคนงานก่อสร้างก็เปรียบเสมือนกับโปรแกรมเมอร์ที่จะต้องสร้างบ้านตามแบบ ที่สถาปนิกหรือนักวิเคราะห์ระบบได้ออกแบบไว้

แผนภาพกระแสข้อมูล เป็นแบบจำลองกระบวนการที่นำมาใช้กับการวิเคราะห์และออกแบบระบบเชิงโครงสร้าง ที่มีการนำมาใช้ตั้งแต่ยุคที่มีการเริ่มใช้ภาษาระดับสูงอย่างภาษาโคบอล โดยแผนภาพกระแสข้อมูลจะแสดงความสัมพันธ์ระหว่างโปรเซส (Process) กับข้อมูล (Data) ที่เกี่ยวข้อง โดยข้อมูลในแผนภาพจะทำให้ทราบถึง

1. ข้อมูลมาจากไหน
2. ข้อมูลไปไหน
3. ข้อมูลเก็บไว้ที่ไหน
4. เกิดเหตุการณ์ใดกับข้อมูลในระหว่างทาง

แผนภาพของกระแสข้อมูลจะแสดงภาพรวมของระบบ และรายละเอียดเกี่ยวกับโปรเซสกับข้อมูล แต่ในบางครั้ง หากต้องการกำหนดรายละเอียดที่นอกเหนือไปจากนี้ นักวิเคราะห์ระบบอาจจำเป็นต้องใช้เครื่องมืออื่นเข้าช่วย เช่น ข้อความสั้นๆ ที่อ่านแล้วง่ายต่อการทำความเข้าใจหรืออัลกอริทึม,ตารางการตัดสินใจ (Decision Table), แบบจำลองข้อมูล (Data Model), คำอธิบายการประมวลผล (Process Description) เป็นต้น ทั้งนี้ขึ้นอยู่กับความต้องการในรายละเอียดเป็นสำคัญ

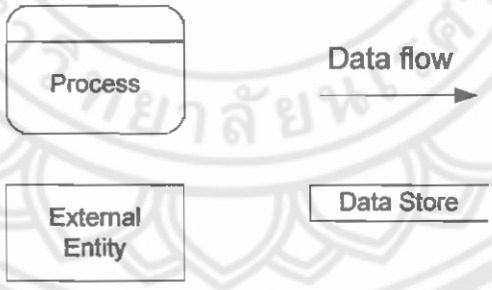
2.5.3 วัตถุประสงค์ของแผนภาพกระแสข้อมูล

วัตถุประสงค์ของการสร้างแผนภาพกระแสข้อมูล เพื่อ

1. **เป็นแผนภาพที่สรุปรวบรวมข้อมูลทั้งหมดที่ได้จากการวิเคราะห์ในรูปแบบของการพัฒนาเชิงโครงสร้าง**
2. **เป็นข้อตกลงร่วมกันระหว่างนักวิเคราะห์ระบบกับผู้ใช้งาน**
3. เป็นแผนภาพที่นำไปใช้ประโยชน์ต่อไปในขั้นต่อตอนของการออกแบบระบบ
4. เป็นแผนภาพที่ใช้ในการอ้างอิง หรือเพื่อใช้สำหรับการปรับปรุง/พัฒนาต่อไปในอนาคต
5. ทราบที่มาและที่ไปของข้อมูลที่ไหลไปยังกระบวนการต่างๆ (Data and Processes)

2.5.4 **สัญลักษณ์ที่ใช้ในแผนภาพกระแสข้อมูล**

แผนภาพกระแสข้อมูล เป็นแผนภาพที่แสดงภาพรวมของความต้องการหลักๆ ของระบบสารสนเทศ ในรูปแบบของไดอะแกรม ซึ่งประกอบด้วย อินพุต เอาต์พุต กระบวนการ และข้อมูล โดยทุกๆคนในทีมงานพัฒนาระบบสามารถเห็นรูปร่างหน้าตาของระบบได้จากแผนภาพนี้ และใช้เป็นแนวทางในการออกแบบระบบและนี่เป็นเหตุผลหนึ่งที่ทำให้แผนภาพกระแสข้อมูลเป็นแบบจำลองที่นิยมใช้งานจนถึงปัจจุบันและจัดเป็นแผนภาพที่ดูแล้วง่ายต่อการทำความเข้าใจ เนื่องจากเป็นแบบจำลองในลักษณะแผนภาพที่มีเพียง 4 สัญลักษณ์หลักๆเท่านั้น ซึ่งแสดงได้ดัง



รูปที่ 2.1 แสดงสัญลักษณ์ที่ใช้สำหรับการเขียนแผนภาพกระแสข้อมูล

2.5.5 โปรเซส (Processes)

เป็นสัญลักษณ์แทนกิจกรรมที่เกิดขึ้นในระบบสารสรเทศ หรือกระบวนการที่ต้องทำในระบบ แผนภาพกระแสข้อมูลจะต้องมีสัญลักษณ์โปรเซสอย่างน้อยหนึ่งโปรเซสเสมอ โดยดาต้า

โฟลว์ที่ได้อินพุตผ่านเข้าไปยังโปรเซส และเมื่อออกจากโปรเซสก็คือเอาต์พุต ดังนั้น ดาต้าโฟลว์ที่เอาต์พุตออกมาย่อมเปลี่ยนแปลงเสมอ เช่น ดาต้าโฟลว์ที่อินพุตเข้าไปยังโปรเซสชื่อ "คำนวณเงินเดือนสุทธิ" ซึ่งประกอบด้วยเงินเดือนภาษี และค่าประกันสังคม ดาต้าโฟลว์ดาต้าโฟลว์ที่เอาต์พุตออกมาจากโปรเซสดังกล่าวคือเงินเดือนสุทธิ เป็นต้น ซึ่งแสดงดังตัวอย่าง รูปที่ 5.7 หรืออีกตัวอย่างหนึ่ง เช่น ดาต้าโฟลว์ที่อินพุตเข้าไปยังโปรเซสชื่อ "จัดการข้อมูล" คือข้อมูลลูกค้า ดังนั้น ดาต้าโฟลว์ที่เอาต์พุตออกมาจากโปรเซสก็คือข้อมูลลูกค้าที่ได้รับการปรับปรุงหรือเปลี่ยนแปลง เป็นต้น

รูปที่ 2.2 แสดงโปรเซส

ลักษณะโปรเซสจำเป็นต้องมีหมายเลขกำกับอยู่เสมอ ซึ่งเรียกว่าหมายเลขโปรเซส ที่โดยมักกำหนดเป็นหมายเลข 1,2,3 ตามลำดับ โดยการลำดับหมายเลขของโปรเซส ไม่ได้หมายความว่าต้องดำเนินกิจกรรมตามลำดับของโปรเซสแต่อย่างใด และที่สำคัญหมายเลขโปรเซสจะซ้ำไม่ได้

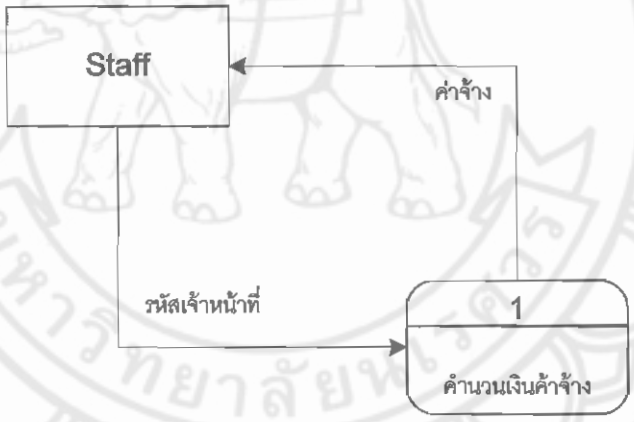
สำหรับชื่อที่ใช้กำกับโปรเซส ปกติมักจะใช้เป็นคำกริยาซึ่งเป็นการกระทำ เช่น ลงทะเบียน, ชำระเงิน, เช่ารถ, พิมพ์รายงาน เป็นต้น และจำนวนโปรเซสควรมีได้ตั้งแต่ 2 ถึง 7 โปรเซสด้วยกัน ซึ่งหากมีมากไปกว่านั้นจะทำให้แผนภาพอ่านยาก แต่อย่างไรก็ตาม ถึงแม้จะไม่มีการกำหนดจำนวนโปรเซสในลักษณะเป็นกฎเกณฑ์ข้อบังคับแต่อย่างใด แต่ก็มีผู้เชี่ยวชาญได้ระบุไว้ว่า จำนวนโปรเซสสูงสุดที่เหมาะสมนั้น ควรอยู่ในช่วงระหว่าง 7 บวกลบด้วย 2 นั่นหมายถึง 5 ถึง 9 โปรเซส

โปรเซสในแผนภาพกระแสข้อมูลจะไม่มีการแสดงรายละเอียดเกี่ยวข้องกับวิธีการทำงาน แต่โปรเซสจะทำให้เรารู้ว่ามีอินพุตอะไรเข้าไปบ้าง และเมื่อผ่านการประมวลผลแล้วได้เอาต์พุตอะไรออกมา ดังนั้น โปรเซสจึงเปรียบเสมือนกล่องดำ (Black Box) ที่นำเสนอเพียงว่าทำหน้าที่อะไร มีอะไรที่อินพุตเข้าไปและมีเอาต์พุตใดที่ออกมา แต่จะไม่แสดงรายลเอียดเกี่ยวกับวิธีการทำงานอย่างไร ซึ่งรายละเอียดวิธีการทำงานของแต่ละ โปรเซสจะปรากฏอยู่ในคำอธิบายการประมวลผล (Process Description)

2.5.6 ดาต้าโฟลว์ (Data Flow)

ดาต้าโฟลว์หรือกระแสข้อมูล จะใช้สัญลักษณ์แทนด้วยเส้นลูกศรที่ไปพร้อมกับข้อมูล ทำให้ทราบถึงข้อมูลที่เคลื่อนไหวไปมาระหว่างโปรเซส ดาต้าสโตร์ และเอกซ์เทอร์นัลเอ็นติตี้ พึงจำไว้ว่าทุกๆโปรเซสในแผนภาพกระแสข้อมู,เมื่อมีดาต้าโฟลว์อินพุตเข้าไป ก็จะต้องมีกาต้าโฟลว์ที่เอาต์พุตออกมาเสมอ โดยอาจมีอินพุตหรือเอาต์พุตของดาต้าโฟลว์มากกว่าหนึ่งจุดก็เป็นได้ แต่โปรเซสจะมี เพียงดาต้าโฟลว์ที่อินพุตเข้าไป หรือมีเพียงแต่ดาต้าโฟลว์ที่เอาต์พุตออกมาจากโปรเซส ก็คงเป็นสิ่งที่ไม่น่าเกิดขึ้นได้ เช่น โปรเซสมีอินพุต แต่ไม่เอาต์พุตหรือโปรเซสมีเพียงแต่เอาต์พุตโดนปราศจากอินพุต ซึ่งถือว่าเป็นสิ่งที่ผิดธรรมชาติ

ในการเขียนแผนภาพกระแสข้อมูล อาจมีดาต้าโฟลว์จำนวนมากในแผนภาพ ซึ่งก็อาจทำให้ดาต้าโฟลว์ของแต่ละเส้นที่ลากโยงไปมาอาจมีการทับซ้อนกันได้ แต่อย่างไรก็ตาม หลักการเขียนแผนภาพกระแสข้อมูลที่ดีไม่ควรมีเส้นดาต้าโฟลว์ทับซ้อนกัน เพราะทำให้แลดูยุ่งเหยิง ไม่มีระเบียบ แต่หากจำเป็นต้องทับซ้อนกัน ก็ขอให้เกิดน้อยที่สุดเท่าที่จะทำได้ โดยเส้นโฟลว์ที่ทับซ้อนกันนั้นควรเป็นเส้นโฟลว์แบบกระโดด (Jump) เพื่อให้ง่ายต่อการดู



รูปที่ 2.3 แสดง ดาต้าโฟลว์ที่เคลื่อนไหวระบบ

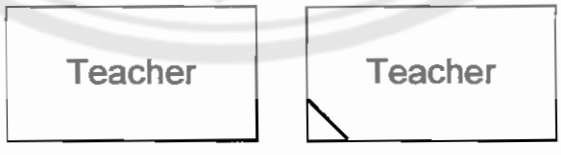
5.5.7 **เอ็กซ์เทอร์นัลเอ็นติตี้** (External Entities)

ในแผนภาพกระแสข้อมูล จะมีดาต้าโฟลว์ที่อินพุตเข้ามาในระบบและดาต้าโฟลว์ที่เอาต์พุตออกจากระบบ **ซึ่งส่วนที่ทำหน้าที่ส่งและรับข้อมูลนี้ เรียกว่าเอ็กซ์เทอร์นัลเอ็นติตี้ สัญลักษณ์เอ็กเทอร์นัลเอ็นติตี้จะมีลักษณะเป็นรูปสี่เหลี่ยมผืนผ้า ซึ่งมีเพียงหน้าที่ในการส่งหรือรับข้อมูลจากโปรเซสเท่านั้น โดนดาต้าโฟลว์ที่อินพุตเข้ามายังระบบถือเป็นแหล่งกำเนิดของข้อมูล** (data source) **ในขณะที่ดาต้าโฟลว์ที่เอาต์พุตออกจากโปรเซสก็จะถูกส่งไปยังปลายทาง** (destination)

ดังนั้น สัญลักษณ์นี้จึงสามารถเรียกได้หลายชื่อด้วยกันไม่ว่าจะเป็น Source, Destination, External **หรือ** Boundary **เป็นต้น ซึ่งล้วนแต่มีความหมายเดียวกันทั้งสิ้น และด้วยการกระทำดังกล่าว ดาต้าโฟลว์ ที่เข้าออกระหว่างเอ็กซ์เทอร์นัลเอ็นติตี้จึงต้องผ่านโปรเซสเสมอ ไม่สามารถเชื่อมต่อเข้าโดยตรงกับดาต้าสโตว์ ได้ เนื่องจากไม่สามารถสื่อความหมายได้ ๆ ได้เลย**

เอ็กเทอร์นัลเอ็นติตี้ สามารถเป็นไปได้ทั้งบุคคล หน่วยงาน หรือระบบงาน สำหรับการพิจารณาบุคคลที่จะมาเป็นเอ็กซ์เทอร์เอ็นติตี้นั้นใช่ว่าจะนำบุคคลในระบบมากำหนดได้ทั้งหมด ตัวอย่างเช่น ระบบร้านเช่าวิดีโอจะประกอบไปด้วยเอกซ์เทอร์นัลเอ็นติตี้ เช่น ลูกค้า (Customer), ผู้จัดการ (Manager), และร้านค้าตัวแทนจำหน่าย (Supplier) แต่หลายคนอาจมีการกำหนดให้พนักงาน (Operator) เป็นเอ็กซ์เทอร์นัลเอ็นติตี้ ซึ่งความจริงนั้นไม่ถูกต้อง บุคคลใดๆ ก็ตามที่ปฏิบัติการกับโปรเซสโดยตรง เช่น พนักงานป้อนข้อมูลหรือเสมียน จะถือเป็นส่วนหนึ่งในการบวนการ ดังนั้น จึงไม่ถือว่าเป็นบุคลภายนอกหรือเอ็กซ์เทอร์เอ็นนัลติตี้ ซึ่งผู้ปฏิบัติการในกระบวนการต่างๆอาจมีการอธิบายในรายละเอียดลงในคำอธิบายการประมวลผลข้อมูล (Process Description) แต่จะไม่ปรากฏอยู่ในแผนภาพกระแสข้อมูล ดังนั้น การพิจารณาถึงบุคคลใดเป็นเอ็กซ์เทอร์นัลเอ็นติตี้หรือไม่นั้น จะพิจารณาถึงบุคคลที่ระบบไม่สามารถควบคุมได้เป็นสำคัญ กล่าวคือจะต้องเป็นสิ่งที่อยู่ภายนอกขอบเขตของระบบประมวลผล แต่อย่างไรก็ตาม ก็มีบุคคลภายในระบบที่จัดเป็นเอ็กซ์เทอร์นัลเอ็นติตี้ โดยธรรมชาติ เช่น ผู้จัดการ (Manger) หรือทีมงาน (Staff) เป็นต้น

เอ็กซ์เทอร์นัลเอ็นติตี้มักจะถูกจัดให้อยู่บริเวณด้านนอกหรือรอบๆ ของแผนภาพเพื่อให้แผนภาพแลดูสวยงาม และง่ายต่อการดู นอกจากนี้ เอ็กเทอร์นัลเอ็นติตี้ ในบางครั้งอาจจำเป็นต้องกระทำซ้ำ (Duplicate) เนื่องจากข้อจัดในด้านของการเชื่อมโยงดาต้าโฟลว์ อาจทำให้เส้นโฟลว์ดังกล่าวทับซ้อนกันหรือข้ามกันไปมาได้ซึ่งเป็นสิ่งที่ไม่ควรกระทำ เพราะจะทำให้แผนภาพดูยุ่งเหยิงไม่เป็นระเบียบ และทำให้แผนภาพอ่านยาก โดยเอ็กเทอร์นัลเอ็นติตี้ที่กระทำซ้ำนั้นสามารถทำซ้ำได้ด้วยการใช้เครื่องหมาย \ (Back Slash) ไว้ที่ตรงมุมล่างซ้ายซึ่งแสดงได้ดังตัวอย่างรูปที่ 5.9



รูปที่ 2.4 แสดง เอ็กซ์เทอร์นัลเอ็นติตี้และเอ็กซ์เทอร์นัลเอ็นติตี้ที่เขียนซ้ำ

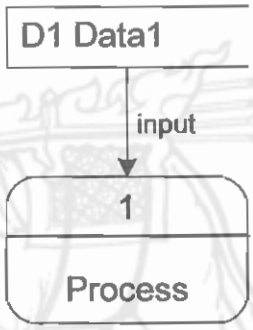
2.5.8 ดาต้าสโตร์ (Data Stores)

เป็นแหล่งที่ใช้เก็บข้อมูล ซึ่งจะไม่สนว่าระบบจะใช้สื่อจัดเก็บข้อมูลในรูปแบบใด ทุกๆ ดาต้าสโตร์จะต้องมีชื่อข้อมูล และมีการกำหนดลาเบล เช่น D1, D2, D3 ตามลำดับ โดยดาต้าสโตร์จะถูกใช้งานโดยโปรเซสและดาต้าสโตร์สามารถกระทำซ้ำได้ ส่วนที่มาของดาต้าสโตร์นั้น จะได้มาจากการสร้างแบบจำลองข้อมูล (Data Model) **ซึ่งจะกล่าวรายละเอียดในบทต่อไป**

สำหรับลูกศรของดาต้าโฟลว์ที่ใช้เชื่อมระหว่างดาต้าสโตร์กับโปรเซส จะมีความหมายดังนี้

ลูกศรจากดาต้าสโตร์ชี้ไปยังโปรเซส (Input)

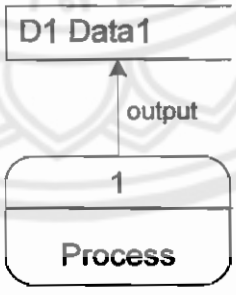
เป็นสัญลักษณ์ของการอินพุต ซึ่งเกี่ยวข้องกับการเรียก (Retrieved) หรือการอ่านข้อมูลจากดาต้าสโตร์ขึ้นมา เช่น อ่านข้อมูลจากแฟ้มนักศึกษาหรือเรียกรายการข้อมูลลงทะเบียนของนักศึกษาขึ้นมาเป็นต้น



รูปที่ 2.5 แสดงโปรเซสที่มีการอินพุตหรือเรียกข้อมูลจากดาต้าสโตร์เข้ามาใช้งาน

ลูกศรจากโปรเซสนชี้ไปยังดาต้าสโตร์ (Output)

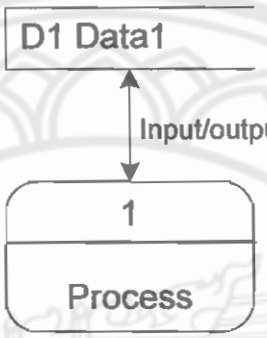
เป็นสัญลักษณ์ขของเอาต์พุต ซึ่เกี่ยวข้องกับการเพิ่มข้อมูลลงในดาต้าสโตร์ และรวมถึงการอัปเดต เช่นการบันทึกข้อมูลนักศึกษาใหม่ หรือการอัปเดตข้อมูลนักศึกษา เป็นต้น



รูปที่ 2.6 แสดงโปรเซสที่มีการเอาต์พุตหรือเพิ่มข้อมูลเข้าไปยังดาต้าสโตร์

ลูกศรบนปลายทั้งสองด้าน (Input/Output)

เป็นสัญลักษณ์ของการเป็นทั้ง อินพุตและเอาต์พุต ซึ่งเกี่ยวข้องกับการอัปเดตข้อมูลลงในดาต้าสโตร์โดยจะมีการดึงข้อมูลจากดาต้าสโตร์ขึ้นมาและมาทำการเปลี่ยนแปลง จากนั้นก็จัดเก็บลงไปใหม่ เช่น การดึงข้อมูลรายการลงทะเบียนของนักศึกษาขึ้นมา และทำการเปลี่ยนแปลงรายวิชาลงทะเบียนจากนั้นก็ดำเนินการจัดเก็บลงไปใหม่เพื่อทำการอัปเดต เป็นต้น



รูปที่ 2.7 แสดงโปรเซสที่มีการอินพุตและเอาต์พุต

2.6 คอนเท็กซ์ไดอะแกรม

2.6.1 แผนภาพกระแสข้อมูลระดับที่ 0 (Level 0 : Context Diagram)

แผนภาพกระแสข้อมูลระดับสูงสุด เรียกว่า คอนเท็กซ์ไดอะแกรม หรือมักเรียกว่า DFD ระดับ 0 โดยแผนภาพดังกล่าวจะมีเพียงหนึ่งโปรเซสที่เป็นชื่อของระบบงาน และมีดาต้าโฟลว์เชื่อมต่อระหว่างโปรเซสกับเอ็กเทอร์นัลเอ็นติตี้ โดยไม่มีดาต้าสโตร์ จุดประสงค์ของคอนเท็กซ์ไดอะแกรมนี้ก็เพื่อแสดงแวดล้อมของระบบเพื่อให้เห็นว่าระบบมีการโต้ตอบกับเอ็กเทอร์นัลเอ็นติตี้ใดบ้าง ส่วนรายละเอียดภายในระบบงาน ว่ามี กระบวนการหรือโปรเซสย่อยใดบ้างนั้น ก็จะแสดงอยู่ในแผนภาพกระแสข้อมูลระดับที่ 1 ต่อไป

2.6.2 แผนภาพกระแสข้อมูลระดับที่ 1 (DFD – Level 1)

แผนภาพกระแสข้อมูลระดับที่ 1 นั้นจะเป็นที่รวมของโปรเซสหลักและข้อมูลหลักๆ ที่เกี่ยวข้อง สำหรับในที่นี้จะเริ่มด้วยการเขียนแผนภาพกระแสข้อมูลระดับที่ 1 ของแต่ละกระบวนการ หรือเรียกว่า ดีเอฟดีแฟร็กเมนต์ (DFD Fragments) เพื่อแสดงเหตุการณ์ของแต่ละกระบวนการ จากนั้นก็ดำเนินการนำดีเอฟดีแฟร็กเมนต์มารวบรวมเข้าด้วยกันเพื่อเป็นไดอะแกรม (DFD – Level1)

2.6.2 **แผนภาพกระแสข้อมูลระดับที่** 2 (DFD – Level 2)

แผนภาพกระแสข้อมูลระดับที่ 2 จะแสดงถึงโปรเซสย่อย (sub process) **ของแผนภาพกระแสข้อมูลระดับที่ 1 ซึ่งโดยปกติแผนภาพกระแสข้อมูลระดับที่ 1 ส่วนใหญ่ยังสามารถแตกโปรเซสออกเป็นส่วนย่อยต่อไปได้อีก เพื่อแสดงถึงกระบวนการทำงานของระบบในรายละเอียดกล่าวคือ แผนภาพกระแสข้อมูลระดับที่ 2 นั้นจะทำการแตกฟังก์ชันการทำงานในโปรเซสของ** แผนภาพกระแสข้อมูลระดับที่ 1 ออกเป็นส่วนๆซึ่งกระบวนการแตกฟังก์ชั่นนี้เรียกว่า Functional Decomposition และหากโปรเซสได้แตกกระจายออกมาเป็นแผนภาพกระแสข้อมูลระดับที่ 2 แล้วไม่สามารถแตกย่อยต่อไปได้อีก (Function Primitive) กระบวนการแตกฟังก์ชันก็จะหยุดที่ระดับ 2 ซึ่งก็ถือว่าเพียงพอต่อความต้องการแล้ว แต่อย่างไรก็ตาม หากแผนภาพกระแสข้อมูลระดับที่ 2 ยังคงสามารถแตกฟังก์ชันเป็นกระบวนการย่อยต่อไปได้อีก กล่าวคือ ยังไม่ใช่เป็น Functional Primitive นั่นหมายถึง จำเป็นต้องแตกกระจายเป็นระดับที่ 3 ต่อไป

2.6.4 **การตรวจสอบความสมดุลของแผนภาพ** (Balancing)

ประการสำคัญอย่างหนึ่งในการเขียนแผนภาพกระแสข้อมูล คือแผนภาพระดับล่างของแต่ละแผนภาพจะต้องมีความสมดุล (Balancing) กับแผนภาพในระดับที่เกี่ยวข้องกันเสมอ การตรวจสอบความสมดุลของแผนภาพ จะทำให้แผนภาพที่สร้างขึ้นมานั้นมีคุณภาพยิ่งขึ้น นอกจากนี้การตรวจสอบความสมดุลของแผนภาพยังเกี่ยวข้องกับดาต้าสโตร์ในแผนภาพกระแสข้อมูลระดับที่ 1 โดยจำนวนดาต้าสโตร์ของแผนภาพกระแสข้อมูลระดับที่ 1 ต้องมีความสมดุลกับแผนภาพอีอาร์ (ER - Diagram) ที่แสดงถึงความสัมพันธ์ของข้อมูลทั้งหมดในระบบ

2.6.5 **แผนภาพกระแสข้อมูลเชิงกายภาพ และเชิงตรรกะ** (Physical and Logical DFDs)

แผนภาพกระแสข้อมูลสามารถเป็นได้ทั้งแบบเชิงกายภาพ (Physical - DFD) หรือเชิงตรรกะ (Logical - DFD) หรืออาจนำทั้งสองมาผสมผสานกัน โดย Physical – DFD ได้ถูกคิดค้นขึ้นโดย Gane , Sarson และ DeMarco ซึ่งสามารถย้อนกลับไปดูขั้นตอนการสร้างแผนภาพกระแสข้อมูลดังรูปที่ 5.3 ซึ่งเป็นขั้นตอนการสร้างแผนภาพกระแสข้อมูลโดยสมบูรณ์แบบ โดยประกอบด้วยการสร้างแผนภาพกระแสข้อมูลทั้งในส่วนลอจิคัลและฟิสิคัลของระบบงานเดิมและระบบงานใหม่ ซึ่งในทางปฏิบัติแล้ว สามารถพบได้ยากมากกับระบบงารที่พัฒนาตามสิ่งที่**เชี่ยวชาญทั้งสามได้แนะนำไว้โดยเคร่งครัด เนื่องจากจำเป็นต้องใช้แรงงาน เวลา และงบประมาณ**มาก

อย่างไรก็ตาม การดำเนินการด้วยการเขียนแผนภาพกระแสข้อมูลจนครบทั้งสี่ขั้นตอนนั้นจะทำให้เกิดความสมบูรณ์ยิ่งขึ้นและมีประโยชน์ต่อนักวิเคราะห์ระบบมากทีเดียว กล่าวคือ จะทำให้

เห็นข้อเปรียบเทียบในรายละเอียดเกี่ยวกับระบบงานเดิมกับระบบงานใหม่ รวมถึงได้เห็นข้อผิดพลาดจากระบบงานเดิมที่ทำให้ผู้ใช้งานไม่พึงพอใจ ซึ่งสิ่งเหล่านี้จะได้รับการแก้ไขและปรับปรุงลงในแผนภาพระบบงานใหม่ที่ได้มีการเพิ่มเติมความต้องการใหม่ๆเข้าไปแล้วนั่นเอง และจากเนื้อหาต่อไปนี้จะกล่าวถึงหลักการสร้าง Physical-DFD จาก Logical-DFDโดยเพื่อให้การนำเสนอกระชับยิ่งขึ้น จึงขอใช้คำว่า "Logical-DFD" แทนคำว่า "แผนภาพกระแสข้อมูลเชิงตรรกะ และ "Physical-DFD" แทนคำว่า "แผนภาพกระแสข้อมูลเชิงกายภาพ

โดยทั่วไปแล้ว แผนภาพกระแสข้อมูลมักถูกนำเสนอให้อยู่ในรูปของแบบจำลองเชิงตรรกะ (Logical-DFD) ซึ่งเปรียบเสมือนแบบพิมพ์เขียวที่ทำให้ทราบว่าระบบต้องทำอะไร (What) บ้างเป็นสำคัญ โดย Logical-DFD ที่สร้างขึ้นจากระยะการวิเคราะห์ เมื่อนำเข้ามาสู่ระยะการออกแบบนักวิเคราะห์อาจมีความจำเป็นต้องนำ Logical-DFD เหล่านี้มาแปลงเป็น Physical-DFD เพื่อสะดวกต่อการพัฒนาโปรแกรม เนื่องจาก Physical-DFD มุ่งเน้นรายละเอียดว่าระบบต้องทำอย่างไร (How) เป็นสำคัญ

Physical-DFD จะมีส่วนประสอบต่างๆ เช่นเชียวกับ Logical-DFD เช่น ดาต้าโฟลว์ (Data Store), กระแสข้อมูล (Data Flows) และรวมถึงกฎเกณฑ์ต่างๆ เช่น ความสมดุลของแผนภาพ (Balancing) และการจำแนกออกเป็นส่วนๆ (Decomposition) ส่วนรายละเอียดที่สร้างความแตกต่างระหว่าง Logical-DFD และ Physical-DFD ก็คือ Physical-DFD จะมีการเพิ่มเติมรายละเอียดเกี่ยวกับคำอธิบายว่าสร้างอย่างไร ซึ่งก่อประโยชน์แก่ผู้พัฒนาได้ทราบถึงรายละเอียดเกี่ยวกับเทคโนโลยีที่ใช้ในการสร้างระบบ

2.7 คำอธิบายการประมวลผล (Process Description)

แผนภาพกระแสข้อมูลมีข้อมูลดีคือ เป็นแผนภาพที่สามารถนำเสนอให้เห็นภาพรวมของระบบได้อย่างดีแต่สำหรับข้อเสียนั้นก็มี คือ ไม่ได้มีการแสดงรายละเอียดภายในของแต่ละโปรเซส ดังนั้นหากต้องการรายละเอียดภายในของแต่ละโปรเซสว่าโปรเซสต่าง ๆ เหล่านั้นมีกระบวนการทำงานอย่างไร จึงจำเป็นต้องจัดทำคำอธิบายการประมวลผลประกอบ หรือที่เรียกว่า Process Description

วัตถุประสงค์ของคำอธิบายการประมวล

1. **เพื่อลดความไม่ชัดเจนหรือความกำกวมของโปรเซส จุดประสงค์อันสำคัญก็คือ ใช้เป็นเกณฑ์ หรือ ข้อบังคับใช้ของนักวิเคราะห์ เพื่อการเรียนรู้รายละเอียดของแต่ละโปรเซสว่ามีกระบวนการทำงานอย่างไรเป็นสำคัญ**

2. **เพื่อความเที่ยงตรงและสามารถกระทำให้สำเร็จตามกระบวนการที่ได้อย่างถูกต้องและเข้าใจตรงกันซึ่งรายละเอียดหรือข้อกำหนดในคำอธิบายการประมวลผลถูกจัดทำได้ไม่ดี ไม่มีความเที่ยงตรง หรือมีความเข้าใจไม่ตรงกันก็อาจทำให้โปรแกรมเมอร์เข้าใจผิดทำให้โปรแกรมที่พัฒนาขึ้นมาผิดพลาด ไม่ตรงกับวัตถุประสงค์ก็เป็นได้**
3. **เพื่อใช้สำหรับการตรวจสอบในขั้นตอนของการออกแบบระบบต่อไป และเพื่อให้เกิด**ความมั่นใจว่าโปรเซสหรือกระบวนการที่ได้รับอินพุตเข้ามา ละทำการประมวลผลออกมาเป็นผลลัพธ์ที่นำเสนอบนแผนภาพกระแสข้อมูลเป็นไปอย่างถูกต้องหรือไม่

2.8 อีอาร์ไดอะแกรม (The Entity Relationship Diagram)

ในการวิเคราะห์เพื่อไม่ได้มาซึ่งแผนภาพอีอาร์ไดอะแกรมนั้น จะใช้พื้นฐานหลักๆ ประการด้วยกันคือ

1. เอ็นนิตี้ (Entitiy)
2. ความสัมพันธ์ (Relationships)
3. แอตตริบิวต์ (Attributes)

2.8.1 เอ็นติตี้ (Entities)

คือบุคคล สถานที่ วัตถุ และรวมถึงเหตุการณ์ที่ทำให้เกิดกลุ่มของข้อมูลที่ต้องการจัดเก็บซึ่งสามารถบ่งชี้ถึงความเป็นเอกลักษณ์เฉพาะตัวได้ โดยตัวอย่างของแต่ละเอ็นติตี้ประกอบด้วย

1. บุคคล(Persons) ตัวอย่างเอ็นติตี้บุคคล เช่น ลูกค้า พนักงาน นักศึกษา ร้านค้า แผนกการเงิน
2. สถานที่(Place) ตัวอย่างเอ็นติตี้เกี่ยวกับสถานที่ เช่น อาคาร ห้องเรียน สาขา
3. วัตถุ(Objects) ตัวอย่างเอ็นติตี้เกี่ยวกับวัตถุ เช่น หนังสือ เครื่องจักร สินค้า วัตถุดิบ
4. เหตุการณ์(Events) ตัวอย่างเอ็นติตี้เกี่ยวกับเหตุการณ์ เช่น ใบอินวอยซ์ (เกิดขึ้นจากเหตุการณ์การซื้อสินค้า) รายการลงทะเบียน (เกิดจากเหตุการณ์ที่นักศึกษาได้ทำการลงทะเบียนวิชาเรียน)
5. แนวความคิด(Concepts) ตัวอย่างเอ็นติตี้เกี่ยวกับแนวความคิด เช่น บัญชี พันธบัตร หุ้น

2.8.2 **ความสัมพันธ์** (Relationships)

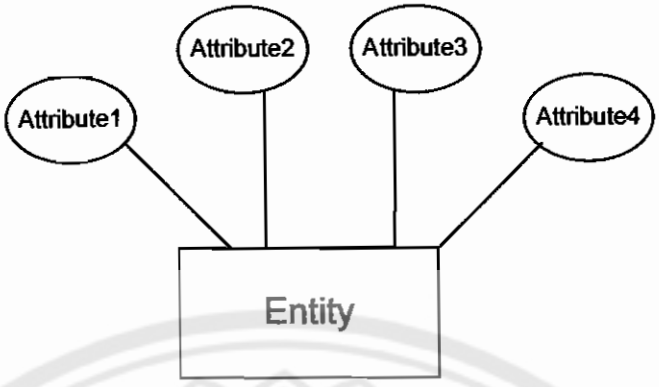
ความสัมพันธ์ในที่นี้หมายถึง ความสัมพันธ์ระหว่างเอ็นติตี้ ซึ่งความสัมพันธ์ดังกล่าวจะเป็นไปตามชนิดของแต่ละความสัมพันธ์ โดยอาจกล่าวอีกนัยหนึ่งว่า ความสัมพันธ์ของแต่ละเอ็นติตี้นี้จะเกิดขึ้นตามธรรมชาติในกระบวนการทางธุรกิจ ซึ่งความสัมพันธ์จะนำเสนอด้วยเหตุการณ์เชื่อมโยงระหว่างเอ็นตินี้

สำหรับข้อกำหนดในความสัมพันธ์(Constraints) จะเป็นกฎเกณฑ์ที่ใช้บังคับเงื่อนไขเพื่อให้การจัดเก็บข้อมูลในฐานข้อมูลเป็นไปอย่างเหมาะสมและถูกต้อง โดยข้อกำหนดความสัมพันธ์จะเป็นเงื่อนไขที่ใช้บังคับส่วนต่างๆ ในแบบจำลอง ซึ่งโปรแกรมจะต้องรักษาให้ถูกต้องตามความเป็นจริงเสมอ โดยพิจารณาจากรูป 6.2

รูปที่ 2.8 แสดงสัญลักษณ์ความสัมพันธ์

2.8.3 **แอตตริบิวต์** (Attributes)

คือคุณสมบัติของเอ็นติตี้ เช่น เอ็นติตี้ลูกค้าจะประกอบด้วยแอตตริบิวต์ รหัสลูกค้า ชื่อ นามสกุล เพศ ที่อยู่ โทรศัพท์ วันเกิด โดยสัญลักษณ์แอตตริบิวต์ในอีอาร์ไดอะแกรมจะใช้สัญลักษณ์รูปวงรี และแอตตริบิวต์ใดที่เป็นคีย์หลัก ก็จะมีการขีดเส้นใต้กำกับใต้ชื่อแอตตริบิวต์นั้น



รูปที่ 2.9 ตัวอย่างแอตตริบิวต์ของลูกค้า

2.9 ความสมดุลระหว่างอีอาร์ไดอะแกรมกับแผนภาพกระแสข้อมูล (Balancing Entity Relationship Diagrams with Data Flow Diagram)

นักวิเคราะห์ระบบจะต้องตรวจสอบกิจกรรมในขั้นตอนสการวิเคราะห์ให้มีความสัมพัธ์กันอย่างถูกต้อง ตัวอย่างเช่น แฟนภาพกระแสข้อมูล(Data and Process) ที่นักวิเคราะห์ระบบสร้างขึ้นมา ถึงแม้ว่าจะเป็นไดอะแกรมที่โฟกัสอยู่บนความสัมพันธ์ระหว่างข้อมูลแต่แบบจำลองทั้งสองจำเป็นต้องข้องเกี่ยวหรือมีความสัมพันธ์กันเสมอโดยดาต้าสโตร์ที่ใช้งานในแผนภาพกระแสข้อมูลนั้น จะต้องมีความสมดุลกับอีอาร์ไดอะแกรม กล่าวคือ กาต้าสโตร์ที่ใช้งานในแผนภาพกระแสข้อมูลก็จะต้องประกฏอยู่ในอีอาร์ไดอะแกรม นั่นก็หมายความว่าเกิดความไม่สมดุลขึ้นในระบบ ซึ่งข้อผิดพลาดดังกล่าว ทำให้ทราบว่าในระบบได้บรรรจุข้อมูลที่ไม่จำเป็นเกิดขุ้นแล้วนั่นเอง สรหรับการตรวจสอบความสมดุลระหว่างอีอาร์ไดอะแกรมกับแผนภาพกระแสข้อมูลนั้นให้พิจารณาจาจำนวยดาต้าสโตร์ที่ใช้งานในแผนภาพกระแสข้อมูลระดับที่ 1 ซึ่งจะต้องมีจำนวนเท่ากับเอ็นติตี้ในอีอาร์ไดอะแกรม

2.10 พจนานุกรมข้อมูล (Data Dictionary)

พจนานุกรมข้อมูลจะประกอบไปด้วยหน่วยข้อมูล หรือข้อมูลย่อย(Data Element) ต่างๆ ของระบบ โดยข้อมูลย่อยคือข้อมูลที่ไม่สามารถแตกย่อยออกไปได้อีก เช่น ข้อมูลลูกค้าประกอบด้วยรหัสลูกค้า ชื่อ และที่อยู่ เป็นต้น สำหรับข้อมูลย่อยเหล่านี้เมื่อนำมารวมกันก็จะเรียกว่าเรคอร์ด และในที่สุดก็จะเป็นโครงสร้างแฟ้มข้อมูลโดยพจนานุกรม คือเอกสารที่ใช้อธิบายรายละเอียดโครงสร้างแฟ้มข้อมูล และรวมถึงรายการจ้อมูลประกอบต่างๆ ซึ่งประกอบด้วยชื่อรีเลชัน(Relation name) , แอตตริบิวต์(Attribute), ชื่อแทน(Aliases name), รายละเอียดข้อมูล(Data

Description), แอตตริบิต์โดเมน(Attribute Domain), การเรียงลำดับดัชนี(Index), คีย์หลัก (Primary Key), คีย์นอก(Foreign Key), ชนิดข้อมูล(Data Type) ว่าเป็นแบบตัวอักษร ตัวเลข และมีขนาดความกว้างเท่าไหร่ นอกจากนี้พจนานุกรมข้อมูลยังรวมถึงรายละเอียดเกี่ยวกับ แหล่งที่เกิดข้อมูล, วันที่สร้างแฟ้มข้อมูล, ผู้ใช้ระบบ, สิทธิการใช้แฟ้มข้อมูล, ความถี่ในการใช้งาน และอื่นๆ โดยสัญลักษณ์และความหมายที่ใช้อธิบายเกี่ยวกับโครงสร้างข้อมูล (Data Structures) ดังตารางที่ 2.1

ตารางที่ 2.1 สัญลักษณ์และความหมายที่ใช้ในพจนานุกรมข้อมูล

สัญลักษณ์	ความหมาย
= (Equal Sign)	ประกอบด้วย
+ (Plus Sign)	และ
{ } (Braces)	การกระทำซ้ำของข้อมูลย่อย
[ ] (Brackets)	การพิจารณาทางเลือกเพียงทางเลือกหนึ่ง
( ) (Optional)	จะมีหรือไม่มีก็ได้

2.11 การนอร์มัลไลเซชัน (Normalization)

การนอร์มัลไลเซชัน เป็นกระบวนการนำโครงร่างรีเลชัน (Relation) มาแตกแป็นรีเลชันหรือตารางต่างๆ ให้อยู่ในรูปที่เรียกว่า รูปแบบบรรทัดฐานหรือ Normal Form โดยมีเป้าหมายหลักสำคัญคือ เพื่อให้รีเลชันที่ได้มานั้นอยู่ในรูปแบบบรรทัดฐานที่เหมาะสม โดยจุดประสงค์ของการนอร์มัลไลเซชันคือ

2.11.1 ลดเนื้อที่ในการจัดเก็บข้อมูล กระบวนการนอร์มัลไลเซชันเป็นการออกแบบเพื่อลดความซ้ำซ้อน(Redundancy) ในข้อมูล ดังนั้น การลดความซ้ำซ้อนในข้อมูลก็ย่อมทำให้ลดเนื้อที่ในการจัดเก็บข้อมูลตามมาด้วย

2.11.2 ลดปัญหาความไม่ถูกต้องของข้อมูล เมื่อข้อมูลไม่เกิดความซ้ำซ้อน ในการปรับปรุงข้อมูลก็จะสามารถทำการปรับปรุงข้อมูลได้จากแหล่งข้อมูลเพียงแหล่งเดียว จึงทำให้ช่วยลดความผิดพลาดที่อาจเกิดขึ้นจากการปรับปรุงข้อมูล (Update Anomalies) ซึ่งประกอบด้วย ข้อผิดพลาดจากการเพิ่มข้อมูล (Insertion Anomalies) ข้อผิดพลาดจากการลบข้อมูล (Deletion Anomalies) และข้อผิดพลาดจากการเปลี่ยนแปลงข้อมูล (Modification Anomalies)

2.12 ความรู้พื้นฐานเกี่ยวกับ ASP (Active Sever Page)

2.12.1 ความหมายของ ASP

ASP หรือ Active Server Pages เป็นโปรแกรมตีความภาษา (Interpreter) ที่ใช้ในการตีความเว็บเพจที่เขียนขึ้นมาโดยใช้ไวยากรณ์หรือ syntax ของภาษา VBscript (ซึ่ง VBscript ก็อาศัยโครงสร้างของภาษา Visual Basic อีกที) แล้วสร้างเว็บเพจผลลัพธ์ขึ้นมา จากนั้นก็จะส่งไปให้ web server เพื่อที่จะให้ web server ส่งต่อไปยัง browser อีกที เนื่องจาก ASP จะต้องทำงานโดยการร้องขอของ web server ดังนั้นจึงจะต้องมีโปรแกรม ASP ติดตั้งที่ web server ด้วย โดยที่ปัจจุบัน เมื่อพูดถึง ASP มักจะหมายถึงเป็น ASP ที่ทำงานในวินโดว์ NT หรือ วินโดว์ 95, 98 (ใช้กับธุรกิจหรืองาน ที่ปริมาณการติดต่อไม่มากนัก หรือใช้ในการทดสอบเพื่อการพัฒนางานไปสู่ระบบใหญ่ต่อไป)

2.12.2 การเขียนโปรแกรมภาษา ASP

ASP เป็นผลิตภัณฑ์ของไมโครซอร์ฟ ปัจจุบัน ASP จะถูกใส่เข้าไปในโปรแกรมที่เป็น web server ของไมโครซอร์ฟ ดังนั้นไม่จำเป็นต้องทำการติดตั้งโปรแกรม ASP อีก กล่าวคือสามารถเรียกใช้ได้เลย ซึ่งการเขียน เว็บเพจให้เป็น .asp สามารถใช้โปรแกรมสร้างข้อความธรรมดาทั่ว ๆ ไปก็ได้ เช่น ใช้ Notepad ในการเขียน .asp การบันทึกก็ให้บันทึกเป็น นามสกุล .asp

การเขียน ASP script จะทำโดยการฝังหรือ embedded ส่วนที่เป็น script ลงไปในเว็บเพจ กล่าวคือหากไม่มีการฝัง ASP script เลยเว็บเพจนั้นก็คือเว็บเพจธรรมดาทั่วไปนั้นเอง การตีความโดย ASP ก็จะทำการตีความไล่ลงไปจากต้นไฟล์ไปทีละบรรทัด บรรทัดไหนมีส่วนของ ASP script อยู่ก็จะทำการตีความก่อนแล้วทยอยส่งผลลัพธ์ออกมาเรื่อยๆ หากเกิดข้อผิดพลาดที่รุนแรงก็จะหยุดการทำงาน ส่วนที่เป็น ASP script จะขึ้นต้นหรือเปิดด้วย tag โดยใช้เครื่องหมาย <% และลงท้ายหรือปิดด้วย %>