บทที่ 2

แนวเหตุผล ทฤษฎีสำคัญหรือสมมุติฐาน

2.1 โครงสร้างของซอร์ฟแวร์ (SOFTWARE STRUCTURE)

จะเป็นวิธีการออกแบบเพื่อหาแนวทางที่จะแก้ไขปัญหาที่เกิดขึ้นแนวทางแก้ไขอาจจะมีได้หลายอย่าง แต่จะเลือกแนวทางที่ดีที่สุดและง่ายต่อการพัฒนาโปรแกรมต่อไป โดยในที่นี้จะใช้วิธีการออกแบบที่เรียกว่า "TOP DOWN" ซึ่งจะมีขั้นตอนดังนี้คือ

TOP-DOWN DESIGN เป็นการออกแบบที่แยกปัญหาที่มีขนาดใหญ่ และซับซ้อนให้มีขนาดเล็กและซับซ้อนน้อยลงได้ ซึ่งแต่ละส่วนที่ออกแบบจะเรียกว่า "โมดูล (Module)"

TOP-DOWN CODING เป็นการเขียนโปรแกรมอยู่ในลักษณะของ โมดูลย่อยตามตามที่ได้ออกแบบไว้

TOP-DOWN TESTING เป็นการตรวจสอบความถูกต้องของการทำงาน ในแต่ละโมดูลย่อย ตามที่ได้เขียนโปรแกรมไว้

นอกจากนั้นแล้วยังมีการใช้วิธีที่เรียกว่า "DECOMPOSITION" ซึ่งจะเป็นการแยกชุดของโปรแกรมออกเป็นส่วนย่อยๆ เพื่ออธิบายลักษณะการทำงานและโครงสร้างข้อมูลที่จะใช้ของโปรแกรมนั้น

2.2 ลักษณะที่ดีของการออกแบบ (CHARACTERISTICS OF GOOD DESIGN)

ลักษณะการออกแบบที่ดี จะต้องประกอบด้วยหลักการดังนี้คือ

2.2.1 การเป็นโมดูล (MODULARITY)

จะเป็นการแบ่งแยกโปรแกรมออกเป็นส่วนๆที่มีการตั้งชื่อและระบุส่วนประกอบที่จะใช้ด้วยซึ่งเราจะเรียกว่า "โมดูล" การเป็นโมดูล (MODULARITY) จะช่วยให้เราสามารถมองเห็นปัญหาที่ซับซ้อนได้ทั้งหมด เพราะจะเป็นการไปแยกส่วนของปัญหาที่ซับซ้อนออกเป็นกลุ่มที่สามารถเข้าใจได้ง่ายขึ้น

2.2.2 ระดับของการจินตนาภาพ (LEVEL OF ABSTRACTION)

โมดูลที่ระดับหนึ่ง ควรสามารถรวมกันให้เป็นโปรแกรมย่อยที่อยู่ระดับเหนือขึ้นไปได้อีก โดยเมื่อเราพิจารณาระดับสูงขึ้นแล้วจะเห็นว่าเราอยู่ในระดับของการจินตนาภาพ (LEVELOF ABSTRACTION) แต่เมื่อเราเคลื่อนย้ายมายังระดับที่ต่ำกว่าเราจะพบรายละเอียดเกี่ยวกับแต่ละโปรแกรมย่อยมากขึ้น ดังนั้นจึงสามารถกล่าวได้ว่าระดับของการจินตนาภาพ จะช่วยให้เราที่จะเข้าใจปัญหาการอ้างอิงตำแหน่งที่โดยระบบงานได้โดยการตรวจสอบเริ่มจากระดับบน และทำงานลงมายังระดับล่างต่อไป

2.2.3 การซ่อนข้อมูล (INFORMATION HIDING)

โมดูลย่อย ควรจะเก็บซ่อนรายละเอียดของข้อมูลภายใน และการประมวลผล (PROCESSING) จากโมดูลอื่นๆ และจะติดต่อกับโมดูลอื่นๆ โดยผ่านข้อมูลที่จำเป็นเท่านั้น ผลที่ได้จะทำให้โมดูลย่อยใดๆ มีความเป็นอิสระมากขึ้นจากโมดูลย่อยอื่นๆ จึงง่ายต่อการทดสอบและพัฒนาโปรแกรมต่อไป

2.2.4 การจับคู่ (COUPLING) [1][2][3]

จะเป็นเครื่องมือวัดความสัมพันธ์กันระหว่างโมดูลย่อย ซึ่งโดยปกติการออกแบบระบบจะลดการจับคู่ระหว่างโมดูลให้น้อยที่สุด เพื่อให้โมดูลย่อยมีความเป็นอิสระต่อกันให้มากที่สุด การจับคู่ (COUPLING) จะเกิดขึ้นได้ ต้องขึ้นอยู่กับหลายสิ่งดังนี้คือ

- การติดต่อกันระหว่างโมดูล จะเป็นการติดต่อกันระหว่างโมดูลย่อยทั้งสอง เช่น โมดูลย่อย A มีความสัมพันธ์ติดต่อกับโมดูลย่อย B ดังนั้นการใช้งานของโมดูลย่อย A จะขึ้นอยู่โมดูลย่อย B

- จำนวนข้อมูลที่ผ่านกันระหว่างโมดูล เช่น โมดูลย่อย A อาจจะผ่านค่าของข้อมูลไปยัง โมดูลย่อย B ดังนั้น โมดูลย่อย A ก็จะขึ้นอยู่กับโมดูลย่อย B

- จำนวนการควบคุมของโมดูลที่มีผลต่อโมดูลย่อยอื่นๆ เช่น โมดูลย่อย A อาจจะผ่านค่าไปยังโมดูลย่อย B ด้วยค่าควบคุม (CONTROL FLAG) ซึ่งโมดูลย่อย B จะกระทำโดยขึ้นอยู่กับค่าควบคุมนั้นๆ

- ระดับของความซับซ้อนในการติดต่อระหว่างโมดูล เช่น ถ้าโมดูลย่อย A จะผ่านค่าควบคุมหนึ่งค่าไปยังโมดูลย่อย B ทันทีแต่สำหรับโมดูลย่อย C และโมดูลย่อย D จะต้องมีการเปลี่ยนค่าควบคุมก่อนที่โมดูลย่อย C จะส่งไปให้โมดูลย่อย D ดั้งนั้นการติดต่อระหว่างโมดูลย่อย A และ B จะมีความซับซ้อนน้อยกว่าโมดูลย่อย C และ D

ชนิดของการ COUPLING เรียงตามความสัมพันธ์จากระดับสูงไปหาระดับต่ำจะมีดังนี้คือ

2.2.4.1 CONTENT COUPLING จะเกิดขึ้นเมื่อโมดูลย่อยหนึ่งเข้าไปเรียกใช้รายการข้อมูลภายในของอีกโมดูลย่อยหนึ่ง หรือเมื่อโมดูลย่อยหนึ่งแยกออกไปทำงานภายในของอีกโมดูลย่อยหนึ่ง

2.2.4.2 COMMON COUPLING จะเกิดขึ้นเมื่อมีการกำหนดค่ารายการข้อมูลในลักษณะของค่าข้อมูลใช้ได้ทั่วไป (Global Data Area) ซึ่งถ้ามีการเปลี่ยนแปลงค่าใน Global Data Area ก็จะมีผลกับโมดูลย่อยอื่นๆที่จะใช้ข้อมูลนั้นๆ

2.2.4.3 CONTROL COUPLING จะเป็นลักษณะที่โมดูลย่อยหนึ่งผ่านค่าควบคุมไปควบคุมการทำงานของโมดูลย่อยอื่นๆได้ ข้อได้เปรียบของแบบนี้คือ จะสามารถให้แต่ละโมดูลย่อยทำงานตามได้เพียง 1 ฟังก์ชั่น ซึ่งจะทำให้จำนวนของข้อมูลของการควบคุมที่ต้องผ่านจากโมดูลย่อยหนึ่งไปอีกโมดูลย่อยหนึ่ง มีได้น้อยลงไป

2.2.4.4 STAMP COUPLING จะเป็นลักษณะที่โปรแกรมย่อยหนึ่งผ่านค่าของโครงสร้างของข้อมูลจากโมดูลย่อยหนึ่งไปยังอีกโปรแกรมย่อยหนึ่งได้ โดยค่าของข้อมูลรูปแบบ และการจัดองค์ประกอบ จำเป็นต้องให้ตรงกันระหว่างโมดูลด้วยกันได้

2.2.4.5 DATA COUPLING จะเป็นลักษณะที่โมดูลย่อยหนึ่งจะผ่านค่าของข้อมูลไปยังอีกโมดูลย่อยหนึ่งได้ แบบนี้จะเป็นแบบที่ง่ายที่สุดที่จะติดตามการทำงานและแก้ไข

2.2.5 การเชื่อมแน่น (Cohesion) [1][2][3]

เป็นการวัดความสัมพันธ์กันภายในโมดูลย่อยโดยปกติจะทำให้ระบบมีการเชื่อมแน่นมากที่สุด เพื่อให้ทุกๆส่วนของการทำงานในโมดูลย่อยได้มีความสัมพันธ์กันได้เป็นฟังก์ชั่นเดียวกัน

ชนิดของการเชื่อมแน่น ซึ่งจะเรียงตามความสัมพันธ์จากระดับต่ำไปหาระดับสูงจะมีดังนี้คือ

2.2.5.1 COINCIDENTAL COHESION

จะเป็นโมดูลย่อย ที่ส่วนประกอบภายในแต่ละส่วนไม่ได้มีความสัมพันธ์กันเลย เช่น ภายในโมดูลย่อยอาจจะประกอบด้วยส่วนในการตรวจสอบรหัสผู้ใช้งาน และส่วนของการพิมพ์ค่าแรงประจำสัปดาห์ ซึ่งทั้งสองส่วนนี้ไม่ได้มีความสัมพันธ์กันเลย

2.2.5.2 LOGICAL COHESION

จะเป็นโมดูลย่อยที่หลายๆส่วนเข้ามาสัมพันธ์กันด้วยเหตุผล เช่น ถ้าตรวจสอบพบว่าค่าของตัวแปรควบคุมเป็นค่าศูนย์ ให้ไปทำงานในโมดูลย่อย A แต่ถ้าเป็นค่าหนึ่งให้ไปทำงานในโมดูลย่อย B โดยทั้งโมดูลย่อย A และ โมดูลย่อย B อาจจะไม่มีสัมพันธ์กันเลย แต่ต้องมาสัมพันธ์กันตามเงื่อนไข

2.2.5.3 TEMPORAL COHESION

จะเป็นโมดูลย่อยที่มีการทำงานหลายๆอย่าง เกิดขึ้นเป็นลำดับในเวลาเดียวกันแต่การทำงานเหล่านั้นก็ไม่ได้มีความสัมพันธ์กัน เช่น ในโมดูลย่อยมีการสั่งให้ทำงานได้ใน DRIVE A และในขณะเดียวกันก็มีการสั่งให้ทำงานได้ใน DRIVE B

2.2.5.4 COMMUNICATIONAL COHESION

จะเป็นโมดูลย่อยที่มีการกระทำหรือฟังก์ชั่นซึ่งแตกต่างกัน แต่มีการเรียกใช้ข้อมูลในชุดเดียวกัน เช่นเรามีการรับข้อมูลรหัสของลูกค้าเพื่อจะไปใช้กับการกระทำที่มีการอ่านชื่อลูกค้าและอ่านเงินเดือนของลูกค้า ซึ่งจะเห็นว่าการกระทำทั้งสองจะเรียกใช้ข้อมูลรหัสของลูกค้าเดียวกัน

2.2.5.5 SEQUENTIAL COHESION

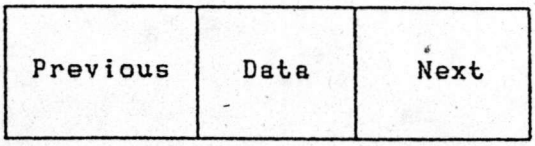
จะเป็นโมดูลย่อยที่ข้อมูลผลลัพธ์จากส่วนหนึ่งของโมดูลย่อย จะกลายเป็นข้อมูลรับเข้าของอีกส่วนหนึ่งของโมดูลย่อยได้ เช่น ส่วนหนึ่งภายในโมดูลย่อยจะคำนวณหาค่าเงินรายได้ทั้งหมดของพนักงานแล้วจะนำผลลัพธ์ที่ได้ไปคำนวณในอีกส่วนหนึ่งเพื่อหาค่าภาษี

2.2.5.6. FUNCTIONAL COHESION

จะเป็นโมดูลย่อยที่ทุกการทำงานของส่วนประกอบย่อย มีความจำเป็นต่อการกระทำของหนึ่งฟังก์ชั่น เช่น ภายในโมดูลย่อยประกอบด้วยการอ่านข้อมูลของลูกค้า คำนวณหาค่าเงินเดือนและการพิมพ์ข้อมูลของลูกค้า จะเห็นว่าแต่ละส่วนประกอบย่อยจะมีความสำคัญเท่ากันซึ่งจะต้องมาทำงานร่วมกันได้

2.3 โครงสร้างข้อมูล (DATA STRUCTURE)

สำหรับการจัดเก็บข้อมูลภายในเครื่องจะใช้โครงสร้างข้อมูลเป็นแบบ Double Linked List ซึ่งจะเป็น Linked List ที่แต่ละโหนดประกอบด้วยฟิลด์ข้อมูล (Data) และฟิลด์พอยน์เตอร์ (Pointer) 2 ฟิลด์ ซึ่งจะใช้สำหรับชี้ไปยังตำแหน่งที่อยู่ของโหนดก่อนหน้า (Previous) และตำแหน่งที่อยู่ของโหนดถัดไป (Next) ซึ่งจะแสดงโครงสร้างโหนดได้ดังรูปที่ 2.1 คือ



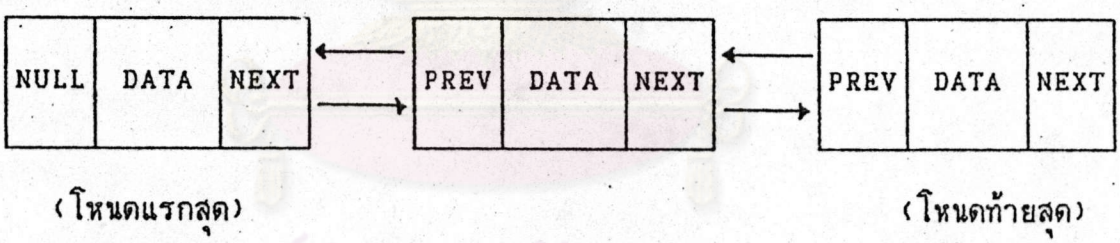
โดย Previous จะเป็นพอยน์เตอร์ที่จะชี้ไปยังตำแหน่งที่อยู่ของโหนดก่อนหน้า

Next จะเป็นพอยน์เตอร์ที่จะชี้ไปยังตำแหน่งที่อยู่ของโหนดถัดไป

Data จะเป็นที่เก็บของข้อมูล

รูปภาพที่ 2.1 แสดงโครงสร้างของข้อมูลแบบ Double Linked List

สำหรับการจัดเก็บข้อมูลภายในเราจะกำหนดให้พอยน์เตอร์ของโหนดก่อนหน้าตัวแรกสุด และพอยน์เตอร์ของโหนดถัดไปตัวสุดท้าย ให้เป็นค่า NULL (ค่าว่างเปล่า) โดยมีการแสดงดังรูปที่ 2.2 ดังนี้คือ



รูปภาพที่ 2.2 แสดงการจัดเก็บข้อมูลภายในแบบ Double Linked List