บทที่ 4

การออกแบบระบบ

ในบทที่แล้วนั้น เราทำงานร่วมกับลูกค้าเพื่อกำหนดวัตถุประสงค์ของระบบ สำหรับขั้นตอนต่อไปเป็นหาหนทางในการแก้ปัญหาต่างๆเหล่านั้น โดยเริ่มจากออกแบบตามแนวคิด(conceptual design) เป็นโครงร่างของการแก้ปัญหาในภาพรวม และออกแบบในรายละเอียดทางด้านเทคนิค (technical design) มีการประชุมปรึกษาหารือถึงคุณภาพของการออกแบบเพื่อหาหนทางที่ดีที่สุดในการแก้ปัญหา เพื่อให้การออกแบบมีคุณภาพสูง

4.1

การออกแบบระบบคืออะไร

การออกแบบระบบคือ การเปลี่ยนปัญหาไปเป็นรายละเอียดของการแก้ปัญหา ซึ่งผลของการออกแบบอาจอยู่ในรูปของขั้นตอนของเป็นการประมวลผล(process) ที่สามารถแก้ปัญหาตามที่ลูกค้าต้องการได้ โดยนักออกแบบระบบต้องมีการติดต่อกับลูกค้าเพื่อให้เข้าใจว่าอะไร(what) ที่ลูกค้าต้องการให้ระบบสามารถกระทำได้ นำข้อมูลที่ได้มาออกแบบเพื่อหาหนทางในการแก้ปัญหา โดยคิดหารูปแบบว่าควรทำอย่างไร(how) จึงจะเหมาะสมที่สุด ซึ่งผลของการออกแบบระบบจะถูกส่งต่อให้กับผู้สร้างระบบต่อไป

การออกแบบระบบนั้นสามารถแบ่งการประมวลผลเป็น 2 ส่วน คือ

1. Conceptual system design เป็นการออกแบบระบบโดยบรรยายถึงหน้าที่ต่างๆ (function) ที่ระบบสามารถกระทำได้ วิธีการเขียนนั้นต้องเขียนด้วยภาษาที่ลูกค้า

CT 484 115

1. Conceptual system design เป็นการออกแบบระบบโดยบรรยายถึงหน้าที่ต่างๆ (function) ที่ระบบสามารถกระทำได้ วิธีการเขียนนั้นต้องเขียนด้วยภาษาที่ลูกค้า

เข้าใจได้ง่าย ไม่อ้างอิงถึงเทคนิคที่ยุ่งยาก มีการอ้างอิงการแก้ปัญหาจากเอกสารกำหนดความต้องการ และเป็นอิสระต่อการสร้างระบบ

2. Technical design เป็นการออกแบบระบบทางด้านเทคนิคซึ่งบรรยายถึงรูปแบบ (form) ของระบบที่สามารถกระทำได้ ในส่วนของสถาปัตยกรรมระบบ ,โครงร่างฮาร์ดแวร์ที่ใช้ ,ซอฟต์แวร์ที่ต้องการ, การติดต่อสื่อสาร, ข้อมูลเข้าและออกของระบบ, รูปแบบของข่ายงาน, โครงสร้างของข้อมูล ซึ่งสรุปได้เป็น 3 กลุ่มดังนี้

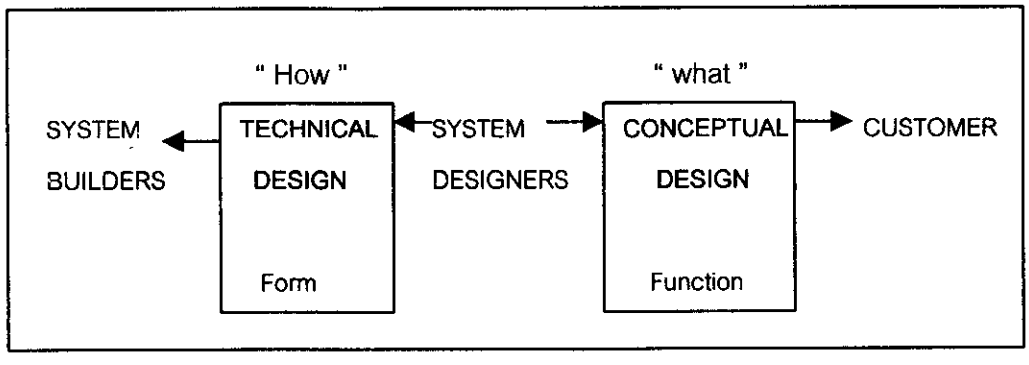
   2.1 system architecture บรรยายถึงองค์ประกอบหลักทางฮาร์ดแวร์ และ หน้าที่ที่กระทำในระบบ

   2.2 system software structure บรรยายถึงองค์ประกอบต่างๆของซอฟต์แวร์ หน้าที่, ลำดับชั้น , ความเกี่ยวพันธ์ของหน้าที่ต่างๆ

   2.3 data บรรยายถึงโครงสร้างข้อมูล และ การไหลของข้อมูลในระบบ

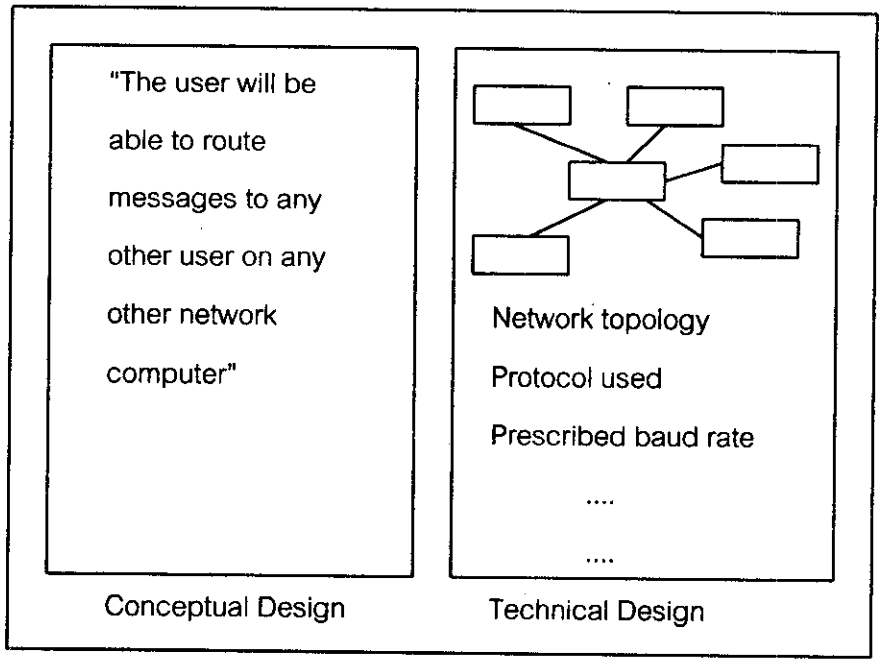
รูปภาพที่ 4.1 แสดงถึงการออกแบบระบบ

รูปภาพที่ 4.1 Two Parts of System Design



รูปภาพที่ 4.2 เป็นการเปรียบเทียบระหว่างการออกแบบตามแนวคิด กับการออกแบบทางเทคนิค

รูปภาพที่ 4.2 Conceptual Design vs. Technical Design



จะเห็นได้ว่าเอกสารในการออกแบบตามแนวคิด เป็นการบรรยายถึงสิ่งที่ระบบสามารถกระทำได้ในที่นี้เพียงแค่บอกแต่เพียงว่า ผู้ใช้ระบบสามารถส่งข้อมูลไปให้กับผู้ใช้ระบบอื่นๆในข่ายงานได้ แต่สำหรับการออกแบบเทคนิคนั้นต้องออกแบบถึงสถาปัตยกรรมของฮาร์ดแวร์ที่ใช้ด้วย เช่นต้องออกแบบรูปแบบของข่ายงาน โปรโตคอลที่ใช้ในระบบ อัตราการส่งผ่านข้อมูล และอื่นๆ ที่ผู้พัฒนาระบบนำไปสร้างระบบต่อไป

4.2

วิธีการออกแบบระบบ

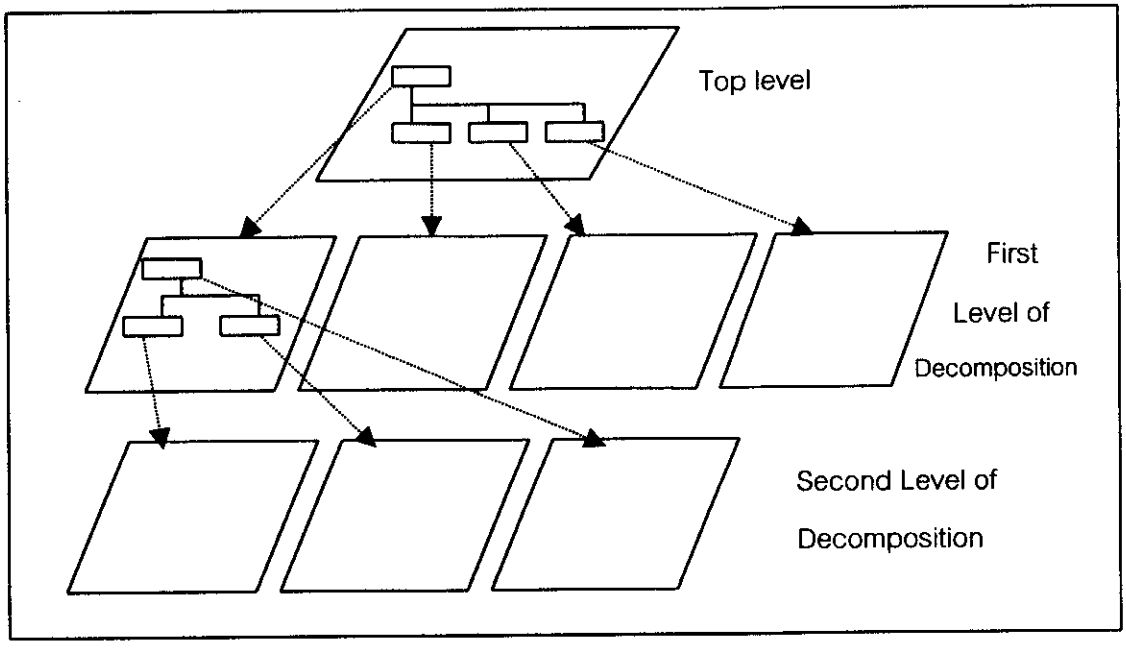
ก่อนอื่นเริ่มจากการนำเอกสารกำหนดความต้องการมาทำความเข้าใจถึงปัญหา และสิ่งที่ต้องกระทำในระบบ โดยมองระบบเป็นนามธรรมเพื่อหาหนทางแก้ปัญหาทางตรรก เริ่มจากแยก

ระบบออกเป็นส่วนย่อยๆเป็นกลุ่มของข้อมูลเข้า, กลุ่มของข้อมูลออก , กลุ่มของโมดูลต่างๆ โดยให้โมดูลต่างๆเป็นอิสระต่อกันมากที่สุดและมีเพียงวัตถุประสงค์เดียวในการปฏิบัติการ ซึ่งสรุปได้ว่าการออกแบบเป็นการกำหนดโมดูลการทำงานต่างๆของระบบรวมทั้งการปฏิสัมพันธ์ต่อกัน

เราสามารถออกแบบโมดูลต่างๆของระบบได้ 2 วิธีคือ

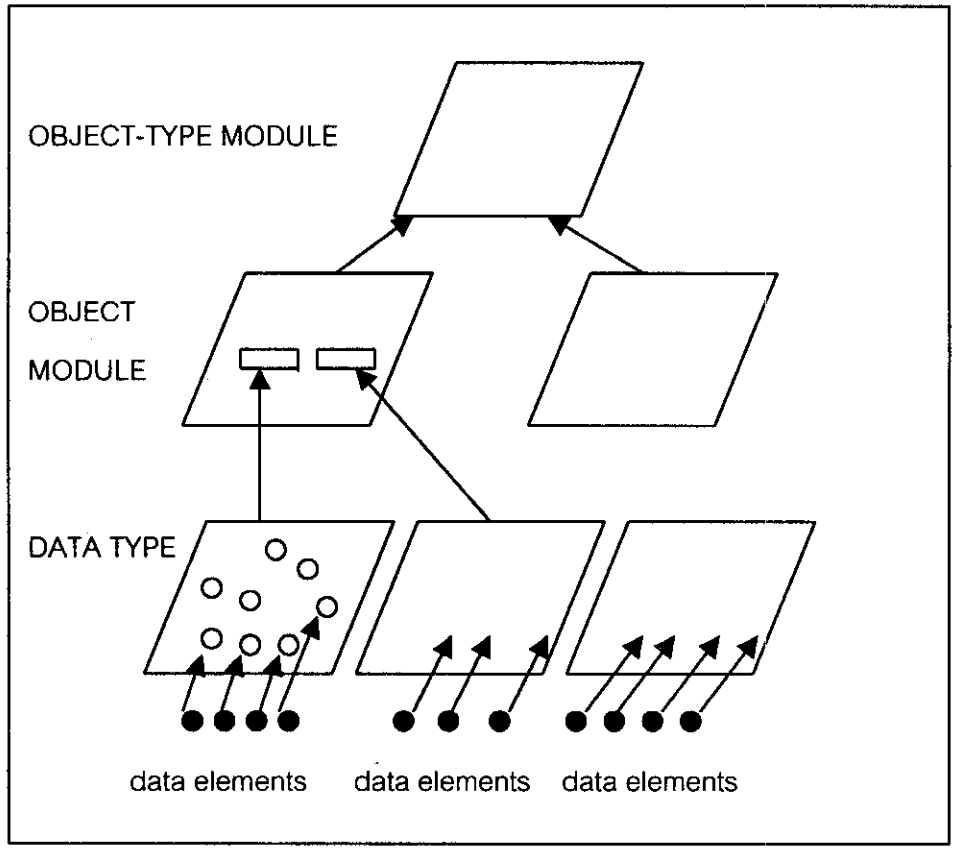
1. Decomposition เป็นการมองภาพรวมของระบบในระดับสูงที่สุดก่อนต่อจากนั้นแบ่งระบบใหญ่ออกเป็นระบบย่อยๆ ในระดับต่อมา ในแต่ละระดับจะพิจารณาถึงกลุ่มของโมดูลย่อยที่เกี่ยวพันธ์กันเพื่อแบ่งออกเป็นระดับถัดมาจนกระทั่งไม่สามารถแบ่งเป็นโมดูลย่อยได้อีก เมื่อออกแบบเสร็จต้องตรวจสอบด้วยว่ามีการระบุหน้าที่ต่างๆครบถ้วนและสมบูรณ์หรือไม่ และผลลัพธ์ของการทำงานนั้นถูกต้องตามความต้องการของลูกค้าหรือไม่ โดยทำการปรับเปลี่ยนเพื่อให้ตรงกับสิ่งที่กำหนดไว้ในเอกสารกำหนดความต้องการ ดังในรูปภาพที่ 4.3 แสดงการแบ่งระบบเป็นระดับชั้นต่างๆ โดยเริ่มจากระดับสูงที่สุด และแบ่งเป็นโมดูลย่อยๆในระดับถัดๆมา

รูปภาพที่ 4.3 Decomposition of System into Levels



2. Composition เป็นการออกแบบอีกวิธีหนึ่งที่พิจารณาส่วนประกอบย่อยก่อนและค่อยๆเพิ่มความสัมพันธ์เป็นจนเป็นส่วนประกอบที่ใหญ่ขึ้น จนกระทั่งเป็นระบบในที่สุด โดยทั่วไปเราจะพิจารณาจากนามธรรมข้อมูลและชนิดของข้อมูลก่อน ต่อจากนั้นพิจารณาถึงหน้าที่ในระบบที่มีการปฏิสัมพันธ์กับข้อมูลต่างๆเหล่านี้ อทิเช่นการสร้าง, การสอบถาม , การแก้ไขปรับปรุง และการลบ เราค่อยๆสร้างระบบจากกิจกรรมของข้อมูลโดยสร้างเป็นโมดูลของเครื่องมือและโมดูลของการจัดการกับข้อมูล ต่อจากนั้นรวมโมดูลย่อยต่างๆที่มีความสัมพันธ์กันไว้ด้วยกันเป็นโมดูลที่ใหญ่ขึ้น ใหญ่ขึ้นเป็นลำดับ จนกระทั่งเป็นระบบที่ต้องการ ดังตัวอย่างในรูปภาพที่ 4.4 ซึ่งแสดงระบบที่สร้างขึ้นจากความสัมพันธ์ของข้อมูล

รูปภาพที่ 4.4 Composition of a System from Data



จากรูปภาพที่ 4.4 แสดงการออกแบบระบบโดยเริ่มจากข้อมูลในระดับล่างสุด พิจารณาถึงกลุ่มของข้อมูลเพื่อกำหนดเป็นชนิดของข้อมูลในลำดับถัดมา ต่อจากนั้น พิจารณาถึงชนิดของข้อมูลที่มีความสัมพันธ์กัน นำมารวมกันเป็น object module และจากความสัมพันธ์ของ object module ต่างๆนำมารวมกันเป็น object-type modules ตัวอย่างของการออกแบบวิธีนี้คือตัวแปลภาษา ซึ่งเราจะมีการออกแบบจากความสัมพันธ์ของตัวอักขระจากคำเป็นนิพจน์เป็นคำสั่งและเป็นโปรแกรมในที่สุด

4.3

คุณลักษณะของการออกแบบระบบที่ดี

ไม่ว่าเราจะออกแบบด้วยวิธีใดก็ตาม ผลของการออกแบบนั้นต้องง่ายต่อความเข้าใจและแก้ไข เนื่องจากในการพัฒนาระบบอาจมีการแก้ไขหรือเปลี่ยนแปลงการออกแบบ เมื่อมีความต้องการระบบที่เปลี่ยนไปหรือมีข้อผิดพลาดเกิดขึ้น ดังนั้นนักออกแบบควรตระหนักถึงการออกแบบที่มีคุณลักษณะที่ดี ดังมีรายละเอียดดังนี้

1. Modularity การออกแบบที่ดีนั้นต้องมีการแบ่งปัญหาของระบบออกเป็นโมดูลย่อยๆโดยแต่ละโมดูลให้มีความเกี่ยวพันธ์กันน้อยที่สุดและให้ผลลัพธ์การทำงานของแต่ละโมดูลมีเพียงวัตถุประสงค์เดียว การแบ่งระบบออกเป็นโมดูลย่อยนี้เพื่อแยกปัญหาใหญ่ออกเป็นปัญหาย่อยๆ ทำให้เราสามารถแก้ปัญหาแต่ละโมดูลย่อยได้ง่าย
2. Levels of Abstraction เมื่อมีการแยกปัญหาออกจากกันแล้วต้องมีการจัดระดับของ โมดูล ซึ่งโมดูลในระดับต่ำกว่าเป็นรายละเอียดที่กำหนดจากโมดูลในระดับที่สูงกว่า สำหรับโมดูลในระดับสูงสุดจะสำคัญที่สุดเพราะเราสามารถหารายละเอียดของปัญหาที่ซ่อนอยู่ในระดับที่ต่ำลงมาได้ ซึ่งการออกแบบเป็นระดับนี้ทำให้เราเข้าใจถึงปัญหาและปัญหาที่ซ่อนอยู่ ซึ่งสามารถติดตามได้เมื่อเลื่อนไปยังระดับที่ต่ำลงมา นอกจากนี้การออกแบบเป็นระดับนี้ทำ

ให้ง่ายต่อการเปลี่ยนแปลงแก้ไขเพราะแต่ละโมดูลเป็นอิสระต่อกันเมื่อมีการแก้ไขในโมดูลหนึ่งจะส่งผลกระทบน้อยมากต่อโมดูลอื่นๆ

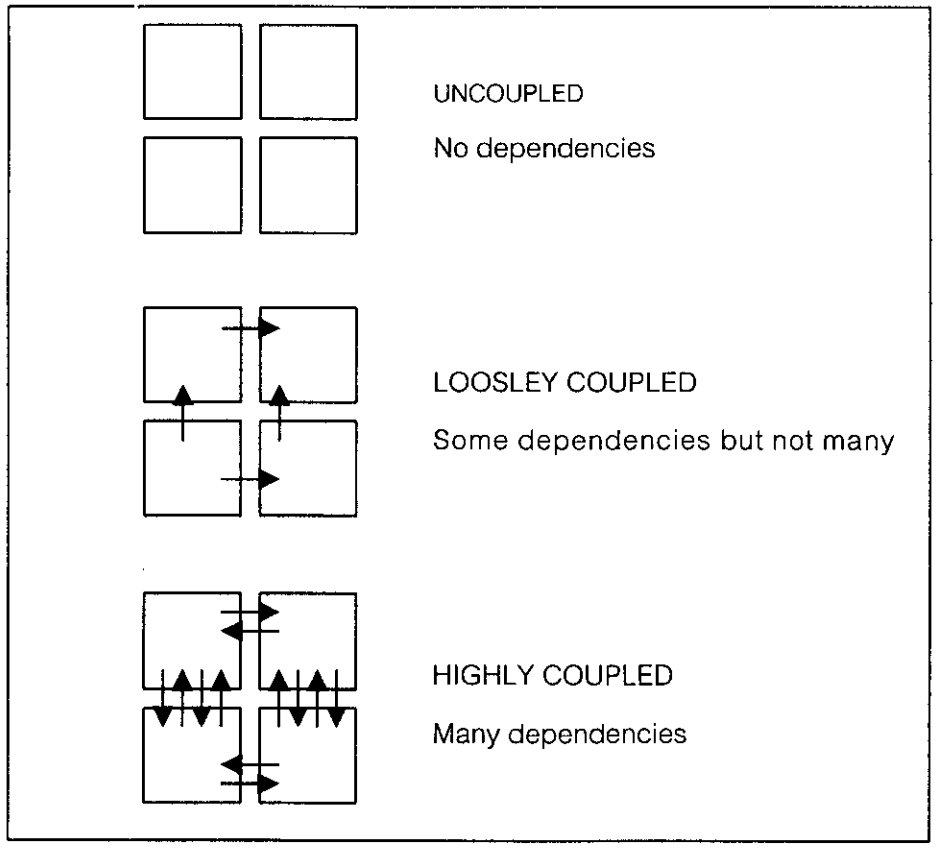
3. Coupling เป็นการวัดความสัมพันธ์ระหว่างโมดูล การออกแบบที่ดีนั้นความสัมพันธ์ระหว่างโมดูลนั้นต้องมีความเกี่ยวพันธ์กันน้อยที่สุด เราสามารถวัดความเกี่ยวพันธ์นี้ได้จากปัจจัยในหลายๆสิ่ง ได้แก่
   - Reference วัดจากการอ้างอิง เช่นโมดูล A ถูกเรียกโดยโมดูล B ดังนั้นโมดูล A ขึ้นกับโมดูล B เป็นต้น
   - Amount of data วัดจากจำนวนของข้อมูลที่มีการส่งผ่านจากโมดูลหนึ่งไปยังอีกโมดูลหนึ่ง เช่นโมดูล A ส่งข้อมูลชนิดอาเรย์ ไปให้ โมดูล B ดังนั้น โมดูล B ขึ้นกับโมดูล A
   - Amount of control วัดจากการควบคุมโมดูล เช่นโมดูล A ส่งสัญญาณควบคุมไปให้โมดูล B ดังนั้นกิจกรรมที่โมดูลB ต้องกระทำจะถูกควบคุมโดยค่าของสัญญาณที่ส่งไป
   - Degree of complexity วัดจากระดับความซับซ้อนในการส่งผ่านระหว่างโมดูล ซึ่งเราสามารถวัดความเกี่ยวพันธ์ระหว่างโมดูลได้จากตัวอย่างในรูปภาพที่ 4.5

ความเกี่ยวพันธ์ของโมดูลนั้นเราสามารถแบ่งออกเป็น 5 ชนิดซึ่งแต่ละชนิดมีความเกี่ยวพันธ์ที่แตกต่างกันไป รูปภาพที่ 4.6 แสดงชนิดต่างๆของความเกี่ยวพันธ์ระหว่างโมดูล

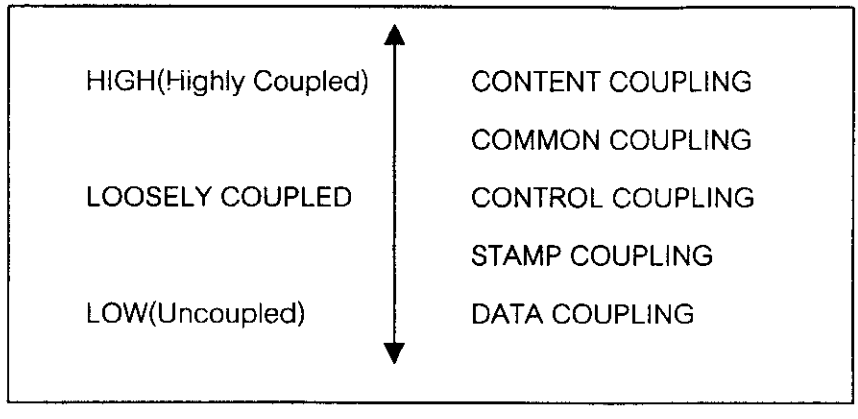
Content coupling มีความเกี่ยวพันธ์ระหว่างโมดูลสูงมาก อาจเป็นเหตุการณ์ที่มีการแก้ไขในโมดูลหนึ่งจะส่งผลต่อการปฏิบัติงานในโมดูลอื่นๆด้วย ตัวอย่างจากรูปภาพที่ 4.7 แสดงให้เห็นว่าโมดูล B มีความเกี่ยวพันธ์กับโมดูล D การปฏิบัติงานของโมดูล B มีการกระโดดข้ามไปทำงานในโมดูล D ซึ่งตามความเป็นจริงแล้วโมดูล D ต้องอยู่ภายใต้การควบคุมของโมดูล C ไม่ควรเกี่ยวข้องกับโมดูล B

สำหรับรูปภาพที่ 4.8 แสดงถึงการทำงานของโมดูล H ซึ่งมีคำสั่งในการกระโดดข้ามเพื่อไปปฏิบัติการในโมดูล G การออกแบบในลักษณะนี้ถือว่าไม่มีคุณภาพอย่างยิ่ง เพราะเมื่อเกิดความผิดพลาดขึ้นไม่ทราบว่าความผิดพลาดนี้เกิดที่ตำแหน่งหรือโมดูลใดกันแน่ เมื่อมีการแก้ไขความผิดพลาดในโมดูลหนึ่งจะส่งผลต่อการทำงานของโมดูลอื่นด้วย ทำให้ต้องตรวจสอบและแก้ไขโมดูลที่เกี่ยวข้องทั้งหมด ซึ่งให้มีความยุ่งยากมากในการแก้ไข

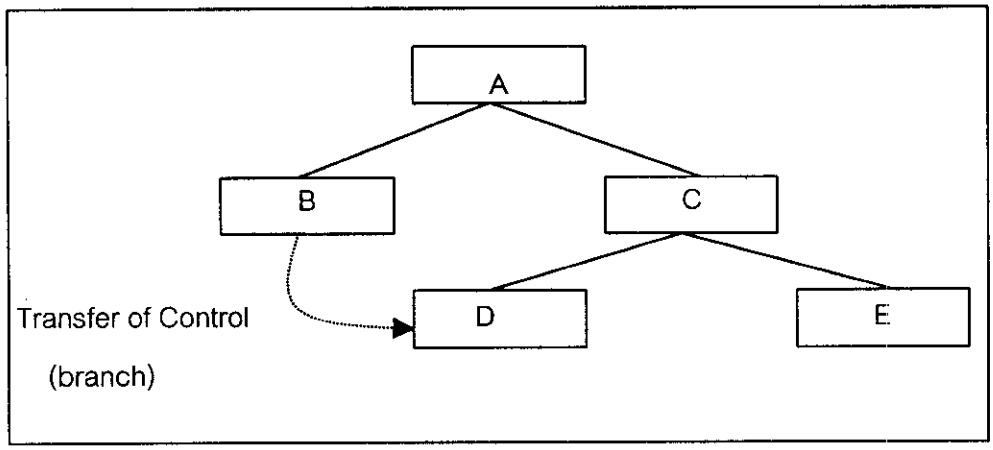
รูปภาพที่ 4.5 Independent vs. Dependent Modules



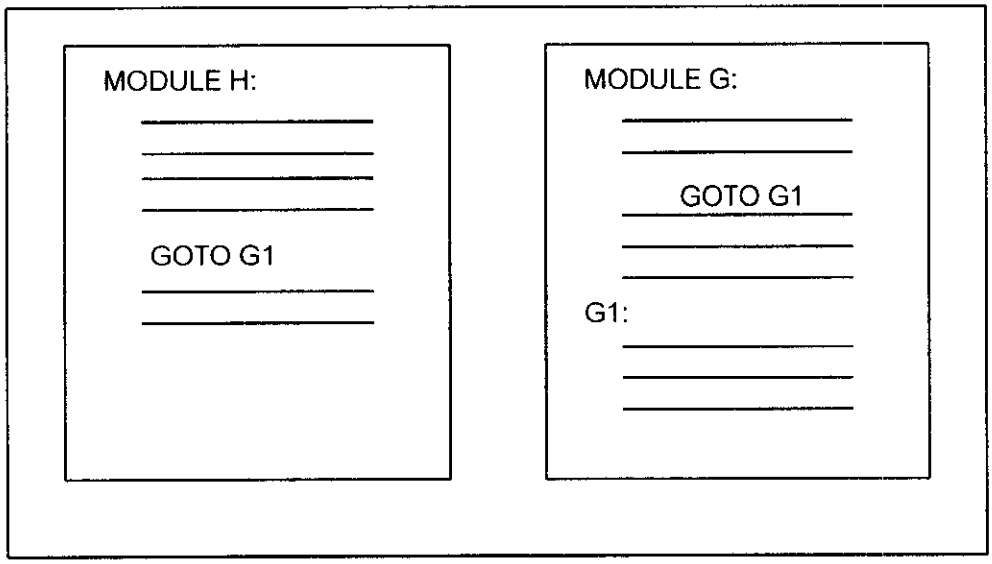
รูปภาพที่ 4.6 Types of Coupling



รูปภาพที่ 4.7 Content Coupling



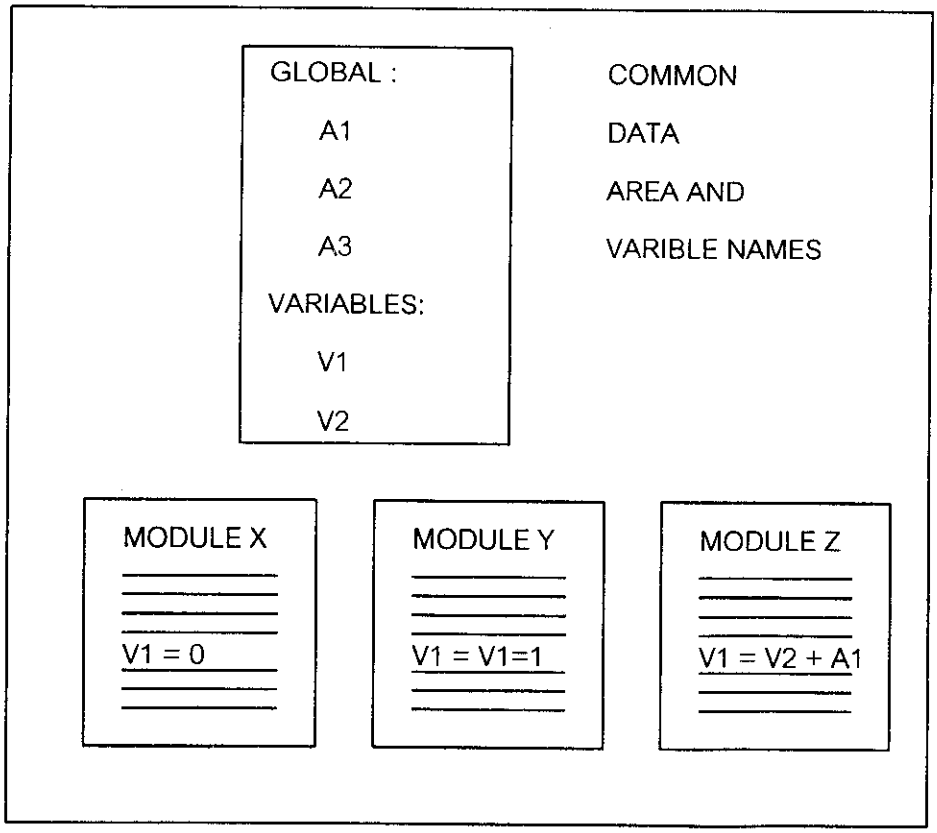
รูปภาพที่ 4.8 Content Coupling Involving Flow of Control



Common coupling เราสามารถออกแบบเพื่อลดความเกี่ยวพันธ์กันระหว่างโมดูลลงได้ โดยนำข้อมูลที่มีการใช้งานร่วมกันไปไว้ในส่วนของ global หรือ common data area ดังรูปภาพที่ 4.9 ซึ่งความยุ่งยากยังคงมีอยู่ เพราะเมื่อมีการเปลี่ยนแปลงค่าของข้อมูลที่เก็บในส่วนนี้ในโมดูลหนึ่งย่อมมีผลกระทบต่อโมดูลอื่นๆที่มีการใช้งานด้วย เมื่อเกิดปัญหาหรือความผิดพลาดเราต้องทำการติดตามค่าของข้อมูลนี้ในโมดูลต่างๆที่มีการปฏิบัติการกับข้อมูลนี้ ซึ่งการออกแบบที่

ดีควรหลีกเลี่ยงการออกแบบในลักษณะนี้ การหลีกเลี่ยงในปัจจุบันใช้วิธีส่งผ่านค่า(pass by value) แทนตัวแปรที่เป็น global

รูปภาพที่ 4.9 Common Coupling



Control coupling เป็นความเกี่ยวพันธ์ที่โมดูลหนึ่งส่งสัญญาณเพื่อไปควบคุมกิจกรรมในโมดูลอื่นๆ ซึ่งเป็นสิ่งที่จำเป็นและเป็นการยากที่จะไม่ให้เกิดแต่ควรออกแบบให้ปริมาณของการควบคุมน้อยที่สุดเท่าที่จะเป็นไปได้

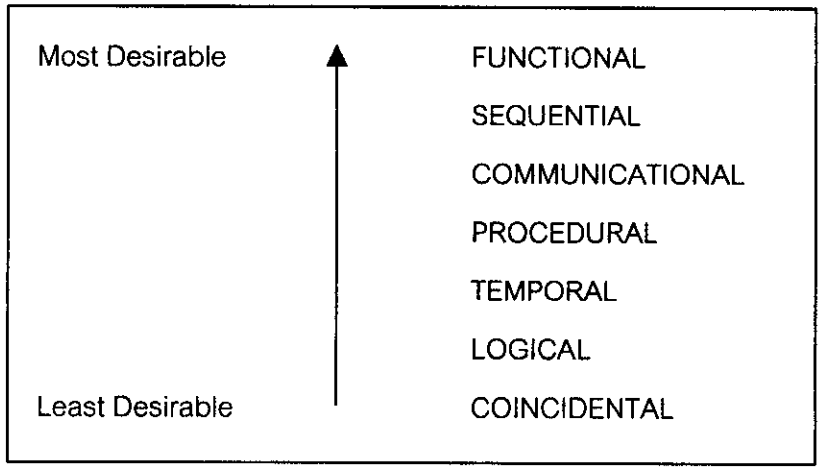
Stamp coupling เป็นความเกี่ยวพันธ์ระหว่างโมดูลเมื่อโมดูลหนึ่งมีการส่งผ่านโครงสร้างข้อมูลไปให้แก่โมดูลอื่น อาจเป็นกลุ่มของสัญญาณควบคุม หรือโครงสร้างข้อมูลก็ได้

Data coupling เป็นการเกี่ยวพันธ์ที่โมดูลหนึ่งส่งผ่านเพียงข้อมูล หรือรูปแบบพื้นฐานซึ่งถือว่ามีคุณสมบัติ Coupling ที่ดีที่สุดเพราะเกิดข้อผิดพลาดได้น้อยมากและถ้ามีความผิดพลาดเกิดขึ้น สามารถติดตามได้โดยง่าย

4. Cohesion

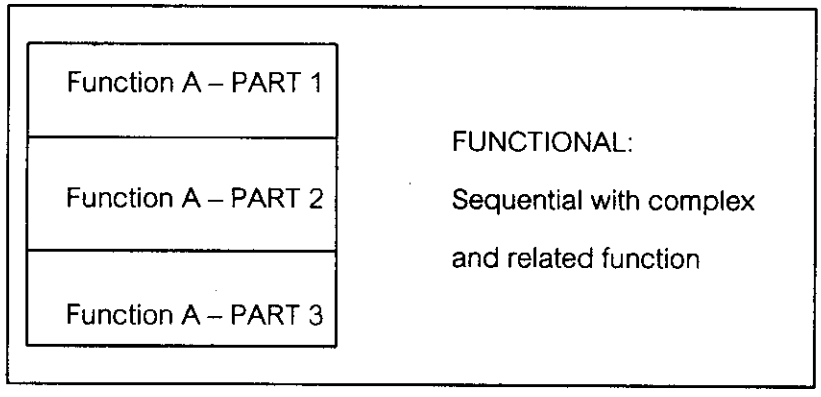
เป็นการวัดความเกี่ยวพันธ์กันภายในโมดูล การออกแบบที่ดีนั้นต้องมีการออกแบบให้ภายในโมดูลมีความเกี่ยวพันธ์ของคำสั่งหรือการประมวลผลต่างๆเป็นไปเพื่อวัตถุประสงค์เดียวกัน การวัดระดับความเกี่ยวพันธ์ของโมดูลนั้นเราแบ่งเป็นหลายชนิด ดังรูปภาพที่ 4.10 โดยการออกแบบที่มีความเกี่ยวพันธ์ที่ดีที่สุดคือ Functional cohesion

รูปภาพที่ 4.10 Types of Cohesion



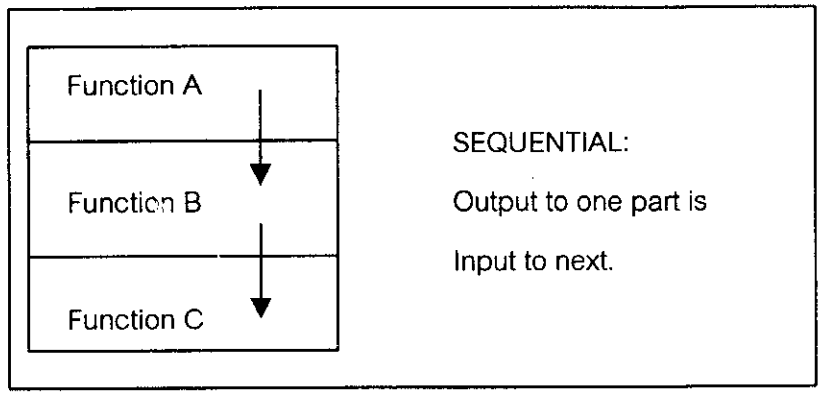
Functional cohesion เป็นการออกแบบที่เราไม่สามารถแบ่งกลุ่มของคำสั่งภายใน Moduleนั้นให้แตกออกไปเป็นโมดูลที่ทำหน้าที่ใดๆได้อีก หน้าที่ต่างๆภายในโมดูลมีความเกี่ยวพันธ์กันมาก หรืออาจจะกล่าวอีกนัยหนึ่งว่า Moduleที่ออกแบบสามารถกระทำกิจกรรมได้เพียงอย่างเดียวเท่านั้น ดังตัวอย่างในรูปภาพที่ 4.11

รูปภาพ 4.11 Functional cohesion



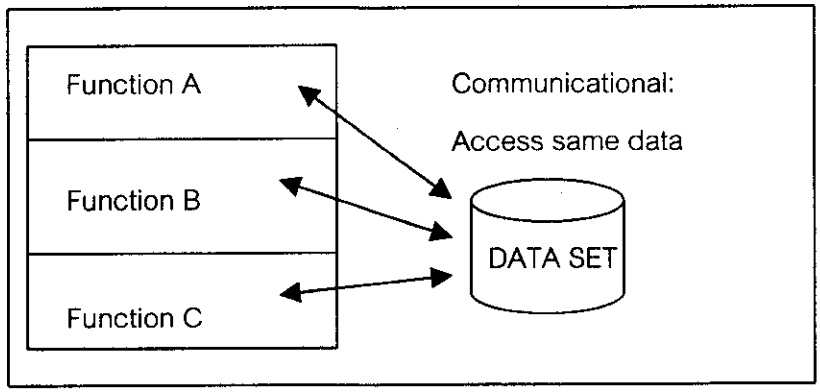
Sequential cohesion เป็นความเกี่ยวพันธ์กันภายในโมดูลที่มีหลายฟังก์ชัน และผลลัพธ์ของฟังก์ชันหนึ่งจะเป็นข้อมูลเข้าของอีกฟังก์ชันหนึ่ง เป็นลำดับของการทำงาน ซึ่งการทำงานอยู่ภายใต้วัตถุประสงค์เดียวกัน ตัวอย่างเช่นถ้าเรามี Module หนึ่ง ซึ่งทำหน้าที่รับข้อมูลและยังทำหน้าที่ในการตรวจสอบความถูกต้องของข้อมูลได้อีกด้วยแสดงว่า Module นี้ยังไม่ใช่ลักษณะของ Functional แต่ Module ดังกล่าวสามารถแยกกิจกรรมได้สองอย่างคือ get data และ check invalid data ดังนั้นเราจะเรียก Module ดังกล่าวว่า sequential รูปภาพที่ 4.12 แสดงตัวอย่าง Sequential cohesion

รูปภาพ 4.12 Sequential cohesion



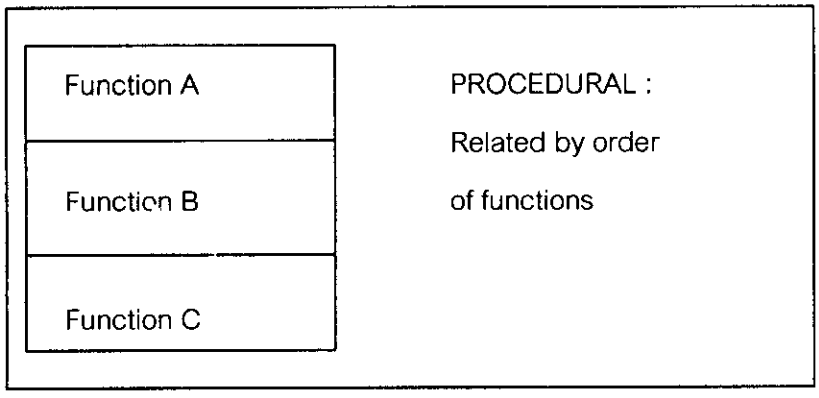
Communication cohesion เป็นความเกี่ยวพันธ์ที่ Module ประกอบด้วยหลายฟังก์ชันที่ทำหน้าที่แยกจากกัน แต่ใช้โครงสร้างของข้อมูลเดียวกัน เช่น โมดูลที่ประกอบด้วยฟังก์ชันการจัดเก็บข้อมูล , ฟังก์ชันการค้นหาข้อมูล , ฟังก์ชันการลบข้อมูลซึ่งใช้กลุ่มของข้อมูลชุดเดียวกัน เป็นต้น รูปภาพที่ 4.13 แสดง Communication cohesion

รูปภาพ 4.13 Communication cohesion



Procedural cohesion มักจะพบในกรณีของการแบ่งโมดูลขนาดใหญ่ให้เป็นฟังก์ชันย่อยไปตามสายการควบคุมโดยไม่คำนึงถึงกิจกรรมที่ทำ เจตนาเพื่อลดภาระการดูแลโมดูลขนาดใหญ่ให้เล็กลงเพื่อความสะดวกในการตรวจสอบโปรแกรม ซึ่งกิจกรรมที่กระทำเป็นไปตามลำดับขั้นตอน การออกแบบในลักษณะนี้แต่ละ Moduleยังคงอยู่ในลักษณะที่ยังมีความสัมพันธ์กันอยู่มาก ดังตัวอย่างในรูปภาพที่ 4.14

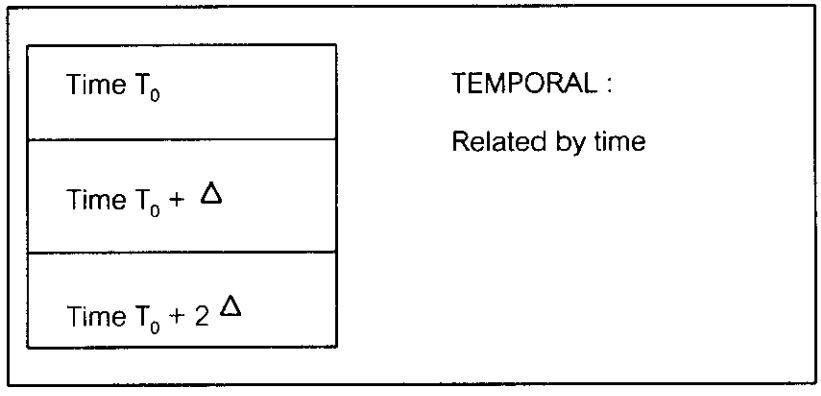
รูปภาพ 4.14 Procedural cohesion



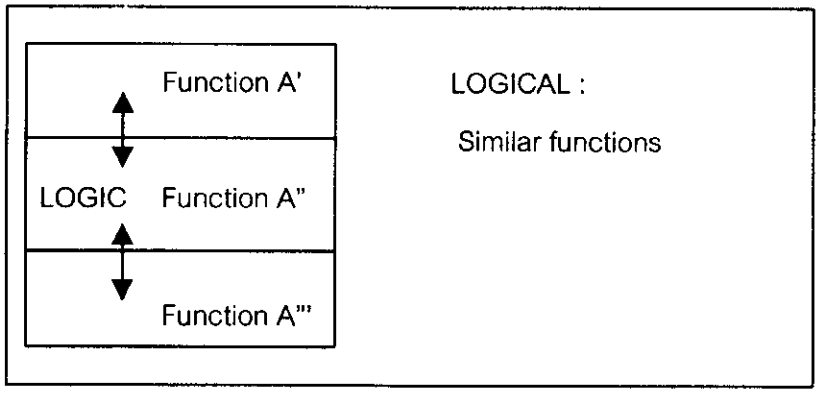
Temporal cohesion เป็นการออกแบบโมดูลที่การทำงานไม่สัมพันธ์กันแต่ต้องประมวลผลในเวลาเดียวกัน เช่นโมดูลในการกำหนดค่าเริ่มต้นของระบบหรือตัวแปรต่างๆ ซึ่งในเวลาหนึ่งต้องมีการกระทำในฟังก์ชันต่างๆพร้อมๆกัน ดังตัวอย่างตามรูปภาพที่ 4.15

Logical cohesion เป็นการออกแบบโมดูลซึ่งไม่เป็นที่ยอมรับ เนื่องจากฟังก์ชันต่างๆภายในโมดูลมีตรรกเหมือนกันหรือคล้ายกัน เช่นโมดูลการอ่านข้อมูลซึ่งประกอบด้วยฟังก์ชั่นในการอ่านข้อมูลจากเทปแม่เหล็ก,ฟังก์ชันการอ่านข้อมูลจากดิสก์,ฟังก์ชันการอ่านข้อมูลจากเครื่องอ่านบาร์โค๊ด เป็นต้น ดังตัวอย่างรูปภาพที่ 4.16

รูปภาพ 4.15 Temporal cohesion



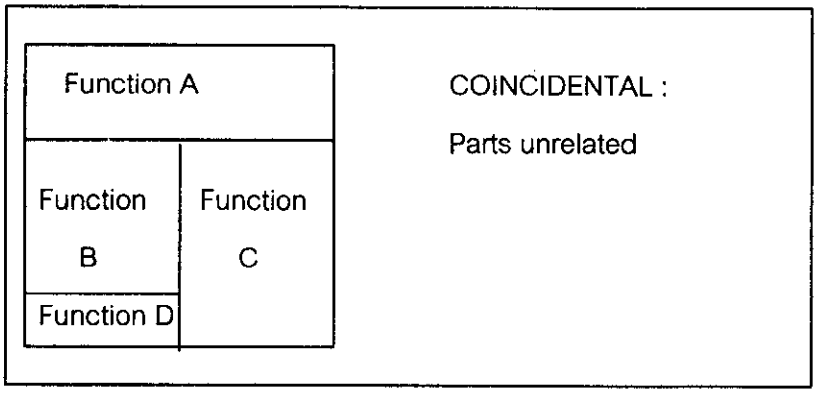
รูปภาพ 4.16 Logical cohesion



การออกแบบในลักษณะของ Temporal และ logical นั้นเป็นลักษณะที่ไม่ดีเพราะยากต่อการเปลี่ยนแปลงและแก้ไข ถ้ามีการเปลี่ยนแปลงในฟังก์ชันหนึ่งต้องกระทำในฟังก์ชันอื่นๆที่เหลือด้วย ต้องค้นหาฟังก์ชันที่มีตรรกคล้ายกันหรือเกี่ยวพันธ์กัน

Coincidental cohesion เป็นการออกแบบที่แย่ที่สุดเนื่องจากโมดูลประกอบด้วยฟังก์ชันที่ไม่มีความเกี่ยวพันธ์กันเลย เช่นภายในโมดูลหนึ่งมีฟังก์ชั่นตรวจสอบสิทธิของผู้ใช้งาน, ฟังก์ชันการพิมพ์สลิปเงินเดือน อยู่ภายในโมดูลเดียวกัน ดังรูปภาพที่ 4.17

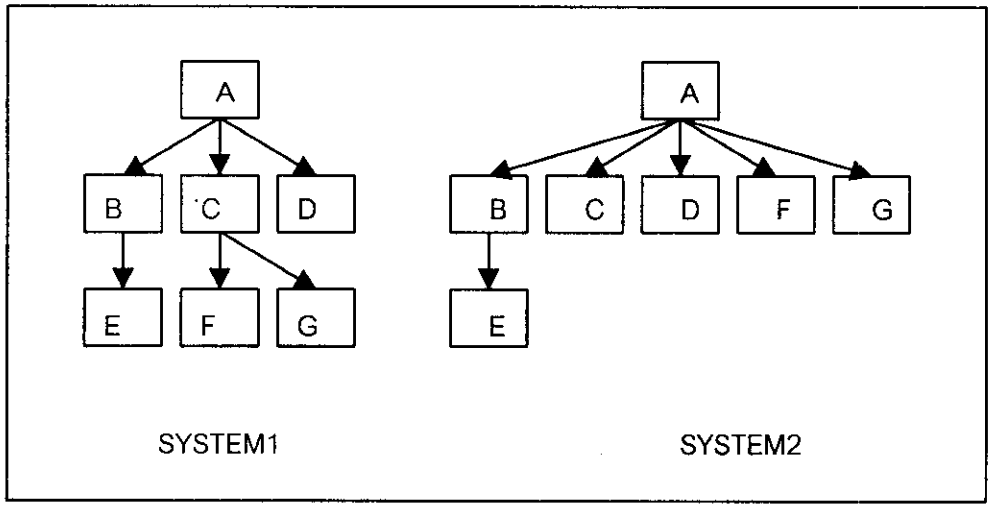
รูปภาพ 4.17 Coincidental cohesion



5. Control Issues

คุณภาพของการออกแบบอีกประการหนึ่งคือวัดจากการควบคุมระหว่างโมดูล ระบบใดก็ตามที่มีการออกแบบให้มีการควบคุมระหว่างโมดูลมีจำนวนน้อยกว่าจะมีคุณภาพที่ดีกว่า

รูปภาพที่ 4.18 ตัวอย่างของ Fan-In และ Fan-Out



จากตัวอย่างในรูปภาพที่ 4.18 เป็นระบบที่มีการออกแบบโมดูลเหมือนกันมีจำนวนเท่ากัน แต่แตกต่างกันที่จำนวนโมดูลในระดับต่างๆไม่เท่ากัน เราสามารถวัดคุณภาพของการออกแบบนี้จาก Fan-In และ Fan-Out ของระบบ จะเห็นได้ว่าทุกโมดูลของระบบทั้งสองมี Fan-In เท่ากันหมดคือมีลูกศรที่ชี้มายังทุกโมดูลมีจำนวนเท่ากับ 1 เส้น แต่มีลูกศรออกจากโมดูลไม่เท่ากัน ในที่นี้โมดูล A ของระบบ1 มี Fan-Out เท่ากับ 3 แต่ของระบบ2 มี Fan-Out เท่ากับ 5 การวัดนั้นระบบที่ดีต้องมี Fan-In และ Fan-Out มีค่าที่น้อย เพราะระบบที่มีการควบคุมระหว่างโมดูลน้อยกว่าจะมีความซับซ้อนน้อยกว่า ส่งผลให้การแก้ไขปรับเปลี่ยนหรือตรวจสอบน้อยกว่าตามไปด้วย

6. Scope of Control and Effect

การออกแบบที่ดีนั้นต้องให้แน่ใจว่าการดำเนินงานในโมดูลหนึ่ง ผลการทำงานใดจะต้องไม่ก่อให้เกิดผลกระทบข้างเคียงให้กับโมดูล อื่นในระบบงานเดียวกัน แต่การออกแบบระบบต้องมีการเรียกใช้โมดูลต่างๆอย่างหลีกเลี่ยงไม่ได้ จากรูปภาพที่ 4.18 พิจารณาระบบ1 ถ้ามีการแก้ไข

โมดูล C จะเห็นว่าโมดูลที่ส่งผลกระทบคือโมดูล F และ G การออกแบบที่ดีต้องควบคุมให้ขอบเขตของการแก้ไขนั้นส่งผลเฉพาะสองโมดูลที่เกี่ยวพันธ์จริงๆเท่านั้น

4.4

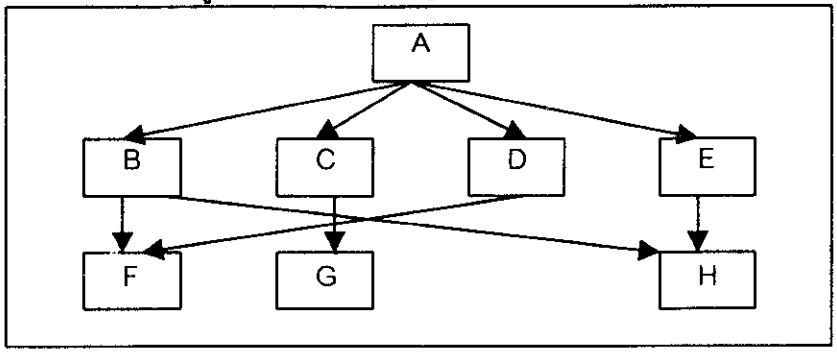
กลยุทธ์ในการออกแบบระบบ

จากการอธิบายในหัวข้อที่แล้ว สามารถสรุปคุณลักษณะของการออกแบบที่ดีได้ดังนี้

1. Low coupling มีความเกี่ยวพันธ์ระหว่างโมดูลต่ำ
2. Highly cohesive มีความเกี่ยวพันธ์ของฟังก์ชันต่างๆภายในโมดูลสูง
3. Minimal number of modules with fan-out มีจำนวนของโมดูลที่ถูกควบคุมจำนวนน้อย
4. Scope of effect of a module limited to its scope of control ออกแบบให้ขอบเขตของผลกระทบต่อโมดูลนั้นสามารถควบคุมได้ในขอบเขตที่จำกัด

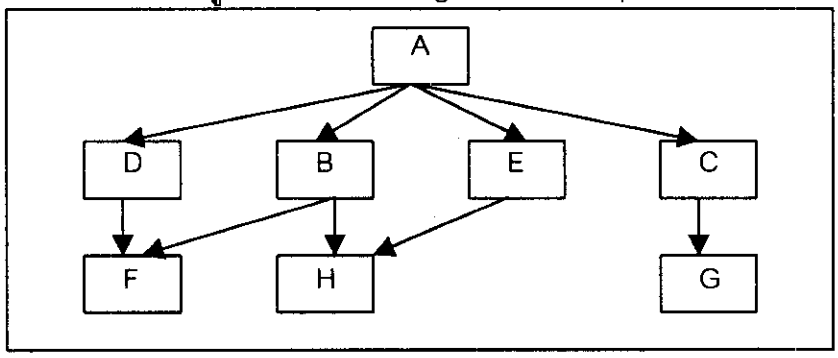
ซึ่งถ้านักออกแบบระบบสามารถกระทำได้ตามคุณลักษณะที่กล่าวมานี้จะเป็นผลดีอย่างมากในการสร้างระบบ ทำให้ง่ายต่อการทดสอบ, ง่ายต่อการแก้ไขให้ถูกต้อง และง่ายต่อการบำรุงรักษาระบบ อันดับแรกในการออกแบบนั้นนักออกแบบระบบต้องทำความเข้าใจกับสิ่งที่ลูกค้าต้องการ นำมาวิเคราะห์ถึงปัญหา และหาหนทางในการแก้ปัญหาการเริ่มต้นครั้งแรกอาจจะยังไม่ค่อยดีนัก การแบ่งโมดูลต่างๆจะยุ่งเหยิง สับสน และทำความเข้าใจยาก

รูปภาพที่ 4.19 Diagram before Simplification



จากรูปภาพที่ 4.19 แสดงไดอะแกรมของระบบที่มีการโยงลูกศรข้ามไปมาซึ่งทำให้ทำความเข้าใจยาก และเกิดความสับสน เราต้องพยายามออกแบบใหม่ปรับเปลี่ยนโครงสร้างให้เข้าใจง่ายขึ้น โดยลดโครงสร้างที่ซับซ้อน ดังเช่นรูปภาพที่ 4.20

รูปภาพที่ 4.20 Diagram After Simplification



ในกรณีที่ระบุความต้องการอยู่ในรูปของตารางตัดสินใจ ในการออกแบบต้องปรับเปลี่ยนเงื่อนไขการทำงานให้อยู่ในรูปแบบที่ง่ายขึ้น ตารางที่ 4.1 แสดงตารางตัดสินใจของกิจกรรมของมหาวิทยาลัยแห่งหนึ่ง ซึ่งมีตัวแปรในการทำกิจกรรมทั้งหมด 4 ตัวแปรคือ W, X ,Y,Z ค่าของตัวแปรต่างๆมีค่าเท่ากับ 0 หรือ 1 ค่าใดค่าหนึ่ง ส่งผลให้เกิดเงื่อนไขแตกต่างกันถึง 16 เงื่อนไข แต่ละเงื่อนไขมีกิจกรรมที่กระทำแตกต่างกัน

นักออกแบบระบบต้องทำการพิจารณาและพยายามที่จะลดความซับซ้อนให้น้อยลง.ให้อยู่ในรูปแบบที่ง่ายต่อการทำความเข้าใจ ซึ่งจากตัวอย่างนี้สามารถประยุกต์ได้โดยแทนความสัมพันธ์ระหว่างตัวแปรกับผลลัพธ์ของการปฏิบัติเป็นสมการ Algebraic ดังนี้

A1 = WXYZ' + WXYZ

A2 = WXY'Z' + W'XY'Z + W'XYZ + WX'Y'Z' + WX'Y'Z + WX'Y'Z'
+ WX'YZ' + WX'YZ + WXY'Z' + WXY'Z

A3 = WXY'Z' + WXY'Z + WXYZ' + WXYZ

A4 = W''X'YZ' + W'X'YZ + W'XY'Z' + W'XY'Z + W'XYZ' + W'XYZ
+ WX'Y'Z' + WX'Y'Z + WX'YZ' + WX'YZ

A5 = W'X'YZ' + W'X'YZ

A6 = W'X'Y'Z' + W'X'Y'Z

ตารางที่ 4.1 ตารางการตัดสินใจ

Variables				Actions					
W	X	Y	Z	A1	A2	A3	A4	A5	A6
0	0	0	0						X
0	0	0	1						X
0	0	1	0				X	X	
0	0	1	1				X	X	
0	1	0	0		X		X		
0	1	0	1		X		X		
0	1	1	0		X		X		
0	1	1	1			X			
1	0	0	0		X		X		
1	0	0	1		X		X		
1	0	1	0		X		X		
1	0	1	1		X		X		
1	1	0	0		X	X			
1	1	0	1		X	X			
1	1	1	0	X		X			
1	1	1	1	X		X			

สมการที่ได้เกิดจากการแทนค่าจากเงื่อนไขของตัวแปร เช่น A6 = W'X'Y'Z' + W'X'Y'Z นั้นมาจากค่าของตัวแปรของ W=0,X=0,Y=0,Z=0 หรือ W=0,X=0,Y=0,Z=1 เราแทน "หรือ" ด้วยเครื่องหมายบวก(+) ตัวแปรที่มีค่าเป็น 0 จะเขียนเป็น Implement ของตัวแปรนั้น ต่อจากนั้นใช้กฎเกณฑ์ทางพีชคณิตบูลีนเพื่อให้มีความซับซ้อนน้อยลง เช่น

A1 = WXYZ' + WXYZ

= WXY(Z'+Z)

= WXY(1)

= WXY

พิจารณาสมการที่เหลือโดยพยายามให้อยู่ในรูปแบบที่ง่ายขึ้น ผลของตารางตัดสินใจเมื่อปรับเปลี่ยนเป็นสมการอย่างง่ายได้ดังตารางที่ 4.2

ตารางที่ 4.2 Simplified Formulae for the Decision Table

A1 = XYZ
A2 = XY' + W'XY + WX'
A3 = WX
A4 = X'Y + W'X + WX'Y'
A5 = W'X'Y
A6 = W'X'Y'

สำหรับวิธีการนี้สามารถประยุกต์ใช้สำหรับกำหนดความต้องการทางตรรกฮาร์ดแวร์ ซึ่งช่วยลดจำนวนเกต(gate) ให้น้อยลงช่วยลดค่าใช้จ่ายฮาร์ดแวร์ อีกทั้งสามารถตรวจสอบความถูกต้องได้อย่างรวดเร็วและส่งผลให้วงจรมีราคาถูกอีกด้วย

4.5

เทคนิคและเครื่องมือในการออกแบบระบบ

ในการออกแบบระบบนั้นมีเทคนิคและเครื่องมือมากมายหลายชนิด ความเหมาะสมของการนำไปใช้นั้นขึ้นอยู่กับคุณลักษณะของระบบด้วยว่าระบบต้องการในเรื่องใดเป็นพิเศษ กรณีที่ต้องการให้ระบบกระทำหน้าที่ต่างๆตามที่ต้องการ การออกแบบก็จะเน้นหนักไปทางด้านการปฏิบัติการหรือการประมวลผล ถ้าความต้องการเน้นหนักไปในเรื่องของข้อมูลจะออกแบบเน้นไปทางด้านความสัมพันธ์ระหว่างข้อมูลมากกว่าการปฏิบัติการ อาทิเช่นออกแบบระบบควบคุมเครื่องจักรใน

โรงงานอุตสาหกรรม เราถือว่าเป็นระบบที่เป็น function-strong แต่ถ้าเป็นการออกแบบระบบเงินเดือน จะเป็น data-strong สำหรับระบบที่อยู่ระหว่างกลางของความต้องการทั้งสองเราเรียกว่า hybrid system

ในการออกแบบระบบตามความเป็นจริงเราสามารถเลือกเครื่องมือมาช่วยในการออกแบบโดยรวมหลายๆวิธีเข้าด้วยกัน เพื่อผู้พัฒนาเข้าใจและสามารถนำไปสร้างระบบได้ สำหรับเครื่องมือที่นิยมใช้ในการออกแบบระบบที่นิยมกันมีด้วยกันหลายเชนิดได้แก่

DATA FLOW DIAGRAM

เป็นการออกแบบโดยเริ่มจากพิจารณาเอกสารระบุความต้องการในส่วนของการกำหนดข้อมูล ซึ่งมีการระบุถึงข้อมูลต่างๆที่ใช้ในระบบจากพจนานุกรมข้อมูล เมื่อวิเคราะห์ถึงข้อมูลที่ต้องการเราอาจสร้างกลุ่มของข้อมูลใหม่โดยพิจารณาจากกฎเกณฑ์ที่กำหนด เช่น A, T, X และ F เป็นข้อมูลที่กำหนดในพจนานุกรมข้อมูล เราอาจนำมาเขียนเป็นความสัมพันธ์ได้ดังนี้

C = A

หมายความว่าค่าของข้อมูล C และข้อมูล A มีค่าสมมูลกัน(equivalent)

C = A * T โดย เครื่องหมาย * หมายถึง และ(and)

ความหมายคือข้อมูล C เกิดจากการนำข้อมูล A รวมกับข้อมูล T

B = [A | T | F] โดยเครื่องหมาย | หมายถึงหรือ(or)

ความหมายคือข้อมูล B มีค่าเท่ากับข้อมูล A หรือ ข้อมูล T หรือข้อมูล F

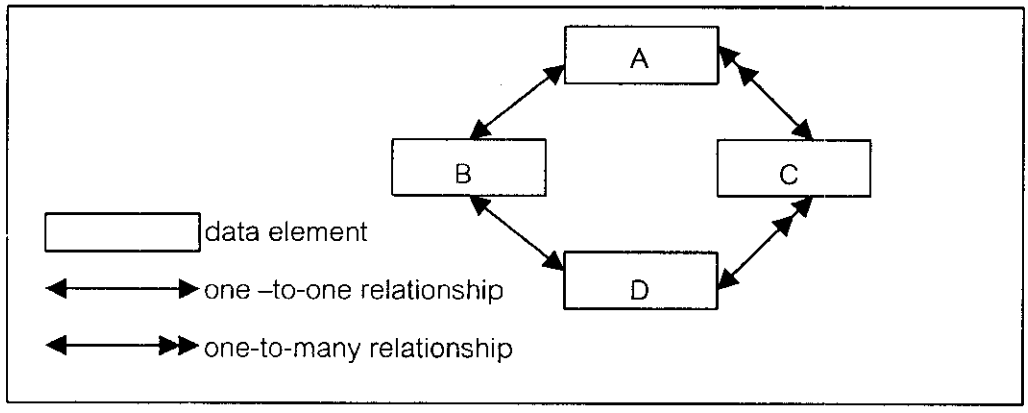
A = 0 { X } 16 โดยเครื่องหมาย { } หมายถึงการทำซ้ำของข้อมูล

ความหมายคือ ข้อมูล A จะมีค่าอยู่ระหว่างศูนย์และค่าสิบหก

C = A(X) โดยเครื่องหมาย () แสดงถึงการเลือกบางสิ่งบางอย่าง

Data dictionary นั้นเพียงแต่มีการแสดงถึงข้อมูลต่างๆที่ใช้ในระบบแต่ไม่ได้แสดงถึงความสัมพันธ์ระหว่างข้อมูล Bachman ได้เสนอแนวความคิดในการแสดงความสัมพันธ์ของข้อมูลเป็นไดอะแกรม โดยแทนรูปสี่เหลี่ยมด้วยข้อมูล ใช้ลูกศรในการเชื่อมโยงสี่เหลี่ยมต่างๆ ดังตัวอย่างในรูปภาพที่ 4.20

รูปภาพที่ 4.20 Data Structure Diagram

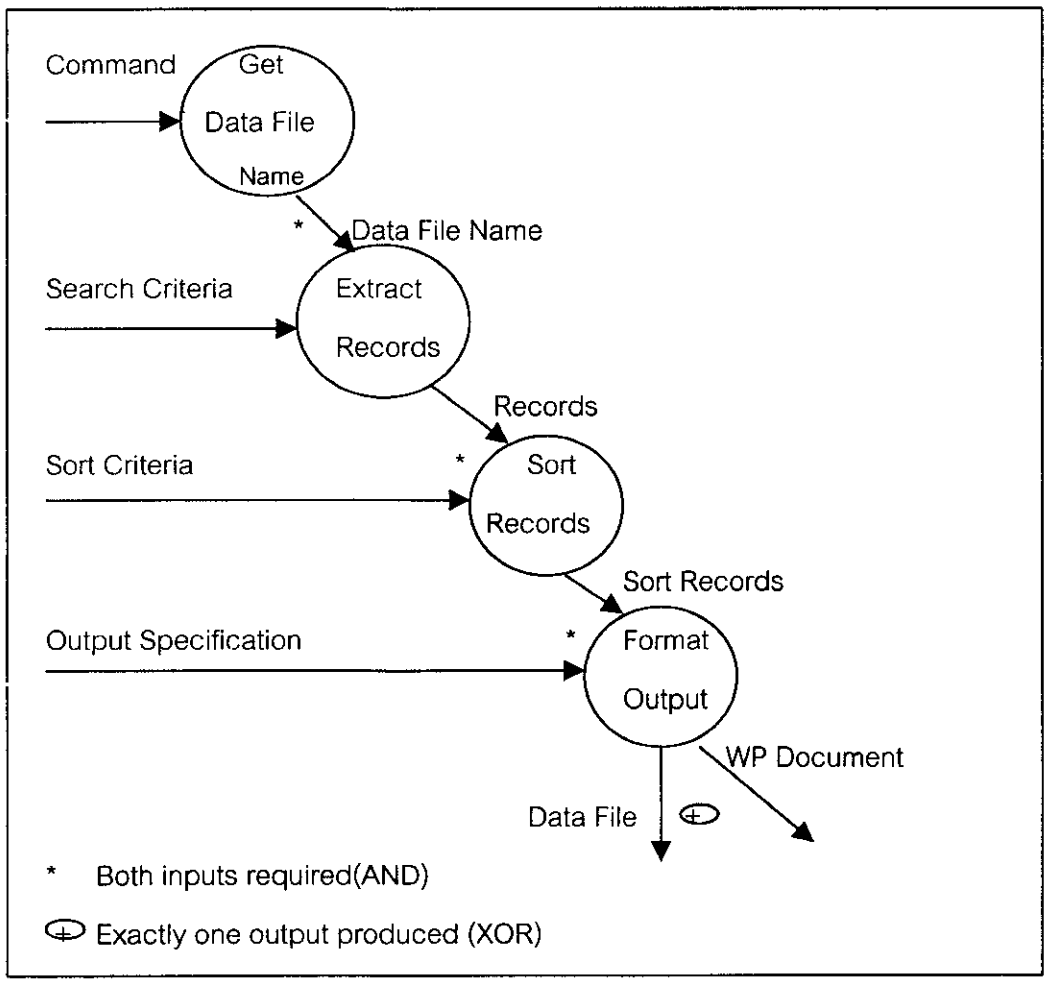


โดยความหมายของลูกศรที่มีปลายหัวเดียวหมายถึง one ถ้าปลายด้านใดมีลูกศรสองหัวเราจะเรียกความสัมพันธ์ว่าเป็น many ต่อมา DeMarco ได้มีใช้ Data Flow Diagram เพื่อใช้สำหรับอธิบายความสัมพันธ์ของข้อมูลต่างๆในระบบ ซึ่งไดอะแกรมนี้ประกอบด้วยลูกศรซึ่งแสดงการไหลของข้อมูล วงกลมแทนถึงการประมวลผลกับข้อมูลซึ่งเป็นคำกิริยาแสดงถึงการกระทำต่อข้อมูลที่ได้รับ และตัวกระทำ(operator) โดยวงกลมหรือการประมวลผลนี้ถือว่าเป็นศูนย์กลางของการเปลี่ยนแปลงของข้อมูล

รูปภาพที่ 4.21 แสดง Data Flow Diagram ของระบบการสอบถามข้อมูล ซึ่งเริ่มจากการป้อนแฟ้มข้อมูลที่ต้องการหา เมื่อได้ชื่อแฟ้มแล้วจะทำการค้นหากลุ่มของข้อมูลที่ต้องการ นำมาเรียงลำดับ จัดรูปแบบผลลัพธ์อาจเก็บในแฟ้มข้อมูลหรือพิมพ์ออกมาอย่างใดอย่างหนึ่ง ในที่นี้ตัวกระทำ * หมายถึง and โดยกิจกรรมจะถูกกระทำต้อง ได้รับข้อมูลครบทั้งสองทางจากรูปมีลูกศรชี้เข้าสู่โปรเซสสองทางด้วยกัน สำหรับตัวกระทำเครื่องหมายบวกมีวงกลมล้อมรอบหมายถึง XOR จากรูปผลของการประมวลผลเกิดผลลัพธ์เพียงผลลัพธ์เดียวทางใดทางหนึ่งเท่านั้น

Jackson ได้มีการพัฒนาเทคนิคในการออกแบบระบบ โดยกำหนดสถานะของข้อมูลเริ่มต้นและความสัมพันธ์ระหว่างกัน โดยใช้ไดอะแกรมต้นไม้บวกกับ regular grammar เพื่อบรรยายข้อมูลที่สร้างขึ้น ซึ่งแบบจำลองนี้เราใช้สำหรับกำหนดการปฏิบัติการที่จำเป็นสำหรับการแก้ปัญหา ดังตัวอย่างในรูปภาพที่ 4.22

รูปภาพที่ 4.21 Data Flow Diagram

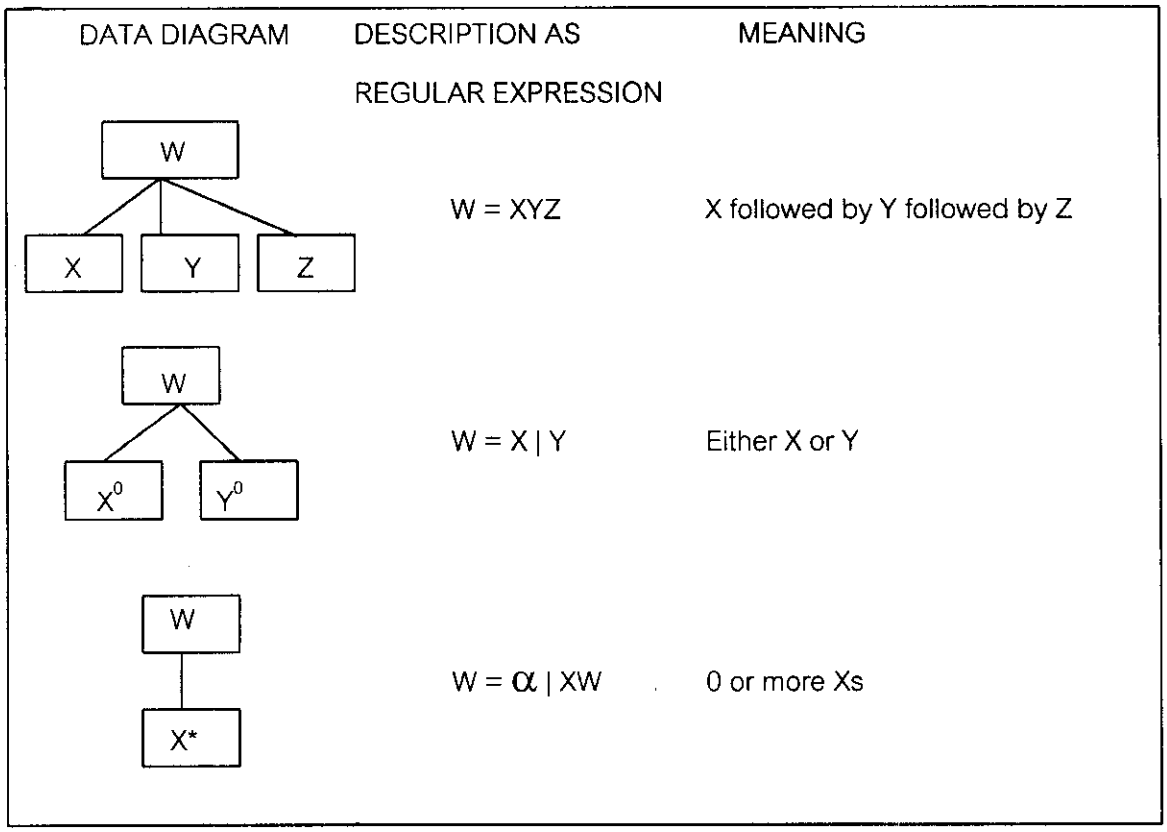


Data Abstraction

ถ้าเราใช้นามธรรมข้อมูลเป็นเครื่องมือในการระบุรายละเอียดความต้องการ เราสามารถใช้นามธรรรมข้อมูลนี้เพื่อออกแบบระบบได้ โดยเริ่มจากพิจารณาแนวความคิดของนามธรรมต่างๆ เหล่านั้น เช่น ความต้องการของระบบต้องการให้เตรียมกลุ่มของข้อมูลให้มีการเรียงลำดับจากค่าน้อยไปยังค่าที่มาก โดยมีการระบุความต้องการดังนี้

Rearrange L in nondecreasing order

รูปภาพที่ 4.22 Jackson Data Structure Diagram



จากความต้องการดังกล่าว ผู้ออกแบบต้องทำความเข้าใจกับการปฏิบัติการ Rearrange ต่อข้อมูล L โดยหาวิธีการเพื่อให้ระบบสามารถกระทำได้ตามที่กำหนด ในที่นี้สามารถเขียนเป็นรายละเอียดของการปฏิบัติการได้ดังนี้

DO WHILE I is between 1 and (length of L) –1

Set LOW to next of smallest value in L(I),...,L(length of L)

Interchange L(I) and L(LOW)

ENDDO

อัลกอริทึมที่เขียนขึ้นเป็นแนวความคิดเพื่อบอกกับเราถึงกระบวนการที่ใช้ในการกระทำ rearrange ต่อข้อมูล L ซึ่งยังไม่ละเอียดเพียงพอในการออกแบบ สำหรับอัลกอริทึมที่ละเอียดกว่านี้เป็นดังนี้

DO WHILE I is between 1 and (length of L) –1

```
        Set LOW to current value of I
        DO WHILE J is between I+1 and (length of L) – 1
            IF  L(LOW) is greater than L(J) then set LOW to current value of J
            ENDIF
        ENDDO
        Set TEMP to L(LOW)
        Set L(LOW) to L(I)
        Set L(I) to TEMP
    ENDDO
```

จากตัวอย่างข้างต้นนี้มีการกำหนดนามธรรมเป็น 3 ระดับโดยในระดับแรกเป็นกำหนดสิ่งที่ระบบต้องการ ในระดับที่สอง จะให้รายละเอียดมากขึ้นทำให้เราเห็นภาพรวมของกระบวนการปฏิบัติในการเตรียมข้อมูล L ได้มากขึ้น สำหรับการนิยามในระดับที่สามนั้นเป็นขั้นตอนการปฏิบัติการโดยละเอียด

สำหรับระบบที่มีหลายกิจกรรมที่กระทำบนกลุ่มข้อมูลเดียวกันนั้น การออกแบบต้องสร้างออปเจ็คโมดูล(object module) เพื่อบรรยายถึงสถานะของข้อมูล(state) ,การปฏิบัติการ(operation) ต่อข้อมูลเป็นการเปลี่ยนจากสถานะปัจจุบันเป็นสถานะใหม่ และ probes เป็นสารสนเทศที่ไม่เกี่ยวกับการเปลี่ยนสถานะ เพื่อช่วยในการปกปิดความลับของข้อมูล(data encapsulation) ได้อีกด้วย ตัวอย่าง สแตก(stack) เราสามารถกำหนดนามธรรมของข้อมูลเป็นออปเจ็กโมดูลเพื่อบรรยายถึงการปฏิบัติการต่างๆที่เกี่ยวข้องกับโครงสร้างข้อมูลนี้ซึ่งทำให้เราเข้าใจและตรวจสอบได้ง่าย ตัวอย่างต่อไปนี้เป็นการสร้างออปเจ็คโมดูล

```
Discussion: Contains a last-in, first-out data structure
State Space: A state is a two-way condition.full/not_full, and a sequence i1,
        i2,...ik of items, where k is a nonnegative integer.
Probes:
        Top
                Preconditions: k not equal to 0
                export : item
```

description: export i_k

full

export: boolean

description: export true if and only if full is set

empty

export: boolean

description: export k = 0

Operations:

Initialize

description: create an empty sequence; set not_full

push

preconditions: not_full is set

imports: item , i

description: k <- k+1

i_k <- i

EITHER

set full

OR

set not_full

END

pop

preconditions : not empty

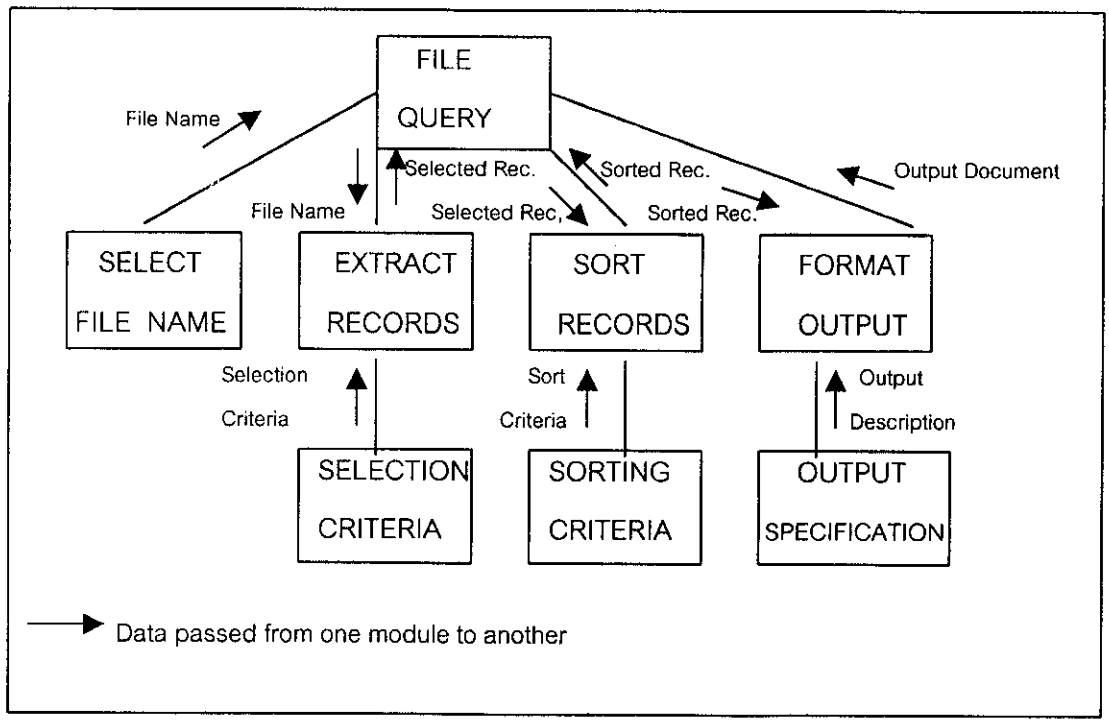
description: k <- k-1; set not_full

การกำหนดข้างต้นเป็นตัวอย่างในการกำหนดออปเจ็กโมดูลของสแตกซึ่งมีการปฏิบัติการกับข้อมูล 2 ลักษณะคือการ pop และ push สำหรับ probes เป็นการกำหนดสารสนเทศอื่นๆ ให้กับโครงสร้างนี้ได้แก่ การตรวจสอบว่าสแตกเต็มหรือสแตกว่าง รวมทั้งตรวจสอบว่าข้อมูลใดอยู่ในตำแหน่งบนสุดของสแตกซึ่งไม่เกี่ยวข้องกับการเปลี่ยนแปลงข้อมูลในสแตก

STRUCTURE CHARTS

สำหรับระบบที่ไม่เกี่ยวพันธ์กับข้อมูลมากนักเราสามารถออกแบบโดยใช้ผังโครงสร้าง(structure charts) เพื่อแสดงลำดับชั้นของโมดูลในระบบ รูปภาพที่ 4.23 เป็นผังโครงสร้าง ซึ่งแสดงถึงการสอบถามข้อมูลมีการส่งผ่านข้อมูลจากโมดูลหนึ่งไปยังอีกโมดูลหนึ่ง โดยใช้สัญลักษณ์ลูกศรแสดงทิศทาง

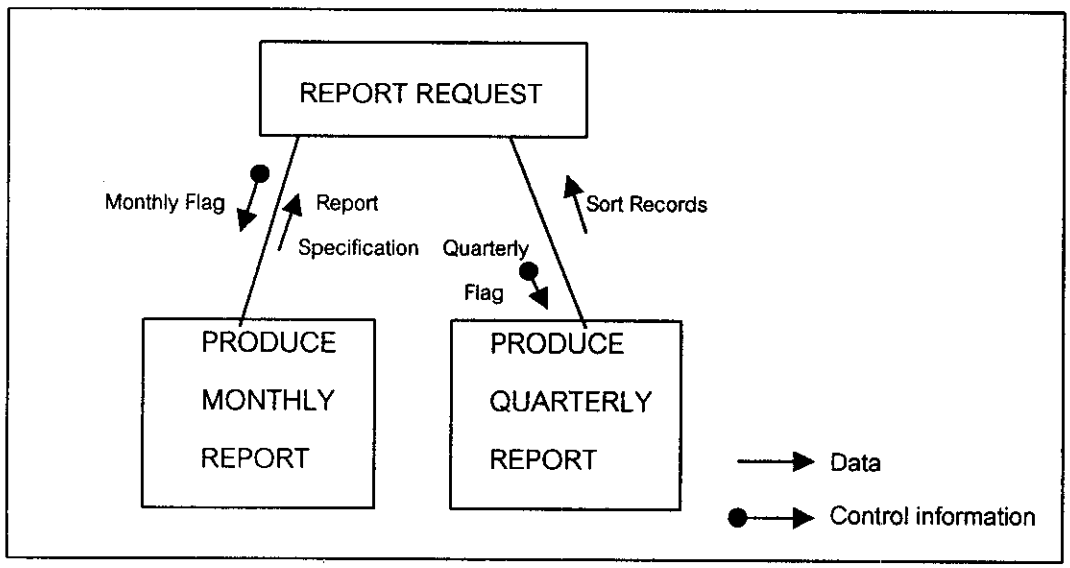
รูปภาพที่ 4.23 Structure Chart for File Quary



ซึ่งการทำงานของผังโครงสร้างนั้นเราถือว่าโมดูลในระดับชั้นที่สูงกว่าจะควบคุมโมดูลในระดับชั้นที่ต่ำกว่า แต่ละโมดูลมีความสัมพันธ์กันก็ต่อเมื่อมีลูกศรซึ่งเป็นการไหลของข้อมูลจากโมดูลหนึ่งไปยังอีกโมดูลหนึ่งกรณีที่ไม่มีลูกศรเลยถือว่าไม่มีความสัมพันธ์กัน การดำเนินกิจกรรมต่างๆจะกระทำจากซ้ายไปขวา ในกรณีบางกิจกรรมกระทำตามเวลาต้องเพิ่มสารสนเทศเพื่อควบคุมเข้าไปในผังโครงสร้างด้วย ตัวอย่างตามรูปภาพที่ 4.24 แสดงส่วนของผังโครงสร้างในการพิมพ์รายงาน โดยมีการระบุสารสนเทศในการควบคุมการทำกิจกรรมในที่นี้ใช้ ลูกศรที่มี

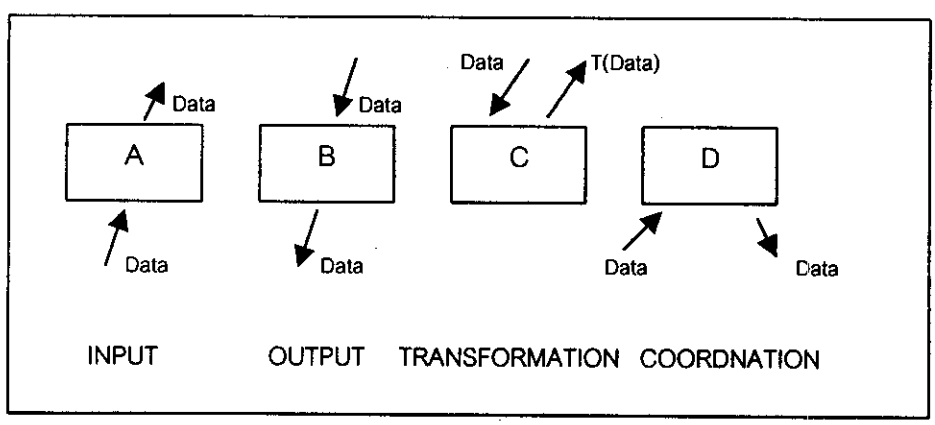
ปลายทีบวงกลมแทนสารสนเทศในการควบคุม สำหรับกิจกรรมที่กระทำถูกระบุด้วยเวลาคือ กระทำทุกๆเดือน และทุกๆสามเดือน

รูปภาพที่ 4.24 Use of Control Flag



ผังโครงสร้างนั้นมีชนิดของข้อมูลที่ไหลผ่านโมดูลแบ่งเป็น 4 ชนิด คือ input , output , transformation และ coordination

รูปภาพที่ 4.27 Four Types of Data Flow



จากรูปภาพที่ 4.27 แสดงชนิดของสารสนเทศที่ไหลผ่านโมดูล กรณีที่เป็น input นั้น ข้อมูลจะส่งเข้าสู่โมดูลในระดับที่ต่ำกว่าและไหลผ่านไปยังโมดูลในระดับสูงกว่าโดยมิได้นำข้อมูลนั้นไปปฏิบัติการแต่อย่างใด ในกรณีของ output ข้อมูลจะไหลเข้าสู่โมดูลในระดับที่สูงกว่าและไหลผ่านไปยังโมดูลในระดับที่ต่ำกว่า สำหรับโมดูล C มีข้อมูลไหลผ่านเข้ามาแต่เกิดเหตุการณ์บางอย่างเกิดขึ้นทำให้ค่าของข้อมูลมีการเปลี่ยนแปลง สำหรับโมดูล D นั้นข้อมูลมาจากระดับที่ต่ำกว่า ซึ่งอาจเป็นข้อมูลควบคุมหรือเตรียมข้อมูลสำหรับโมดูลอื่นในระดับที่ต่ำกว่า

Rapid Prototyping

การกำหนดความต้องการของลูกค้านั้นบางครั้งลูกค้าก็ไม่แน่ใจถึงรายละเอียดต่างๆ หรือไม่ยอมรับในบางเรื่องของระบบเพราะไม่เห็นภาพหรือจินตนาการไม่ได้ ดังนั้นถ้าเรามีการสร้างต้นแบบหรือโครงร่างของระบบให้ลูกค้าได้เห็น ลูกค้าสามารถที่จะมองภาพของระบบใหม่ได้และมีความคิดในเรื่องที่ต้องการที่อยากให้ระบบสามารถกระทำได้ ทำให้สามารถติดต่อกับผู้พัฒนาระบบได้ง่ายขึ้น ดังนั้นเทคนิคของการสร้างต้นแบบจึงถูกนำมาใช้เพื่อสร้างระบบจำลองให้กับลูกค้า โดยต้นแบบนี้ประกอบไปด้วยหน้าจอ รายงาน และเมนูการทำงานต่างๆ ซึ่งยังไม่สามารถกระกิจกรรมต่างๆได้จริง เราถือว่าเป็นโครงร่างของระบบโดยคร่าวๆสร้างขึ้นเพื่อให้ลูกค้าตรวจสอบถึงความต้องการว่าครบถ้วนถูกต้องหรือไม่ เพื่อลดความผิดพลาดและความผิดพลาดก่อนจะนำไปสร้างเป็นระบบจริงและทั้งจะเป็นข้อเสนอแนะจากผู้ใช้งานว่าควรเพิ่มอะไรลงไปอีก

ในการสร้างต้นแบบนี้ผู้พัฒนายังได้ประโยชน์อีกประการหนึ่งคือสามารถประมาณการสิ่งที่ผู้ใช้ต้องการได้ เช่นฮาร์ดแวร์ ,ซอฟต์แวร์, บุคลากร ,สวนสนับสนุน ,การฝึกอบรมการใช้งาน, พัฒนาเอกสารของระบบ ฯลฯ Boehm ,Gray ,Seewaldt ได้ทำการศึกษาถึงโครงการที่ใช้ต้นแบบเป็นเครื่องมือในการพัฒนาระบบ พบว่าโครงการเหล่านี้สามารถประหยัดค่าความพยายาม (effort)ในการพัฒนาถึง 45 เปอร์เซนต์และจำนวนบรรทัดคำสั่งมีขนาดเล็กลงถึง 40 เปอร์เซ็นต์ เมื่อเปรียบเทียบกับการพัฒนาแบบดั้งเดิม จะเห็นได้ว่าเทคนิคที่ใช้ในการพัฒนามีส่วนสำคัญที่ส่งผลให้ระบบมีความเร็วและมีประสิทธิภาพที่ดี

UML(Unified Modeling Language)

เป็นแนวคิดและสร้างระบบงานในลักษณะโลกของความเป็นจริง โดยมองสิ่งต่างๆเป็นวัตถุหรือออบเจ็กต์ ซึ่งออบเจ็กต์ต่างๆจะมีความเป็นอิสระไม่ขึ้นต่อกัน แต่มีการทำงานร่วมกันโดยติดต่อ

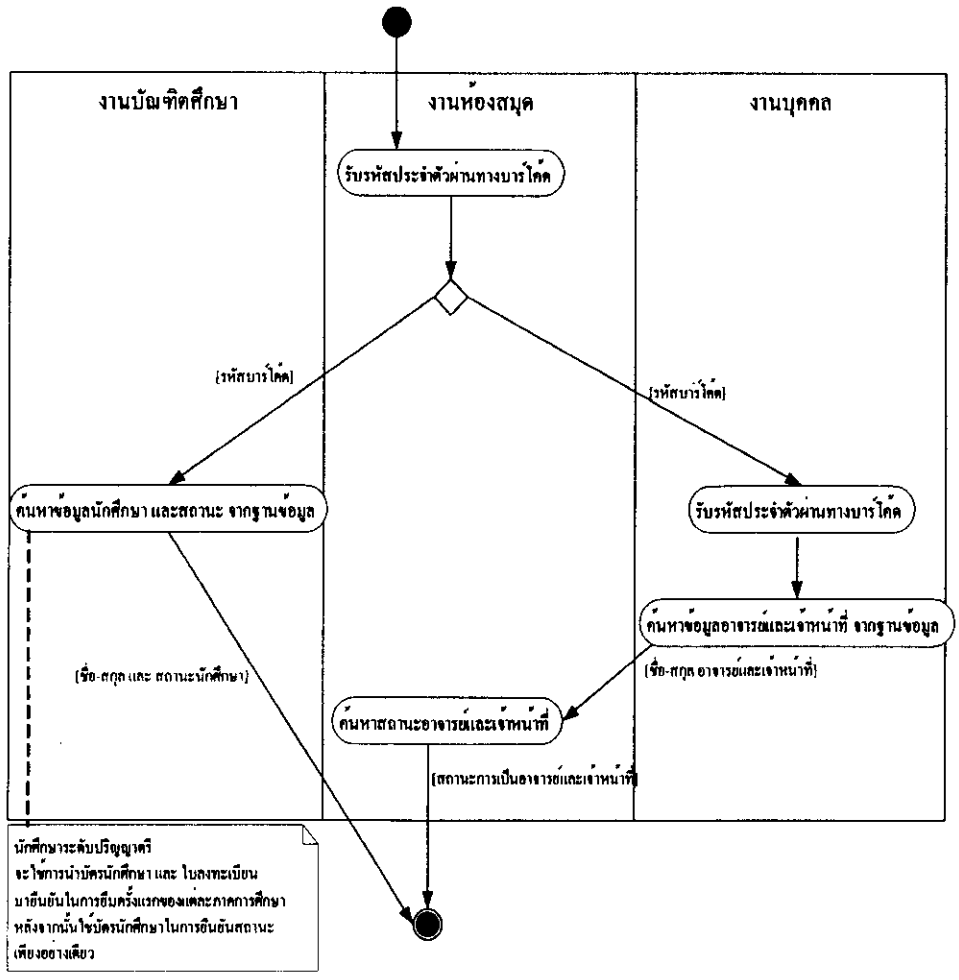
ถึงกันโดยการส่งเมสเสจ (message) วิธีการเชิงวัตถุจะมีการกำหนดคลาส(class) ซึ่งเป็นการจัดกลุ่มให้แก่ออบเจ็กต์ต่างๆที่มีคุณสมบัติหรือพฤติกรรมบางอย่างเหมือนกัน เมื่อเวลาใช้งานเราจะไม่ใช้งานคลาสโดยตรงแต่จะสร้างอินสแตนซ์(instance) ของคลาสขึ้นมาใช้งานแทน UML เป็นภาษาหนึ่งที่ใช้ในการสร้างแบบพิมพ์เขียวให้แก่กระบวนการพัฒนาซอฟต์แวร์ ซึ่งจะทำให้ผู้พัฒนามีความเข้าใจในระบบงานมากยิ่งขึ้น ข้อดีของ UML สรุปได้ดังนี้

- UML รวมข้อดีของโมเดลต่างๆเอาไว้ ได้แก่ Data Model , Business Model , Object Model , Component Model
- เป็นภาษาที่เป็นมาตรฐานเปิด โปรแกรมทุกภาษาในปัจจุบันสนับสนุน UML
- ภาษา UML คลอบคลุมทุกส่วนในวัฐจักรพัฒนาระบบ
- เป็นภาษาที่ง่ายต่อการเรียนรู้ การทำความเข้าใจ และนำมาใช้งานกับงานประยุกต์ที่ซับซ้อนมากๆได้
- เป็นภาษาที่ได้รับการยอมรับอย่างแพร่หลาย

จากตัวอย่างข้างล่างนี้ เป็น USE CASE DIAGRAM งานตรวจสอบผู้เข้ามาบริการยืมหรือคืนหนังสือ โดยระบบจะทำการตรวจสอบและนำเข้าข้อมูลผู้ใช้บริการจากฐานข้อมูลส่วนทะเบียนนักศึกาาจากงานบัณฑิตศึกษาสำหรับนักศึกษาระดับปริญญาโท ในกรณีที่เป็นนักศึกษาระดับปริญญาตรีจะกำหนดให้นักศึกษานำบัตรนักศึกษาและใบลงทะเบียนมายืนยันในการยืมครั้งแรก ข้อมูลอาจารย์และเจ้าหน้าที่จะขอจากฝ่ายบุคคลเป็นรหัสเพื่อนำมาสร้างบาร์โค้ดและทำบัตรสมาชิก การยืมหรือคืนนั้นระบบจะตรวจสอบสถานะของผู้ยืม ซึ่งระบบจะให้บริการสำหรับผู้เข้าบริการที่มีสิทธิเท่านั้น ซึ่งแบบจำลองนี้เป็นมุมมองที่ช่วยให้นักวิเคราะห์ระบบกับผู้ใช้ระบบสามารถสื่อสารเข้าใจได้ตรงกันว่าผู้ใช้ระบบ ทำให้ทราบได้ว่าระบบสามารถทำกิจกรรมอะไรได้บ้าง โดยอธิบายเป็นลำดับของเหตุการณ์ และให้ผู้ใช้ระบบได้เข้ามามีส่วนร่วมในการวิเคราะและออกแบบระบบตั้งแต่ในขั้นตอนแรกๆของการพัฒนาระบบ

ตัวอย่าง ระบบห้องสมุดที่มีการออกแบบโดยใช้ UML

Activity Diagram ของการตรวจสอบผู้ใช้บริการ



Application Generators

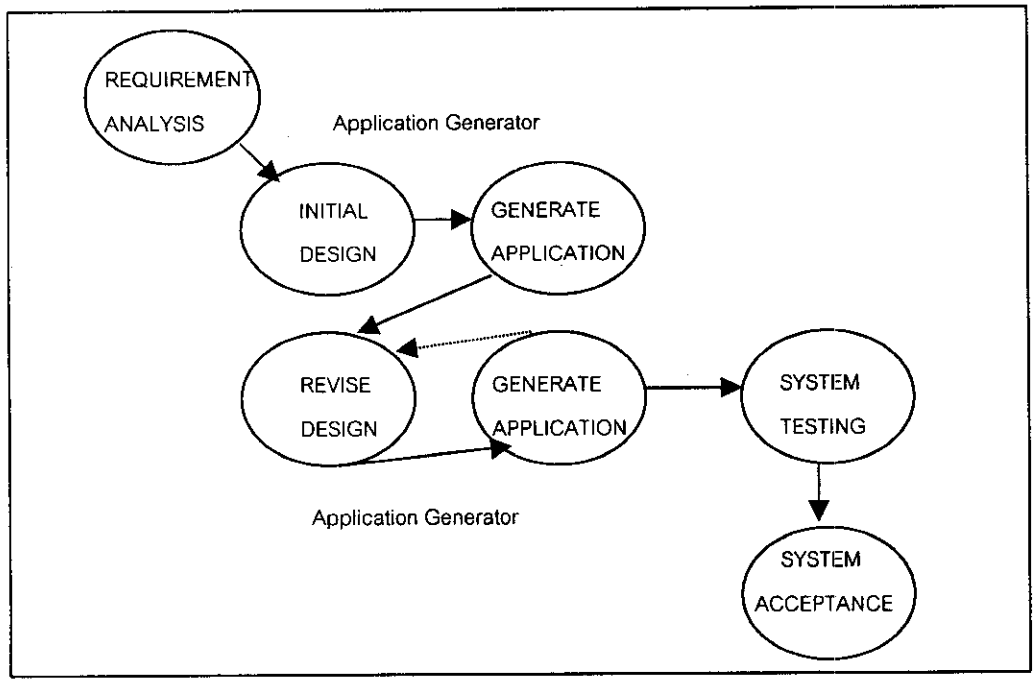
เป็นโปรแกรมที่แปลความต้องการของระบบให้เป็นระบบโดยอัตโนมัติ โดยสามารถสร้างเป็นโปรแกรมประยุกต์ที่ระบุได้ตามความต้องการ ซึ่งโปรแกรมภาษาสมัยใหม่เพียงแต่ผู้ใช้ออกแบบหน้าจอหรือออกแบบโครงสร้างของข้อมูลหรือออกแบบรายงาน โปรแกรมเหล่านี้สามารถสร้างงานประยุกต์เพื่อทำงานตามที่ได้ออกแบบได้โดยอัตโนมัติ โดยทั่วไปแล้วเราใช้สำหรับเป็นเครื่องมือในการพัฒนาในส่วนของระบบย่อยต่างๆของระบบ และนำมาเชื่อมโยงถึงความสัมพันธ์ในภายหลัง

เนื่องจากผู้พัฒนาไม่ต้องเขียนคำสั่งในส่วนต่างๆเหล่านี้ทำให้ลดขั้นตอนในการพัฒนาในส่วนของการออกแบบโปรแกรม, การสร้าง, สำหรับการทดสอบก็ไม่ยุ่งยาก ทำให้ลดเวลาในการพัฒนาระบบลงได้มาก โดยทั่วไปแล้วโปรแกรมภาษาในยุคที่ 4 ส่วนมากเป็น Application Generators เสียส่วนมากและเรานิยมนำไปสร้างต้นแบบเพราะสามารถสร้างได้อย่างรวดเร็ว

รูปภาพที่ 4.28 แสดงถึงการพัฒนาระบบเมื่อใช้ Application Generator ซึ่งเส้นปะหมายถึงการกระทำซ้ำ โดยการออกแบบระบบจะหยุดการทำซ้ำเมื่อลูกค้าและผู้พัฒนาพอใจ

นอกจากนี้ปัจจุบันมี โปรแกรมสำเร็จรูปที่ช่วยในการออกแบบมากมายที่ช่วยนักออกแบบระบบได้ อทิเช่น Visible Analyst design package ที่พัฒนาโดย Visible Systems Corporation ซึ่งช่วยสร้าง Data Flow โดยตรวจสอบความถูกต้องของการไหลของข้อมูล ,สร้างเอกสารต่างๆในการออกแบบ รวมทั้งเอกสารอ้างอิงระหว่างการเอกสารระบุความต้องการกับการออกแบบ ช่วยให้ลูกค้าสามารถติดตามและเข้าใจได้ง่ายขึ้น

รูปภาพที่ 4.28 Application Generator in Iteration development



4.6

การทวนสอบและการคำนวณการออกแบบ

เมื่อนักออกแบบระบบออกแบบระบบเรียบร้อยแล้ว ต้องมีการตรวจสอบความถูกต้อง (validation) และการทวนสอบคุณภาพของการออกแบบ(verification)อีกครั้งหนึ่ง เพื่อให้แน่ใจว่าสิ่งที่ได้ออกแบบสามารถแก้ไขปัญหาและกระทำกิจกรรรมต่างๆได้ตามความต้องการของลูกค้าได้ครบถ้วนถูกต้อง นอกจากนี้ต้องทวนสอบคุณภาพของการออกแบบด้วยว่าออกแบบได้ดีหรือไม่มีวิธีการออกแบบใดที่ดีกว่านี้อีกหรือไม่ ซึ่งถ้าการออกแบบเราใช้เครื่องมืออัตโนมัติ (automated tools) ช่วยในการออกแบบเครื่องมือต่างๆเหล่านี้สามารถตรวจสอบและทวนสอบให้เราได้เลยแต่ก็ไม่ทุกระบบเป็นเฉพาะระบบย่อยๆเท่านั้น

สำหรับในหัวข้อนี้เราจะมาทำความรู้จักกับวิธีการในการตรวจสอบความถูกต้องและวัดคุณภาพของการออกแบบ เพื่อเป็นแนวทางในการออกแบบที่มีคุณภาพต่อไป

Mathematical Validation

ตามสามัญสำนึกวิธีตรวจสอบความถูกต้องของระบบก็คือทดลองใส่ข้อมูลเข้าไปในระบบแล้วดูผลลัพธ์ที่ได้ ถ้าถูกต้องแสดงว่าระบบทำงานได้ถูกต้องถ้าไม่ถูกต้องแสดงว่าระบบทำงานไม่ถูกต้อง การตรวจสอบความถูกต้องของการออกแบบระบบก็ใช้วิธีเช่นเดียวกันแต่เราจะพิจารณาถึงข้อมูลที่มีการเข้าสู่โมดูลต่างๆการประมวลผลและผลลัพธ์ที่ได้จากโมดูลนั้นๆว่ามีความถูกต้องหรือไม่ การแสดงความถูกต้องเราจะดูการทำงานของแต่ละโมดูล ดังนี้

- ถ้าข้อมูลมีความถูกต้อง เมื่อเป็นข้อมูลเข้า(input) ของโมดูลใดๆแล้ว จะได้ผลลัพธ์ (output) ที่ถูกต้องตามที่ได้คาดหวังไว้
- การประมวลผลของโมดูลต้องปราศจากข้อผิดพลาด

Weighted Matrices

การวัดคุณภาพของการออกแบบนั้น เราจะวัดจากคุณลักษณะต่างๆอันได้แก่ Modularity , Testability , Maintainaility , Efficiency , Ease of Understanding , Ease of Modufication , Consistency สมมุติว่าเราออกแบบเพียงวิธีเดียวเราก็ไม่สามารถทราบได้ว่าดีหรือไม่ดีอย่างไร แต่ถ้ามีการออกแบบในการแก้ปัญหาหลายๆวิธีเราสามารถนำแต่ละวิธีมาเปรียบเทียบกันและพิจารณาถึงวิธีที่ดีที่สุดได้ โดยพิจารณาถึงเป้าหมายของการออกแบบและคุณลักษณะเด่นๆของระบบที่เราต้องการ ให้น้ำหนัก(weight)มีค่าสูงสำหรับปัจจัยที่ต้องการให้มีมากๆ ค่าของปัจจัยอื่นๆมีค่าลดหลั่นกันไปตามความต้องการ ดังตารางที่ 4.4 แสดงการกำหนดน้ำหนักของปัจจัยต่างๆที่เป็นคุณลักษณะที่ดีของระบบที่เราต้องการ น้ำหนักมีค่าเท่ากับ 5 ในที่นี้เป็นปัจจัยที่เป็นเป้าหมายของระบบที่เราออกแบบโดยเน้นให้เป็น Modularity มากๆ สำหรับการออกแบบนี้ไม่เน้นถึงประสิทธิภาพโดยให้น้ำหนักมีค่าเท่ากับ 2 มีค่าน้อยที่สุด ค่าของน้ำหนักต่างๆเหล่านี้ขึ้นอยู่กับเป้าหมายของการออกแบบนั่นเองว่าต้องการปัจจัยใดมากน้อยแค่ไหน

ตารางที่ 4.4 Weighted Design Criteria

Characteristic	Weight
Modularity	5
Testability	4
Maintainability	3
Efficiency	2
Ease of Understanding	3
Ease of Modification	3
Consistency	4

เมื่อเราพิจารณาถึงน้ำหนักในการวัดถึงคุณภาพของระบบที่เหมาะสมแล้ว เราจะนำวิธีการออกแบบเพื่อแก้ปัญหาหลายๆวิธีที่คิดขึ้นมาพิจารณาเปรียบเทียบกัน โดยนำมาพิจารณาทีละปัจจัย ให้คะแนนตามความเหมาะสมโดยกำหนดคะแนนเต็ม 10 คะแนน สมมุติว่าระบบหนึ่งมีการออกแบบการแก้ปัญหาได้ 3 วิธี เรานำแต่ละวิธีมาให้น้ำหนักคะแนนถึงคุณลักษณะต่างๆดังตารางที่ 4.5

เมื่อให้น้ำหนักคะแนนเรียบร้อยแล้วการจะเลือกการออกแบบใดดีนั้นเราจะมาทำการคำนวณโดยนำคะแนนที่ได้คูณกับค่าน้ำหนักโดยกระทำทุกๆปัจจัยและนำมารวมกัน ดังรูปภาพที่ 4.29 เป็นการหาผลรวมของ นำน้ำหนักและคะแนนของการออกแบบ1 ทุกปัจจัยมาคูณกัน มีค่าเท่ากับ 184

ตารางที่ 4.5 Weighted Design Matrix

Characteristic	Weight	Design 1	Design 2	Design 3
Modularity	5	8	6	7
Testability	4	7	7	10
Maintainability	3	9	10	8
Efficiency	2	4	6	9
Ease of Understanding	3	10	8	8
Ease of Modification	3	9	9	8
Consistency	4	6	6	7

รูปภาพที่ 4.29 Computing a Design's Score

Characteristic	Weight			Design 1	Design 2	Design 3
Modularity	5	→	40	← 8	6	7
Testability	4	→	28	← 7	7	10
Maintainability	3	→	27	← 9	10	8
Efficiency	2	→	8	← 4	6	9
Ease of Understanding	3	→	30	←10	8	8
Ease of Modification	3	→	27	← 9	9	8
Consistency	4	→	24	← 6	6	7
			=184			

ตารางที่ 4.6 Scores for Weighted Design Matrix

Characteristic	Weight	Design 1	Design 2	Design 3
Modularity	5	8	6	7
Testability	4	7	7	10
Maintainability	3	9	10	8
Efficiency	2	4	6	9
Ease of Understanding	3	10	8	8
Ease of Modification	3	9	9	8
Consistency	4	6	6	7
SCORE:		184	170	193

นำออกแบบ2 และการออกแบบ3 คำนวณในลักษณะเดียวกัน จะได้ผลลัพธ์ของคะแนนที่แตกต่างกันของทั้ง 3 วิธี ดังผลลัพธ์การคำนวณในตารางที่ 4.6 คะแนนที่ได้นี้เราสามารถใช้สำหรับการตัดสินใจถึงวิธีการออกแบบในการแก้ปัญหาที่ดีที่สุดได้ โดยวัดจากคะแนนที่สูงที่สุดจากการวิเคราะห์การออกแบบทั้งสามวิธีนี้สรุปได้ว่าการออกแบบที่ดีที่สุดคือการออกแบบ3

Design Reviews

เมื่อออกแบบเสร็จเรียบร้อยแล้วเราจะนำการออกแบบนี้ไปให้ลูกค้าตรวจดูอีกครั้งหนึ่งเพื่อตกลงถึงการพัฒนาระบบต่อไป โดยการตรวจดู(review) นี้จะกระทำในสองส่วนคือ preliminary design review และ critical design review

Preliminary Design Review เป็นการพบกับลูกค้าเพื่อตรวจสอบความถูกต้องของการออกแบบแนวคิด(conceptual design) โดยกลุ่มบุคลากรที่เกี่ยวข้องในการตรวจสอบประกอบด้วย

- ลูกค้าที่ช่วยในการกำหนดความต้องการของระบบ
- นักวิเคราะห์ระบบผู้ช่วยในการกำหนดความต้องการของระบบ
- ผู้ใช้ระบบ

CT 484 149

- นักออกแบบระบบ
- ผู้พัฒนาระบบคนอื่นๆที่ไม่ได้มีส่วนร่วมในการพัฒนาระบบที่ผ่านมา
- ผู้ควบคุมการประชุม
- ผู้จดบันทึกการประชุม

โดยจำนวนของบุคลากรที่ตรวจสอบนั้นขึ้นอยู่กับขนาดและความซับซ้อนของระบบในการพัฒนา ผู้ควบคุมการประชุมจะเป็นผู้นำในการตรวจสอบโดยแบ่งการตรวจสอบระบบเป็นส่วนๆ มีการประชุมปรึกษาหารือกันในแต่ละส่วน ตัวแทนของแต่ละกลุ่มจะทำการแสดงถึงกิจกรรมต่างๆที่ตนเองรับผิดชอบ ผู้จดบันทึกการประชุม (secretary) ทำหน้าที่บันทึกรายละเอียดต่างๆของการประชุมเพื่อทำเป็นรายงานการประชุม สำหรับผู้พัฒนาคนอื่นๆที่ยังไม่มีส่วนร่วมในการพัฒนาระบบเมื่อฟังการบรรยายอาจมีแนวความคิดใหม่ๆที่ดีเพื่อแนะนำให้ดีขึ้นได้ นักวิเคราะห์ระบบแสดงการออกแบบตามแนวคิดเป็นโครงสร้างและคุณลักษณะของระบบที่ลูกค้าต้องการ ในแง่ของฮาร์ดแวร์ที่ใช้ การปฏิสัมพันธ์กับระบบอื่น ข้อมูลเข้าและผลลัพธ์ของการทำงาน โดยตรวจสอบกิจกรรมต่างๆของระบบไม่ว่าจะเป็นเมนู,ไดอะล็อก,รูปแบบรายงานและส่วนช่วยเหลือเมื่อเกิดความผิดพลาด และถ้าระบบมีการพัฒนาหลายระยะ(phase) ต้องมีการบรรยายคุณลักษณะของระบบทุกๆระยะ ในกรณีที่ผลของการตรวจสอบนั้นพบว่ามีข้อผิดพลาดมากต้องตกลงกันในการออกแบบใหม่อีกครั้ง

Critical Design Review ถ้าการตรวจสอบในส่วนแรกผ่านพ้นไปด้วยดีประสบผลสำเร็จ ลูกค้าจะรู้สึกดีมีความสุขกับระบบใหม่ที่จะพัฒนาขึ้น การตรวจสอบในส่วนต่อไปจะเริ่มขึ้นโดยมีการประชุมปรึกษาหารือกันในส่วนของการออกแบบเทคนิค (technical design) โดยผู้ร่วมประชุมประกอบด้วย

- นักวิเคราะห์ระบบผู้ช่วยกำหนดรายละเอียดของระบบ
- นักออกแบบระบบ
- ผู้ควบคุมการประชุม
- ผู้จดบันทึกการประชุม
- นักออกแบบโปรแกรม
- ผู้พัฒนาระบบอื่นๆที่ไม่มีส่วนในการพัฒนาระบบที่ผ่านมา

การประชุมจะมีบรรยากาศคล้ายๆกับการการตรวจสอบในส่วนแรก แต่จะเน้นในเรื่อง

การออกแบบทางด้านเทคนิคถึงวิธีการในการแก้ปัญหาและคุณภาพของการออกแบบเป็นสำคัญ การบรรยายเป็นการบรรยายโดยคำพูดและมีเอกสารประกอบโดยบรรยายถึงวิธีการและเหตุผลในการแก้ปัญหา ซึ่งนักออกแบบโปรแกรมเมื่อได้รับการบรรยายการออกแบบระบบในส่วนต่างๆ ของระบบแล้วต้องมีความเข้าใจและสามารถนำไปออกแบบโปรแกรมต่อไปได้ กรณีที่เกิดปัญหาขึ้นต้องนำการออกแบบไปแก้ไขตามที่ที่ประชุมได้พิจารณาตัดสินใจและนำมาตรวจสอบใหม่อีกครั้ง

ประโยชน์ของการตรวจสอบการออกแบบนั้นทำให้เราทราบถึงข้อผิดพลาดที่เกิดขึ้นในการออกแบบโดยมีการประชุมกันระหว่างผู้พัฒนาระบบและลูกค้า ซึ่งเป็นผลดีกว่าที่มีการสร้างระบบจนเสร็จแล้วระบบนั้นมีข้อผิดพลาดที่ยอมรับไม่ได้ซึ่งอาจยากต่อการแก้ไข เพราะความผิดพลาดที่ตรวจพบในขั้นตอนนี้สามารถทำการแก้ไขได้ทันท่วงทีก่อนที่จะสร้างระบบ

4.7

เอกสาร

เอกสารที่เกิดขึ้นในขั้นตอนการออกแบบระบบมีด้วยกันสองส่วนด้วยกันคือเอกสารสำหรับลูกค้าที่บรรยายเป็นข้อความถึงการแก้ปัญหาของระบบ อธิบายถึงหน้าที่และกิจกรรมต่างๆที่ระบบสามารถกระทำได้ อีกส่วนหนึ่งเป็นเอกสารสำหรับนักออกแบบโปรแกรม(program designer) ซึ่งอธิบายถึงเทคนิคที่ใช้ในการแก้ปัญหา โดยทั่วไปประกอบด้วยหัวข้อต่างๆดังนี้

- เมนู และ รูปแบบของหน้าจอ โดยอธิบายเป็นไดอะแกรมเป็นลำดับชั้นหรือเครื่องมือที่ให้เห็นภาพรวมของระบบ, การไหลของข้อมูล, ความสัมพันธ์ระหว่างโมดูลต่างๆ
- เครื่องมือที่ใช้ปฏิสัมพันธ์กับผู้ใช้ อทิเช่น ฟังก์ชันคีย์, touch screen, เมาส์ ฯลฯ
- รูปแบบรายงาน
- ข้อมูลเข้า : แหล่งข้อมูลเข้า, รูปแบบของข้อมูล, ขนาดของข้อมูลที่เก็บ
- ข้อมูลออก: แหล่งนำส่งข้อมูล , รูปแบบของข้อมูล, จำนวนของเอกสาร

- คุณลักษณะทั่วไปของกิจกรรมต่างๆในระบบ ถ้าระบบมีการประมวลผลแบบกระจายต้องอธิบายถึงรูปแบบของข่ายงาน การประมวลผลของโหนดในข่ายงานต่างๆ การจัดสรรทรัพยากรต่างๆให้กับผู้ใช้ในข่ายงาน, ข้อกำหนดในเรื่องเวลาในการทำงานและการประมวลผล , บูรณภาพของข้อมูลที่ใช้ในข่ายงาน และระเบียบในการปฏิบัติเมื่อระบบล้มเหลว
- ข้อกำหนดถึงประสิทธิภาพของระบบ เช่นความเร็วในการปฏิบัติงาน , ความสามารถในการตรวจสอบความผิดพลาด , ระบบรักษาความปลอดภัย
- กระบวนการในการจัดการข้อมูล

เอกสารการออกแบบนี้ต้องมีบรรยายโดยอ้างอิงกับเอกสารระบุความต้องการเพื่อให้ทราบว่ากิจกรรมต่างๆที่ได้ออกแบบเป็นการแก้ปัญหาที่จุดใด ซึ่งการอ้างอิงนี้มีประโยชน์ในการติดตามในกรณีที่มีการแก้ไขเกิดขึ้นทำให้ทราบถึงตำแหน่งถึงความผิดพลาดโดยดูจากการอ้างอิงนี้ได้อย่างรวดเร็ว

4.8

สรุป

การออกแบบระบบจะเกี่ยวข้องกับบุคลากรสองกลุ่มคือลูกค้าและนักออกแบบโปรแกรมโดยเราจะออกแบบตามแนวคิด(conceptual design) สำหรับลูกค้าเพื่อบอกถึงหนทางในการแก้ปัญหาโดยเขียนเป็นเอกสารที่เป็นการอธิบายให้ลูกค้าเข้าใจได้ง่าย สำหรับนักออกแบบโปรแกรมนั้นเป็นผู้ที่ต้องพัฒนาระบบต่อไปดังนั้นการออกแบบระบบจะเป็นแนวของเทคนิคโดยการออกแบบจะเน้นถึงคุณภาพของการออกแบบที่ดีจากคุณลักษณะต่างๆ อทิเช่น Modularity , low coupling , high coehsive เป็นต้น การออกแบบอาจมีวิธีการในการแก้ปัญหาหลายวิธี วิธีการคัดเลือกวิธีการแก้ปัญหาเราใช้วิธีการนำมาเปรียบเทียบและให้น้ำหนักคะแนนโดยพิจารณาถึงเป้าหมายหลักของระบบที่เราต้องการให้น้ำหนักลดหลั่นกันไปตามความสำคัญ ต่อจากนั้นนำ

มาให้คะแนนและคิดคำนวณค่าผลรวมของแต่ละปัจจัยออกมาเป็นคะแนนรวม การออกแบบที่มีคะแนนมากที่สุดจะเป็นวิธีการที่เราเลือกในการแก้ปัญหา และก่อนที่จะนำไปพัฒนาในขั้นตอนต่อไปต้องมีการตรวจสอบจากทีมงานและลูกค้าโดยเข้าที่ประชุมเพื่อถกเถียงและทำความเข้าใจให้ตรงกัน ผลลัพธ์ของการประชุมถ้ามีข้อผิดพลาดจะนำไปแก้ไขและเข้าที่ประชุมใหม่จนทุกฝ่ายพอใจ เป็นอันว่าการออกแบบระบบมีความสมบูรณ์และสามารถนำไปพัฒนาระบบในขั้นต่อไป

4.9

แบบฝึกหัด

1. จงให้คำจำกัดความของการออกแบบระบบ
2. จงอธิบายถึงความแตกต่างระหว่าง Conceptual design กับ Technical design
3. ใครเป็นบุคคลที่รับผิดชอบในการออกแบบระบบ และกิจกรรมในการออกแบบระบบมีอะไรบ้าง
4. จงอธิบายถึงความแตกต่างระหว่าง Decomposition และ Composition
5. คุณลักษณะที่ดีของการออกแบบที่ดีมีอะไรบ้าง
6. ในกรณีที่ระบบ data-strong ท่านจะเลือกเครื่องมือใดในการออกแบบถึงจะเหมาะสมจงให้เหตุผล
7. Coupling คืออะไร มีกี่ชนิดอะไรบ้าง
8. Cohesion คืออะไร มีกี่ชนิดอะไรบ้าง
9. Fan-in , Fan-out คืออะไรมีส่วนเกี่ยวข้องกับการออกแบบระบบอย่างไร
10. Prototype คืออะไร มีประโยชน์อย่างไรในการออกแบบระบบ
11. ท่านมีวิธีการเลือกการออกแบบอย่างไรในกรณีที่มีการออกแบบหลายๆวิธี
12. Design Reviews คืออะไร มีประโยชน์อย่างไรในการออกแบบระบบ

4.10

ปฏิบัติ

จากเอกสารระบุความต้องการในบทที่แล้วให้นักศึกษาปฏิบัติการในขั้นตอนการออกแบบระบบ โดยมีการอ้างอิงการออกแบบกับเอกสารระบุความต้องการ โดยเริ่มทำการออกแบบโดยแบ่งปัญหาออกเป็นส่วนย่อยๆ พยายามให้เป็น Modularity , level abstraction , low coupling , high cohesion มีโครงสร้างที่ไม่ซับซ้อนและกำหนดขอบเขตของการทำงานของฟังก์ชันต่างๆ ของระบบให้ดี โดยเลือกเครื่องมือที่เหมาะสมในการออกแบบ ทำให้ทราบถึงฟังก์ชันต่างๆของระบบ ความสัมพันธ์ต่างๆระหว่างโมดูลต่างๆ นอกจากนี้ต้องมีออกแบบนามธรรมของข้อมูล การปฏิบัติการต่างๆกับข้อมูลต่างๆเหล่านี้เขียนเป็นพจนานุกรมข้อมูล

มีการใช้ Automated Tools เพื่อสร้าง Prototype อาทิเช่น เมนู, หน้าจอในการป้อนข้อมูล , ไดอะล็อก, รูปแบบรายงาน ,คำสั่งสอบถาม ฯลฯ โดยจัดทำเป็นเอกสารและนำไป review กับทีมงานและลูกค้า เพื่อแก้ไขให้ตรงกับความต้องการของลูกค้ามากที่สุด เมื่อเสร็จสมบูรณ์ให้นำรายละเอียดต่างๆทำเป็นรายงานการออกแบบส่งอาจารย์ โดยบรรยายให้ครบตามเนื้อหาที่บรรยายในหัวข้อที่ 4.7