บทที่ 1

วิศวกรรมซอฟต์แวร์

มีคนกล่าวไว้ว่าวิศวกรรมซอฟต์แวร์เป็นศิลปะแขนงหนึ่งทางวิทยาศาสตร์ บุคคลที่เรียนสาขาวิทยาการคอมพิวเตอร์ไม่เพียงแต่จะมีความรู้ทางด้านทฤษฎีและการปฏิบัติที่สามารถพัฒนาโปรแกรมต่างๆได้เท่านั้น สิ่งที่สำคัญคือต้องมีความสามารถในการออกแบบและพัฒนาระบบให้สามารถช่วยเพิ่มผลิตผลได้มากขึ้น มีวิธีการในการพัฒนาโปรแกรมให้มีประสิทธิภาพสูงและมีคุณภาพที่ดี หลายๆคนที่เพิ่มเริ่มศึกษาวิธีการเขียนโปรแกรม เมื่อเขียนโปรแกรมได้ตามความต้องการจะดีใจ และเชื่อมั่นว่าโปรแกรมที่เขียนขึ้นนั้นดีที่สุดแล้ว แต่เมื่อเวลาผ่านพ้นไปมีประสบการณ์มากขึ้นได้พบได้พัฒนาโปรแกรมใหม่ๆมากมายและกลับมามองย้อนถึงอดีตจะทราบถึงความแตกต่างว่าโปรแกรมที่เขียนในอดีตนั้นไม่ดีเท่าที่ควร เห็นได้ว่าในการแก้ปัญหาลักษณะงานหนึ่ง ถ้าให้โปรแกรมเมอร์หลายๆคนที่มีประสบการณ์แตกต่างกันแก้ปัญหา วิธีการของโปรแกรมเมอร์แต่ละคนในการแก้ปัญหาจะมีความแตกต่างกันออกไป บางคนเขียนโปรแกรมสั้น บางคนเขียนโปรแกรมยาว อัลกอริทึมที่ใช้ในการแก้ปัญหาของแต่ละบุคคลก็แตกต่างกันออกไป มีผลให้ประสิทธิภาพของการทำงานโปรแกรม มีความเร็วหรือช้าแตกต่างกัน ปัจจุบันเราสามารถชี้ได้ว่าโปรแกรมใดดีกว่าโปรแกรมใดในแง่ของความง่ายในการแก้ไขข้อผิดพลาดหรือการปรับปรุงเปลี่ยนแปลงโปรแกรม ความง่ายในการใช้งานของผู้ใช้ ความง่ายต่อการเข้าใจของโปรแกรมเมอร์ รวมถึงประสิทธิภาพที่วัดด้วยความรวดเร็วในการทำงาน

วิศวกรรมซอฟต์แวร์ เป็นวิชาที่พัฒนาขึ้นมาเพื่อเน้นในการออกแบบ และพัฒนาซอฟต์แวร์ให้มีคุณภาพสูง โดยหาหนทางหรือวิธีการในการแก้ปัญหาเพื่อทำให้ซอฟต์แวร์ที่พัฒนาขึ้นมีคุณลักษณะที่ดีตรงกับความต้องการของลูกค้า ,ผู้ใช้ระบบ และ ผู้พัฒนาระบบ การพัฒนาซอฟต์แวร์มักจะมีปัญหามากมายเกิดขึ้นในระหว่างขั้นตอนการพัฒนาซอฟต์แวร์ซึ่งมีผลให้การพัฒนาซอฟต์แวร์ล้มเหลว การเรียนวิชานี้เพื่อที่จะให้ผู้พัฒนาซอฟต์แวร์ได้ทราบถึงหลัก

การพื้นฐานทางวิศวกรรมในการพัฒนาระบบ รวมทั้งการนำเครื่องมือและเทคนิคต่างๆ มาใช้ในการพัฒนาซอฟต์แวร์

1.1

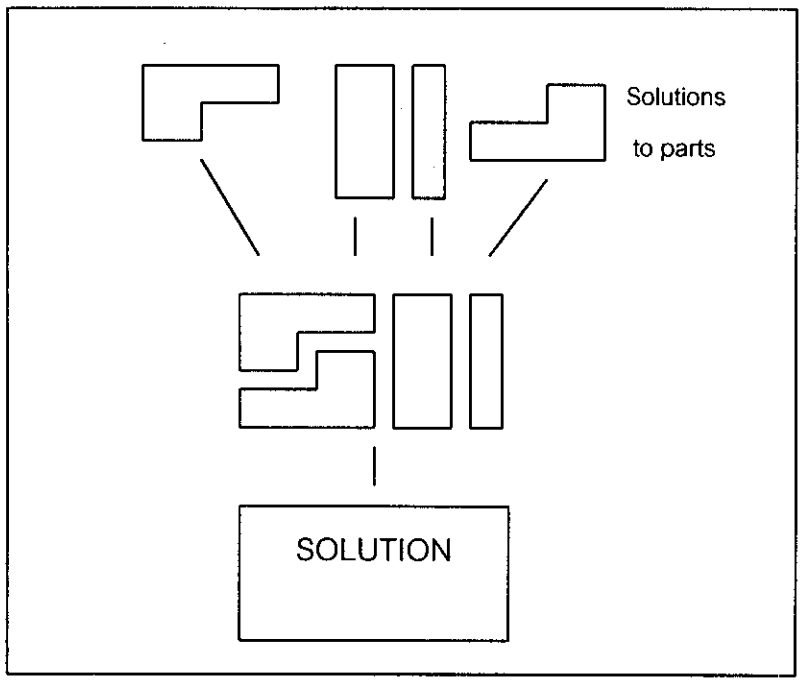
วิศวกรรมซอฟต์แวร์คืออะไร

ผู้พัฒนาระบบงานในปัจจุบันมีความสามารถแตกต่างกัน การพัฒนาระบบงานอย่างไรที่เราถือว่าประสบผลสำเร็จในการพัฒนาระบบ หลายคนอาจมองจากความพึงพอใจของผู้ใช้ระบบเป็นสำคัญ ผู้ใช้ระบบสามารถใช้โปรแกรมได้อย่างสะดวก สามารถช่วยงานที่กระทำเป็นประจำซ้ำๆ กันได้ การทำงานมีความรวดเร็วและสะดวกกว่าในอดีต สามารถเพิ่มผลผลิต และทำให้เกิดความประทับใจกับลูกค้าเพราะสามารถปฏิบัติการได้อย่างรวดเร็ว แต่การพัฒนาระบบบางครั้งก็ไม่เป็นที่ประทับใจของผู้ใช้เหมือนกัน กล่าวคือใช้งานยาก ยุ่งยากในการปฏิบัติ ไม่ตรงกับความต้องการ เกิดความผิดพลาดง่าย ทำให้ไม่เป็นที่ยอมรับ มีผลให้ระบบล้มเหลว ทำให้เสียเงิน เสียแรงงาน และเสียเวลา โดยเปล่าประโยชน์

โดยทั่วไปการแก้ปัญหาของนักวิทยาการคอมพิวเตอร์จะใช้ความรู้ทางด้านคอมพิวเตอร์และการประมวลผลที่ได้ศึกษามาช่วยในการแก้ปัญหาต่างๆ โดยมองถึงความสัมพันธ์ของระบบคอมพิวเตอร์ที่สามารถแก้ปัญหาได้แต่เพียงอย่างเดียว แต่ปัญหาบางเรื่องไม่สามารถใช้เครื่องคอมพิวเตอร์อย่างเดียวเพื่อแก้ปัญหา การแก้ปัญหาที่ถูกต้องนั้นเราต้องเข้าใจถึงธรรมชาติของปัญหา และเลือกใช้เทคโนโลยี เช่นเครื่องมือ (tools)ที่เหมาะสม รวมทั้งเทคนิคต่างๆ เพื่อช่วยในการแก้ปัญหา

การแก้ปัญหา

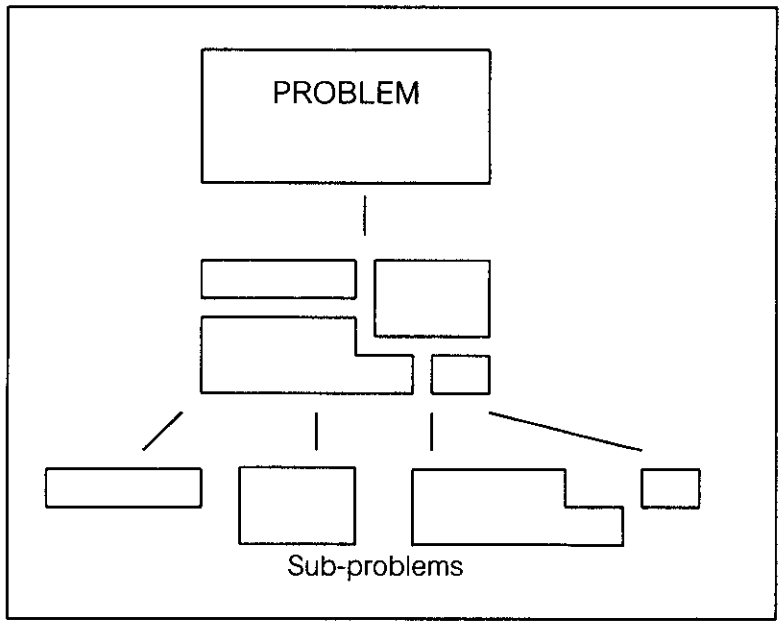
เริ่มจากการวิเคราะห์ปัญหาโดยแบ่งปัญหาหลัก ออกเป็นปัญหาย่อยๆ ทำให้ปัญหานั้นถูกแตกออกเป็นส่วนๆ โดยจะแตกออกไปเป็นส่วนย่อยๆให้ย่อยที่สุดเท่าที่จะเป็นไปได้ เราสามารถทำความเข้าใจกับปัญหาย่อยๆและแก้ปัญหาในแต่ละส่วนย่อยๆได้ง่าย จากรูปภาพที่ 1.1



รูปภาพที่ 1.1 The Process of Analysis

analysis เป็นโปรเซสที่มีการแบ่งโครงสร้างขนาดใหญ่ให้เป็นส่วนย่อยๆ สำหรับรูปภาพที่ 1.2 แสดงส่วนกลับ: synthesis เป็นการนำปัญหาย่อยๆมารวมเข้าด้วยกันเป็นโครงสร้างที่ใหญ่ ซึ่งเทคนิคในการแก้ปัญหาใดๆนั้นต้องประกอบด้วยสองส่วนด้วยกัน คือ การวิเคราะห์ปัญหาเพื่อกำหนดธรรมชาติของปัญหาโดยพยายามแบ่งปัญหาออกเป็นส่วนย่อยๆ และสังเคราะห์ปัญหาเป็นการหาหนทางหรือวิธีการในการแก้ปัญหาจากส่วนย่อยๆซึ่งเป็นผลมาจากการวิเคราะห์นั่นเอง ในที่สุดเราก็สามารถแก้ปัญหาทั้งหมดได้

การหาวิธีการแก้ปัญหานั้นเราจำเป็นต้องหาเครื่องมือ(tools) และเทคนิค(techniques) ที่ช่วยให้เราสามารถแก้ปัญหาได้ง่ายขึ้น เพื่อผ่อนแรง และช่วยให้มีความถูกต้องมากที่สุด ทั้งยังเพิ่มประสิทธิภาพและผลิตผลให้มากขึ้น ซึ่งจะส่งผลให้ผลิตภัณฑ์ที่เราสร้างขึ้นมีคุณภาพดี ตัวอย่างที่ใกล้ตัวเรา อทิเช่นในชีวิตประจำวันถ้าเราต้องการเขียนจดหมายให้กับลูกค้าหลายๆคน ด้วยข้อความที่ซ้ำๆกัน ในสมัยก่อนเราคงต้องเขียนด้วยลายมือถ้าลายมือสวยก็คงไม่เป็นปัญหาอะไร แต่ถ้าไม่สวยอ่านยากก็ไม่เป็นที่ประทับใจกับลูกค้าเท่าไหร่ ต่อมามีการใช้เครื่องพิมพ์ดีด



รูปภาพที่ 1.2 The Process of Synthesis

ใช้เป็นเครื่องมือในการพิมพ์จดหมาย กล่าวได้ว่าถ้าใครต้องการพิมพ์จดหมายจะนึกถึงเครื่องพิมพ์ดีดเป็นอันดับแรกเพราะว่าตัวหนังสือที่พิมพ์สวยงามและง่ายต่อการอ่าน ส่งผลให้คุณภาพของจดหมายดีกว่าจดหมายที่เขียนด้วยลายมือ ในปัจจุบันจะเห็นว่าเครื่องมือในการพิมพ์จะเปลี่ยนแปลงไปตามเทคโนโลยี สมัยนี้การพิมพ์จดหมายทุกคนคงนึกถึงเครื่องคอมพิวเตอร์กันแล้ว เพราะดีกว่าเครื่องพิมพ์ดีดในแง่ของตัวอักษรที่มีหลายขนาดมีหลายแบบ การลบ การแก้ไข การบันทึก การนำกลับมาใช้ สะดวกมากขึ้นและสามารถผลิตได้เป็นจำนวนมากในเวลาอันรวดเร็ว อาจกล่าวได้ว่าเครื่องพิมพ์ดีดหรือเครื่องคอมพิวเตอร์เป็นเครื่องมือที่ช่วยในการแก้ปัญหาซึ่งให้ความสะดวก ความรวดเร็ว มีคุณภาพสูง และเพิ่มผลผลิตได้มากขึ้น สำหรับในการแก้ปัญหาบางอย่าง อาจไม่จำเป็นต้องมีเครื่องมือพิเศษอะไรเพื่อช่วยแก้ปัญหา แต่อาจใช้เทคนิคเพื่อช่วยในการแก้ปัญหาได้ เทคนิคที่กล่าวนี้อาจเป็นเคล็ดลับ ซึ่งอาจเกิดจากประสบการณ์ในการทำงานหรือเป็นพรสวรรค์ของแต่ละบุคคลก็ได้ เช่น การทำอาหารของแม่ครัว เชื่อได้ว่าแต่ละท่านมีรสมือในการปรุงอาหารรสชาติไม่เหมือนกัน ถึงแม้จะมีเครื่องปรุงเหมือนๆกันก็ตาม ความอร่อยที่แตกต่างกันของอาหารเกิดจากการลำดับขั้นตอนหรือวิธีการในการปรุงที่แตกต่างกัน ซึ่งไม่เกี่ยวข้องกับเครื่องมือที่ใช้เพราะการปรุงอาหารใช้เตา หม้อ กระทะ อาหารสดๆ ฯลฯ เหมือนๆ

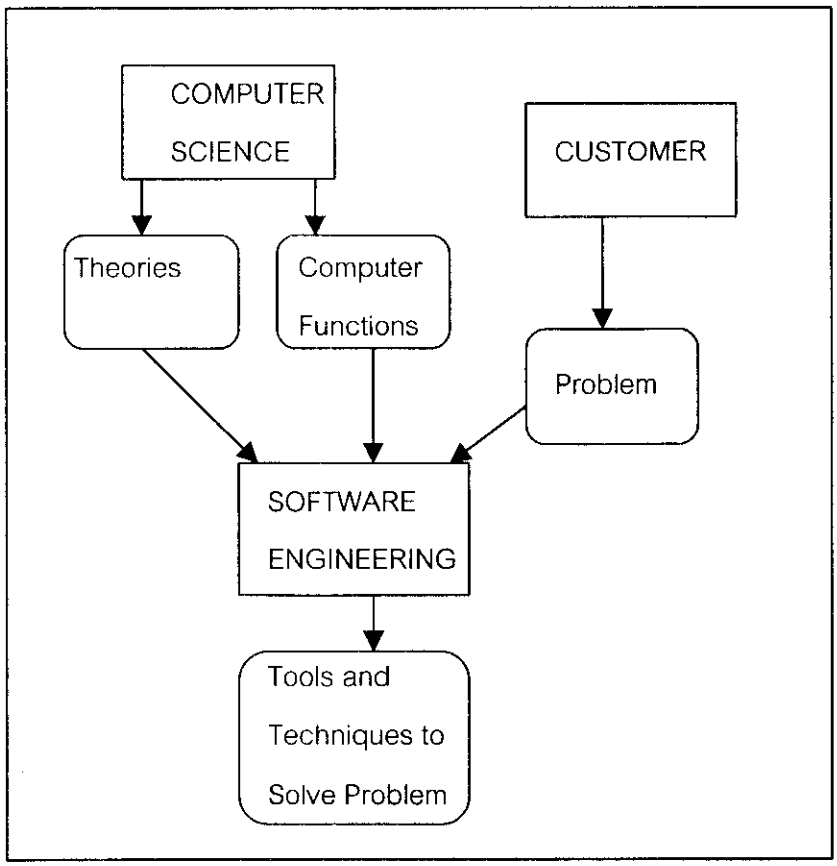
กัน แต่ความอร่อยของอาหารเกิดจากเทคนิคของการปรุงของแม่ครัวแต่ละท่านที่แตกต่างกัน ในแง่ของการพัฒนาโปรแกรม เทคนิคในการพัฒนาเป็นส่วนที่สำคัญในการแก้ปัญหาอีกประการหนึ่ง เทคนิคที่กล่าวถึงนี้คือขั้นตอนกระบวนการในการแก้ปัญหา (procedure or method)นั่นเอง การพัฒนาซอฟต์แวร์ที่มีคุณภาพต้องมีขั้นตอนในการแก้ปัญหาที่ดี และต้องมีเครื่องมือที่ช่วยในการพัฒนาโดยเลือกให้เหมาะสม จะส่งผลทำให้ผลิตภัณฑ์ซอฟต์แวร์มีคุณภาพดี

รูปภาพที่ 1.3 แสดงความสัมพันธ์ระหว่างวิทยาการคอมพิวเตอร์และวิศวกรรมซอฟต์แวร์ ซึ่งวิทยาการคอมพิวเตอร์เกี่ยวข้องการออกแบบฮาร์ดแวร์และการนำทฤษฎีมาประยุกต์ทำงานตามขั้นตอนวิธีเพื่อแก้ปัญหาตามที่ลูกค้าต้องการ สำหรับวิศวกรรมคอมพิวเตอร์แก้ปัญหาโดยใช้เครื่องมือและเทคนิค

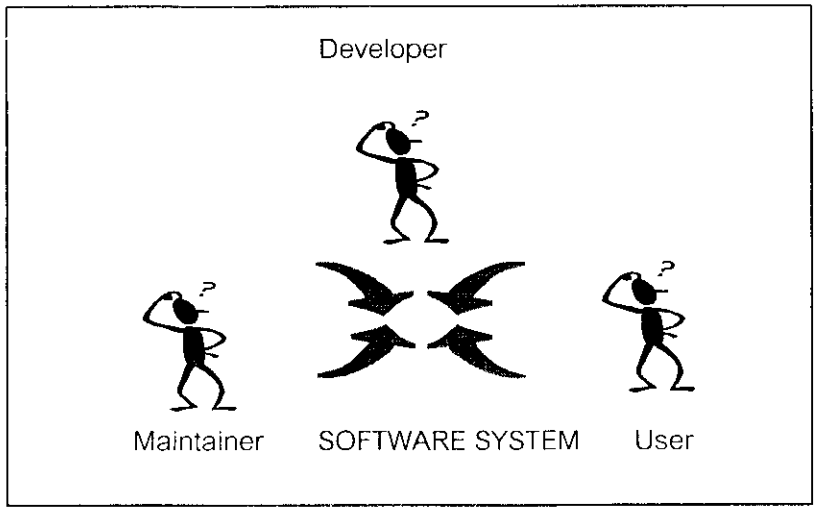
1.2

คุณภาพของซอฟต์แวร์

วิศวกรรมซอฟต์แวร์คือกลยุทธ์ของการผลิตซอฟต์แวร์ที่มีคุณภาพ ความแตกต่างของคุณภาพที่ว่าดีหรือไม่ดีนั้น เราวัดจากอะไร ? โดยทั่วไปแล้วคุณภาพของซอฟต์แวร์ที่ดีนั้นขึ้นอยู่กับปัจจัยหลายๆอย่าง ผู้ที่ตัดสินอาจเป็นผู้ใช้ระบบ ซึ่งวัดจากความง่ายในการเรียนรู้ ง่ายในการใช้งาน ถ้าเป็นผู้วิเคราะห์และออกแบบระบบหรือโปรแกรมเมอร์ วัดคุณภาพจากการบำรุงรักษาโปรแกรม การแก้ไขปรับปรุงระบบเก่าให้มีประสิทธิภาพดีขึ้น รูปภาพที่ 1.4 แสดงให้เห็นว่าซอฟต์แวร์ที่มีคุณภาพสูง ต้องมีคุณลักษณะที่เอื้อประโยชน์ต่อผู้ใช้ , ผู้พัฒนาระบบ และ ผู้บำรุงรักษาระบบ



รูปภาพที่1.3 Relationship Between Computer Science and Software Engineering

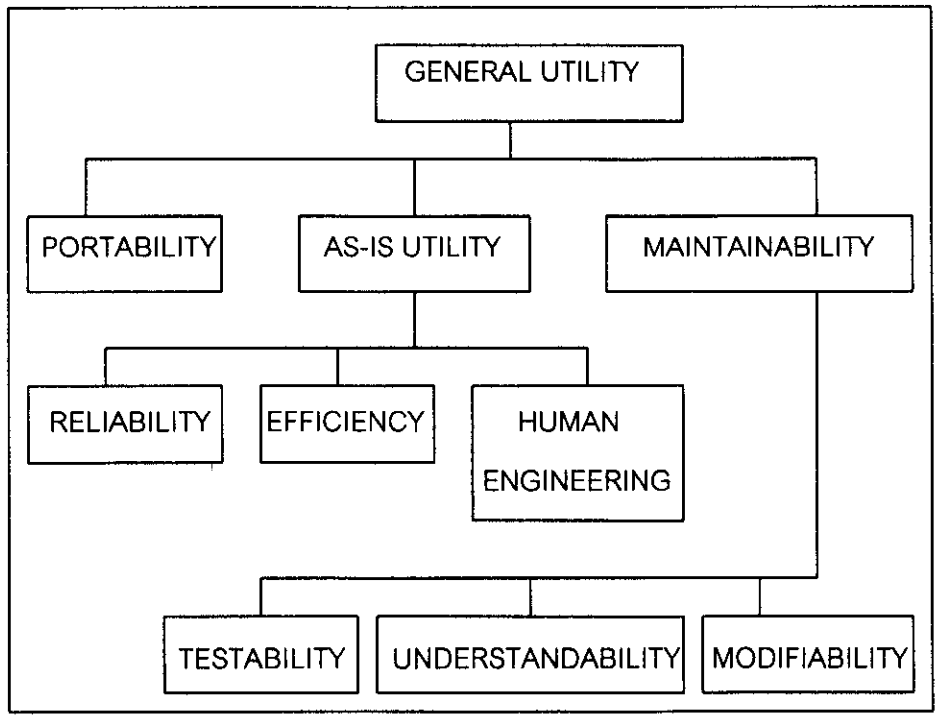


รูปภาพที่ 1.4 Judges of Software Quality

Boehm([BOE78]) ได้กล่าวถึงคุณลักษณะของซอฟต์แวร์ที่มีคุณภาพโดยวัดจากผู้เกี่ยวข้องกับซอฟต์แวร์ 3 กลุ่ม คือ กลุ่มแรกคือผู้ใช้ วัดจากประโยชน์ต่อการใช้งาน(useful)เป็นสำคัญ ซอฟต์แวร์ที่มีคุณภาพจะต้องสามารถกระทำงานตามที่ลูกค้าต้องการได้อย่างถูกต้อง กลุ่มที่สองคือผู้บำรุงรักษาระบบ วัดคุณภาพจากการอัฟเกรดและการเปลี่ยนแปลงระบบ เช่นผู้ใช้ต้องการใช้ระบบจากสถานที่ตั้งอื่นๆ หรือจากคอมพิวเตอร์เครื่องอื่นๆที่ต่างกัน ผู้บำรุงรักษาระบบต้องจัดการได้โดยง่าย สามารถย้ายซอฟต์แวร์จากเครื่องคอมพิวเตอร์ระบบเก่า ไปยังเครื่องคอมพิวเตอร์ระบบใหม่ที่มีการปรับเปลี่ยนอุปกรณ์ฮาร์ดแวร์หรือระบบซอฟต์แวร์ได้โดยปราศจากความผิดพลาดหรือความยุ่งยาก ตัวอย่าง ถ้าตัวแปลภาษาโปรแกรมภาษาซี ถูกแทนที่ด้วยตัวแปลภาษาซีอีกตัวหนึ่งที่พัฒนาขึ้นใหม่อาจเป็นรุ่นใหม่ๆ ผู้บำรุงรักษาซอฟต์แวร์ต้องสามารถติดตั้งซอฟต์แวร์นี้ได้โดยง่ายการทำงานของซอฟต์แวร์สามารถกระทำงานได้ถูกต้องและต้องไม่ลดประสิทธิภาพลงจากเดิม กลุ่มสุดท้ายคือโปรแกรมเมอร์ผู้บำรุงรักษาระบบ ทำหน้าที่เปลี่ยนแปลงระบบตามที่ลูกค้าต้องการ ป้องกันและแก้ไขข้อผิดพลาดที่อาจเกิดขึ้น ซอฟต์แวร์ที่มีคุณภาพต้องเอื้อประโยชน์ให้กับโปรแกรมเมอร์โดยสามารถให้โปรแกรมเมอร์ทราบตำแหน่งของคำสั่งที่ผิดพลาดได้ง่าย เข้าใจและแก้ไขความผิดพลาดได้ง่าย

ผู้ใช้ทั้งสามกลุ่มมีความคาดหวังเหมือนกันว่า ระบบต้องมีความน่าเชื่อถือ และมีประสิทธิภาพ ความน่าเชื่อถือวัดได้จากระดับของความถูกต้องของผลลัพธ์จากการทำงานของระบบ ซอฟต์แวร์ที่มีคุณภาพต้องมีระดับของความถูกต้องอยู่ในเกณฑ์สูง นอกจากนี้ต้องใช้เวลาในการปฏิบัติงาน หรือการตอบสนองกับผู้ใช้อย่างรวดเร็วภายในเวลาที่ผู้ใช้ยอมรับได้รวมทั้งซอฟต์แวร์ต้องสามารถเข้าใจและเรียนรู้ง่ายด้วย ระบบบางระบบมีความน่าเชื่อถือและมีประสิทธิภาพที่ดีเยี่ยมแต่ผู้ใช้ไม่สามารถเข้าใจได้ง่าย ใช้งานยาก ก็มีผลให้ซอฟต์แวร์นี้ไม่เป็นที่ยอมรับ เราสามารถสรุปได้ว่าซอฟต์แวร์ที่มีคุณภาพต้องมีคุณลักษณะดังนี้

(1) สามารถทำงานตามที่ผู้ใช้ต้องการให้กระทำได้อย่างถูกต้อง
(2) ใช้ทรัพยากรคอมพิวเตอร์อย่างถูกต้องและมีประสิทธิภาพ
(3) ง่ายต่อการเรียนรู้ และใช้งาน
(4) ผู้พัฒนาซอฟต์แวร์ สามารถออกแบบ ถอดรหัส ทดสอบ และ บำรุงรักษาระบบได้ง่าย



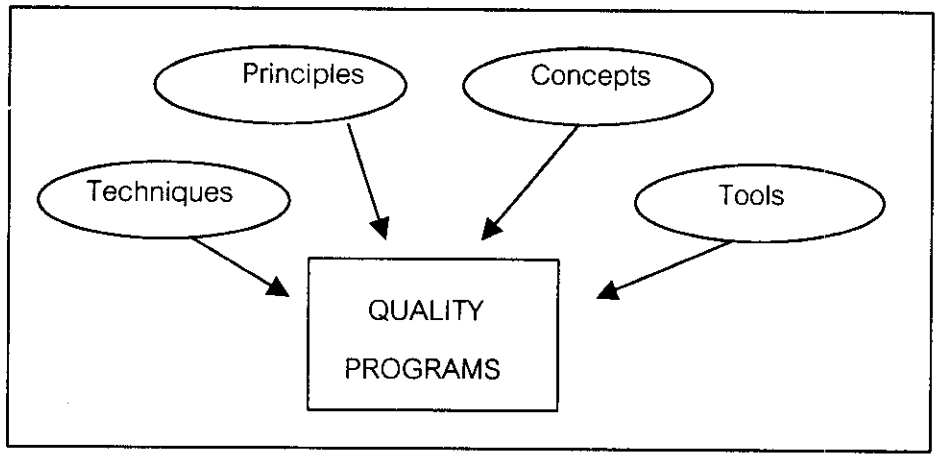
รูปภาพที่ 1.5 Characteristics of software

การผลิตซอฟต์แวร์ที่มีคุณภาพนั้น ถึงแม้ว่าเราจะทราบถึงหลักการในการสร้างหรือมีความเชี่ยวชาญภาษาโปรแกรมที่ใช้ในการพัฒนาซอฟต์แวร์เป็นอย่างดีก็ยังไม่เพียงพอ เรายังต้องมีความรู้ในเรื่องอื่นๆอีก ตามรูปภาพที่ 1.6 แสดงให้เห็นว่าเราต้องเรียนรู้ถึงแนวความคิด เทคนิค และเครื่องมือต่างๆที่ช่วยในการออกแบบและพัฒนาโปรแกรม เราจึงสามารถสร้างโปรแกรมที่มีคุณภาพได้

งานและความรับผิดชอบ

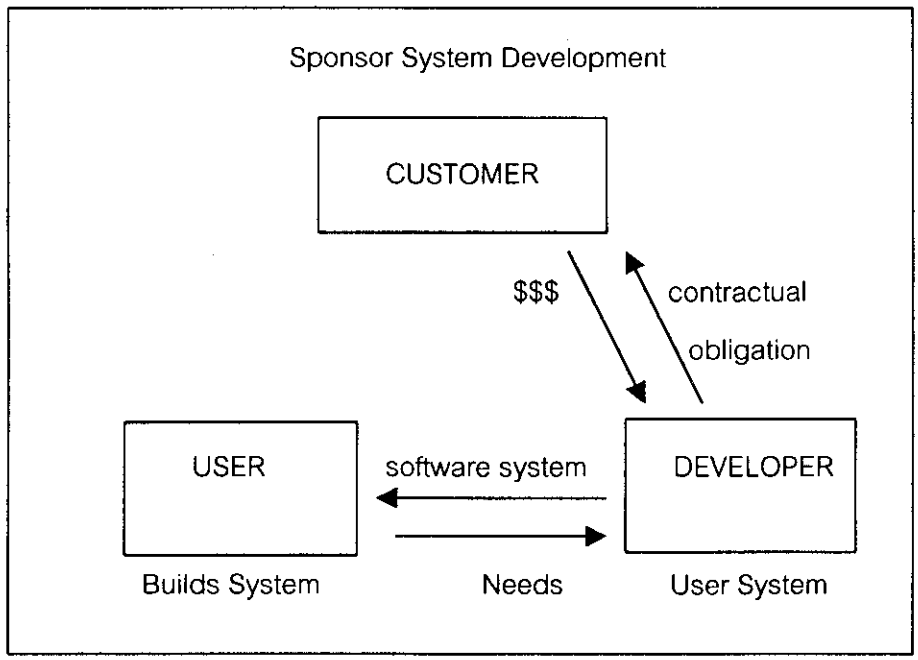
การพัฒนาโครงงานซอฟต์แวร์เริ่มจากการติดต่อสื่อสารระหว่างลูกค้าและผู้พัฒนาซอฟต์แวร์ โดยทำความเข้าใจถึงความต้องการและความจำเป็นในการสร้างระบบให้ถ่องแท้เสียก่อน จากการสอบถาม รวบรวมข้อมูล สังเกตุ ออกแบบสอบถาม เพื่อทำความเข้าใจกับบุคคลต่างๆที่เกี่ยวข้องกับระบบทั้งหมด ก่อนที่จะสร้างระบบขึ้นมา

จำนวนของบุคคลในทีมงานพัฒนาซอฟต์แวร์จะขึ้นกับขนาดและระดับความยากง่ายของโครงการ ซึ่งหน้าที่และความรับผิดชอบของแต่ละบุคคลจะแตกต่างกันออกไปขึ้นอยู่กับขนาดของโครงการ ถ้าเป็นโครงการขนาดใหญ่เราอาจได้รับมอบหมายให้ทำเฉพาะงาน แต่ถ้าเป็นโครงงานขนาดเล็กเราอาจได้ทำในหลายๆหน้าที่ โดยทั่วไปแล้วผู้ที่เกี่ยวพันธ์ในโครงงานทั้งหมดจะแบ่งเป็น 3 กลุ่มใหญ่ๆ คือ ลูกค้า ผู้ใช้ระบบ และผู้พัฒนาระบบ ลูกค้าในที่นี้อาจเป็นบริษัท องค์กร หรือบุคคลที่ลงทุนเสียค่าใช้จ่ายในการพัฒนาระบบซอฟต์แวร์ ผู้พัฒนาระบบอาจเป็นบริษัทซอฟต์แวร์ หรือเป็นองค์กร หรือเป็นบุคคลใดๆที่รับจ้างสร้างระบบซอฟต์แวร์ตามที่ลูกค้าต้องการ สำหรับผู้ใช้ระบบคือบุคคลใดๆ หรือเจ้าหน้าที่ หรือพนักงานที่ใช้ระบบในการทำงานเป็นประจำ โดยทั่วไปนั่งอยู่หน้าเครื่องเทอร์มินัล รูปภาพที่ 1.7 แสดงความสัมพันธ์ระหว่างลูกค้า ผู้ใช้ระบบ และผู้พัฒนาระบบ



รูปภาพที่ 1.6 How to Build Quality Programs

ความสัมพันธ์ของบุคลากรทั้งสามกลุ่มนั้น เริ่มจากลูกค้าเจรจาตกลงกับผู้พัฒนาระบบ โดยมีการพูดคุยถึงความต้องการต่างๆในระบบ ซึ่งผู้พัฒนาระบบต้องทำความเข้าใจถึงสิ่งที่ลูกค้าและผู้ใช้ระบบต้องการ เพราะในบางกรณีผู้ที่มาติดต่ออาจไม่ใช่ผู้ใช้ระบบจริงๆ ผู้พัฒนาต้องไปติดต่อกับผู้ใช้ระบบจริงเพื่อให้ได้ข้อมูลที่ถูกต้องและเป็นจริงที่สุด และก่อนที่จะตกลงทำสัญญา ผู้พัฒนาต้องทราบระบบ(system) อย่างกว้างๆเสียก่อน ว่าจะมีแนวทางในการสร้างหรือแก้ปัญหา และกำหนดผลลัพธ์ในสภาพแวดล้อมที่ลูกค้าต้องการได้อย่างไร



รูปภาพที่ 1.7 Reletionships among Customer,User,and Developer

1.3

ระบบ

โครงงานที่พัฒนาขึ้นมา โดยทั่วไปจะมีการติดต่อกับผู้ใช้ผ่านอุปกรณ์ฮาร์ดแวร์ โดยรับข้อมูลจากภายนอกระบบมาประมวลผลกับฐานข้อมูลที่มีอยู่ หรือเป็นการติดต่อสื่อสารข้อมูลระหว่างระบบคอมพิวเตอร์อื่นๆ ดังนั้นการรับข้อมูลต่างๆเข้ามาเพื่อกระทำงานเป็นสิ่งที่สำคัญอย่างยิ่ง เราต้องทราบว่าข้อมูลที่รับเข้ามาเป็นอย่างไร มีขอบเขต(boundary) เป็นอย่างไร ข้อมูลที่ถูกต้องเป็นอย่างไร ข้อมูลที่ผิดพลาดเป็นอย่างไร สมมุติว่าเราได้รับมอบหมายให้เขียนโปรแกรม

พิมพ์สลิปเงินเดือนของพนักงานในบริษัท สิ่งที่โปรแกรมต้องกระทำได้คือสามารถรับจำนวนชั่วโมงทำงานของพนักงานทุกๆคนได้ คำนวณเงินเดือนที่ได้รับจากอัตราค่าจ้างตามชั่วโมงที่ทำงานจริง คำนวณภาษีที่ต้องจ่าย คำนวณรายรับและรายจ่ายอื่นๆ กล่าวคือเราต้องทราบถึงการดำเนินงานหรือกิจกรรมที่เกี่ยวข้องทั้งหมดที่มีอยู่ในระบบทั้งหมด และผลิตเป็นรายงานเงินเดือนของพนักงานแต่ละคนซึ่งเป็นผลลัพธ์จากการทำงานของระบบนี้

สมาชิกของระบบ

ระบบหนึ่งๆ ประกอบด้วย**กิจกรรม**(activities)ต่างๆมากมาย กิจกรรมหมายถึงเหตุการณ์ต่างๆที่เกิดขึ้นในระบบ ซึ่งอาจเป็นกิจกรรมการเคลื่อนย้ายข้อมูล(move)จากที่หนึ่งไปอีกที่หนึ่ง เช่น ข้อมูล A ถูกย้ายจากแฟ้มข้อมูลหนึ่งไปยังแฟ้มข้อมูลอีกแฟ้มหนึ่ง กิจกรรมการเปลี่ยนแปลงค่าหรือคุณลักษณะของข้อมูล เช่นการคำนวณค่าของข้อมูลทำให้ข้อมูลเปลี่ยนแปลงไปจากเดิม หรือกิจกรรมการรวมข้อมูล เป็นต้น

กิจกรรมหนึ่งๆนั้นประกอบด้วยสมาชิกที่เกี่ยวพันธ์กัน เรียกว่า **ออฟเจ็ก**(objects) หรือ **เอ็นติตี้**(entities) เช่นกิจกรรมการพิมพ์ระเบียนนักศึกษาที่เกรดเฉลี่ยน้อยกว่า 2.0 ซึ่งกิจกรรมนี้เป็นการนำกลุ่มของระเบียนนักศึกษาเฉพาะที่ได้เกรดเฉลี่ยน้อยกว่า 2.0 พิมพ์เท่านั้น ระเบียนต่างๆนี้เองที่เราเรียกว่าเอ็นติตี้ โดยทั่วไปเอ็นติตี้จะอยู่ในรูปของตาราง(table) หรือเมตริกซ์(matrix) หรือฟิลด์ของระเบียนที่มีรูปแบบ เช่น ระเบียนของพนักงาน ประกอบด้วยฟิลด์หรือออฟเจ็กหรือเอ็นติตี้ ดังนี้

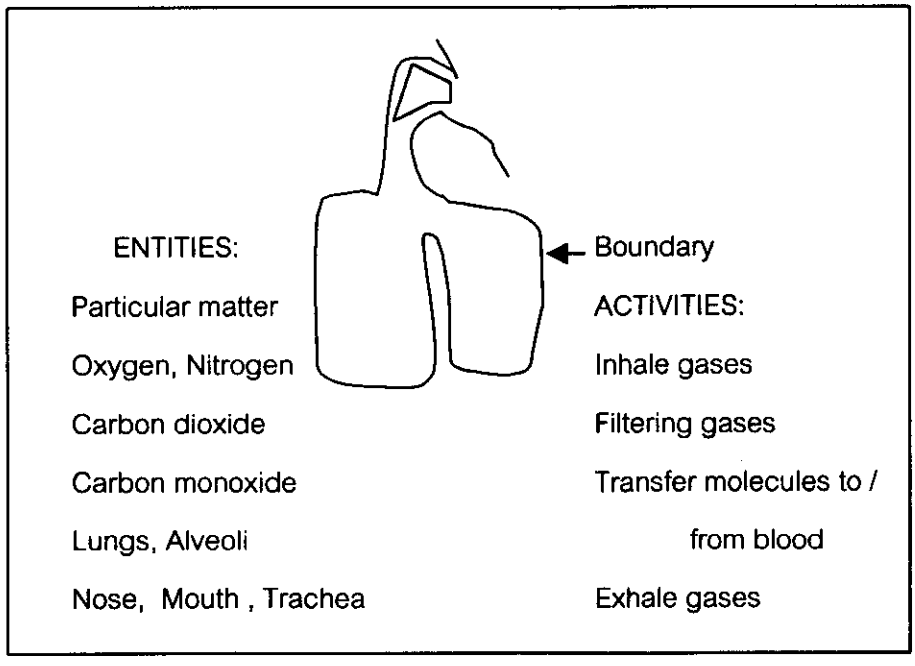
First name	Zip code
Middle name	Salary per hour
Last name	Benefits per hour
Street address	Vacation hours accrued
City	Sick leave accrued
State	

นอกจากนี้ ขนาด ความสัมพันธ์ ชนิดของข้อมูลของแต่ละฟิลด์ ตำแหน่งเริ่มต้นของระเบียน ความยาวของฟิลด์ รวมทั้งการรวมหลายๆระเบียนเป็นแฟ้มข้อมูลและคุณลักษณะของแฟ้มก็ถือว่าเป็นเอ็นติตี้ทั้งสิ้น

เมื่อเอ็นติตี้ และกิจกรรมถูกกำหนดขึ้น เราต้องทราบว่าเอ็นติตี้นั้นๆมีความสัมพันธ์หรือเกี่ยวข้องกับกิจกรรมใดบ้าง เอ็นติตี้อยู่ที่ใดบ้าง มีอยู่ในแฟ้มเรียบร้อยแล้ว หรือต้องสร้างใหม่ในระหว่างดำเนินกิจกรรม บางเอ็นติตี้ถูกใช้งานในหลายๆกิจกรรม เราต้องทราบด้วยว่าการใช้เอ็นติตี้ต่างๆนั้นอยู่ในขอบเขต(boundary)ของระบบหรือไม่

ดังนั้นแนวความคิดในการกำหนดระบบนั้นคือการรวบรวมกลุ่มของเอ็นติตี้ กลุ่มของกิจกรรม รายละเอียดของความสัมพันธ์ระหว่างเอ็นติตี้และกิจกรรม รวมทั้งกำหนดขอบเขตของระบบ นอกจากนี้ยังรวมถึงทุกๆสิ่งที่มีการเกี่ยวพันธ์กับเอ็นติตี้อีกด้วย

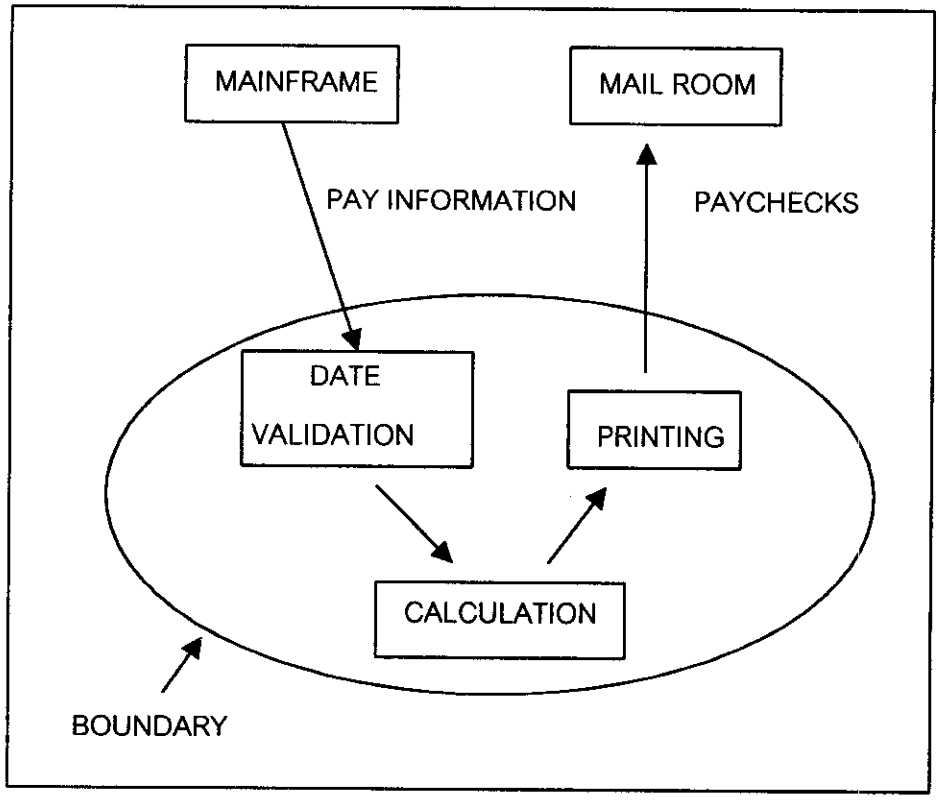
ตัวอย่างระบบการหายใจของมนุษย์ เราสามารถกำหนดขอบเขตเฉพาะในร่างกายเท่านั้น เอ็นติตี้คืออวัยวะต่างๆภายในร่างกายที่เกี่ยวข้องกับการหายใจเช่น จมูก หลอดลม ก๊าซออกซิเจน ก๊าซคาร์บอนไดออกไซด์ ปอด และอื่นๆ สำหรับกิจกรรมในระบบเป็นการกระทำระหว่างเอ็นติตี้ต่างๆเช่น กิจกรรมการหายใจเข้า กิจกรรมหายใจออก และอื่นๆ รูปภาพที่ 1.8 แสดงระบบการหายใจที่ประกอบด้วยกิจกรรมและเอ็นติตี้ต่างๆ ซึ่งมีการรับเอ็นติตี้ที่เข้าสู่ระบบนำเอ็นติตี้ต่างๆที่เกี่ยวข้องมาประมวลผล แลผลิตเป็นเอาต์พุต ดังภาพ



รูปภาพที่ 1.8 Respiratory

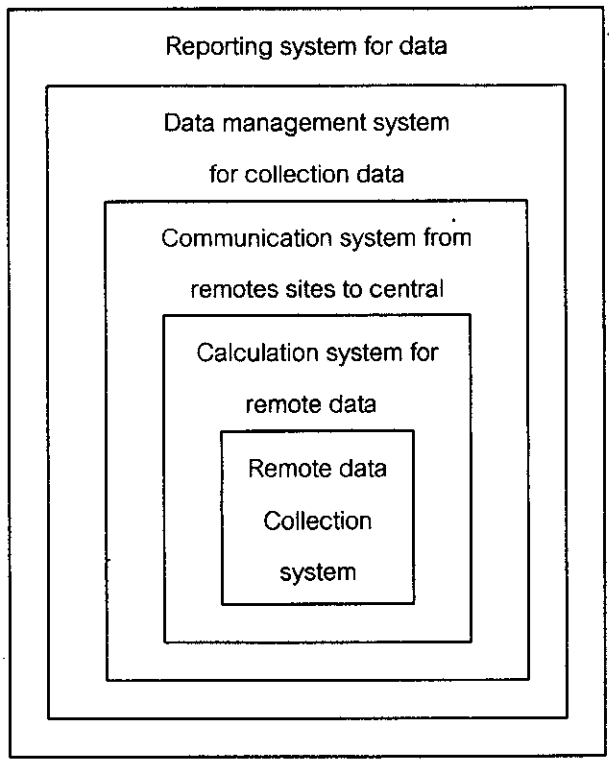
สำหรับการพัฒนาระบบขึ้นมานั้น เราต้องทราบถึงความสัมพันธ์ของสมาชิกภายในระบบอย่างชัดเจน ในแง่ของขอบเขตของระบบ ความสามารถของระบบตามที่ผู้ใช้ต้องการ การปฏิบัติการของระบบ จุดเริ่มต้นของระบบ การติดต่อกับระบบภายนอก รวมทั้งผลลัพธ์ซึ่งเป็นจุดสิ้นสุดของระบบ

ตัวอย่างระบบ Paycheck ระบบนี้เป็นระบบการคิดเงินเดือนให้กับพนักงานในบริษัทแห่งหนึ่ง โดยอินพุตที่เข้าสู่ระบบเป็นข้อมูลการจ่ายเงินเดือนของพนักงานทั้งหมดจากเครื่องคอมพิวเตอร์เมนเฟรมของบริษัท โดยระบบนี้จะนำข้อมูลของพนักงานทุกคนมาคำนวณและส่งเงินเดือนของพนักงานทุกคนไปที่ mailroom เพื่อแจกจ่ายและส่งให้กับพนักงานแต่ละคน รูปภาพที่ 1.9 แสดงระบบ Paycheck ระบบนี้เริ่มต้นด้วยการรับอินพุตซึ่งเป็นเอ็นติตี้ PAY INFORMATION เข้าสู่ระบบต่อจากนั้นมีกิจกรรมการตรวจสอบวันที่ ต่อด้วย กิจกรรมการคำนวณ และกิจกรรมการพิมพ์ ซึ่งความสัมพันธ์ของแต่ละเอ็นติตี้และกิจกรรมเป็นการทำงานตามลำดับ จุดจบหรือผลลัพธ์ของระบบคือPAYCHECKS ที่ส่งไปยัง mail room



รูปภาพที่ 1.9 Paycheck System

เมื่อกล่าวถึงระบบ เกือบทุกระบบจะต้องมีความเกี่ยวพันธ์กับระบบอื่นๆ จะเป็นระบบที่ทำงานเพียงลำพังนั้นมีน้อยมากจะต้องมีความเกี่ยวข้องกับระบบอื่นๆที่อยู่รอบๆ จากตัวอย่างที่กล่าวข้างต้น ระบบหายใจของมนุษย์ ต้องเกี่ยวข้องกับระบบอื่นๆซึ่งทำงานอยู่ภายในร่างกายมนุษย์ เช่นระบบย่อยอาหาร ระบบหมุนเวียน ระบบประสาท และอื่นๆอีกมากมาย ซึ่งมนุษย์เราถ้าขาดระบบใดระบบหนึ่งไม่ได้เพราะทุกๆระบบทำงานสัมพันธ์กัน



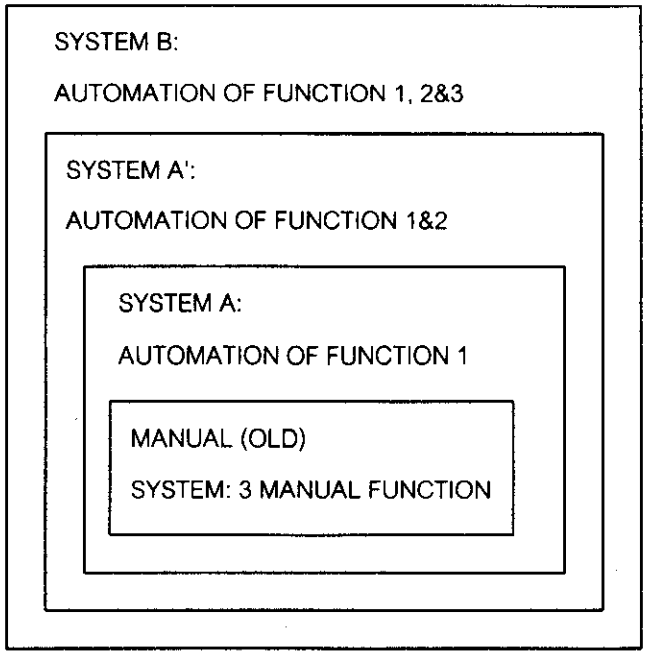
รูปภาพที่ 1.10 Layers of Water Monitoring System

การพัฒนาระบบใดๆ เราต้องเข้าใจถึงความสัมพันธ์ต่างๆภายในระบบอย่างดี ตัวอย่างถ้าเราได้รับมอบหมายให้พัฒนาระบบการตรวจวัดปริมาณของน้ำในแม่น้ำแห่งหนึ่ง เราต้องทราบว่าการตรวจวัดนั้นใช้อินพุตอะไร มาจากไหน มีวิธีการประมวลผลอย่างไร ขั้นตอนการทำงานเป็นอย่างไร ผลลัพธ์ที่ต้องการเป็นเช่นใด มีการส่งผ่านข้อมูลไปยังที่ใด อย่างไร เพื่อที่จะได้ทราบถึงเอ็นติตี้ กิจกรรม ความสัมพันธ์ และขอบเขตทั้งหมดของระบบ ในที่นี้สมมุติว่าระบบ

นี้มีอินพุตจากหลายๆแหล่ง โดยการวัดปริมาณน้ำจะไม่วัดที่จุดๆเดียว ข้อมูลจะถูกรวบรวมมาจาก หลายๆจุดในแม่น้ำสายนี้ที่มีสาขาตั้งอยู่ โดยในระบบมีคอมพิวเตอร์ศูนย์กลางอยู่ในที่แห่งหนึ่งและมีการส่งผ่านข้อมูลมาจากคอมพิวเตอร์สาขาที่อยู่ในที่ๆห่างไกลออกไป คอมพิวเตอร์สาขาสามารถประมวลผลปริมาณน้ำ ณ ที่ตั้งของตนเองและนำส่งข้อมูลมายังคอมพิวเตอร์ศูนย์กลางได้ ดังนั้นคอมพิวเตอร์ศูนย์กลางจะทำหน้าที่รับข้อมูลจากคอมพิวเตอร์สาขาเพื่อรวบรวมและเก็บข้อมูลลงในฐานข้อมูลส่วนกลาง พร้อมกับสร้างรายงานจากฐานข้อมูลที่มีอยู่ตามความต้องการของผู้ใช้ เนื่องจากระบบนี้เป็นระบบที่เกี่ยวข้องกับการติดต่อสื่อสารมีการทำงานหลายขั้นตอน วิธีการที่ดีนั้นเราจำเป็นต้องแบ่งกิจกรรมออกเป็นส่วนย่อยๆ ถ้าเรากำหนดขอบเขตรายละเอียดของระบบได้อย่างถูกต้อง การสร้างระบบขนาดใหญ่จากระบบย่อยๆนำมาประกอบกันจะทำให้การพัฒนาระบบง่ายขึ้น รูปภาพที่ 1.10 แสดงถึงการแบ่งระดับของ Water Monitoring System ซึ่งในแต่ละระดับระบุหน้าที่การทำงานของระบบอย่างชัดเจน โดยสี่เหลี่ยมแต่ละอันจะแทนขอบเขตของการทำงานของระบบนั้น ในที่นี้สี่เหลี่ยมที่อยู่ในสุดจะเป็นระบบที่เล็กที่สุดออฟเจ็กหรือกิจกรรมของส่วนนี้มีความสัมพันธ์กับสี่เหลี่ยมในระดับนอก และมีความสัมพันธ์กันเกี่ยวโยงกันเป็นลำดับ จนได้ผลลัพธ์ที่ต้องการ

โดยทั่วไปแล้วความคิดในการสร้างระบบใหม่ขึ้นมานั้นเนื่องจากว่าเกิดปัญหาในการทำงานของระบบเก่าเช่นความล่าช้า การผิดพลาดของการทำงาน หรือต้องการเพิ่มประสิทธิภาพของการทำงานนั้นดีขึ้น ดังนั้นระบบใหม่มักจะใช้แทนที่ระบบเก่าซึ่งมีการทำงานด้วยแรงงานมนุษย์ หรืออาจเป็นเครื่องจักรอัตโนมัติรุ่นเก่าที่ล้าสมัย การพัฒนาระบบบางครั้งผู้ใช้ระบบอาจเป็นปัญหาที่สำคัญ มีผู้ใช้เป็นจำนวนมากที่ไม่มีความรู้ทางด้านคอมพิวเตอร์ ต่อต้านไม่ยอมรับระบบใหม่ ดังนั้นเราต้องเข้าใจถึงความแตกต่างในการทำงานของระบบเก่าและระบบใหม่นำมาออกแบบและพัฒนาในลักษณะค่อยเป็นค่อยไป โดยค่อยๆเปลี่ยนแปลงระบบทีละส่วน ทีละส่วน จนสมบูรณ์ จากรูปที่ 1.11 แสดงการเปลี่ยนระบบ A ไปเป็นระบบ B ซึ่งการพัฒนาจะค่อยๆเปลี่ยนแปลงจาก A เป็น A' และ เป็น B ตามลำดับ

การพัฒนาระบบนั้นเราอาจเพิ่มระยะในการพัฒนา เพื่อให้ระบบเก่าค่อยๆเปลี่ยนแปลงอาจเป็นการเปลี่ยนแปลงทางด้านฮาร์ดแวร์หรือซอฟต์แวร์ทีละน้อยๆ เช่นในระยะแรกเพิ่มฮาร์ดแวร์ใหม่บางชิ้นเข้าไปเพื่อให้ผู้ใช้คุ้นเคยเสียก่อน ระยะที่สองมีการแทนซอฟต์แวร์แทนการทำงานเดิมที่กระทำด้วยแรงงานคน ระยะต่อๆไปก็ค่อยๆเพิ่มโน่นเพิ่มนี่ไป จนกระทั่งเป็นระบบใหม่ที่ได้



รูปภาพที่ 1.11 Incremental Steps from Old System to New System

ออกแบบไว้ การพัฒนาในลักษณะนี้เป็นการปรับเปลี่ยนที่ค่อยเป็นค่อยไป ผู้พัฒนาต้องมองภาพของการพัฒนาระบบประกอบด้วยสองลักษณะคือ สถิตย์ (Statically) และ จลย์(Dynamically) กล่าวคือ สถิตย์ เราจะมองในแง่ของระบบทำงานประจำในแต่ละวันได้อย่างไรเพราะการปรับเปลี่ยนจะต้องสามารถทำงานประจำได้ตามปกติ ในขณะที่ จลย์ เรามองภาพของระบบที่ต้องปรับเปลี่ยนจากขั้นตอนหนึ่งไปยังขั้นตอนต่อๆไป จนกระทั่งในที่สุดได้ระบบที่สมบูรณ์ ซึ่งผู้พัฒนาต้องทราบว่าในขั้นตอนไหนปรับตรงส่วนใดและจะเชื่อมต่อการทำงานปกติได้อย่างไร

1.4

วิศวกรรม

เมื่อเราเข้าใจระบบแล้ว ขั้นต่อไปคือการสร้างระบบ(engineering) วิทยาการคอมพิวเตอร์ถือว่าการสร้างระบบเป็นศิลปะ(art) ทางด้านวิทยาศาสตร์ เพราะต้องใช้ความเชี่ยวชาญและความชำนาญของผู้สร้างเป็นสำคัญ ผู้สร้างระบบแต่ละท่านมีการใช้เทคนิค วิธีการ และเครื่องมือต่างๆมาช่วยในการสร้างให้ผลผลิตออกมาไม่เหมือนกัน กล่าวกันว่าการสร้างระบบเปรียบเสมือนกับการสร้างบ้าน ซึ่งบ้านแต่ละหลังมีความสวยงาม ความคงทน ความปราณีต คุณภาพแตกต่างกัน ราคาก็แตกต่างกันไปขึ้นกับวัสดุที่ใช้และการจัดการของผู้สร้าง เฉกเช่นกับการสร้างระบบผู้สร้างสามารถเสาะหาเครื่องมือมาช่วยในการสร้างแตกต่างกันออกไป อาจเลือกใช้ตัวแปลภาษาใหม่ๆที่มีความเร็วในการทำงานสูงมาใช้ในการพัฒนาโปรแกรม หรือใช้โปรแกรมพิเศษเป็นเทคนิคและเครื่องมือที่ช่วยให้การเรียงลำดับ การค้นหาข้อมูล การบันทึกข้อมูลที่รวดเร็วและช่วยประหยัดเนื้อที่ในระบบ หรือใช้เทคนิคใหม่ๆมาช่วยให้การพัฒนาระบบสามารถวัดความก้าวหน้าของงานและจัดการระบบได้ดีขึ้น เป็นต้น เครื่องมือโปรแกรมที่รวบรวมขึ้นเพื่อช่วยในการสร้างระบบเรียกว่า Programmer's Workbench

การสร้างบ้าน

ถ้าเราต้องการมีบ้านสักหลัง ท่านจะทำอย่างไร แต่ถ้าเป็นข้าพเจ้าต้องเลือกดูแบบบ้านในทำเลที่ถูกใจ ใกล้ที่ทำงาน และสนนราคาที่สามารถซื้อได้โดยไม่เดือดร้อน แต่เลือกไปเลือกมาสวยๆถูกใจไปหมด แต่ราคาสู้ไม่ไหว จึงคิดจะซื้อที่และปลูกบ้านเอง ดังนั้นข้าพเจ้าจึงต้องติดต่อหาผู้รับเหมาเพื่อบอกถึงความต้องการต่างๆที่เกี่ยวข้องกับบ้านทั้งหมดเมื่อเข้าใจตรงกันทั้งสองฝ่ายมีการตกลงทำสัญญาในเรื่องของค่าใช้จ่ายและเวลาในการสร้าง ต่อจากนั้นสถาปนิกเขียนแบบตามความต้องการของเรา ถ้าเราไม่ชอบใจก็สามารถเปลี่ยนแปลงแบบได้ตามความต้องการจนเราพอใจ จึงเริ่มสร้างบ้านมีการตกแต่งในส่วนของภายในและภายนอกของบ้าน ซึ่งเราสามารถ

เปลี่ยนแปลงแบบหรืออุปกรณ์ต่างๆได้อีก จนกระทั่งเสร็จสมบูรณ์ตามความต้องการ ซึ่งสามารถสรุปขั้นตอนของการสร้างบ้านของผู้รับเหมาโดยทั่วๆไปได้ดังนี้

(1) กำหนดและวิเคราะห์ถึงความต้องการของลูกค้า

(2) ผลิตและสร้างเอกสารการออกแบบของบ้านโดยภาพรวม เป็นกระดาษพิมพ์เขียว

(3) ผลิตเอกสารซึ่งเป็นรายละเอียดในปลีกย่อยต่างๆของบ้าน

(4) กำหนดและออกแบบองค์ประกอบต่างๆที่เกี่ยวข้อง

(5) สร้างแต่ละองค์ประกอบของบ้าน

(6) ทดสอบแต่ละองค์ประกอบเพื่อให้ได้มาตรฐาน

(7) นำแต่ละองค์ประกอบมารวมกัน

(8) ทำการตรวจสอบความเรียบร้อย แก้ไขเป็นครั้งสุดท้าย เพื่อส่งมอบให้กับผู้ว่าจ้าง

บ้านที่สร้างนี้ต้องตรงตามแบบทุกประการและได้มาตรฐาน ในแง่โครงสร้าง อุปกรณ์ที่ใช้ต้องได้มาตรฐานตามที่กำหนด(ISO) และมีการประกันคุณภาพของงาน

การสร้างระบบ

การสร้างระบบ มีขั้นตอนคล้ายกับการสร้างบ้าน เริ่มจากกลุ่มของผู้ใช้ ลูกค้า และผู้พัฒนาระบบมีการพบปะกัน **เพื่อวิเคราะห์และกำหนดความต้องการ**(requirement)ทำให้ทราบถึงขอบเขต เอ็นติตี้ และกิจกรรมต่างๆที่มีการกระทำภายในระบบและภายนอกระบบ ต่อจากนั้นจะนำความต้องการทั้งหมดมา**ออกแบบระบบ**(system design) เป็นภาพรวมของระบบที่สามารถกระทำได้ นำการออกแบบระบบไปตรวจสอบกับลูกค้าเพื่อความเข้าใจที่ตรงกัน ถ้าลูกค้ายอมรับ การออกแบบระบบนี้จะนำไป**ออกแบบโปรแกรม**(program design)ซึ่งเป็นรายละเอียดอาจเป็นหน้าที่หรือเหตุการณ์ที่เกิดขึ้นในระบบ ต่อจากนั้น**เขียนโปรแกรม** (program) เพื่อให้สามารถทำงานตามที่ต้องการได้ พร้อมทั้ง**ทดสอบในแต่ละส่วนของโปรแกรม** (unit testing) เพื่อให้สามารถทำงานได้อย่างถูกต้อง และนำแต่ละส่วนของโปรแกรมที่ทดสอบมารวมกันเป็น**การทดสอบรวม** (integration testing) การทดสอบในขั้นตอนสุดท้ายเรียกว่า**ทดสอบระบบ**(system testing) ซึ่งทดสอบว่าระบบสามารถทำงานตามหน้าที่และการโต้ตอบได้ตามที่ต้องการหรือไม่ เมื่อระบบสามารถกระทำได้ตามที่ต้องการท้ายสุดของการสร้างระบบคือ การ**ส่งมอบระบบ** (delivery) เพื่อให้ผู้ใช้ทำงานแทนที่ระบบเดิม หลังจากนั้นเป็นขั้นตอนการ**บำรุง**

รักษาระบบ(maintenence) ขั้นตอนนี้เป็นการดูแลระบบให้สามารถทำงานได้ตามปกติหรือมีคุณภาพมากขึ้น ซึ่งอาจเป็นการแก้ไขความผิดพลาดที่เกิดขึ้นหรือการเปลี่ยนแปลงความต้องการจากเดิม เป็นต้น สรุปได้ว่าขั้นตอนการพัฒนาซอฟต์แวร์ประกอบด้วยขั้นตอนต่างๆ สรุปได้ดังนี้

(1) การกำหนดและวิเคราะห์ความต้องการ

(2) การออกแบบระบบ

(3) การออกแบบโปรแกรม

(4) การเขียนโปรแกรม

(5) การทดสอบโมดูล

(6) การทดสอบรวม

(7) การทดสอบระบบ

(8) การส่งมอบระบบ

(9) การบำรุงรักษาระบบ

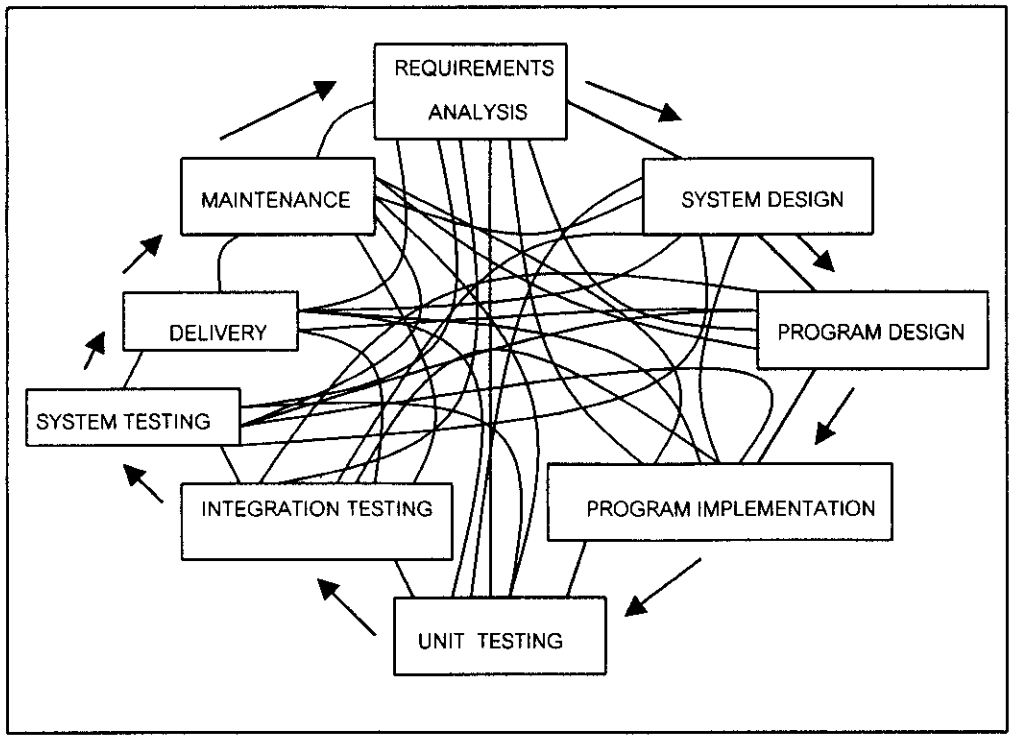
การทำงานในแต่ละขั้นตอนนี้สามารถกระทำซ้ำได้ เช่นออกแบบระบบเสร็จเรียบร้อยแล้ว แต่ลูกค้าพบว่ายังมีความต้องการบางอย่างไม่ครบถ้วนไม่ปรากฏในเอกสาร เราต้องเพิ่มความต้องการเข้าไป บางครั้งต้องออกแบบระบบอีกครั้งหนึ่งเพื่อให้ครอบคลุมความต้องการทั้งหมด ซึ่งการปรับเปลี่ยนในบางกรณีมีผลให้มีการเขียนโปรแกรมใหม่เพิ่ม การทดสอบที่เพิ่มขึ้น รูปภาพที่ 1.12 แสดงให้เห็นถึง วัฏจักรและความเป็นไปได้ในการพัฒนาระบบซ้ำในขั้นตอนต่างๆ

สรุปจุดประสงค์ของการเรียนวิชานี้เพื่อให้ทราบถึงขั้นตอนในการพัฒนาซอฟต์แวร์ รวมถึงกฎเกณฑ์ เครื่องมือ และ วิธีการ ที่จะช่วยให้ระบบซอฟต์แวร์มีคุณภาพ

ทีมงานในการพัฒนาระบบ

ผู้พัฒนาระบบ(developer) ทำหน้าที่สร้างระบบโดยทำงานเป็นทีมงานประกอบด้วยบุคคลต่างๆ ที่มีความเชี่ยวชาญที่แตกต่างกัน ในขั้นตอนแรกของการพัฒนาต้องมีการพบปะพูดคุยกับลูกค้าและผู้ใช้ระบบ เพื่อวิเคราะห์และกำหนดระบบ โดยทำความเข้าใจถึงหน้าที่ทั้งหมดที่ระบบกระทำได้ เขียนเป็นเอกสารระบุความต้องการเพื่อใช้ในการอ้างอิงในการพัฒนาระบบในขั้นตอนต่อไป ผู้ที่ทำหน้าที่นี้เรียกว่า**นักวิเคราะห์ระบบ(analyst)** ต่อจากนั้นนักวิเคราะห์ระบบจะทำงานรวมกัน**ผู้ออกแบบระบบ(designers)** เพื่อสร้างรายละเอียดในระดับย่อยๆที่ระบบ

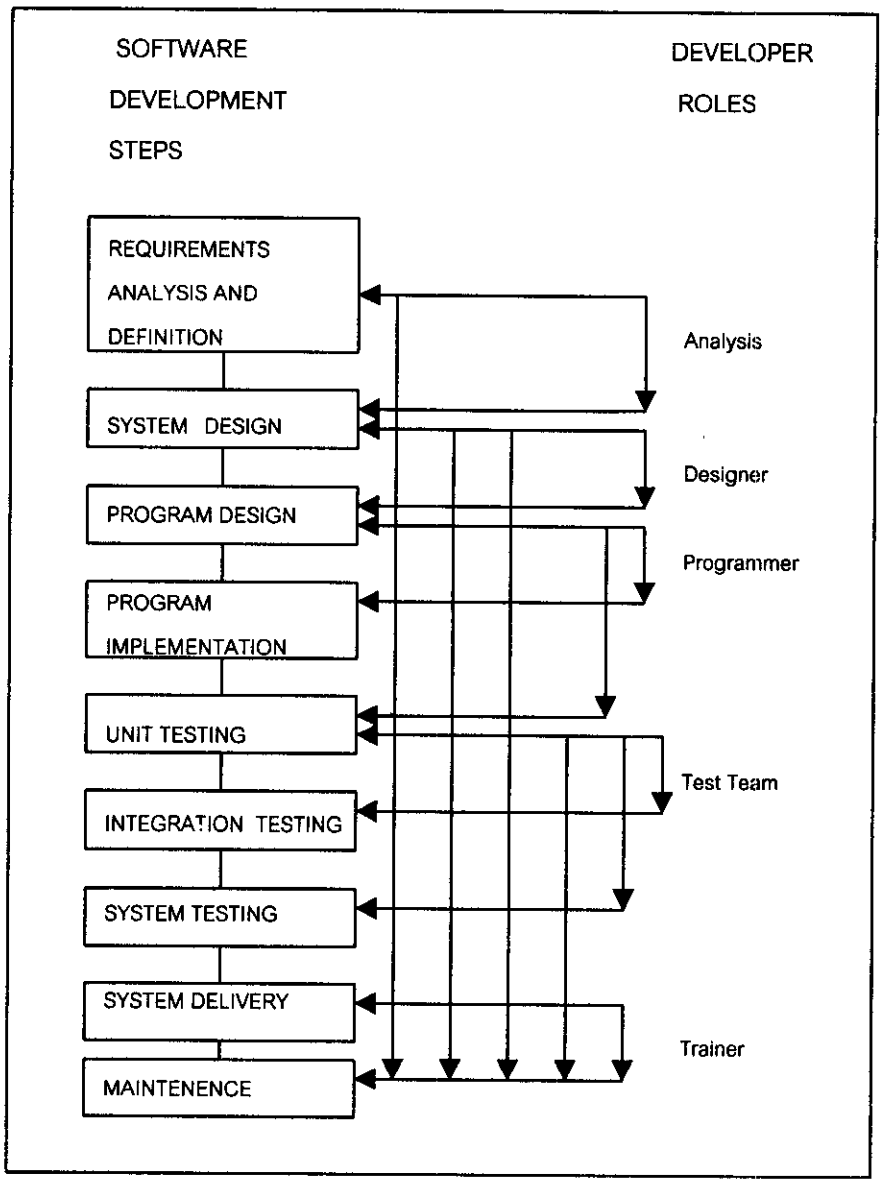
สามารถกระทำได้ ต่อจากนั้นผู้ออกแบบระบบจะทำงานร่วมกับ**โปรแกรมเมอร์** (programmers) เพื่อให้ผู้เขียนโปรแกรมสามารถเขียนคำสั่งโปรแกรมตามความต้องการได้อย่าง



รูปภาพที่ 1.12 The Development Cycle

ถูกต้อง หลังจากนั้นเป็นหน้าที่ของ**ผู้ทดสอบระบบ** (testers) ทำหน้าที่หาข้อผิดพลาดที่อาจเกิดขึ้นในโปรแกรม การทำงานของผู้ทดสอบระบบจะทำงานร่วมกับลูกค้าเพื่อพิสูจน์ว่าระบบสามารถทำงานตามที่ลูกค้าต้องการได้ เมื่อลูกค้ายอมรับจะมีการฝึกฝนการใช้โปรแกรมให้กับผู้ใช้ระบบ โดย**ผู้ฝึก**(trainers) ถึงแม้จะมีการส่งมอบระบบให้ลูกค้าแล้ว ไม่ใช่ว่างานในการพัฒนาจะจบลง ถ้ามีข้อผิดพลาดเกิดขึ้นในระบบหรือความต้องการของระบบเปลี่ยนไป **ทีมงานบำรุงรักษา**(maintenance team) จะเป็นกลุ่มที่ทำหน้าที่รับผิดชอบแก้ไข ปรับปรุง เปลี่ยนแปลงตามความต้องการซึ่งอาจเป็นการแก้ไขการออกแบบ แก้ไขคำสั่งโปรแกรม มีการทดสอบระบบใหม่ และเมื่อมีการปรับเปลี่ยนเกิดขึ้นผู้ฝึกจะต้องสอนหน้าที่ใหม่ๆที่เพิ่มเติมขึ้นมาให้กับผู้ใช้

ระบบเพื่อให้สามารถทำงานได้ตามปกติ รูปภาพที่ 1.13 แสดงถึงสมาชิกของทีมงานในการพัฒนาที่เกี่ยวข้องในขั้นตอนต่างๆของวัฏจักรในการพัฒนาซอฟต์แวร์



รูปภาพที่ 1.13 Roles of the Development Team

สำหรับระบบที่มีขนาดใหญ่และซับซ้อนมากๆ จำนวนเอกสารในระบบก็จะมีมากตามไปด้วย ความสามารถในการบำรุงรักษาระบบนั้นเป็นไปได้ยาก ในทีมงานอาจมี **บรรณารักษ์** (Librarians) เป็นบุคคลที่ช่วยสนับสนุนทีมงานในการพัฒนาระบบ โดยทำหน้าที่เตรียมและเก็บเอกสารทั้งหมดที่ใช้ในระหว่างการพัฒนาระบบรวมทั้งรายละเอียดความต้องการ รายละเอียดของการออกแบบทั้งหมด เอกสารโปรแกรม คู่มือผู้ใช้ แผนการทดสอบและอื่นๆที่เกี่ยวข้องกับเอกสารทั้งหมด นอกจากนี้บุคคลที่ทำหน้าที่ร่วมกับบรรณารักษ์ โดยทำหน้าที่ในการหาเอกสารอ้างอิงหรือ หาตำแหน่งของโปรแกรมคำสั่งที่มีการเปลี่ยนแปลงตามความต้องการใหม่รวมทั้งส่วนของโปรแกรมที่มีผลกระทบกับการแก้ไขเปลี่ยนแปลง เพื่อช่วยผู้พัฒนาในการบำรุงรักษาระบบเรียกว่า configuration management team โดยทั่วไปผู้บำรุงรักษาระบบจะเป็นทีมงานคนละทีมกับผู้ออกแบบและเขียนโปรแกรม และจำนวนของบุคคลในการพัฒนาขึ้นอยู่กับขนาดของโครงงาน ถ้าโครงงานขนาดเล็กอาจมีผู้พัฒนาเพียงสองหรือสามคนและช่วยกันทำงานในหลายๆหน้าที่ แต่ถ้าเป็นโครงงานขนาดใหญ่ทีมงานมีหลายคนหัวหน้าโครงการมีการจัดสรรหน้าที่โดยแยกเป็นกลุ่มๆในการทำงานและมีหน้าที่แตกต่างกันออกไป ขึ้นอยู่กับความชำนาญและประสพการณ์ของแต่ละบุคคล

1.5

ปัญหาในการพัฒนาซอฟต์แวร์

ในตอนต้นนั้นเราได้เปรียบเทียบการพัฒนาซอฟต์แวร์เหมือนกับการสร้างบ้าน บ้านที่เราหวังและตั้งใจปลูกตามความต้องการแต่เมื่อเข้าไปอยู่บางอย่างก็ถูกใจบางอย่างก็ไม่ชอบอยากปรับปรุงเปลี่ยนแปลง บางอย่างไม่เป็นไปตามความต้องการ เหมือนกับการพัฒนาซอฟต์แวร์เมื่อพัฒนาเสร็จลูกค้าอาจไม่ชอบใจในหลายๆสิ่ง เช่น ไม่เป็นไปตามที่ต้องการ หรือมีขั้นตอนในการพัฒนาที่ยุ่งยากหรือคุณภาพไม่ดี เราสามารถสรุปปัญหาในการพัฒนาซอฟต์แวร์ได้ดังนี้

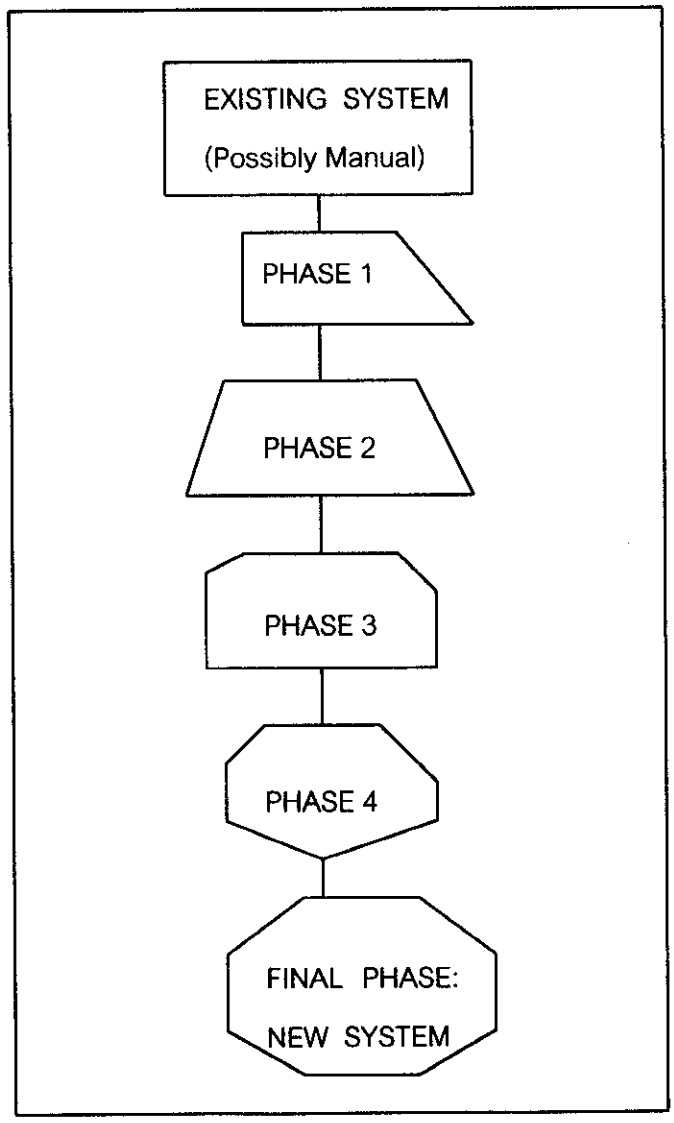
(1) Changing Constraints and Requirement

ในระยะของการพัฒนาซอฟต์แวร์นั้นเรามีการพบปะพูดคุยกับลูกค้าอย่างสม่ำเสมอ โดยนำเสนอความคืบหน้าในการทำงานทุกๆระยะเพื่อสร้างระบบตรงกับความต้องการ บางครั้งลูกค้ามีความต้องการไม่คงที่มีการปรับเปลี่ยนบ่อยครั้ง ในบางกรณีก็ก่อให้เกิดปัญหาในการพัฒนาซอฟต์แวร์ได้เช่นเดียวกัน เช่น ผู้พัฒนาตัดสินใจเลือกฮาร์ดแวร์และซอฟต์แวร์ระบบหนึ่งเพื่อใช้ในโครงงาน แต่เมื่อความต้องการของลูกค้าเปลี่ยนไป ความต้องการของลูกค้านั้นทำได้ยากในการที่จะกระทำจากฮาร์ดแวร์และซอฟต์แวร์ที่มีอยู่ ผู้พัฒนาต้องเลือกใช้ฮาร์ดแวร์และซอฟต์แวร์ใหม่ที่สามารถกระทำได้ตามความต้องการได้ ดังนั้นข้อเสนอแนะประการหนึ่งก็คือในการเลือกเครื่องมือต้องเลือกเครื่องมือที่ใช้ในการพัฒนาซอฟต์แวร์ให้สามารถมีความหยืดหยุ่นในหลายๆกรณีจะทำให้ปัญหาในการพัฒนานั้นลดน้อยลง

(2) Phased Development System

การพัฒนาระบบที่มีขนาดใหญ่และซับซ้อนมากๆ เราต้องค่อยๆแบ่งการพัฒนาเป็นระยะๆ เพื่อให้ง่ายต่อการสร้างและพัฒนา โดยเริ่มจากระบบเล็กๆที่มีความซับซ้อนน้อยๆก่อน โดยระบบที่สร้างขึ้นใหม่จะแทนที่ระบบเก่าโดยค่อยๆเพิ่มความซับซ้อนขึ้นเป็นลำดับ อย่างเช่น ในระยะแรกเราสร้างระบบโดยปรับเปลี่ยนจากระบบเดิมในบางส่วน นำระบบใหม่ที่สร้างขึ้นแทนที่ระบบเก่าค่อยๆให้ผู้ใช้ระบบคุ้นเคย ต่อจากนั้นเริ่มสร้างระบบในระยะที่สอง เพิ่มความต้องการของลูกค้ามากกว่าระยะแรกโดยสร้างและปรับเปลี่ยนมากขึ้นและนำไปรวมกับระบบที่พัฒนาในระยะแรก กระทำในลักษณะนี้จนกระทั่งระบบสมบูรณ์ วิธีการที่เรามีการสร้างและปรับเปลี่ยนระบบเก่าโดยนำส่วนที่สร้างผนวก(merge)กับระบบที่กำลังกระทำงานอยู่นี้เราเรียกว่า phased development

รูปภาพที่ 1.14 แสดงถึงระยะในการพัฒนาซึ่งเริ่มจากระบบที่กำลังดำเนินงานอยู่ในปัจจุบัน ในระยะที่หนึ่งเราเริ่มสร้างระบบใหม่โดยมีการปรับเปลี่ยนการทำงานในระบบเดิมบางส่วนและติดตั้งระบบให้สามารถทำงานได้ตามปกติ ในระยะที่สองมีการเพิ่มความต้องการมากขึ้นปรับเปลี่ยนระบบและนำไปแทนที่ระบบที่กำลังดำเนินงาน การปรับเปลี่ยนต่างๆจะกระทำเป็นระยะๆและนำสิ่งที่พัฒนาขึ้นใหม่ไปผนวกกับระบบเก่า จนกระทั่งระยะสุดท้ายเป็นระบบที่สมบูรณ์ตรงกับความต้องการของลูกค้าทุกประการ



รูปภาพที่ 1.14 Phases Development

สมมุติเราได้รับมอบหมายให้พัฒนาระบบการควบคุมการสับเปลี่ยนสัญญาณโทรศัพท์ในบริษัทแห่งหนึ่ง ในการพัฒนาเราไม่สามารถทดสอบหรือปรับเปลี่ยนระบบได้โดยทันทีทีใด เนื่องจากลูกค้าไม่ต้องการให้ทั้งระบบต้องขาดการติดต่อสื่อสาร ต้องพยายามให้ส่งผลกระทบต่อการการทำงานเดิมให้น้อยที่สุด การพัฒนาระบบนี้จึงต้องแบ่งลำดับขั้นตอนการพัฒนาออกเป็นระยะๆ โดยในระยะแรกอาจวางเป้าหมายสำหรับหมายเลขโทรศัพท์ที่ขึ้นต้นด้วย 1 เท่านั้น ต่อ

จากนั้นในระยะที่สอง ทดสอบกับหมายเลขโทรศัพท์ที่ขึ้นต้นด้วยหมายเลข 2 หมายเลข 3 จนกระทั่งถึงหมายเลข 9 ในระยะต่อๆไป เป็นการทดสอบการติดต่อกันระหว่างหมายเลข จนกระทั่งระบบนี้สามารถติดต่อกันได้ทุกหมายเลข ซึ่งการทำงานนี้เป็นลักษณะที่พัฒนาระบบใหม่แทนที่ระบบเดิมที่ทำงานอยู่ การทำงานประกอบด้วย 2 ระบบที่ทำงานขนานกันไป นั่นคือ development system และ production system โดย development system ใช้สำหรับออกแบบ เขียนคำสั่งโปรแกรม และทดสอบในส่วนของระบบใหม่หรือในระยะถัดไปเมื่อเราแน่ใจว่าระบบใหม่สามารถผนวกหรือแทนที่ระบบที่กำลังปฏิบัติการได้ เราจะนำคำสั่งโปรแกรมใหม่ผนวกกับคำสั่งโปรแกรมเก่าเพื่อผลิตเป็นระบบใหม่(active production system) เนื่องจากระบบนี้ซับซ้อนและมีการการดำเนินงานหลายระยะ การจัดการที่ดีนั้นเราควรเก็บประวัติของการเปลี่ยนแปลงการทำงานของระบบในระยะต่างๆในเอกสาร เพื่อใช้สำหรับการทำความเข้าใจและสืบค้น เพื่อประโยชน์ในการแก้ไขความผิดพลาดที่อาจเกิดขึ้นในภายหลัง

(3) Interaction with Other Systems

ระบบส่วนมากไม่สามารถทำงานได้เพียงลำพังต้องมีการติดต่อกับระบบอื่นๆ ในแง่ของการรับและส่งสารสนเทศต่างๆ ยิ่งระบบที่มีความซับซ้อนมากๆอาจต้องมีการพัฒนาแบบขนานกัน(concurrent)ทั้งระบบเก่าและระบบใหม่ ดังนั้นในการพัฒนาระบบเป็นการยากที่จะแน่ใจได้ว่าข้อมูลที่ติดต่อระหว่างระบบนั้นถูกต้องและสมบูรณ์

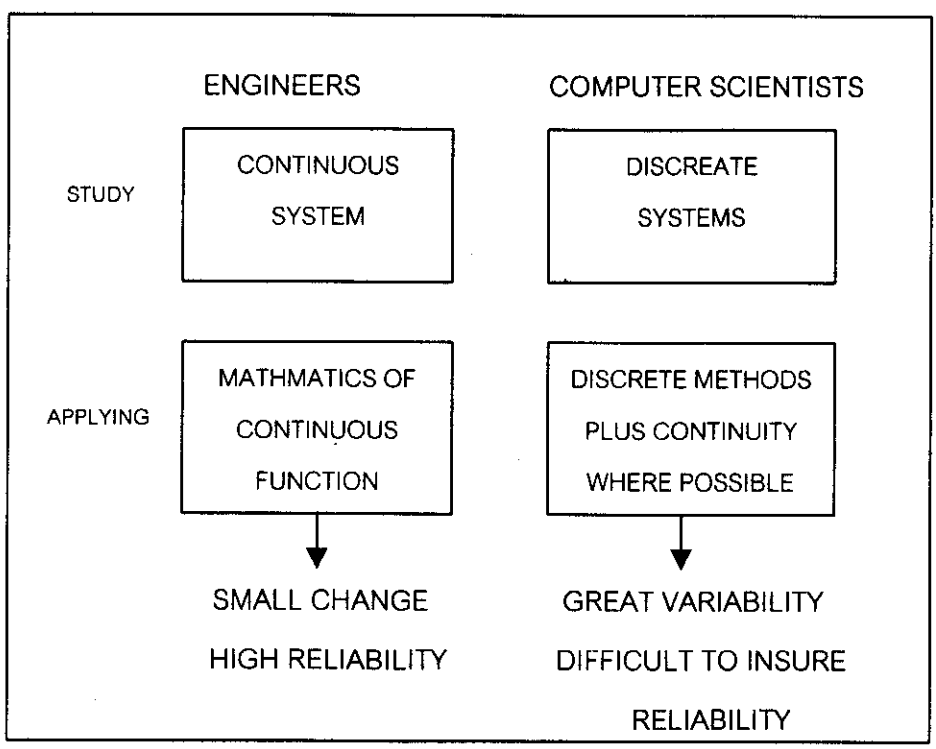
(4) The Nature of Computer System Themselves

ปัญหาในการพัฒนาระบบนอกจากเรื่องการติดต่อสื่อสารกับผู้ใช้ หรือความยากในการปรับเปลี่ยนความต้องการในแต่ละระยะในการพัฒนาแล้ว ชนิดของงานและเครื่องมือที่กระทำในระบบก็เป็นปัญหาที่สำคัญอีกประการหนึ่ง ระบบคอมพิวเตอร์ เราสามารถแบ่งออกเป็น 3 ระบบใหญ่ๆคือ

- Discrete system หรือที่เราเรียกว่า Digital system เป็นระบบตัวเลข ที่นับได้
- Continuous system หรือที่เรียกว่า Analog system เป็นระบบที่ไม่สามารถบ่งบอกเป็นตัวเลขได้ เราวัดเป็นค่าของความต่อเนื่อง
- Hybrid system เป็นระบบที่รวมทั้งสองระบบข้างต้นไว้ด้วยกัน

นักวิศวกรนั้นจะทำงานข้องเกี่ยวกับ continuous system มีการใช้ค่าความต่อเนื่อง

ทางคณิตศาสตร์มาช่วยในการทำงานประกอบความเข้าใจ โดยแทนเป็นภาพวาดแสดงถึงความต่อเนื่องของข้อมูลเป็นงานที่เกี่ยวข้องกับการวิเคราะห์ สำหรับนักวิทยาการคอมพิวเตอร์ จะทำงานเกี่ยวข้องกับ discrete system โดยเกี่ยวข้องกับข้อมูลที่เป็นตัวเลข มีการใช้ตรรกทางคณิตศาสตร์มาช่วยในการตรวจสอบ สำหรับ hybrid system จะเป็นการรวมสองระบบที่มีการทำงานหลายระยะทั้งมีการประมวลผลเป็นตัวเลข และมีการใช้คณิตศาสตร์เพื่อช่วยในการวิเคราะห์ผสมผสานกัน การเลือกระบบที่เหมาะสมจะช่วยให้เราเข้าใจความซับซ้อนของระบบได้ง่ายขึ้นช่วยในการทดสอบระบบเพื่อเป็นหลักประกันความน่าเชื่อถือของระบบ รูปภาพที่ 1.15 แสดงภาพสรุปของความแตกต่างของการเรียนรู้และการนำไปใช้ของนักวิศวกร และนักวิทยาการคอมพิวเตอร์



รูปภาพที่ 1.15 Comparison of Engineering with Computer Science

1.6

ชนิดของโครงงานซอฟต์แวร์

การพัฒนาซอฟต์แวร์นั้นแต่ละโครงงานมีความยากง่ายและความซับซ้อนแตกต่างกัน เราอาจมองง่ายๆจากขนาดของโปรแกรมหรือจากเอกสารการอ้างอิงที่ใช้ในโครงงานทั้งหมด ถ้ามีขนาดและจำนวนมากแสดงว่าโครงงานมีขนาดใหญ่มากมีความซับซ้อนมากแต่ในความเป็นจริงยังมีปัจจัยอีกหลายๆอย่างด้วยกันที่ส่งผลให้ซอฟต์แวร์มีความยากและซับซ้อนแตกต่างกันออกไป คุณลักษณะเหล่านี้สามารถสรุปได้ จากตารางที่ 1.1 โดยวัดระดับความยากง่ายของคุณลักษณะเหล่านี้แบ่งออกเป็น 3 ระดับ คือ ระดับต่ำ(low) ระดับกลาง(medium) และระดับสูง (high)

ตารางที่ 1.1 Degrees of difficulty for Project Development

Characteristic	Low	Moderate	High
Number of functions performed	Small	Medium	Large
Novelty of Function	Standard application	Similar to existing system but with a Few new functions	New theory or approach; never been built before
Number of users requiring multi-user or concurrent access	1	Several	many

ตารางที่ 1.1 Degrees of difficulty for Project Development (ต่อ)

Characteristic	Low	Moderate	High
Multi-tasking	No	Some	Yes
interactive VS. Batch access	Batch or minimal interactive	Highly interactive	highly interactive
Response-time requirements	Off-line; Non-critical	interactive; moderate response time acceptable	Real-time
Need for distributed processing	None	2 computer	3 or more computers
Amount of data stored	Will fit on single disk	Requires 2 or more disk	Requires system to manage disk access
Structure of data	Simple data relationships	Moderately complex relationships	Highly complex relationships
Accuracy of data	Low degree	Moderate degree	High degree
Transaction size	Small	Medium	Large
Remote VS. Local	Local only	Remote	Remote access
Criticality;	Can tolerate	Can tolerate	Can tolerate
tolerance for downtime	several hours of downtime	short periods of downtime	no downtime
Security needs	None	Moderate	High

ตารางที่ 1.1 Degrees of difficulty for Project Development (ต่อ)

Characteristic	Low	Moderate	High
Interaction with other systems	None	Some but well-defined	Much; possible parallel development
Number of phases of development	None	Few	Many
Need for manual Override	No	No	Yes
Dependence on hardware	Independent of hardware	Some	Tied to specific hardware constraints
Stability of specification	Fixed customer requirements	Some changes may occur	Frequent changes in specification
User Sophistication	Familiarity with automated system	Some familiarity with automated system	Naive
Developer sophistication	Has developed similar system with similar tools	Experience with tools, not with application	No experience

ซอฟต์แวร์ที่เราพัฒนามีระดับความยากง่ายแตกต่างกัน ถ้าระบบมีวัตถุประสงค์ในการทำงานน้อยเช่นเป็นระบบคำนวณหาภาษีของพนักงานซึ่งมีวัตถุประสงค์ในการทำงานเพียง

ประการเดียว การออกแบบระบบนี้นับว่าง่ายมาก แต่ในระบบเช่นระบบคลังสินค้าเป็นระบบที่ซับซ้อนขึ้นมีการทำงานที่เกี่ยวพันธ์กัน ผู้ใช้มีหลายระดับและมีความต้องการที่แตกต่างกันไป ซึ่งส่งผลให้การสร้างระบบมีความยากกว่าระบบแรก ดังนั้นจำนวนของงานที่กระทำในระบบ(number of functions performed) จึงเป็นคุณลักษณะหนึ่งที่มีผลต่อระดับความยากง่ายของซอฟต์แวร์ ในบางกรณีถ้าความต้องการของลูกค้าเป็นสิ่งที่เราไม่คุ้นเคยอาจเป็นทฤษฎีใหม่หรือวิธีการใหม่ๆ (novelty of function) ที่เราไม่เคยสร้างมาก่อน การพัฒนาระบบจะยากขึ้นเพราะเราต้องหาทางแก้ปัญหาที่เราไม่เคยพบมาก่อน ดังนั้นยิ่งมีสิ่งใดที่ใหม่ๆในระบบที่เราไม่คุ้นมากเท่าใดจะส่งผลให้การพัฒนาระบบมีความยากตามไปด้วยเท่านั้น

ปัจจุบันมีการใช้เทคโนโลยีสารสนเทศโดยมีการติดต่อผ่านเครือข่ายกันทั่วทุกมุมโลก ระบบคอมพิวเตอร์ส่วนมากมีผู้ใช้งานในระบบมากมาย มีความต้องการใช้ทรัพยากรร่วมกัน ต้องการติดต่อสื่อสารถึงกัน ดังนั้นในระบบที่มีผู้ใช้จำนวนมาก(multi-user) หรือมีการประมวลผลแบบขนาน (concurrent access) ผู้พัฒนาโปรแกรมต้องเขียนโปรแกรมเพื่อโต้ตอบ (interactive)และสนองตอบต่อความต้องการให้กับผู้ใช้อย่างทันท่วงทีในแง่ของเวลาในการตอบสนอง (response-time) ต้องรวดเร็วต้องเป็นที่ยอมรับของผู้ใช้ระบบ ดังนั้นในระบบใดที่มีผู้ใช้ระบบจำนวนมาก สามารถประมวลผลได้มากกว่าหนึ่งโปรแกรม(multi-tasking) และต้องการการโต้ตอบกับระบบโดยทันทีทันใดในระยะเวลาที่รวดเร็ว ต้องเป็นระบบที่มีระดับความยากในระดับสูงกว่าระบบที่มีผู้ใช้งานน้อยหรือต้องการความรวดเร็วในการทำงานน้อยกว่าเสมอ

จำนวนข้อมูลที่เก็บ(amount of data) ในระบบเป็นปัจจัยหนึ่งที่สำคัญถ้าสื่อที่ใช้เก็บไม่สามารถเก็บข้อมูลได้หมดในดิสก์เพียงหนึ่งตัว ต้องใช้ดิสก์หลายๆตัวหรือใช้สื่ออื่นๆเพิ่มเติมการเข้าถึงหรือการจัดการจะยากตามไปด้วยเพราะข้อมูลถูกจัดเก็บกระจัดกระจายทำให้ยากต่อการรวบรวม การเข้าถึงข้อมูล และถ้าในระบบมีโครงสร้างของข้อมูล(structure of data) ที่แตกต่างกันเป็นจำนวนมากและมีความสัมพันธ์ที่ซับซ้อนมาก ก็ยิ่งทำให้การสร้างระบบนั้นยากตามไปด้วยต้องออกแบบถึงความสัมพันธ์ให้เกิดความซ้ำซ้อนน้อยที่สุด และถ้าระบบมีการประมวลผลแบบกระจาย (distribute processing) มีตัวประมวลผลหลายๆตัวต้องมีการจัดการในแง่ของการทำงานและการติดต่อสื่อสารระหว่างกัน ซึ่งเป็นการยากในการควบคุมสารสนเทศต่างๆที่ติดต่อ

ระหว่างกันให้มีความถูกต้อง และถ้าขนาดของรายการ(transaction size)ข้อมูลที่ติดต่อระหว่างกันมีขนาดใหญ่มาก ทำให้อาจเกิดความผิดพลาดของข้อมูลได้ง่าย ยิ่งถ้าระบบต้องการความถูกต้องของข้อมูล(accuracy of data) ในระดับสูงจะทำให้การพัฒนาระบบมีความยากตามไปด้วย เนื่องจากต้องพัฒนาระบบให้มีส่วนของการป้องกันการขโมยข้อมูลหรือเปลี่ยนแปลงข้อมูลระหว่างทางหรือความผิดพลาดของข้อมูลในขณะที่มีการส่งผ่านข้อมูล ซึ่งต้องเพิ่มระบบรักษาความปลอดภัย(security needs)ให้กับระบบมีการเข้ารหัสและถอดรหัสและมีส่วนสำหรับการตรวจสอบสิทธิของผู้ใช้ระดับต่างๆ ซึ่งการประมวลผลถ้าเป็นการติดต่อในระยะไกล(remote) ผ่านข่ายงานโทรศัพท์ จะมีความยากกว่าการประมวลผลภายในองค์กร(local) เพราะความสามารถของระบบยังขึ้นอยู่กับอุปกรณ์ทางด้านฮาร์ดแวร์ที่เราใช้งานอีกด้วย ถ้าลูกค้าระบุถึงฮาร์ดแวร์(dependence on hardware)ที่ใช้ในระบบ เราต้องพัฒนาโปรแกรมให้สามารถทำงานกับฮาร์ดแวร์ซึ่งยากกว่าที่จะพัฒนาระบบได้โดยอิสระ ในระบบงานบางอย่างที่เกี่ยวพันธ์กับชีวิตมนุษย์เช่นระบบช่วยรักษาผู้ป่วย หรือระบบที่ควบคุมการส่งยานอวกาศหรือสำรวจอวกาศ ระบบเหล่านี้ถือว่าเป็นระบบที่ต้องกระทำได้โดยต่อเนื่องเราถือว่าเป็นระบบวิกฤต(criticality) กล่าวคือระบบจะล้มเหลวไม่ได้ การพัฒนาระบบเหล่านี้จะมีความยากมากเนื่องจากเราต้องสร้างระบบออกแบบให้มีสองระบบทำงานซ้อนกันไปคือมีระบบจริงและระบบสำรอง ถ้าระบบจริงล้มเหลวหรือผิดปกติ ระบบสำรองจะทำงานแทนที่โดยอัตโนมัติทันที

การออกแบบให้มีระยะของการพัฒนาระบบ (phases of development) จำนวนมากจะทำให้การพัฒนาโครงงานมีระดับยาก เพราะยากต่อการติดต่อสื่อสารระหว่างกัน ต้องมีการนำคำสั่งโปรแกรมของระยะเก่ามาผนวกกับระยะใหม่ให้ถูกต้อง และถ้าผู้ใช้ระบบต่อต้านไม่ยอมที่จะให้ความร่วมมือ(need for manual override) หรือถ้าเป็นการติดต่อในลักษณะแบบขนาน (parallel development) ยิ่งทำให้ยากมากขึ้น นอกจากนี้ยังมีปัจจัยอื่นๆอีกเช่น การปรับเปลี่ยนความต้องการของลูกค้า(stability of specification) ถ้าลูกค้ามีการเปลี่ยนแปลงบ่อยๆส่งผลให้การพัฒนางานยาก รวมทั้งประสบการณ์ของผู้ใช้ระบบและผู้พัฒนาระบบ(user and developer sophistication) ด้วยว่ามีความคุ้นเคยกับเครื่องมือต่างๆที่ใช้ในระบบมากน้อยแค่ไหน ถ้าไม่เคยใช้หรือไม่มีประสบการณ์เลยจะทำให้การพัฒนาโครงงานนี้มีความยากตามไปด้วย

1.7

กรณีศึกษา

Weaver Farm เป็นพื้นที่ใหม่หลายพันเอเคอร์ที่เป็นส่วนหนึ่งของวณอุทยานแห่งชาติแห่งหนึ่ง มีธรรมชาติที่สวยงามประกอบด้วย ทะเลสาบขนาดใหญ่ ภูเขา ป่าไม้ มีแม่น้ำลำธารสายเล็กๆที่ไหลลงสู่ทะเลสาบ ใช้สำหรับการท่องเที่ยวเพื่อพักผ่อนหย่อนใจ โดยมีพื้นที่หลายๆจุดที่นักท่องเที่ยวสามารถเดินทางไปหาความสำราญได้โดยการเดิน หรือ ล่องเรือ หรือ การปั่นจักรยาน ปัจจุบันมีนักท่องเที่ยวสนใจมาพักผ่อนหย่อนใจเป็นจำนวนมาก

ผู้บริหาร Weaver Farm ต้องการนำระบบคอมพิวเตอร์เพื่อช่วยงานและตัดสินใจ ดังมีรายละเอียดดังนี้

(1) ระบบสามารถเก็บรายละเอียดต่างๆของพื้นที่ทั้งหมด โดยสร้างเป็นแผนที่จำลอง โดยแบ่งพื้นที่เป็น 100 ตาราง-ฟุต-เซลล์ ในแต่ละเซลล์จะเก็บคุณลักษณะทางชีววิทยาแขนงที่ว่าด้วยความสัมพันธ์ของสิ่งมีชีวิตต่อกันเองและต่อสิ่งแวดล้อม คุณลักษณะเหล่านี้ได้แก่การวัดคุณภาพของดิน ชนิดของดิน ต้นไม้หรือพุ่มไม้ที่ปลูกในพื้นที่นี้ และถ้าเป็นแม่น้ำลำธาร จะวัดคุณภาพของน้ำ เก็บไว้ในเซลล์ต่างๆนี้ด้วย

(2) ระบบสามารถเก็บรายละเอียดของการสร้างสิ่งก่อสร้างต่างๆ ในพื้นที่ เช่น ยุ้งข้าว ที่พัก คอร์ดเทนนิส โต๊ะปิคนิก ท่าเรือ อู่เรือ และอื่นๆ

(3) ระบบสามารถสร้างแผนงานการบำรุงรักษาสถานที่ต่างๆ รวมถึงพื้นที่ธรรมชาติ และการก่อสร้างสิ่งใหม่ๆในพื้นที่นี้

(4) ระบบสามารถจำลองผลกระทบของสภาวะแวดล้อมของการเลือกใช้พื้นที่ที่แตกต่างกัน โดยระบบสามารถช่วยให้ผู้บริหารตัดสินใจหาพื้นที่ที่ดีที่สุดเพื่อใช้ประโยชน์โดยไม่ส่งผลกระทบต่อสภาพแวดล้อมเดิม

เราลองมาวิเคราะห์ถึงระดับความยากง่ายของโครงงานนี้โดยเปรียบเทียบกับข้อมูลจากตารางที่ 1.1 ดังแสดงในตารางที่ 1.2 ดังนี้

ตารางที่ 1.2 Difficulty of Weaver Farm project Development

Characteristic	Probable Value for Farm	Difficulty
Number of functions performed	Large	H
Novelty of Function	Similar to existing systems but with a few new functions	M
Number of users requiring multi-user or concurrent access	Several	M
Multi-tasking	Some	M
Interactive VS. batch access	Highly interactive	H
Response-time requirements	Interactive; moderate response time acceptable	M
Need for distributed processing	2 computer	M
Amount of data stored	Requires system to manage disk access	H
Structure of data	Highly complex relationships	H
Accuracy of data	Moderate degree	M
Transaction size	Large	H

ตารางที่ 1.2 Difficulty of Weaver Farm project Development (ต่อ)

Characteristic	Probable Value for Farm	Difficulty
Remote VS. Local	Local only	L
Criticality; tolerance for downtime	Can tolerate several hours of downtime	L
Security needs	Moderate	M
Interaction with other systems	None	L
Number of phases of development	Few	M
Need for manual Override	No	L
Dependence on hardware	Independent of hardware	L
Stability of Specification	Some changes may occur	M
User Sophistication	Some familiarity with automated system	M
Developer sophistication	Experience with tools, not with application	M

จากตารางที่ 1.2 ค่าของความซับซ้อนอยู่ในระดับต่ำ(L) 5 คุณลักษณะ ระดับกลาง(M) 11 คุณลักษณะ และระดับสูง(H) 5 คุณลักษณะ เราสามารถสรุปได้ว่าโครงงานนี้ระดับความยากง่ายในระดับปานกลาง

1.8

สรุป

บทนี้เป็นการแนะนำแนวความคิดของวิศวกรรมซอฟต์แวร์ ที่มีการนำเทคนิคและเครื่องมือมาช่วยในการแก้ปัญหาเพื่อให้ซอฟต์แวร์ที่พัฒนาหรือสร้างขึ้นมีคุณภาพ ซึ่งในขั้นตอนแรกต้องทำความเข้าใจกับปัญหาโดยพยายามมองให้เห็นกลุ่มขององค์ประกอบต่างๆที่มีความเกี่ยวพันธ์กัน ต่อจากนั้นแตกปัญหาใหญ่ออกเป็นปัญหาย่อยๆ และทำการแก้ปัญหาจากปัญหาย่อยๆเหล่านี้และรวมกันเป็นการแก้ปัญหาทั้งระบบ การสร้างซอฟต์แวร์ที่มีคุณภาพต้องทำความเข้าใจถึงแนวความคิด เครื่องมือ เทคนิค และกฎเกณฑ์ต่างๆ นำมาประยุกต์ใช้ในการพัฒนาซอฟต์แวร์

ระบบ หนึ่งประกอบด้วย เอ็นติตี้ กิจกรรม ความสัมพันธ์ และขอบเขตของระบบ ผู้ที่เกี่ยวพันธ์กับระบบจะประกอบด้วยบุคคล 3 กลุ่มคือลูกค้า ผู้ใช้ระบบ และผู้พัฒนาระบบ โดยการพัฒนาระบบประกอบด้วย 9 ขั้นตอนประกอบด้วย การวิเคราะห์ความต้องการ การออกแบบระบบ การออกแบบโปรแกรม การสร้างโปรแกรม การทดสอบโมดูล การทดสอบรวม การทดสอบระบบ การส่งมอบระบบ และการบำรุงรักษาระบบ ซึ่งในแต่ละขั้นตอนสามารถมีการย้อนกลับกระทำซ้ำในขั้นตอนที่ผ่านๆมาได้ถ้าตรวจพบว่าระบบนั้นทำงานไม่ตรงกับความต้องการของลูกค้า ปัญหาที่เกิดขึ้นในการพัฒนาซอฟต์แวร์เป็นเรื่องของความต้องการของลูกค้าไม่คงที่ ความซับซ้อนของโครงงานที่มีผลต่อการติดต่อสื่อสารกับระบบอื่นๆ การแบ่งการพัฒนาเป็นหลายระยะ รวมทั้งระบบคอมพิวเตอร์ที่ใช้ในการพัฒนาด้วย

ซอฟต์แวร์ที่มีคุณภาพสูงต้องสามารถทำงานตามที่ลูกค้าต้องการได้ ใช้ทรัพยากรคอมพิวเตอร์อย่างถูกต้องและมีประสิทธิภาพ ง่ายต่อผู้ใช้ในการเรียนรู้และใช้งาน และผู้พัฒนาสามารถออกแบบ เขียนคำสั่ง และบำรุงรักษาระบบได้ง่าย

ความยากง่ายในการพัฒนาซอฟต์แวร์สามารถแบ่งออกเป็น 3 ระดับคือต่ำ กลางและสูง เราสามารถวัดความยากง่ายของซอฟต์แวร์ได้โดยวิเคราะห์และประเมินจากคุณลักษณะต่างๆ ของระบบเช่น หน้าที่ของระบบ จำนวนของข้อมูล จำนวนของผู้ใช้ในระบบ เวลาในการตอบสนองกับระบบ ระบบรักษาความปลอดภัย ฮาร์ดแวร์ที่ใช้ ระยะในการพัฒนา ประสบการณ์ของผู้พัฒนาระบบ และฯลฯ

1.9

แบบฝึกหัด

1. วิศวกรรมซอฟต์แวร์คืออะไร จงอธิบายพอเข้าใจ
2. คุณลักษณะของซอฟต์แวร์ที่ดีมีอะไรบ้าง
3. สมาชิกของระบบมีอะไรบ้าง จงยกตัวอย่างประกอบพอเข้าใจ
4. จงอธิบายถึง Software Development Life cycle คืออะไร มีขั้นตอนอย่างไรบ้าง
5. บุคลากรที่สำคัญในทีมงานพัฒนาซอฟต์แวร์มีใครบ้าง ทำหน้าที่อย่างไร จงอธิบายพอเข้าใจ
6. ปัญหาในการพัฒนาซอฟต์แวร์มีอะไรบ้าง จงอธิบายให้เข้าใจ
7. คุณลักษณะใดบ้างที่สามารถวัดความยากง่ายในการพัฒนาซอฟตแวร์ จงยกตัวอย่างประกอบให้เห็นจริง

1.10

ปฏิบัติ

หลังจากเรียนบทที่ 1 แล้ว ให้นักศึกษาจัดกลุ่มกันกลุ่มละไม่เกิน 8 คน โดยเลือกระบบงานที่ต้องการทำโครงงาน 1 ระบบที่น่าสนใจ จากระบบงานจริง เช่น ระบบขายสินค้า ระบบร้านเช่า ระบบโรงพยาบาล ระบบการเงิน ระบบพนักงาน เป็นต้น ให้ทำการศึกษาระบบโดยทำเอกสารส่งงานประกอบด้วย

1. วัตถุประสงค์ของระบบ
2. ขอบเขตของระบบ
3. กิจกรรมทั้งหมดของระบบ
4. ข้อมูลที่เกี่ยวข้องของระบบ
5. พิจารณาว่าระบบนี้มีความยากง่ายอย่างไรโดยใช้ข้อมูลจากตารางที่ 1.1