

วิศวกรรมซอฟต์แวร์

มีคนกล่าวไว้ว่าวิศวกรรมซอฟต์แวร์เป็นศิลปะแขนงหนึ่งทางวิทยาศาสตร์ บุคคลที่เรียนสาขาวิทยาการคอมพิวเตอร์ไม่เพียงแต่จะมีความรู้ทางด้านทฤษฎีและการปฏิบัติที่สามารถพัฒนาโปรแกรมต่างๆได้เท่านั้น สิ่งที่สำคัญคือต้องมีความสามารถในการออกแบบและพัฒนาระบบให้สามารถช่วยเพิ่มผลิตผลได้มากขึ้น มีวิธีการในการพัฒนาโปรแกรมให้มีประสิทธิภาพสูงและมีคุณภาพที่ดี หลายคนที่ไม่เริ่มศึกษาวิธีการเขียนโปรแกรม เมื่อเขียนโปรแกรมได้ตามความต้องการจะดีใจ และเชื่อมั่นว่าโปรแกรมที่เขียนขึ้นนั้นดีที่สุดในแล้ว แต่เมื่อเวลาผ่านไปมีประสบการณ์มากขึ้นได้พบได้พัฒนาโปรแกรมใหม่ๆมากมายและกลับมามองย้อนอดีตจะทราบถึงความแตกต่างว่าโปรแกรมที่เขียนในอดีตนั้นไม่ดีเท่าที่ควร เห็นได้ว่าการแก้ปัญหาลักษณะงานหนึ่ง ถ้าให้โปรแกรมเมอร์หลายคนที่มีประสบการณ์แตกต่างกันแก้ปัญหา วิธีการของโปรแกรมเมอร์แต่ละคนในการแก้ปัญหาจะมีความแตกต่างกันออกไป บางคนเขียนโปรแกรมสั้น บางคนเขียนโปรแกรมยาว อัลกอริทึมที่ใช้ในการแก้ปัญหาของแต่ละบุคคลก็แตกต่างกันออกไป มีผลให้ประสิทธิภาพของการทำงานโปรแกรม มีความเร็วหรือช้าแตกต่างกัน ปัจจุบันเราสามารถชี้ได้ว่าโปรแกรมใดดีกว่าโปรแกรมใดในแง่ของความง่ายในการแก้ไขข้อผิดพลาดหรือการปรับปรุงเปลี่ยนแปลงโปรแกรม ความง่ายในการใช้งานของผู้ใช้ ความง่ายต่อการเข้าใจของโปรแกรมเมอร์ รวมถึงประสิทธิภาพที่วัดด้วยความรวดเร็วในการทำงาน

วิศวกรรมซอฟต์แวร์ เป็นวิชาที่พัฒนาขึ้นมาเพื่อนำมาใช้ในการออกแบบ และพัฒนาซอฟต์แวร์ให้มีคุณภาพสูง โดยหาหนทางหรือวิธีการในการแก้ปัญหาเพื่อทำให้ซอฟต์แวร์ที่พัฒนาขึ้นมีคุณลักษณะที่ดีตรงกับความต้องการของลูกค้า ผู้ใช้ระบบ และ ผู้พัฒนาระบบ การพัฒนาซอฟต์แวร์มักจะมีปัญหามากมายเกิดขึ้นในระหว่างขั้นตอนการพัฒนาซอฟต์แวร์ซึ่งมีผลให้การพัฒนาซอฟต์แวร์ล้มเหลว การเรียนวิชานี้เพื่อที่จะให้ผู้พัฒนาซอฟต์แวร์ได้ทราบถึงหลัก

การพื้นฐานทางวิศวกรรมในการพัฒนาระบบ รวมทั้งการนำเครื่องมือและเทคนิคต่างๆ มาใช้ในการพัฒนาซอฟต์แวร์

1.1

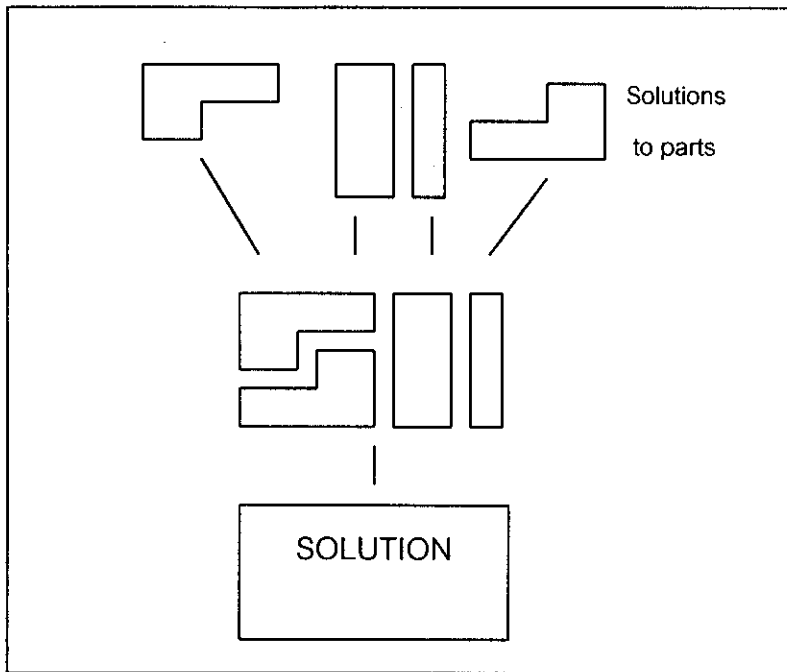
วิศวกรรมซอฟต์แวร์คืออะไร

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

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

การแก้ปัญหา

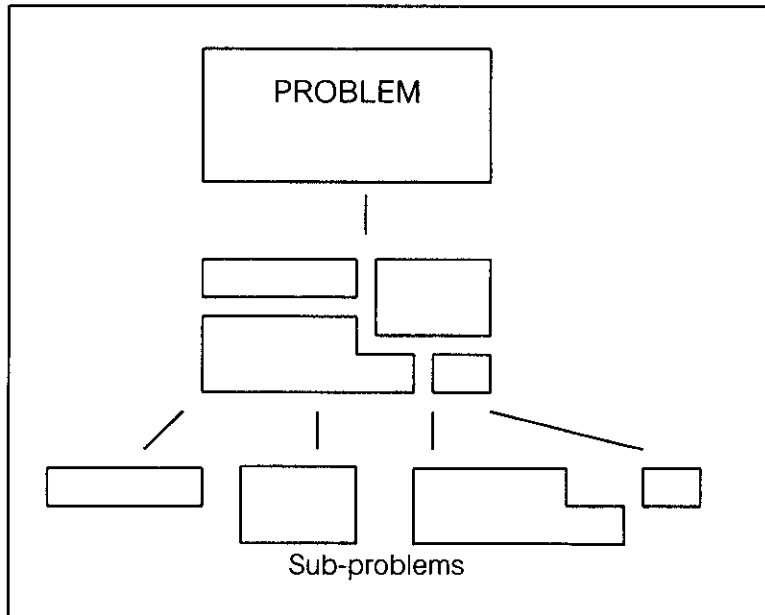
เริ่มจากการวิเคราะห์ปัญหาโดยแบ่งปัญหาหลัก ออกเป็นปัญหาย่อยๆ ทำให้ปัญหานั้นถูกแตกออกเป็นส่วนๆ โดยจะแตกออกไปเป็นส่วนย่อยๆ ให้อยู่ที่จุดเท่าที่จะเป็นไปได้ เราสามารถทำความเข้าใจกับปัญหาย่อยๆ และแก้ปัญหาในแต่ละส่วนย่อยๆ ได้ง่าย จากรูปภาพที่ 1.1



รูปภาพที่ 1.1 The Process of Analysis

analysis เป็นโปรเซสที่มีการแบ่งโครงสร้างขนาดใหญ่ให้เป็นส่วนย่อยๆ สำหรับรูปภาพที่ 1.2 แสดงส่วนกลับ: synthesis เป็นการนำปัญหาย่อยๆ มารวมเข้าด้วยกันเป็นโครงสร้างที่ใหญ่ ซึ่งเทคนิคในการแก้ปัญหาใดๆ นั้นต้องประกอบด้วยสองส่วนด้วยกัน คือ การวิเคราะห์ปัญหาเพื่อกำหนดธรรมชาติของปัญหาโดยพยายามแบ่งปัญหาออกเป็นส่วนย่อยๆ และสังเคราะห์ปัญหาเป็นการหาหนทางหรือวิธีการในการแก้ปัญหาจากส่วนย่อยๆ ซึ่งเป็นผลมาจากการวิเคราะห์นั่นเอง ในที่สุดเราก็สามารถแก้ปัญหาทั้งหมดได้

การหาวิธีการแก้ปัญหานั้นเราจำเป็นต้องหาเครื่องมือ(tools) และเทคนิค(techniques) ที่ช่วยให้เราสามารถแก้ปัญหาได้ง่ายขึ้น เพื่อผ่อนแรง และช่วยให้มีความถูกต้องมากที่สุด ทั้งยังเพิ่มประสิทธิภาพและผลิตผลให้มากขึ้น ซึ่งจะส่งผลให้ผลิตภัณฑ์ที่เราสร้างขึ้นมีคุณภาพดี ตัวอย่างที่ใกล้ตัวเรา อาทิเช่นในชีวิตประจำวันถ้าเราต้องการเขียนจดหมายให้กับลูกค้าหลายๆ คน ด้วยข้อความที่ซ้ำๆ กัน ในสมัยก่อนเราคงต้องเขียนด้วยลายมือถ้าลายมือสวยก็คงไม่เป็นปัญหาอะไร แต่ถ้าไม่สวยอ่านยากก็ไม่ใช่ที่ประทับใจกับลูกค้าเท่าไร ต่อมามีการใช้เครื่องพิมพ์ดีด



รูปภาพที่ 1.2 The Process of Synthesis

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

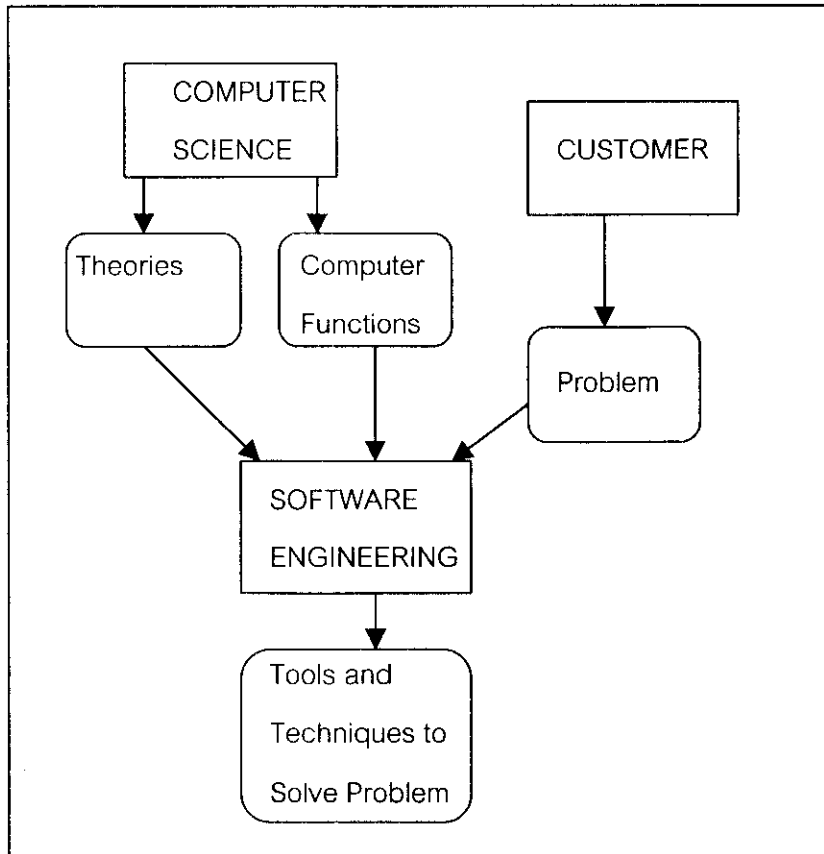
กัน แต่ความอร่อยของอาหารเกิดจากเทคนิคของการปรุงของแม่ครัวแต่ละท่านที่แตกต่างกัน ในแง่ของการพัฒนาโปรแกรม เทคนิคในการพัฒนาเป็นส่วนที่สำคัญในการแก้ปัญหาอีกประการหนึ่ง เทคนิคที่กล่าวถึงนี้คือขั้นตอนกระบวนการในการแก้ปัญหา (procedure or method) นั่นเอง การพัฒนาซอฟต์แวร์ที่มีคุณภาพต้องมีขั้นตอนในการแก้ปัญหาที่ดี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

รูปภาพที่ 1.3 แสดงความสัมพันธ์ระหว่างวิทยาการคอมพิวเตอร์และวิศวกรรมซอฟต์แวร์ ซึ่งวิทยาการคอมพิวเตอร์เกี่ยวข้องกับการออกแบบฮาร์ดแวร์และการนำทฤษฎีมาประยุกต์ทำงานตามขั้นตอนวิธีเพื่อแก้ปัญหาตามที่ลูกค้าต้องการ สำหรับวิศวกรรมคอมพิวเตอร์แก้ปัญหาโดยใช้เครื่องมือและเทคนิค

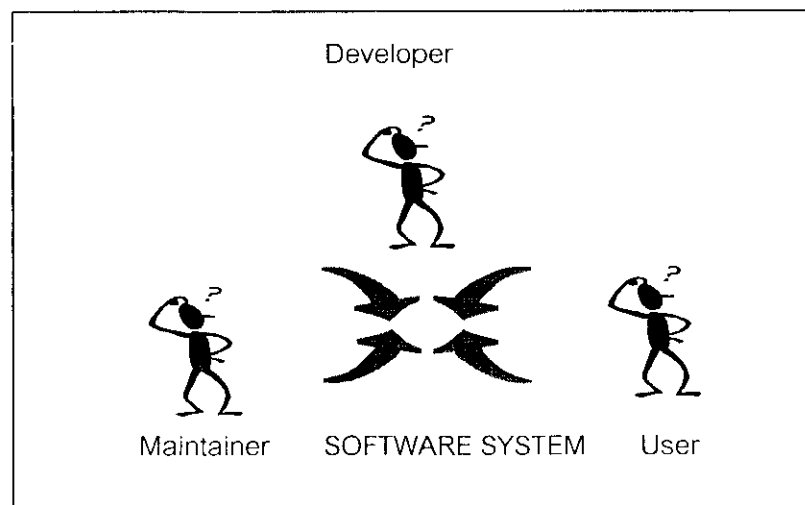
1.2

คุณภาพของซอฟต์แวร์

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



รูปภาพที่ 1.3 Relationship Between Computer Science and Software Engineering

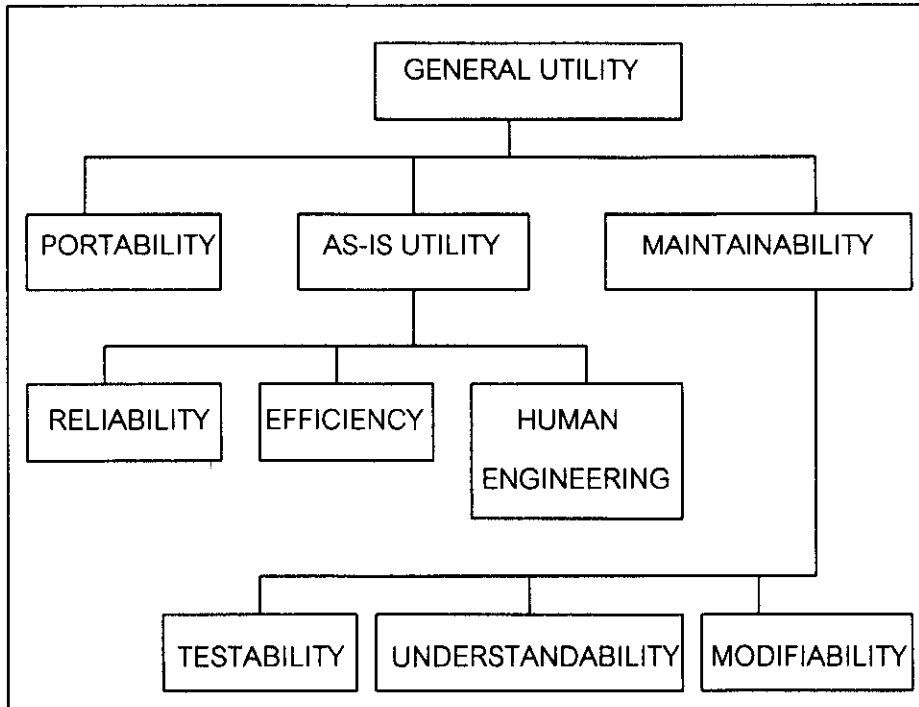


รูปภาพที่ 1.4 Judges of Software Quality

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

ผู้ใช้ทั้งสามกลุ่มมีความคาดหวังเหมือนกันว่า ระบบต้องมีความน่าเชื่อถือ และมีประสิทธิภาพ ความน่าเชื่อถือวัดได้จากระดับของความถูกต้องของผลลัพธ์จากการทำงานของระบบ ซอฟต์แวร์ที่มีคุณภาพต้องมีระดับของความถูกต้องอยู่ในเกณฑ์สูง นอกจากนี้ต้องใช้เวลาในการปฏิบัติงาน หรือการตอบสนองกับผู้ใช้อย่างรวดเร็วภายในเวลาที่ผู้ใช้ยอมรับได้รวมทั้งซอฟต์แวร์ต้องสามารถเข้าใจและเรียนรู้ได้ง่าย ระบบบางระบบมีความน่าเชื่อถือและมีประสิทธิภาพที่ดีเยี่ยมแต่ผู้ใช้ไม่สามารถเข้าใจได้ง่าย ใช้งานยาก ก็มีผลให้ซอฟต์แวร์นี้ไม่เป็นที่ยอมรับ เราสามารถสรุปได้ว่าซอฟต์แวร์ที่มีคุณภาพต้องมีคุณลักษณะดังนี้

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



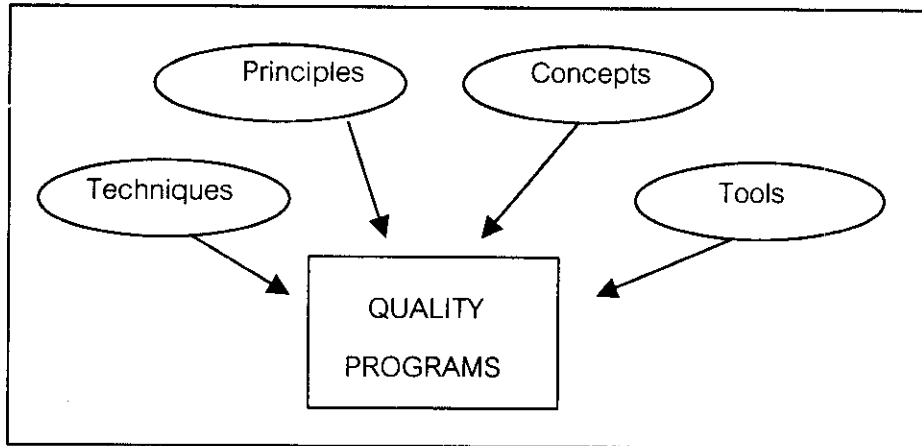
รูปภาพที่ 1.5 Characteristics of software

การผลิตซอฟต์แวร์ที่มีคุณภาพนั้น ถึงแม้ว่าเราจะทราบถึงหลักการในการสร้างหรือมีความเชี่ยวชาญภาษาโปรแกรมที่ใช้ในการพัฒนาซอฟต์แวร์เป็นอย่างดีก็ยังไม่เพียงพอ เรายังต้องมีความรู้ในเรื่องอื่นๆอีก ตามรูปภาพที่ 1.6 แสดงให้เห็นว่าเราต้องเรียนรู้ถึงแนวความคิด เทคนิค และเครื่องมือต่างๆที่ช่วยในการออกแบบและพัฒนาโปรแกรม เราจึงสามารถสร้างโปรแกรมที่มีคุณภาพได้

งานและความรับผิดชอบ

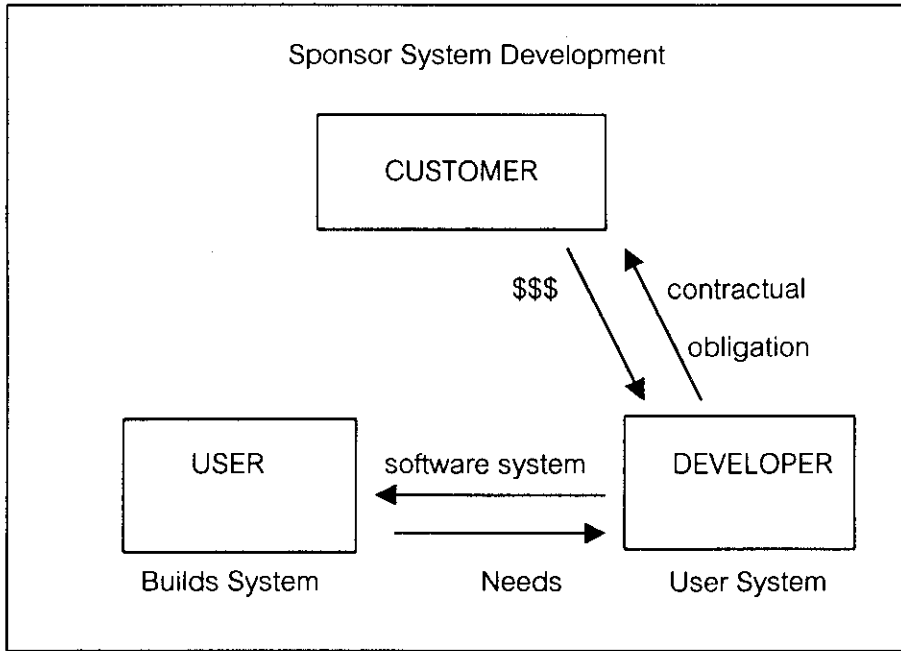
การพัฒนาโครงการซอฟต์แวร์เริ่มจากการติดต่อสื่อสารระหว่างลูกค้าและผู้พัฒนาซอฟต์แวร์ โดยทำความเข้าใจถึงความต้องการและความจำเป็นในการสร้างระบบให้ต้องแท้เสียก่อน จากการสอบถาม รวบรวมข้อมูล สังเกต ออกแบบสอบถาม เพื่อทำความเข้าใจกับบุคคลต่างๆที่เกี่ยวข้องกับระบบทั้งหมด ก่อนที่จะสร้างระบบขึ้นมา

จำนวนของบุคคลในทีมงานพัฒนาซอฟต์แวร์จะขึ้นกับขนาดและระดับความยากง่ายของโครงการ ซึ่งหน้าที่และความรับผิดชอบของแต่ละบุคคลจะแตกต่างกันออกไปขึ้นอยู่กับขนาดของโครงการ ถ้าเป็นโครงการขนาดใหญ่เราอาจได้รับมอบหมายให้ทำเฉพาะงาน แต่ถ้าเป็นโครงการขนาดเล็กเราอาจได้ทำในหลายๆหน้าที่ โดยทั่วไปแล้วผู้ที่เกี่ยวพันในโครงการทั้งหมดจะแบ่งเป็น 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 Relationships among Customer, User, and Developer

1.3

ระบบ

โครงการที่พัฒนาขึ้นมา โดยทั่วไปจะมีการติดต่อกับผู้ใช้ผ่านอุปกรณ์ฮาร์ดแวร์ โดยรับข้อมูลจากภายนอกระบบมาประมวลผลกับฐานข้อมูลที่มีอยู่ หรือเป็นการติดต่อสื่อสารข้อมูลระหว่างระบบคอมพิวเตอร์อื่นๆ ดังนั้นการรับข้อมูลต่างๆเข้ามาเพื่อการทำงานเป็นสิ่งสำคัญอย่างยิ่ง เราต้องทราบว่าคุณสมบัติที่รับเข้ามาเป็นอย่างไร มีขอบเขต(boundary) เป็นอย่างไร ข้อมูลที่ถูกต้องเป็นอย่างไร ข้อมูลที่ผิดพลาดเป็นอย่างไร สมมุติว่าเราได้รับมอบหมายให้เขียนโปรแกรม

พิมพ์สลิปเงินเดือนของพนักงานในบริษัท สิ่งที่โปรแกรมต้องกระทำได้คือสามารถรับจำนวนชั่วโมงทำงานของพนักงานทุกคนได้ คำนวณเงินเดือนที่ได้รับจากอัตราค่าจ้างตามชั่วโมงที่ทำงานจริง คำนวณภาษีที่ต้องจ่าย คำนวณรายรับและรายจ่ายอื่นๆ กล่าวคือเราต้องทราบถึงการดำเนินงานหรือกิจกรรมที่เกี่ยวข้องทั้งหมดที่มีอยู่ในระบบทั้งหมด และผลิตเป็นรายงานเงินเดือนของพนักงานแต่ละคนซึ่งเป็นผลลัพธ์จากการทำงานของระบบนี้

สมาชิกของระบบ

ระบบหนึ่งๆ ประกอบด้วยกิจกรรม(activities)ต่างๆมากมาย กิจกรรมหมายถึงเหตุการณ์ต่างๆที่เกิดขึ้นในระบบ ซึ่งอาจเป็นกิจกรรมการเคลื่อนย้ายข้อมูล(move)จากที่หนึ่งไปอีกที่หนึ่ง เช่น ข้อมูล A ถูกย้ายจากแฟ้มข้อมูลหนึ่งไปยังแฟ้มข้อมูลอีกแฟ้มหนึ่ง กิจกรรมการเปลี่ยนแปลงค่าหรือคุณลักษณะของข้อมูล เช่นการคำนวณค่าของข้อมูลทำให้ข้อมูลเปลี่ยนแปลงไปจากเดิมหรือกิจกรรมการรวมข้อมูล เป็นต้น

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

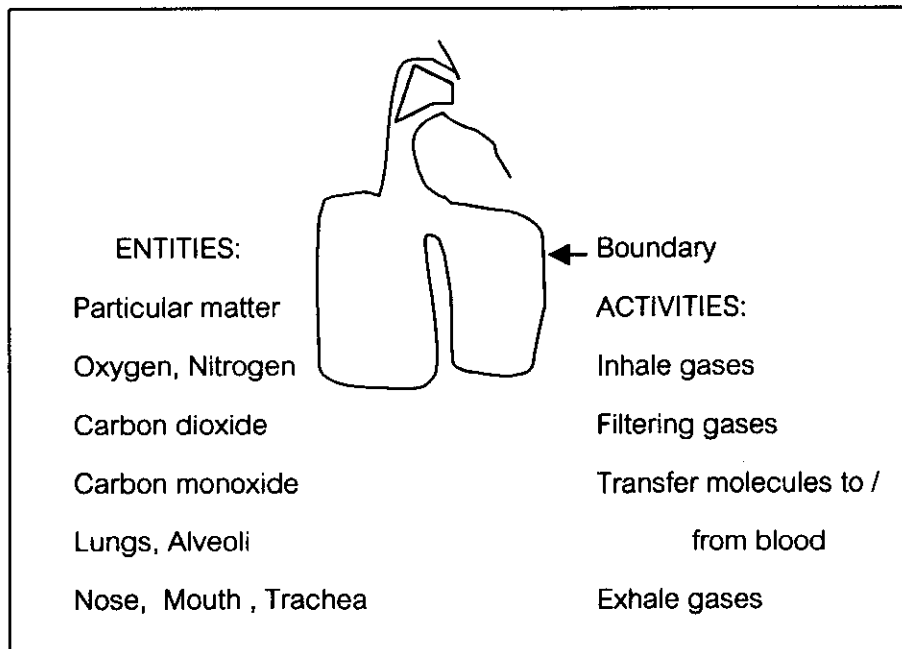
First name	Zip code
Middle name	Salary per hour
Last name	Benefits per hour
Street address	Vacation hours accrued
City	Sick leave accrued
State	

นอกจากนี้ ขนาด ความสัมพันธ์ ชนิดของข้อมูลของแต่ละฟิลด์ ตำแหน่งเริ่มต้นของระเบียน ความยาวของฟิลด์ รวมทั้งการรวมหลายๆระเบียนเป็นแฟ้มข้อมูลและคุณลักษณะของแฟ้มก็ถือว่าเป็นเอ็นติตี้ทั้งสิ้น

เมื่อเอนิตี และกิจกรรมถูกกำหนดขึ้น เราต้องทราบว่เอนิตีนั้นๆมีความสัมพันธ์หรือเกี่ยวข้องกับกิจกรรมใดบ้าง เอนิตีที่อยู่ใต้ใดบ้าง มีอยู่ในแฟ้มเรียบร้อยแล้ว หรือต้องสร้างใหม่ในระหว่างดำเนินกิจกรรม บางเอนิตีที่ถูกใช้งานในหลายๆกิจกรรม เราต้องทราบด้วยว่าการใช้เอนิตีต่างๆนั้นอยู่ในขอบเขต(boundary)ของระบบหรือไม่

ดังนั้นแนวความคิดในการกำหนดระบบนั้นคือการรวบรวมกลุ่มของเอนิตี กลุ่มของกิจกรรม รายละเอียดของความสัมพันธ์ระหว่างเอนิตีและกิจกรรม รวมทั้งกำหนดขอบเขตของระบบ นอกจากนี้ยังรวมถึงทุกๆสิ่งที่มีการเกี่ยวข้องกับเอนิตีอีกด้วย

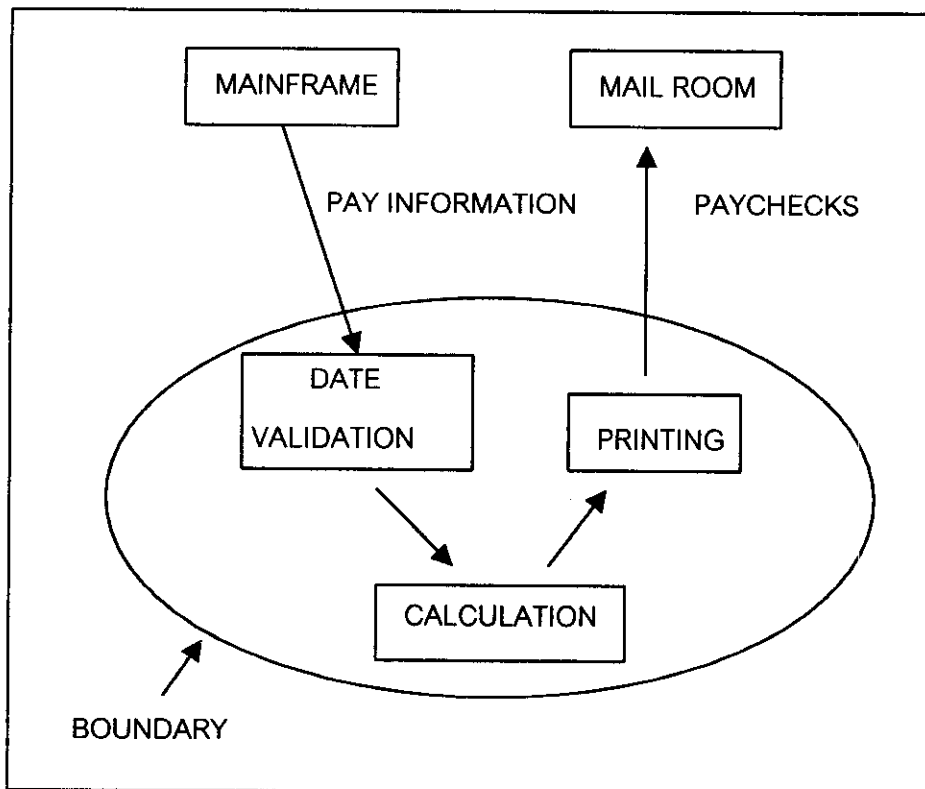
ตัวอย่างระบบการหายใจของมนุษย์ เราสามารถกำหนดขอบเขตเฉพาะในร่างกายเท่านั้น เอนิตีคืออวัยวะต่างๆภายในร่างกายที่เกี่ยวข้องกับการหายใจเช่น จมูก หลอดลม ก๊าช ออกซิเจน ก๊าชคาร์บอนไดออกไซด์ ปอด และอื่นๆ สำหรับกิจกรรมในระบบเป็นการกระทำระหว่างเอนิตีต่างๆเช่น กิจกรรมการหายใจเข้า กิจกรรมหายใจออก และอื่นๆ รูปภาพที่ 1.8 แสดงระบบการหายใจที่ประกอบด้วยกิจกรรมและเอนิตีต่างๆ ซึ่งมีการรับเอนิตีที่เข้าสู่ระบบ นำเอนิตีต่างๆที่เกี่ยวข้องมาประมวลผล แลผลิตเป็นเอาต์พุต ดังภาพ



รูปภาพที่ 1.8 Respiratory

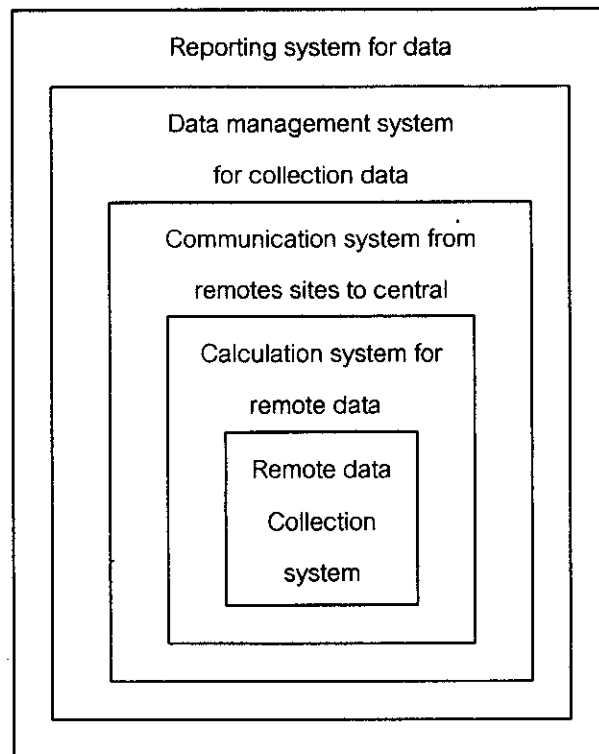
สำหรับการพัฒนาระบบขึ้นมา นั้น เราต้องทราบถึงความสัมพันธ์ของสมาชิกภายในระบบอย่างชัดเจน ในแง่ของขอบเขตของระบบ ความสามารถของระบบตามที่เราใช้ต้องการ การปฏิบัติการของระบบ จุดเริ่มต้นของระบบ การติดต่อกับระบบภายนอก รวมทั้งผลลัพธ์ซึ่งเป็นจุดสิ้นสุดของระบบ

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



รูปภาพที่ 1.9 Paycheck System

เมื่อก้าวถึงระบบ เกือบทุกระบบจะต้องมีความเกี่ยวข้องกับระบบอื่นๆ จะเป็นระบบที่ทำงานเพียงลำพังนั้นมีน้อยมากจะต้องมีความเกี่ยวข้องกับระบบอื่นๆที่อยู่รอบๆ จากตัวอย่างที่กล่าวข้างต้น ระบบหายใจของมนุษย์ ต้องเกี่ยวข้องกับระบบอื่นๆซึ่งทำงานอยู่ภายในร่างกายมนุษย์ เช่นระบบย่อยอาหาร ระบบหมุนเวียน ระบบประสาท และอื่นๆอีกมากมาย ซึ่งมนุษย์เราถ้าขาดระบบใดระบบหนึ่งไม่ได้เพราะทุกระบบทำงานสัมพันธ์กัน



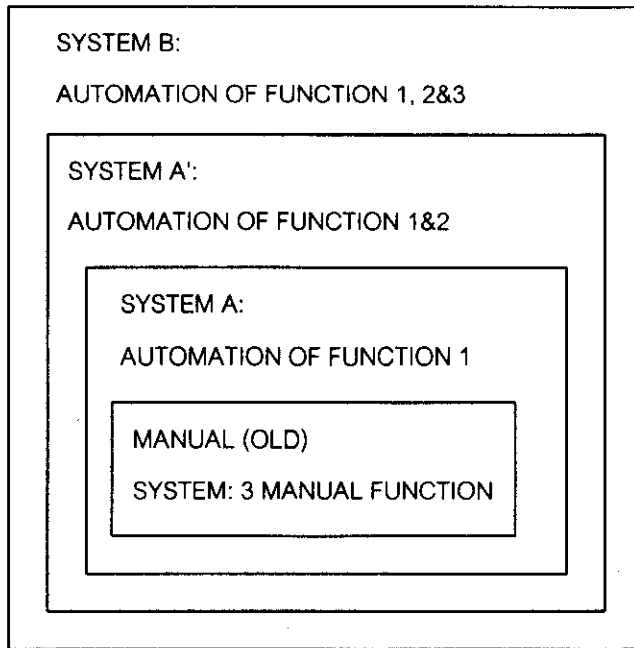
รูปภาพที่ 1.10 Layers of Water Monitoring System

การพัฒนาระบบใดๆ เราต้องเข้าใจถึงความสัมพันธ์ต่างๆภายในระบบอย่างดี ตัวอย่างถ้าเราได้รับมอบหมายให้พัฒนาระบบการตรวจวัดปริมาณของน้ำในแม่น้ำแห่งหนึ่ง เราต้องทราบว่าการตรวจวัดนั้นใช้อินพุตอะไร มาจากไหน มีวิธีการประมวลผลอย่างไร ขั้นตอนการทำงานเป็นอย่างไร ผลลัพธ์ที่ต้องการเป็นเช่นใด มีการส่งผ่านข้อมูลไปยังที่ใด อย่งไร เพื่อที่จะได้ทราบถึงเอ็นดีดี กิจกรรม ความสัมพันธ์ และขอบเขตทั้งหมดของระบบ ในที่นี้สมมุติว่าระบบ

นี้มีอินพุตจากหลายๆแหล่ง โดยการวัดปริมาณน้ำจะไม่วัดที่จุดๆเดียว ข้อมูลจะถูกรวบรวมมาจาก หลายๆจุดในแม่น้ำสายนี้ที่มีสาขาตั้งอยู่ โดยในระบบมีคอมพิวเตอร์ศูนย์กลางอยู่ในที่แห่งหนึ่งและมีการส่งผ่านข้อมูลมาจากคอมพิวเตอร์สาขาที่อยู่ในที่ๆห่างไกลออกไป คอมพิวเตอร์สาขาสามารถประมวลผลปริมาณน้ำ ณ ที่ตั้งของตนเองและนำส่งข้อมูลมายังคอมพิวเตอร์ศูนย์กลางได้ ดังนั้นคอมพิวเตอร์ศูนย์กลางจะทำหน้าที่รับข้อมูลจากคอมพิวเตอร์สาขาเพื่อรวบรวมและเก็บข้อมูลลงในฐานข้อมูลส่วนกลาง พร้อมกับสร้างรายงานจากฐานข้อมูลที่มีอยู่ตามความต้องการของผู้ใช้ เนื่องจากระบบนี้เป็นระบบที่เกี่ยวข้องกับการติดต่อสื่อสารมีการทำงานหลายขั้นตอน วิธีการที่ใช้นั้นเราจำเป็นต้องแบ่งกิจกรรมออกเป็นส่วนย่อยๆ ถ้าเรากำหนดขอบเขตรายละเอียดของระบบได้อย่างถูกต้อง การสร้างระบบขนาดใหญ่จากระบบย่อยๆนำมาประกอบกัน จะทำให้การพัฒนากระบวนการง่ายขึ้น รูปภาพที่ 1.10 แสดงถึงการแบ่งระดับของ Water Monitoring System ซึ่งในแต่ละระดับระบุหน้าที่การทำงานของระบบอย่างชัดเจน โดยสี่เหลี่ยมแต่ละอันจะแทนขอบเขตของการทำงานของระบบนั้น ในที่นี้สี่เหลี่ยมที่อยู่ในสุดจะเป็นระบบที่เล็กที่สุด ออฟเจ็ทหรือกิจกรรมของส่วนนี้มีความสัมพันธ์กับสี่เหลี่ยมในระดับนอก และมีความสัมพันธ์กันเกี่ยวโยงกันเป็นลำดับ จนได้ผลลัพธ์ที่ต้องการ

โดยทั่วไปแล้วความคิดในการสร้างระบบใหม่ขึ้นมาขึ้นมานั้นเนื่องจากว่าเกิดปัญหาในการทำงานของระบบเก่าเช่นความล่าช้า การผิดพลาดของการทำงาน หรือต้องการเพิ่มประสิทธิภาพของการทำงานนั้นดีขึ้น ดังนั้นระบบใหม่มักจะใช้แทนที่ระบบเก่าซึ่งมีการทำงานด้วยแรงงานมนุษย์ หรืออาจเป็นเครื่องจักรอัตโนมัติรุ่นเก่าที่ล้าสมัย การพัฒนาระบบบางครั้งผู้ใช้ระบบอาจเป็นปัญหาที่สำคัญ มีผู้ใช้เป็นจำนวนมากที่ไม่มีความรู้ทางด้านคอมพิวเตอร์ ต่อด้านไม่ยอมรับระบบใหม่ ดังนั้นเราต้องเข้าใจถึงความแตกต่างในการทำงานของระบบเก่าและระบบใหม่ นำมาออกแบบและพัฒนาในลักษณะค่อยเป็นค่อยไป โดยค่อยๆเปลี่ยนแปลงระบบทีละส่วน ทีละส่วน จนสมบูรณ์ จากรูปที่ 1.11 แสดงการเปลี่ยนระบบ A ไปเป็นระบบ B ซึ่งการพัฒนาจะค่อยๆเปลี่ยนแปลงจาก A เป็น A' และ เป็น B ตามลำดับ

การพัฒนาระบบนั้นเราอาจเพิ่มระยะในการพัฒนา เพื่อให้ระบบเก่าค่อยๆเปลี่ยนแปลง อาจเป็นการเปลี่ยนแปลงทางด้านฮาร์ดแวร์หรือซอฟต์แวร์ทีละน้อยๆ เช่นในระยะแรกเพิ่มฮาร์ดแวร์ใหม่บางชิ้นเข้าไปเพื่อให้ผู้ใช้คุ้นเคยเสียก่อน ระยะที่สองมีการแทนซอฟต์แวร์แทนการทำงานเดิมที่กระทำด้วยแรงงานคน ระยะต่อไปก็ค่อยๆเพิ่มโน้นเพิ่มนี้ไป จนกระทั่งเป็นระบบใหม่ที่ได้



รูปภาพที่ 1.11 Incremental Steps from Old System to New System

ออกแบบไว้ การพัฒนาในลักษณะนี้เป็นการปรับเปลี่ยนที่ค่อยเป็นค่อยไป ผู้พัฒนาต้องมองภาพของการพัฒนาระบบประกอบด้วยสองลักษณะคือ สถิตย (Statically) และ จลย(Dynamically) กล่าวคือ สถิตย เราจะมองในแง่ของระบบทำงานประจำในแต่ละวันได้อย่างไรเพราะการปรับเปลี่ยนจะต้องสามารถทำงานประจำได้ตามปกติ ในขณะที่ จลย เรามองภาพของระบบที่ต้องปรับเปลี่ยนจากขั้นตอนหนึ่งไปยังขั้นตอนต่อไป จนกระทั่งในที่สุดได้ระบบที่สมบูรณ์ ซึ่งผู้พัฒนาต้องทราบว่าในขั้นตอนไหนปรับตรงส่วนใดและจะเชื่อมต่อการทำงานปกติได้อย่างไร

วิศวกรรม

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

การสร้างบ้าน

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

เปลี่ยนแปลงแบบหรืออุปกรณ์ต่างๆได้อีก จนกระทั่งเสร็จสมบูรณ์ตามความต้องการ ซึ่งสามารถสรุปขั้นตอนของการสร้างบ้านของผู้รับเหมาโดยทั่วไปได้ดังนี้

- (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 เ็นิตติ์ และกิจกรรมต่างๆที่มีการกระทำภายในระบบและภายนอกระบบ ต่อจากนั้นจะนำความต้องการทั้งหมดมาออกแบบระบบ(system design) เป็นภาพรวมของระบบที่สามารถกระทำได้นำการออกแบบระบบไปตรวจสอบกับลูกค้าเพื่อความเข้าใจที่ตรงกัน ถ้าลูกค้ายอมรับ การออกแบบระบบนี้จะนำไปออกแบบโปรแกรม(program design)ซึ่งเป็นรายละเอียดอาจเป็นหน้าที่หรือเหตุการณ์ที่เกิดขึ้นในระบบ ต่อจากนั้นเขียนโปรแกรม (program) เพื่อให้สามารถทำงานตามที่ต้องการได้ พร้อมทั้งทดสอบในแต่ละส่วนของโปรแกรม (unit testing) เพื่อให้สามารถทำงานได้อย่างถูกต้อง และนำแต่ละส่วนของโปรแกรมที่ทดสอบมารวมกันเป็นการทดสอบรวม (integration testing) การทดสอบในขั้นตอนสุดท้ายเรียกว่าทดสอบระบบ(system testing) ซึ่งทดสอบว่าระบบสามารถทำงานตามหน้าที่และการโต้ตอบได้ตามที่ต้องการหรือไม่ เมื่อระบบสามารถกระทำได้ตามที่ต้องการท้ายสุดของการสร้างระบบคือ การส่งมอบระบบ (delivery) เพื่อให้ผู้ใช้ทำงานแทนที่ระบบเดิม หลังจากนั้นเป็นขั้นตอนการบำรุง

รักษาระบบ(maintenance) 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 เราต้องเพิ่มความ ต้องการเข้าไป บางครั้งต้องออกแบบระบบอีกครั้งหนึ่งเพื่อให้ครอบคลุมความต้องการทั้งหมด ซึ่งการปรับเปลี่ยนในบางกรณีมีผลให้มีการเขียนโปรแกรมใหม่เพิ่ม การทดสอบที่เพิ่มขึ้น รูปภาพที่ 1.12 แสดงให้เห็นถึง วัฏจักรและความเป็นไปได้ในการพัฒนาระบบซ้ำในขั้นตอนต่างๆ

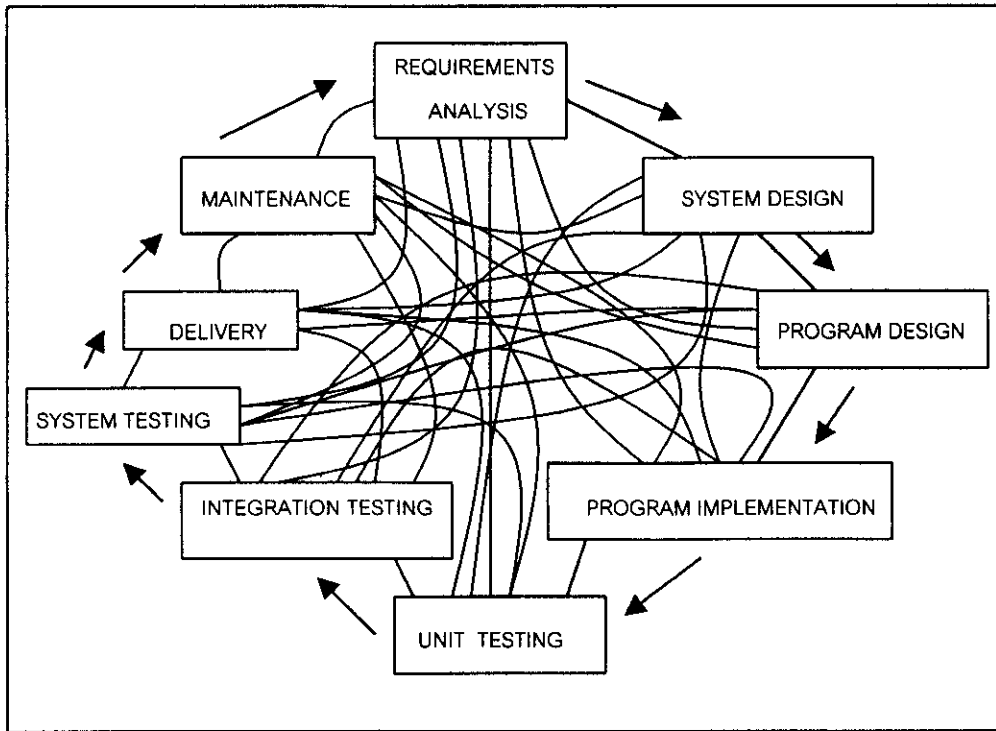
สรุปจุดประสงค์ของการเรียนวิชานี้เพื่อให้ทราบถึงขั้นตอนในการพัฒนาซอฟต์แวร์ รวมถึงกฎเกณฑ์ เครื่องมือ และ วิธีการ ที่จะช่วยให้ระบบซอฟต์แวร์มีคุณภาพ

ทีมงานในการพัฒนาระบบ

ผู้พัฒนาระบบ(developer) ทำหน้าที่สร้างระบบโดยทำงานเป็นทีมงานประกอบด้วยบุคคลต่างๆ ที่มีความเชี่ยวชาญที่แตกต่างกัน ในขั้นตอนแรกของการพัฒนาต้องมีการพบปะพูดคุยกับลูกค้า และผู้ใช้ระบบ เพื่อวิเคราะห์และกำหนดระบบ โดยทำความเข้าใจถึงหน้าที่ทั้งหมดที่ระบบ กระทำได้ เขียนเป็นเอกสารระบุความต้องการเพื่อใช้ในการอ้างอิงในการพัฒนาระบบในขั้นตอนต่อไป ผู้ที่ทำหน้าที่นี้เรียกว่านักวิเคราะห์ระบบ(analyst) ต่อจากนั้นนักวิเคราะห์ระบบจะทำงานรวมกันผู้ออกแบบระบบ(designers) เพื่อสร้างรายละเอียดในระดับย่อยๆที่ระบบ

สามารถกระทำได้

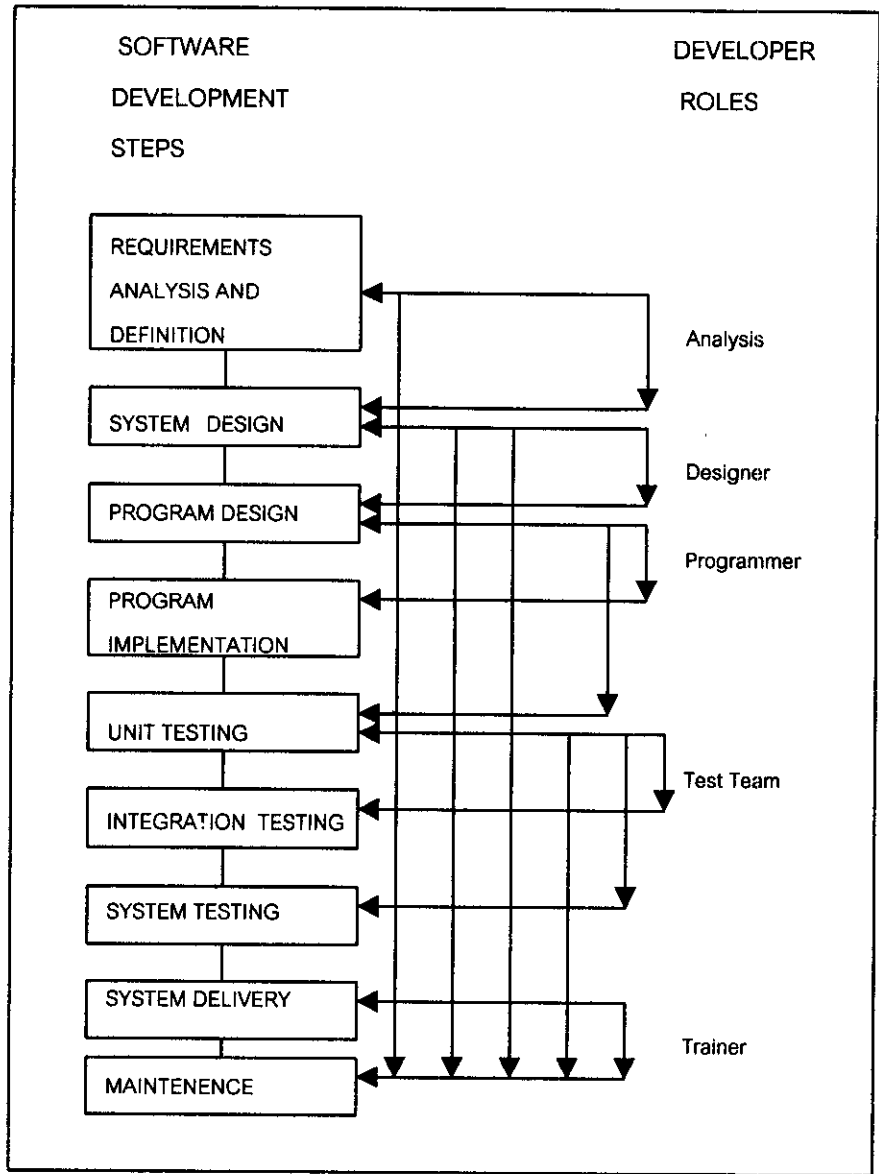
ต่อจากนั้นผู้ออกแบบระบบจะทำงานร่วมกับโปรแกรมเมอร์ (programmers) เพื่อให้ผู้เขียนโปรแกรมสามารถเขียนคำสั่งโปรแกรมตามความต้องการได้อย่าง



รูปภาพที่ 1.12 The Development Cycle

ถูกต้อง หลังจากนั้นเป็นหน้าที่ของผู้ทดสอบระบบ (testers) ทำหน้าที่หาข้อผิดพลาดที่อาจเกิดขึ้นในโปรแกรม การทำงานของผู้ทดสอบระบบจะทำงานร่วมกับลูกค้าเพื่อพิสูจน์ว่าระบบสามารถทำงานตามที่ลูกค้าต้องการได้ เมื่อลูกค้ายอมรับจะมีการฝึกฝนการใช้โปรแกรมให้กับผู้ใช้ระบบ โดยผู้ฝึก (trainers) ถึงแม้จะมีการส่งมอบระบบให้ลูกค้าแล้ว ไม่ใช่ว่างงานในการพัฒนาจะจบลง ถ้ามีข้อผิดพลาดเกิดขึ้นในระบบหรือความต้องการของระบบเปลี่ยนไป ทีมงานบำรุงรักษา (maintenance team) จะเป็นกลุ่มที่ทำหน้าที่รับผิดชอบแก้ไข ปรับปรุง เปลี่ยนแปลงตามความต้องการซึ่งอาจเป็นการแก้ไขการออกแบบ แก้ไขคำสั่งโปรแกรม มีการทดสอบระบบใหม่ และเมื่อมีการปรับเปลี่ยนเกิดขึ้นผู้ฝึกจะต้องสอนหน้าที่ใหม่ๆ ที่เพิ่มเติมขึ้นมาให้กับผู้ใช้

ระบบเพื่อให้สามารถทำงานได้ตามปกติ รูปภาพที่ 1.13 แสดงถึงสมาชิกของทีมงานในการพัฒนาที่เกี่ยวข้องในขั้นตอนต่างๆของวัฏจักรในการพัฒนาซอฟต์แวร์



รูปภาพที่ 1.13 Roles of the Development Team

สำหรับระบบที่มีขนาดใหญ่และซับซ้อนมากๆ จำนวนเอกสารในระบบก็จะมีมากตามไปด้วย ความสามารถในการบำรุงรักษาระบบนั้นเป็นไปได้ยาก ในทีมงานอาจมี บรรณารักษ์ (Librarians) เป็นบุคคลที่ช่วยสนับสนุนทีมงานในการพัฒนาระบบ โดยทำหน้าที่เตรียมและเก็บเอกสารทั้งหมดที่ใช้ในระหว่างการพัฒนากระบวนการรวมทั้งรายละเอียดความต้องการ รายละเอียดของการออกแบบทั้งหมด เอกสารโปรแกรม คู่มือผู้ใช้ แผนการทดสอบและอื่นๆที่เกี่ยวข้องกับเอกสารทั้งหมด นอกจากนี้บุคคลที่ทำหน้าที่ร่วมกับบรรณารักษ์ โดยทำหน้าที่ในการหาเอกสารอ้างอิงหรือ หาดำแหน่งของโปรแกรมคำสั่งที่มีการเปลี่ยนแปลงตามความต้องการใหม่รวมทั้งส่วนของโปรแกรมที่มีผลกระทบกับการแก้ไขเปลี่ยนแปลง เพื่อช่วยผู้พัฒนาในการบำรุงรักษาระบบเรียกว่า configuration management team โดยทั่วไปผู้บำรุงรักษาระบบจะเป็นทีมงานคนละทีมกับผู้ออกแบบและเขียนโปรแกรม และจำนวนของบุคคลในการพัฒนาขึ้นอยู่กับขนาดของโครงการ ถ้าโครงการขนาดเล็กอาจมีผู้พัฒนาเพียงสองหรือสามคนและช่วยกันทำงานในหลายๆหน้าที่ แต่ถ้าเป็นโครงการขนาดใหญ่ทีมงานมีหลายคนหัวหน้าโครงการมีการจัดสรรหน้าที่โดยแยกเป็นกลุ่มๆในการทำงานและมีหน้าที่แตกต่างกันออกไป ขึ้นอยู่กับความชำนาญและประสบการณ์ของแต่ละบุคคล

1.5

ปัญหาในการพัฒนาซอฟต์แวร์

ในตอนต้นนั้นเราได้เปรียบเทียบการพัฒนาซอฟต์แวร์เหมือนกับการสร้างบ้าน บ้านที่เราหวังและตั้งใจปลูกตามความต้องการแต่เมื่อเข้าไปอยู่บางอย่างก็ถูกใจบางอย่างก็ไม่ชอบอยากปรับปรุงเปลี่ยนแปลง บางอย่างไม่เป็นไปตามความต้องการ เหมือนกับการพัฒนาซอฟต์แวร์เมื่อพัฒนาเสร็จลูกค้าอาจไม่พอใจในหลายๆสิ่ง เช่น ไม่เป็นไปตามที่ต้องการ หรือมีขั้นตอนในการพัฒนาที่ยุ่งยากหรือคุณภาพไม่ดี เราสามารถสรุปปัญหาในการพัฒนาซอฟต์แวร์ได้ดังนี้

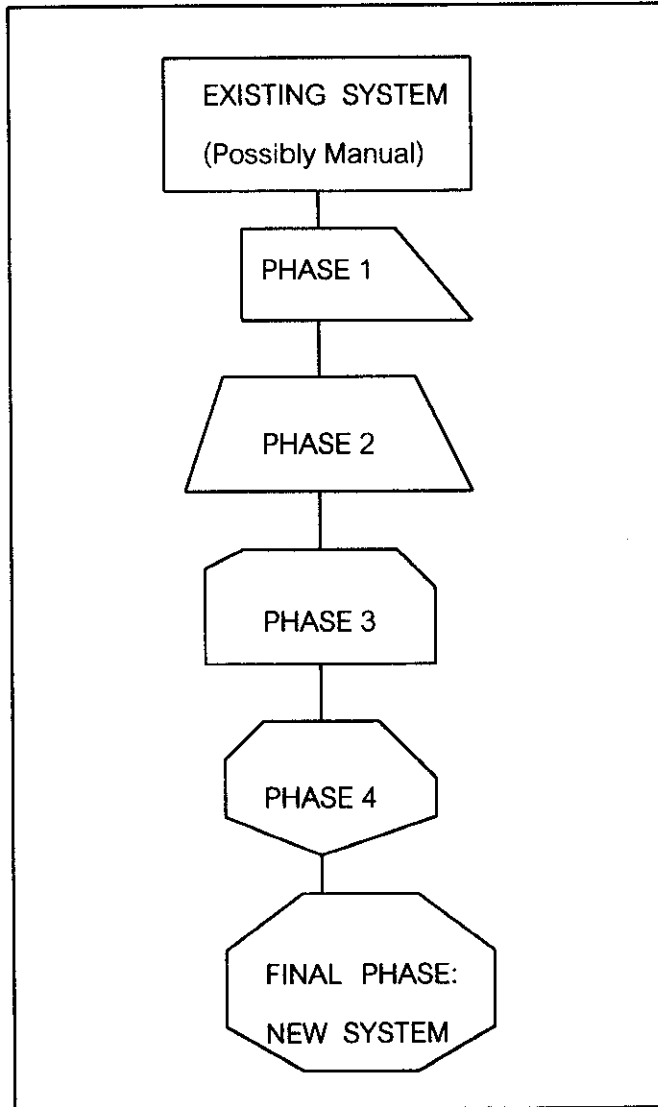
(1) Changing Constraints and Requirement

ในระยะของการพัฒนาซอฟต์แวร์นั้นเรามีการพบปะพูดคุยกับลูกค้าอย่างสม่ำเสมอ โดยนำเสนอความคืบหน้าในการทำงานทุกๆระยะเพื่อสร้างระบบตรงกับความต้องการ บางครั้งลูกค้ามีความต้องการไม่คงที่มีการปรับเปลี่ยนบ่อยครั้ง ในบางกรณีก็ก่อให้เกิดปัญหาในการพัฒนาซอฟต์แวร์ได้เช่นเดียวกัน เช่น ผู้พัฒนาตัดสินใจเลือกฮาร์ดแวร์และซอฟต์แวร์ระบบหนึ่งเพื่อใช้ในโครงการ แต่เมื่อความต้องการของลูกค้าเปลี่ยนไป ความต้องการของลูกค้านั้นทำได้ยากในการที่จะกระทำจากฮาร์ดแวร์และซอฟต์แวร์ที่มีอยู่ ผู้พัฒนาต้องเลือกใช้ฮาร์ดแวร์และซอฟต์แวร์ใหม่ที่สามารถกระทำได้ตามความต้องการได้ ดังนั้นข้อเสนอแนะประการหนึ่งก็คือในการเลือกเครื่องมือต้องเลือกเครื่องมือที่ใช้ในการพัฒนาซอฟต์แวร์ให้สามารถมีความยืดหยุ่นในหลายๆกรณีจะทำให้ปัญหาในการพัฒนานั้นลดน้อยลง

(2) Phased Development System

การพัฒนาระบบที่มีขนาดใหญ่และซับซ้อนมากๆ เราต้องค่อยๆแบ่งการพัฒนาเป็นระยะๆ เพื่อให้ง่ายต่อการสร้างและพัฒนา โดยเริ่มจากระบบเล็กๆที่มีความซับซ้อนน้อยๆก่อน โดยระบบที่สร้างขึ้นใหม่จะแทนที่ระบบเก่าโดยค่อยๆเพิ่มความซับซ้อนขึ้นเป็นลำดับ อย่างเช่นในระยะแรกเราสร้างระบบโดยปรับเปลี่ยนจากระบบเดิมในบางส่วน นำระบบใหม่ที่สร้างขึ้นแทนที่ระบบเก่าค่อยๆให้ผู้ใช้ระบบคุ้นเคย ต่อจากนั้นเริ่มสร้างระบบในระยะที่สอง เพิ่มความต้องการของลูกค้ามากกว่าระยะแรกโดยสร้างและปรับเปลี่ยนมากขึ้นและนำไปรวมกับระบบที่พัฒนาในระยะแรก กระทำในลักษณะนี้จนกระทั่งระบบสมบูรณ์ วิธีการที่เรามีการสร้างและปรับเปลี่ยนระบบเก่าโดยนำส่วนที่สร้างผนวก(merge)กับระบบที่กำลังทำงานอยู่นี้เราเรียกว่า phased development

รูปภาพที่ 1.14 แสดงถึงระยะในการพัฒนาซึ่งเริ่มจากระบบที่กำลังดำเนินงานอยู่ในปัจจุบัน ในระยะที่หนึ่งเราเริ่มสร้างระบบใหม่โดยมีการปรับเปลี่ยนการทำงานในระบบเดิมบางส่วนและติดตั้งระบบให้สามารถทำงานได้ตามปกติ ในระยะที่สองมีการเพิ่มความต้องการมากขึ้นปรับเปลี่ยนระบบและนำไปแทนที่ระบบที่กำลังดำเนินงาน การปรับเปลี่ยนต่างๆจะกระทำเป็นระยะๆและนำสิ่งที่พัฒนาขึ้นใหม่ไปผนวกกับระบบเก่า จนกระทั่งระยะสุดท้ายเป็นระบบที่สมบูรณ์ตรงกับความต้องการของลูกค้าทุกประการ



รูปภาพที่ 1.14 Phases Development

สมมติเราได้รับมอบหมายให้พัฒนาระบบการควบคุมการสับเปลี่ยนสัญญาณโทรศัพท์
 ในบริษัทแห่งหนึ่ง ในการพัฒนาเราไม่สามารถทดสอบหรือปรับเปลี่ยนระบบได้โดยทันทีที่ใด
 เนื่องจากลูกค้าไม่ต้องการให้ทั้งระบบต้องขาดการติดต่อสื่อสาร ต้องพยายามให้ส่งผลกระทบต่อ
 การการทำงานเดิมให้น้อยที่สุด การพัฒนาระบบนี้จึงต้องแบ่งลำดับขั้นตอนการพัฒนาออกเป็น
 ระยะๆ โดยในระยะแรกอาจวางเป้าหมายสำหรับหมายเลขโทรศัพท์ที่ขึ้นต้นด้วย 1 เท่านั้น ต่อ

จากนั้นในระยะที่สอง ทดสอบกับหมายเลขโทรศัพท์ที่ขึ้นต้นด้วยหมายเลข 2 หมายเลข 3 จนกระทั่งถึงหมายเลข 9 ในระยะต่อไป เป็นการทดสอบการติดต่อกันระหว่างหมายเลข จนกระทั่งระบบนี้สามารถติดต่อกันได้ทุกหมายเลข ซึ่งการทำงานนี้เป็นลักษณะที่พัฒนาระบบใหม่แทนที่ระบบเดิมที่ทำงานอยู่ การทำงานประกอบด้วย 2 ระบบที่ทำงานขนานกันไป นั่นคือ development system และ production system โดย development system ใช้สำหรับออกแบบ เขียนคำสั่งโปรแกรม และทดสอบในส่วนของระบบใหม่หรือในระยะถัดไปเมื่อเราแน่ใจว่าระบบใหม่สามารถผนวกหรือแทนที่ระบบที่กำลังปฏิบัติการได้ เราจะนำคำสั่งโปรแกรมใหม่ผนวกกับคำสั่งโปรแกรมเก่าเพื่อผลิตเป็นระบบใหม่(active production system) เนื่องจากระบบนี้ซับซ้อนและมีการดำเนินงานหลายระยะ การจัดการที่ตุนั้นเราควรเก็บประวัติของการเปลี่ยนแปลงการทำงานของระบบในระยะต่างๆในเอกสาร เพื่อให้สำหรับการทำความเข้าใจและสืบค้น เพื่อประโยชน์ในการแก้ไขความผิดพลาดที่อาจเกิดขึ้นในภายหลัง

(3) Interaction with Other Systems

ระบบส่วนมากไม่สามารถทำงานได้เพียงลำพังต้องมีการติดต่อกับระบบอื่นๆ ในแง่ของการรับและส่งสารสนเทศต่างๆ ยิ่งระบบที่มีความซับซ้อนมากๆอาจต้องมีการพัฒนาแบบขนานกัน(concurrent)ทั้งระบบเก่าและระบบใหม่ ดังนั้นในการพัฒนาระบบเป็นการยากที่จะแน่ใจได้ว่าข้อมูลที่ติดต่อกันระหว่างระบบนั้นถูกต้องและสมบูรณ์

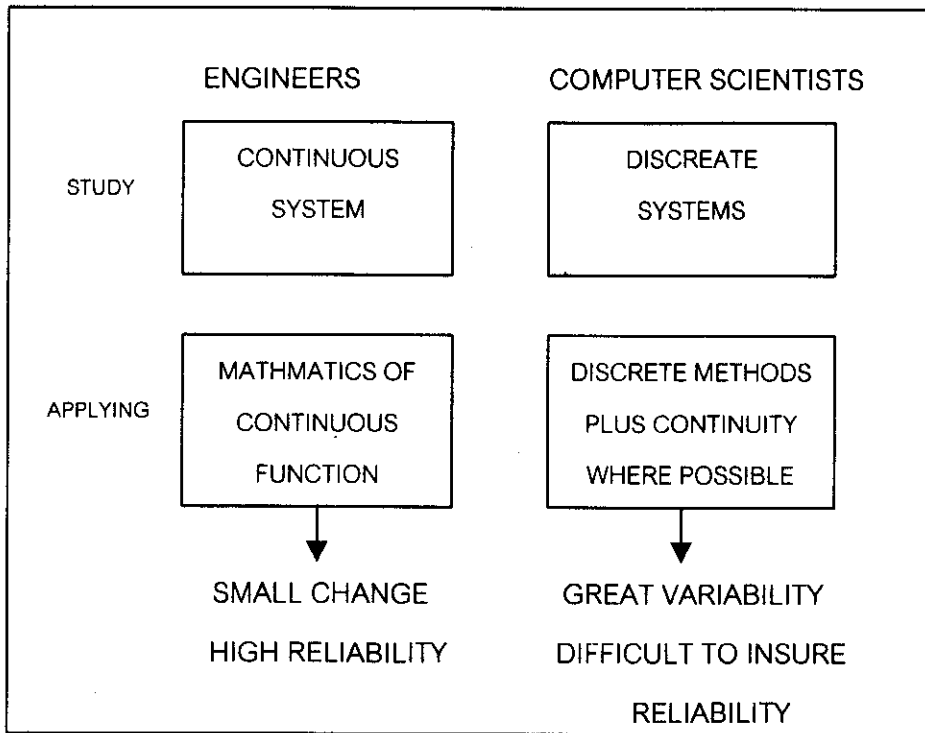
(4) The Nature of Computer System Themselves

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

ชนิดของโครงการซอฟต์แวร์

การพัฒนาซอฟต์แวร์นั้นแต่ละโครงการมีความยากง่ายและความซับซ้อนแตกต่างกัน เราอาจมองง่าย ๆ จากขนาดของโปรแกรมหรือจากเอกสารการอ้างอิงที่ใช้ในโครงการทั้งหมด ถ้ามีขนาดและจำนวนมากแสดงว่าโครงการมีขนาดใหญ่มากมีความซับซ้อนมากแต่ในความเป็นจริงยังมีปัจจัยอีกหลายอย่างด้วยกันที่ส่งผลให้ซอฟต์แวร์มีความยากและซับซ้อนแตกต่างกันออกไป 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; tolerance for downtime	Can tolerate several hours of downtime	Can tolerate short periods of downtime	Can tolerate 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 แต่ในระบบเช่นระบบคลังสินค้าเป็นระบบที่ซับซ้อนขึ้นมีการทำงานที่เกี่ยวข้องกัน ผู้ใช้มีหลายระดับและมีความต้องการที่แตกต่างกันไป ซึ่งส่งผลให้การสร้างระบบมีความยากกว่าระบบแรก ดังนั้นจำนวนของงานที่กระทำในระบบ(number of functions performed) จึงเป็นคุณลักษณะหนึ่งที่มีผลต่อระดับความง่ายของซอฟต์แวร์ ในบางกรณีถ้าความต้องการของลูกค้าเป็นสิ่งที่เราไม่คุ้นเคยอาจเป็นทฤษฎีใหม่หรือวิธีการใหม่ๆ (novelty of function) ที่เราไม่เคยสร้างมาก่อน การพัฒนาระบบจะยากขึ้นเพราะเราต้องหาทางแก้ปัญหาที่เราไม่เคยพบมาก่อน ดังนั้นยังมีสิ่งใดที่ใหม่ๆในระบบที่เราไม่คุ้นมากเท่าใดจะส่งผลให้การพัฒนาระบบมีความยากตามไปด้วยเท่านั้น

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

จำนวนข้อมูลที่เก็บ(amount of data) ในระบบเป็นปัจจัยหนึ่งที่สำคัญถ้าสื่อที่ใช้เก็บไม่สามารถเก็บข้อมูลได้หมดในดิสก์เพียงหนึ่งตัว ต้องใช้ดิสก์หลายๆตัวหรือใช้สื่ออื่นๆเพิ่มเติมการเข้าถึงหรือการจัดการจะยากตามไปด้วยเพราะข้อมูลถูกจัดเก็บกระจายทำให้ยากต่อการรวบรวม 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 ในระบบงานบางอย่างที่เกี่ยวข้องกับชีวิตมนุษย์เช่นระบบช่วยรักษาผู้ป่วย หรือระบบที่ควบคุมการส่งยานอวกาศหรือสำรวจอวกาศ ระบบเหล่านี้ถือว่าเป็นระบบที่ต้องกระทำได้โดยต่อเนื่องเราถือว่าเป็นระบบวิกฤต(criticality) กล่าวคือระบบจะล้มเหลวไม่ได้ การพัฒนาระบบเหล่านี้จะมีความยากมากเนื่องจากเราต้องสร้างระบบออกแบบให้มีสองระบบทำงานซ้อนกันไปคือมีระบบจริงและระบบสำรอง ถ้าระบบจริงล้มเหลวหรือผิดปกติ ระบบสำรองจะทำงานแทนที่โดยอัตโนมัติทันที

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

กรณีศึกษา

Weaver Farm เป็นพื้นที่ใหม่หลายพันเอเคอร์ที่เป็นส่วนหนึ่งของวนอุทยานแห่งชาติแห่งหนึ่ง มีธรรมชาติที่สวยงามประกอบด้วย ทะเลสาบขนาดใหญ่ ภูเขา ป่าไม้ มีแม่น้ำลำธารสายเล็กๆที่ไหลลงสู่ทะเลสาบ ใช้สำหรับการท่องเที่ยวเพื่อพักผ่อนหย่อนใจ โดยมีพื้นที่หลายๆจุดที่นักท่องเที่ยวสามารถเดินทางไปหาความสำคัญได้โดยการเดิน 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 คอร์ตเทนนิส โต๊ะปิกนิก ท่าเรือ อู่เรือ และอื่นๆ
- (3) ระบบสามารถสร้างแผนงานการบำรุงรักษาสถานที่ต่างๆ รวมถึงพื้นที่ธรรมชาติและทำการก่อสร้างสิ่งใหม่ๆในพื้นที่นี้
- (4) ระบบสามารถจำลองผลกระทบของสภาวะแวดล้อมของการเลือกใช้พื้นที่ที่แตกต่างกัน โดยระบบสามารถช่วยให้ผู้บริหารตัดสินใจหาพื้นที่ที่ดีที่สุดเพื่อใช้ประโยชน์โดยไม่ส่งผลกระทบต่อสภาพแวดล้อมเดิม

เราลองมาวิเคราะห์ถึงระดับความยากง่ายของโครงการนี้โดยเปรียบเทียบกับข้อมูล
จากตารางที่ 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 เราสามารถสรุปได้ว่าโครงการนี้ระดับความยากง่ายในระดับปานกลาง

1.8

สรุป

บทนี้เป็นการแนะนำแนวความคิดของวิศวกรรมซอฟต์แวร์ ที่มีการนำเทคนิคและเครื่องมือมาช่วยในการแก้ปัญหาเพื่อให้ซอฟต์แวร์ที่พัฒนาหรือสร้างขึ้นมีคุณภาพ ซึ่งในขั้นตอนแรกต้องทำความเข้าใจกับปัญหาโดยพยายามมองให้เห็นกลุ่มขององค์ประกอบต่างๆที่มีความเกี่ยวข้องกัน ต่อจากนั้นแตกปัญหาใหญ่ออกเป็นปัญหาย่อยๆ และทำการแก้ปัญหาจากปัญหาย่อยๆเหล่านี้และรวมกันเป็นการแก้ปัญหาทั้งระบบ การสร้างซอฟต์แวร์ที่มีคุณภาพต้องทำความเข้าใจถึงแนวความคิด เครื่องมือ เทคนิค และกฎเกณฑ์ต่างๆ นำมาประยุกต์ใช้ในการพัฒนาซอฟต์แวร์

ระบบ หนึ่งประกอบด้วย เ็นิตตี้ กิจกรรม ความสัมพันธ์ และขอบเขตของระบบ ผู้ที่เกี่ยวข้องกับระบบจะประกอบด้วยบุคคล 3 กลุ่มคือลูกค้า ผู้ใช้ระบบ และผู้พัฒนาระบบ โดย การพัฒนาระบบประกอบด้วย 9 ขั้นตอนประกอบด้วย การวิเคราะห์ความต้องการ การออกแบบระบบ การออกแบบโปรแกรม การสร้างโปรแกรม การทดสอบโมดูล การทดสอบรวม การทดสอบระบบ การส่งมอบระบบ และการบำรุงรักษาระบบ ซึ่งในแต่ละขั้นตอนสามารถมีการย้อนกลับกระทำซ้ำในขั้นตอนที่ผ่านๆมาได้ถ้าตรวจพบว่าระบบนั้นทำงานไม่ตรงกับความต้องการของลูกค้า ปัญหาที่เกิดขึ้นในการพัฒนาซอฟต์แวร์เป็นเรื่องของความต้องการของลูกค้า ไม่คงที่ ความซับซ้อนของโครงการที่มีผลต่อการติดต่อสื่อสารกับระบบอื่นๆ การแบ่งการพัฒนาเป็นหลายระยะ รวมทั้งระบบคอมพิวเตอร์ที่ใช้ในการพัฒนาด้วย

ซอฟต์แวร์ที่มีคุณภาพสูงต้องสามารถทำงานตามที่ลูกค้าต้องการได้ ใช้ทรัพยากรคอมพิวเตอร์อย่างถูกต้องและมีประสิทธิภาพ ง่ายต่อผู้ใช้ในการเรียนรู้และใช้งาน และผู้พัฒนาสามารถออกแบบ เขียนคำสั่ง และบำรุงรักษาระบบได้ง่าย

ความยากง่ายในการพัฒนาซอฟต์แวร์สามารถแบ่งออกเป็น 3 ระดับคือต่ำ กลางและสูง เราสามารถวัดความยากง่ายของซอฟต์แวร์ได้โดยวิเคราะห์และประเมินจากคุณลักษณะต่างๆ ของระบบเช่น หน้าที่ของระบบ จำนวนของข้อมูล จำนวนของผู้ใช้ในระบบ เวลาในการตอบสนองกับระบบ ระบบรักษาความปลอดภัย ฮาร์ดแวร์ที่ใช้ ระยะในการพัฒนา ประสิทธิภาพของผู้พัฒนาระบบ 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. คุณลักษณะใดบ้างที่สามารถวัดความยากง่ายในการพัฒนาซอฟต์แวร์ จงยกตัวอย่างประกอบให้เห็นจริง

ปฏิบัติ

หลังจากเรียนบทที่ 1 แล้ว ให้นักศึกษาจัดกลุ่มกันกลุ่มละไม่เกิน 8 คน โดยเลือกระบบงานที่ต้องการทำโครงการ 1 ระบบที่น่าสนใจจากระบบงานจริง เช่น ระบบขายสินค้า ระบบร้านอาหาร ระบบโรงพยาบาล ระบบการเงิน ระบบพนักงาน เป็นต้น ให้ทำการศึกษาระบบโดยทำเอกสารส่งงานประกอบด้วย

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

บทที่ 2

การวางแผนโครงการ

การพัฒนาระบบนั้นมักมีกิจกรรมที่ต้องกระทำหลายขั้นตอนโดยกระทำเป็นลำดับ เริ่มต้นต้องมีการตกลงทำสัญญากับลูกค้าเพื่อระบุถึงความต้องการของลูกค้าที่ต้องการ มีการตกลงราคากันจนพอใจกันทั้งสองฝ่าย รวมถึงกำหนดเวลาที่ใช้พัฒนาและการส่งมอบระบบให้แน่นอน ตลอดจนเงื่อนไขอื่นๆที่ต้องการ ดังนั้นผู้บริหารโครงการต้องให้ประสบการณ์ของตนเองที่มีอยู่ ในการประมาณการเพราะถ้าประมาณการในสิ่งต่างๆผิดพลาดไปจากความเป็นจริงมากมายย่อมไม่ส่งผลดีในการพัฒนาโครงการเพราะจะส่งผลกระทบต่อโครงการที่พัฒนาอยู่ขาดทุนหรือล้มเหลวไม่ประสบผลสำเร็จตามที่ต้องการ ดังนั้นการทำแผนงานในการพัฒนาระบบเป็นเครื่องมือหนึ่งที่สามารถช่วยผู้บริหารโครงการใช้สำหรับประมาณการเวลาและค่าใช้จ่าย รวมทั้งสามารถควบคุมกิจกรรมต่างๆพร้อมทั้งวัดความก้าวหน้าของกิจกรรมต่างๆในโครงการได้ ทำให้การพัฒนาโครงการนี้ประสบผลสำเร็จตรงตามเป้าหมายที่ต้องการได้

2.1

การสร้างแผนงาน

หลังจากที่ผู้พัฒนาระบบและลูกค้าได้พบปะพูดคุยกันถึงรายละเอียดต่างๆที่ลูกค้าต้องการ มักจะมีคำถามจากลูกค้าถามผู้พัฒนาระบบเสมอๆว่า

- (1) คุณเข้าใจปัญหาและความต้องการของผมใช่ไหม ?
- (2) คุณสามารถสร้างระบบที่สามารถแก้ปัญหานี้ได้ใช่ไหม ?
- (3) คุณจะใช้ระยะเวลาานเท่าใด ?

(4) คุณจะคิดราคาเท่าไร ?

สำหรับคำตอบในข้อ (3) และข้อ (4) เราสามารถหาคำตอบได้ไม่ยากจากการสร้างแผนงานของโครงการงาน(project schedule) แผนงานของโครงการนั้นคือวัฏจักรในการพัฒนาซอฟต์แวร์ที่แบ่งออกเป็นระยะ(stages)หรือเฟส(phases) ในแต่ละระยะแบ่งออกเป็นงาน(tasks) หรือ กิจกรรม(activities) วิเคราะห์ความสัมพันธ์ของกิจกรรมต่างๆในระบบ และทำการประมาณในแต่ละกิจกรรม นำมารวมเป็นระบบใหญ่เราก็สามารถประมาณการได้ว่าใช้เวลาและราคาเท่าใด

การส่งมอบ

เมื่อผู้พัฒนาเริ่มสร้างระบบตามความต้องการของลูกค้า เพื่อความมั่นใจกันทั้งสองฝ่ายต้องมีการส่งมอบ(delivery)งานเป็นระยะๆ เพื่อให้ลูกค้าได้ทราบว่าเรามีการพัฒนาระบบไปถึงไหนแล้ว ? ถูกต้องหรือไม่ ? ซึ่งอาจมีการระบุในสัญญาที่กระทำต่อกันด้วยว่ามีการส่งมอบงานทั้งหมดกี่ระยะ? เมื่อใดบ้าง ? งานกระทำถึงไหน ? มีการชำระเงินในแต่ละงวดเท่าใด ? จนกระทั่งงานเสร็จสิ้น โดยทั่วไปการส่งมอบงานให้กับลูกค้าอาจเป็นเอกสาร(documents) หรือการแสดง(demonstrations) การทำงานของระบบ ตามข้อตกลงซึ่งอาจประกอบด้วย

- (1) เอกสาร
- (2) การแสดงหน้าที่ที่ระบบกระทำได้
- (3) การแสดงความถูกต้องของระบบ
- (4) การแสดงความน่าเชื่อถือ และระบบย่อยของระบบ

กิจกรรมและหลักชัย

หลักชัย(milestones) เป็นการวัดความก้าวหน้าของการพัฒนาระบบ สมมติว่าในการพัฒนาระบบมีกิจกรรมในการผลิตคู่มือผู้ใช้(user's manual) กิจกรรมนี้เราใช้เวลาในการพัฒนาผลิตหลายวันในการลำดับขั้นตอนการทำงานในหน้าจอต่างๆของระบบ การบรรยายรายละเอียดการทำงานต่างๆอย่างละเอียดเพื่อให้แน่ใจว่าผู้ใช้งานระบบสามารถนำไปใช้ประโยชน์ได้โดยง่าย จนกระทั่งเสร็จเป็นรูปเล่มสมบูรณ์ หลักชัยของกิจกรรมนี้คือการส่งมอบเอกสารคู่มือผู้ใช้ให้กับลูกค้าเรียบร้อยแล้วนั่นเอง

การวิเคราะห์โครงการเราต้องทราบถึงความแตกต่างระหว่างกิจกรรมและหลักชัย กิจกรรมนั้นเป็นส่วนหนึ่งของโครงการซึ่งต้องมีการใช้เวลาในการดำเนินงาน เป็นช่วงของเวลาที่มีจุดเริ่มต้นและจุดสิ้นสุด ส่วนหลักชัยเป็นเพียงจุดของเวลาที่กระทำกิจกรรมเสร็จสมบูรณ์

ระยะของแผนงาน

การสร้างแผนงานเริ่มจากแบ่งโครงการออกเป็นระยะ(phase) ในแต่ละระยะจะแบ่งออกเป็นขั้นตอน(step) และแต่ละขั้นตอนจะแบ่งออกเป็นส่วนย่อยๆเท่าที่จะแบ่งได้เป็นกิจกรรม(activity) รูปภาพที่ 2.1 แสดงการแบ่งโครงการออกเป็นส่วนย่อยๆ และตารางที่ 2.1 แสดงการแบ่งโครงการการสร้างบ้านออกเป็น 2 ระยะ คือ ระยะการสร้างทิวทัศน์ และระยะการสร้างบ้าน โดยแต่ละระยะจะแบ่งออกเป็นขั้นตอนต่างๆ เช่นในระยะที่สองแบ่งเป็น 3 ขั้นตอนคือขั้นตอนที่ 1 เตรียมพื้นที่ ขั้นตอนที่ 2 สร้างภายนอกบ้าน ขั้นตอนที่ 3 ตกแต่งภายในบ้าน และแต่ละขั้นตอนจะแบ่งออกเป็นกิจกรรมต่างๆ ซึ่งต้องมีการดำเนินการจนกระทั่งเสร็จสมบูรณ์ เช่นกิจกรรม 2.2 เป็นการก่อสร้างกำแพง ต้องมีการเริ่มต้นดำเนินงานสมมุติเริ่มวันที่ 5 ใช้เวลา 5 วัน ในการดำเนินงาน เราสามารถวัดความก้าวหน้าของกิจกรรมนี้ได้จากวันและเวลาที่กระทำกิจกรรมนี้เสร็จสมบูรณ์ ซึ่งคือวันที่ 10 นั่นเอง สำหรับหลักชัยของกิจกรรมต่างๆของการสร้างบ้านแสดงในตารางที่ 2.2

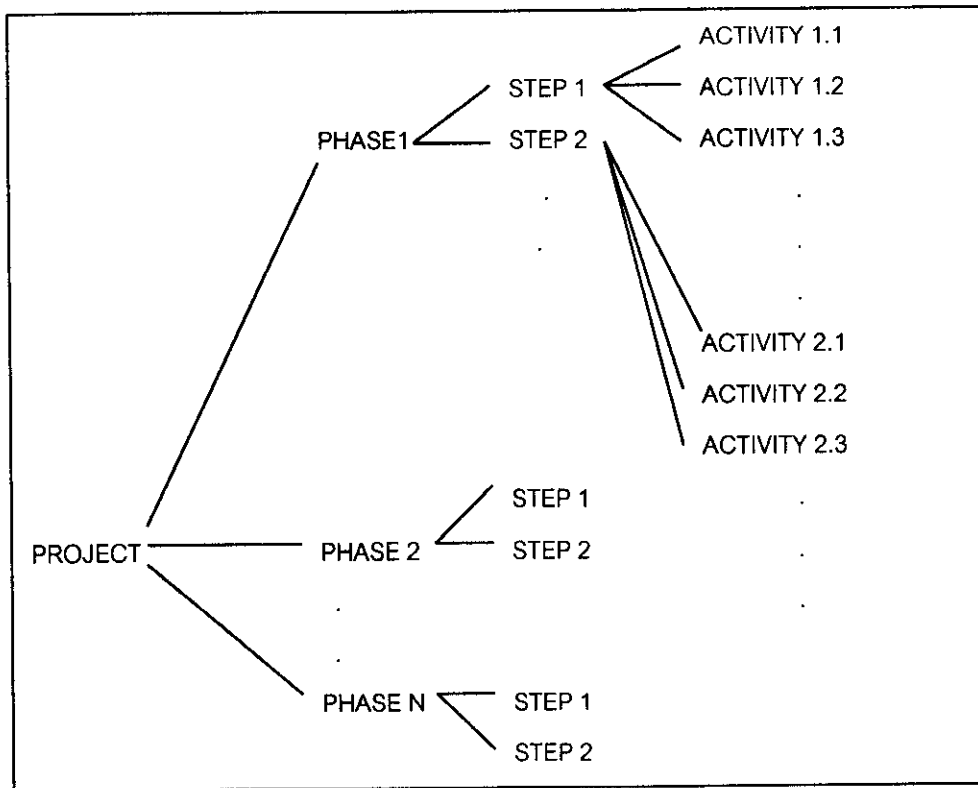
Work Breakdown Structure และ Activity Graphs

เมื่อแบ่งโครงการออกเป็นระยะ ขั้นตอน และกิจกรรม พร้อมกับหลักชัยเพื่อวัดความก้าวหน้าของโครงการแล้ว เมื่อพิจารณาถึงกิจกรรมต่างๆ เห็นได้ว่าบางกิจกรรมมีความสัมพันธ์กัน แต่บางกิจกรรมไม่เกี่ยวข้องกันเลย กิจกรรมบางอย่างสามารถแยกกันดำเนินงานได้ เราสามารถนำความสัมพันธ์ของกิจกรรมต่างๆมาสร้างเป็นกราฟกิจกรรม(Activity Graphs)

รูปภาพที่ 2.2 แสดงถึงความสัมพันธ์ของกิจกรรมในการสร้างบ้าน ซึ่งโหนด(node) ของกราฟแทนหลักชัยของโครงการ และเส้นที่เชื่อมต่อระหว่างโหนดแทนกิจกรรมที่ดำเนินงาน

ความสัมพันธ์ของกิจกรรมต่างๆเหล่านี้ เป็นเครื่องมือที่ทำให้เราทราบข้อมูลหลายๆอย่างเพื่อใช้ในการจัดการโครงการได้อย่างดี เช่นทราบถึงกิจกรรมแรกที่ต้องกระทำ ทราบถึงกิจกรรมต่อไป ทราบถึงกิจกรรมหรือเหตุการณ์ที่ต้องเกิดขึ้นหรือดำเนินการก่อน(precursor)กิจกรรมอื่นๆ

อย่างเช่นกิจกรรม 1.4 จะเริ่มกระทำต้องกระทำกิจกรรม 1.3 ให้เสร็จสิ้นก่อน หรือกิจกรรมที่ 2.3 และ กิจกรรม 3.1 จะเริ่มปฏิบัติก็ต่อเมื่อกิจกรรม 2.2 ต้องกระทำเสร็จก่อน การดำเนินงานของกิจกรรม 2.3 และกิจกรรม 3.1 นั้นเป็นอิสระต่อกันไม่ขึ้นต่อกันดังนั้นการทำงานสามารถกระทำงานขนานกันไป หรือพร้อมๆกันได้ ซึ่งหนดกิจกรรม 2.2 มีทางแยกในการดำเนินการสำหรับกิจกรรม 1.2 และ กิจกรรม 1.1 ที่มีการขีดเส้นประนั้น หมายความว่า การเริ่มต้นกระทำกิจกรรม 1.3 นั้นต้องกระทำกิจกรรม 1.2 และ 1.1 เสร็จสิ้นเสียก่อน ซึ่งกราฟกิจกรรมนี้ทำให้เราทราบลำดับของกิจกรรมต่างๆในโครงการอย่างชัดเจน นอกจากนี้เราสามารถประมาณระยะเวลาในการดำเนินงานของแต่ละกิจกรรมได้เรียกว่าduration ซึ่งช่วยให้เราสามารถวางแผนในการดำเนินงานโดยทราบว่ากิจกรรมนั้นๆวันไหนเป็นวันเริ่มต้น วันไหนต้องเสร็จ(due date) เมื่อประมาณการจนครบทุกกิจกรรม เราจะทราบถึงระยะเวลาทั้งหมดที่ใช้ในการพัฒนาโครงการซึ่งเป็น endpoint ที่ต้องส่งมอบระบบงาน ทำให้เราสามารถประมาณการระยะเวลาที่ใช้ในการพัฒนาโครงการนี้ได้



รูปภาพที่ 2.1 Separating project into Activities

ตารางที่ 2.1 Phases, Steps and Activities of Building a House

Phase 1 : Landscaping the lot

Step 1 : Clearing and grubbing

Step 2 : Seeding the turf

Step 3 : Planting shrubs and trees

Phase 2 : Building the house

Step 1 : Site preparation

Activity 1.1 : Surveying

Activity 1.2 : Obtaining permits

Activity 1.3 : Excavating

Activity 1.4 : Obtaining materials

Step 2 : Building exterior

Activity 2.1 : Foundation

Activity 2.2 : Outside walls

Activity 2.3 : Exterior plumbing

Activity 2.4 : Exterior electrical

Activity 2.5 : Exterior siding

Activity 2.6 : Exterior painting

Activity 2.7 : Doors and fixtures

Activity 2.8 : Roof

Step 3 : Finishing interior

Activity 3.1 : Interior plumbing

Activity 3.2 : interior electrical

Activity 3.3 : Wallboard

Activity 3.4 : interior painting

Activity 3.5 : Floor covering

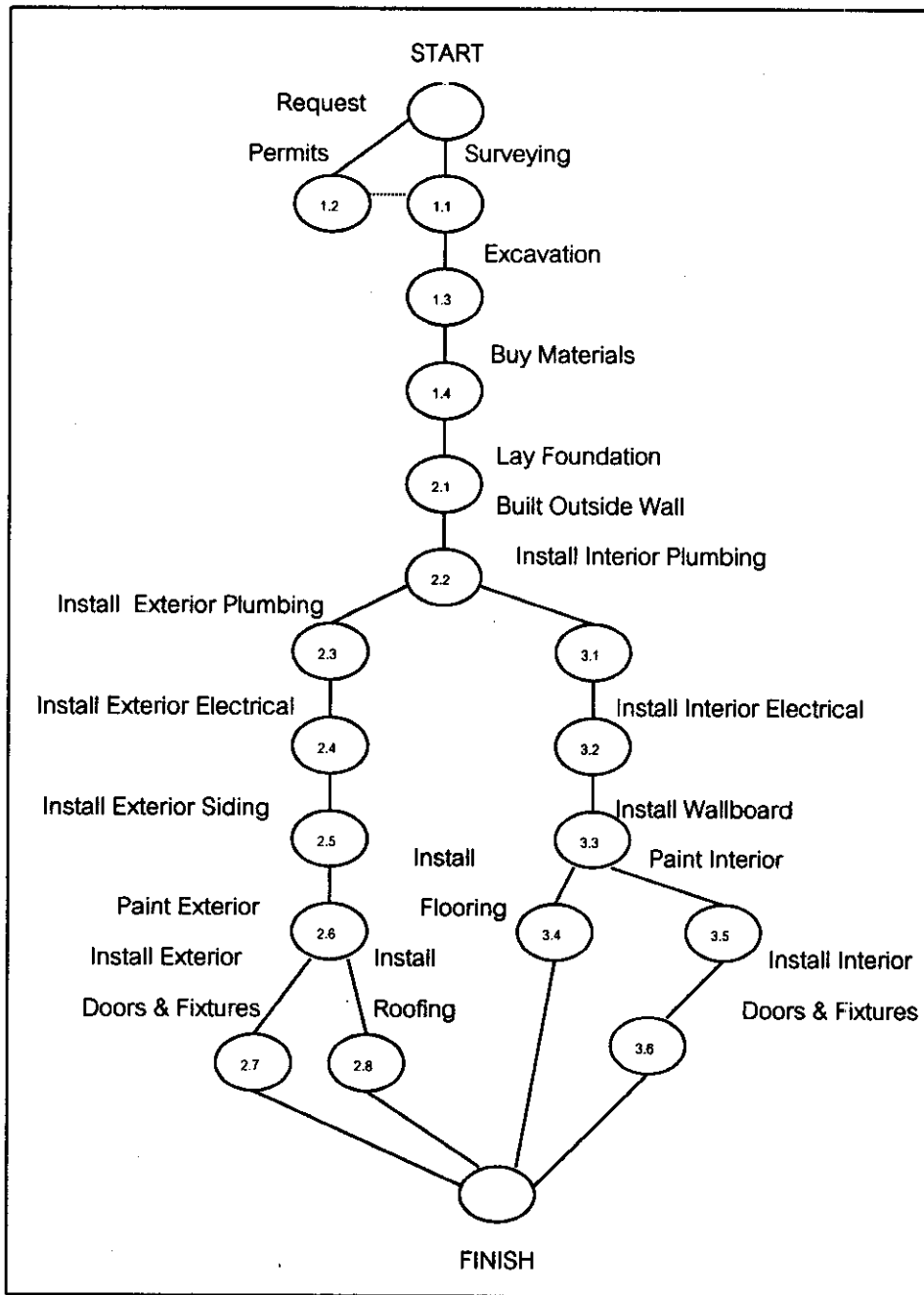
Activity 3.6 : Doors and fixtures

ตารางที่ 2.2 Milestones in Building a House

- 1.1 Survey complete
- 1.2 Permits issued
- 1.3 Excavation complete
- 1.4 Materials on hand
- 2.1 Foundation laid
- 2.2 Outside walls complete
- 2.3 Exterior plumbing complete
- 2.4 Exterior electrical work complete
- 2.5 Exterior siding complete
- 2.6 Exterior siding complete
- 2.7 Doors and fixtures mounted
- 2.8 Roof complete
- 3.1 Interior plumbing complete
- 3.2 Interior electrical complete
- 3.3 Wallboard in place
- 3.4 Interior painting complete
- 3.5 Floor covering laid
- 3.6 Doors and fixtures mounted

Estimating Completion

การประมาณการเวลาของโครงการนั้น เราเพียงเพิ่มระยะเวลาประมาณการของกิจกรรมต่างๆ โดยเขียนระหว่างเส้นที่เชื่อมระหว่างโหนด ดังรูปภาพที่ 2.3 สำหรับกิจกรรมที่ 2.7 , 2.8 , 3.4 และ 3.6 เป็นกิจกรรมที่ต้องเกิดก่อนจะสิ้นสุดโครงการ ดังนั้นเส้นที่เชื่อมระหว่างกิจกรรมเหล่านี้กับโหนดสิ้นสุดโครงการต้องมีค่าเท่ากับ 0



รูปภาพที่ 2.2 Activity Graph for Building a House

หลังจากเราเพิ่มระยะเวลาจนครบทุกกิจกรรมแล้วสิ่งที่เราได้ตามมาคือทราบถึงแผนงานในการดำเนินงานของกิจกรรมต่างๆ กล่าวคือถ้าเราเริ่มต้นโครงการในวันที่ 1 กิจกรรม 1.1 และ

1.2 จะเริ่มปฏิบัติจนกระทั่งเสร็จซึ่งใช้เวลารวม 15 วัน ในวันที่ 16 เราต้องเริ่มกิจกรรม 1.3 และดำเนินการปฏิบัติจนครบ 10 วัน ในวันที่ 26 เริ่มกิจกรรม 1.4 ต่อไป

Critical Path Method

เมื่อพิจารณาถึงเส้นทางในการดำเนินงานในโครงการจากจุดเริ่มต้นไปถึงจุดสิ้นสุดของโครงการ จะเห็นได้ว่ามีหลายเส้นทางและใช้เวลาไม่เท่ากัน เราสามารถวิเคราะห์เวลาที่น้อยที่สุดที่ใช้ในแต่ละกิจกรรมได้

จากรูปภาพที่ 2.3 เมื่อพิจารณากิจกรรมที่ 1.1 และ 1.2 จะเห็นได้ว่าการดำเนินงานของทั้งสองกิจกรรมเป็นลักษณะทำงานขนานกันไปเมื่อถึงหลักชัยกิจกรรมที่ 1.3 จึงจะเริ่มต้น กิจกรรมที่ 1.2 ใช้เวลา 15 วันแต่กิจกรรมที่ 1.1 นั้นใช้เวลาในการทำงานเพียงแค่ 3 วัน ดังนั้นกิจกรรมที่ 1.1 จึงมีเวลาเหลือเพื่อที่จะดำเนินงานเนื่องจากต้องรอกิจกรรมที่ 1.2 ทำงานเสร็จ

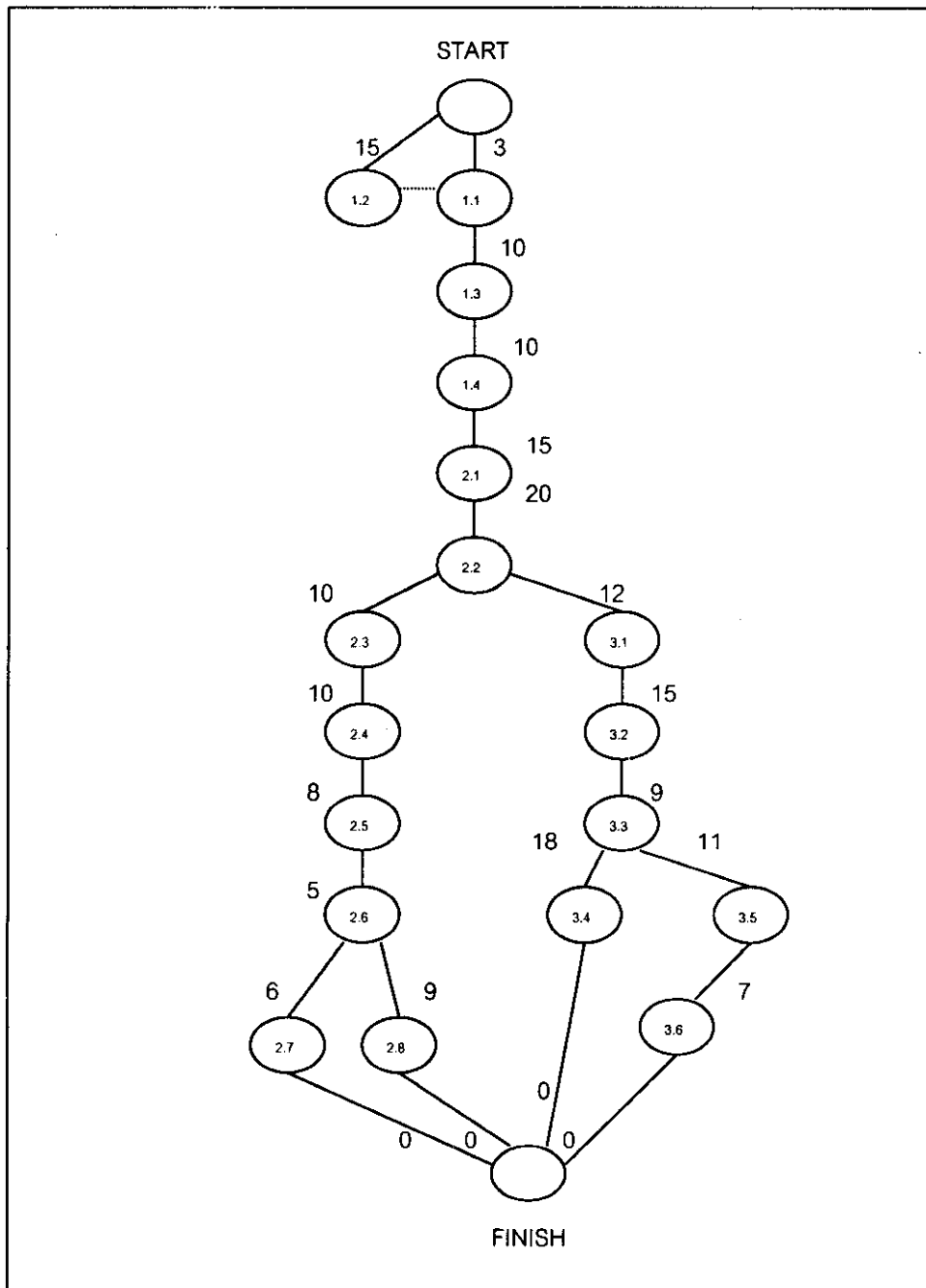
เราเรียกระยะเวลาจริงในการดำเนินกิจกรรมใดๆจนถึงหลักชัยว่า real time หรือ actual time และระยะเวลาทั้งหมดที่สามารถกระทำได้จนถึงหลักชัยว่า Available time ผลต่างของ available time และ real time เรียกว่า slack time

$$\text{slack time} = \text{available time} - \text{real time}$$

ดังนั้นกิจกรรมที่ 1.1 $\text{slack time} = 15 - 3 = 12$ วัน

การวิเคราะห์เส้นทางและหลักชัยในการดำเนินงานในโครงการนี้ เราเรียกว่า critical path method หรือ CPM ผู้บริหารโครงการสามารถกำหนดวันเริ่มต้นกิจกรรม(earliest start time) และวันสุดท้ายของการเริ่มกิจกรรม(latest start time) ได้ เพื่อใช้ในการจัดการ

Activity	Earliest Start Time	Latest Start Time	Slack
1.1	1	13	12
1.2	1	1	0
1.3	16	16	0
1.4	26	26	0
2.1	36	36	0
2.2	51	51	0



รูปภาพที่ 2.3 Activity Graph with Duration of Activity

เมื่อดำเนินกิจกรรมจนถึงหลักชัยของกิจกรรม 2.2 แสดงว่าใช้เวลาทั้งหมดเท่ากับ 70 วัน
การดำเนินกิจกรรมต่อไปมีถึง 4 ทางด้วยกันคือ

- (1) หลักชัยเริ่มจาก 2.3 ไปถึง กิจกรรม 2.7 ใช้เวลาดำเนินงานรวม 39 วัน
- (2) หลักชัยเริ่มจาก 2.3 ไปถึง กิจกรรม 2.8 ใช้เวลาดำเนินงานรวม 42 วัน
- (3) หลักชัยเริ่มจาก 3.1 ไปถึง กิจกรรม 3.4 ใช้เวลาดำเนินงานรวม 54 วัน
- (4) หลักชัยเริ่มจาก 3.1 ไปถึง กิจกรรม 3.6 ใช้เวลาดำเนินงานรวม 54 วัน

โดยกิจกรรมที่ 2.7 , 2.8 , 3.4 และ 3.6 ต้องกระทำเสร็จสิ้นโครงการถึงจะสิ้นสุด เมื่อพิจารณาในเส้นทางที่ยาวที่สุดใช้เวลา 54 วัน ดังนั้นเราสามารถทราบว่า ระยะเวลาในการพัฒนาโครงการใช้เวลาทั้งสิ้น $70+54 = 124$ วัน เส้นทางที่ใช้เวลานานที่สุดเราถือว่าเป็นเส้นทางวิกฤต สำหรับเส้นทางที่ใช้เวลาน้อยกว่าแสดงว่าเกิด slack time ดังนั้นเส้นทางที่เริ่มจากกิจกรรม 2.3 ไปจนถึงกิจกรรม 2.8 มี slack time เท่ากับ 12 วัน และเส้นทางที่เริ่มจากกิจกรรมที่ 2.3 ไปถึงกิจกรรมที่ 2.7 มี slack time เท่ากับ 15 วัน ตารางที่ 2.4 แสดงผลลัพธ์ และ slack time ของทุกกิจกรรมจากกราฟที่ 2.3

หลักชัยที่ 2.6 มีทางแยกสองทางคือ กิจกรรม 2.7 กับ กิจกรรม 2.8 โดยเส้นที่เชื่อมต่อระหว่างกิจกรรม 2.6 กับ 2.8 จะยาวกว่า กล่าวคือใช้เวลาในการดำเนินกิจกรรมนานกว่า มีผลทำให้กิจกรรมที่อยู่ในเส้นทางที่สั้นกว่าเกิด slack time

เส้นทางที่ยาวที่สุดในการดำเนินการกิจกรรมในโครงการนั้นจะได้ค่า slack time ของกิจกรรมมีค่าเท่ากับ 0 เราจะเรียกเส้นทางนี้ว่า เส้นทางวิกฤต(critical path) กิจกรรมในเส้นทางนี้ผู้บริหารโครงการต้องให้ความสำคัญเป็นพิเศษถ้าควบคุมการดำเนินการไม่ดีจะมีผลให้โครงการนั้นเกิดความล่าช้า เกิดผลเสียหายตามมา การประมาณการเวลาที่สมควรใช้ข้อมูลของโครงการที่คล้ายกันที่พัฒนาเสร็จสมบูรณ์มาแล้ว เป็นการประมาณการค่าคาดหวัง (expected value)ซึ่งเป็นจุดของช่วงเวลา และความแปรปรวน(variance) ซึ่งเป็นความกว้างของช่วงเวลา ในลักษณะของกราฟประฆังคว่ำที่เรียกว่า normal distribution

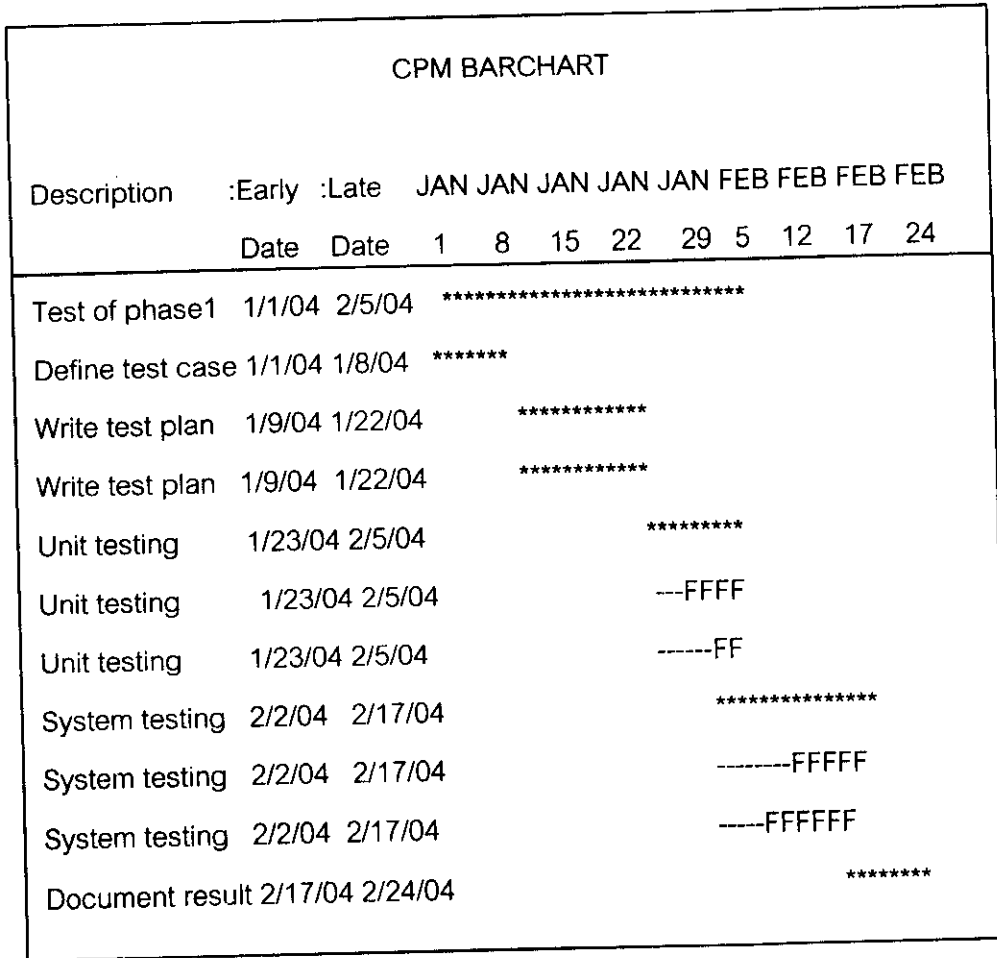
เทคนิคที่นิยมใช้ในการวิเคราะห์เส้นทางวิกฤตวิธีหนึ่งเรียกว่า The Program Evaluation and review Technique (PERT) เป็นเทคนิคที่ผู้บริหารโครงการสามารถกำหนดความน่าจะเป็นของเวลาเริ่มต้นและวันสุดท้ายของการกระทำกิจกรรมต่างๆของโครงการ เป็นกราฟกิจกรรมโดยคำนวณทางเดินวิกฤต และกิจกรรมในการดำเนินงานของแผนงาน

ตารางที่ 2.4 Slack Time for Project Activities

Activity	Earliest Start Time	Latest Start Time	Slack
1.1	1	13	12
1.2	1	1	0
1.3	16	16	0
1.4	26	26	0
2.1	36	36	0
2.2	51	51	0
2.3	71	83	12
2.4	81	93	12
2.5	91	103	12
2.6	99	111	12
2.7	104	119	15
2.8	104	116	12
3.1	71	71	0
3.2	83	83	0
3.3	98	98	0
3.4	107	107	0
3.5	107	107	0
3.6	118	118	0
Finish	124	124	0

รูปภาพที่ 2.4 เป็นส่วนหนึ่งของการสร้าง BAR CHART จากโปรแกรม project management package โดยแบ่งกิจกรรมในการพัฒนาโครงการและแสดงวันที่เริ่มต้นและวันสิ้นสุดของกิจกรรมต่างๆเป็นช่วงของเวลา เครื่องหมาย *(asterisks) แทน critical path เครื่องหมาย - (dash) และ F แทนไม่ใช่เส้นทางวิกฤต โดยเครื่องหมาย F แทน float หรือ slack time

รูปภาพที่ 2.4 CPM Bar Chart



2.2

บุคลากรโครงการ

ปัจจัยอย่างหนึ่งที่มีผลต่อการประมาณการเวลาและค่าใช้จ่ายในการพัฒนาโครงการซอฟต์แวร์ นั้นคือบุคลากรที่ทำงานในโครงการนั้นๆ เพราะบุคลากรแต่ละคนมีประสบการณ์ ความสามารถ

แตกต่างกันออกไป ทีมงานในการพัฒนาซอฟต์แวร์นั้นมีจำนวนมากหรือน้อยแตกต่างกันไปขึ้นอยู่กับขนาดของโครงการ ผู้บริหารโครงการที่ดีจะต้องสามารถกำหนดจำนวนของทีมงานและกำหนดหน้าที่ความรับผิดชอบได้อย่างเหมาะสม ซึ่งเรียกว่า checks and balances

Personnel Characteristics

โดยทั่วไปทีมงานในการพัฒนาโครงการจะต้องกระทำงานตามแผนงานในวัฏจักรในการพัฒนาระบบอันประกอบด้วย

1. Requirements analysis
2. System design
3. Program design
4. Program implementation
5. Testing
6. Training
7. Maintenance
8. Quality assurance

บุคลากรในทีมงานพัฒนาระบบอย่างน้อยต้องประกอบด้วยนักวิเคราะห์ระบบ ผู้ออกแบบระบบ ผู้เขียนโปรแกรม ผู้ทดสอบระบบและอื่นๆ ซึ่งบุคลากรในทีมงานที่ดีต้องมีคุณสมบัติดังนี้

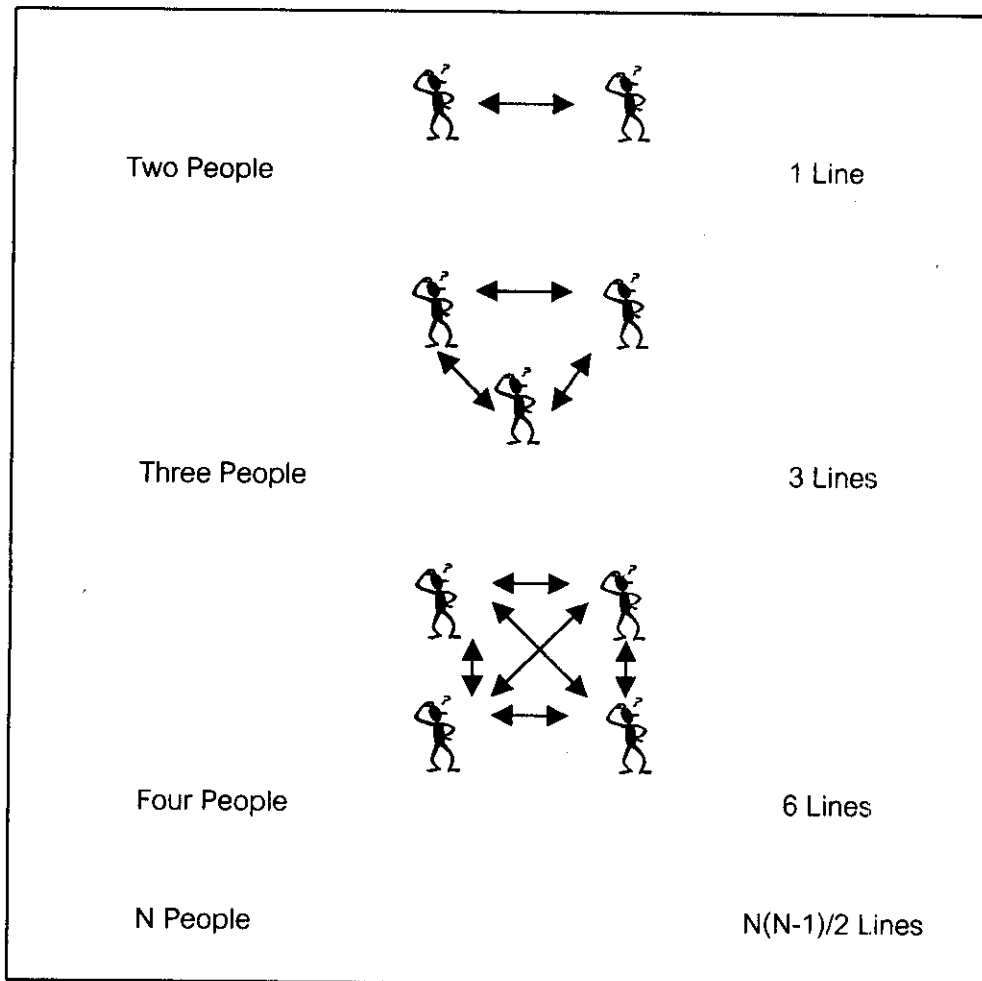
- ความสามารถในการกระทำงานที่ได้รับมอบหมาย ผู้บริหารโครงการต้องทราบว่าคุณบุคลากรในทีมงานแต่ละคนมีความสามารถหรือถนัดงานใดเป็นพิเศษ บางคนเขียนโปรแกรมเก่ง บางคนไม่ชอบเขียนโปรแกรมแต่ชอบทดสอบโปรแกรมบางคนชอบออกแบบระบบถ้าให้ออกแบบระบบจะกระทำได้ดีกว่างานอื่นๆ ดังนั้น การจ่ายงานให้กับบุคคลแต่ละคนอย่างเหมาะสมและตามความถนัด ทำให้ทีมงานไม่รู้สึกเครียดแต่รู้สึกสบายใจ และสนุกกับงาน มีผลช่วยเพิ่มผลผลิตในการพัฒนาซอฟต์แวร์
- ความสนใจ เป็นปัจจัยหนึ่งที่สามารถกำหนดความสำเร็จของบุคลากรในโครงการได้ เพราะมนุษย์เราถ้าชอบอะไรหรือสนใจอะไรก็สามารถกระทำงานนั้นได้ดี และถ้าเป็นงานที่ถนัดที่ชอบและกระทำได้ ย่อมใฝ่รู้ศึกษาค้นคว้าและสนใจในสิ่งใหม่ๆ หา

วิธีการแก้ปัญหาหรือกระทำสิ่งที่ดีกว่าแบบเดิมๆที่เคยกระทำมา ซึ่งทำให้การพัฒนาโครงการมีทางเลือกในการแก้ปัญหาที่ดีขึ้น

- **ประสบการณ์ในงานประยุกต์ที่คล้ายกัน** รวมทั้งเครื่องมือ และเทคนิค ผู้ที่มีประสบการณ์ในการทำงานโดยเคยทำงานมาหลายปีกับผู้เพิ่งเริ่มทำงาน เป็นที่แน่นอนว่าผู้ที่มีประสบการณ์มากกว่าย่อมสามารถทำงานที่ได้รับมอบหมายได้ดีกว่าในแง่ของเครื่องมือและเทคนิคก็คุ้นเคย ยิ่งเป็นงานประยุกต์ที่เคยทำมาแล้วก็ยิ่งดีใหญ่ สามารถนำงานเก่ามา reuse ทำให้ประหยัดเวลาและค่าใช้จ่าย
- **การฝึกอบรม** เป็นปัจจัยหนึ่งในการเลือกบุคลากรในทีมงานนอกจากสามารถทำงานตามที่ได้รับมอบหมายได้ และถ้ามีประสบการณ์เคยได้รับการฝึกฝนอบรมการปฏิบัติงานมาแล้วทำให้การทำงานคล่องตัวมากยิ่งขึ้น
- **ความสามารถในการสื่อสารกับบุคคลอื่น** เนื่องจากในทีมงานต้องมีการติดต่อระหว่างกันรวมทั้งติดต่อกับบุคคลภายนอกเช่นกับผู้ใช้ระบบและลูกค้า ความสามารถในการติดต่อสื่อสารเป็นปัจจัยหนึ่งที่ส่งผลกระทบต่อความก้าวหน้าของโครงการ เพราะถ้าพูดไม่รู้เรื่องหรือการสื่อสารไม่เข้าใจกันจะก่อให้เกิดปัญหาในการปฏิบัติงานในโครงการ ถ้าทีมงานมีขนาดเล็กปัญหาคงไม่มากเท่าใด แต่ถ้าทีมงานขนาดใหญ่การติดต่อระหว่างกันนั้นอาจเกิดความคลาดเคลื่อนได้สูง จากรูปภาพที่ 2.5 แสดงถึงการเพิ่มขึ้นของเส้นในการติดต่อสื่อสารระหว่างกัน ถ้ามีบุคคลเพียง 2 คนจะมีเส้นการสื่อสารเพียงเส้นเดียว ถ้ามีบุคคล 3 คนจะมีเส้นการติดต่อสื่อสารระหว่างกันทั้งหมด 3 เส้น ถ้ามีบุคคล 4 คนในทีมงานต้องมีเส้นการติดต่อสื่อสารระหว่างกันถึง 6 เส้น และถ้ามีบุคลากรในทีมงาน n คน จะต้องมีการติดต่อระหว่างกันในทีมงานถึง $n(n-1)/2$ เส้น ดังรูปภาพที่ 2.5
- **ความสามารถในการแบ่งเบาความรับผิดชอบกับบุคคลอื่น** เนื่องจากกิจกรรมต่างๆในการพัฒนาโครงการมีความเกี่ยวพันกัน ทีมงานที่ดีต้องมีน้ำใจในการแบ่งเบาภาระงานอื่นๆ รวมทั้งมีเหตุผลในการทำงานโดยยอมรับมติของที่ประชุม ไม่กระทำงานโดยใช้เหตุผลของตนเองเป็นหลัก ถ้อยที่ถ้อยอาศัยกัน
- **ความชำนาญในการจัดการ** ทีมงานที่ดีต้องมีการบริหารจัดการที่ดี ดังนั้นการควบคุมโครงการเป็นปัจจัยที่สำคัญประการหนึ่ง ถ้าควบคุมไม่ดีจะส่งผลกระทบต่อ

ทีมงานทุกคนเพราะถ้าเกิดความล่าช้าขึ้น จะส่งผลกระทบต่อการทำงานของบุคลากรคนอื่นที่ต้องทำงานต่อกัน เกิดความเสียหายในการพัฒนาโครงการ

รูปภาพที่ 2.5 Increase in Lines of Communication



ปัจจัยดังกล่าวข้างต้นนั้นมีผลต่อคุณภาพของทีมงานในการพัฒนา ซึ่งสามารถสรุปได้ดังรูปภาพที่ 2.6

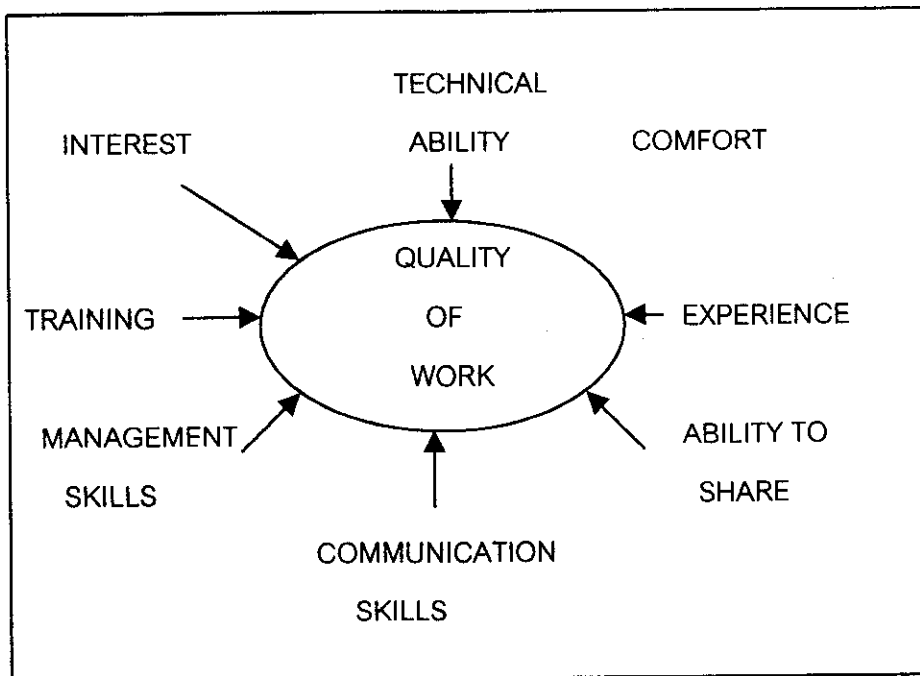
Project Team Organization

การจัดโครงสร้างขององค์กรในที่ทีมงานพัฒนาโครงการที่ดีและเหมาะสม สามารถช่วยเพิ่มคุณภาพในการพัฒนาโครงการได้ดีขึ้น กล่าวคือถ้าวิธีการดำเนินกิจกรรมต่างๆมีการจัดการ มีระบบ และระเบียบที่ดี จะทำให้การปฏิบัติงานมีความคล่องตัวและรวดเร็ว ซึ่งการจัดโครงสร้างขององค์กรในที่ทีมงานใดๆก็ต้องขึ้นกับปัจจัยหลายอย่าง ประกอบด้วย

- สไตล์การทำงานของบุคลากรในที่ทีมงาน
- จำนวนของบุคลากรที่ทำงานในโครงการ
- สไตล์การจัดการของลูกค้า

การจัดโครงสร้างขององค์กรที่นิยมกันวิธีหนึ่งเรียกว่า chief programmer team

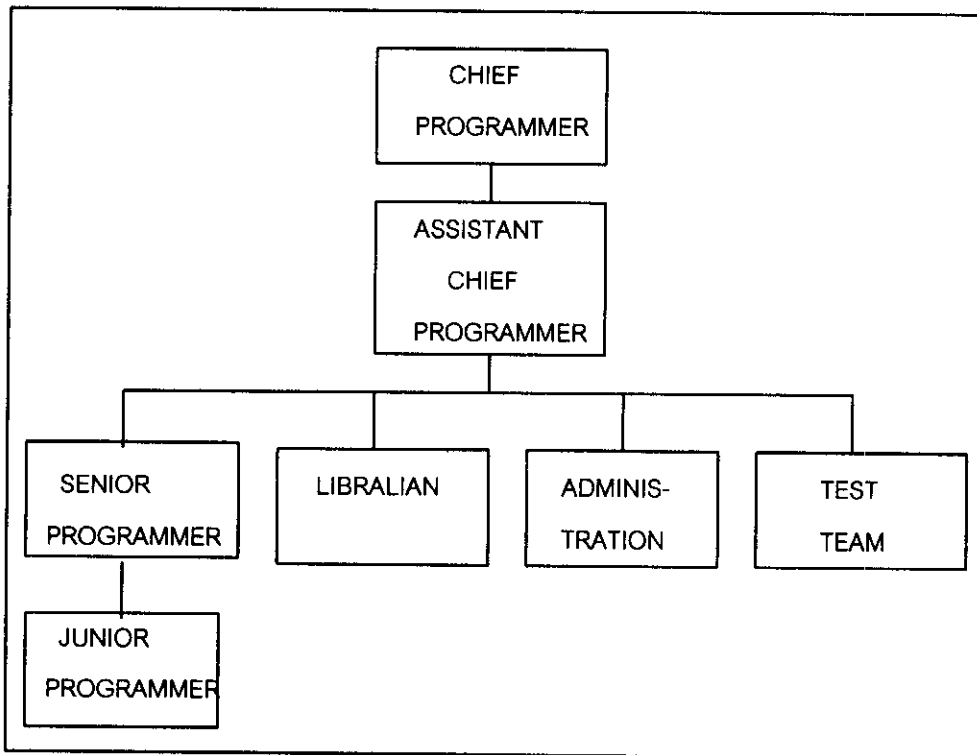
รูปภาพที่ 2.6 Characteristics that Affect Quality of Work



การจัดโครงสร้างวิธีนี้ตามรูปภาพที่ 2.8 เป็นสไตล์ที่เหมาะสมกับทีมงานที่ทำงานในลักษณะของ extrovert กล่าวคือทำงานเมื่อได้รับมอบหมายหรือสั่งการ ไม่ชอบที่จะมาถกเถียงหรือสื่อสาร

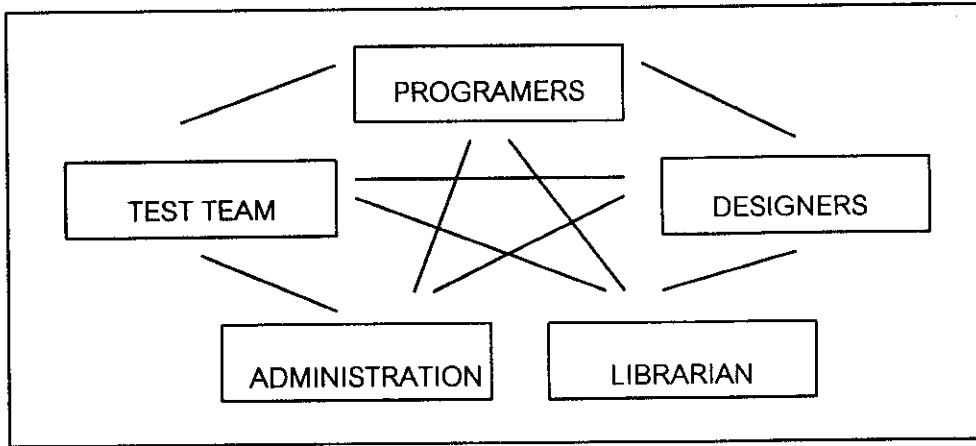
ระหว่างกัน การจัดโครงสร้างที่เหมาะสมควรเป็นลำดับขั้น มีหัวหน้าโปรแกรมเมอร์เป็นผู้ที่รับผิดชอบต่อการออกแบบและพัฒนาโครงการ แจกจ่ายงานให้กับสมาชิกทุกคนในทีมงานตามความถนัด โดยต้องตัดสินใจที่จะกระทำกิจกรรมต่างๆรวมทั้งการแก้ปัญหาต่างๆที่อาจเกิดขึ้นในการพัฒนาโครงการทั้งหมด แต่ถ้าบุคลากรส่วนมากของทีมงานในลักษณะของ introvert คือชอบที่คิดชอบถาม ชอบพูดคุยติดต่อสื่อสารถึงความเป็นไปรวมทั้งมีความคิดมีข้อคิดเห็นในการปฏิบัติงาน โครงสร้างแบบนี้ไม่เหมาะสมนัก เราอาจจัดโครงสร้างอีกวิธีหนึ่งที่เรียกว่า egoless ดังรูปภาพที่ 2.9

รูปภาพที่ 2.8 Structure of Chief Programmer Team



การจัดโครงสร้างวิธีนี้ทุกคนในทีมงานทุกคนมีสิทธิเท่าเทียมกันในการตัดสินใจ อาทิเช่นการเลือกเทคนิคที่จะนำมาใช้ในการออกแบบหรือใช้ในการทดสอบ การตัดสินใจเลือกวิธีใดนั้นจะใช้วิธีการประชุมปรึกษาหารือกันและทุกคนในทีมงานมีส่วนร่วมในการโหวตเพื่อหาข้อสรุป

รูปภาพที่ 2.9 Egoless programming Structure



จำนวนของบุคลากรในทีมงานก็เป็นอีกปัจจัยหนึ่งในการจัดโครงสร้างบุคลากรด้วย ถ้ามีขนาดใหญ่จะก่อให้เกิดปัญหาในการติดต่อสื่อสารระหว่างกัน ถ้ามีการจัดโครงสร้างแบบ egoless กว่าที่จะหาข้อสรุปในการทำงานก็ค่อนข้างจะยากดังที่มีคนกล่าวว่ามีคนก็มาก ความดังนั้นการจัดโครงสร้างแบบ chief programmer team จะเหมาะสมกว่าแบบ egoless เพราะวิธีนี้มีการติดต่อสื่อสารกันในทีมงานกันน้อย การสั่งงานหรือมอบหมายงานจะกระทำโดยตรงจากหัวหน้าทีมงานโปรแกรมเมอร์ ซึ่งมีรูปแบบที่ชัดเจนและแน่นอน เป็นมาตรฐานในการปฏิบัติ มีความคงที่กว่า

นอกจากนี้ลูกค้า หรือความต้องการของผู้ใช้งานก็มีส่วนที่เกี่ยวข้องกับการจัดการโครงสร้างของทีมงานด้วย เช่นลูกค้าต้องการบุคลากรเพื่อทำการทดสอบระบบ โดยเป็นคนละชุดกับผู้ที่ออกแบบหรือพัฒนาโปรแกรม เมื่อลูกค้าต้องการแบบนี้จะจัดโครงสร้างที่ขัดต่อความต้องการก็ไม่ก่อให้เกิดผลดีแต่อย่างใด ดังนั้นจากความต้องการดังกล่าวข้างต้นการจัดการโครงสร้างควรเป็นแบบ chief programmer team จึงจะเหมาะสมกว่าเพราะจะได้มอบหมายงานให้กับบุคคลใดบุคคลหนึ่งไปจะสะดวกมากกว่า

จากข้อมูลที่กล่าวข้างต้นสามารถสรุปลักษณะหรือข้อดีข้อเสียของการจัดการโครงสร้างทั้งสองชนิดได้ดังตารางที่ 2.5

ตารางที่ 2.5 Comparison of Organizational Structure

Highly Structured	Loosely Structured
High certainty	Uncertainty
Repetition	New Techniques or technology
Large projects	Small project

การจัดโครงสร้างทีมงานนั้นเราสามารถใช่วิธีการผสมผสานทั้งสองลักษณะก็ได้เพื่อให้เหมาะสมกับการปฏิบัติงานและให้เกิดประสิทธิผลให้มากที่สุด เช่นทีมงานมีการจัดโครงสร้างของกลุ่มโปรแกรมเมอร์ที่เขียนระบบย่อยเป็นแบบ egoless คือมีการปรึกษาหารือกันภายในกลุ่มถึงเทคนิคหรือวิธีการในการเขียนโปรแกรม โดยที่โครงสร้างของทีมงานทั้งหมดเป็น chief programmer team หรืออาจมีการจัดโครงสร้างของทีมงานทดสอบเป็น เป็นแบบ egoless แต่เลือกบุคคลหนึ่งเป็นผู้รับผิดชอบในการตัดสินใจ อย่างนี้เป็นต้น

2.3

ค่าใช้จ่ายในการสร้างระบบ

หลังจากที่เราสร้างแผนงานการสร้างโครงการ และมีการจัดโครงสร้างทีมงานที่เหมาะสมแล้ว ขั้นตอนต่อไปคือต้องประมาณการถึงค่าใช้จ่ายทั้งหมดที่ใช้ในการสร้างระบบ

Types of Costs

ค่าใช้จ่ายทั้งหมดที่เกิดขึ้นในการพัฒนาโครงการนั้น เราสามารถแบ่งออกเป็น 5 กลุ่มคือ

1. **Computer-related** เป็นค่าใช้จ่ายของฮาร์ดแวร์ และซอฟต์แวร์ ที่จัดซื้อ หรือจัดหามาใช้ในการพัฒนาโครงการ เช่น โปรเซสเซอร์, หน่วยความจำ, อุปกรณ์ในการรับข้อมูล , อุปกรณ์ใน

- การแสดงผล, โปรแกรมระบบ, case tools, โปรแกรมจัดการฐานข้อมูล, อุปกรณ์ในการติดต่อสื่อสาร หรือเครื่องมือต่างๆที่นำมาใช้ในโครงการ
2. **Facilities** เป็นค่าใช้จ่ายทางด้านสาธารณูปโภค ความสะดวกสบายในการพัฒนาโครงการงาน เช่น ค่าเช่าสำนักงาน, ระบบรักษาความปลอดภัย, ค่าแอร์, ค่าน้ำ, ค่าไฟ, ค่าโทรศัพท์, ค่าเฟอร์นิเจอร์และอุปกรณ์ในสำนักงาน และอื่นๆ
 3. **People** ค่าใช้จ่ายในส่วนนี้ได้แก่ เงินที่ต้องจ่ายให้กับบุคลากรในที่ทำงาน การประมาณการในส่วนนี้สามารถประมาณการได้จากความพยายาม (effort) ในการพัฒนาโครงการจากวิธี work breakdown structure ซึ่งได้มีการประมาณการระยะเวลาที่ใช้ในการพัฒนาของกิจกรรมต่างๆในโครงการโดยสามารถนำเวลามาคูณกับอัตราจ้าง จะได้เป็นค่าใช้จ่ายที่ต้องจ่ายในแต่ละกิจกรรม แต่การประมาณการวิธีนี้มีความคลาดเคลื่อนสูงเนื่องจากระยะเวลาที่ใช้ในการพัฒนาของกิจกรรมต่างๆนั้นขึ้นกับปัจจัยอื่นๆอีก เช่น สไตล์การทำงาน, โครงสร้างของทีมงาน, ความสามารถ, ความสนใจ, ประสบการณ์ของผู้พัฒนา และอื่นๆอีกมากมาย ดังนั้นการประมาณการค่าใช้จ่ายในส่วนนี้ต้องอาศัยผู้บริหารโครงการที่มีประสบการณ์และความสามารถสูงจึงจะทำให้อผลการประมาณการเกิดความผิดพลาดน้อยลง
 4. **Project complexity** ค่าใช้จ่ายในส่วนนี้จะคิดจากความซับซ้อนในการพัฒนาที่ส่งผลกระทบต่อระยะเวลาของการพัฒนาโครงการ โดยคำนวณจากปริมาณของเอกสารที่ต้องการและจำนวนของระบบย่อยที่เกี่ยวข้องกันโครงการ ความซับซ้อนของระบบย่อยต่างๆมีผลต่อค่าใช้จ่ายเพราะต้องใช้เวลามากขึ้นในการออกแบบที่มีคุณภาพ
 5. **Project methods and tools** วิธีการจัดการเป็นปัจจัยหนึ่งที่มีผลต่อค่าใช้จ่ายของโครงการ อาทิเช่นการกำหนดมาตรฐานในการทำงาน ขั้นตอนในการดำเนินกิจกรรม เอกสาร การเก็บรายการเปลี่ยนแปลงที่เกิดขึ้นในการดำเนินกิจกรรม รวมทั้งวิธีการในการฝึกอบรมเนื่องจากต้องเสียเวลาในการฝึกสอนหรือเรียนรู้ ค่าใช้จ่ายในส่วนนี้ถือว่าเป็นค่าใช้จ่ายที่แฝงอยู่ในการพัฒนาโครงการแต่ต้องนำไปรวมกับค่าใช้จ่ายของผลิตภัณฑ์ด้วย

Cost Estimation

มีปัจจัยต่างๆที่ส่งผลกระทบต่อการประมาณการมากมาย เป็นการยากที่จะประมาณการได้อย่างถูกต้องหรือเกิดความคลาดเคลื่อนน้อยที่สุด แต่อย่างไรก็ตามมีวิธีการในการประมาณการค่าใช้จ่ายอยู่หลายวิธีที่สามารถใช้เป็นแนวทางเพื่อช่วยผู้บริหารโครงการตัดสินใจได้ดังนี้

1. Analogies เป็นวิธีการประมาณการค่าใช้จ่ายโดยเปรียบเทียบกับโครงการเก่าที่คล้ายๆกัน ที่พัฒนาเสร็จสมบูรณ์แล้ว เช่นโครงการ A เหมือนกับ โครงการ B มาก แต่โครงการ B ขนาดเล็กเป็นครึ่งหนึ่งของโครงการ A ดังนั้นค่าใช้จ่ายของโครงการ B เป็นครึ่งหนึ่งของโครงการ A เป็นต้น
2. Work break down structure เป็นการประมาณการค่าใช้จ่ายโดยแบ่งตามหน่วยงานหรือกิจกรรมต่างๆที่กระทำในโครงการโดยนำเวลาที่ใช้ในแต่ละหน่วยงานมาคิดค่าใช้จ่ายแยกกันไป ต่อจากนั้นนำมาหาผลรวมของโครงการทั้งหมด โดยแท้จริงแล้ววิธีนี้เป็นเครื่องมือที่ช่วยในการวัดความก้าวหน้าของโครงการมิใช่เป็นเครื่องมือในการประมาณการค่าใช้จ่าย ดังนั้นผลลัพธ์มีโอกาสเกิดความผิดพลาดสูง
3. Cost per unit of measurement เป็นวิธีที่ใช้จำนวนบรรทัดคำสั่งเพื่อประมาณการค่าใช้จ่าย โดยผู้บริหารโครงการต้องหาค่าของจำนวนบรรทัดที่เหมาะสมเพื่อใช้ในการประมาณการประมาณการจำนวนบรรทัดคำสั่งที่คืนั้นควรประมาณการจากขนาดของโปรแกรมที่คิดว่าดีที่สุด(a) , ขนาดของโปรแกรมที่แย่ที่สุด(b) และขนาดปานกลาง(m) นำมาคำนวณโดยใช้สูตร $(a+4m+b) / 6$ ค่าที่ได้จะเป็นค่าที่เฉลี่ยซึ่งเหมาะสมสำหรับนำไปใช้ประมาณการค่าใช้จ่าย หลังจากได้ค่าของขนาดโครงการที่เหมาะสมเราจะนำไปคำนวณหาค่าใช้จ่ายจากสูตรที่เกิดจากการวิเคราะห์ห้วงระหว่างจำนวนบรรทัดกับค่าใช้จ่าย ซึ่งเป็นข้อมูลที่เก็บรวบรวมไว้ในอดีต Walston and Felix [WAL77] ได้ทำการวิจัยถึงความสัมพันธ์ระหว่างความพยายาม คน-เดือน (E) กับ จำนวนพันบรรทัดคำสั่ง (L) ของโครงการต่างๆสามารถสรุปความสัมพันธ์ออกมาเป็นสมการได้ดังนี้

$$E = 5.2 L^{0.91}$$

สมการนี้เป็นแบบจำลองที่เรียกว่า resource model ต่อมา Nanus and Farr [NAN64] ได้ทดสอบโดยนำกลุ่มของข้อมูลที่แตกต่างกัน มาวิเคราะห์จะได้แบบจำลองค่าใช้จ่ายเป็นดังนี้

$$E = k 5.2 L^{0.91} \quad \text{โดยค่า } k \text{ คือค่าคงที่}$$

การประมาณการค่าใช้จ่ายวิธีนี้เกิดความคลาดเคลื่อนสูงเพราะการพัฒนาโครงการที่เลือกภาษาโปรแกรมที่แตกต่างกันจะมีความยากง่ายแตกต่างกัน จำนวนบรรทัดคำสั่งไม่เท่ากัน และต้องใช้เวลาในการพัฒนาแตกต่างกันไป เช่นเขียนด้วยภาษาปาสคาลเพียง 20 บรรทัด แต่เขียนด้วยภาษาแอสเซมบลีถึง 200 บรรทัด เป็นต้น

4. Equation เป็นวิธีการประมาณการค่าใช้จ่ายที่นำปัจจัยอื่นๆที่ส่งผลกระทบต่อการพัฒนาโครงการมาพิจารณาร่วมกับจำนวนบรรทัดคำสั่ง แบบจำลองการประมาณการในลักษณะนี้เป็น dynamic models กล่าวคือผลลัพธ์ของค่าที่ได้จะเปลี่ยนแปลงไปตามเวลาที่เปลี่ยนไป ปัจจุบันมีแบบจำลองการประมาณการในลักษณะนี้มากมาย

Effort and Work Environment Models

วิธีประมาณการค่าใช้จ่ายที่ใช้ในปัจจุบันมีมากมายหลายวิธี ในที่นี้จะขอกกล่าวถึงแบบจำลองค่าความพยายามและสภาพแวดล้อมการทำงานที่ได้วิเคราะห์ขึ้นมาและนิยมใช้กันดังมีรายละเอียดดังนี้

1. Putnam ได้สร้างแบบจำลองโดยนำความสัมพันธ์ของจำนวนบรรทัดคำสั่ง, ค่าใช้จ่ายในการพัฒนา และเวลาที่ใช้ในการพัฒนาโครงการจนเสร็จสมบูรณ์ มาวิเคราะห์ได้เป็นสมการการประมาณการรูปแบบดังนี้

$$L = c K^{1/3} T^{4/3}$$

โดย L = จำนวนพันบรรทัดคำสั่ง

T = เวลาที่ใช้ในการพัฒนาเป็นปี

K = ค่าความพยายามในการพัฒนาโครงการ

C = ค่าคงที่ ซึ่งแทนภาวะของเทคโนโลยีที่ใช้ในโครงการ

แบบจำลองนี้นิยมใช้ประมาณการโครงการขนาดใหญ่เพราะระยะเวลาในการพัฒนาเป็นปี

2. Esterling ได้ศึกษาถึงปัจจัยต่างๆที่มีผลต่อปริมาณงานของบุคคลในสำนักงาน เรียกว่า work environment โดยแบบจำลองจะประกอบด้วยตัวแปรต่างๆที่ส่งผลต่อการทำงานในสำนักงานดังนี้

a = เศษของวันที่ใช้สำหรับบริหารงานหรือทำงานที่ไม่เกี่ยวข้องกับโครงการ

t = จำนวนนาที่เฉลี่ยที่ถูกขัดจังหวะในการทำงาน

r = จำนวนนาฬิกาที่กลับไปทำงานใหม่หลังจากที่ถูกขัดจังหวะ

k = จำนวนของการขัดจังหวะต่อวันต่อคนที่ทำงานในโครงการ

p = จำนวนของการขัดจังหวะต่อวันต่อคนที่ไม่มีส่วนร่วมในโครงการ

i = ค่าใช้จ่ายปกติต่อบุคคล

d = ส่วนต่างของค่าใช้จ่ายล่วงเวลากับค่าใช้จ่ายปกติ

g = จำนวนชั่วโมงของการทำงานล่วงเวลา

แบบจำลองนี้คำนวณการได้ประโยชน์จากการทำงานต่อบุคคลในที่มงาน โดยวัดการทำงานมาตรฐานคือ 8 ชั่วโมงต่อ 1 วัน สมการมีรูปแบบดังนี้

$$w = (1/8)(8-8a+g-(4r/60)-((p(t+r)/60)+(((k(n-1)(t+))/60))$$

ดังนั้นถ้าต้องการทราบถึงเวลาของการทำงานที่ได้ประโยชน์ของสมาชิกทั้งหมดของทีมงานต่อวัน ก็เพียงแต่นำจำนวนของบุคลากรทั้งหมดในที่มงานคูณด้วยค่า w ผลลัพธ์ จะบอกเราให้ทราบได้ ถ้าเราต้องการทราบถึงประโยชน์ที่ได้รับใน 1 สัปดาห์ให้นำค่า 5 ไปคูณ

$$5nw = \text{useful hours of work per week}$$

โดย n คือจำนวนของบุคลากรทั้งหมดที่ทำงานในโครงการ

สมมุติถ้าเราประมาณการค่าใช้จ่ายในการสร้างระบบเท่ากับ 400 ชั่วโมง เราสามารถใช้แบบจำลองของ Esterling เพื่อหาค่าใช้จ่ายของพนักงานต่อหนึ่งวันได้

จากสูตร $c = nS(gd + 8(i+1))$ โดย S เป็นเงินเดือนพนักงานเฉลี่ย และค่าใช้จ่ายโครงการต่อคน-วันคือ c/nw

- Boehm ได้ทำการรวบรวมโครงการ 64 โครงการที่พัฒนาเสร็จสมบูรณ์แล้วมาวิเคราะห์ พบว่าจำนวนความพยายาม (effort) คน-เดือน มีความสัมพันธ์กับจำนวนพันบรรทัดคำสั่งที่สามารถแปลเป็นภาษาเครื่องได้ (I) ในสภาพแวดล้อมที่แตกต่างกันจะมีความสัมพันธ์ที่แตกต่างกัน โดยสร้างแบบจำลองการประมาณการค่าใช้จ่ายเรียกว่า COCOMO model ซึ่งแบ่งการประมาณการได้เป็น 3 ภาวะ คือ

- Organic mode เป็นโปรแกรมที่สามารถทำงานได้ด้วยตนเอง เป็นอิสระจากโปรแกรมอื่นๆ เช่นโปรแกรมที่พัฒนาสำหรับผู้ใช้งาน

$$M = 2.4 I^{1.05}$$

- Semi-detached mode เป็นโปรแกรมที่ใช้โดยโปรแกรมอื่น เช่นโปรแกรมตัดประโยชน์ที่สนับสนุนในงานของผู้ใช้อื่นๆ

$$M = 3.0 I^{1.12}$$

- Embedded mode เป็นโปรแกรมที่มีการปฏิสัมพันธ์กับโปรแกรมอื่นๆ เช่นโปรแกรมระบบ ซึ่งควบคุมการทำงานของระบบคอมพิวเตอร์

$$M = 3.6 I^{1.20}$$

นอกจากนี้ยังพบอีกว่ามีปัจจัยต่างๆอีกหลายอย่างที่ใช้เป็นตัวคุณความพยายาม ซึ่งเป็น ตัวแปรที่ทำให้ค่าใช้จ่ายเพิ่มมากขึ้นหรือน้อยลงกว่าเดิม ซึ่งสามารถแบ่งได้ถึง 15 ตัวแปรรวม 4 กลุ่ม คือ

กลุ่มที่ 1 System Parameters:

- Required reliability
- Size of the data bases
- Complexity of the system

กลุ่มที่ 2 Hardware Parameters:

- Constraints on execution time
- Constraints on storage
- Volatility of the virtual machine
- Response time

กลุ่มที่ 3 Personnel Parameters:

- Ability of the analysis
- Ability of the programmers
- Experience with the application
- Experience with the hardware
- Experience with the programming languages ,development tools

กลุ่มที่ 4 Project Parameters:

- Use of modern programming practices
- Use of development tools

- Existence of required development schedule

ตัวแปรทั้ง 15 ตัวนี้เรารวม เรียกว่า cost driver ซึ่งในแต่ละโครงการจะมีสภาพแวดล้อมแตกต่างกัน ซึ่งสามารถแบ่งได้เป็นหลายๆระดับมีการกำหนดค่าของตัวคุณในระดับต่างๆ เช่น Required reliability แบ่งเป็น 5 ระดับตามผลกระทบที่มีต่อโครงการ กำหนดตัวคุณ ความพยายามมีค่าแตกต่างกันดังตารางที่ 2.6

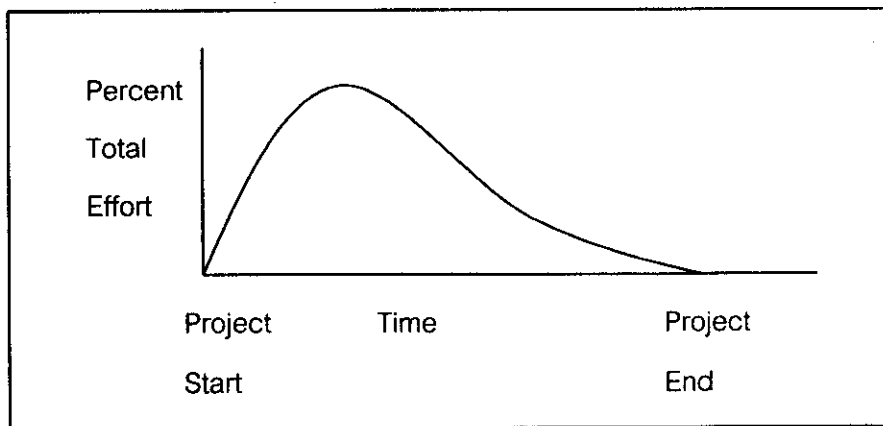
จากตารางที่ 2.6 ถ้าโครงการต้องการความน่าเชื่อถือสูงมากมากเนื่องจากเป็นโครงการที่เสี่ยงกับชีวิตมนุษย์ดังนั้นค่าความพยายามต้องมีค่ามากขึ้นกว่าปกติ จากตารางจะเห็นว่าตัวคุณมีค่า 1.4 คือใช้ค่าความพยายามมากกว่าปกติถึง 40 %

ตารางที่ 2.6 Reliability Categories and Multipliers

Tolerance	Effect	Multiplier
Very low	Slight inconvenience	0.75
Low	Looses easily recovered	0.88
Nominal	Loss recoverable with Moderate difficulty	1.00
High	High financial loss	1.15
Very high	Risk to human life	1.40

Norden[NOR63] ได้ศึกษาถึงค่าใช้จ่ายในการพัฒนาโครงการ พบว่าจะมีการกระจายค่าใช้จ่ายตามแสดงในรูปภาพที่ 2.11 ซึ่งมีการกระจายค่าใช้จ่ายในรูปของระฆังคว่ำ โดยค่าใช้จ่ายส่วนมากจะอยู่ในขั้นตอนของการวางแผนงานหรือเริ่มต้นโครงการ ค่าความพยายามทั้งหมดคือพื้นที่ใต้เส้นโค้ง ซึ่งรูปเส้นโค้งจะแตกต่างกันขึ้นอยู่กับวันที่โครงการเสร็จสมบูรณ์

รูปภาพที่ 2.11 Rayleigh Distribution of Project Effort



2.4

เครื่องมือและเทคนิคที่ใช้สำหรับประมาณการ

เครื่องมือและเทคนิคที่ใช้สำหรับประมาณการค่าใช้จ่ายและเวลานั้นมีมากมายหลายวิธีการรวมเรียกว่า Project management aids ซึ่งอาจประมาณการโดยมนุษย์หรือเป็นโปรแกรมอัตโนมัติ ซึ่งสามารถช่วยผู้พัฒนาในการวางแผนงาน วางแผนทางด้านการเงิน หรือควบคุมความก้าวหน้าของโครงการแก่ลูกค้าได้ สำหรับในหัวข้อนี้ขอยกตัวอย่างเครื่องมือโปรแกรมที่เรียกว่า Easytrak system ซึ่งพัฒนาโดย Planning Control International เป็นโปรแกรมที่ให้ผู้ใช้งานสามารถป้อนกิจกรรมต่างๆในโครงการพร้อมทั้งเวลาในการพัฒนาที่ประมาณการเป็นข้อมูลเข้า โปรแกรมนี้ใช้เทคนิค work breakdown structure , Gantt chart โดยแสดงกิจกรรมที่สามารถทำงานขนานกัน ทำให้ผู้ควบคุมโครงการเข้าใจและสามารถสั่งการได้อย่างถูกต้อง นอกจากนี้ยังสามารถสร้างกราฟทรัพยากรที่แสดงถึงบุคลากรที่รับผิดชอบในกิจกรรมต่างๆของโครงการในแต่ละระยะของการพัฒนา ทำให้สามารถโยกย้ายปรับเปลี่ยนระหว่างกิจกรรมได้ทันทีเมื่อเกิดปัญหาขึ้น ซึ่ง ผัง Easytrak ช่วยให้เราทราบถึงภาพรวมของแผนงานโครงการซึ่งสามารถช่วยในการ

ตัดสินใจถึงงบประมาณ โดยนำค่าใช้จ่ายจริงกับค่าใช้จ่ายประมาณการมาเปรียบเทียบ เพื่อวิเคราะห์ถึงการประมาณการในอนาคตได้อีกด้วย

2.5

การวางแผนโครงการ

การติดต่อกับลูกค้านั้นต้องมีการตกลงปรึกษาหารือกันถึงสิ่งที่ลูกค้าต้องการ ปัญหาคือความต้องการของลูกค้าไม่แน่นอนมีการปรับเปลี่ยนเสมอๆ ดังนั้นจึงส่งผลต่อการพัฒนา การแก้ไขปัญหานั้นโดยการสร้างเอกสารซึ่งบรรจุรายละเอียดต่างๆที่ชัดเจนของการพัฒนา โดยวัตถุประสงค์ของแผนงานเพื่อเป็นเอกสารที่ใช้แทนความเข้าใจระหว่างผู้พัฒนากับลูกค้าโดยอธิบายถึงกิจกรรมต่างๆที่ต้องกระทำ องค์ประกอบของเอกสารแผนงานประกอบด้วย

1. ขอบเขตของโครงการ เป็นการอธิบายถึงสิ่งที่ลูกค้าต้องการทั้งหมด โดยกำหนดเป็นขอบเขตที่ชัดเจนที่ระบบสามารถกระทำได้
2. เวลาในการพัฒนา โดยแบ่งโครงการออกเป็นกิจกรรมย่อยๆ กำหนดระยะเวลาในแต่ละกิจกรรม วันที่ส่งมอบระบบ โดยใช้แกนต์ชาร์ต (Gantt chart) เป็นเครื่องมือแสดงกิจกรรมต่างๆทำให้ลูกค้าสามารถเข้าใจขั้นตอนการพัฒนาโครงการ กิจกรรมที่ต้องกระทำก่อนหรือกระทำพร้อมๆกัน เป็นต้น
3. การจัดโครงสร้างทีมงานในการพัฒนา โดยระบุบุคลากรในทีมงานทั้งหมด หน้าที่ความรับผิดชอบของแต่ละบุคคล รวมทั้งผู้ที่มีอำนาจตัดสินใจ
4. รายละเอียดทางเทคนิคของระบบ โดยกำหนดถึงฮาร์ดแวร์และซอฟต์แวร์ที่ใช้โครงการ เช่นอุปกรณ์ที่ใช้ในการป้อนข้อมูล ชนิดของคอมพิวเตอร์ที่ใช้ ตัวแปลภาษาที่ใช้ การปฏิสัมพันธ์กับระบบอื่นๆ ความสามารถของเทอร์มินัล รวมทั้งข้อกำหนดของเวลาในการตอบสนอง เวลาในการประมวลผล เป็นต้น

5. มาตรฐานของโครงการ กระบวนการหรือวิธีการปฏิบัติ เช่น
 - วิธีการที่ใช้ในการออกแบบระบบ รวมทั้งอัลกอริทึม หรือเทคนิคที่ใช้
 - วิธีการที่ใช้ในการออกแบบโปรแกรม รวมทั้งภาษาออกแบบโปรแกรม
 - วิธีการที่ใช้ในการเขียนโปรแกรม รวมทั้งภาษาโปรแกรมที่เลือกใช้
 - วิธีการที่ใช้ในการทดสอบ
6. แผนการประกันคุณภาพ เหมาะสำหรับโครงการขนาดใหญ่เพื่อเป็นการรับประกันถึงคุณภาพของฮาร์ดแวร์และซอฟต์แวร์
7. เครื่องมือหรือเทคนิคที่ใช้ในการพัฒนาพิเศษ โดยพัฒนาถึงความจำเป็นของเครื่องมือต่างๆเหล่านี้ที่นักออกแบบ หรือโปรแกรมเมอร์ หรือผู้ฝึกอบรม หรือผู้ทดสอบระบบจำเป็นต้องใช้เพื่อช่วยในการพัฒนาโครงการ
8. แผนการจัดการโครงร่าง เป็นการเก็บประวัติการแก้ไขที่เกิดขึ้นในการพัฒนา เพื่อประโยชน์ในการติดต่อสื่อสารระหว่างกันในที่งาน ในกรณีที่ลูกค้ามีความต้องการที่เปลี่ยนไปจากเดิม
9. แผนงานการจัดทำเอกสาร โดยกำหนดเอกสารที่ต้องจัดทำทั้งหมดในโครงการ กำหนดผู้รับผิดชอบในการจัดทำ เวลาที่เริ่มจัดทำ เวลาที่ต้องจัดทำเสร็จ และเมื่อมีการเปลี่ยนแปลงเกิดขึ้นกำหนดผู้รับผิดชอบในการปรับปรุงเอกสารให้ถูกต้องด้วย
10. แผนการจัดการข้อมูล อธิบายถึงวิธีการรวบรวมข้อมูล การจัดการกับข้อมูลเหล่านี้ เช่น การบันทึก การเก็บ การสืบค้น เป็นต้น
11. แผนการจัดการทรัพยากร อธิบายถึงสื่อที่ใช้ในการจัดเก็บ จำนวนข้อมูลทั้งหมดที่ต้องจัดเก็บ การสำรองข้อมูล เป็นต้น
12. แผนการทดสอบ โดยอธิบายถึงวิธีการสร้างข้อมูลทดสอบ วิธีการในการทดสอบ โปรแกรมย่อยต่างๆ วิธีการทดสอบระบบ เป็นต้น
13. แผนการฝึกอบรม โดยอธิบายถึงการฝึกอบรมที่ต้องกระทำในชั้นเรียน กำหนดเวลา และอธิบายถึงความจำเป็นในกรณีที่ต้องมีการฝึกอบรมหลายระดับ
14. แผนการรักษาความปลอดภัย อธิบายถึงการสร้างรหัสผ่านสำหรับผู้ใช้ในระดับต่างๆเพื่อให้เข้าถึงข้อมูล หรือเอกสาร ได้ตามความจำเป็น

15. แผนการบำรุงรักษา โดยอธิบายถึงความรับผิดชอบที่ต้องกระทำหลังจากระบบเสร็จสมบูรณ์แล้ว ในแง่ของการปรับเปลี่ยนคำสั่งโปรแกรม การปรับเปลี่ยนฮาร์ดแวร์ หรือการแก้ไขข้อผิดพลาดที่อาจเกิดขึ้นในการทำงาน
- แผนงานนี้ใช้แสดงภาพรวมของการทำงานของระบบเพื่อช่วยให้เกิดความชัดเจนก่อนที่จะเริ่มการพัฒนาโครงการ

2.6

กรณีศึกษา

สมมติว่าผู้จัดการโครงการในระบบงานหนึ่งมีการแบ่งกิจกรรมในการพัฒนาเป็น 3 ระยะดังนี้

Phase 1: สร้างฐานข้อมูล

Phase 2: พัฒนาแผนงานการบำรุงรักษาระบบ

Phase 3: พัฒนาแบบจำลองระบบย่อย

และสามารถแบ่งการทำงานในระยะแรกออกเป็นขั้นตอนต่างๆได้ดังนี้

Step 1.1 : สร้างพจนานุกรมข้อมูล

Step 1.2 : กำหนดความสัมพันธ์ระหว่างข้อมูล

Step 1.3 : สร้างตารางฐานข้อมูล

Step 1.4 : สร้างหน้าจอในการป้อนข้อมูล

Step 1.5 : สร้างภาษาสอบถามข้อมูล

Step 1.6 : สร้างรายงาน

ในขั้นตอนที่ 1.1 เป็นกิจกรรมต่างๆที่ต้องกระทำได้ดังนี้

Activity 1.1.1 : รวบรวมข้อมูลทั้งหมด

Activity 1.1.2 : กำหนดชนิดของข้อมูล ความยาว แหล่งข้อมูล เงื่อนไข

Activity 1.1.3 : จัดความซ้ำซ้อนของข้อมูล

Activity 1.1.4 : สร้างพจนานุกรมข้อมูล

ซึ่งผู้จัดการโครงการต้องพิจารณาถึงกิจกรรมที่ต้องกระทำในขั้นตอนต่างๆที่เลือกว่ามีกิจกรรมอะไรบ้าง ซึ่งกิจกรรมต่างๆที่ต้องกระทำต้องใช้ประสบการณ์ของผู้บริหารโครงการถ้ายิ่งแบ่งได้ละเอียดมากก็จะทำให้การประมาณการมีความคลาดเคลื่อนน้อยลง สามารถควบคุมกิจกรรมต่างๆได้อย่างละเอียด นอกจากนี้ต้องพิจารณาด้วยว่ากิจกรรมใดบ้างที่ขึ้นกับกิจกรรมอื่นกล่าวคือก่อนจะเริ่มปฏิบัติได้ต้องรอให้กิจกรรมที่เกิดก่อนหน้าเสร็จสมบูรณ์ก่อน รวมทั้งพิจารณาด้วยว่ากิจกรรมใดบ้างเป็นอิสระไม่ขึ้นกับกิจกรรมใดๆสามารถเริ่มทำงานได้พร้อมๆกับกิจกรรมอื่นได้ ทำให้ผู้บริหารโครงการสามารถจัดการในการเริ่มต้นกิจกรรมต่างๆได้อย่างมีประสิทธิภาพ การพิจารณาต้องกระทำไปเป็นลำดับพร้อมทั้งประมาณการเวลาที่ใช้ในแต่ละกิจกรรมด้วยว่าใช้เวลาอย่างน้อยเท่าใดในที่นี่แสดงตัวอย่างเฉพาะ Phase 1 ดังมีรายละเอียดดังนี้

กิจกรรม	กิจกรรมที่กระทำก่อน	เวลาที่ใช้
1.1.1	-	10
1.1.2	1.1	15
1.1.3	1.2	5
1.1.4	1.3	10
1.2.1	1.4	10
1.3.1	2.1	10
1.3.2	3.1	16
1.4.1	2.1	7
1.4.2	4.1	2
1.4.3	4.2	5
1.4.4	4.3	3
1.5.1	4.2	5
1.5.2	5.1	6
1.5.3	5.2	3

1.6.1	4.2	2
1.6.2	6.1	7
1.6.3	6.2	1

ต่อจากนั้นนำความสัมพันธ์ต่างๆของกิจกรรมไปสร้างเป็นกราฟกิจกรรมเพื่อแสดงให้เห็นถึงความสัมพันธ์ระหว่างกิจกรรมต่างๆให้ชัดเจนขึ้น กำหนดหลักชัยและระยะเวลาที่ใช้ในการพัฒนาของแต่ละกิจกรรม ซึ่งกราฟนี้จะทำให้เราทราบถึงกิจกรรมที่ต้องควบคุมเพราะเป็นเส้นทางวิกฤต รวมทั้งกิจกรรมที่สามารถเริ่มปฏิบัติกรเข้าได้ พร้อมทั้งสามารถทราบเวลาทั้งหมดในการพัฒนาโครงการได้

จากกราฟกิจกรรมนี้เราสามารถประมาณการเวลาทั้งหมดที่ใช้ในโครงการตัวอย่างนี้เท่ากับ 76 วันเนื่องจากเป็นเส้นทางที่ยาวที่สุด ส่วนเส้นทางอื่นๆนั้นสามารถเริ่มช้ากว่ากำหนดได้ การประมาณการค่าใช้จ่ายนี้เราสามารถให้แบบจำลองอัตโนมัติเพื่อช่วยในการประมาณการได้ สมมุติว่าเลือกใช้แบบจำลองโคโคโม ผู้บริหารโครงการต้องประมาณการเวลาจำนวนพันบรรทัดที่สามารถแปลเป็นภาษาเครื่องได้ในที่นี้สมมุติว่ามีค่าเท่ากับ 10000 บรรทัดคำสั่ง ต้องเลือกภาวะในการพัฒนาในที่นี้เลือก เซไมติเท็ด และต้องพิจารณาตัวคุณความพยายามทั้ง 15 ลักษณะที่ส่งผลกระทบต่อการพัฒนา ดังมีรายละเอียดดังนี้

Development Mode: SEMIDETACHED

Thousands of New Source Instructions (KDSI) : 10.0

PRODUCT ATTRIBUTES

RELY Required Software Reliability : NOMINAL

DATA Database Size : VHIGH

CPLX Product Complexity : HIGH

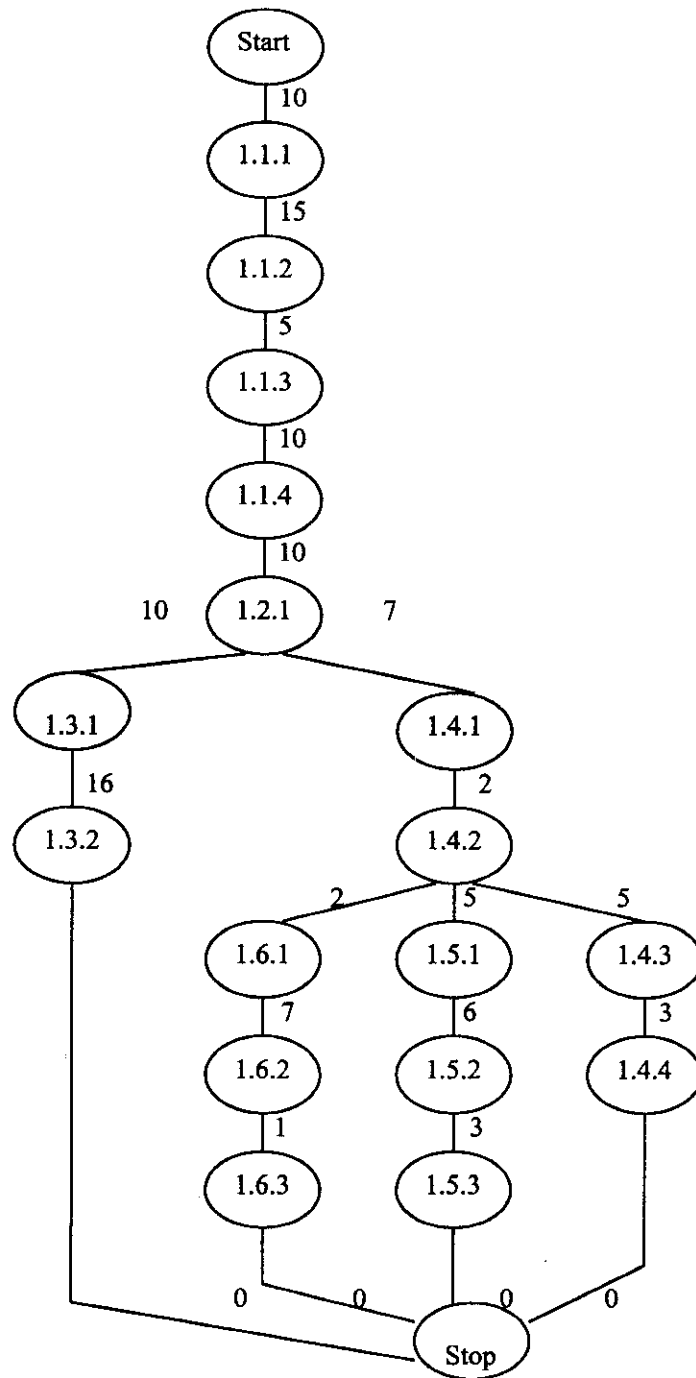
COMPUTER ATTRIBUTES

TIME Execution Time Constraint : NOMINAL

STOR Main Storage Constraint : VHIGH

VIRT Virtual machine Volatility : NOMINAL

TURN Computer Turnaround Time : NOMINAL



PERSONAL ATTRIBUTES

ACAP Analyst Capability : VHIGH

AEXP Applications Experience : HIGH

PCAP Programmer Capability : VHIGH
VEXP Virtual Machine Experience : NOMINAL
LEXP Programming Language Experience : HIGH

PROJECT ATTRIBUTES

MODP Use of Modern programming practices : HIGH
TOOL Use of Software Tools : HIGH
SCED Required Development Schedule : NOMINAL

ผลลัพธ์ของการประมาณการได้ค่าดังนี้

EFFORT : 22.71 man-months
PRODUCTIVITY : 440.28 DSI/man-months
SCHEDULE : 7.46 months
AVERAGE STAFFING : 3.05 FSP(full-time equivalent software personnel)

แต่ถ้าทีมงานไม่มีความรู้หรือไม่มีประสบการณ์ในภาษาโปรแกรมและงานประยุกต์

PERSONAL ATTRIBUTES

ACAP Analyst Capability : VHIGH
AEXP Applications Experience : LOW
PCAP Programmer Capability : NOMINAL
VEXP Virtual Machine Experience : NOMINAL
LEXP Programming Language Experience : VLOW

ค่าของการประมาณการจะสูงขึ้นดังนี้

EFFORT : 68.10 man-months
PRODUCTIVITY : 146.85 DSI/man-months
SCHEDULE : 10.95 months
AVERAGE STAFFING : 6.22 FSP(full-time equivalent software personnel)

สรุป

ในบทนี้กล่าวถึงการวางแผนโครงการ ซึ่งเป็นหน้าที่ของผู้บริหารโครงการที่จะต้องกระทำ ก่อนพัฒนาโครงการ การดำเนินงานใดๆถ้ามีการวางแผนที่ดีจะลดความเสี่ยงต่อความล้มเหลว การวางแผนโครงการนั้นจะเน้นถึงการควบคุมและการวัดความก้าวหน้าในการพัฒนามีเครื่องมือที่นิยมใช้คือ work breakdown structure ซึ่งสามารถกระทำได้โดยใช้แรงงานมนุษย์ หรือใช้โปรแกรมอัตโนมัติก็ได้ โดยเครื่องมือนี้แบ่งโครงการออกเป็นกิจกรรมต่างๆที่ต้องกระทำมีการประมาณการเวลาที่ใช้ในแต่ละกิจกรรม วิเคราะห์ถึงเวลาที่ต้องเริ่มปฏิบัติในกิจกรรมต่างๆเพื่อให้ผู้บริหารโครงการสามารถจัดการหรือแก้ปัญหาที่อาจเกิดขึ้นได้ทันเวลาที่ ทั้งยังสามารถประมาณการเวลาและค่าใช้จ่ายรวมของโครงการได้อีกด้วย นอกจากนี้ได้กล่าวถึงแบบจำลองประมาณการซึ่งเป็นสมการเอ็กซ์โปเนนเชียล ซึ่งเกิดจากการรวบรวมข้อมูลของโครงการที่เสร็จสมบูรณ์แล้วนำมาวิเคราะห์เป็นแบบจำลองเพื่อสนับสนุนในการตัดสินใจของผู้บริหารโครงการ เช่น แบบจำลองโคโคโม เป็นต้น นอกจากนี้ยังมีปัจจัยอีกหลายอย่างที่ส่งผลต่อค่าใช้จ่ายของโครงการ อาทิเช่น การจัดโครงสร้างของทีมงาน ความซับซ้อนของโครงการ เครื่องมือและเทคนิคที่ใช้ในโครงการ ฯลฯ ดังนั้นการประมาณการเวลาและค่าใช้จ่ายที่ได้ค่าที่ใกล้เคียงกับความเป็นจริงต้องนำปัจจัยต่างๆเหล่านี้มาคิดคำนวณด้วย สำหรับในประเทศไทยของเราแบบจำลองต่างๆของต่างประเทศไม่เหมาะสำหรับใช้ประมาณการในบ้านเรา เพราะมีสภาพแวดล้อมในการทำงานแตกต่างกัน ค่าที่ได้จะคลาดเคลื่อนสูงมากๆ ดังนั้นในแต่ละหน่วยงานต้องมีการเก็บข้อมูลของโครงการที่เสร็จสมบูรณ์แล้วของแต่ละหน่วยงาน นำมาวิเคราะห์วิจัย เพื่อหาค่าสมการที่เหมาะสมกับหน่วยงานนั้นๆ หรือใช้สำหรับเปรียบเทียบค่าใช้จ่ายสำหรับโครงการที่คล้ายๆกัน ซึ่งช่วยเป็นข้อมูลสำหรับตัดสินใจของผู้บริหารของหน่วยงาน ถือว่าเป็นทางเลือกหนึ่งในการประมาณการค่าใช้จ่าย

2.8

แบบฝึกหัด

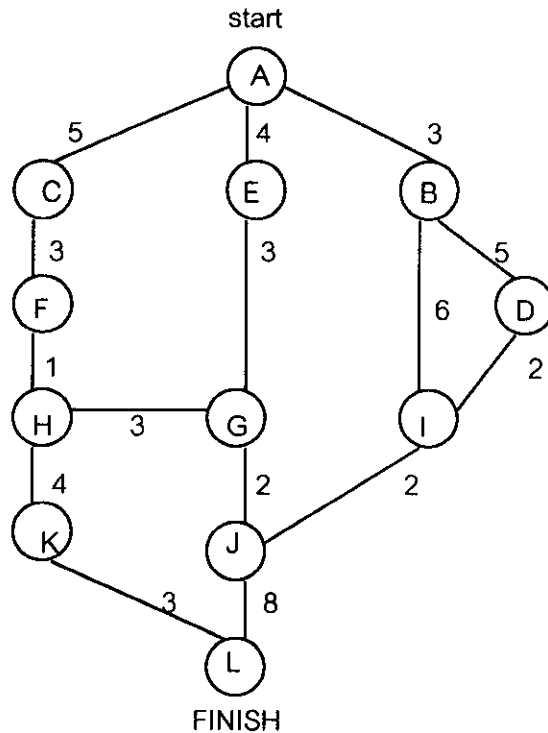
1. ในการวางแผนงานโครงการหนึ่ง หัวหน้าโครงการได้แบ่ง Activities , Time Estimates และ Precursor (เหตุการณ์ที่เกิดก่อนการเริ่มต้นกิจกรรม) ดังนี้

Activities	Time Estimate(days)	Precursor
Step1:		
Activity 1.1	3	-
Activity 1.2	8	1.1
Activity 1.3	7	1.2
Step2:		
Activity 2.1	3	1.1
Activity 2.2	5	1.1
Activity 2.3	8	2.1 , 2.2
Activity 2.4	4	2.3
Activity 2.5	6	1.3 , 2.4
Step3:		
Activity 3.1	4	2.5
Activity 3.2	5	2.5
Activity 3.3	3	3.1
Activity 3.4	4	3.2, 3.3

- 1.1 จงเขียน Activity Graph และ เวลาของกิจกรรมต่างๆ ของโครงการนี้
1.2 จงเขียนตาราง Slack Time ของกิจกรรมต่างๆ ของโครงการนี้ ประกอบด้วย

Activity , Earliest Start Time , Lastert Start Time และ Slack จนวนจบโครงการ

2. จงหา Critical path ของ Activity graph ต่อไปนี้



2.9

ปฏิบัติ

สำหรับวิชานี้ถ้าจะเรียนให้ได้ประโยชน์มากที่สุด นักศึกษาต้องมีการพัฒนาซอฟต์แวร์ระบบงานขึ้นมา โดยใช้ความรู้ทางทฤษฎีไปประยุกต์ใช้กับระบบงาน ดังนั้นการปฏิบัติงานสำหรับบทนี้ให้ทำกรณีศึกษาโดยใช้ระบบงานเดิมจากบทที่ 1 เป็นงานที่เราจะพัฒนาซอฟต์แวร์ที่มีคุณภาพกันให้นักศึกษาทำแผนโครงการโดยจัดทำเป็นเอกสารส่งอาจารย์อื่นประกอบด้วย

1. ขอบเขตของโครงการ
2. ระยะเวลาของโครงการ
3. การจัดโครงสร้างของทีมงาน
4. รายละเอียดทางเทคนิค
5. วิธีการ หรือกระบวนการ มาตรฐานที่ใช้ในโครงการ
6. การประกันคุณภาพ
7. เครื่องมือหรือเทคนิคที่ใช้ในการพัฒนาพิเศษ
8. การวางแผนการจัดการโครงร่าง
9. การวางแผนเอกสาร
10. การวางแผนการจัดการข้อมูล
11. การวางแผนการจัดการทรัพยากร
12. การวางแผนการทดสอบ
13. การวางแผนฝึกอบรม
14. การวางแผนเรื่องความปลอดภัย
15. การวางแผนการบำรุงรักษาระบบ

ในการวางแผนงานส่วนนี้อาจช่วยกันทำหรือเลือกสมาชิกคนใดคนหนึ่งคิดว่าเก่งมีความรู้มากที่สุดหรือมีความเป็นผู้นำให้เป็นหัวหน้าโครงการ เพื่อจัดทำแผนโครงการ สำหรับโครงการขอให้ความซับซ้อนอยู่ในเกณฑ์ปานกลางที่สามารถให้โครงการเสร็จสมบูรณ์ได้ภายในระยะเวลา 3 เดือน

