รายการอ้างอิง

1. วิโรจน์ อัศวรังสี และ พุทธพร แสงรัตนเดช. คอมพิวเตอร์ ทำงานอย่างไร. กรุงเทพมหานคร: บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2538.
2. Randall W. Jensen และ Charles C. Tonies. Software Engineering. New Jersey: Prentice – Hall, Inc.,1979.
3. Stephen R. Schach. Software Engineering. Richard D. Irwin,Inc. and akenassociates,inc., 1990.
4. Robert L. Glass. Building quality Software. New Jersey : Prentice-hall, Inc.,1992.
5. Michael G. Jenner. Software Quality Management and ISO 9001. John Wiley&Sons, Inc., 1995.
6. Ian Sommerville. Software Engineering, Fourth Edition. Addison-Wasley Publishing.,1993.
7. Standards Coordinating committee of the IEEE Computer Society. IEEE Standard Computer Dictionary. New York : The Institute of Electrical and Electronics Engineering,1990.
8. W.S. Humphrey and W.L. Sweet. A Method for Assessing the software Engineering Capability of contractors. Pennsylvania : software Engineering Institute, Carnegie – mellon University, 1987.
9. Taratip Suwannasart. Towards Development of a Testing Maturity Model. Ph.D. dissertation, Illinois Institute of Technology, 1996.
10. Robert N. Charette. Software Engineering Environments. New York : McGraw – Hill, Inc.,1987.
11. General Electric Company.,Corporate Information System. Software Engineering Handbook. NewYork : : McGraw-Hill Book Company, 1996.
12. Ian Sommerville. Software engineering. London : addison – Wesley Publishers, 1982.
13. Roger S. Pressman. A Manager' s Guide to Software Engineering. NewYork : McGraw – Hill Companies, Inc,. 1993.
14. David J. Smith. Achieving Quality Software, Third Edition. London : Chapman&Hall,1995.
15. E.M. Bennatoan. On Time Within Budget, second Edition. John Wiley&Sons, Inc., 1995.
16. W.S. Humphrey. Managing The Software Process. addison – Wesley, 1989.
17. W.S. Humphrey. Introduction to Software Process Improvement. Software Engineering Institute, CMU/SEI-92-TR-7,1992.
18. M.C. Paulk, B.Curtis and M.B. Chrissis. Capability Maturity Model for Software. Software Engineering Institute, CMU/SEI-91-TR-24,1991.

19. M.C. Paulk, B.Curtis, M.B. Chrissis and Charles V.Weber. Capability Maturity Model for Software, Version 1.1. Software Engineering Institute,CMU/SEI-93-TR-24,1993.

20. M.C. Paulk, C.V.Weber, S.Garcia, M.B. Chrissis. And M.Bush. Key Practices of the Capability Maturity Model,version1.1. Software Engineering Institute, CMU/SEI-93TR-25,1993.

21. Mark C.Pualk, Bill Curtis, Mary Beth Chrissis and Charles V. Weber. Capability Maturity Model, Version 1.1. IEEE Software July 1993 : 18-27.

22. David Rugg. Using Capability Evaluation to Select a Contractor. IEEE Software July 1993 : 36-45.

23. Terry B. Bollinger and clement MC Gowan. A Critical Look at Software Capability Evaluation. IEEE Software July 1991 : 25-41.

ภาคผนวก

ภาคผนวก ก.

1. <u>รายละเอียดข้อเสนอสิ่งที่ต้องการ (Request for Proposal - RFP)</u> เป็นเอกสารทางด้านเทคนิค รวบรวมโดยผู้ใช้งาน เพื่อคัดเลือกบริษัทผู้พัฒนาซอฟต์แวร์ (กรณีที่โครงการติดต่อกับผู้พัฒนาภายนอก) ประกอบด้วยรายละเอียดต่างๆ ดังนี้

   1.1 <u>คำชี้แจงปัญหาและวัตถุประสงค์ของโครงการ</u> ได้แก่

   - คำอธิบายรายละเอียดสถานการณ์ในปัจจุบันของระบบงานที่ดำเนินอยู่
   - คำอธิบายรายละเอียดของปัญหาต่างๆ ที่เกิดขึ้น
   - เอกสารต่างๆ เพื่อช่วยสนับสนุนงาน เช่น รายงาน ผังโครงสร้าง ตัวอย่างต่างๆ
   - วัตถุประสงค์ของโครงการ

   1.2 <u>ความต้องการทางด้านเทคนิค</u> ได้แก่

   - การเชื่อมโยงระบบใหม่ กับระบบเดิมที่มีอยู่
   - รายละเอียดฐานข้อมูลที่ต้องการ
   - โครงสร้างสถาปัตยกรรมของระบบเครือข่าย และการติดต่อสื่อสาร
   - มาตรฐานของรัฐบาล หรือ หน่วยงานที่เกี่ยงข้อง
   - ความเชื่อถือได้ในความถูกต้องของระบบ
   - ข้อจำกัดด้านเวลา (Timing Constraints)
   - ภาษาโปรแกรม (Programming language) ที่ใช้
   - คอมพิวเตอร์หลัก (Host Computer) ที่ต้องการ

   1.3 <u>ข้อมูลทั่วไป</u> ได้แก่

   - ใครควรเป็นผู้ตอบคำถามใน RFP
   - จะต้องทำอย่างไรจึงจะได้ข้อมูลเพิ่มเติม หรือ ข้อมูลที่ชัดเจน
   - วันที่ และสถานที่ ของตารางการประชุมที่จะพบกับผู้เสนอข้อมูลที่คาดว่าเหมาะสม
   - เกณฑ์ในการคัดเลือกบริษัทผู้พัฒนาซอฟต์แวร์
   - ข้อมูลทั่วไปอื่นๆ

1.4 <u>ความต้องการทางด้านต้นทุน</u> ได้แก่

- โครงสร้างราคาที่ใช้ (สำหรับบริการ , ตัวผลิตภัณฑ์ , การจัดหา)
- ความสมเหตุสมผลของต้นทุนต่างๆ (รายละเอียดการคิดต้นทุน)
- การแบ่งราคาในแต่ละขั้นตอนงาน
- ประเภทของสัญญาในการพฒนาซอฟต์แวร์ที่เสนอ
- การเปรียบเทียบต้นทุนในแต่ละทางเลือกที่จะสามารถแก้ปัญหาได้

1.5 <u>เอกสารต่าง ๆ ที่เกี่ยวข้อง</u> ได้แก่

- เอกสารมาตรฐานต่างๆ ที่ต้องมี
- เอกสารต่างๆ ที่ระบบเดิมมีอยู่
- รายละเอียดของผลิตภัณฑ์

1.6 <u>สิ่งที่ต้องการให้ส่งมอบ</u> ได้แก่

- เอกสารต่างๆ
- ซอฟแวร์
- การฝึกอบรม
- ฮาร์ดแวร์ หรือ อุปกรณ์ที่เกี่ยวข้อง
- การรับประกันสำหรับระบบการจัดส่ง
- เครื่องมือที่ใช้ในการพัฒนาซอฟต์แวร์ และเครื่องมือที่ใช้ทดสอบ

1.7 <u>รูปแบบของรายละเอียดข้อเสนอ</u> ได้แก่

- รายละเอียดข้อเสนอทางด้านเทคนิค
- รายละเอียดข้อเสนอทางด้านการจัดการ
- รายละเอียดข้อเสนอทางด้านราคา
- ขอบเขตของงานที่ต้องทำ
- รายละเอียดประกอบอื่นๆ เช่น งบการเงินของบริษัทผู้ยื่นข้อเสนอข้อมูลทางด้านเทคนิคของผู้ยื่นข้อเสนอ รายละเอียดประวัติการศึกษา และประวัติการทำงานของบุคคลที่จะเข้ามารับผิดชอบโครงการของบริษัทผู้ยื่นข้อเสนอ

1.8 <u>ตารางเวลาในการยื่นข้อเสนอ และการตัดสินใจคัดเลือก</u> ได้แก่

- วันที่วันสุดท้ายที่จะรับข้อเสนอของแต่ละบริษัท
- วันที่ที่คาดว่าจะตัดสินใจคัดเลือก
- ตารางเวลาที่ต้องการจะให้ระบบงานพัฒนาเสร็จ

<u>คำชี้แจง</u>

วัตถุประสงค์ ของการจัดทำคำขอให้เสนอราคา "Request for Proposal" (RFP) ขึ้นเพื่อความสะดวกในการที่บริษัทฯ จะได้สามารถคัดเลือกบริษัทผู้ขายที่มีคุณสมบัติเหมาะสมที่สุดในการเสนอระบบโปรแกรมคอมพิวเตอร์ทั้งระบบตามความต้องการแก่บริษัทฯ ได้เป็นอย่างดี

<u>เงื่อนไขและหลักการ</u>

RFP ฉบับนี้มีวัตถุประสงค์ เพื่อเป็นการตรวจสอบถึงคุณสมบัติ และความสามารถของระบบงานที่บริษัทฯ ผู้ขายได้เสนอขาย.....

.....

.....

<u>การเปิดเผยข้อมูล</u>

การออกหนังสือ RFP ของบริษัทถือเป็นความลับของบริษัท ซึ่งผู้ขายสัญญาว่าจะไม่ทำให้บริษัทได้รับผลกระทบกระเทือนต่อชื่อเสียง หรือทางการค้า .....

.....

<u>การจัดส่งคำเสนอ</u>

ผู้ขายจะต้องจัดส่งคำเสนอให้แก่บริษัทเป็นอย่างช้าภายในวันที่ .....

เวลา ..... โดยส่งคำเสนอมาที่คุณ .....ฝ่าย.....

ที่อยู่ .....ถนน.....แขวง.....

เขต..... กรุงเทพฯ ..... โทร. .....

<u>รูปแบบของการตอบ RFP</u>

ผู้ขายจะต้องตอบรายละเอียดตามข้อกำหนดความต้องการของบริษัทอย่างละเอียดในแต่ละข้อด้วยการระบุเพียงหมายเลข 1 หรือ 2

ซึ่ง 1 หมายถึง "ได้" คือคำตอบที่โปรแกรมระบบงานของผู้ขายสามารถปฏิบัติเป็นไปตามความต้องการของบริษัทได้อย่างถูกต้อง

2 หมายถึง "ไม่ได้" คือคำตอบว่าโปรแกรมระบบงานของผู้ขายกำลังอยู่ในระหว่างการแก้ไขปรับปรุงอยู่ ผู้ขายจะต้องระบุวันที่ ที่คาดว่าจะสามารถติดตั้งระบบใหม่นี้ได้ ต่อท้ายหมายเลขที่ตอบด้วย

การเจรจาต่อรอง

ผู้ขายจะต้องระบุถึงขอบเขตความรับผิดชอบที่ผู้ขายต้องการให้บริษัทรับผิดชอบในหน้า
ที่ใดบ้างผู้ขายจะต้องกำหนดค่าใช้จ่ายและค่าบริการต่าง ๆ พร้อมด้วยใบเสนอราคา.....

ข้อมูลเกี่ยวกับบริษัท

บริษัท .....จัดตั้งขึ้นเมื่อ.....
มีทุนจดทะเบียน .....
ปัจจุบันมีทุนจดทะเบียน .....
มียอดขายกว่า .....
มีพนักงานทั้งสิ้น .....

ข้อมูลการขายของบริษัท

บริษัท เป็นผู้จัดจำหน่าย.....โดยมีผลิตภัณฑ์สินค้าหลักดังนี้

1. .....
2. .....

เครื่องคอมพิวเตอร์ของบริษัทในขณะนี้

CPU Type	Hard disk	Brand	Unit
.....	.....	.....	.....
.....	.....	.....	.....
.....	.....	.....	.....

ระบบงานที่ใช้คอมพิวเตอร์ของบริษัทในขณะนี้

1. ระบบบัญชี
2. ระบบขายสด / เชื่อ

ข้อมูลเกี่ยวกับระบบงานของบริษัท

1. ระบบบัญชี
2. ระบบขายอะไหล่

ข้อมูลรายการในแต่ละระบบงาน

1. ระบบบัญชี

จำนวนบัญชี	.....	บัญชี
รายการเฉลี่ยต่อวัน	.....	รายการ
รายการสูงสุดต่อวัน	.....	รายการ
รายการต่ำสุด	.....	รายการ

1. คุณสมบัติโดยทั่วไปของระบบ

โปรแกรมทุกระบบงานที่ผู้ขายจะเสนอนั้นจะต้องเป็นโปรแกรมระบบงานที่สามารถเชื่อมโยงเข้าด้วยกันได้ โดยมีฐานข้อมูลเพียงหนึ่งเดียวเท่านั้น เพื่อลดความซ้ำซ้อนในการปฏิบัติงาน โดยมีรายละเอียดต่าง ๆ อย่างน้อยดังนี้

1.1 โปรแกรมทุกระบบจะต้องเป็นระบบ Realtime Interactive ยกเว้นในระบบ ..... บัญชีที่อนุญาติให้เป็น Batch ได้บ้างในบาง Module

1.2 ระบบงานจะต้องสามารถให้ป้อนข้อมูลย้อนหลังได้โดยมีระบบการป้องกัน ..... อย่างเพียงพอ

17. รายละเอียดของระบบงานแบบเบ็ดเสร็จ (Turnkey)

17.1 บริษัทต้องการให้ผู้ขายทำการแก้ไขโปรแกรมหรือดัดแปลงโปรแกรมระบบงานเพื่อให้เข้ากับความต้องการของบริษัทตามที่ได้ระบุไว้ใน RFP นี้ พร้อมกับการให้บริการบำรุงรักษาระบบดังกล่าวอย่างต่อเนื่องแก่บริษัท

17.2 ผู้ขายจะต้องส่งรายละเอียดเกี่ยวกับการแก้ไขโปรแกรมโดยละเอียด ราคา ระยะเวลา การแก้ไขพร้อมกับตอบ RFP นี้

18. ข้อมูลเกี่ยวกับโปรแกรมระบบงาน

18.1 วันที่ที่พัฒนาโปรแกรมระบบงาน .....

18.2 วันที่ที่ทำการแก้ไขโปรแกรมระบบงานครั้งล่าสุด .....

18.6 โปรแกรมระบบงาน และระบบฐานข้อมูลที่ใช้

18.6.1 ของ Oracle .....

18.6.2 ของ Progress .....

18.6.3 ของ Sybase .....

18.6.4 ของ Informix .....

18.6.5 ของ Ingres .....

18.6.6 อื่น ๆ .....

18.7 ภาษาที่ใช้ในการพัฒนาระบบงาน .....

18.8 คู่มือเอกสารระบบงานที่จัดให้บริษัท

18.8.1 คู่มือการติดตั้งระบบ .....

18.8.2 คู่มือการปฏิบัติระบบงาน .....

18.8.3 รายละเอียดเกี่ยวกับระบบ System Specification .....

18.8.4 รายละเอียดเกี่ยวกับโปรแกรม Program Specification.....

18.8.5 คู่มือระเบียบปฏิบัติงานของผู้ใช้ .....

18.8.6 ระเบียบกฎเกณฑ์เกี่ยวกับวิธีการพิสูจน์และการ Balancing .....

18.8.7 ขอสำเนาคู่มือดังกล่าวมากกว่า 1 ชุด .....

18.8.8 การปรับปรุงเกี่ยวกับคู่มือให้ทันสมัยขึ้น ผู้ขายยินดีจัดส่งคู่มือที่แก้ไขเปลี่ยนแปลงนี้แก่บริษัทโดยไม่คิดค่าใช้จ่ายใด ๆ ทั้งสิ้นอีก .....

18.9 จำนวนพนักงานของบริษัท

18.9.1 ในประเทศไทย .....

18.9.2 ในแถบทวีปเอเชีย .....

18.9.3 ทั่วโลก .....

18.16 ผู้ขายจะส่งมอบ Source Code แก่บริษัท

18.16.1 ราคาที่เสนอนี้ได้รวม Source Code ด้วย .....

18.16.2 หากไม่รวมถ้าบริษัทต้องการ Source Code ..... ผู้ขายยินดีที่จะกำหนดราคาไว้ (โปรดระบุราคาด้วย)

19. การบริการหลังการขาย

19.1 ระยะเวลาที่ผู้ขายจะดำเนินแก้ไขโปรแกรมที่มีข้อผิดพลาดหรือ .....
กรณีที่มีปัญหา (ที่บริษัท.....) ได้โดยเร็วที่สุดกี่ ชม., นาที

20. คุณสมบัติของผู้ขาย

20.1 เพื่อให้การพิจารณาคัดเลือกผู้ขายสามารถดำเนินการไปด้วยความ .....
รวดเร็วบริษัทใคร่ขอให้ผู้ขายจัดส่งรายละเอียดต่าง ๆ เกี่ยวกับตัวบริษัท
ผู้ขายตลอดจน อำนาจความรับผิดชอบที่ผู้ขายได้มีสำหรับการจัดจำหน่าย
โปรแกรมระบบงานที่ผู้ขายได้จัดเสนอขายนี้

20.2 ข้อมูลที่ผู้ขายจะจัดส่งให้แก่บริษัทควรจะมีรายละเอียดอย่างน้อยดังนี้

20.2.1 รายละเอียดเกี่ยวกับงบกำไรขาดทุน และงบดุลของบริษัท .....
ที่ผ่านการรับรองและตรวจสอบแล้วย้อนหลัง 2 ปี

20.2.2 รายละเอียดประสบการณ์ที่ผู้ขายมีเกี่ยวกับสินค้านี้ .....

20.2.3 โครงสร้างและรูปแบบองค์กรของผู้ขาย .....

21. ราคาการลงทุน

ผู้ขายจะต้องเสนอราคาการลงทุนและค่าใช้จ่ายโดยละเอียดเกี่ยวกับระบบงานดังกล่าวข้างต้นทั้งหมดในลักษณะเป็นแบบเบ็ดเสร็จทั้งหมด โดยที่ผู้ขายจะต้องจัดเตรียมและจัดหาผู้เชี่ยวชาญในการติดตั้งระบบตลอดจนผู้บริหารโครงการ เพื่อให้การติดตั้งระบบงานดังกล่าวข้างต้นเป็นไปโดยรวดเร็วที่สุดอีกด้วย

21.1 ระบบงานบัญชี ราคา

21.1.1 ราคาค่าโปรแกรมครั้งแรก .....

21.1.2 ค่าติดตั้งระบบ (รวมถึงค่าบริหารโครงการ และฝึกอบรม) .....

21.1.3 ค่าแก้ไขโปรแกรมและดัดแปลงให้เข้ากับความต้องการของบริษัท.....

รวมค่าใช้จ่ายทั้งสิ้น .....

รวมค่าบำรุงรักษาตลอดปี .....

รูปภาพแสดงรายละเอียดข้อเสนอสิ่งที่ต้องการ

ภาคผนวก ข

2. การควบคุมและการติดตามตรวจสอบการดำเนินงานโครงการซอฟต์แวร์ (Software Project Control and Monitoring)

2.1 การควบคุมการดำเนินงานโครงการซอฟต์แวร์ (Software Project Control) ผู้จัดการโครงการจำเป็นต้องอาศัยเครื่องมือในการจัดการควบคุมโครงการที่มีความเที่ยงตรงแม่นยำเพื่อควบคุมการดำเนินงานในกิจกรรมต่างๆ ของการพัฒนาซอฟต์แวร์

ตัวอย่างของเครื่องมือที่ช่วยในการควบคุมการดำเนินงานได้แก่

1. **เครื่องมือที่ช่วยในการจัดการบริหารรายละเอียดขอบเขตโครงร่างซอฟต์แวร์ (Configuration Management Tools)** เนื่องจากผู้พัฒนาซอฟต์แวร์แต่ละคนมีคอมพิวเตอร์เป็นของตัวเองซึ่งต่อเชื่อมโยงเข้ากับระบบเครือข่าย จึงมีอิสระมากพอในการปรับปรุงแก้ไขโครงสร้างไฟล์ของโครงการทั้งหมด ฉะนั้นเครื่องมือที่จะมาช่วยในการจัดการบริหารรายละเอียดขอบเขตโครงร่างซอฟต์แวร์ จึงจำเป็นต้องจัดทำไลบรารี่ของไฟล์ทั้งหมดเป็นศูนย์กลาง (Central File Library) ซึ่งอาจจะอยู่บน Server และมีการตรวจสอบควบคุมการผ่านเข้าออกไปใช้ไฟล์ต่างๆ ในฐานข้อมูลการพัฒนาซอฟต์แวร์ของโครงการที่อยู่ศูนย์กลางนี้ด้วย
2. **การรายงานผลการดำเนินงานตามช่วงเวลา (Time Reporters)** รายงานนี้จะแสดงให้ทราบว่าโครงการใช้เวลาไปในกิจกรรมการพัฒนาซอฟต์แวร์ขั้นตอนต่างๆ มากน้อยเพียงใด
3. **การติดตามตรวจสอบการดำเนินงานแบบทันทีทันใด (On-Line Task Monitors)** เครื่องมือนี้จะรายงานกิจกรรมต่างๆ และความก้าวหน้าของโครงการที่ทีมงานพัฒนาซอฟต์แวร์ได้ดำเนินการไป ซึ่งอาจแสดงผลการทำงานผ่านทางหน้าจอภาพคอมพิวเตอร์ได้ทันทีทันใด
4. **การเผยแพร่ข้อมูลหรือการได้รับข้อมูลข่าวสารโครงการแบบทันทีทันใด (Information Disseminators)** เป็นสิ่งจำเป็นสำหรับผู้จัดการโครงการ และผู้พัฒนาซอฟต์แวร์ จดหมายอิเล็กทรอนิคส์ (Electronic Mail) เป็นเครื่องมือง่ายๆ ที่มีความสามารถเหมาะสมแก่การนำมาใช้

2.2 การติดตามตรวจสอบการดำเนินงานโครงการซอฟต์แวร์ (Software Project Monitoring) แบ่งออกเป็น 3 หัวข้อใหญ่ๆ ได้ดังนี้ คือ

2.2.1 <u>การรายงานตามช่วงระยะเวลา (Periodic Reports)</u> เป็นวิธีการดำเนินการที่เป็นทางการเพื่อถ่ายทอดข้อมูลการทำงานตามขั้นตอนการทำงานปกติจากทีมงานพัฒนาซอฟต์แวร์ไปยังผู้จัดการโครงการ ซึ่งสามารถแบ่งออกได้เป็นการรายงานแบบต่างๆ ดังนี้

1. รายงานสถานะของการทำงานตามช่วงเวลาแบบจัดทำเป็นลายลักษณ์อักษร (Periodic Written Status Reports)
2. รายงานแบบปากเปล่า (Verbal Reports)
3. การประชุมสถานะของโครงการ (Project Status Meetings)
4. การสาธิตผลิตภัณฑ์ (Product Demonstrations - Demos)

1. **<u>รายงานสถานะของการทำงานตามช่วงเวลาแบบจัดทำเป็นลายลักษณ์อักษร (Periodic Written Status Reports)</u>** เป็นรายงานที่สมาชิกของทีมงานพัฒนาซอฟต์แวร์ทุกคนต้องจัดทำขึ้นเป็นช่วงเวลาตามที่กำหนด อาจเป็นอาทิตย์ละหนึ่งครั้ง หรือ 2 อาทิตย์ครั้งซึ่งในรายงานควรประกอบด้วย 3 ส่วนใหญ่ๆ ดังนี้

1.1 กิจกรรมต่างๆ ที่ดำเนินการในช่วงเวลาที่ผ่านมาก่อนถึงการรายงานงวดปัจจุบัน แต่ละหน่วยงานควรจะเขียนรายงานอธิบายรายละเอียดกิจกรรมต่างๆ ที่ได้กระทำผ่านมาในช่วงระยะเวลาของการรายงานผลการดำเนินงาน ซึ่งอาจจะอธิบายเป็นรายละเอียด 2-3 บรรทัดในแต่ละกิจกรรม ซึ่งกิจกรรต่างๆ ควรจะสัมพันธ์สอดคล้องกับรายละเอียดของงานโครงการ (Project Task List) หรือโครงสร้างงานที่แบ่งเป็นรายละเอียดงานย่อยๆ (Work Breakdown Structure - WBS)

Task ID.	Description	Status	Assigned To	Comments
1.	Management And Administration			
2.	Software Development			
2.1	Software Requirement Analysis			
2.2	Software Design			
2.2.1	Control Logic			
2.2.2	Common Interface			
2.2.3	Display Utilities			
2.2.3.1	Display Formatter	Complete	J. Smith	
2.2.3.2	Screen Driver	Started	F. Brown	
2.2.3.3	Shape Generator	Started	A. Black	

2.2.4	Communications
2.3	Software Coding
2.4	Software Integration
3.	Procurement & Development Support

ตารางแสดงรายละเอียดงานตามโครงสร้างงาน (Work Breakdown Structure Task List)

โครงการซอฟต์แวร์ที่มีความซับซ้อนสามารถแบ่งออกเป็นองค์ประกอบย่อย ๆ ที่ง่ายต่อการจัดการ และการติดตามตรวจสอบการดำเนินงานตามกิจกรรมย่อย ๆ จะสะดวกและง่ายกว่าการจัดการทั้งโครงการ ซึ่งการแบ่งรายละเอียดในแต่ละโครงการ อาจมีวิธีการที่แตกต่างกันไปขึ้นอยู่กับ วัตถุประสงค์ของผู้จัดการโครงการแต่ละคน ซึ่งการรายงานสถานะของการทำงาน และการติดตามตรวจสอบการดำเนินงานจะทำตามรายละเอียดงานตามโครงสร้างงานดังกล่าว

รายละเอียดงานตามโครงสร้างงานแบบละเอียด (High Level Work Breakdown Structure Tasks) เป็นดังนี้คือ

- Software Development
  - Requirements Analysis
- Prototype Development
  - Protype Specification
  - Prototype Design
  - Prototype Implementation
- Design
  - Top Level Design
  - Detailed Design
- Implementation
  - Coding
  - Unit Test
- Integration
  - Software Integration
  - Hardware / Software Integration

- Testing
  - Alpha Testing
  - Beta Testing
  - Acceptance
- Installation
- Maintenance
  - Error Correction
  - Software Enhancement
- Management
  - Planning
  - Staffing
  - Administration And Services
  - Budget Administration
  - Personnel Management
  - Quality Assurance
  - Configuration Management
- Training
- Procurement
  - Acquisition Of Development Tools
  - Acquisition Of System Components (Off The Shelf)
  - Equipment Selection
  - Vendor Selection
  - Ordering Procedure
  - Inventory Control
- Documentation
  - Technical Writing
  - Project Publishing Activities
  - Development Documentation
    - Non- Deliverable Development Documentation
    - Deliverable Development Documentation
  - Maintenance Documentation
  - User Documentation

1.2 <u>กิจกรรมต่าง ๆ ที่วางแผนจะดำเนินการในช่วงเวลาก่อนถึงการรายงานงวดถัดไป</u> แต่ละหน่วยงานควรจะเขียนรายงานอธิบายรายละเอียดกิจกรรมต่าง ๆ ที่วางแผนว่าจะดำเนินการในช่วงระยะเวลาก่อนถึงการรายงานงวดถัดไป ซึ่งอาจจะอธิบายเป็นรายละเอียด 2-3 บรรทัดในแต่ละกิจกรรม

1.3<u>รายละเอียดปัญหาต่าง ๆ ที่เกิดขึ้น</u> แต่ละหน่วยงานควรจะเขียนรายละเอียดปัญหาต่าง ๆ ที่เกิดขึ้นในช่วงเวลาที่ผ่านมา รวมทั้งปัญหาต่าง ๆ ที่เคยถูกรายงานให้ทราบแล้ว แต่ยังไม่ได้ถูกแก้ไข เพราะว่าปัญหาเดิม ๆ เหล่านี้ อาจเกิดขึ้นอีกได้

รายงานเหล่านี้ควรมีรายละเอียดต่าง ๆ ได้แก่

1. วันที่ของรายงาน
2. ช่วงระยะเวลาในการรายงาน เช่น 3 กค. ถึง 10 กค. 2540
3. ชื่อของรายงาน เช่น รายงานสถานะของทีมงานการติดต่อสื่อสาร
4. ชื่อของ

การจัดเตรียมรายงานสถานะการทำงานตามงวดเวลา (Periodic Status Report) นี้ควรใช้เวลาประมาณ 20-30 นาที ผู้พัฒนาซอฟต์แวร์แต่ละคนควรเขียนรายงานส่งให้หัวหน้าทีม เพื่อว่าหัวหน้าทีมจะได้นำรายงานต่าง ๆ เหล่านั้นไปรวบรวมจัดทำเป็นรายงานสถานะการทำงานของทีมงาน ซึ่งกิจกรรมนี้ควรใช้เวลาประมาณ 30-45 นาที (การรวบรวมจะทำได้สะดวกยิ่งขึ้นถ้ารายงานนั้นถูกจัดทำและส่งผ่านโดยทางจดหมายอิเลคทรอนิคส์) เพื่อหัวหน้าทีมแต่ละคนจะจัดส่งให้กับผู้จัดการโครงการต่อไป

From ; John Doe,Team leader

To : Frank Smith, Project Manager

Date : 15 June 1993

User interface team : weekly status report

for the period 5-12 June 1993

1. Activities during the report period:

1.1 The design of the user help screens (activity 3.12.6) was completed on schedule. The design specs were submitted to configuration control.

1.2 Coding of the command pass through modules (activity group 5.12) continues , and is currently behind schedule by about 1 week

: :

: :

2. Activities planned for next week :

2.1 Coding of the command pass through modules (activity group 5.12) will be completed, and unit tests will be started

2.2 Two members of the team (ED and Joan) will attend a two day course on the programmer's interface to the new user interface package. This is an unscheduled activity that was approved at the last project meeting. This will not delay the schedule , due to the early completion of the command pass through modules (see Section 1.2 above) .

: :

: :

3. Problems:

3.1 The user interface package we originally planned to use was found to be inadequate for the project. Two team members will study the new proposed package (see Section 2.2 above) . If the new package is also found to be unsuitable, then this will severely impact our development schedule.

3.2 One of our team members(Jack Brown) has been using an old VT100 terminal instead of a workstation for the past two weeks , due to the acute shortage of workstation . This is the reason why Jack's task 5.12 was not completed this week, as scheduled .

รูปภาพแสดงตัวอย่างรายงานสถานะการทำงานประจำสัปดาห์

2. <u>รายงานแบบปากเปล่า (Verbal reports)</u> แม้นว่ารายงานต่างๆ และการประชุมทีมงานเป็นสิ่งจำเป็นเพื่อจะได้มาซึ่งข้อมูลต่างๆ ก็ตาม แต่ไม่มีสิ่งใดจะดีเท่ากับการติดต่อสื่อสารโดยตรงระหว่างผู้จัดการโครงการ และผู้พัฒนาซอฟต์แวร์ แต่ละคนโดยตรง ซึ่งอาจกระทำโดยการสนทนาอย่างไม่เป็นทางการบ่อยๆ จัดในสถานที่ที่ไม่เป็นทางการ (ไม่ควรเป็นห้องประชุมหรือห้องทำงานของผู้จัดการโครงการ) เพื่อที่จะได้ข้อมูลที่ถูกต้องครบถ้วน

ผู้จัดการโครงการควรจะระมัดระวัง หรือ ป้องกันความผิดพลาดที่อาจเกิดขึ้นตามทฤษฎี " 90/50 Syndrome " ซึ่งเป็นลักษณะที่จะใช้เวลา 50 % ในการทำงานให้เสร็จ 90% ของงานทั้งหมด และใช้เวลา 50% เพื่อทำงาน 10% ที่เหลือ เพราะฉะนั้นผู้จัดการโครงการควรระมัดระวังคำพูดที่มีความแตกต่างกันอย่างมากระหว่างคำว่า " เกือบจะเสร็จ " และ " เสร็จสมบูรณ์ "

3. <u>การประชุมสถานะของโครงการ (Project status meetings)</u> การประชุมสถานะของโครงการควรจะจัดให้มีเป็นช่วงเวลาสม่ำเสมอ อย่างน้อยสัปดาห์ละครั้ง ช่วงเวลาที่เหมาะ

สมควรเป็นวันสุดท้ายของสัปดาห์ หรือ วันแรกของสัปดาห์ จัดขึ้นอย่างสม่ำเสมอ และประชุมภายในเวลาที่กำหนด ถ้าผู้ร่วมประชุมคนใดไม่สามารถเข้าประชุมได้ ควรได้รับการอนุมัติจากผู้จัดโครงการก่อน และควรส่งตัวแทนเป็นสมาชิกในทีมงานเข้าร่วมประชุมแทน

ผู้จัดการโครงการจะจัดเตรียมการประชุมโดยการตรวจสอบทบทวนรายงานสถานะการทำงาน ที่หัวหน้าทีมแต่ละทีมส่งมาให้ ซึ่งรายงานเหล่านี้ควรจัดส่งให้ก่อนการประชุมอย่างน้อย 2-3 ชั่วโมง

การประชุมจะมีผู้เข้าร่วมเป็นสมาชิกโครงการที่มีหน้าที่หลักในงานต่าง ๆ หัวหน้าทีมงานแต่ละทีม การประชุมเริ่มต้นด้วยการรายงานกิจกรรมต่าง ๆ ของโครงการ และเรื่องทั่ว ๆ ไป โดยผู้จัดการโครงการ แล้วตามด้วยการให้ผู้เข้าร่วมประชุมแต่ละคนรายงานผลของกิจกรรมต่าง ๆ ที่ตนเองรับผิดชอบ แต่ละคนควรใช้เวลาประมาณ 5-10 นาที ปัญหาต่าง ๆ ที่เกิดขึ้นควรให้ผู้เข้าร่วมประชุมทุกคนได้แสดงความคิดเห็น และร่วมกันแก้ปัญหา

ปัญหาต่าง ๆ ที่เกิดขึ้นควรจะมีการสรุป แนวทางแก้ไขในระหว่างการประชุม แต่ถ้าปัญหาใดใช้เวลามากกว่า 5 นาที ควรจะเลื่อนไปเป็นการประชุมเพื่อปัญหานั้น ๆ โดยเฉพาะภายหลังการประชุมสถานะของโครงการ

การประชุมสถานะของโครงการทุกครั้งควรมีการจดบันทึกรายงานการประชุมด้วย ซึ่งมีรายละเอียด ดังนี้

1. วันที่ประชุม
2. ชื่อของการประชุม
3. รายชื่อผู้เข้าร่วมการประชุม
4. รายชื่อผู้ไม่เข้าร่วมประชุม
5. รายละเอียดกิจกรรมที่กระทำ (ชื่อผู้รับผิดชอบ , กิจกรรม , วันที่ควรเสร็จ)
6. รายละเอียดการตัดสินใจ แก้ปัญหาต่าง ๆ

รายงานประชุมควรจัดพิมพ์ และแจกจ่ายให้กับผู้ที่เกี่ยวข้องในทันที ซึ่งไม่ควรเกินสิ้นวันนั้น เพราะบางกิจกรรมมีรายละเอียด จะต้องกระทำให้เสร็จในทันทีภายในวันที่ประชุมนั้น ถ้าโครงการขนาดใหญ่อาจมีเลขา เพื่อทำหน้าที่จดรายงานการประชุม แต่โครงการขนาดเล็ก ผู้จัดการโครงการอาจมอบหมายงานนี้ให้ผู้เข้าร่วมประชุม แต่ละคนผลัดเปลี่ยนกันไปทุกสัปดาห์ก็ได้

4. <u>การสาธิตผลิตภัณฑ์ (Product demonstration - demos)</u> การสาธิตผลิตภัณฑ์สามารถช่วยให้ผู้จัดการโครงการติดตามสถานะการทำงานของโครงการได้ว่าดำเนินการไปถึงไหน มีปัญหาอย่างไร ต้องปรับปรุงเปลี่ยนแปลงอะไรบ้าง แต่จะได้ข้อมูลเพียงคร่าว ๆ เนื่องจาผู้พัฒนาจะสาธิต

ผลิตภัณฑ์ให้ดูเฉพาะสิ่งที่เขาต้องการให้ผู้จัดการโครงการเห็นเท่านั้น อาจทำให้ไม่ทราบถึงสถานะการทำงาน และปัญหาที่แท้จริง เหมือนกับการรายงานผลการดำเนินงานโครงการแบบอื่นๆ

2.2.2 การปรับปรุงตารางเวลาการทำงานให้ทันสมัยอยู่เสมอ (Updating the schedule)

แผนงานการพัฒนาซอฟต์แวร์ของโครงการควรจะต้องมีการตรวจสอบ ทบทวนเป็นประจำตามช่วงระยะเวลาที่กำหนด และตารางการทำงานควรปรับปรุงให้ทันสมัยอยู่เสมอเมื่อมีการตรวจพบเหตุการณ์ต่างๆ ที่สำคัญเกิดขึ้น ตัวอย่างเช่น ถ้าการตรวจสอบพบว่ามีกิจกรรมหลายอย่างที่ช้ากว่าแผนงานตารางเวลาที่กำหนดไว้ หรือ พบว่ามีกิจกรรมใหม่ๆ ที่ต้องเพิ่มเติมเข้าไปในรายละเอียดกิจกรรม ก็ควรจะต้องจัดทำแผนงานตารางเวลาการทำงานชิ้นใหม่

ตารางด้านล่างนี้จะแสดงรายละเอียดหัวข้อต่างๆ ที่สำคัญของแผนงานตารางเวลาที่ควรจะถูกตรวจสอบทบทวนอยู่เสมอ หรือควรตรวจสอบเมื่อพบเหตุการณ์ผิดปกติเกิดขึ้น

1. Activity list
2. Personnel assignments
3. Risk list and risk analysis
4. Resource allocation
5. Third party status (subcontractors , vendors , suppliers)
6. Schedule chart (Gantt)
7. Precedence network (PERT)
8. Approved requirements and design changes

ตารางแสดงรายละเอียดหัวข้อต่างๆ ที่สำคัญของแผนงานตารางเวลา (Schedule Update checklist)

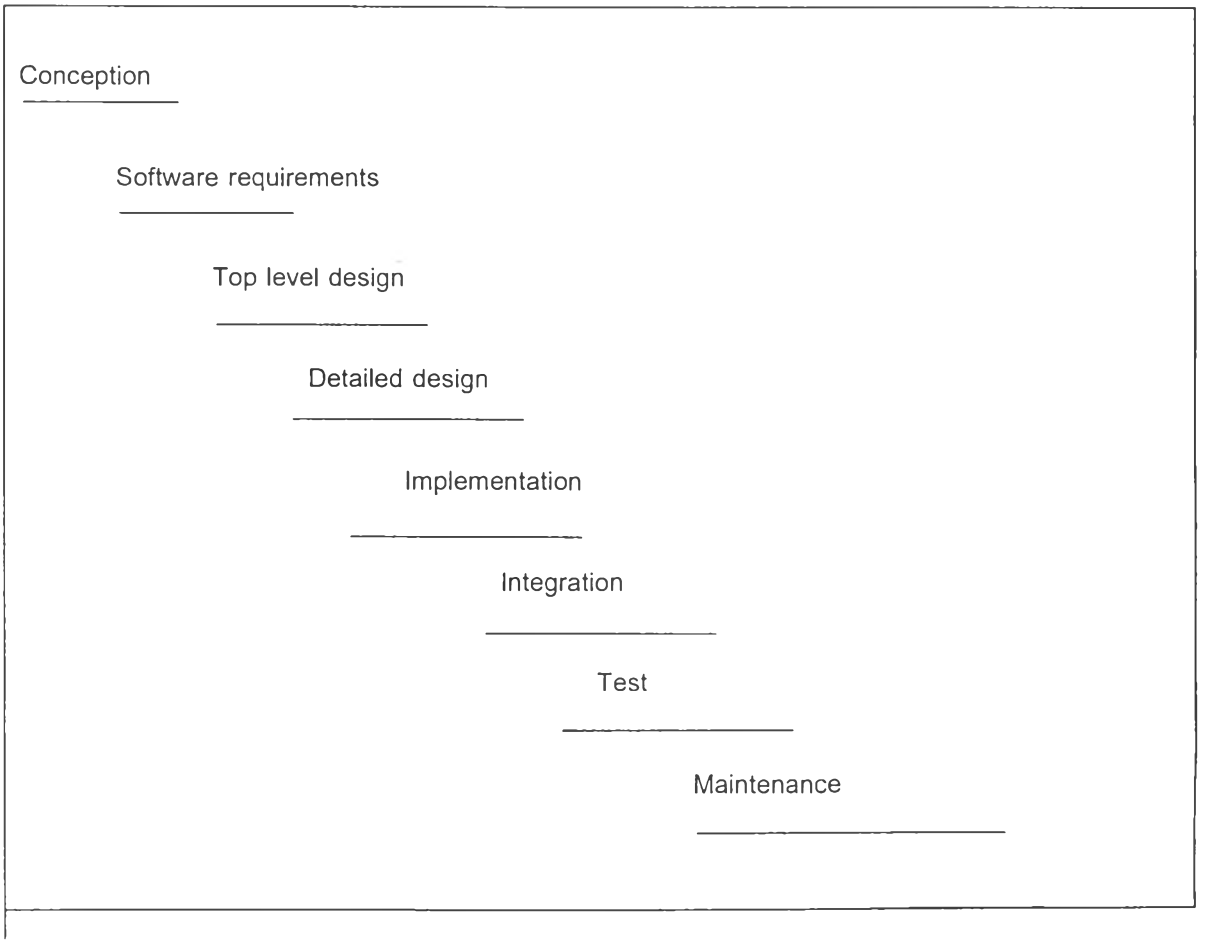
ภาคผนวก ค.

3. **ขั้นตอนการทำงานในการพัฒนาซอฟต์แวร์ (Software development life Cycle)** มีหลายรูปแบบ แต่ในที่นี้จะขอยกตัวอย่างวงจรชีวิตของการพัฒนาซอฟต์แวร์ 2 แบบคือ

   3.1 วงจรชีวิตของการพัฒนาซอฟต์แวร์ แบบ Waterfall

   3.2 วงจรชีวิตของการพัฒนาซอฟต์แวร์ ตาม IEEE Standard 610.12

   3.1 **ขั้นตอนการพัฒนาซอฟต์แวร์ตามวงจรชีวิตในการพัฒนาซอฟต์แวร์รูปแบบ Waterfall (The Waterfall model of the Software development life cycle)** แบ่งออกเป็น



รูปภาพแสดงวงจรชีวิตการพัฒนาซอฟต์แวร์รูปแบบ Waterfall

1. **ขั้นตอนแนวคิดเบื้องต้น (The Concept Phase)** หรือถูกเรียกโดย IEEE ว่าขั้นตอน Concept Exploration Phase จะเป็นการจัดเตรียมข้อมูลพื้นฐานเกี่ยวกับ

    1. การจัดเตรียมข้อเสนอสิ่งที่ต้องการ (Request for proposal – RFP) ซึ่งจะใช้ประโยชน์เมื่อโครงการ ติดต่อกับผู้พัฒนาภายนอก ซึ่งประกอบด้วยรายละเอียดปัญหา และวัตถุประสงค์ของโครงการ ความต้องการทางด้านเทคนิค ต้นทุน รายละเอียดสิ่งที่ต้องการให้จัดส่ง (เอกสาร ซอฟต์แวร์ การฝึกอบรม ฮาร์ดแวร์ และอุปกรณ์ที่เกี่ยวข้อง เครื่องมือในการพัฒนา และทดสอบ) ขอบเขตของงานที่ต้องดำเนินการและตารางการยื่นข้อเสนอ และตารางการตัดสินใจเลือกผู้พัฒนา รวมทั้งตารางเวลาในการพัฒนางานสำเร็จ
    2. การกำหนดขอบเขตความต้องการซอฟต์แวร์ (ขั้นตอนต่อไป)
    3. แผนงานเริ่มต้น และการจัดเตรียมการประเมินงาน

    ในขั้นตอนนี้จะเกิดเอกสารสำหรับโครงการขึ้น 2 ชนิดคือ

    1. คำอธิบายรายละเอียดผลิตภัณฑ์ (The Product Description) เป็นเอกสารเกี่ยวกับทางด้านการตลาดใช้ในการแสดงรายละเอียดคุณสมบัติโดยทั่วไปของผลิตภัณฑ์
    2. เอกสารรายละเอียดแนวคิดเบื้องต้น (A Concept Document) เป็นเอกสารทางด้านเทคนิคและแบบฟอร์มที่เป็นกิจกรรมหลักของขั้นตอนนี้ ได้แก่ Request for proposal – RFP และแผนงานเบื้องต้น

    ปัญหาต่าง ๆ ที่อาจเกิดขึ้นในขั้นตอนแนวคิดเบื้องต้นนี้ได้แก่

    1. การขาดแคลนการสนับสนุนอย่างเต็มที่จากฝ่ายจัดการมีเพียงงบประมาณขั้นต้นเพียงเล็กน้อยที่ถูกแบ่งมาให้ในขั้นตอนนี้
    2. ปัญหาของทีมงานพัฒนาซอฟต์แวร์ที่เกี่ยวเนื่องมาจากผู้ใช้ไม่สามารถกำหนดขอบเขตสิ่งที่ต้องการได้แน่ชัด
    3. การไม่มีความสามารถของฝ่ายเทคนิคที่ไม่สามารถจัดการสิ่งใด ๆ ได้มากกว่าการประเมินงานอย่างคร่าว ๆ เท่านั้น
    4. ปัญหาต่าง ๆ ส่วนมากเกี่ยวข้องกับการจัดตั้งทีมงานการพัฒนาโครงการในขั้นเริ่มต้น การจัดคนที่เหมาะสมสำหรับทีมงานของโครงการรวมถึงผู้จัดการโครงการเป็นสิ่งที่ยากลำบาก
    5. ถ้ามีผู้รับผิดชอบในกิจกรรมต่าง ๆ ในขั้นตอนแนวคิดเบื้องต้นมากเกินไป (เช่น รับผิดชอบวิเคราะห์ด้านการตลาด รับผิดชอบด้านการประเมินงาน และรับผิดชอบด้านการจัดทำเอกสารแนวคิดเบื้องต้น) จะทำให้ขาดการประสานงานที่ดีระหว่างกัน
    6. เมื่อมีการจัดตั้งทีมงานเริ่มต้น และจัดทำแผนงานการพัฒนาซอฟต์แวร์อย่างคร่าว ๆ แล้วนั้น จะพบว่าแต่ละคนมีความคิดที่ต้องการจะให้ผลิตภัณฑ์ออกมาในรูปแบบแตกต่างกันไป ทำ

ให้กลายเป็นปัญหาได้ซึ่งควรมีการสาธิตระบบงาน (Demo Systems) หรือการทำแบบจำลอง (Prototypes)

2. <u>ขั้นตอนการกำหนดรายละเอียดความต้องการซอฟต์แวร์ (The Software Requirements Phase)</u> หรือเรียกว่าขั้นตอนการกำหนดขอบเขต (The Definition Phase) เป็นขั้นตอนในการพัฒนาซอฟต์แวร์อย่างเป็นทางการขั้นแรก ซึ่งจะประกอบด้วยการกำหนดรายละเอียดของระบบซอฟต์แวร์ที่จะพัฒนา ซึ่งจะนำไปสู่การกำหนดขอบเขตความต้องการระบบงานซอฟต์แวร์ เพื่อใช้ในการออกแบบระบบต่อไป

<u>ในขั้นตอนนี้จะเกิดเอกสารขึ้น 3 ชนิดคือ</u>

1. เอกสารรายละเอียดความต้องการระบบงานซอฟต์แวร์ (Software requirements specification document)
2. แผนงานการพัฒนาโครงการ (Project development plan)
3. แผนงานการทดสอบซอฟต์แวร์ (Software test plan)

<u>ปัญหาต่าง ๆ ที่อาจเกิดขึ้นในขั้นตอนการกำหนดรายละเอียดความต้องการซอฟต์แวร์</u> ได้แก่

1. รายละเอียดความต้องการที่ผิดพลาดซึ่งเป็นผลเนื่องมาจาก แบบจำลองที่นำมาใช้ไม่ตรงกับระบบงานจริง เอกสารที่ใช้อ้างอิงผิดพลาดผิดขั้นตอน
2. ความต้องการที่ได้ไม่ชัดเจน
3. ความต้องการไม่ครบถ้วน ถูกละทิ้งบางส่วน
4. ความต้องการถูกเปลี่ยนแปลงอยู่เสมอ
5. ปัญหาที่เกิดขึ้นจากการอนุมัติความต้องการอย่างเป็นทางการ ขั้นตอนนี้จะสำเร็จได้ต่อเมื่อมีการอนุมัติอย่างเป็นทางการ
6. การวิเคราะห์ความเป็นได้ของความต้องการ บางครั้งเป็นการยากที่จะวิเคราะห์ความเป็นไปได้ของโครงการก่อนจะถึงขั้นตอนการนำระบบงานคอมพิวเตอร์ไปใช้ ซึ่งอาจจะใช้แบบจำลองเข้าไปช่วยในการทดสอบความต้องการ การให้ผู้เชี่ยวชาญด้านการวิเคราะห์ความเป็นไปได้เข้ามามีส่วนร่วมเป็นการชั่วคราว ถ้าความต้องการเดิมไม่สามารถเป็นไปได้ต้องขออนุมัติความต้องการที่เป็นทางเลือกอื่นเป็นต้น
7. ทีมงาน การจัดบุคคลากรให้เหมาะสมกับทีมงานพัฒนากระทำได้ยากแต่ควรจะทำให้เรียบร้อยในขั้นตอนนี้ ซึ่งอาจแก้ไขปัญหา โดยการใช้พนักงานชั่วคราว เพิ่มอัตรากำลัง จัดฝึกอบรมบุคลากรที่มีอยู่
8. การขาดแคลนงบประมาณ และอุปกรณ์ที่ใช้สนับสนุนการกำหนดรายละเอียดความต้องการซอฟต์แวร์ ซึ่งอาจแก้ไขปัญหาโดยการใช้เครื่องมือ อุปกรณ์ที่เช่ายืมมาชั่วคราว ใช้แบบ

จำลองซอฟต์แวร์แทนการใช้อุปกรณ์ที่มีราคาแพง ของบประมาณเพิ่มเติมเพื่อจัดหาอุปกรณ์พื้นฐานที่มาช่วยในการทำงาน

3. <u>ขั้นตอนการออกแบบ (The Design phase)</u> มักจะถูกแบ่งออกเป็น 2 ขั้นคือ

ขั้นการออกแบบขั้นต้น (Top level design) และขั้นการออกแบบรายละเอียด (Detailed design) ซึ่งการแบ่งขั้นตอนการออกแบบเป็น 2 ระดับนี้มีประโยชน์กว่าการใช้ระดับเดียว ซึ่งจะมีความสำคัญมากในโครงการขนาดกลาง และขนาดใหญ่ เพราะข้อผิดพลาดในการออกแบบหลักจะถูกพบในช่วงต้นเร็วที่สุดเท่าที่จะเป็นได้ เมื่อข้อผิดพลาดหลักๆ ในการออกแบบถูกพบในช่วงสุดท้ายของการออกแบบขั้นต้นจะง่ายในการแก้ไขมากกว่า ถ้าพบหลังจากการออกแบบรายละเอียดทั้งหมดเสร็จแล้ว

<u>ในขั้นตอนการออกแบบซอฟต์แวร์นี้จะเกิดเอกสารขึ้นดังนี้คือ</u>

1. เอกสารรายละเอียดความต้องการ (Design specification) สำหรับโครงการใหญ่ จะมีทั้งราย ละเอียดการออกแบบขั้นต้น (Top level design specification) และรายละเอียดการออกแบบขั้นละเอียด (detailed design specification)
2. แผนงานการรวมกันของซอฟต์แวร์ (Integration plan)
3. รายละเอียดการทดสอบในกรณีต่างๆ ซึ่งจะอธิบายในรายละเอียดแต่ละจุดที่จะใช้การทดสอบระดับต่ำ (low level test)

<u>ปัญหาต่าง ๆ ที่อาจเกิดขึ้นในขั้นตอนการออกแบบซอฟต์แวร์ได้แก่</u>

1. ปัจจัยที่เกี่ยวข้องกับเทคนิคการออกแบบ ซึ่งเป็นผลเนื่องมาจากความต้องการซอฟต์แวร์ที่ไม่อาจ เป็นไปได้ การออกแบบที่ซับซ้อน และการนำคอมพิวเตอร์ไปประยุกต์ใช้ที่ยุ่งยาก แต่ปัญหาทุกอย่างควรแก้ไขให้เสร็จเรียบร้อยภายในขั้นตอนการออกแบบนี้ เพราะว่าถ้าปล่อยให้กระบวนการ
2. ซอฟต์แวร์ของโครงการดำเนินไปเรื่อยๆ จะทำให้ต้องเสียค่าใช้จ่ายเพิ่มขึ้น การหาบุคคลากรได้ยาก ซึ่งเป็นผลเนื่องมาจากในขั้นดอนการกำหนดรายละเอียดความต้องการซอฟต์แวร์แล้ว ผู้จัดการโครงการจะต้องมีหน้าที่กำหนดคนในการรับผิดชอบตำแหน่งทุกตำแหน่งของโครงการ โดยเร็วที่สุดล่วงหน้าก่อนที่จะเกิดความต้องการบุคคลากรจริงๆ
3. การจัดหาทรัพยากรที่ใช้ในการพัฒนาซอฟต์แวร์ ซึ่งเป็นผลเนื่องมาจากในขั้นตอนการกำหนดรายละเอียดความต้องการซอฟต์แวร์ ซึ่งอาจจะใช้วิธีเช่า หรือกู้เงินมาเพื่อซื้ออุปกรณ์ต่างๆ ซึ่งต้องควบคุมเพื่อไม่ให้เกิดปัญหาตามมาภายหลัง

4. ความสัมพันธ์ระหว่างลูกค้ากับทีมงานพัฒนาซอฟต์แวร์ อาจเกิดปัญหาขึ้นในระหว่างขั้นตอนการออกแบบได้ ซึ่งหลังจากมีการอนุมัติความต้องการซอฟต์แวร์แล้ว ก็ควรมีการตรวจสอบทบทวนอย่างเป็นทางการในขั้นตอนการออกแบบ และให้ลูกค้ามีส่วนร่วมรับผิดชอบในการออกแบบ
5. การใช้เทคนิคการออกแบบที่ไม่มีโครงสร้างถูกต้อง (Unstructured approach)
6. การขาดแคลนการควบคุมการเปลี่ยนแปลง และขอบเขตโครงร่างการออกแบบ
7. การขาดแคลนการตรวจสอบทบทวนที่เหมาะสม
8. มีการเข้าใจผิดเกี่ยวกับรายละเอียดของซอฟต์แวร์ที่ต้องการ ทำให้การออกแบบเกิดความผิดพลาด

4. <u>ขั้นตอนการนำซอฟต์แวร์ไปใช้งาน (The Implementation Phase)</u> ในขั้นตอนนี้ระบบย่อยต่างๆ ของซอฟต์แวร์จะถูกเขียนโปรแกรม (Coding) และเริ่มต้นทดสอบในแต่ละส่วนย่อยๆ (Unit Test) ซึ่งการทดสอบในแต่ละส่วนย่อยๆ จะรับผิดชอบโดยโปรแกรมเมอร์ในแต่ละระบบย่อย ซึ่งจะทำการทดสอบทันทีหลังจากที่เขียนโปรแกรมเสร็จ

วงจรชีวิตในการพัฒนาซอฟต์แวร์ใช้อัตราส่วนของความพยายาม และเวลาในการพัฒนาเป็น 40-20-40 ซึ่งหมายความว่า จะใช้เวลา 40% ในการกำหนดรายละเอียดความต้องการ และการออกแบบ (requirements and design specification) , เวลา 20% ใช้ในขั้นตอนการนำซอฟต์แวร์ไปใช้งาน (Implementation) นั่นคือ การเขียนโปรแกรม และการทดสอบโปรแกรมส่วนย่อย (Coding and unit testing) และเวลา 40% ใช้ในการรวบรวมซอฟต์แวร์ และการทดสอบระบบ (integration and testing) แต่แนวโน้มในปัจจุบันมีการลดเวลาในขั้นตอนการนำซอฟต์แวร์ไปใช้ และเพิ่มเวลาเข้าไปในขั้นตอนการกำหนดรายละเอียดความต้องการแทน เพราะว่าขั้นตอนในการพัฒนาซอฟต์แวร์ขั้นตอนแรกๆ จะใช้ต้นทุนน้อยกว่าขั้นต่อๆ มา เพราะฉะนั้นถ้าความพยายามถูกทุ่มเทไปในขั้นตอนการกำหนดรายละเอียดความต้องการ และการออกแบบเมื่อถึงขั้นตอนการรวบรวมซอฟต์แวร์ และการทดสอบระบบจะดำเนินการได้ง่าย และมีประสิทธิภาพมากขึ้น

ขั้นตอนการนำซอฟต์แวร์ไปใช้งานจะรวมกิจกรรมหลักต่าง ๆ ดังนี้คือ

1. การพัฒนาโปรแกรมซอฟต์แวร์
2. การจัดเตรียมการรวบรวมซอฟต์แวร์ และการทดสอบระบบ (ขั้นตอนต่อไป)
3. การพัฒนาแผนงานการบำรุงรักษาระบบงานซอฟต์แวร์ (Maintenance Plan)

<u>**ในขั้นตอนนี้จะเกิดเอกสารสำหรับโครงการขึ้น 3 ชนิดคือ**</u>

1. เอกสารบันทึกรายละเอียดต่างๆ ของโปรแกรมเมอร์ , เอกสารที่เป็นเงื่อนไขในการเขียนโปรแกรม , เอกสารการทดสอบระบบย่อย และเอกสารการแก้ปัญหาที่เกิดขึ้นในขั้นตอนการนำซอฟต์แวร์ไปใช้งาน

2. แผนงานการบำรุงรักษาระบบงานซอฟต์แวร์ (Maintenance plan) และเอกสารรายละเอียดต่าง ๆที่จำเป็นในการบำรุงรักษาระบบงานซอฟต์แวร์
3. เอกสารสำหรับผู้ใช้งานชุดแรก ได้แก่ คู่มือที่เกี่ยวข้องกับการใช้งาน (reference manuals) และคู่มือแนะนำการปฏิบัติงาน (operator guides)

ปัญหาที่อาจเกิดขึ้นในขั้นตอนการนำซอฟต์แวร์ไปใช้งาน ได้แก่

1. ปัญหาเรื่องการเปลี่ยนแปลงที่ไม่สิ้นสุด เป็นปัญหาที่สามารถแก้ไขโดยการเคร่งครัดเรื่องการเปลี่ยนแปลง และการควบคุมการเปลี่ยนแปลงอย่างเป็นลำดับ การเปลี่ยนแปลงที่เกิดขึ้นอย่างต่อเนื่อง มักจะนำความเสียหายให้เกิดขึ้นกับการพัฒนาซอฟต์แวร์ การเปลี่ยนแปลงต่าง ๆ ต้องถูกจัดทำอย่างพอควรเหมาะสม โดยเฉพาะในทีมงานที่มีขนาดใหญ่อาจไม่ค่อยระมัดระวังเรื่องการเปลี่ยนแปลงที่มีความสำคัญ ทำให้เกิดความเสียหายต่อโครงการพัฒนาซอฟต์แวร์ได้
2. ปัญหาในการติดต่อประสานงานระหว่างสมาชิกของทีมงาน ปัญหานี้จะมีความรุนแรงมากในโครงการขนาดใหญ่ การติดต่อสื่อสารระหว่างสมาชิกทีมงานที่มีขนาดใหญ่มักจะใช้เวลามาก ซึ่งสามารถลดเวลาได้โดยการแบ่งโครงการใหญ่ออกเป็นระบบย่อย ๆ และมอบหมายให้ทีมงานอิสระรับผิดชอบในแต่ละระบบย่อย ๆซึ่งจะทำให้ช่องทางการติดต่อสื่อสารที่เป็นทางการถูกกำหนดขึ้นระหว่างทีมงานที่มีขนาดเล็ก เพื่อไม่สูญเสียเวลาในการติดต่อสื่อสารระหว่างกันและลดปัญหาความผิดพลาดที่อาจเกิดขึ้นได้
3. ปัญหาในการควบคุม และการตรวจสอบบริษัทผู้ผลิตซอฟต์แวร์ภายนอกองค์กรและผู้ขายซอฟต์แวร์ ในบางกรณี ระบบงานย่อย ๆ จะถูกมอบหมายให้กับบริษัทผู้ผลิตซอฟต์แวร์ภายนอกองค์กร ซึ่งควรจะมีการควบคุมอย่างเต็มเวลา เนื่องจากมักไม่มีการรายงานด้วยวาจาหรือเขียนเป็นลายลักษณ์อักษรที่พอเพียง ฉะนั้นการไปเยี่ยมยังสถานที่ปฏิบัติงานจริงเป็นสิ่งจำเป็นที่ต้องกระทำสม่ำเสมอ เพื่อให้แน่ใจว่าข้อมูลที่ได้รับมาถูกต้องเชื่อถือได้

5. ขั้นตอนการรวบรวมระบบงานซอฟต์แวร์ และการทดสอบ (The Integration and Test Phase) ในระหว่างขั้นตอนนี้จะมีการรวมระบบย่อย ๆ ของซอฟต์แวร์เข้าเป็นระบบเดี่ยว และทำการทดสอบคุณสมบัติของระบบว่าตรงตามที่ต้องการหรือไม่ และจะมีการประเมินเพื่อแจกแจงปัญหาที่ต้องแก้ไขก่อนที่ระบบจะเสร็จสมบูรณ์

ในขั้นตอนนี้จะจัดเตรียมพื้นฐานสำหรับสิ่งต่าง ๆ ได้แก่

1. การสร้างระบบงานซอฟต์แวร์จากองค์ประกอบของซอฟต์แวร์ต่าง ๆ
2. การรวบรวมระบบงานซอฟต์แวร์เข้ากับอุปกรณ์ฮาร์ดแวร์
3. การตัดสินว่าระบบงานที่พัฒนาขึ้นตรงตามขอบเขตรายละเอียดความต้องการที่ ได้กำหนดไว้หรือไม่
4. การสร้างคุณภาพให้กับระบบงาน

ในขั้นตอนนี้มักจะแบ่งออกเป็น 2 ส่วนใหญ่ๆ คือ ขั้นตอนการรวบรวมระบบงาน (Integration phase) และขั้นตอนการทดสอบระบบงาน (Testing phase) ในโครงการใหญ่ๆ การแบ่งส่วนเช่นนี้เป็นสิ่งที่จำเป็น โดยเฉพาะเมื่อการทดสอบถูกจัดทำโดยกลุ่มอิสระที่แยกออกต่างหาก (The independent test team) ซึ่งจะทำงานควบคู่ไปกับผู้พัฒนาซอฟต์แวร์ แต่ในโครงการขนาดเล็กๆ เหตุผลในการรวมกิจกรรม 2 อย่างนี้เป็นขั้นตอนเดียวก็เนื่องจากว่า การรวบรวมระบบงานซอฟต์แวร์จะดำเนินการให้สำเร็จโดยปราศจากการทดสอบที่ควรจะกระทำควบคู่ไปด้วยกันไม่ได้ เพราะฉะนั้นถ้ากิจกรรม 2 อย่างนี้ถูกจัดทำโดยทีมงานทีมเดียวกันก็ควรจะรวมขั้นตอนให้อยู่ในขั้นตอนเดียวกัน

<u>เทคนิค และวิธีการในการรวบรวมระบบงาน</u> (Techniques and method of intergration) มีดังนี้คือ

1. **<u>เทคนิคจากบนลงล่าง (top down approach)</u>** จะเริ่มต้นจากการเตรียมนำระบบงานหลักที่สำคัญๆ ไปใช้งานก่อน แล้วจึงค่อยๆรวบรวมระบบย่อยเล็กๆ เข้าไป
2. **<u>เทคนิคจากล่างขึ้นบน (bottom up)</u>** จะเริ่มต้นจากระบบย่อยเล็กๆ ในระบบล่างสุดแล้วค่อยๆรวมกันเพิ่มขึ้นเรื่อยๆ เป็นกลุ่มขนาดใหญ่ขึ้นจนประกอบเป็นระบบทั้งหมดที่สมบูรณ์ เทคนิคจากล่างขึ้นบนมักจะไม่ค่อยแนะนำให้ใช้เป็นกลยุทธ์ในการรวบรวมระบบงานซอฟต์แวร์ ส่วนมากจะแนะนำให้ใช้เทคนิคจากบนลงล่าง เพราะง่ายกว่า และใกล้เคียงกับสภาพความเป็นจริงมากกว่า แต่ในความเป็นจริงการรวบรวมระบบงานซอฟต์แวร์ที่ประสบความสำเร็จ มักจะใช้ 2 เทคนิคนี้ร่วมกัน
3. **<u>เทคนิคจากภายในสู่ภายนอก (Inside out)</u>** เป็นเทคนิคที่ถูกนำมาใช้ในการพัฒนาระบบฐานข้อมูลขนาดใหญ่ เริ่มต้นด้วยการสร้างโครงสร้างแฟ้มข้อมูลภายใน(Internal File Structure)ก่อน แล้วตามด้วยการเพิ่มเติมส่วนตรรกศาสตร์ของการประมวลผลข้อมูล (data processing logic) เข้าไปแล้วสุดท้ายก็เพิ่มเติมส่วนการติดต่อใช้งาน (Human interface) เข้าไป เทคนิคนี้จะใช้ได้ดีกับระบบที่ประกอบด้วยส่วนประกอบแยกตามหน้าที่เป็นขั้นๆ ตามลำดับ แต่ก็มีข้อเสียคือ การนำส่วนการติดต่อใช้งานเข้าไปรวมเป็นลำดับสุดท้าย เลยอาจทำให้ต้องเขียนโปรแกรมชั่วคราวขึ้นมาใช้ทำการทดสอบ เพื่อให้สามารถแสดงผลลัพธ์ได้ ทำให้การทดสอบช้า และยากขั้นตอนการทดสอบจะเริ่มต้นพร้อมกับการรวบรวมระบบงานซอฟต์แวร์ และจะดำเนินการไปเรื่อยๆ จนกระทั่งขั้นตอนสุดท้าย คือจัดส่งระบบงานให้กับผู้ใช้งาน หรือลูกค้า

<u>ประเภทของการทดสอบมีแบบต่าง ๆ ดังนี้คือ</u>

1. การทดสอบการรวมกันของระบบ (Intergration testing) ซึ่งดำเนินการโดยผู้รวบรวมระบบ (System integrators)

2. การทดสอบแบบอิสระ (Independent testing) ซึ่งดำเนินการโดยกลุ่มผู้ทดสอบภายนอก เพื่อให้มั่นใจว่าการทดสอบระบบนั้นไม่มีอคติ (Unbiased testing)
3. การทดสอบการติดตั้งระบบ (Installation testing) รวมถึงการทดสอบผลการปฏิบัติงานโดยทั่วไป เมื่อระบบงานชุดแรกถูกติดตั้งในสภาพแวดล้อมที่ปฏิบัติงานจริง เพื่อให้มั่นใจว่าระบบถูกติดตั้งอย่างสมบูรณ์
4. การทดสอบแบบอัลฟ่า และเบต้า (Alpha and beta testing) การทดสอบนี้จะกระทำภายใต้สภาพแวดล้อมระบบงานจริง ซึ่งการทดสอบแบบอัลฟ่าจะทดสอบระบบโดยไม่มีข้อมูลจริง แต่การทดสอบแบบเบต้าจะทดสอบระบบโดยใช้ข้อมูลจริงตามข้อจำกัดต่างๆ เพื่อที่จะได้แก้ไขปัญหาต่างๆ ที่อาจเกิดขึ้น
5. การทดสอบการยอมรับระบบงาน (Acceptance Testing) ซึ่งเป็นขั้นตอนขั้นสุดท้ายของโครงการ ซึ่งโครงการจะสำเร็จสมบูรณ์จะสังเกตได้จาก การที่ลูกค้า หรือผู้ใช้ยอมรับผลิตภัณฑ์งานที่พัฒนาขึ้นมา

ในขั้นตอนนี้เอกสารต่างๆ ทั้งหมดจะต้องเสร็จสมบูรณ์ และพร้อมที่จะจัดส่ง ซึ่งได้แก่

1. เอกสารการบำรุงรักษาระบบงาน (Maintenance Documentation)
2. เอกสารสำหรับผู้ใช้ที่เสร็จสมบูรณ์ (Final user Documentation)
3. เอกสารการพัฒนาระบบงานทั้งหมดที่ปรับปรุงล่าสุด (All updated development documentation)
4. เอกสารการทดสอบระบบงาน และรายงานการทดสอบระบบ(Test documentation and test reports)

ปัญหาต่าง ๆ ที่อาจเกิดขึ้นในขั้นตอนการรวบรวมซอฟต์แวร์ และการทดสอบระบบ ได้แก่

1. ความผิดพลาดที่อาจเกิดขึ้นนาทีสุดท้าย (Last minute failures) ซึ่งได้แก่ข้อผิดพลาดที่เกิดขึ้นตอนออกแบบ (design errors) และปัญหาที่เกิดขึ้นตอนเตรียมความพร้อมการนำไปใช้งาน (implementation problems) ซึ่งเป็นปัญหาที่ยากที่จะพบถ้าไม่มีการใช้งานระบบจริง และมักจะไม่พบตอนต้นๆของโครงการ
2. ปัญหาที่เกิดขึ้นจากบุคคลที่สาม (Third party problems) รวมทั้งการจัดส่งล่าช้าของผู้ขายและบุคคลที่ติดต่อภายนอก และความเสียหายที่อาจเกิดขึ้นในระบบ และองค์ประกอบย่อยๆ
3. การเปลี่ยนแปลงในนาทีสุดท้าย (Last Minute changes) ปัญหานี้เกิดขึ้นในทุกๆ ขั้นตอน แต่จะกลายเป็นปัญหาที่รุนแรงขณะที่โครงการกำลังดำเนินไปอย่างต่อเนื่อง การเปลี่ยนแปลงในภายหลังจะทำให้เกิดต้นทุนมาก

4. งบประมาณบานปลาย (Budget overruns) ปัญหานี้เกิดขึ้นจากข้อผิดพลาดจากการเปลี่ยนแปลง และข้อผิดพลาดในการออกแบบ หรืออาจเกิดจากข้อผิดพลาดในการวางแผนโครงการ
5. ปัญหาจากแรงจูงใจของพนักงาน (Staff motivation problems) ปัญหานี้มักเกิดขึ้นในตอนท้ายของโครงการระหว่างขั้นตอนการทดสอบขั้นสุดท้าย
6. ปัญหาจากการยอมรับระบบงานของโครงการ ปัญหานี้มักเกิดขึ้นในโครงการที่ใช้ราคาคงที่ (fixed price projects)

6. ขั้นตอนการบำรุงรักษาระบบงาน (The Maintenance Phase) ขั้นตอนการบำรุงรักษาระบบงานถือเป็นขั้นตอนสุดท้ายของวงจรชีวิตการพัฒนาซอฟต์แวร์ และเป็นจุดเชื่อมต่อระหว่างผลิตภัณฑ์ซอฟต์แวร์ที่เสร็จสมบูรณ์กับการพัฒนาผลิตภัณฑ์ใหม่ คำว่า " การบำรุงรักษาซอฟต์แวร์ (Software maintenance) " อาจทำให้เกิดความสับสนเพราะทำให้เข้าใจว่าหมายถึงความต้องการที่จะซ่อมบำรุงผลิตภัณฑ์ที่เสื่อมคุณภาพ หรือต้องการจะเปลี่ยนชิ้นส่วนที่เสีย เช่นระบบทางด้านไฟฟ้า หรือเครื่องยนต์ แต่ซอฟต์แวร์ไม่เป็นเช่นนั้นเพราะตัวซอฟต์แวร์เอง จะไม่เปลี่ยนแปลง หรือเสื่อมสภาพถ้าไม่มีคนเข้าไปเกี่ยวข้อง

IEEE ได้ให้คำจำกัดความของคำว่า " การบำรุงรักษาซอฟต์แวร์ " ว่าหมายถึงการแก้ไขข้อผิดพลาดที่อาจเกิดขึ้นในซอฟต์แวร์ก่อนการจัดส่งซึ่งรวมถึงการเปลี่ยนแปลงเพื่อปรับปรุงคุณภาพ หรือการปรับสภาพผลิตภัณฑ์ให้เข้ากับสภาพแวดล้อมที่เปลี่ยนแปลงไป

กิจกรรมที่เป็นส่วนหนึ่งของการบำรุงรักษา ได้แก่

1. การปรับปรุงเอกสารต่างๆ ให้ทันสมัยอยู่เสมอ
2. การปรับปรุงหลักสูตรการฝึกอบรมให้แก่ผู้ใช้
3. การปรับปรุงแก้ไขซอฟต์แวร์ (Modifying Software)
4. การควบคุมขอบเขตโครงร่างของซอฟต์แวร์ (Configuration Control)

การปรับปรุงแก้ไขซอฟต์แวร์จะรวมกิจกรรมต่างๆ เช่นเดียวกับการพัฒนาซอฟต์แวร์ไม่ว่าจะเป็นการรวบรวมรายละเอียดที่ต้องการอย่างเป็นทางการ การออกแบบ การนำไปใช้งาน การรวบรวม และทดสอบซึ่งต้องใช้งบประมาณในการดำเนินงานด้วย เปรียบเสมือนว่าเป็นโครงการซอฟต์แวร์เล็กๆ โครงการหนึ่ง

การควบคุมขอบเขตโครงร่างของซอฟต์แวร์ เป็นงานที่สำคัญในขั้นตอนนี้เพราะจะควบคุมจัดการเกี่ยวกับการเปลี่ยนแปลงต่างๆ ของซอฟต์แวร์ และควบคุมเกี่ยวกับชุดของซอฟต์แวร์ (Release and Version of Software) ที่ออกมาเป็นลำดับก่อนหลัง จะได้สะดวกในการสนับสนุนงานต่างๆ

ขั้นตอนการบำรุงรักษาระบบงานนี้ต้องการทีมงานขนาดเล็ก ซึ่งอาจจัดตั้งขึ้นมาเพื่อบำรุงรักษาผลิตภัณฑ์ซอฟต์แวร์หลายๆชนิด โดยจัดการดูแลเกี่ยวกับ การควบคุมรายละเอียดขอบเขตโครงร่างของซอฟต์แวร์ (Configuration control) การติดตั้งระบบ (Installation) และงานด้านวิศวกรรมซอฟต์แวร์ รวมถึงการบำรุงรักษาเอกสารต่างๆ ของระบบงานด้วย

เอกสารต่าง ๆ ที่ต้องปรับปรุงในระหว่างขั้นตอนนี้ ได้แก่

1. เอกสารรายละเอียดชุดของซอฟต์แวร์ (Version release documentation)
2. รายงานปัญหาต่างๆ (Problem reports)
3. เอกสารการพัฒนาระบบงานทั้งหมด (All development documentation)
4. เอกสารคู่มือการใช้งาน (User documentation)
5. บันทึกรายละเอียดการบำรุงรักษา (Maintenance logs) และรายงานการบริการลูกค้า (Customer service reports)

ปัญหาต่าง ๆ ที่อาจเกิดขึ้นในขั้นตอนการบำรุงรักษาระบบงานได้แก่

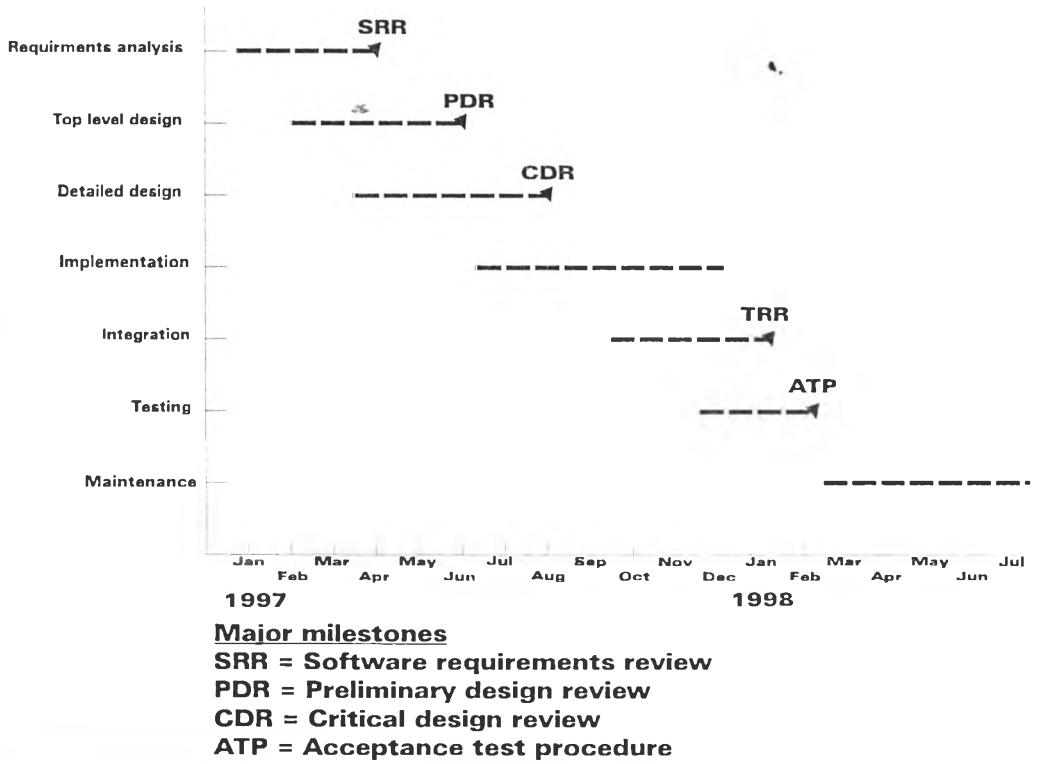
1. การขาดแคลนบุคคลากรที่มีความรู้ความสามารถด้านการบำรุงรักษาระบบงาน เนื่องจากการว่าจ้างวิศวกรที่เต็มใจจะมาทำงานด้านบำรุงรักษาระบบงานทำได้ยาก
2. ปัญหาด้านงบประมาณ เนื่องจากงบประมาณทางด้านการบำรุงรักษาระบบงานมักได้มาจากค่าธรรมเนียมบริการรายเดือน หรือรายปีของระบบที่ผู้ใช้ต้องการจะบำรุงรักษาเท่านั้น
3. การขาดแคลนอุปกรณ์ที่ช่วยสนับสนุน และใช้ในการทดสอบ ซึ่งปัญหานี้เป็นผลเนื่องมาจากการขาดแคลนงบประมาณสำหรับกิจกรรมการบำรุงรักษาระบบงานนั่นเอง

3.2 วงจรชีวิตการพัฒนาซอฟต์แวร์ตาม IEEE Standard 610.12

จะเป็นช่วงเวลาเริ่มต้น ตั้งแต่ตัดสินใจที่จะพัฒนาผลิตภัณฑ์ซอฟต์แวร์ และสิ้นสุดลงเมื่อซอฟต์แวร์นั้นถูกจัดส่งให้ผู้ใช้ ซึ่งจะแบ่งออกเป็นขั้นตอนต่างๆ ได้แก่ ขั้นตอนการวิเคราะห์ความต้องการ (Requirements phase) ขั้นตอนการออกแบบ(Design phase) ขั้นตอนการนำระบบไปใช้งาน (Implement phase) ขั้นตอนการทดสอบระบบงาน (test phase)และบางครั้งจะมีขั้นตอนการติดตั้งระบบงาน และขั้นตอนการออกระบบงาน (Check out phase) ด้วย ซึ่งรายละเอียดต่างๆ จะคล้ายกับในวงจรชีวิตการพัฒนาซอฟต์แวร์แบบ Waterfall

ภาคผนวก ง

4. <u>แผนงานเบื้องต้นของโครงการ (Master Project Plan)</u> เป็นแผนงานพัฒนาซอฟต์แวร์อย่างคร่าวๆ ของโครงการ แสดงรายละเอียดกิจกรรมที่จะกระทำในแต่ละขั้นตอน และระยะเวลา มักจะแสดงในรูป Gantt Chart



รูปภาพแสดงตัวอย่างแผนงานเบื้องต้นของโครงการ

ภาคผนวก จ

5. **เอกสารรายละเอียดความต้องการระบบงานซอฟต์แวร์ (Software Requirements Specification Document - SRS)** เป็นเอกสารที่จัดทำขึ้นเพื่อแสดงรายละเอียดความต้องการทั้งหมด เพื่อนำไปใช้ในการออกแบบ และทดสอบระบบงานซอฟต์แวร์ มีรูปแบบแนะนำดังตัวอย่างด้านล่างนี้

SOFTWARE REQUIRMEMENTS SPECIFICATION

1.0 ขอบเขตของงาน (SCOPE)

2.0 รายละเอียดเอกสารที่จะนำมาใช้ (Applicable Documents)

3.0 รายละเอียดข้อมูลทั่วไป และข้อมูลการติดต่อเชื่อมโยงระบบงาน (Interfaces / Information Description)

- 3.1 รายละเอียดข้อมูลการติดต่อเชื่อมโยงระบบงาน (Interface Description)
  - 3.1.1 Hardware
  - 3.1.2 Software
  - 3.1.3 Human
  - 3.1.4 Packaging
- 3.2 แผนผังโครงสร้างการติดต่อเชื่อมโยงระบบงาน (Interface Diagram)
- 3.3 แผนผังโครงสร้างทางเดินของข้อมูล (Data Flow Diagrams)

4.0 หน้าที่ต่าง ๆ (Functions)

- 4.n รายละเอียดของฟังก์ชั่น n
  - 4.n.1 ข้อมูลนำเข้า (Inputs)
  - 4.n.2 การประมวลผล (Processing)
  - 4.n.3 ผลลัพธ์ต่าง ๆ (Outputs)
  - 4.n.4 ความต้องการทางด้านการออกแบบ (Design Requirements)

5.0 ฐานข้อมูล (Database)

- 5.1 คุณสมบัติ (Characteristics)
- 5.2 รายละเอียดของข้อมูล (discrete Data Items)
- 5.3 การเข้าถึงข้อมูล (Access)

6.0 การรับรองคุณภาพ (Quality Assurance)

- 6.1 ความต้องการด้านความถูกต้องของระบบงาน (Validation Requirements)
- 6.2 การตรวจสอบความถูกต้อง (Validation Tests)
- 6.3 ทรัพยากรที่ใช้ (Resources)

6.4 การรวมระบบงานซอฟต์แวร์ (System Integration)

7.0 แผนผังโครงสร้างระบบงานซอฟต์แวร์ (Software Block Diagram)

8.0 รายละเอียดอื่นๆ (NOTES)

9.0 ภาคผนวก (Appendix)

10.0 อธิบายศัพท์ต่างๆ (Glossary)

รูปภาพแสดงตัวอย่างเอกสารรายละเอียดความต้องการระบบงานซอฟต์แวร์

ภาคผนวก ฉ.

6. <u>แผนงานการพัฒนาโครงการ (Project Development Plan)</u> เป็นการกำหนดขอบเขต สภาพแวดล้อมการทำงาน และข้อมูลพื้นฐานทั่วไป ของโครงการซอฟต์แวร์ทั้งหมด ซึ่งจะเชื่อมโยงกับตารางเวลาการทำงาน และต้นทุนแผนงานการพัฒนาโครงการแต่ละโครงการจะแตกต่างกันไป ขึ้นอยู่กับขนาด และความซับซ้อนของโครงการนั้น แต่โดยทั่วไป จะประกอบด้วยหัวข้อต่างๆ ดังนี้

<u>หัวเรื่อง</u>	<u>วัตถุประสงค์</u>
1. ขอบเขตงาน (Scope)	เพื่อกำหนดขอบเขตของงานที่จะดำเนินการ
2. รายละเอียดงานย่อย และการกำหนดส่งมอบ (Tasks and Deliverables)	กำหนดรายละเอียดงานย่อยๆ ที่ต้องดำเนินการ และส่งมอบงาน
3. ทรัพยากรที่ใช้ (Resources)	กำหนดขนาด และประเภททรัพยากรที่ต้องใช้
4. ตารางเวลาการทำงาน (Schedule)	แสดงรายละเอียดตารางเวลาการทำงาน และการส่งมอบ
5. ต้นทุน (Cost)	แสดงรายละเอียดต้นทุนของซอฟต์แวร์โดยประมาณ

SOFTWARE PLAN

1.0 ขอบเขตงาน (Scope)
- 1.1 วัตถุประสงค์ (objective)
- 1.2 หน้าที่ และขอบเขต (Functions and Bounds)
  - 1.2.n รายละเอียดหน้าที่หลัก (Description of major function)
- 1.3 ผลการปฏิบัติงาน (Performance)
- 1.4 ความถูกต้องเชื่อถือได้ (Reliability)
- 1.5 การติดต่อเชื่อมโยงระบบงาน (System Interfaces)
- 1.6 ข้อจำกัดด้านตารางเวลาการทำงาน (Schedule Constraints)

2.0 รายละเอียดงานย่อย และการส่งมอบ (Task and Deliverables)
- 2.1 รายละเอียดงานย่อยต่างๆ (Tasks) ตามขั้นตอนการพัฒนาซอฟต์แวร์
- 2.2 การส่งมอบงาน (Deliverables)
  - 2.2.1 เอกสารต่างๆ ทั้งหมด (Documentation)
  - 2.2.2 การจัดการฝึกอบรมการใช้งาน (Training)
  - 2.2.3 การติดตั้งระบบงาน (Installation)

3.0 ทรัพยากรที่ใช้ (Resources)
- 3.1 บุคลากร (People) (ต้องพิจารณาจำนวนคนที่ต้องการ , ประสบการณ์)
- 3.2 Hardware ที่ต้องใช้

3.3 Software ที่ต้องใช้ (เช่น ระบบปฏิบัติการ , Compilers , โปรแกรมที่ใช้ทดสอบระบบโปรแกรมระบบงาน , โปรแกรมฐานข้อมูล เป็นต้น)

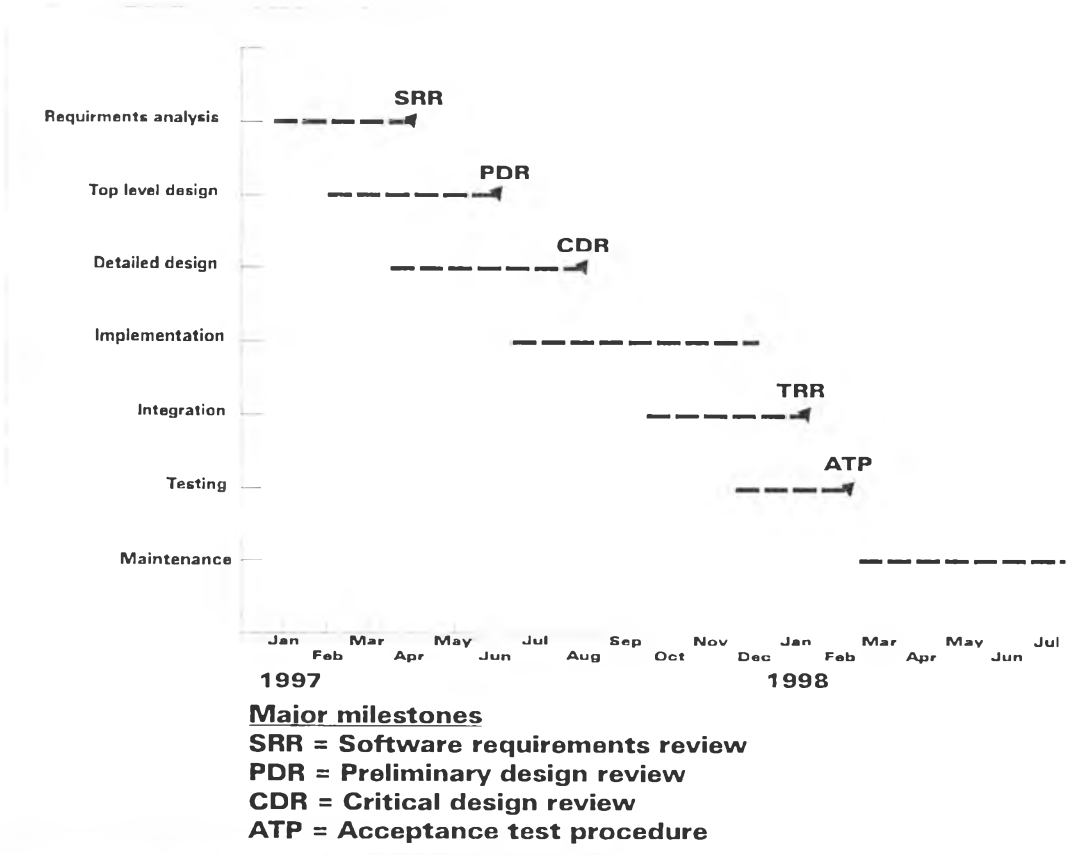
4.0 ต้นทุนต่าง ๆ (Cost) (เช่น เงินเดือนพนักงาน , เวลาที่ใช้ในการประมวลผล , การจัดเตรียมอุปกรณ์ , ค่าพาหนะ เป็นต้น)

5.0 ตารางเวลาการทำงาน (Schedule)

5.1 การกระจายทรัพยากรที่ใช้ (Allocation of Resources)

6.0 ผังโครงสร้างองค์กร (Personnel Organization) ใช้สำหรับอธิบายรายละเอียดหน้าที่งานและความรับผิดชอบของแต่ละคน (ใช้ในโครงการขนาดใหญ่)

รูปภาพแสดงตัวอย่างแผนงานการพัฒนาโครงการ



รูปภาพแสดงตารางเวลาการทำงานโครงการ (Project Schedule)

ภาคผนวก ช.

7. <u>แผนงานการทดสอบซอฟต์แวร์ (Software Test Plan)</u> เป็นเอกสารที่เกิดขึ้นในขั้นตอนการกำหนดรายละเอียดความต้องการซอฟต์แวร์ เพื่อแสดงรายละเอียดขอบเขตของการทดสอบซอฟต์แวร์ แผนงานตารางเวลา และขั้นตอนในการดำเนินงาน ซึ่งควรจะประกอบด้วยรายละเอียดดังตัวอย่างด้านล่างนี้

SOFTWARE TEST PLAN

1.0 **วัตถุประสงค์** : กำหนดขอบเขตของการทดสอบองค์ประกอบต่าง ๆ ของผลิตภัณฑ์ซอฟต์แวร์

2.0 **แผนงานสำหรับการทดสอบอย่างไม่เป็นทางการ**

3.0 **แผนงานการทดสอบหน่วยย่อยต่าง ๆ (Unit Test Plans)**

- 3.1.1 รายละเอียดความต้องการต่าง ๆ ในการทดสอบหน่วยย่อย (Unit test requirements)
- 3.1.2 รายละเอียดความรับผิดชอบในการทดสอบหน่วยย่อย (Unit test responsibilities)
- 3.1.3 รายละเอียดตารางเวลาการทำงานในการทดสอบหน่วยย่อย (Unit test schedule)

4.0 **แผนงานการรวบรวมระบบงาน และการทดสอบซอฟต์แวร์ (Integration and test plans)**

5.0 **ทรัพยากรที่ต้องการใช้สำหรับการทดสอบอย่างไม่เป็นทางการ** ได้แก่

- 5.1.1 สั่งอำนวยความสะดวกต่าง ๆ
- 5.1.2 บุคลากร
- 5.1.3 อุปกรณ์ฮาร์ดแวร์
- 5.1.4 อุปกรณ์ฮาร์ดแวร์ที่ใช้เชื่อมโยงติดต่อ และช่วยสนับสนุน
- 5.1.5 แหล่งจัดหาทรัพยากร
- 5.1.6 ขอบเขตรายละเอียดการทดสอบ

6.0 **แผนงานสำหรับการทดสอบอย่างเป็นทางการ**

7.0 **ราบละเอียดความต้องการต่าง ๆ ในการทดสอบอย่างเป็นทางการ**

8.0 **การสรุปรายละเอียดการทดสอบอย่างเป็นทางการ**

9.0 **แผนงานตารางเวลาการทดสอบอย่างเป็นทางการ**

10.0 **การวิเคราะห์ข้อมูลที่ใช้ในการทดสอบ**

11.0 **สมมติฐาน และข้อจำกัดต่าง ๆ ที่ใช้ในการทดสอบ**

12.0 **ภาคผนวก**

รูปภาพแสดงตัวอย่างแผนงานการทดสอบซอฟต์แวร์ (Software test Plan)

ภาคผนวก ซ

8. <u>**เอกสารรายละเอียดการออกแบบ (Design Specification Document)**</u> เป็นเอกสารที่สำคัญที่สุดในการพัฒนาซอฟต์แวร์ เพราะการออกแบบจะเป็นเสมือนแนวทางให้กับขั้นตอนการนำซอฟต์แวร์ไปใช้งาน และการทดสอบ ซึ่งจะแสดงให้เห็นถึงความสัมพันธ์ระหว่างรายละเอียดหน้าที่ต่างๆ (functional details) และโครงสร้างของซอฟต์แวร์ (Software structure) มีรายละเอียดดังตัวอย่างด้านล่างนี้

DESIGN SPECIFICATION DOCUMENT

1.0 **ขอบเขตของงาน** (เช่น ฮาร์ดแวร์ , ซอฟต์แวร์ , การติดต่อเชื่อมโยงระบบ หน้าที่หลักต่างๆ และส่วนที่เกี่ยวข้องกับการประมวลฐานข้อมูล ข้อจำกัดของการออกแบบที่สำคัญ เป็นต้น)

2.0 **เอกสารที่เกี่ยวข้อง** ได้แก่ คู่มือมาตราฐานต่างๆ ที่ถูกอ้างถึงในเอกสารการออกแบบ

3.0 **รายละเอียดการออกแบบ (Design Description)** ได้แก่

- 3.1 ความสัมพันธ์ระหว่างหน้าที่หลักของซอฟต์แวร์ และโครงสร้างของซอฟต์แวร์ (Functional Allocation)
- 3.2 ผังทางเดินของข้อมูล/ โครงสร้าง (Data Flow / structure)
  - 3.2.1 รายละเอียดทางเดินของข้อมูล โครงสร้างข้อมูล และองค์ประกอบต่างๆ
  - 3.2.2 การแสดงรูปภาพของผังทางเดินของข้อมูล / โครงสร้าง
- 3.3 โครงสร้างของซอฟต์แวร์ (Software Structure) จะแสดงความสัมพันธ์ระหว่างระบบย่อยต่างๆ และลำดับขั้นของการควบคุม และการประมวลผล

4.0 **ระบบย่อยต่าง ๆ (Modules)**

- 4.X รายละเอียดของระบบย่อย X
  - 4.X.1 รายละเอียดการประมวลผล (Processing Narrative)
  - 4.X.2 รายละเอียดการติดต่อเชื่อมโยงระบบ (Interface Description)
  - 4.X.3 รายละเอียดภาษาที่ใช้ในการออกแบบ (Design Language Description)
  - 4.X.4 ระบบย่อยต่างๆ ที่ถูกใช้ (Modules Used)
  - 4.X.5 ข้อคิดเห็นต่างๆ (เช่น ข้อจำกัดต่างๆ ข้อผิดพลาดที่พบ)
  - 4.X.6 การจัดการข้อมูล (Data Organization)

5.0 **โครงสร้างข้อมูล และข้อมูลพื้นฐานทั่วไปที่ใช้ในฐานข้อมูล (File structure and Global data)**

- 5.1 รายละเอียดของโครงสร้างแฟ้มข้อมูล (เช่น ชื่อ ขนาด วิธีการเข้าถึงข้อมูล เป็นต้น)

5.1.1 โครงสร้างแฟ้มข้อมูล (File structure)

5.1.2 รายละเอียดรูปแบบ และองค์ประกอบของข้อมูลในแต่ละแฟ้มข้อมูล (logical Record Description)

5.2 ข้อมูลพื้นฐานทั่วไปที่ใช้ในฐานข้อมูล (Global data)

5.3 ความสัมพันธ์ระหว่างระบบย่อยต่าง ๆ และแฟ้มข้อมูล หรือข้อมูลพื้นฐานที่ใช้ (Cross Reference)

6.0 **การตรวจสอบรับรองคุณภาพ (Quality Assurance)**

6.1 ความต้องการในการตรวจสอบความถูกต้อง (Validation Requirements) แสดงรายละเอียดสิ่งที่ต้องการตรวจสอบ หรือจัดทำแผนงาน และขั้นตอนทดสอบความถูกต้อง

6.2 การทดสอบความถูกต้อง (validation Tests) แสดงรายละเอียดประเภท และเทคนิคการทดสอบที่จะจัดทำ

6.3 ทรัพยากรที่ใช้ (Resources) (เช่น บุคลากร เวลาคอมพิวเตอร์ที่ใช้ ซอฟต์แวร์ที่ต้องการเพิ่มเติม)

6.4 การรวบรวมระบบงาน (Integration) แสดงรายละเอียดความต้องการซึ่งไม่สามารถตรวจสอบได้จนกว่าจะมีการทดสอบการรวมกันของระบบงาน

7.0 **แผนผังโครงสร้างซอฟต์แวร์ (Software Block Diagram)** แสดงรายละเอียดผังโครงสร้างของแต่ละหน้าที่ของซอฟต์แวร์ และความสัมพันธ์

8.0 **หมายเหตุรายละเอียดประกอบต่าง ๆ แสดงรายละเอียดข้อมูลเพิ่มเติมต่าง ๆ**

9.0 **ภาคผนวก**

10.0 **อธิบายคำศัพท์ต่าง ๆ ที่ใช้**

รูปภาพแสดงตัวอย่างเอกสารรายละเอียดการออกแบบ (Design specification Document)

ภาคผนวก ฎ

9. **เครื่องมือหรือเทคโนโลยี (Software Tools) ที่ช่วยในการดำเนินงานในแต่ละขั้นตอนการทำงานในการพัฒนาซอฟต์แวร์** ได้แก่

4.1 **เครื่องมือที่ช่วยในการวิเคราะห์ความต้องการ (Software Tools for requirements analysis)**

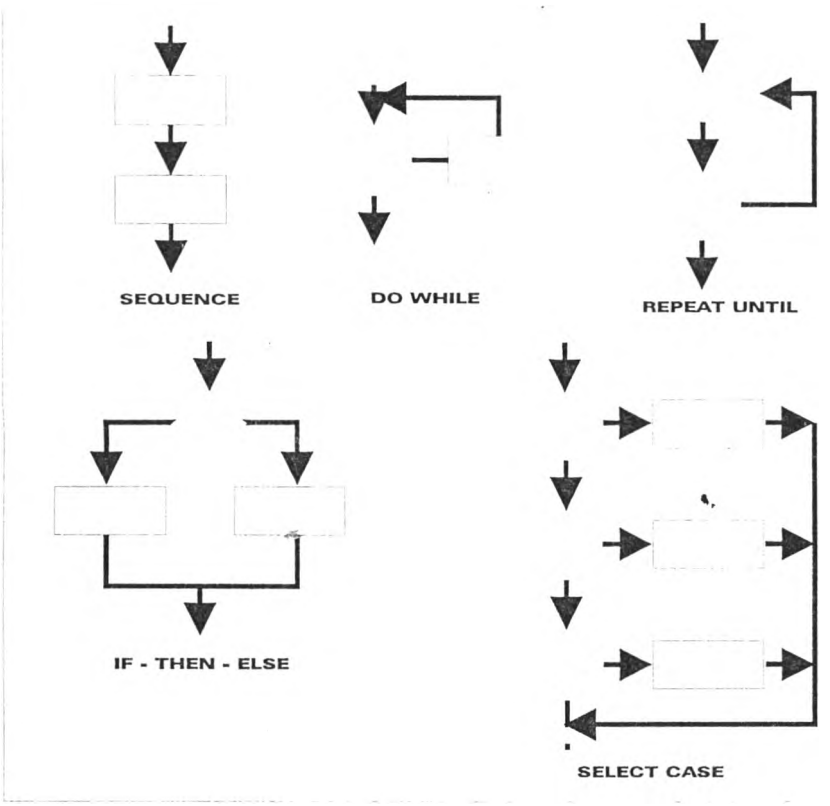
4.1.1 **PSL/PSA (Problem Statement Analyzer)** เป็นเครื่องมือช่วยในการกำหนดรายละเอียดขอบเขตความต้องการซอฟต์แวร์ PSL เป็นภาษาที่ใช้อย่างเป็นทางการสำหรับอธิบายรายละเอียดระบบ ส่วน PSA เป็นซอฟต์แวร์สำเร็จรูปเพื่อการวิเคราะห์ โดยจะประมวลผลจากรายละเอียด PSL PSL/PSA จะทำการสร้างผังทางเดินของข้อมูล (Data Flow Diagram) ขึ้นมาและจัดเตรียมสิ่งอำนวยความสะดวกในการจัดทำรายละเอียดของข้อมูล (Data Dictionary) และการออกแบบโครงสร้าง (Structured design)

4.2 **เครื่องมือที่ช่วยในการออกแบบ (Software Tools for design)**

4.2.1 **เครื่องมือในรูปแบบกราฟฟิค (graphical Tools)** มี 2 แบบคือ

- ผังแสดงโครงสร้าง (Flowcharts) เป็นเครื่องมือช่วยในการออกแบบขั้นละเอียด (Detailed design) เพื่อใช้แสดงโครงสร้างของโปรแกรมหรือในบางกรณี ใช้แสดงรายละเอียดของข้อมูลและการเข้าถึงข้อมูล

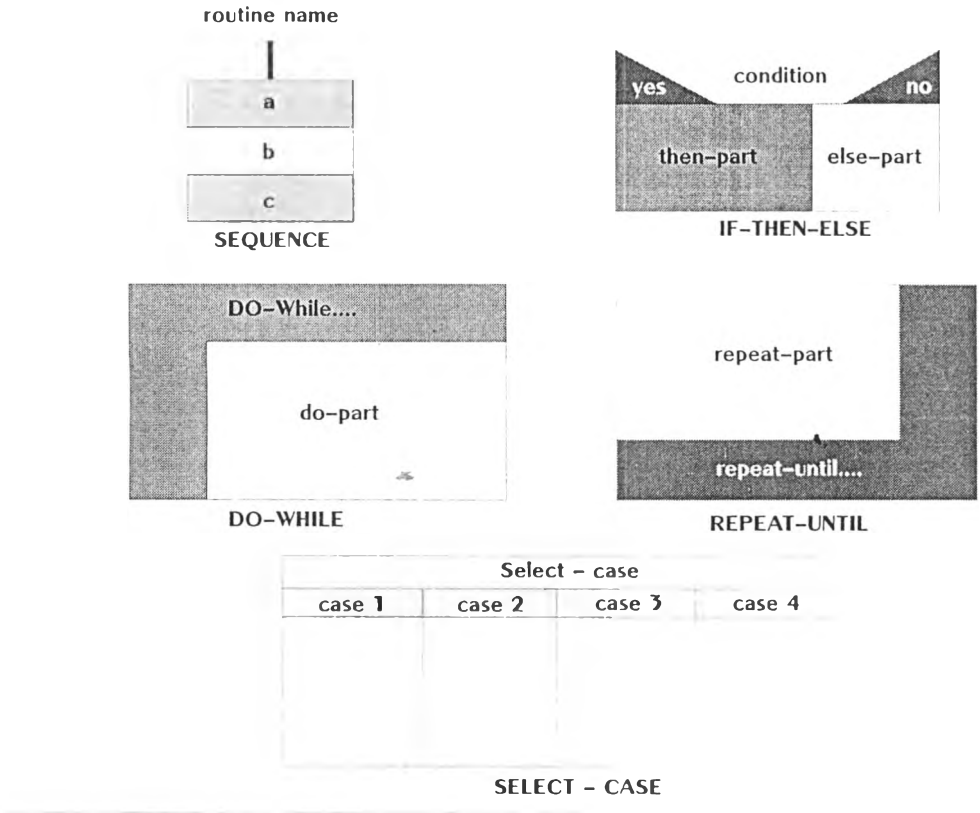
ผังแสดงโครงสร้างนี้เป็นที่รู้จักกันทั่วไป สามารถแสดงรายละเอียดเงื่อนไข ความสัมพันธ์ต่าง ๆ กิจกรรมที่ต้องกระทำ และรายละเอียดย่อย ๆ ข้อดีก็คือเป็นที่คุ้นเคยใช้กันอย่างแพร่หลาย อ่านและเข้าใจง่าย และเป็นที่ยอมรับโดยทั่วไป



รูปภาพแสดงตัวอย่างผังแสดงโครงสร้าง (Flowchart Constructs)

ปัจจุบันมีเครื่องมือที่ชื่อว่า Flowcharter เป็นโปรแกรมซอฟต์แวร์ที่รับค่ารายละเอียดการออกแบบ หรือรายละเอียดรหัสโปรแกรมเข้าไปแล้ว Flowcharter นี้ จะทำการสร้างผลลัพธ์ออกมาเป็นผังแสดงโครงสร้างโปรแกรม (Program flowchart)

- <u>ผังแสดงโครงสร้างแบบกล่อง (Box Diagrams)</u> เป็นเครื่องมือที่ช่วยในการออกแบบโครงสร้างอีกอย่างหนึ่ง บางครั้งเรียกว่า Nassi-Shneiderman หรือ Chapin Charts แต่ไม่ค่อยเป็นที่นิยมแพร่หลาย เท่ากับ Flowchart วิธีการนี้จะแสดงรายละเอียดขั้นตอนต่าง ๆ ในรูปกล่องสี่เหลี่ยม และอาจมีลักษณะซ้อน ๆ กันสำหรับกิจกรรมที่ต้องกระทำ



รูปภาพแสดงตัวอย่างผังแสดงโครงสร้างแบบกล่อง (Box Diagrams)

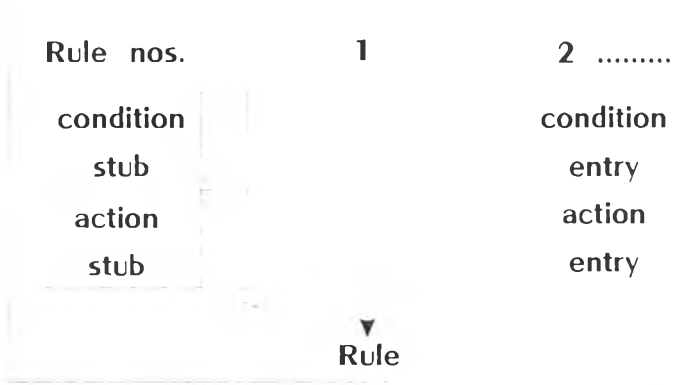
4.2.2 **เครื่องมือในรูปแบบภาษาที่ช่วยในการออกแบบโปรแกรม (Program Design Language-PDL)** หรือเรียกว่า Pseudo-Code หรือ Structured-English ก็ได้ เป็นวิธีการที่จะเตรียมโครงสร้างของข้อมูลและการประมวลผลออกมาในรูปแบบของข้อความ เพื่อง่ายในการนำไปแปลงเป็นรหัสโปรแกรม หรือช่วยในการเขียนโปรแกรมได้สะดวกขึ้น

เครื่องมือชนิดนี้เป็นเครื่องมือที่มีความยืดหยุ่นมากเหมาะแก่การออกแบบขั้นละเอียดมากกว่าเครื่องมือชนิดอื่น ๆ ง่ายต่อการเรียนรู้การจัดทำก็ง่ายสามารถแก้ไขได้สะดวก และที่สำคัญที่สุดคือง่ายมากในการนำไปแปลงเป็นรหัสโปรแกรม

```
perform calculations “a”
REPEAT
        perform operation “b”
        REPEAT
                IF Condition “x1”
                        perform “c”
                        IF Condition “x3”
                                REPEAT
                                    perform calculation “e”
                                UNTIL Condition “x4”
                        ELSE
                                    perform "d"
                        END IF
                ELSE
                        perform “f”
                        REPEAT
                                    perform operation “g”
                        UNTIL  Condition “x2”
                END IF
                Perform “h”
        UNTIL Condition “x5”
    UNTIL Condition “x6”
perform operation “i”
EXIT
```

รูปภาพแสดงตัวอย่างภาษาที่ช่วยในการออกแบบ (Design Language Example)

4.2.3 <u>เครื่องมือที่ช่วยในการออกแบบในรูปแบบตาราง (Tobular Tools)</u> หรือเรียกว่า ตารางการตัดสินใจ (Decision Table) เป็นเครื่องมือที่ช่วยในการออกแบบอย่างเป็นระบบเพื่อแสดงกิจกรรมต่างๆ เงื่อนไข และความสัมพันธ์ของกิจกรรม



รูปภาพแสดงรูปแบบพื้นฐานของตารางการตัดสินใจ

ตารางจะประกอบด้วย 4 ส่วนหลักๆ ดังแสดงในภาพ แม้ว่าเราจะสามารถแปลงตารางการตัดสินใจไปเป็นรหัสโปรแกรมได้โดยไม่มีข้อผิดพลาดเกิดขึ้น แต่ถ้าเปรียบเทียบกับเครื่องมือชนิดอื่น ตารางการตัดสินใจนี้จะค่อนข้างใช้ยากกว่า แต่มีข้อดีคือตรวจสอบความถูกต้องได้ง่าย และสามารถแสดงถึงความสัมพันธ์ของกิจกรรมที่ซับซ้อนให้อยู่ภายใน 1 หน้ากระดาษได้ ซึ่งเครื่องมือชนิดอื่นไม่สามารถทำได้

Rules	1	2	3	4	5	6	7	8
Capacity more than 1000	Y	Y	Y	Y	N	N	N	N
poor maintenance record	Y	Y	Y	Y	N	N	N	N
more than 10 yrs. Old	Y	N	Y	Y	Y	N	N	N
priority treatment	X	X	X	X	X			
other treatment						X	X	X

รูปภาพแสดงตัวอย่างตารางการตัดสินใจ (Decision Table Example)

นอกจากเครื่องมือที่ช่วยในการออกแบบดังกล่าวไปแล้วข้างต้น ในปัจจุบันยังมีเครื่องมือที่ช่วยในการออกแบบทางด้านวิศวกรรมอย่างเป็นระบบ (Computer Aided System Engineering - CASE) เครื่องมือนี้จะช่วยสนับสนุนในการสร้างผังไดอะแกรม ตรวจสอบความถูกต้องของการออกแบบ การควบคุมไลบรารี่ของข้อมูล การควบคุมขอบเขตโครงร่าง การจัดทำผลลัพธ์ และ การรายงาน

4.3 <u>เครื่องมือที่ช่วยในการนำซอฟต์แวร์ไปใช้งาน (Software Tools for Implementation)</u> เป็นวิธีการหรือเทคนิคที่ใช้อำนวยความสะดวกในขั้นตอนการสร้างรหัสโปรแกรม (Generation of source code) และการตรวจสอบความถูกต้องของโปรแกรม (Verification of source code) แบ่งออกเป็น 3 กลุ่มใหญ่ๆ ดังนี้ คือ

4.3.1 **<u>เครื่องมือที่ช่วยในการจัดเตรียมรหัสโปรแกรม (Source code Preparation)</u>** เครื่องมือที่ใช้ในการบันทึก (Editor) มักพบในทุกๆ ระบบปฏิบัติการซึ่งโปรแกรมนี้ควรมีโครงสร้างคำสั่งที่เรียนรู้และใช้งานได้ง่าย

โปรแกรมที่ช่วยในการบันทึกที่ดีควรมีคุณสมบัติคือ ทำงานได้ทั้งเป็นแบบบันทัด และตัวอักษร ควรมีคำสั่งในการค้นหา (Find) และคำสั่งแทรกตัวอักษร (Insert) คำสั่งที่ใช้สั้นกระทัดรัด สะกด และเข้าใจง่าย ข้อความที่แสดงความผิดพลาด (Error Messages) ควรจะอ่านแล้วเข้าใจ นอกจากนั้นควรมีการสำรองข้อมูลไว้เผื่อความเสียหายของรหัสโปรแกรมด้วย

ในปัจจุบันเนื่องจากอยู่ในยุคของภาษาโปรแกรมระดับที่ 4 (4th Generation Langrage) จะมีเครื่องมือที่ทันสมัยมาขึ้นในการช่วยเขียนโปรแกรม เช่น เครื่องมือที่ใช้เขียนโปรแกรมในลักษณะตาราง สามารถระบุตัวแปร เงื่อนไขของโปรแกรม รูปแบบของจอภาพ และรายงานที่ต้องการได้ โดยไม่ต้องเขียนโปรแกรมด้วยคำสั่งยาวๆ หลายๆ บรรทัด และไม่ต้องมีไวยากรณ์โปรแกรมที่ซับซ้อน สามารถนำไปใช้งานได้ทันที ไม่ต้องผ่านการประมวลผลโดยตัวแปลภาษา (Compiler)

4.3.2 **<u>เครื่องมือที่ช่วยในการประมวลผลภาษา (Language Processing</u>**) เครื่องมือหลายชนิดถูกสร้างขึ้นมาเพื่อช่วยเพิ่มความสมบูรณ์ของตัวแปลภาษา (Programming Langrage Compiler) ซึ่งได้แก่ Precompiler , Macroprocessor และคอมไพเลอร์ชนิดพิเศษ

- Precompiler ช่วยในการประมวลผลก่อนโดยทันที ก่อนที่จะเข้าไปประมวลผลโดยคอมไพเลอร์ จริง ซึ่งจะช่วยสร้างรูปแบบของไวยากรณ์ที่ถูกต้องและเป็นที่ยอมรับของคอมไพเลอร์
- Macroprocessor เป็นซอฟต์แวร์ที่ช่วยในส่วนก่อนการคอมไพล์เช่นกัน ใช้ประโยชน์สำหรับการตรวจสอบความผิดพลาดของโปรแกรม (Debugging)
- คอมไพเลอร์ชนิดพิเศษ จะมีการเพิ่มเติมคุณสมบัติพิเศษเข้าไป เช่น การสร้างรหัสโปรแกรมบางส่วนเพิ่มเติมให้โดยอัตโนมัติ

นอกจากนี้ยังมีเครื่องมือที่ช่วยในขั้นตอนการเขียนโปรแกรม และการตรวจสอบความถูกต้องอีกหลายชนิด ได้แก่

- Comparator เป็นเครื่องมือทางด้านซอฟต์แวร์ที่ช่วยในการเปรียบเทียบระหว่างโปรแกรมคอมพิวเตอร์ 2 โปรแกรม เปรียบเทียบไฟล์ หรือชุดของข้อมูล เพื่อให้ทราบถึงความเหมือนหรือความแตกต่าง
- Cross - Reference Generator เป็นเครื่องมือทางด้านซอฟต์แวร์ ที่เมื่อนำเข้าข้อมูลรหัสโปรแกรมเข้าไป จะทำการสร้างผลลัพธ์ เป็นรายละเอียดที่จะแสดงถึงตัวแปรทั้งหมดของโปรแกรมนั้น และการกำหนดเงื่อนไขต่างๆ พร้อมทั้งยังระบุด้วยว่าบรรทัดไหนของโปรแกรมที่ใช้ตัวแปร หรือ การกำหนดเงื่อนไขนั้นๆ
- Decompiler เป็นเครื่องมือทางด้านซอฟต์แวร์ที่ช่วยในการแปลโปรแกรมคอมพิวเตอร์จากรูปแบบภาษาเครื่อง (Machine language) ไปเป็นรูปแบบที่ใกล้เคียงกับต้นฉบับโปรแกรมภาษาระดับสูง
  (Original high order language program) ซึ่งจะตรงกันข้ามกับตัวคอมไพเลอร์

4.4 <u>เครื่องมือที่ช่วยในการทดสอบ (Software Tools for Testing)</u> เป็นวิธีการ หรือเทคนิคที่ช่วยในการตรวจสอบว่าผลการทำงาน และหลักเกณฑ์ในการออกแบบ เป็นที่น่าพอใจ หรือไม่ ซึ่งได้แก่

4.4.1 <u>เครื่องมือวิเคราะห์แผนผังโครงสร้างโปรแกรม (Program Flow Analyzer)</u> เป็นซอฟต์แวร์ที่ช่วยในการตรวจสอบความถูกต้องของการใช้รูปแบบ ของการคำนวณทางสถิติในโปรแกรม เพื่อป้องกันความผิดพลาดที่อาจจะเกิดขึ้น

4.4.2 <u>Test Drivers, Scripts, Data Generators</u> เป็นซอฟต์แวร์ที่ช่วยในการทดสอบ ซึ่งจะจัดเตรียมข้อมูลนำเข้าสำหรับการทดสอบ การควบคุม และติดตามการประมวลผล และการรายงานผลการทดสอบ เพื่อช่วยลดความผิดพลาดที่จะตรวจพบเมื่อทดสอบอีกครั้งTest Data Generators หรือ เรียกว่า Test Case generator เป็นซอฟต์แวร์ที่ช่วยในการสร้างข้อมูลทดสอบให้จากการนำเข้ารายละเอียดโปรแกรม ขอบเขตการทดสอบ และคุณสมบัติของโครงสร้างข้อมูล

4.4.3 <u>การทดสอบสภาพแวดล้อมการทำงานจริง (Test Bed)</u> เป็นการทดสอบคอมพิวเตอร์ทุกเครื่อง และการเชื่อมโยงติดต่อระหว่างเครื่องคอมพิวเตอร์ทุกเครื่อง และการเชื่อมโยงติดต่อระหว่างเครื่องคอมพิวเตอร์ และโปรแกรม (Hardware Software Interfaces) รวมทั้งข้อมูลนำเข้า และผลลัพธ์จริง เพื่อทดสอบโปรแกรมว่า เวลาที่ใช้ไปจริงเป็นเท่าไร

4.4.4 <u>เครื่องมือที่ช่วยในการวิเคราะห์ผลการปฏิบัติงาน (Performance Analysis Tools)</u> จะช่วยในการวิเคราะห์ และวัดผลเกี่ยวกับผลการดำเนินงาน และประสิทธิภาพ เครื่องมือในกลุ่มนี้ได้แก่

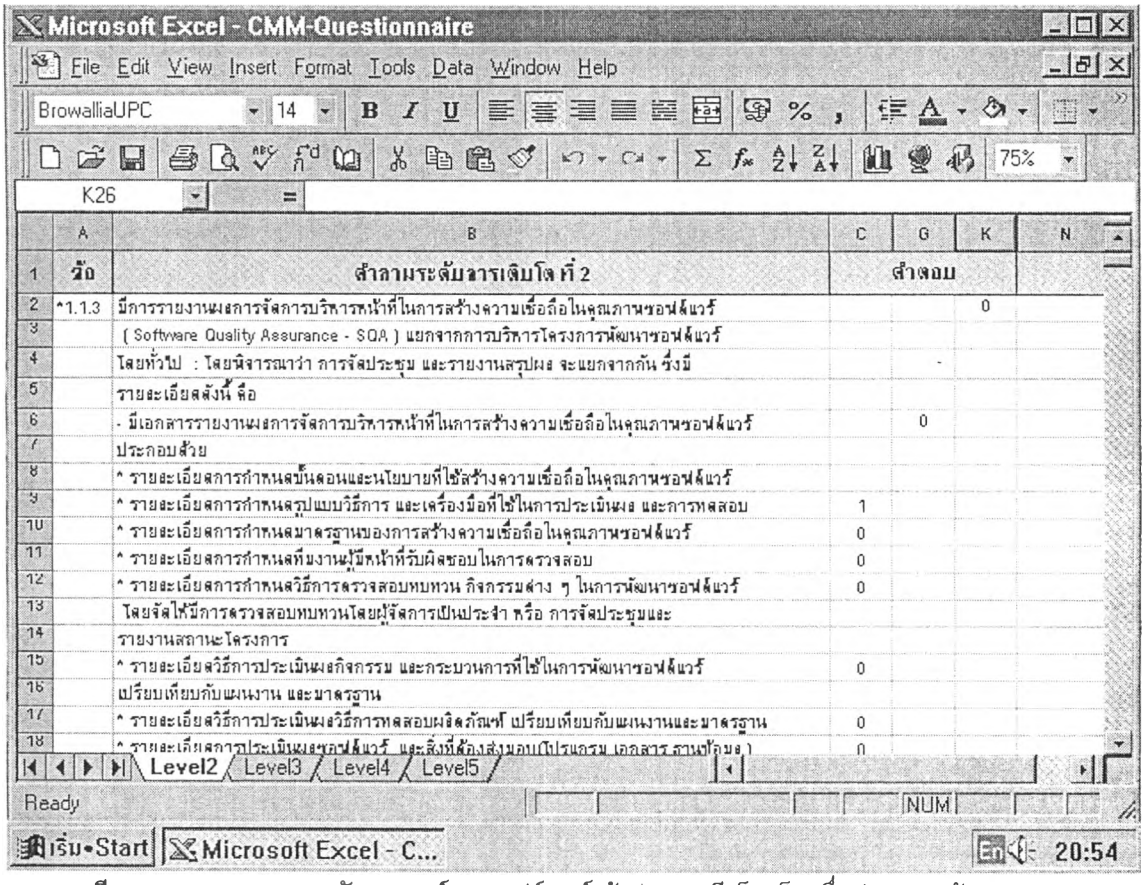
- Monitors เป็นโปรแกรมซอฟต์แวร์ หรืออุปกรณ์ทางด้านฮาร์ดแวร์ที่ใช้ในการดำเนินการอย่างต่อเนื่องกับระบบ หรือ องค์ประกอบต่างๆ ของระบบ ในการบันทึก วิเคราะห์ หรือตรวจสอบ การดำเนินงาน เช่น การนับและจับเวลาในการประมวลผลการรอคอย ตรวจสอบกิจกรรม Input output หรือตรวจสอบกิจกรรมการประมวลผลของ CPU เป็นต้น
- Timing Analyzer เป็นโปรแกรมซอฟต์แวร์ที่ช่วยในการประมาณค่า หรือ การวัด เวลาที่ใช้ในการประมวลผลของโปรแกรมคอมพิวเตอร์ หรือส่วนของโปรแกรม โดยการหาผลรวมของเวลาที่ใช้ในการประมวลผล แต่ละคำสั่งไปเรื่อยๆ ตามขั้นตอนของโปรแกรม หรือใช้วิธีแทรกจุดที่ต้องการวัดผลเป็นช่วงๆ ในโปรแกรม แล้ววัดผลของเวลาที่มาใช้ในการประมวลผลเป็นช่วงๆ ไป ในแต่ละจุด

ภาคผนวก ฏ

10. คู่มือการใช้งานโปรแกรม

ในการวิจัยเรื่องการออกแบบการพัฒนาองค์กรซอฟต์แวร์เข้าสู่ระบบซีเอ็มเอ็ม จะประกอบด้วยโปรแกรม2 ส่วนคือ

1. โปรแกรมแบบทดสอบระดับการเติบโตขององค์กรซอฟต์แวร์ ชื่อไฟล์ Cmmq&a.xls เป็นโปรแกรมแบบทดสอบ แสดงคำถามในแต่ละระดับของ CMM ตั้งแต่ระดับที่ 2 ถึงระดับที่ 5 ให้ผู้ใช้บันทึกคำตอบลงในตารางโปรแกรมจะทำการรวมคะแนนและแบ่งระดับให้ว่า องค์กรที่ตอบแบบทดสอบนั้นอยู่ในระดับการเติบโตใด
2. โปรแกรมการออกแบบพัฒนาองค์กรเข้าสู่ระบบซีเอ็มเอ็ม ชื่อไฟล์ Start.htm เป็นโปรแกรมแสดง



รายละเอียดการออกแบบการพัฒนาองค์กรซอฟต์แวร์เข้าสู่ระบบซีเอ็มเอ็ม ซึ่งประกอบด้วยรายละเอียดวิธีการปฏิบัติหลักในแต่ละระดับแบ่งออกเป็น วัตถุประสงค์ เป้าหมายที่ต้องปฏิบัติในแต่ละระดับ รายละเอียดการดำเนินการ สิ่งที่ต้องเตรียมเพื่อสนับสนุนการดำเนินการ การตรวจสอบว่าบรรลุเป้าหมายหรือไม่ ข้อพิจารณาเพื่อปรับปรุงองค์กร และข้อควรคำนึงในการนำมาใช้กับสังคมไทย

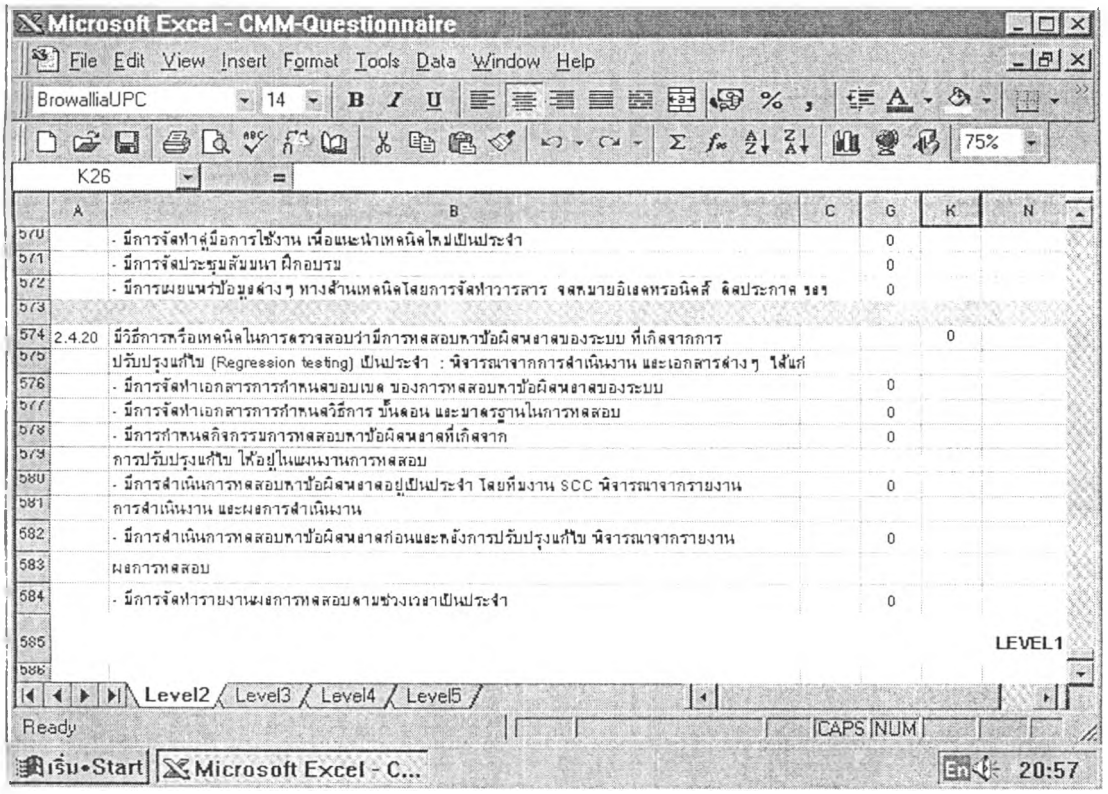
1. โปรแกรมแบบทดสอบระดับการเติบโตขององค์กรซอฟต์แวร์ ชื่อไฟล์ Cmmq&a.xls ต้องทำงานภายใต้โปรแกรม Microsoft Excel 97 ประกอบด้วยหมายเลขข้อ คำถามแต่ละระดับการเติบโตและคำตอบ

   1.1 ผู้ใช้จะต้องเริ่มจาก Sheet Level 2 ก่อน

1.2 การกรอกแบบสอบถามในแต่ละช่อง แบ่งออกเป็น

- ช่อง C ใช้ตอบคำถามข้อย่อย(ซึ่งเป็นช่องที่ละเอียดที่สุด)ที่มีเครื่องหมาย *
- ช่อง G ใช้ตอบคำถามข้อที่มีเครื่องหมาย –
- ช่อง K ใช้ตอบคำถามข้อที่เป็นข้อใหญ่

โดยเริ่มกรอกจากช่องที่ละเอียดที่สุดของคำถามแล้วโปรแกรมจะทำการรวมคะแนนแล้วแจ้งให้ทราบโดยอัตโนมัติ



1.3 ในแต่ละช่วงคำถามด้านล่างขวามือตรงช่อง N จะปรากฏผลการรวมคะแนนถ้าระดับไหนประกอบด้วยคำถาม 2 แบบ (ทั้งที่มี * และไม่มี * หน้าหมายเลขข้อ) โปรแกรมจะแสดงผลเป็น 2 ช่วงคือ ท้ายคำถามข้อที่มี * และท้ายของคำถามข้อที่ไม่มี *

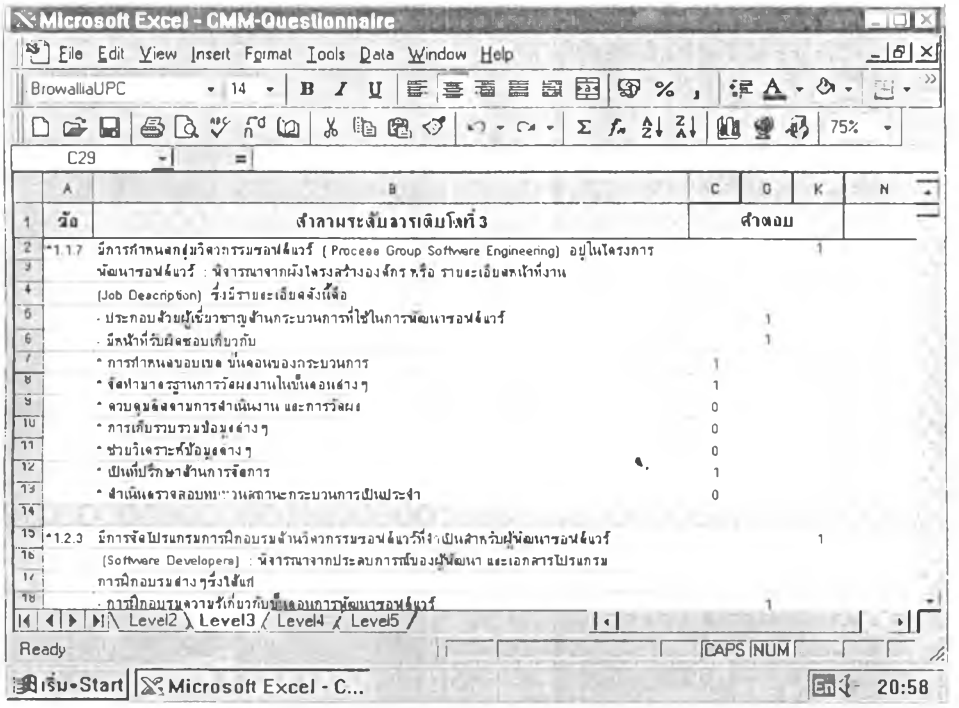
ผลลัพธ์ที่ได้จะบอกให้ทราบว่าอยู่ระดับใด และต้องไปกรอกในระดับถัดไปหรือไม่ ตัวอย่างเช่น

NEXT2 หมายถึงให้ทำแบบทดสอบระดับที่ 2 ต่อไป

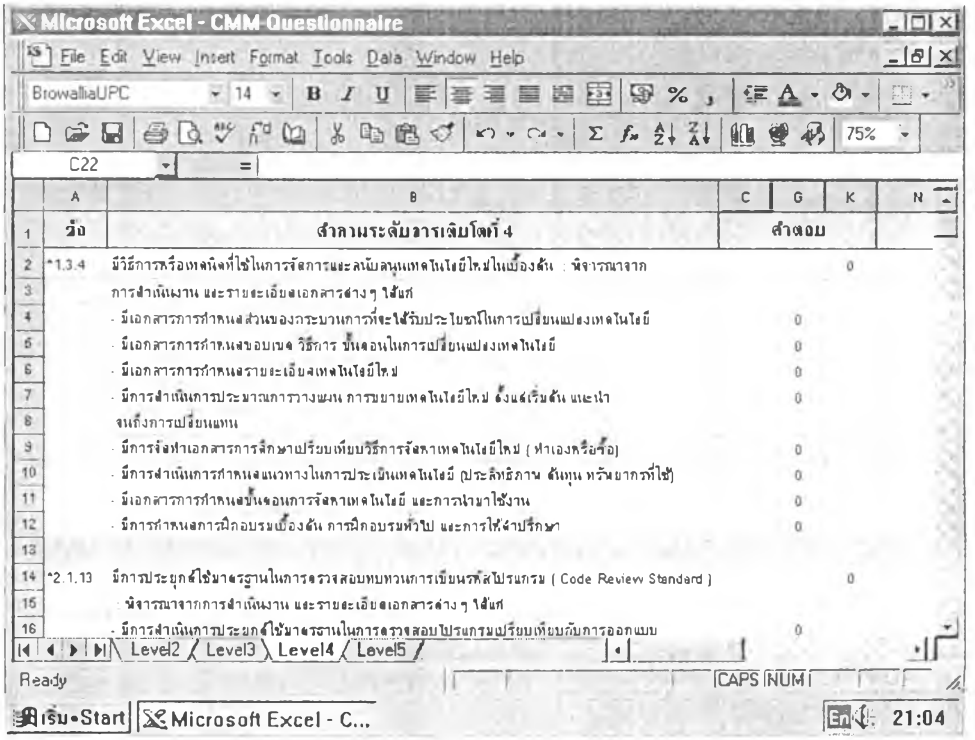
NEXT3 หมายถึงให้ทำแบบทดสอบระดับที่ 3 ต่อไป

LEVEL2 หมายถึงองค์กรมีระดับการเติบโตที่ 2

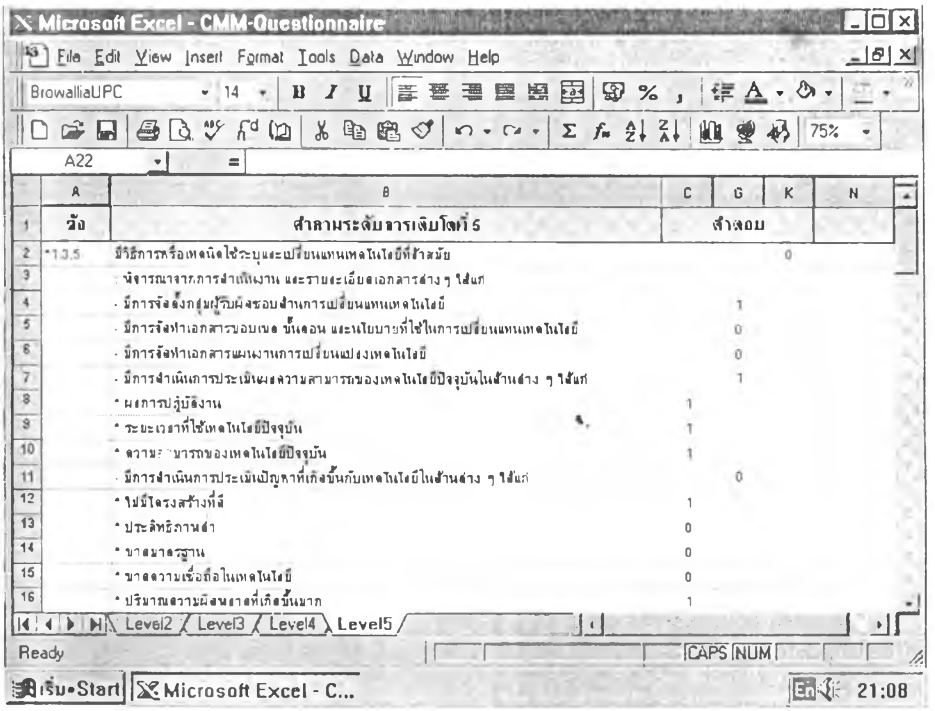
LEVEL3 หมายถึงองค์กรมีระดับการเติบโตที่ 3



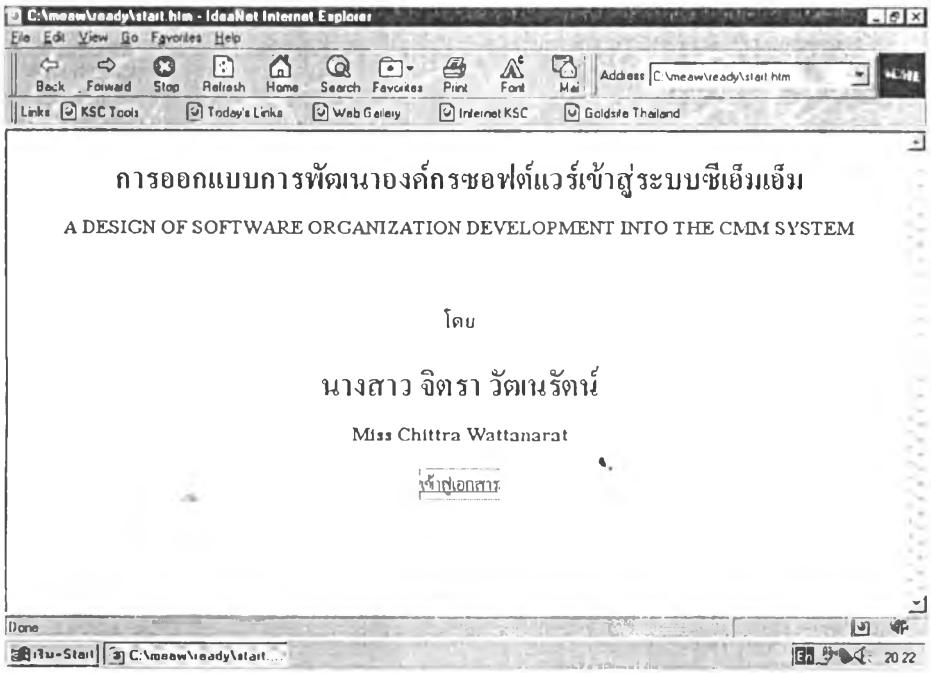
รูปภาพแสดงตัวอย่างคำถามระดับการเติบโตที่ 3



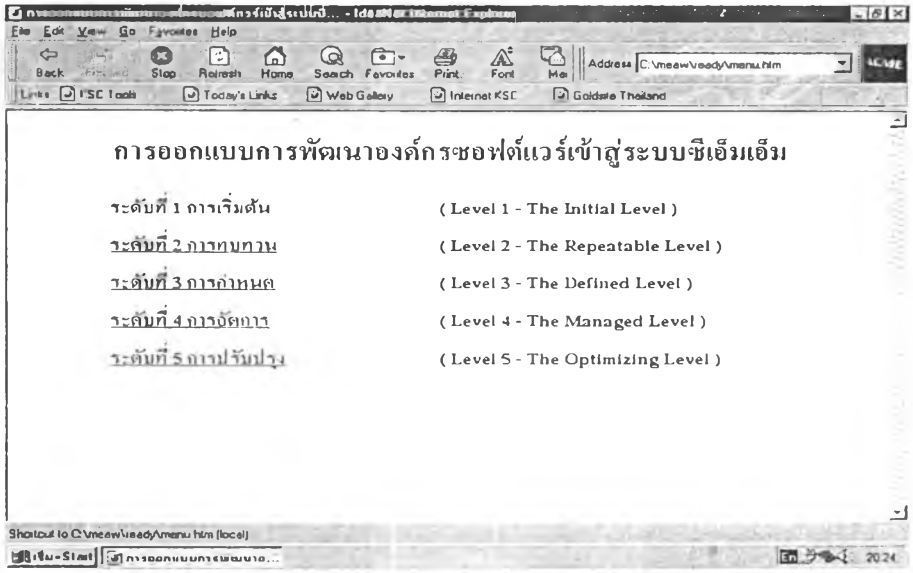
รูปภาพแสดงตัวอย่างคำถามระดับการเติบโตที่ 4

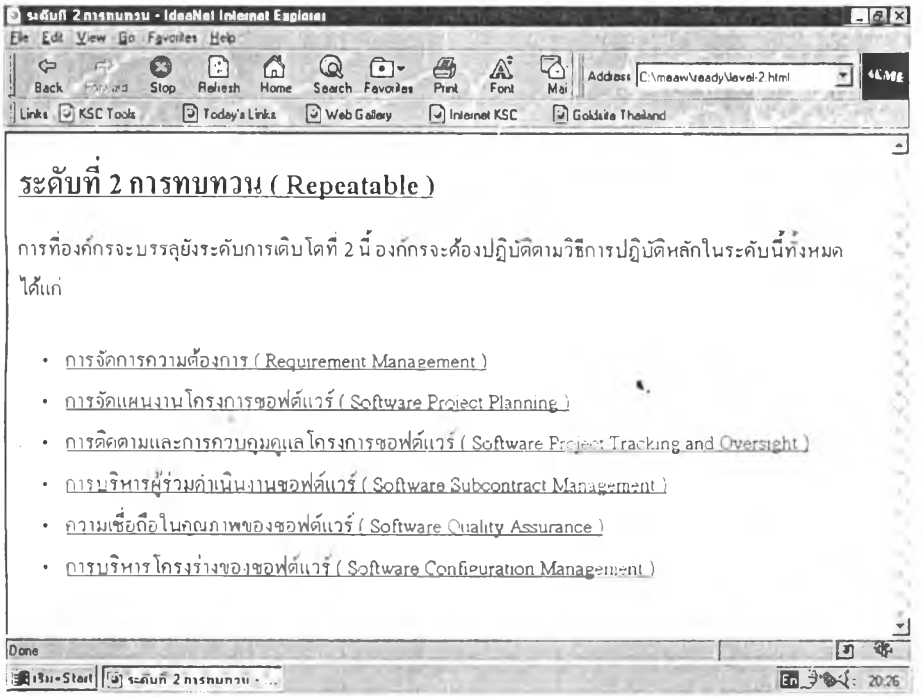


รูปภาพแสดงตัวอย่างคำถามระดับการเติบโตที่ 5

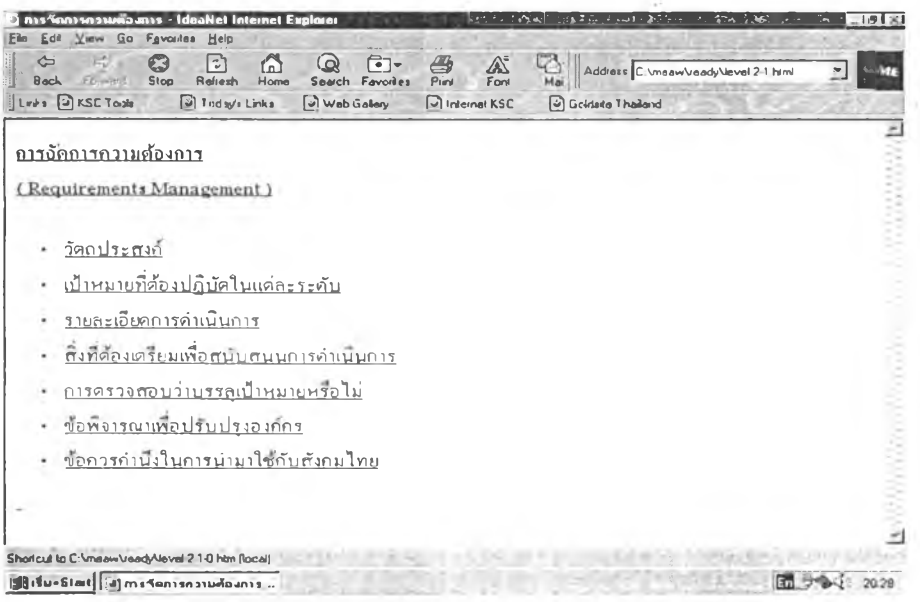


2. โปรแกรมการออกแบบการพัฒนาองค์การซอฟต์แวร์เข้าสู่ระบบซีเอ็มเอ็ม ชื่อไฟล์ Start.htm ทำงานภายใต้โปรแกรม Browser Internet Explorer v3.02 หลังจากที่ผู้ใช้ทำแบบทดสอบเรียบร้อยแล้วผู้ใช้จะทราบว่าองค์กรของตนเองอยู่ในระดับใด ผู้ใช้ก็จะสามารถเข้ามาเลือกดูในระดับของตัวเองเพื่อเป็นแนวทางในการพัฒนาองค์กรต่อไป

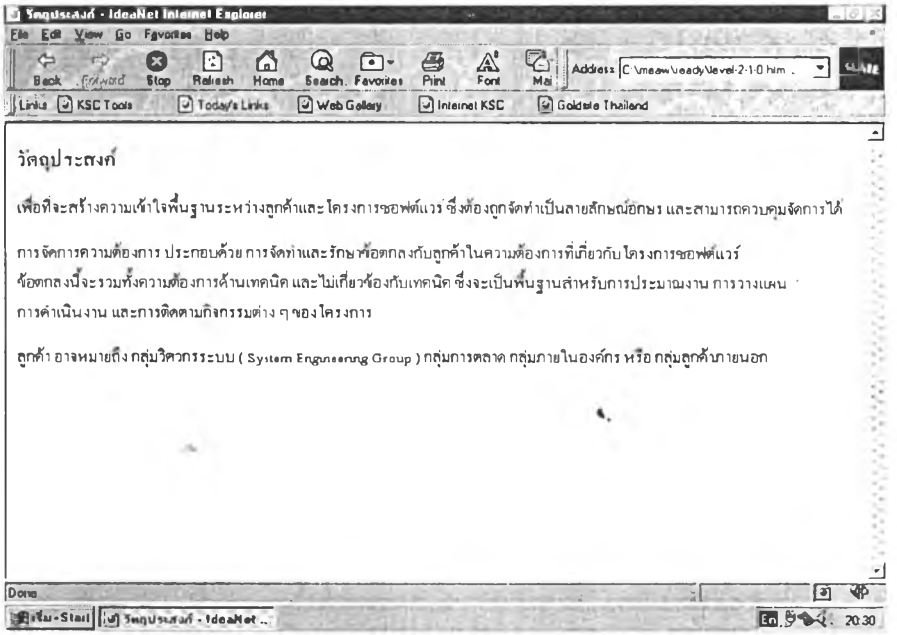




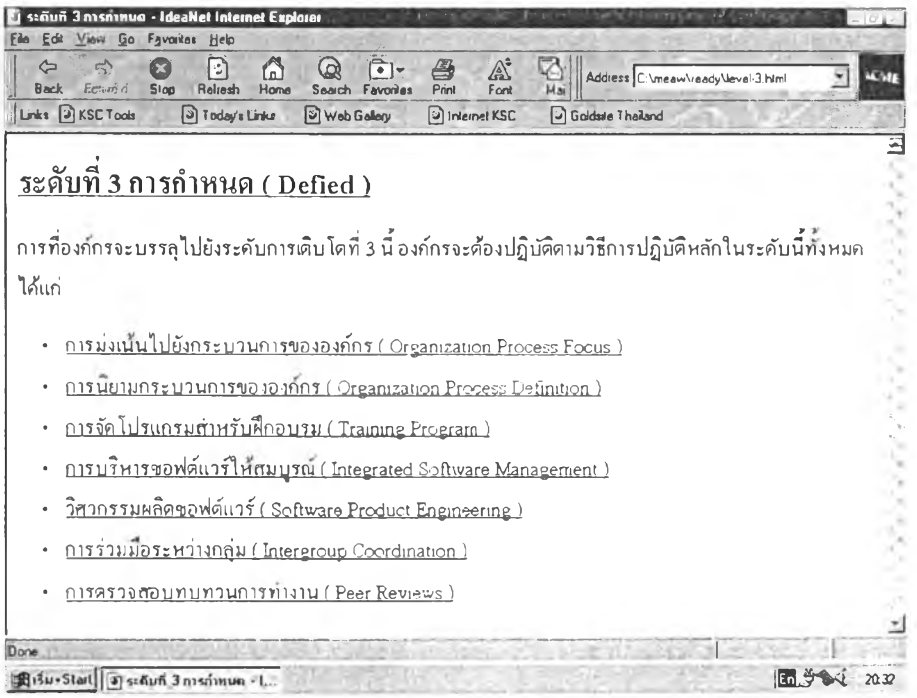
รูปภาพแสดงตัวอย่างรายละเอียดวิธีปฏิบัติหลักของระดับที่ 2



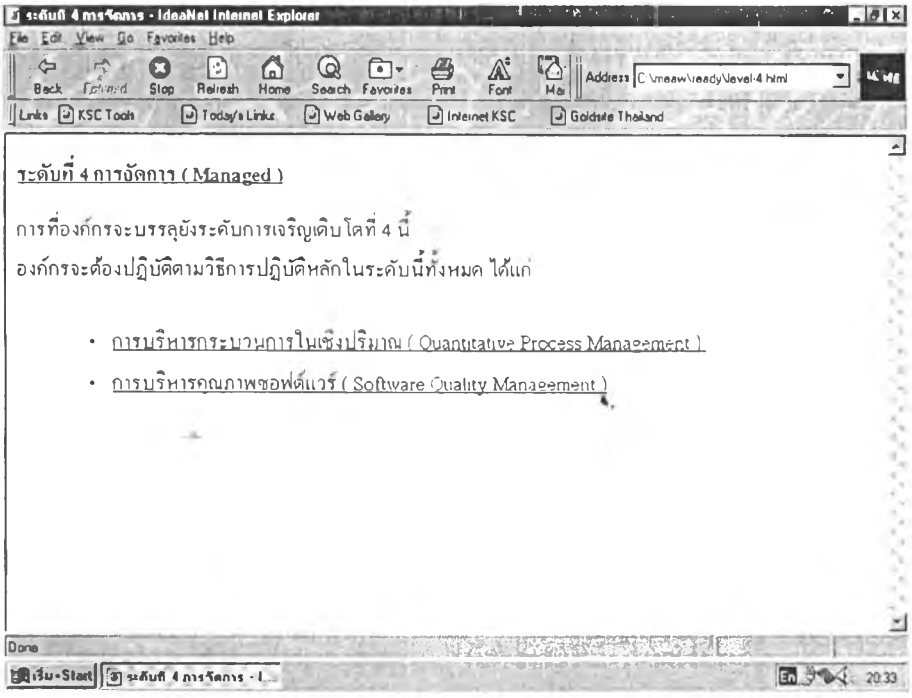
รูปภาพแสดงตัวอย่างองค์ประกอบของแต่ละวิธีปฏิบัติหลัก



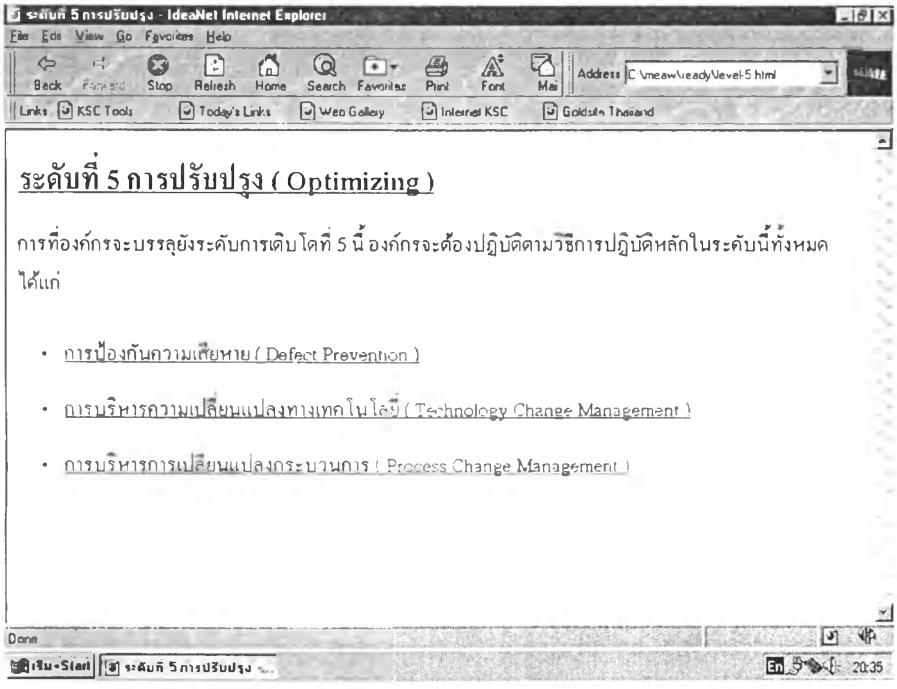
รูปภาพแสดงตัวอย่างรายละเอียดขององค์ประกอบ



รูปภาพแสดงตัวอย่างรายละเอียดวิธีปฏิบัติหลักของระดับที่ 3



รูปภาพแสดงตัวอย่างรายละเอียดวิธีปฏิบัติหลักของระดับที่ 4



รูปภาพแสดงตัวอย่างรายละเอียดวิธีปฏิบัติหลักของระดับที่ 5

ประวัติผู้เขียน

นางสาว จิตรา วัฒนรัตน์ เกิดวันที่ 10 ตุลาคม พ.ศ. 2510 ที่กรุงเทพมหานคร สำเร็จการศึกษาปริญญาตรี บัญชีบัณฑิต คณะพาณิชยศาสตร์และการบัญชี มหาวิทยาลัยธรรมศาสตร์ ในปีการศึกษา 2531 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตร์มหาบัณฑิต ที่จุฬาลงกรณ์มหาวิทยาลัย เมื่อปี พ.ศ. 2537 ปัจจุบันทำงานที่บริษัท เบ็ท เทอร์ฟาร์ม่า จำกัด เป็นบริษัท ในเครือเบทาโกร กรุงเทพมหานคร