สรุป

CE4403 Software Engineering

Software Cost (ต้นทุนของซอฟต์แวร์)

คนส่วน มากจะคิดว่าฮาร์ดแวร์มีค้นทุนสูงว่าซอฟต์แวร์ แต่ที่ตริงแล้วการใช้งานซอฟต์แวร์ที่มีลิขสิทธิ์นั้นมีต้นทุนค่าใช้จ่ายที่ แพงกว่าฮาร์ดแวร์

1. ต้นทุนหรือค่าใช้จ่ายในการปรับเปลี่ยน บำรุงรักษาซอฟต์แวร์จะเท่ากับหรือใกล้เคียงกับต้นทุนของฮาร์ดแวร์
2. ต้นทุนในการบำรุงรักษาซอฟต์แวร์จะมีค่ามากกว่าการพัฒนาซอฟต์แวร์ขึ้นมาใหม่
3. นักวิศวกรรมซอฟต์แวร์จะต้องมีการวิเคราะห์ และพิจารณาในการบำรุงรักษา หรือปรับเปลี่ยนระบบโดยพิจารณาจากต้นทุนเป็นหลัก

ซอฟต์แวร์ คืออะไร ?

1. ซอฟต์แวร์ คือ ชุดคำสั่งที่ใช้ในการควบคุมการทำงานของคอมพิวเตอร์ หรือเอกสารประกอบ
2. ซอฟต์แวร์ คือ โปรแกรมคอมพิวเตอร์ และเอกสารประกอบ

ซอฟต์แวร์ที่ ผลิตขึ้นมี 2 ลักษณะ คือ

1. พัฒนาขึ้นเพื่อใช้งานภาพในองค์กร
2. พัฒนาขึ้นเพื่อจำหน่ายในเชิงพาณิชย์

Software Engineering ?

คือ การนำหลักการทางวิศวกรรมมาใช้กับทุกขั้นตอนของการของการพัฒนาซอฟต์แวร์ ตั้งแต่ก่อนพัฒนาจนได้ซอฟต์แวร์

ความท้าทายทางด้านวิศวกรรมซอฟต์แวร์

1. Legacy การพัฒนาโปรแกรมใหม่ให้สามารถทำงานร่วมกับโปรแกรมเดิมได้
2. Heterogeneity ความไม่เข้ากันของฮาร์ดแวร์และซอฟต์แวร์
3. Delivery ความจำกัดด้านเวลา ความผิดพลาดในเรื่องของเวลา

ความแตกต่างของ Computer Science และ Software Engineering

1. Computer Science จะเน้นทางด้านทฤษฎี หลักการความรู้พื้นฐานที่จะเป็นต่อการนำไปประยุกต์ใช้ในชีวิตประจำวัน

2. Software Engineering จะเน้นการปฏิบัติจริงโดยนำ ทฤษฎีของ Computer Science มาใช้ในทุกขั้นตอนของการผลิตซอฟต์แวร์

ความแตกต่างระหว่าง System Engineering และ Software Engineering

- System Engineering จะเกี่ยวข้องกับทุกๆ ขั้นตอนของการพัฒนาระบบ ทั้งด้าน ฮาร์ดแวร์ ซอฟต์แวร์ และโพรเสซ

- System Engineering จะเกี่ยวข้องกับการกำหนดรายละเอียดของระบบ การออกแบบระบบทางด้านสถาปัตยกรรม การรวมระบบและพัฒนา

CASE (Computer-Aided Software Engineering) คือ ซอฟต์แวร์ที่พัฒนาขึ้นมาเพื่อใช้สร้างซอฟต์แวร์

- Upper-CASE ขั้นตอนการพัฒนาในช่วงแรก เครื่องมือที่ใช้ เช่น MS-Project, Visio, E-Draw
- Lower-CASE ขั้นตอนการเขียนโปรแกรม การทดสอบ เครื่องมือที่ใช้ เช่น MS-Studio, Eclipse, Edit Plus

<u>คุณสมบัติของซอฟต์แวร์ที่ดี</u>

1. Maintainability ต้องมีความสามารถในการบำรุงรักษา จะต้องมีการเปลี่ยนแปลงเพื่อตอบสนองต่อความต้องการของผู้ใช้ที่เปลี่ยนแปลงไป การเปลี่ยนแปลงจะต้องไม่ส่งผลกระทบต่อการทำงานของระบบ
2. Dependability ความสามารถในการพึ่งพา ความน่าเชื่อถือ ต้องผ่านการตรวจสอบในทุกฟังก์ชัน
3. Efficiency ความสามารถในด้านประสิทธิภาพ เช่น ประหยัดทรัพยากรของเครื่อง
4. Usability ความสามารภในการใช้งาน เช่น ความสะดวก ความปลอดภัย สามารถเรียนรู้การใช้งานได้เร็ว

ความรับผิดชอบทางจริยธรรมของนักวิศวกรรมซอฟต์แวร์

1. ความลับ นักวิศวกรรมซอฟต์แวร์ จะต้องรักษาความลับของลูกค้าและนายจ้างแม้จะไม่มีการลงนามเป็นรายลักษณ์อักษร
2. ความสามารถ ไม่อวดความสามารถที่ไม่เป็นจริงและไม่ควรรับงานที่ไม่ถนัด
3. เคารพสิทธิทางปัญญา จะต้องระมัดระวังไม่ละเมิดกฎหมาย
4. ไม่ควรใช้ความถนัดทางด้านเทคนิคการใช้งานคอมพิวเตอร์ ผิดวัตถุประสงค์ เช่น ปล่อยไวรัส

ACM ได้ร่วมมีกับ IEEE กำหนด Code of Ethics ขึ้น ซึ่งก็คือประมวลเบื้องจ้นที่นักวิศวกรรมซอฟต์แวร์ต้องปฏิบัติร่วมกัน มีทั้งหมด 8 ข้อ

1. Public จะต้องทำหน้าที่โดยคำนึงถึงผลประโยชน์ส่วนรวมด้วย
2. Client and Employer ต้องคำนึงถึงความต้องการของลูกค้าและนายจ้าง
3. Product จะต้องผลิตผลงานด้วยมาตรฐานสูงสุดตามหลักวิชาการ
4. Judgment ต้องตัดสินใจได้อย่างอิสละ และเป็นตัวของตัวเอง
5. Management หากนักวิศวกรรมซอฟต์แวร์ เป็นผู้บริหารหรือผู้จัดการจะต้องสนับสนุนและเผยแพร่หลักจริยธรรม
6. Profession ต้องยึดมั่นในคุณธรรม รักษาชื่อเสียงในวิชาชีพของตนเอง
7. Colleagues ต้องมีความเป็นธรรมและสนับสนุนเพื่อนร่วมงาน
8. Self จะต้องพัมนาตนเองอยู่เสมอ

อุปสรรคในการรักษา ประมวลเบื้องต้น 8 ข้อ

1. ขัดแย้งในหลักการของผู้บริหารสูงสุด
2. นายจ้างปฏิบัติอย่างไม่เป็นธรรม
3. เข้าร่วมการสร้างอาวุธทำลายล้างร้ายแรง

The Software Process

Software Process คือ กระบวนการที่พัฒนาซอฟต์แวร์ให้ประสบความสำเร็จ แบ่งออกเป็น 4 ขั้นตอน

1. Specification การกำหนดความต้องการของระบบ โดยสอบถามจาก User
2. Development การพัฒนาระบบ รวมถึงการออกแบบระบบ
3. Validation การทวนสอบ คือขั้นตอนของการทดสอบการทำงานของระบบว่าตรงตามความต้องการของผู้ใช้งานหรือไม่
4. Evolution การปรับปรุงระบบ ในอนาคตเมื่อความต้องการในการใช้งานระบบเปลี่ยนไป จะต้องทำงานปรับปรุงระบบให้ตรงกับความต้องการที่เปลี่ยนไปด้วย

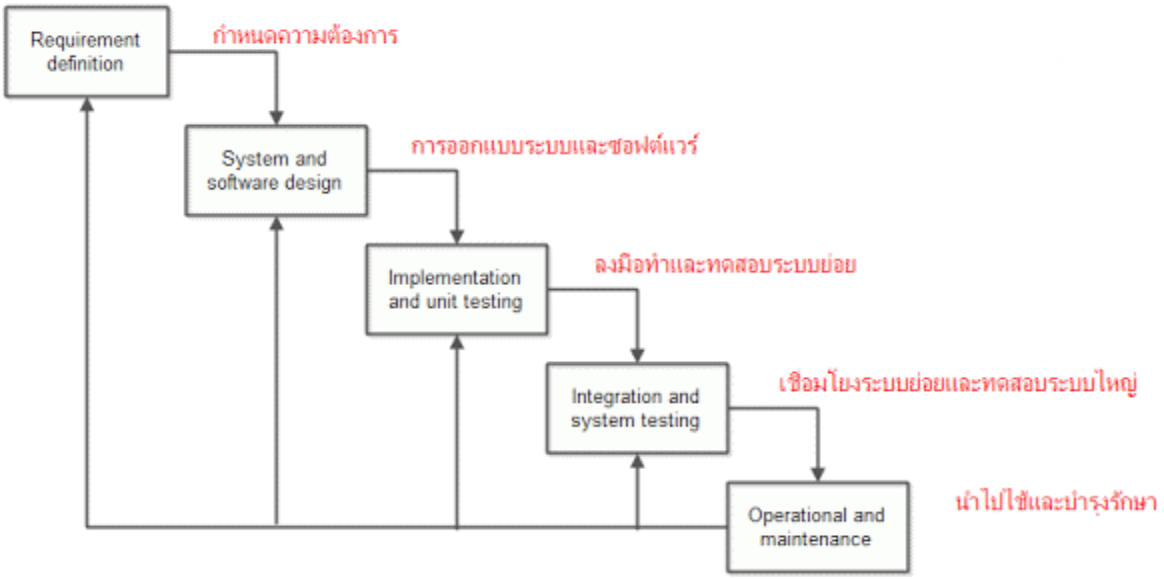
Software Process Model คือ แบบจำลองในการพัฒนาซอฟต์แวร์ มีอยู่ 3 มุมมอง (แบบ)

1. Workflow มุมมองเน้นกิจกรรมของโปรแกรม นิยมเขียนด้วย Flowchart
2. Data-Flow มุมมองเกี่ยวข้องกับข้อมูลเป็นหลัก นิยมเขียนด้วย Data flow Diagram
3. Role/Action มุมมองที่ให้คววามสำคัญว่าใครทำอะไร นิยมเขียนด้วย User case Diagram

แบบจำลองทั่วๆ ไป เช่น

1. Waterfall แบบจำลองน้ำตก
2. Evolutionary development แบบจำลองของการพัฒนาแล้วเพิ่มเติมไปเรื่อยๆ
3. Formal transformation แบบจำลองที่เป็นระเบียบชัดเจนผิดพลาดไม่ได้
4. Integration from reusable มุมมองการนำ Component กลับมาใช้ใหม่

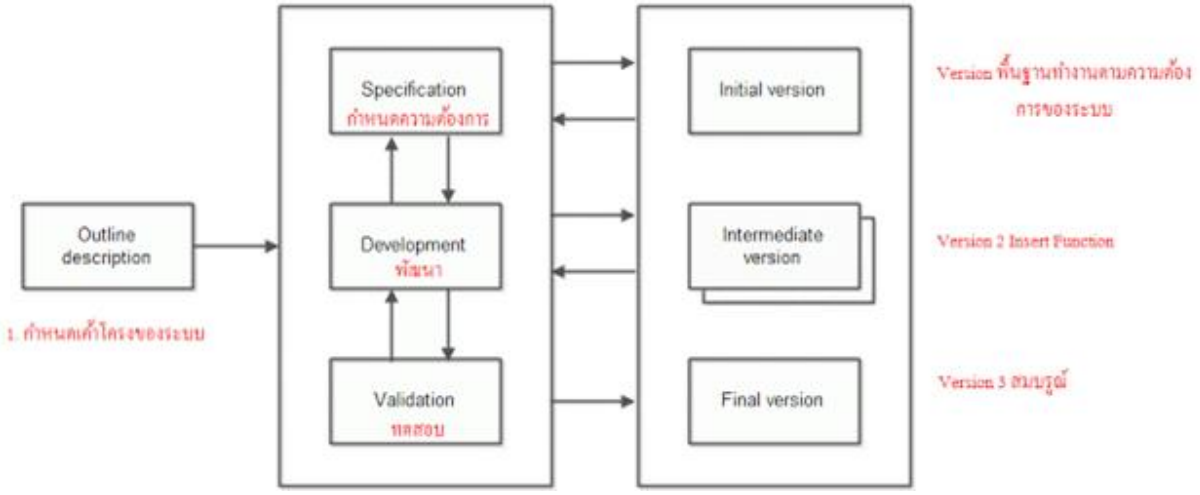
1. Waterfall Model



Waterfall Model

เป็นแบบจำลองที่ประกอบขึ้นด้วยขั้นตอนที่ต่อเนื่องเป็นลำดับ ตั้งแต่ลำดับแรกจนถึงลำดับสุดท้ายว่ามีอะไรบ้าง ไหลจากที่สูงลงสู่ที่ต่ำ สามารถย้อนกลับไปแก้ไขในขั้นตอนก่อนหน้าได้ **(ข้อเสีย(**แต่จะมีความยุ่งยากในการในการปรับเปลี่ยน เช่น ต้องการในขั้นตอนที่ 3 ก็ต้องย้อนกลับไปขั้นตอนที่ 1, 2 ด้วย

2. Evolutionary development



Evolutionary Model

การพัฒนาแบบนี้ทำให้เกิดการปรับปรุง ปรับเปลี่ยนความต้องการให้อยู่ในรูปแบบของ Version มี 4 ขั้นตอน

1. Requirement วิเคราะห์ความต้องการ
2. System Design ออกแบบระบบ
3. Coding and Testing ขั้นตอนการเขียนและทดสอบระบบ
4. Assignment การประเมินผล

ปัญหา

1. ขาดความชัดเจนในการทำงาน
2. ระบบที่สร้างขึ้นมาไม่ค่อยเป็นระบบเท่าไร
3. ผู้ใช้ระบบ ต้องมีความชำนาญสูง

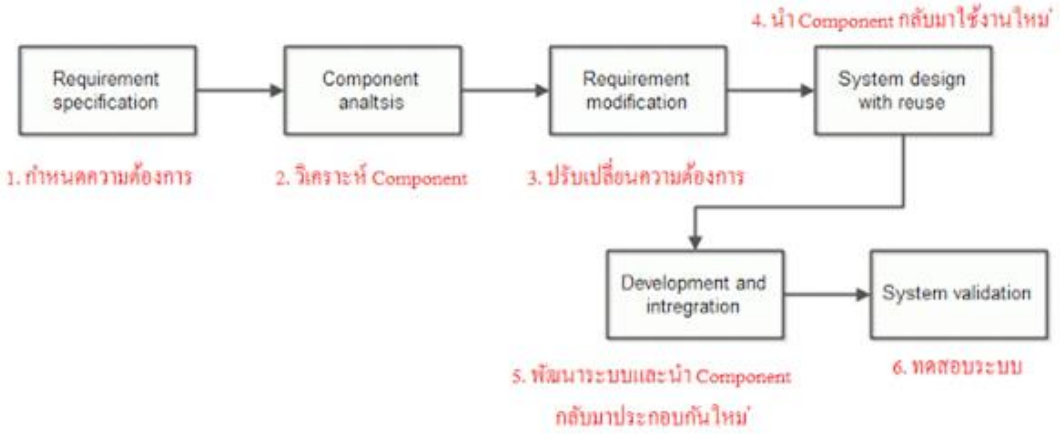
ข้อเสีย

1. อาจจะรบกวนเวลาทำงานของลูกค้า ทำให้เกิดความลำคาญใจ
2. บอกความคืบหน้าในการทำงานลำบาก

การนำแบบจำลองไปใช้

1. ใช้กับระบบเล็กหรือขนาดกลาง
2. นำไปใช้ในส่วนของระบบงานขนาดใหญ่
3. นำไปใช้ในโครงงานที่มีอายุการดำเนินงานสั้นๆ

3. CBSE (Component-Based Software Engineering)



Reuse-oriented development

ข้อดี

1. รวดเร็วขึ้น
2. ประหยัดต้นทุน
3. ระบบมีความเชื่อมั่นเพิ่มมากขึ้น
4. ความเสี่ยงที่

ข้อเสีย

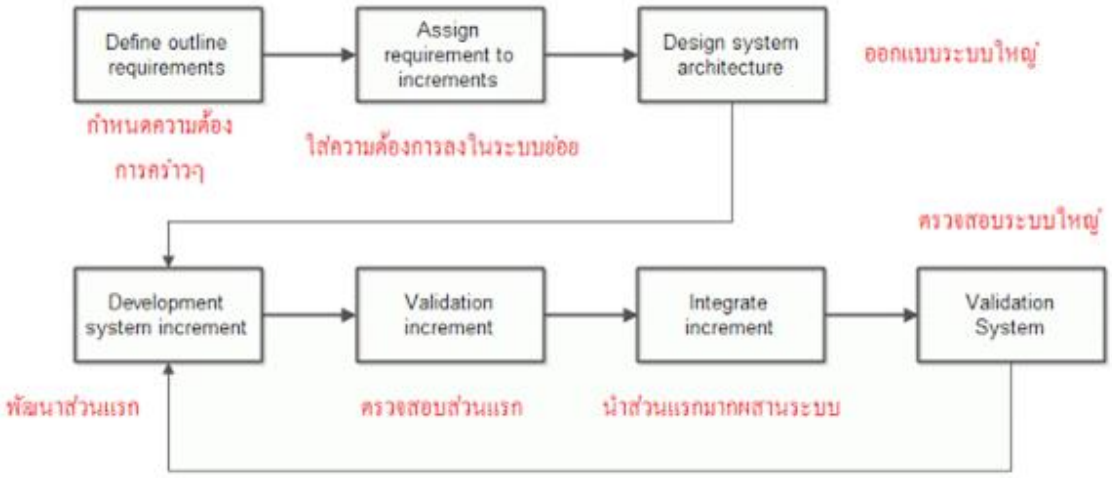
1. หา Component ที่ตรงตามความต้องการได้ยาก
2. ยากในการปรับปรุง Component ให้ตรงตามความต้องการ

4. Process Iteration

การพัฒนาแบบวนซ้ำๆ)loop) มี 2 วิธี

1. Increment delivery การส่งระบบเพิ่มเติมไปเรื่อยๆ ทีละชุดจนเสร็จ
2. Spiral delivery พัฒนาแบบก้หอย

4.1 Increment delivery

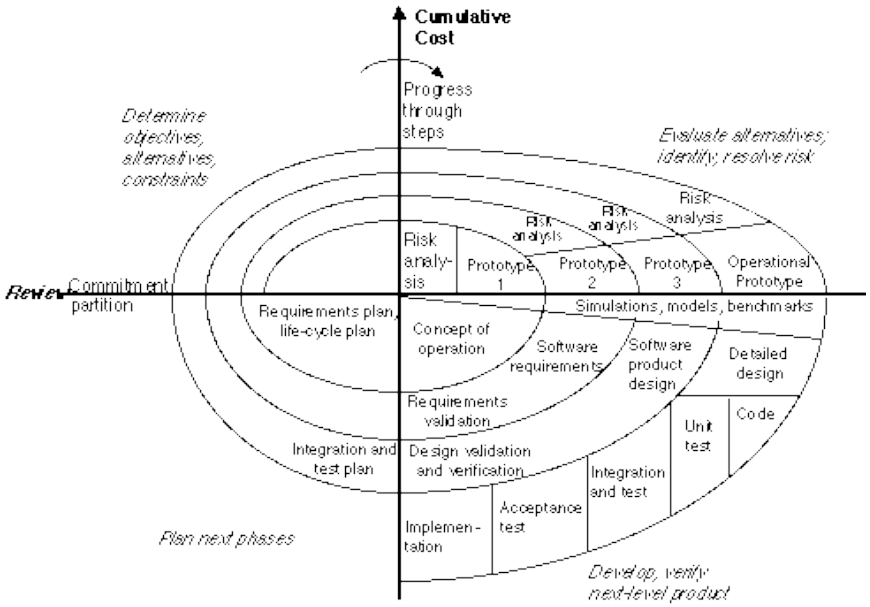


Increment delivery

ประโยชน์

1. ลูกค้าได้รับประโยชน์ในการส่งมอบเร็วขึ้น คือการที่ลูกค้าสามารถนำระบบบางส่วนไปใช้งานได้
2. ลดความเสี่ยงที่โครงการจะล่ม
3. ฟังก์ชันที่มีความเสี่ยงจะถูกทดสอบมากที่สุด ทำให้เกิดความผิดพลาดได้น้อย

4.2 Spiral delivery



Spiral Development

เป็นการพัฒนาหนุมเป็นรอบๆ เป็นเกียววนคล้ายก้นหอย แบ่งออกเป็น 4 ส่วน

1. กำหนดวัตถุประสงค์และความสำคัญ
2. ประเมินความเสี่ยง
3. พัฒนา และตรวจสอบความต้องการ
4. วางแผน

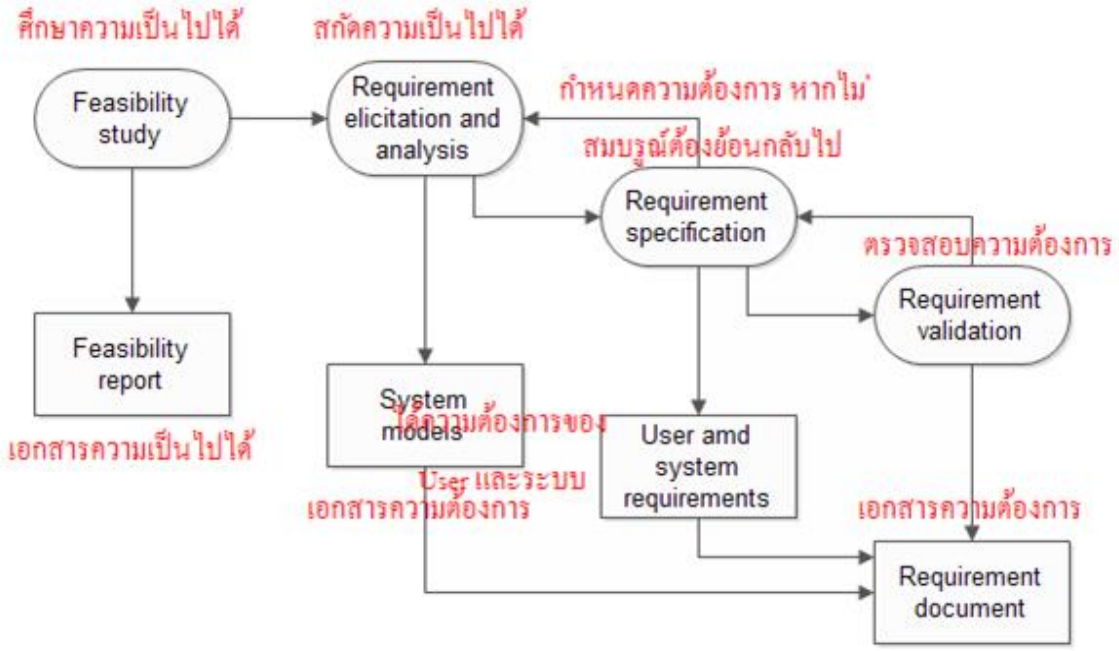
Process Activity

Process Activity คือ กิจกรรมขั้นตอนการจัดทำ Software

ขั้นตอนที่ 1 Software Specification กระบวนกำหนดความต้องการ

คือ กระบวนการวิศวกรรมความต้องการ ซึ่งก็คือการกำหนดความเป็นไปได้ให้กับระบบ

Requirements engineering process



Requirements engineering process

1. Feasibility Study ศึกษาความเป็นไปได้ เช่น งบประมาณ เวลา Software Hardware
2. Requirement elicitation and analysis สกัดความต้องการเพื่อให้รู้ความต้องการของระบบ

3. Requirement specification กำหนดความต้องการ

4. Requirement validation ขั้นการตรวจสอบว่าตรงจามความต้องการหรือไม่

ขั้นตอนที่ 2 Software design and implementation

Design เป็นขั้นตอนการออกแบบและเขียนโปรแกรม

Implementation การนำไปใช้งาน

Debugging process แบบจำลองการแก้ไขโปรแกรม

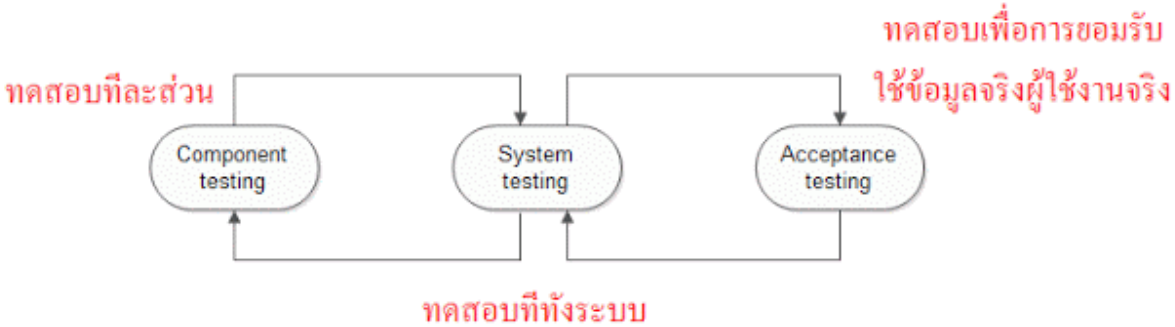
1. หาตำแหน่งที่ผิดพลาดของโปรแกรม
2. ออกแบบว่าจะแก้ไข ข้อผิดพลาดที่เกิดขึ้นอย่างไร
3. ซ่อมบำรุง ทำการแก้ไข
4. ทดสอบการแก้ไข ว่าถูกต้องและผ่านหรือไม่

ขั้นตอนที่ 3 Software validation การทดสอบซอฟต์แวร์

Verification ทดสอบว่าซอฟต์แวร์ที่พัฒนาขึ้นตรงกัยความต้องการหรือไม่ โปรแกรมเมอร์ทดสอบ

Validation ทดสอบว่าตรงกับความต้องการของลูกค้าหรือไม่

Testing Process



Testing Process

1. Component or unit testing การทดสอบทีละส่วน ทีละระบบ
2. System testing การทดสอบทั้งระบบรวมกัน
3. Acceptance testing การทดสอบเพื่อการยอมรับ จากผู้ใช้งานจริง และใช้ข้อมูลจริง

ขั้นตอนที่ 4 Software evolution ขั้นตอนในการเปลี่ยนแปลงโปรแกรมตามความต้องการของ User *** (MA)**

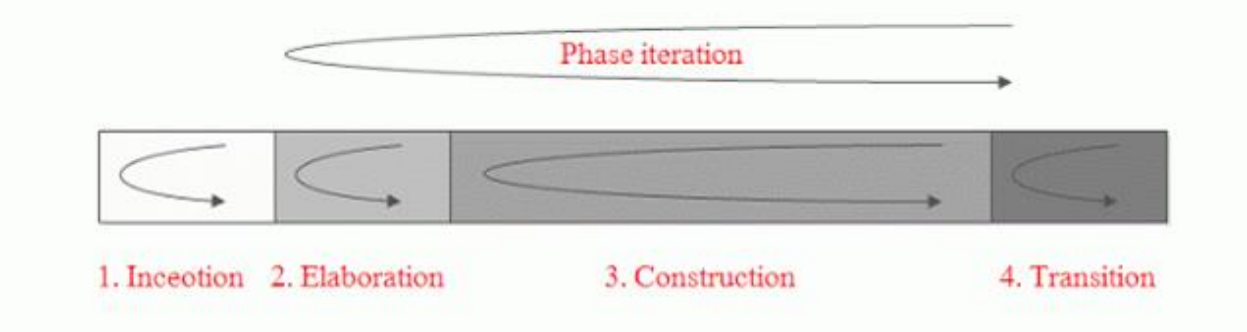
The Rational Unified Process คือ คอมพิวเตอร์ที่ช่วยในการสร้างซอฟต์แวร์ RUP เขียนอยู่บน UML

การเขียนโปรแกรมแบบ UML มี 3 มุมมอง

1. แสดงขั้นตอนการทำงานให้เห็นเป็นขั้นๆ
2. แสดงให้เห็นกิจกรรมว่ามีอะไรบ้าง
3. แสดงให้เห็นถึงการปฏิบัติงานที่ดี

RUP phase model แบ่งออกเป็น 4 ขั้นตอน ***แต่ละขั้นมีการทำซ้ำบ่อยครั้ง

*****สถาปัตยกรรม มีจุดเด่น คือ การค้นหาความเสี่ยงและวิเคราะความเสี่ยง**



RUP Phase Model

1. Inception ระยะเริ่มต้นของการดำเนินงาน กำหนดขอบเขตหน้าที่หลักของระบบ
2. Elaboration ทำความเข้าใจระบบ
3. Construction ออกแบบ เขียน ทดสอบ โดยแบ่งออกเป็นส่วนๆ โดยให้ Programmer ช่วยกันเขียน แล้วค่อยนำมารวมกัน หลังจากนั้นจะได้ ซอฟต์แวร์ และ เอกสารของซอฟต์แวร์
4. Transition ขั้นตอนการส่งมอบระบบ

ข้อปฏิบัติการใช้ RUP

1. พัฒนาโปรแกรมแบบซ้ำๆ หากไม่สมบูรณ์ให้กลับไปทำใหม่
2. บริหารความต้องการให้ดีว่า ความต้องการไหนมีความสำคัญกว่า
3. ควรใช้งาน Component ที่มีอยู่แล้ว

4. ยึด Model RUP มาช่วยในการออกแบบ
5. ตรวจสอบคุณภาพของซอฟต์แวร์ให้ดีอยู่เสมอ
6. ควบคุมการเปลี่ยนแปลงของซอฟต์แวร์ให้น้อยที่สุด

วิศวกรรมความต้องการ

กระบวนการวิศวกรรมความต้องการ แบ่งออกเป็น 4 ขั้นตอน คือ

1. ศึกษาความเป็นไปได้

เป็นขั้นตอนการศึกษาเพื่อตัดสินใจว่ามีความคุ้มค่าในการลงทุนหรือไม่

- ตรงกับวัตถุประสงค์ขององค์กรหรือไม่
- เทคโนโลยีในปัจจุบันสามารถนำมาพัฒนาระบบได้หรือไม่ สามารถทำได้ไหม
- ระบบใหม่สามารถทำงานร่วมกับระบบเก่าได้หรือไม่

ในการศึกษาความเป็นไปได้จะทำให้

1. ถ้าไม่พัฒนาระบบจะเกิดอะไรขึ้น
2. ปัญหาที่พบมีอะไรบ้าง
3. ระบบใหม่สามารถช่วยในการแก้ไขปัญหาได้อย่างไร
4. ต้องใช้เทคโนโลยีอะไรและใช้ความสามารถพิเศษอะไรบ้าง

2. ค้นหาและวิเคราะห์ความต้องการ

ในขั้นตอนนี้เพื่อนำความต้องการที่ได้ไปสร้างเป็นแบบจำรอง ในการค้นหาและวิเคราะห์ความต้องการ เราต้องส่งเจ้าหน้าที่เข้าไปทำงานร่วมกับลูกค้าเพื่อค้นหาว่าในการทำงานของ User มีความต้องการอย่างไร มีขั้นตอนอย่างไร นอกจากนี้เรายังต้องสอบถามความต้องการจากผู้เกี่ยวข้องคนอื่นๆด้วย

3. กำหนดความต้องการ

นำความต้องการที่ได้มาทำการวิเคราะห์แล้วทำการกำหนดเป็นความต้องการของ User และ System ทำให้ในขั้นตอนนี้เราได้ความต้องการของ user และ system

4. ตรวจสอบความต้องการ

ตรวจสอบความต้องการที่ได้ เพื่อลดความผิดพลาดหรือปัญหาที่จะเกิดขึ้น ความต้องการที่ทับซ้อนกัน ความต้องการ ที่มากเกินต้นทุนที่กำหนดไว้ เมื่อสิ้นสุดการตรวจสอบเราจะได้เอกสารความต้องการที่ถูกต้อง

Functional Requirements (มักจะขึ้นต้นด้วย “จะ”) เช่น

1. ผู้ใช้จะสามารถค้นหาบทความทั้งหมดหรือสามารถเลือกค้นหาได้

2. ระบบจะให้ผู้อ่านสามารถอ่านบทความที่เก็บเอาไว้ในฐานข้อมูลได้

<u>Non-functional requirement</u> แบ่งออกได้เป็น 3 ด้าน

1. Product requirement ความต้องการทางด้านผลิตภัณฑ์ เช่น ประสิทธิภาพของผลิตภัณฑ์ ความน่าเชื่อถือ
2. Organizational requirement ความต้องการทางด้านองค์กร เช่น นโยบายขององค์กร ข้อกำหนดขององค์กร ระเบียบปฏิบัติของลูกค้า
3. External requirement ความต้องการภายนอก เช่น กฎหมาย นโยบายขององค์กร ของสังคม เกี่ยวกับสินค้า หลักทางด้านจริยธรรม

รูปแบบในการตรวจสอบ

1. ตรวจสอบความเที่ยงตรงจากตัวแทนผู้ใช้
2. ความสมบูรณ์ของความต้องการ
3. ความเป็นไปได้ของความต้องการทางด้าน เทคโนโลยี
4. สามารถพิสูจน์ได้ เช่น ทำให้ลูกค้าเห็นได้ Prototype

ปัญหาในการวิเคราะห์ความต้องการ

1. Stakeholder ไม่มีความรู้ความเข้าใจ
2. Stakeholder อธิบายความต้องการด้วยศัพท์ทางเทคนิคที่เขาเข้าใจ และเราไม่เข้าใจ อาจทำให้เกิดการเข้าใจผิดได้
3. Stakeholder มีความต้องการที่แตกต่างกัน เราต้องแบ่งกลุ่มความต้องการเพื่อไม่ให้ขัดแย้งกัน
4. การเมืองและความขัดแย้งในองค์กร ส่งผลต่อระบบ อาจทำให้เกิดความผิดพลาดล้มเหลวในการพัฒนาได้
5. มีการเปลี่ยนแปลงความต้องการระหว่างการพัฒนา เช่น SK คนใหม่เข้ามา ธุรกิจเปลี่ยน เปลี่ยนเทคโนโลยี

ทำไมต้องมีการปรับเปลี่ยนความต้องการ

1. ในระบบมีผู้ใช้หลายกลุ่ม เราต้องจัดลำดับความสำคัญของความต้องการ เพื่อไม่ให้เกิดความขัดแย้งใน SK
2. งบประมาณไม่เพียงพอ
3. สภาพแวดล้อม และ เทคโนโลยี เมื่อเกิดการเปลี่ยนแปลง ก็จะต้องมีการเปลี่ยนแปลงความต้องการให้มีความสอดคล้อง และ เหมาะสม

System Models

การสร้างต้นแบบจำลองของ Software

เป็นขั้นตอนการนำความต้องการที่เก็บรวบรวมได้ มาสร้างเป็นแบบจำลอง (Model)

System modeling การสร้างตัวต้นแบบ

การสร้างตัวต้นแบบมีวัตถุประสงค์

1. เพื่อให้เข้าใจว่าหน้าที่หลักของระบบทำงานอย่างไร ผู้ที่ควรจะต้องเข้าใจถึงกระบวนการคือ SA เจ้าของระบบ User Programmer
2. เพื่อให้สามารถอธิบายได้ว่าระบบเก่าและระบบใหม่มีความแตกต่างกันอย่างไร โดยแบ่งออกเป็น
3. มุมมองได้แก่
    1. มุมมองภาพรวมของระบบ
    2. มุมมองพฤติกรรมของระบบ
    3. มุมมองโครงสร้างของระบบ

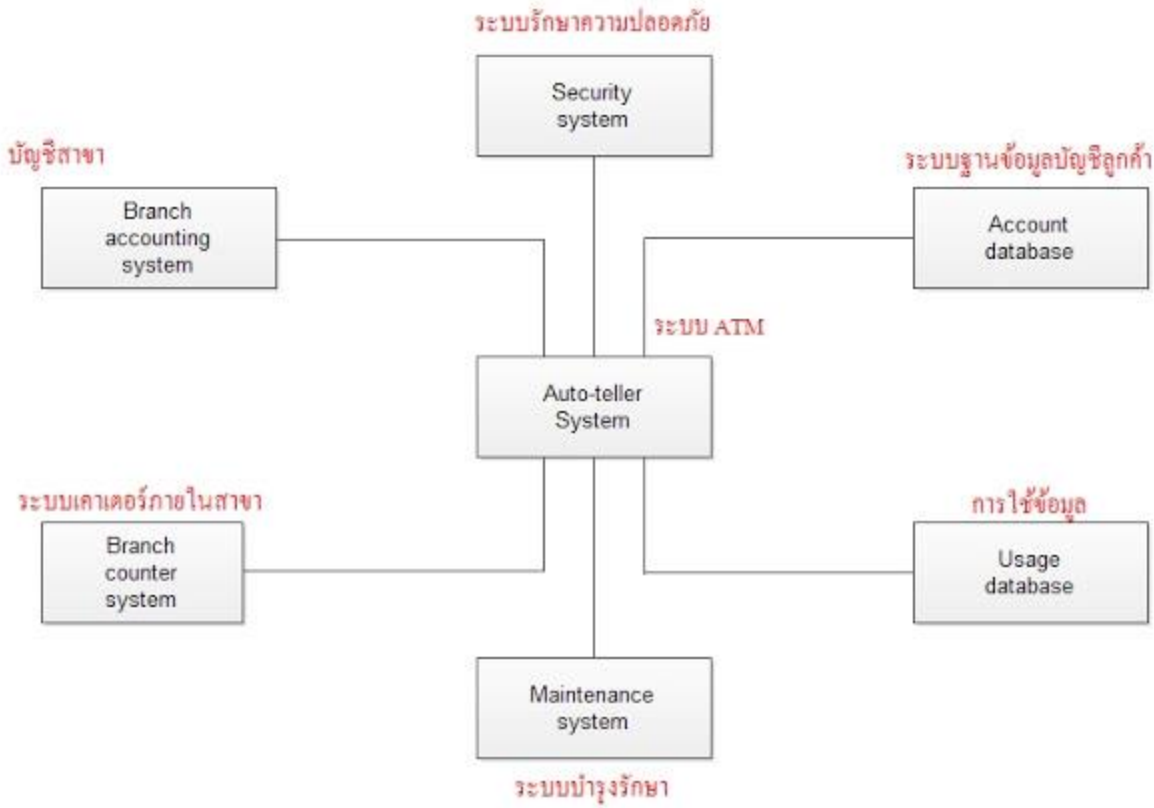
Model Types ประเภทของตัวต้นแบบ

1. Date processing model แบบจำลองแสดงการประมวลผลข้อมูล จะอธิบายพฤติกรรมของระบบ
2. Composition model แบบจำลองอธิบายองค์ประกอบของ Entities
3. Architectural model อธิบายสถาปัตยกรรมของระบบ โดยจะแสดงให้เห็นถึงระบบย่อยภายในด้วย
4. Classification model อธิบายการจัดแบ่งประเภทของ Entities
5. Stimulus/response model อธิบายตัวกระตุ่นหรือสิ่งกระตุ้น

Context Model

แสดงขอบเขตของระบบ ว่ามีการเชื่อมโยงกับระบบอื่นๆ อย่างไร context models จะเน้นขอบเขตของระบบ ไม่เน้นรายระเอียดของระบบ

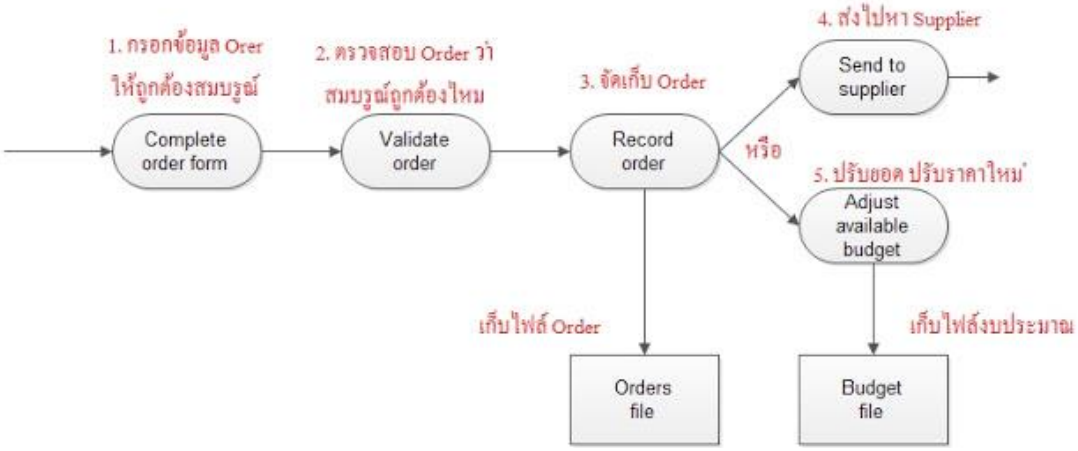
ตัวอย่าง Context model ของระบบ ATM



Behavioral models แบบจำลองพฤติกรรม แบ่งเป็น 2 รูปแบบ

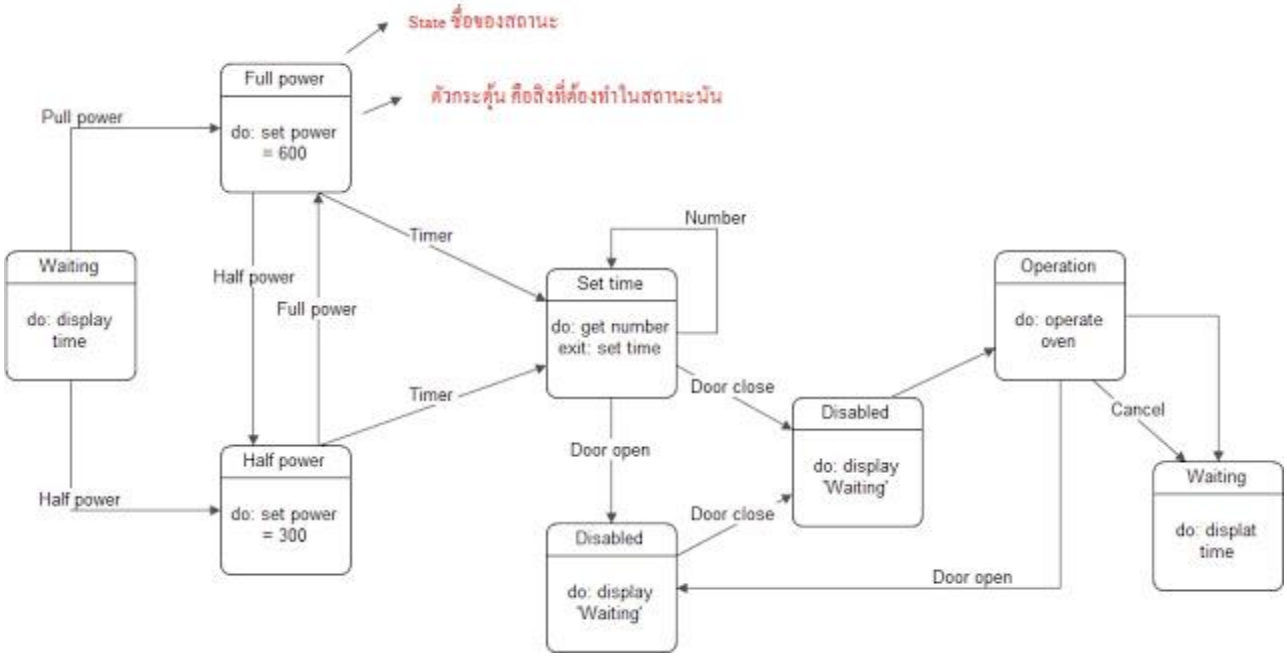
1. Data processing เน้น DFD เน้นแสดงกระบวนการทำงานของระบบ

   <u>ตัวอย่าง</u> Order Processing DFD มีทั้งหมด 5 ขั้นตอน สามารถอธิบายได้ดังรูป



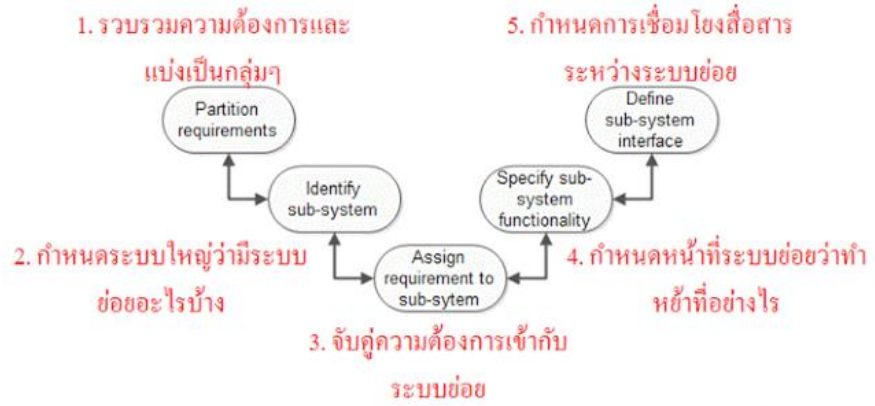
2. State machine models เน้นสถานการณ์เปลี่ยนแปลงสถานะของอุปกรณ์ทางด้านวิศวกรรมที่สร้างขึ้น

<u>ตัวอย่าง</u> Microwave oven model มีทั้งหมด 8 State สามารถอธิบายได้ดังรูป



System design

ขั้นตอนการออกแบบระบบ มี 5 ขั้นตอน



System Design Process

1. รวบรวมความต้องการมาแบ่งกลุ่มเป็นส่วนๆ
2. กำหนดว่าระบบใหญ่ควรมีระบบย่อยอะไรบ้าง
3. จับคู่ความต้องการเข้ากับระบบย่อย
4. กำหนดหน้าที่ของระบบย่อย
5. กำหนดส่วนติดต่อระหว่างระบบย่อยว่ามีการเชื่อมต่อ สื่อสารกันอย่างไร

ปัญหาของการออกแบบระบบ

1. ไม่รู้ว่าใครเป็นผู้รับผิดชอบ
2. ปัญหาที่ยาก จะถูกผลักดันให้เป็นหน้าที่ของซอฟต์แวร์ ทำให้ซอฟต์แวร์มีขนาดใหญ่และต้นทุนที่สูง
3. ฮาร์ดแวร์ ซอฟต์แวร์และ Platform ไม่เหมาะกัน เช่น Windows, Mac, Linux

3. Sub system development ขั้นตอนการพัฒนาระบบ

- การพัฒนามักจะพัฒนาแบบคู่ขนานระหว่างฮาร์ดแวร์ ซอฟต์แวร์ และ Communication
- การพัฒนามักใช้ซอฟต์แวร์สำเร็จรูปมาใช้
- ระหว่างทีมพัฒนาในองค์กรอาจเกิดอุปสรรคได้

4. System integration ขั้นตอนการนำระบบย่อยมารวมกัน

เริ่มจากการนำ Sub system ที่สำคัญที่สุดก่อน นำมาทำงานร่วมกัน เมื่อทำงานได้ก็นำ Sub system ต่อไป ทำไปเลื่อยๆ การรวมระบบในรูปแบบนี้จะทำให้ระบบล้ม หรือ เกิดความผิดพลาดน้อยลง>

5. System installation ขั้นตอนการติดตั้งและนำไปใช้

ปัญหาการติดตั้ง

1. สภาพแวดล้อมไม่เหมาะสม ไม่อำนวย
2. คนต่อต้านการนำระบบใหม่ไปใช้งาน เช่น กลัวหมดความสำคัญ
3. ระบบใหม่ที่พัฒนาขึ้นต้องทำงานคู่ขนานกับระบบเก่า ระบบอาจเข้ากันได้ลำบาก
4. พื้นที่การติดตั้งไม่เพียงพอ
5. การวางแผนอบรมการใช้งานระบบ

ปัญหาการนำไปใช้งาน

1. ความต้องการที่ไม่คาดคิดจะโผล่ออกมา เมื่อพัฒนาเสร็จสิ้นแล้ว
2. ผู้ใช้งาน ไม่ใช้ระบบที่เราออกแบบ กรอกข้อมูลผิดตำแหน่ง
3. เวลาทำงานร่วมกับระบบอื่นอาจมีปัญหาได้
   - ปัญหาทางกายภาพ

- ปัญหาการแปลงข้อมูล

6. System evolution ขั้นตอนการปรับปรุงระบบ

เมื่อใช้งานไปแล้วต้องมีการพัฒนาโปรแกรมรุ่นใหม่ออกมา เพื่อให้มีการพัฒนาตามฮาร์ดแวร์ ซอฟต์แวร์ และ Technology ใหม่ๆ เมื่อปรับปรุงจะเกิดค่าใช้จ่าย

7. System decommissioning ขั้นตอนการปลดละวาง เลิกใช้งาน

ปัญหาการเชื่อมโยงระหว่าง Database ตัวใหม่กับตัวเก่า

User Interface Design

การออกแบบหน้าจอการทำงานของระบบหรือที่เรียกว่าส่วนติดต่อระหว่างผู้ใช้งานกับระบบนั้น นักวิศวกรรมซอฟต์แวร์จะต้องให้ความสำคัญในการพูดคุยกับผู้ใช้มากที่สุด เพื่อเก็บรวบรวมความต้องการ

The Design Process (กระบวนการออกแบบส่วนติดต่อกับผู้ใช้ User Interface)

มีทั้งหมด 6 ขั้นตอน ดังนี้

1. ขั้นตอนการเก็บรวบรวมความและวิเคราะห์ความต้องการจาก User ว่า User มีกิจกรรมใดที่ต้องทำบ้าง มีความต้องการอย่างไรบ้าง
2. สร้างตัวต้นแบบใน Prototype Paper แล้วนำตัวต้นแบบที่สร้างขึ้นไปตรวจสอบกับ User อีกครั้ง ว่าตรงกับความต้องการหรือไม่ (ในขั้นตอนนี้ Output ที่ได้ คือได้การออกแยยตัวต้นแบบ Design prototype) กรณีที่ไม่ตรงตามความต้องการหรือผู้ใช้มีความต้องการเพิ่มเติมจะต้องมีการวนกลับไปแก้ไขตัวตนแบบ
3. นำตัวต้นแบบที่ได้จากขั้นตอนที่ 2 ไปตรวจสอบกับ end user ว่าตรงตามความต้องการหรือไม่ กรณีที่ไม่ตรงตามความต้องการหรือผู้ใช้มีความต้องการเพิ่มเติมจะต้องมีการวนกลับไปแก้ไขตัวตนแบบ
4. เมื่อตัวตนแบบที่สร้างขึ้นบนกระดาษผ่านการตรวจสอบความต้องการจาก User และ End User แล้ว นำมาสร้างตัวตนแบบจริงๆ บนคอมพิวเตอร์ที่สามารถกดคลิกปุ่ม หรือกรอกข้อมูลต่างๆ ได้จริงๆ
5. นำตัวต้นแบบทีสร้างขึ้นไปประเมินกับ end user ในกรณีที่ Prototype ที่สร้างขึ้นไม่ตรงกับความต้องการหรือมีความต้องการบางอย่างที่ end user จะต้องมีการวนกลับไปแก้ไขตัวตนแบบจนกว่า end user พอใจ ในขั้นตอนนี้ผลลัพธ์ที่ได้คือ Prototype จริงๆ ที่สามารถใช้งานได้จริงๆ ในสถานการณ์จริง กับลูกค้าจริง
6. นำ Prototype มาสร้างเป็น User Interface

Usability attributes หลักในการประเมิน User interface

1. Learnability สามารถเรียนรู้ได้ง่าย เช่น User ที่ไม่เคยรู้จักการใช้งานระบบมาก่อนสามารถเรียนรู้การใช้งานได้ง่าย

2. Speed of operation ความรวดเร็วในการทำงานของระบบ เช่น ระบบที่สร้างขึ้นมีความรวดเร็วมีความรวดเร็วในการประมวลผล ในการตอบสนองต่อการใช้งานของผู้ใช้
3. Robustness ความทนทานในการใช้งาน เช่นความทนทานของระบบเมื่อผู้ใช้งานกรอกข้อมูลผิดพลาด ระบบจะมีการแจ้งเตือนอย่างไร
4. Recoverability ความสามารถในการกู้คืน เช่น เมื่อระบบเกิดความล้มเหลวในการทำงาน ระบบสามารถกู้คืนการทำงานที่เป็นสถานะปกติโยใช้เวลาเท่าไร
5. Adaptability ความสามารถในการใช้งาน คือ ในเรื่องของประสิทธิภาพการใช้งานระบบ

Simple evaluation techniques (เทคนิคในการประเมิน User interface)

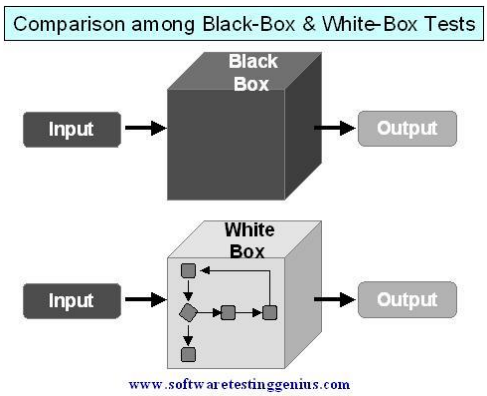
1. สร้างแบบสอบถามให้ User กรอกข้อมูลแล้วนำมาประมวลผล
2. ใช้การบันทึก Video บันทึกภาพขณะที่ผู้ใช้งาน ทดลองใช้งานระบบ แล้วดูปฏิกิริยาตอบสนองตอนใช้งาน
3. เขียนโปรแกรมให้ทำหน้าที่เก็บข้อมูลว่า User ได้ทำอะไรกับระบบบ้าง เพื่อดูการกระทำของ User เพื่อใช้ในการประเมิน
4. เพิ่มปุ่มให้ User กรอกข้อมูลในหน้านั้นเลยว่า User มีความรู้สึกอย่างไรกับระบบ

Software Engineering Test

เป็นกระบวนการที่เกิดขึ้นหลังจากการพัฒนา Software เสร็จเรียบร้อยแล้ว เพื่อตรวจสอบความผิดพลาดในส่วนต่างๆ ที่เกิดขึ้น แล้วทำการแก้ไข นอกจากนี้เพื่อประเมินคุณภาพและปรับปรุงคุณภาพของ Software

คำศัพท์ในการแก้ไขข้อผิดพลาด

1. Error การกระทำผิด คือค่าจริงที่ได้จากการทำงานที่ไม่ถูกต้อง
2. Fault ความผิดพลาดหรือข้อบกพร่อง คือกระบวนการทำงานของระบบประมวลผลที่ผิดปกติ
3. Failure ล้มเหลว คือ SW ไม่สามารถทำงานหรือรันโปรแกรมต่อไปได้ รวมถึงไม่สามารถแสดงข้อมูลแจ้งเตือนข้อผิดพลาดที่เกิดขึ้นได้



การทดสอบแบบกล่องดำ Black box testing

เป็นการทดสอบการทำงานของ SW ในเชิงพฤติกรรม 8nv การทดสอบผลของการทำงานของ SW ในแต่ละหน้าที่ ตามข้อกำหนดความต้องการ ทดสอบโดยมองให้เห็นกระบวนการทำงาน สนใจเฉพาะผลลัพธ์ที่ได้เท่านั้น

การทดสอบแบบกล่องขาว White box testing

เป็นการทดสอบการทำงานของระบบโดยมองลึกลงไปถึง Code คำสั่งต่างๆ ที่อยู่ภายในระบบ

- ทดสอบทุกเส้นทางในกระบวนการ จะต้องสามารถทำงานได้อย่างถูกต้อง
- ทดสอบการทำงานวนซ้ำ Loop
- ทดสอบกระบวนการตัดสินใจในทุกตรรกะ
- ทดสอบโครงสร้างข้อมูลภายในระบบ

Integration testing มี 2 วิธี

1. Top Down Approach ทดสอบการทำงานของระบบแบบบนลงล่าง เป็นการทำสอบการทดงานในฟังก์ชั่นใหญ่ด้านบนก่อนแล้วค่อยๆ ทดสอบฟังก์ชั่นย่อยต่างๆ ที่อยู่ภายใน
2. Button up Approach ทดสอบการทำงานของระบบแบบล่างขึ้นบน เป็นการทดสอบการทำงานในฟังก์ชั่นการทำงานย่อยภายในฟังก์ชั่นใหญ่ก่อนแล้วค่อยนำฟังก์ชั่นย่อยมาทดสอบรวมกับฟังก์ชั่นใหญ่ด้านบน

Project Manager

สิ่งที่ Project Manager ต้องทำมี 6 อย่าง

1. เขียน Proposal ว่าระบบที่จะพัฒนาจะมีคุณสมบัติอย่างไร ทำประโยชน์อะไรให้องค์กรได้บ้าง เอามาใช้แก้ไขปัญหาอะไรและศึกษาความคุ่มค่าในการพัฒนา
2. เขียนแผนในการพัฒนา
3. ประเมินต้นทุนในการพัฒนา

4. ติดตามดูการพัฒนา ว่าตรงตามที่กำหนดไว้หรือไม่ ตรงกับความต้องการไหน

5. คัดเลือกบุคลากร เข้าทีมพัฒนา จัดสรรตำแหน่งและประเมินความสามารถ

6. นำเสนอรายงานในทุกๆ ช่วงของ Milestone

2 เรื่องสำคัญที่ Project manager ต้องดูแล

1. Project staffing เรื่องของการบริหารจัดการบุคลากร

    1.1 จัดหาบุคลากรที่เหมาะสม

    1.2 มอบหมายงานที่เหมาะสม

    1.3 พัฒนาบุคลากร

2. Project planning สำคัญที่สุดและใช้เวลามากที่สุด จะต้องมีการวางแผนตั้งแต่เริ่มจนจบ มักจะมีการเปลี่ยนแปลงตลอด

โครงสร้างการวางแผนโครงการ

1. Project organization กำหนดลักษณะโครงสร้างของโครงการ

2.Risk analysis มีการวิเคราะห์ความเสี่ยง

3. Hardware and software requirement วิเคราะห์ความต้องการทั้งหมดในแง่ของ hardware software

4. Work breakdown แตกงานออกเป็นงานย่อยๆ

5. Monitoring and reporting machines ติดตามดูการพัฒนา

Risk management process

1. Risk Identification บอกให้ได้ว่าอะไรบ้างคือความเสี่ยง

2. Risk Analysis วิเคราะห์ความเสี่ยง ความเป็นไปได้ที่จะเกิดขึ้น

3. Risk planning วางแผนควบคุมความเสี่ยง

4. Risk monitoring ติดตามดูความเสี่ยง
