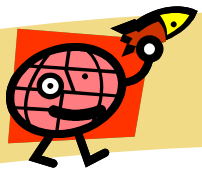
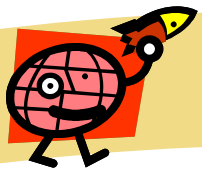


Generic View of Process

กระบวนการประกันคุณภาพซอฟต์แวร์

ซอฟต์แวร์มีต้นทุนมาจากการความรู้ที่ได้รับการรวบรวมขึ้นมา (Embodied Knowledge) โดยเริ่มแรกอาจกระจัดกระจาย แฝงอยู่ และไม่สมบูรณ์ เมื่อเริ่มพัฒนาซอฟต์แวร์ ซึ่งเป็นเสมือนกระบวนการเรียนรู้ทางสังคมอย่างหนึ่ง ทำให้เกิดการแลกเปลี่ยนความคิด ก่อให้เกิดความรู้ และเปลี่ยนไปสู่ตัวซอฟต์แวร์ ก่อให้เกิดกระบวนการโต้ตอบระหว่างผู้ใช้กับผู้ออกแบบ / ผู้ใช้กับเครื่องมือที่ปรับเปลี่ยน / ผู้ออกแบบกับเครื่องมือที่ปรับเปลี่ยน ซึ่งเป็นกระบวนการซ้ำแล้วซ้ำอีก



Generic View of Process

นิยามของกระบวนการทางซอฟต์แวร์ (Software Process)

" กรอบงานของการสร้างซอฟต์แวร์ที่มีคุณภาพสูง "

คุณภาพของซอฟต์แวร์ = ?

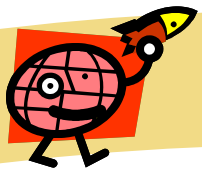
Boehm กำหนดคุณลักษณะของซอฟต์แวร์ที่มีคุณภาพ โดยวัดจากผู้ที่มีส่วนเกี่ยวข้องกับซอฟต์แวร์ 3 กลุ่ม คือ

กลุ่มที่ 1 คือ ผู้ใช้
วัดจากประโยชน์ต่อการใช้งาน (useful)

กลุ่มที่ 2 คือ ผู้บำรุงรักษาระบบ
วัดคุณภาพจากการอัฟเกรดและการเปลี่ยนแปลงระบบ

กลุ่มที่ 3 คือ โปรแกรมเมอร์ ที่ทำหน้าที่เปลี่ยนแปลงระบบตามที่ลูกค้าต้องการ
วัดจากความเอื้อประโยชน์ให้สามารถแก้ไขความผิดพลาดได้ง่าย

ซึ่งทั้งสามกลุ่ม มีความคาดหวังว่าระบบต้องมีความน่าเชื่อถือ และมีประสิทธิภาพ



Generic View of Process

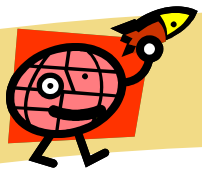
จากวิกฤตการณ์ซอฟต์แวร์ที่ส่วนใหญ่มักมีคุณภาพต่ำ ไม่สามารถพัฒนาซอฟต์แวร์ตรงตามความต้องการของผู้ใช้ เวลาส่งมอบและค่าใช้จ่ายที่ประมาณไว้ไม่ตรงตามแผน จึงสรุปปัญหาของโครงงานซอฟต์แวร์จากสาเหตุต่าง ๆ ดังต่อไปนี้

1. การสื่อสารที่ไม่ชัดเจนระหว่างผู้พัฒนาซอฟต์แวร์กับผู้ใช้งานระบบ ซึ่งอาจทำให้เกิดความเข้าใจผิด ทำให้ซอฟต์แวร์ที่พัฒนาไม่ตรงตามความต้องการ ไม่สามารถแก้ไขปัญหาได้ตรงตามวัตถุประสงค์
2. ไม่สามารถจัดการกับความเสี่ยง รวมทั้งความเปลี่ยนแปลงในความต้องการใหม่ ๆ ที่อาจเกิดขึ้นได้ในอนาคตหรือในระหว่างการพัฒนา
3. สมาชิกในทีมงานไม่มีมาตรฐานที่แน่นอนในการพัฒนาซอฟต์แวร์ ย่อมส่งผลต่อการนำโปรแกรมมาประกอบรวมกัน (Integrated)
4. ซอฟต์แวร์ไม่มีคุณภาพ จำเป็นต้องบำรุงรักษาอยู่ตลอดเวลา
5. ซอฟต์แวร์มีความซับซ้อนจนเกินไป
6. ความไม่ใส่ใจของหัวหน้าโครงการต่อการบริหารจัดการ ซึ่งอาจประเมินสถานะโครงการต่ำเกินไป หรือขาดประสบการณ์ของหัวหน้าโครงการ
7. ขั้นตอนหรือกระบวนการพัฒนาซอฟต์แวร์ไม่มีหลักการที่แน่นอน มีความน่าเชื่อถือต่ำ ไม่มีระบบการตรวจสอบที่ชัดเจน และขนาดเครื่องมือสนับสนุน

Generic View of Process

คุณภาพซอฟต์แวร์ย่อมเกิดจากกระบวนการผลิตซอฟต์แวร์ หากกระบวนการมีคุณลักษณะหรือมีประสิทธิภาพไม่เพียงพอ จะส่งผลให้ผลิตภัณฑ์ซอฟต์แวร์ไม่มีคุณภาพ อีกทั้งยังอาจทำให้ต้นทุนสูง และดำเนินงานล่าช้ากว่ากำหนดได้

กระบวนการทางซอฟต์แวร์ คือกรอบงานของการสร้างซอฟต์แวร์ที่มีคุณภาพสูง กระบวนการทางซอฟต์แวร์เป็นตัวกำหนดแนวทางที่ซอฟต์แวร์จะถูกสร้างขึ้น ในขณะที่วิศวกรรมซอฟต์แวร์จะรวมไปถึงเทคโนโลยีในกระบวนการ ได้แก่ วิธีเชิงเทคนิค และเครื่องมือทันสมัยต่างๆ

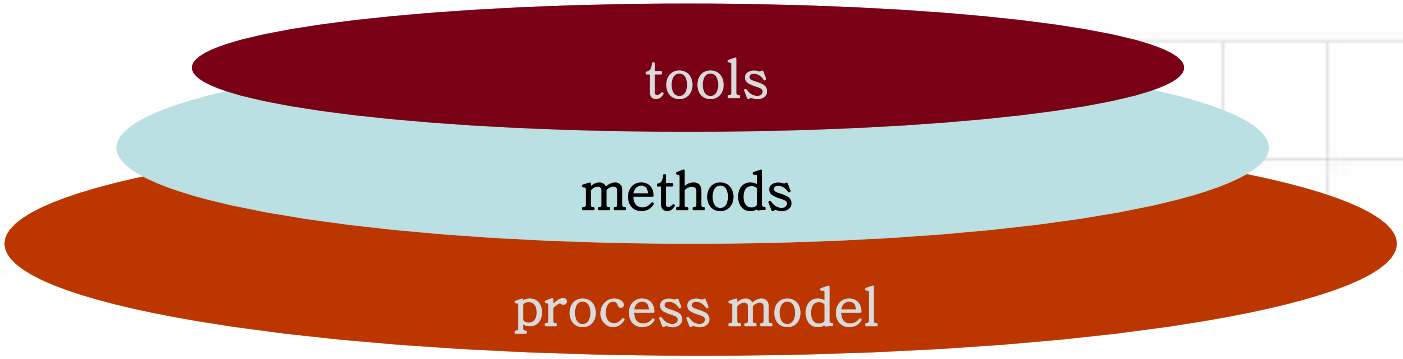


Generic View of Process

หากกล่าวถึงวิศวกรรมซอฟต์แวร์ในแง่ของการเป็นเทคโนโลยีของการผลิตซอฟต์แวร์แล้ว วิศวกรรมซอฟต์แวร์จะเป็นเทคโนโลยีชนิดที่เรียกว่า “เทคโนโลยีแบบชั้น” (Layered Technology) การดำเนินงานวิศวกรรมซอฟต์แวร์ จะประกอบไปด้วย 3 ระดับชั้น ดังนี้

A Layered Technology

Software Engineering



a “quality” focus

Generic View of Process

วิศวกรรมซอฟต์แวร์เป็นเทคโนโลยีแบบชั้น :

คุณภาพ (A Quality Focus)

: เป็นชั้นเทคโนโลยีเริ่มต้นที่ช่วยสนับสนุนวิศวกรรมซอฟต์แวร์ ในการพัฒนาให้เกิดความเหมาะสม เหมือนข้อผูกมัดขององค์กรที่มีคุณภาพ (a product should meet its specification) ซึ่งคุณภาพที่ดีต้องตรงกับความต้องการลูกค้า (efficiency, reliability) ผู้พัฒนา (maintainability, reusability) ผู้ใช้ (usability, efficiency) แต่พบว่า
- บางความต้องการยากที่จะทำให้เกิดคุณภาพ เพราะกำกวม หรือขัดแย้งระหว่างกันได้
- คุณลักษณะซอฟต์แวร์ที่ได้อาจไม่สมบูรณ์ หรือขาดองค์ประกอบที่ควรจะมี

กระบวนการ (Process Model)

: เป็นชั้นเทคโนโลยีพื้นฐานของวิศวกรรมซอฟต์แวร์ ซึ่งเชื่อมโยงเทคโนโลยีต่าง ๆ ในขณะพัฒนาซอฟต์แวร์ โดยกิจกรรมงานจะถูกกำหนด มีระยะการดำเนินงาน เพื่อให้เกิดมั่นใจในคุณภาพ และสามารถจัดการได้อย่างเหมาะสมเมื่อมีการแก้ไขปรับปรุง

Generic View of Process

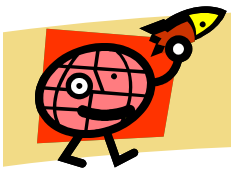
วิศวกรรมซอฟต์แวร์เป็นเทคโนโลยีแบบชั้น :

วิธี/เทคนิค (Method/Techniques)

: เป็นแนวทางด้านเทคนิคที่จะบอกถึงการสร้างซอฟต์แวร์ รวมเอาวิธีหลาย ๆ วิธี เช่น การติดต่อสื่อสาร การวิเคราะห์ความต้องการ แบบจำลอง การออกแบบ การสร้างโปรแกรม การทดสอบ และบำรุงรักษา ซึ่งเป็นหลักการพื้นฐานของเทคโนโลยีทุกสาขา

เครื่องมือเชิงวิศวกรรมซอฟต์แวร์ (Tools)

: เป็นสิ่งสนับสนุนให้กระบวนการ และวิธีการดำเนินงานไปได้อย่างอัตโนมัติ หรือกึ่งอัตโนมัติ โดยเครื่องมือทำให้เกิดสารสนเทศหรือนำไปใช้ประโยชน์ได้กับระบบอื่น ๆ ด้วย ถือว่าเป็นระบบที่ช่วยสนับสนุนการพัฒนาซอฟต์แวร์ เราเรียนกว่า computer-aided software engineering



Process Model

แบบจำลองการพัฒนา/ผลิตซอฟต์แวร์ (Process Model)

แบบจำลองใช้สำหรับชี้นำถึงกิจกรรมหลัก (key Activities) ในการพัฒนาซอฟต์แวร์ ด้วยการกำหนดรายละเอียดหรือข้อบัญญัติไว้ในแต่ละกิจกรรมในแต่ละขั้นตอนที่มีลำดับขั้นตอนการพัฒนาที่ชัดเจน เพื่อต้องการให้การพัฒนาซอฟต์แวร์ดำเนินต่อไปโดยเกิดปัญหาน้อยที่สุด ปัจจุบันมีการคิดค้นแบบจำลองต่าง ๆ ให้เลือกใช้งานมากมาย ส่วนการตัดสินใจว่าจะเลือกใช้แบบจำลองใดขึ้นอยู่กับปัจจัยหลายอย่าง เช่น ขนาดของโครงการ ความเหมาะสม และระดับความเสี่ยง เป็นต้น

สาเหตุสำคัญที่จำเป็นต้องใช้แบบจำลองการพัฒนาซอฟต์แวร์ เพราะ

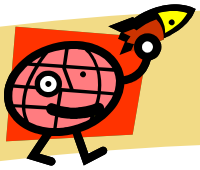
1. แบบจำลองพัฒนาซอฟต์แวร์จะมีการแตกขั้นตอนของกระบวนการพัฒนาในแต่ละเฟส (phase)
2. ซอฟต์แวร์ที่พัฒนามีความซับซ้อน
3. การแบ่งเป็นกระบวนการพัฒนาเป็นเฟสหรือระยะ จะทำให้ง่ายต่อการจัดการ
4. แต่ละเฟสมีแนวทางต่าง ๆ ให้เลือกปฏิบัติ

Process Model

แบบจำลองพัฒนาซอฟต์แวร์ที่สำคัญ

แบบจำลองมักผนวกกระบวนการในลักษณะ การทวนซ้ำเป็นรอบ (Iteration) การพัฒนาแบบก้าวหน้า (Incremental) และการจัดทำต้นแบบ (Prototyping) ซึ่งเป็นกระบวนการลดความเสี่ยงในกรณีซอฟต์แวร์มีการเปลี่ยนแปลงในความต้องการอยู่ตลอดเวลา แบบจำลองที่สำคัญและนิยมนำมาใช้ในการพัฒนาได้แก่

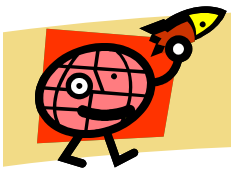
- Waterfall Model : แบบจำลองน้ำตก
- Incremental Model : แบบจำลองค่อยๆ สร้างเพิ่ม
- RAD Model : แบบจำลองอาร์เอดี
- Prototype Model : แบบจำลองต้นแบบ
- Spiral Model : แบบจำลองวนก้นหอย/เวียนวน
- Agile Model ★



Process Model

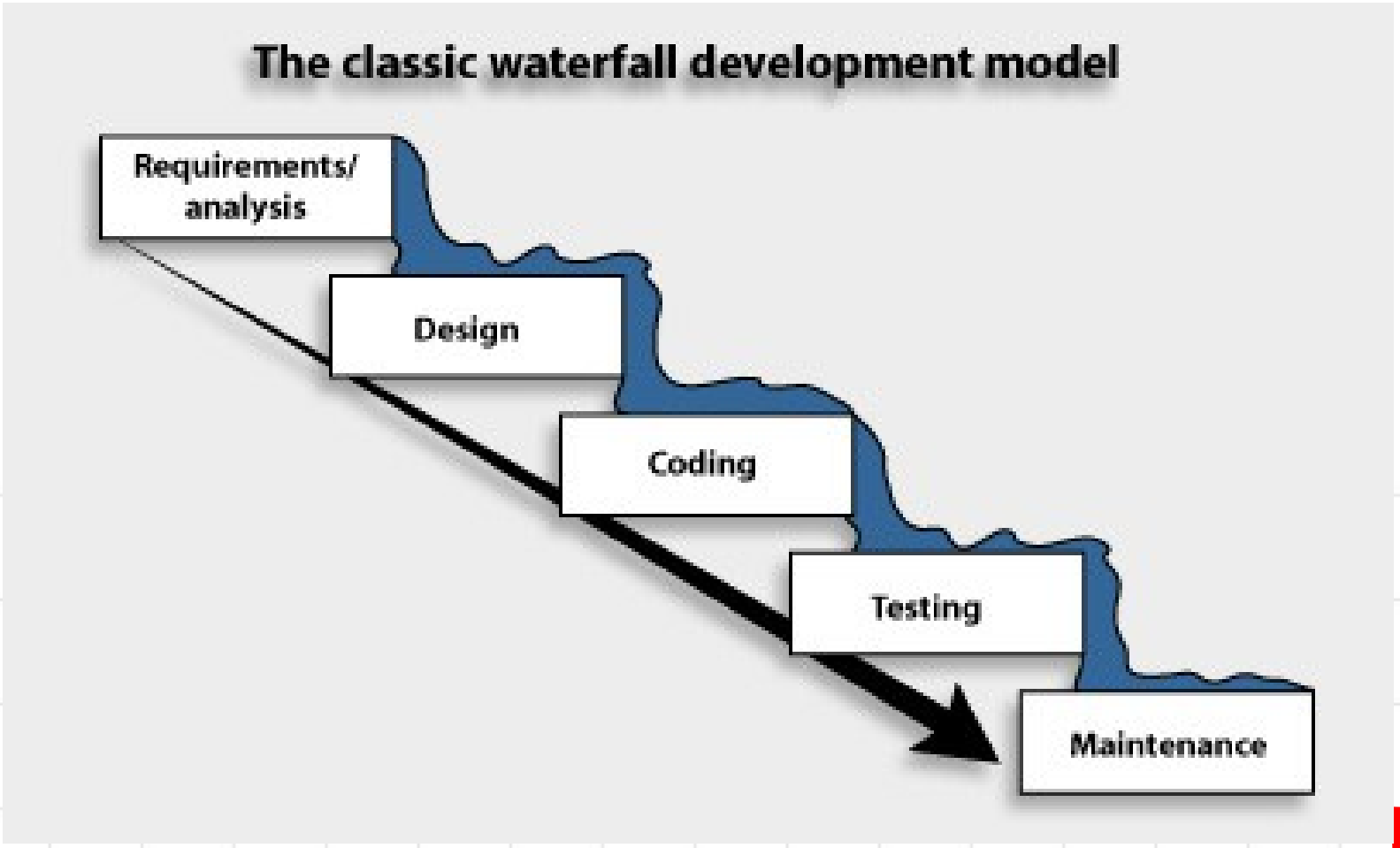
Waterfall Model (แบบจำลองน้ำตก)

- เผยแพร่ในปี ค.ศ.1970 เป็นวิธีการที่ได้รับการพัฒนาขึ้นมาตั้งแต่แรกเริ่มที่มีกระบวนการ พัฒนาซอฟต์แวร์ บางครั้งเรียกว่า วงจรชีวิตแบบคลาสิก (classic life cycle) ตามแนวทาง SDLC
- แบ่งกระบวนการทำงานออกเป็นขั้นตอนต่าง ๆ ตามลำดับจึงเรียกลักษณะของแบบจำลองว่า Sequential Model
- ขั้นตอนในแต่ละช่วงจะสืบเนื่องกันไปจากขั้นหนึ่งสู่อีกขั้นหนึ่งตามลำดับเหมือนสายน้ำตก
- สามารถย้อนกลับไปปรับปรุงขั้นตอนก่อนหน้าได้ตามลำดับ โดยเพิ่มคุณสมบัติการทวนซ้ำเป็นรอบ (Iteration)
- แนวทางนี้ผู้ใช้จะเห็นระบบใหม่ก็ต่อเมื่อโครงการเสร็จสิ้น

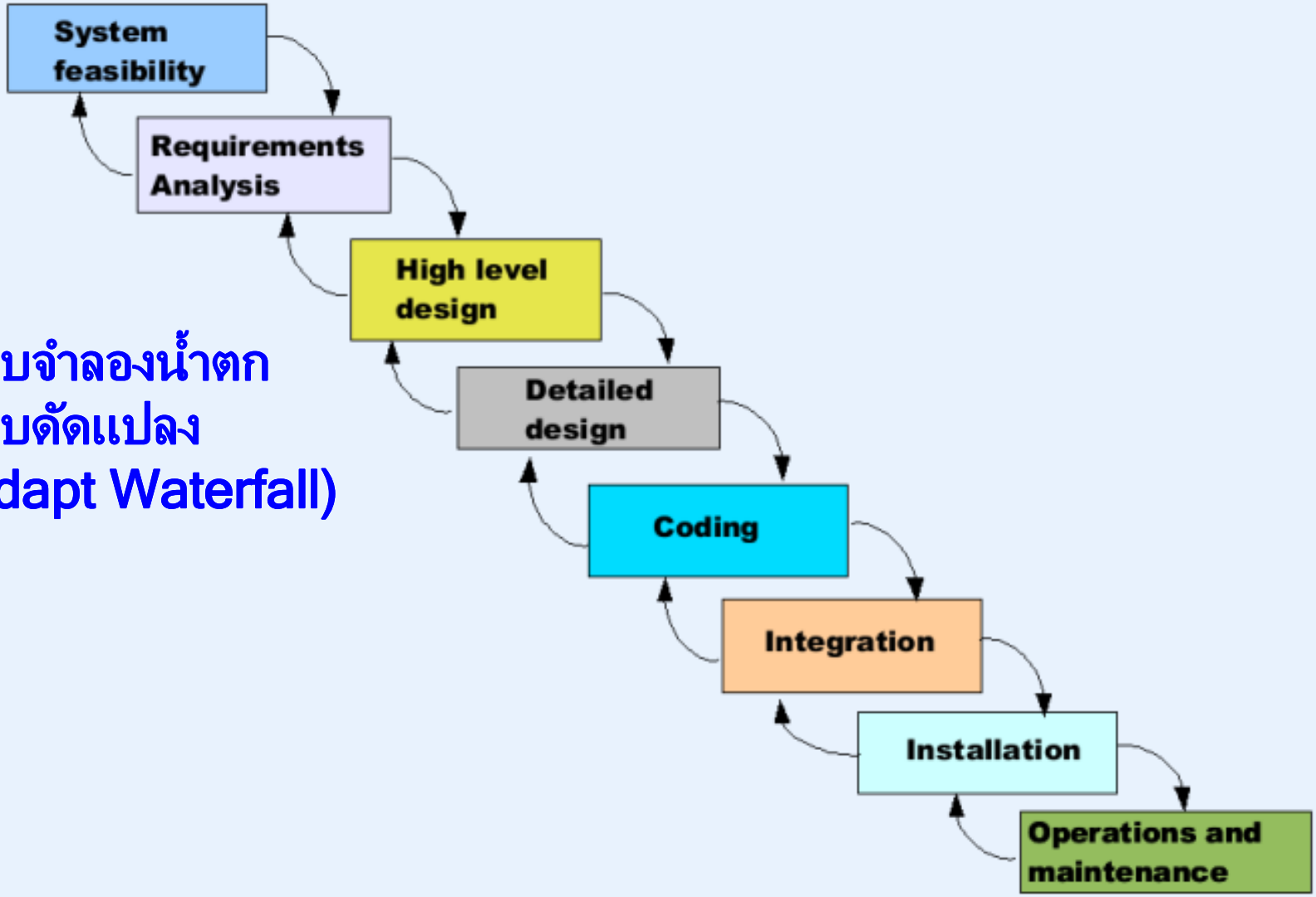


Process Model

แบบจำลองน้ำตก (Waterfall Model)



Process Model



แบบจำลองน้ำตก
แบบดัดแปลง
(Adapt Waterfall)

Process Model

แบบจำลองน้ำตก (Waterfall Model)

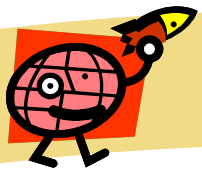
- **Requirements analysis and definition**
  - การวิเคราะห์และให้คำจำกัดความของระบบงาน
- **System and software design**
  - ในการออกแบบระบบ ผู้ออกแบบต้องคำนึงถึงโครงสร้างของฮาร์ดแวร์และซอฟต์แวร์ที่จำเป็นต้องพัฒนา
  - ในการออกแบบซอฟต์แวร์เป็นการกำหนดโครงสร้างหลักของการโปรแกรมที่จะพัฒนา
- **Implementation and unit testing**
  - การกำหนดสร้างและทดสอบหน่วยย่อย
  - เป็นการแบ่งส่วนของซอฟต์แวร์ออกเป็นหน่วยย่อย ๆ
  - ทำการตรวจสอบความถูกต้องหลังจากเขียนโปรแกรมเสร็จสิ้น

Process Model

แบบจำลองน้ำตก (Waterfall Model)

- **Integration and system testing**
  - การเชื่อมโยงและทดสอบทั้งระบบ
  - ทำการตรวจสอบหลังจากนำโปรแกรมย่อยในแต่ละส่วนมารวมกัน

- **Operation and maintenance**
  - การติดตั้งใช้งานและการบำรุงรักษา
  - เป็นขั้นตอนที่ใช้เวลานานที่สุด
  - ขั้นตอนในการแก้ไขข้อผิดพลาด การปรับแต่ง



Process Model

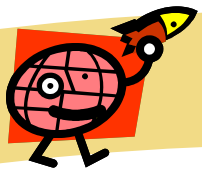
แบบจำลองน้ำตก (Waterfall Model)

❑ ข้อดี

- แบ่งงานยากให้เป็นงานที่เล็ก งานต่อการจัดการ
- มีการกำหนด product ที่ต้องส่งมอบในแต่ละงานอย่างชัดเจน
- *Simple, step-by-step, focused, and easy to follow..*

❑ ข้อเสีย

- ถ้า ค้นพบข้อผิดพลาดของขั้นที่เสร็จสิ้นแล้ว ไม่สามารถแก้ไขได้ การแก้ไขจำเป็นต้องเริ่มรอบ (Iteration) ใหม่ ซึ่งในความเป็นจริง หลังการทำงานในแต่ละขั้นควรสามารถย้อนไปแก้ไขความผิด-พลาดในขั้นใดใดก็ได้ก่อนหน้า
- ดังนั้นในทางปฏิบัติขั้นตอนการทำงานในWaterfall จึงไม่เป็นเชิงเส้น (Linear)
- ข้อเสียหลักคือ ลูกค้าเห็นและทดลองใช้Software ก็ต่อเมื่อถึงขั้น-ตอนสุดท้าย หากมีบางอย่างที่ไม่ตรงกับความต้องการของลูกค้า การแก้ไขยาก แพง เสียเวลา



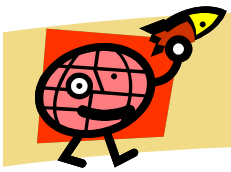
Process Model

แบบจำลองค่อย ๆ สร้างเพิ่ม (Incremental Model)

เป็นแบบจำลองที่มีส่วนผสมของ Waterfall model กับการทำซ้ำลักษณะลูป (Loop) จะเห็นว่าแบบจำลองนี้จะมีลำดับเชิงเส้นแบบวนซ้ำหลายรอบ โดยวางให้เหลื่อมกันตามเวลาปฏิทิน โดยขจัดข้อเสียของ Waterfall Model ที่มีกระบวนการทดสอบอยู่ตอนท้าย ๆ แต่ Incremental Model จะพัฒนาแบบทวนซ้ำเป็นรอบพร้อมกับมีระบบตรวจสอบ

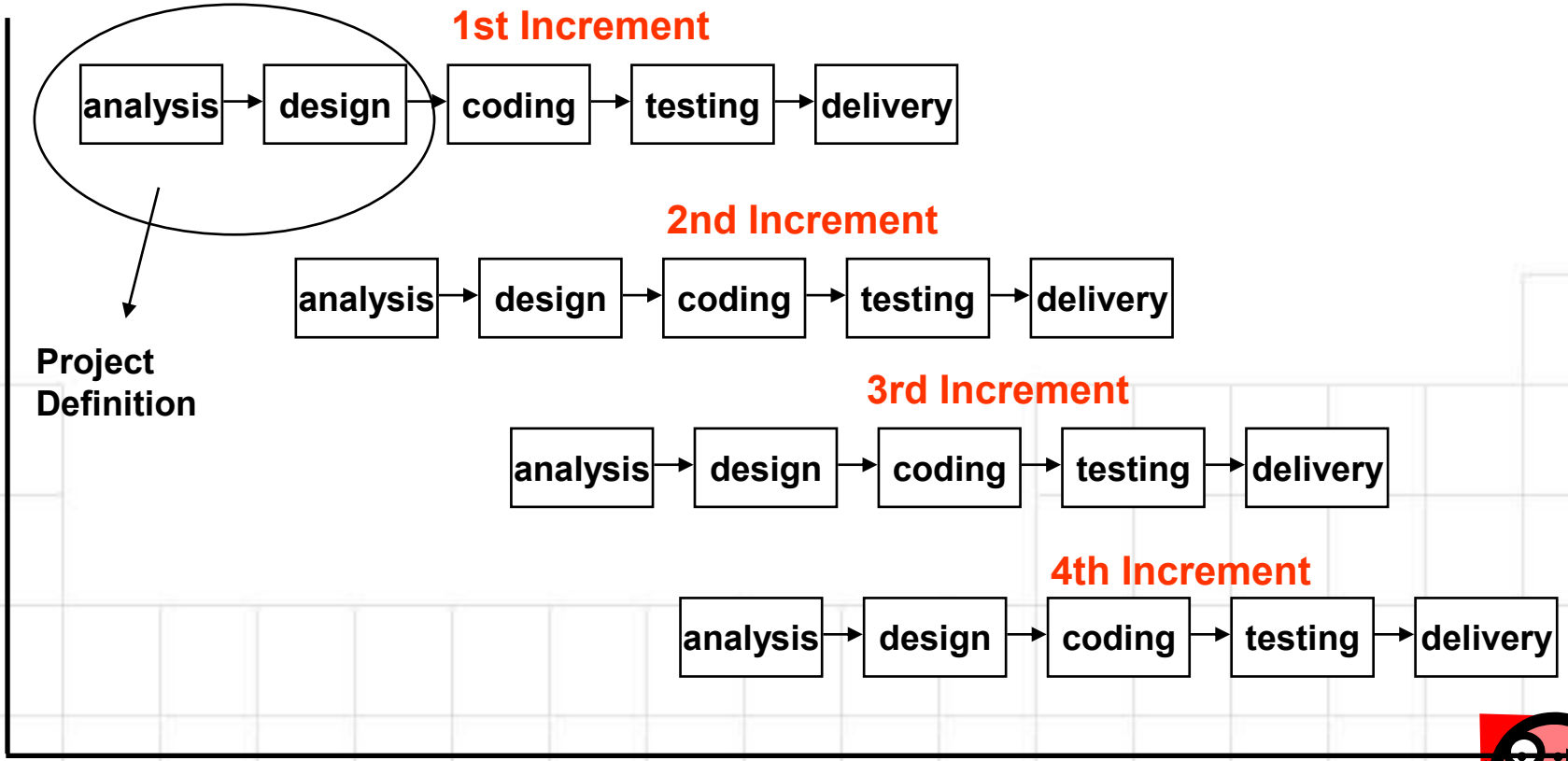
Verification : Are we building the product right ?
การตรวจสอบความถูกต้องตามข้อกำหนด (Specification) หรือพยายามค้นหาข้อผิดพลาดจากการประมวลผลโปรแกรม

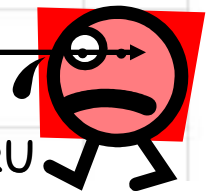
Validation : Are we building the right product ?
ตรวจสอบรายละเอียดของผลิตภัณฑ์ว่าตรงตามความต้องการของผู้ใช้หรือไม่



Process Model

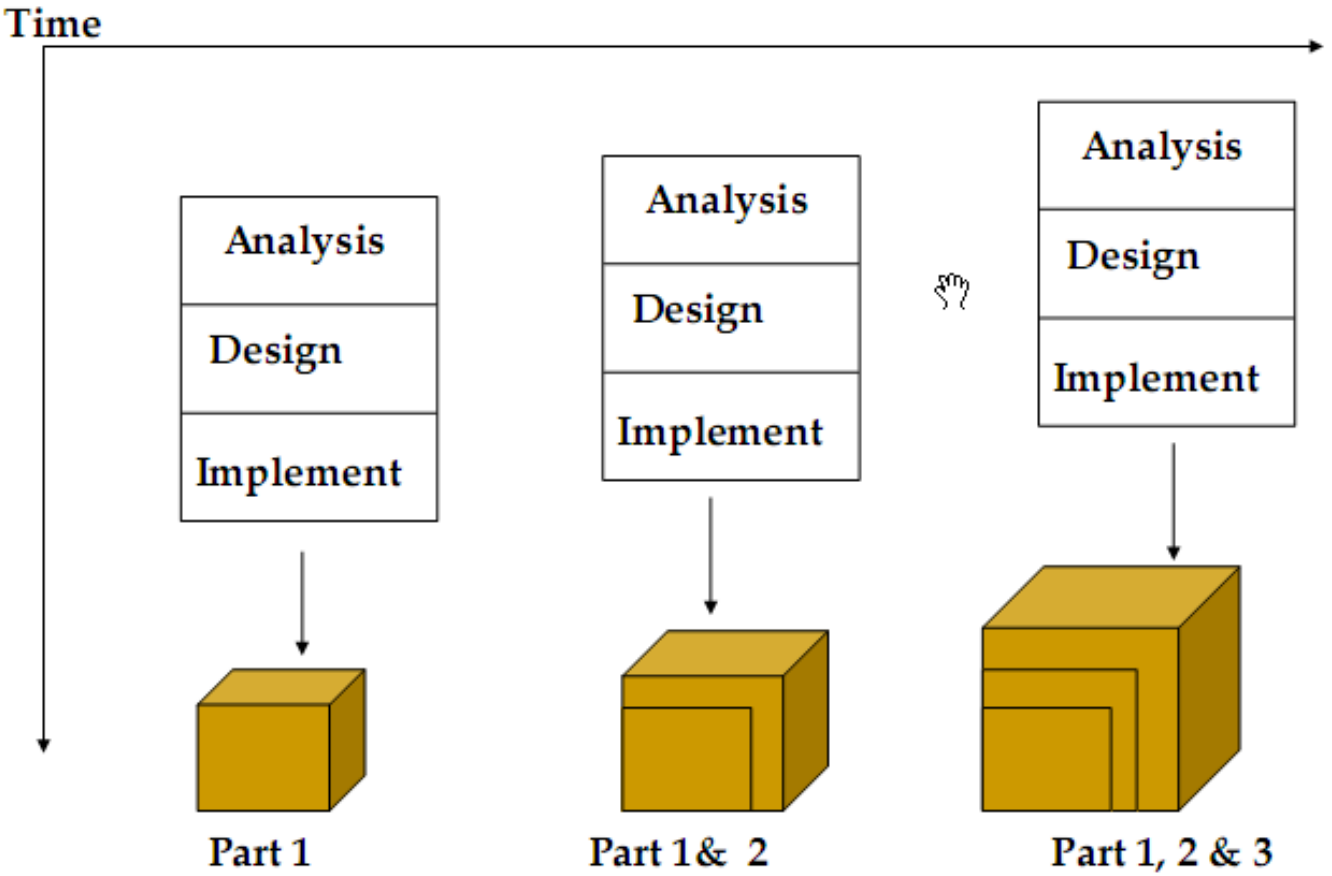
เราจะแบ่งงานออกเป็นงานย่อยๆ และเรียกงานย่อยๆ นั้นว่า Increment แล้วจึงค่อยพัฒนางานให้สำเร็จไปเป็นแต่ละ Increment ไป โดยเราจะทำการพัฒนางานที่เป็นแกนของโครงการก่อน แล้วค่อยพัฒนา Increment ย่อยอื่นๆ ตามความสำคัญของงาน ไปจนจบโครงการ ซึ่งการพัฒนาในแต่ละ Increment จะต้องมีการตรวจสอบและยอมรับจาก customer





Process Model

แบบจำลองแบบค่อยๆสร้างเพิ่ม มีลักษณะคล้ายกับแบบวิวัฒนาการ ต่างกันที่ระบบที่สร้างได้ในตอนแรกยังไม่ใช่ระบบที่สมบูรณ์ แต่เป็นส่วนหนึ่งของทั้งหมด จะต้องเพิ่มส่วนอื่นๆไปเรื่อยจนได้ระบบที่สมบูรณ์ ในช่วงรอบแรกๆนั้นเก็บข้อมูลแค่คร่าวๆ แล้วค่อยรื้อปรับปรุงในรอบหลังๆ



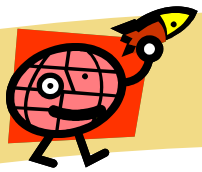
Process Model

แบบจำลองค่อย ๆ สร้างเพิ่ม (Incremental Model)

แต่โครงการที่จะใช้การพัฒนาแบบ incremental ได้ จะต้องเป็นโครงการที่ มีการเปลี่ยนแปลงของ requirement ไม่มาก ซึ่งการเปลี่ยนแปลงจะต้องไม่กระทบต่อ Increment ที่ได้ทำการพัฒนาและส่งมอบไปแล้ว และเป็นโครงการที่ใช้เวลาไม่นานมากนัก

- **ข้อดี**
  - การพัฒนาซอฟต์แวร์ทำได้อย่างรวดเร็วโดยดำเนินงานระหว่างวงจรซอฟต์แวร์
  - มีความยืดหยุ่นสูง ค่าใช้จ่ายน้อยในการเปลี่ยนแปลงขอบเขตหรือความต้องการ
  - ตรวจสอบข้อผิดพลาดได้ง่ายและจัดการความเสี่ยงได้ง่ายเพราะความเสี่ยงของชิ้นงานถูกกำหนดและจัดการขณะที่ทำซ้ำ
  - ส่งมอบงานได้เร็ว ลดอัตราเสี่ยงของการเลิกจ้าง
  - นักวิศวกรและ customer สามารถมองเห็นความก้าวหน้าของโครงการได้
- **ข้อเสีย**
  - ระหว่างรอบการผลิตซ้ำต้องเข้มงวดและตรวจสอบไม่ให้เกิดการทับซ้อนกัน
  - ปัญหาอาจจะปรากฏในโครงสร้างทางสถาปัตยกรรมของระบบได้ เพราะว่า ความต้องการบางส่วนอาจไม่ได้ถูกรวมไว้ตั้งแต่เริ่มก่อนเข้ามาในวงจรของการพัฒนาซอฟต์แวร์

project calendar time



Process Model

จากแบบจำลองการพัฒนาซอฟต์แวร์ทั้งแบบ Waterfall และ Incremental อาจทำให้สับสนระหว่างคำว่า Iteration และ Incremental จึงสรุปได้ดังนี้

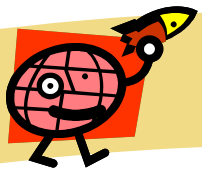
- **การทวนซ้ำเป็นรอบ (Iteration)**

  คือ การทวนซ้ำของงาน ซึ่งมีความเป็นไปได้ว่า การพัฒนางานใด ๆ อาจมีความจำเป็นต้องมีการทวนซ้ำของงานมากกว่าหนึ่งรอบ เพื่อให้งานตรงกับความต้องการมากที่สุด การทวนซ้ำมากกว่าหนึ่งรอบถือเป็นการทบทวนความถูกต้องและลดความเสี่ยง

- **การพัฒนาแบบก้าวหน้า (Incremental)**

  คือ วิธีการพัฒนาส่วนย่อยของระบบให้สมบูรณ์ โดยแต่ละระบบย่อยหรือแต่ละส่วนที่พัฒนานั้นอาจมีจำนวนการทวนซ้ำหลายรอบ จากนั้นก็ทำการเพิ่มระดับความก้าวหน้าในส่วนงานที่ทำเสร็จ และท้ายสุดก็จะนำส่วนงานย่อยที่เสร็จสมบูรณ์เหล่านั้นมาประกอบรวมกัน ให้เป็นหนึ่งระบบที่สมบรูณ์

project calendar time



Process Model

แบบจำลองอาร์เอดี (RAD Model)

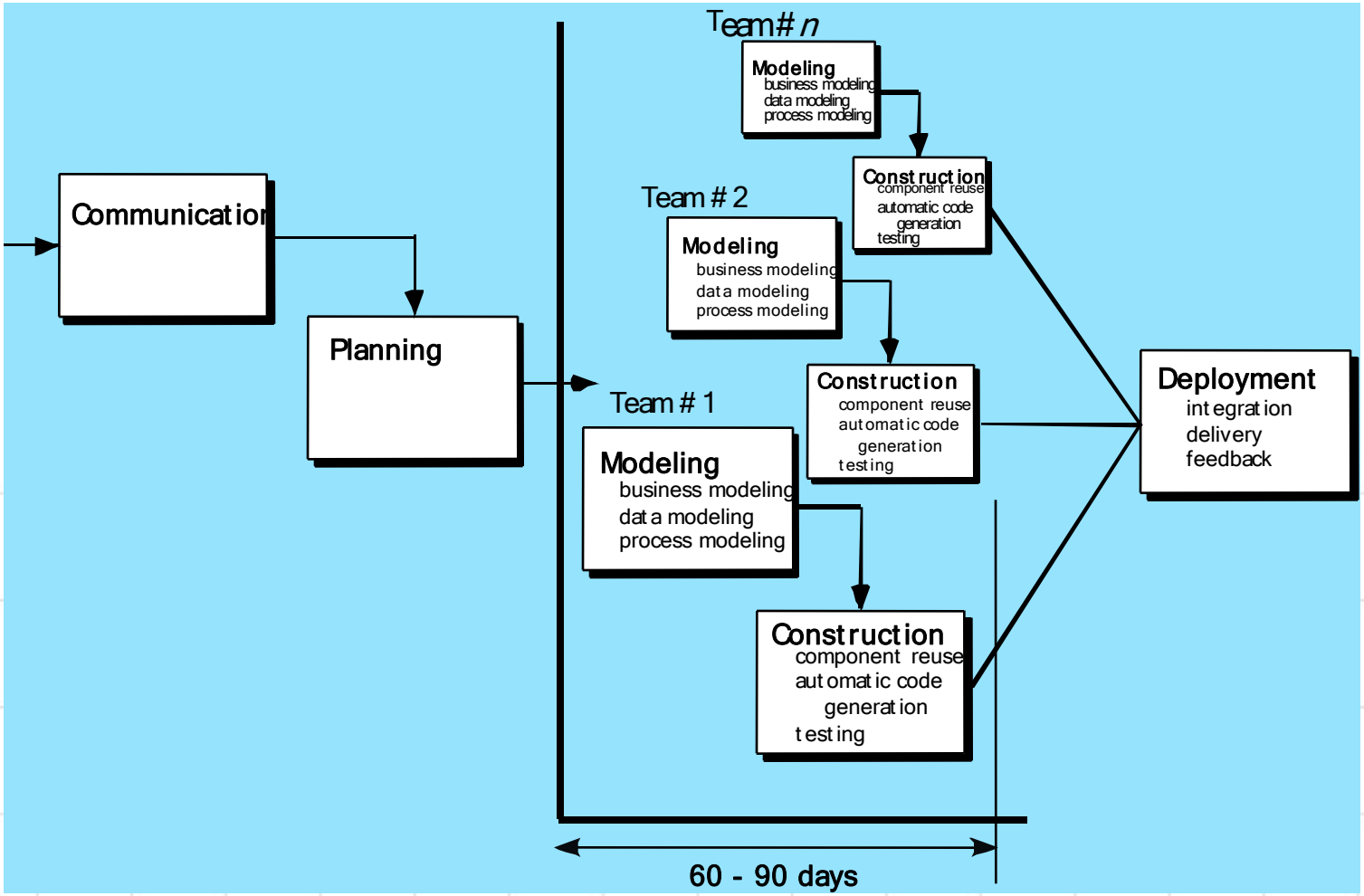
RAD (Rapid Application Development) เป็นแบบจำลองกระบวนการซอฟต์แวร์แบบค่อยเพิ่มขึ้นที่เน้นวงจรพัฒนาสั้น ๆ เพื่อการพัฒนาแอพพลิเคชั่นอย่างรวดเร็ว เป็นการดัดแปลงจาก Waterfall Model โดยนำวิธีการประกอบคอมโพเนนท์ย่อยเข้าด้วยกัน โดยมุ่งเน้นด้านการลดต้นทุนและระยะเวลาในการพัฒนา เพื่อความคล่องตัวจึงจำเป็นต้องมี**ทีมงาน**ที่มีความรู้ความสามารถเฉพาะ เช่น ผู้เชี่ยวชาญด้านเทคโนโลยีสารสนเทศกับกลุ่มผู้ใช้งานมาร่วมกันพัฒนา เป็นต้น

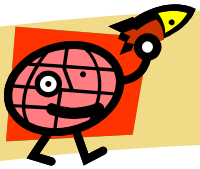
เทคนิคสำคัญในการพัฒนาซอฟต์แวร์แบบ RAD

1. พัฒนาต้นแบบได้อย่างรวดเร็ว
2. เป็นแหล่งรวบรวมเครื่องมือเพื่อการพัฒนาแอปพลิเคชัน
3. มีทีมงานที่เชี่ยวชาญการใช้เครื่องมือเหล่านั้น
4. มีกรอบระยะเวลาการพัฒนาที่จำกัด
5. มีแนวปฏิบัติแบบพัฒนาแอปพลิเคชั่นแบบร่วมกัน

Process Model

แบบจำลองอาร์เอดี (RAD Model)





Process Model

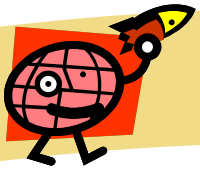
แบบจำลองอาร์เอดี (RAD Model)

ข้อดี

- ใช้เวลาในการพัฒนาสั้น
- ลดค่าใช้จ่ายเพราะใช้หลักการนำกลับมาใช้ใหม่ และสร้างแบบแบ่งส่วนประกอบย่อย ๆ

ข้อเสีย

- โครงการส่วนใหญ่ที่ใช้วิธีนี้ จำเป็นต้องมีทรัพยากรบุคคลจำนวนมากเพื่อแบ่งเป็นทีมหลาย ๆ ทีมยกเว้นโครงการที่ไม่สามารถขยายส่วนได้
- โครงการอาจล้มเหลวได้โดยง่าย ถ้านักพัฒนาและลูกค้าไม่ทำตามกิจกรรมที่จำเป็นสำหรับการพัฒนาอย่างรวดเร็ว
- ระบบที่ไม่สามารถแบ่งเป็นโมดูลได้ จะมีปัญหาในการสร้างคอมโพเนนท์ที่จะใช้
- ถ้าระบบต้องการประสิทธิภาพสูง ๆ โดยการปรับแต่งส่วนเชื่อมต่อคอมโพเนนท์ วิธีการนี้อาจไม่ได้ผล
- อาจไม่เหมาะสมกับระบบที่มีความเสี่ยงด้านเทคนิคสูง และเกี่ยวข้องกับเทคโนโลยีใหม่ ๆ



Process Model

แบบจำลองกระบวนการเชิงวิวัฒน์ (Evolutionary Process Model)

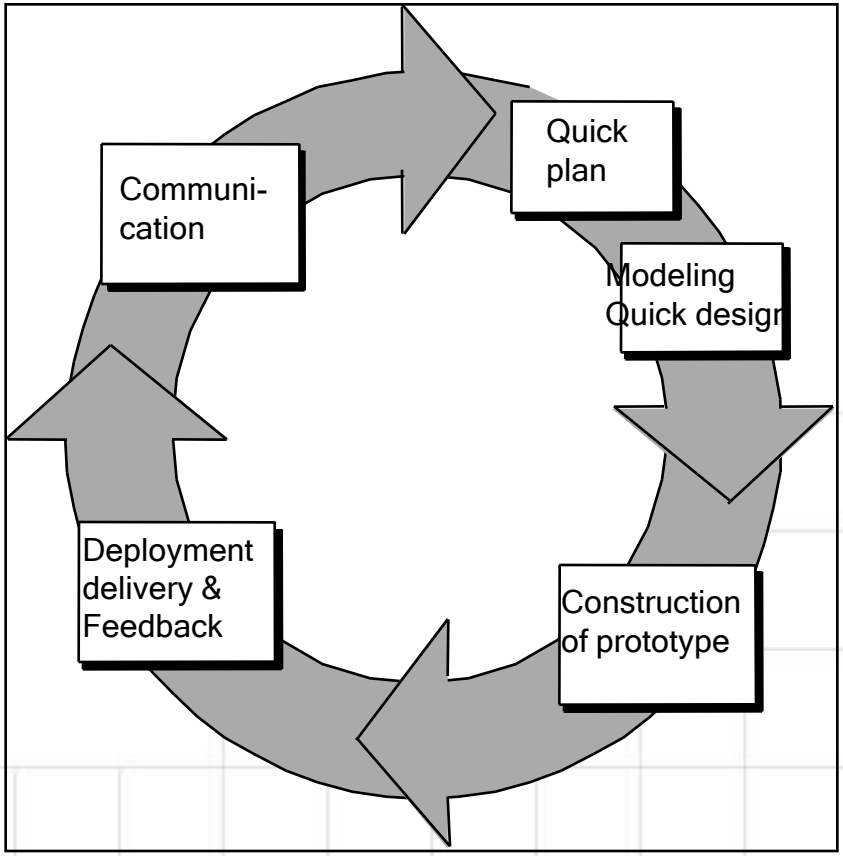
ซอฟต์แวร์จะมีวิวัฒนาการตามกาลเวลาที่ผ่านไป ความต้องการผลิตภัณฑ์และธุรกิจมักเปลี่ยนแปลงไปเพื่อพัฒนาก้าวหน้าขึ้น ดังนั้นเมื่อการพัฒนาไม่สอดคล้องกับการตลาดจึงทำให้เกิดซอฟต์แวร์ที่ไม่สมบูรณ์ 100% แต่มักจะออกเวอร์ชั่นที่ทำงานอย่างจำกัดออกมาก่อน และขยายการทำงานต่อไปเรื่อย ๆ ซอฟต์แวร์เวอร์ชั่นหลัง ๆ จะมีความสมบูรณ์เพิ่มมากขึ้น ดังเช่น แบบจำลองหลัก 2 รูปแบบ ต่อไปนี้

- แบบจำลองต้นแบบ (Prototyping Model)
- แบบจำลองสไปรัล (Spiral Model)

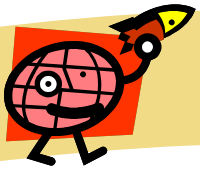
Process Model

แบบจำลองต้นแบบ (Prototyping Model)

- step 1: Requirements gathering
-step 2: A "quick design" -> focuses on visible functions and behaviors of the product
- step 3: Prototype construction
- step 4: Customer evaluation of the prototype loop back step 1.



บ่อยครั้งที่ลูกค้ามักบอกเป้าหมายทั่วไปของซอฟต์แวร์ โดยไม่ระบุรายละเอียดด้าน Input-Process-Output ที่ต้องการ ซึ่งทำให้ผู้พัฒนาไม่มั่นใจในการเลือกใช้อัลกอริทึ่มหรือโครงการ ดังนั้นวิธีการที่ดีที่สุดคือการสร้างต้นแบบเพื่อช่วยให้นักวิศวกรและลูกค้าได้เข้าใจกันดียิ่งขึ้น



Process Model

แบบจำลองต้นแบบ (Prototyping Model)

- **ข้อดี**
  - ง่ายและรวดเร็วตามความต้องการของลูกค้า
  - ลูกค้าสามารถตรวจสอบต้นแบบแต่ละขั้นและจัดหาข้อมูลเพิ่มเติมพร้อมผลตอบรับ
  - เป็นแนวคิดที่ดี ถ้าพบกรณีดังนี้
    - ✓ ลูกค้าไม่สามารถหากำหนดความต้องการโดยละเอียดได้
    - ✓ มีความยุ่งยากในการสื่อสารเรื่องระบบกับลูกค้า
    - ✓ มีการใช้เทคโนโลยี, ฮาร์ดแวร์ และอัลกอริทึ่มใหม่ ๆ
    - ✓ เป็นการพัฒนาจุดหลักของแอพพลิชั่นระบบ
- **ข้อเสีย**
  - ลูกค้าสัมผัสกับต้นแบบโดยไม่ทราบว่าสร้างมาอย่างหยาบ ๆ และหากมีการปรับปรุงบำรุงรักษาให้เกิดความสมบูรณ์ลูกค้าอาจไม่เข้าใจ
  - ลูกค้าบางครั้งอาจกังวลว่าต้นแบบไม่ใช่ซอฟต์แวร์ที่ใช้งานได้จริง
  - ผู้พัฒนาต้องระลึกอยู่เสมอว่าต้นแบบ คือ ระบบเบื้องต้น เครื่องมือหรืออัลกอรึทึ่มที่ใช้นำมาอย่างง่าย ๆ เพื่อนำเสนอ ห้ามยึดติดเพราะเครื่องมือเหล่านี้อาจไม่เหมาะสมกันสถานการณ์จริง

Process Model

แบบจำลองสไปรัล (Spiral Model)

แบบจำลองสไปรัลเกิดจากการวิจัยของ Barry Boehm (1988) เป็นแบบจำลองที่รวมลักษณะการวนซ้ำของการสร้างต้นแบบกับการทำงานอย่างเป็นขั้นตอน และมีการควบคุมของแบบจำลองน้ำตกเข้าด้วยกัน

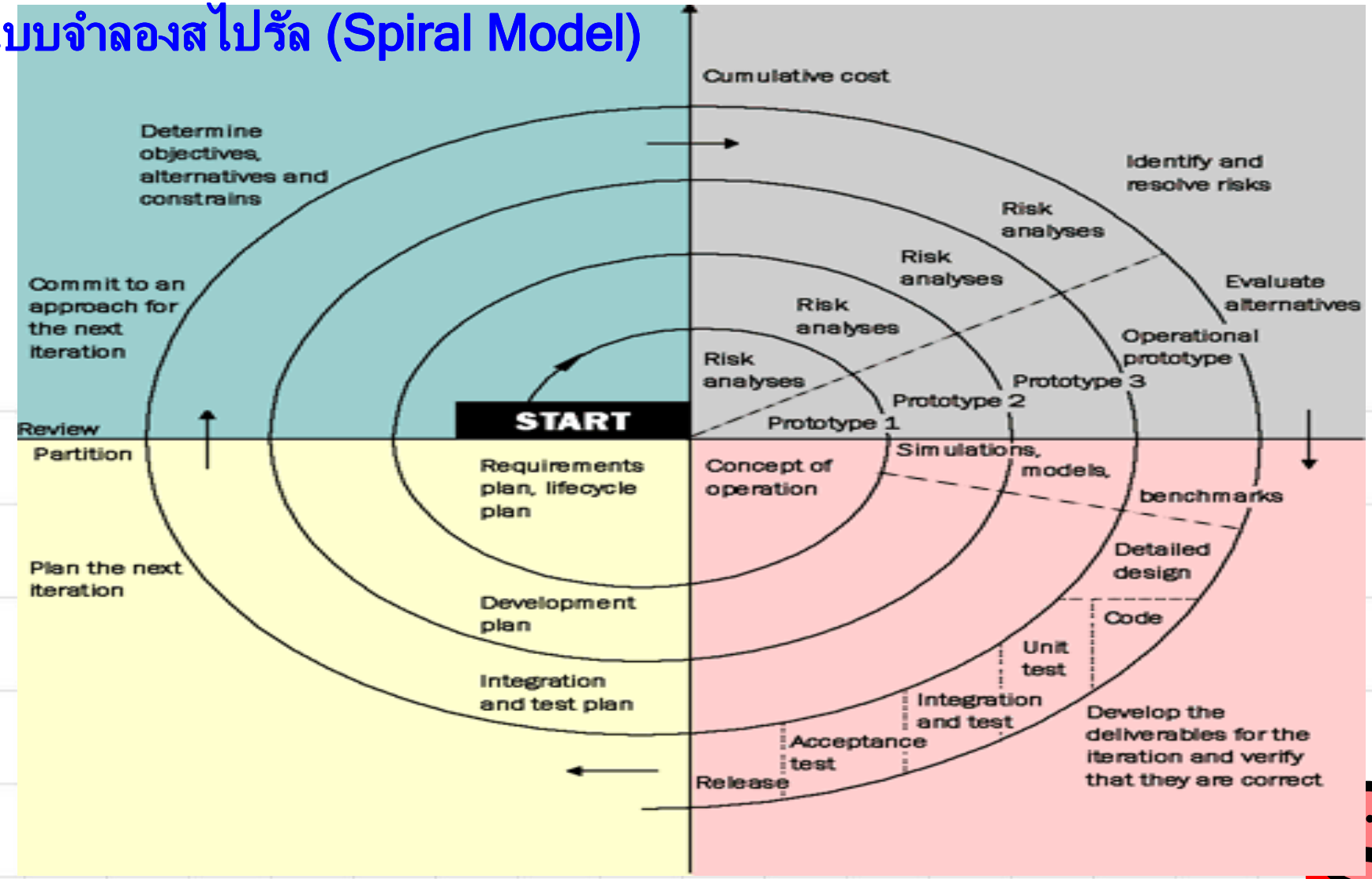
วิธีการพัฒนาแบบ "วนเวียน" (spiral model) นี้จะวนเวียนอยู่ระหว่างงาน 4 ขั้นตอน

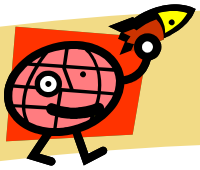
1. การวิเคราะห์ความต้องการ (Requirement Analysis)
2. การวิเคราะห์ความเสี่ยง (Risk Analysis)
3. การออกแบบต้นแบบ (Design Prototype)
4. การพัฒนาต้นแบบและนำมาประกอบรวมกัน (Develop and Integrate Prototype)

วิธีนี้จะเน้นเรื่องการวิเคราะห์ความเสี่ยงในช่วงที่ 2 ซึ่งการวิเคราะห์ความเสี่ยง (risk-analysis) เป็นการวิเคราะห์ว่า ในการพัฒนาซอฟต์แวร์มีปัญหาอะไรบ้าง แต่ละปัญหาอาจส่งผลกระทบรุนแรงในระดับใด และมีโอกาสเกิดขึ้นได้มากเพียงใด แล้วรีบจัดการกับปัญหาที่มีโอกาสเกิดขึ้นและมีผลกระทบสูงก่อน ปัญหาดังกล่าวไม่จำเป็นว่าจะต้องเป็นปัญหาทางด้านเทคนิค แต่อาจจะเป็นปัญหาในการบริหารงานโครงการ ปัญหาเรื่องคน ฯลฯ เพื่อให้โครงการประสบความสำเร็จได้มากขึ้น ตัวอย่างเช่น หัวหน้าโครงการกำลังคิดจะลาออก ซึ่งถ้าลาออกจริง โครงการก็จะล้มเหลว การเจรจากับหัวหน้าโครงการจึงเป็นเรื่องเร่งด่วนกว่าเรื่องเทคนิคทั้งหมด

Process Model

แบบจำลองสไปรัล (Spiral Model)





Process Model

แบบจำลองสไปรัล (Spiral Model)

ขอบเขตการทำงานที่สำคัญ:

- The customer communication task
  เพื่อให้การปฏิสัมพันธ์ที่ดีและให้ผลลัพธ์ที่ตรงตามความต้องการระหว่างลูกค้าและผู้พัฒนา
- The planning task
  เพื่อวางแผนและกำหนดทรัพยากร เวลา และข้อมูลสารสนเทศอื่น ๆ ที่เกี่ยวข้องกับโครงงาน
- The risk analysis task
  เพื่อประกันคุณภาพความเสี่ยงเชิงเทคนิคและการจัดการ
- The engineering task
  เพื่อผลิตงานนำเสนอตัวแอพพลิเคชั่นของระบบให้แก่ลูกค้าหรือผู้พัฒนาเพื่อให้เกิดความเข้าใจร่วมกัน
- The construction and release task
  เพื่อสร้าง ทดสอบ ติดตั้ง และ จัดหาเครื่องสนับสนุนการใช้งานแก่ลูกค้า หรือผู้ใช้งาน อาทิ เอกสารคู่มือ และฝึกการอบรม เป็นต้น
- The customer evaluation task
  เพื่อให้ได้ผลตอบรับของลูกค้าที่จะนำมาทำให้เกิดวิวัฒนาการของสร้างผลิตภัณฑ์ขณะที่อยู่ในสถานะดำเนินงาน และได้ผลตอบรับด้านพัฒนาขณะที่กำลังอยู่ในสถานะติดตั้ง

Process Model

แบบจำลองสไปรัล (Spiral Model)

- **ข้อดี**
  - ใช้การวิเคราะห์ความเสี่ยงจำนวนมากสูง
  - ดีต่อโครงการที่มีขนาดใหญ่และมีความเสี่ยงสูง
  - ซอฟต์แวร์จะถูกผลิตภายในช่วงวงจรการพัฒนา

- **ข้อเสีย**
  - เสียค่าใช้จ่ายสูงในการทำแบบจำลอง
  - การวิเคราะห์ความเสี่ยงต้องการบุคลากรที่มีความเชี่ยวชาญ
  - ความสำเร็จของผลิตภัณฑ์ขึ้นอยู่กับช่วงการวิเคราะห์ความเสี่ยง หากความเสี่ยงสำคัญไม่ถูกค้นพบ อาจเกิดปัญหาขึ้นในภายหลังได้
  - ไม่เหมาะในการนำไปใช้กับโครงการที่มีขนาดเล็ก

Process Model

สรุปแบบจำลองกระบวนการเชิงวิวัฒน์

จุดอ่อนของแบบจำลองกลุ่มนี้

(1) การพัฒนาซอฟต์แวร์ตามแบบกระบวนการนี้ สร้างปัญหาในการวางแผนโครงการ เพราะความไม่แน่นอนของจำนวนการทำซ้ำ เพราะผู้บริหารที่วางแผนโครงการมักมองเวลาในลักษณะเชิงเส้น เริ่มเมื่อไหร่ สิ้นสุดเมื่อไหร่ จึงมักขัดแย้งกัน

(2) กระบวนการนี้ไม่ได้กำหนดความเร็วของวิวัฒนาการซอฟต์แวร์ ถ้าวิวัฒนาการของซอฟต์แวร์เกิดขึ้นเร็วโดยไม่มีช่วงพัก กระบวนการจะยุ่งเหยิง สับสน โดยเฉพาะผู้ใช้งาน แต่ถ้าเกิดขึ้นช้าก็จะกระทบกับความสามารถในการผลิต โดยเฉพาะทีมพัฒนา

(3) กระบวนการนี้ควรเน้นที่ความยืดหยุ่น และความสามารถในการขยายตัวมากกว่าคุณภาพที่สูง ดังคำกล่าวที่ว่า “*เราควรให้ความสำคัญกับการพัฒนาให้เสร็จทันเวลาก่อนที่โอกาสทางการตลาดจะหมดไป มากกว่า ไปพัฒนาซอฟต์แวร์ที่ไม่มีจุดบกพร่องเลย*”