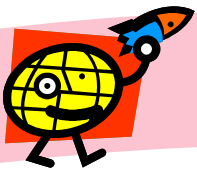


# Software Process

แนวคิดใหม่เกิดจากแนวคิดที่ว่าในระหว่างทำการพัฒนาระบบอยู่นั้น ความต้องการ ของซอฟต์แวร์มักจะปรับเปลี่ยนอยู่เสมอ ดังนั้นแบบจำลองแบบน้ำตกที่มองว่าการกำหนดความต้องการระบบเป็นจุดเริ่มต้นการทำงาน การพัฒนาซอฟต์แวร์ในขั้นตอนต่อไปอาจไม่สามารถทำได้ เนื่องจากความต้องการที่เปลี่ยนแปลงไป **อีกทั้งปัญหาที่เกิดจากการสื่อสารที่ไม่ครบถ้วนหรือปัจจัยภายนอก เช่นเทคโนโลยีที่เปลี่ยนไป ปัจจัยทางธุรกิจ**

ดังนั้นกระบวนการพัฒนาหรือวงจรชีวิตซอฟต์แวร์ที่ดีควรมีแนวคิดของการเปลี่ยนแปลงความต้องการ มาเป็นส่วนหนึ่งในการกำหนดขั้นตอนกระบวนการในการพัฒนาซอฟต์แวร์





# Agile

## Agile คืออะไร

หลักการทางวิศวกรรมซอฟต์แวร์ที่รวมเอาแนวทางพัฒนาเข้าด้วยกัน เพื่อมุ่งสร้างความพึงพอใจให้กับลูกค้าและสามารถส่งมอบซอฟต์แวร์แบบค่อยๆ เพิ่มขึ้นแก่ลูกค้า โดยใช้ทีมงานขนาดเล็กที่กระตือรือร้น ใช้วิธีการแบบไม่เป็นทางการผลิตงานด้านวิศวกรรมซอฟต์แวร์ถ้าจำเป็น และใช้วิธีการแบบเรียบง่าย ส่วนแนวทางการพัฒนาเน้น การส่งมอบมากกว่าการวิเคราะห์ออกแบบ และการสื่อสารอย่างต่อเนื่องกับลูกค้า

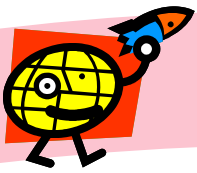
### ผลิตผล :

ซอฟต์แวร์รุ่นต่าง ๆ ที่ทำงานได้จริงโดยส่งมอบรุ่นที่เพิ่มขึ้นตามกำหนดเวลาที่ตกลงไว้

### ตรวจสอบ :

ทีมงานการเห็นพ้องต้องกันว่า กระบวนการทำมาแล้วได้ผล ผลิตซอฟต์แวร์รุ่นต่าง ๆ ส่งมอบได้เป็นที่พอใจของลูกค้า



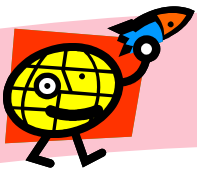


# หลักการของ Agile

## หลักการของ Agile บัญญัติเอาไว้ 12 ข้อชัดเจนดังนี้

1. เป้าหมายที่สำคัญที่สุดคือการสร้างความพึงพอใจให้กับลูกค้าด้วยการส่งมอบผลิตภัณฑ์คุณภาพอย่างรวดเร็วและต่อเนื่อง
2. ต้อนรับการเปลี่ยนแปลงความต้องการแม้ว่าจะอยู่ในช่วงปลายของการพัฒนา กระบวนการควบคุมการเปลี่ยนแปลงเพื่อให้ได้มาซึ่งข้อได้เปรียบทางการแข่งขันสำหรับลูกค้า
3. ส่งมอบซอฟต์แวร์ที่ทำงานได้อย่างต่อเนื่อง จากจำนวน 2-3 สัปดาห์ไปจนถึง 2-3 เดือน โดยมุ่งหวังจะลดช่วงเวลาเหล่านีลง
4. ผู้ที่ทำงานเชิงธุรกิจและทีมพัฒนาจะต้องทำงานด้วยกันเป็นรายวันตลอดโครงการ
5. สร้างความกระตือรือร้นให้กับบุคคล จัดสภาพแวดล้อมและการสนับสนุนที่ต้องการ และไว้วางใจว่าจะทำงานสำเร็จ
6. วิธีการที่มีประสิทธิภาพและประสิทธิผลสูงสุด คือ การแลกเปลี่ยนข้อมูลกันภายในทีม พัฒนาด้วยการสนทนาโดยตรง
7. ซอฟต์แวร์ที่ทำงานได้คือข้อมูลหลักที่ใช้บ่งบอกความก้าวหน้า





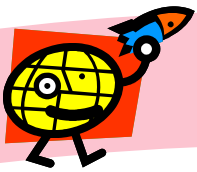
# หลักการของ Agile

## หลักการของ Agile (ต่อ)

8. กระบวนการส่งเสริมการพัฒนาที่ยั่งยืน ผู้สนับสนุน ผู้พัฒนา และผู้ใช้ควรสามารถดำรงไว้ซึ่งก้าวไปข้างหน้าอันสม่ำเสมอไปด้วยกันอย่างไม่มีการสิ้นสุด
9. การให้ความสนใจอย่างต่อเนื่องต่อความสามารถทางเทคนิคและการออกแบบที่ดีช่วยเพิ่มความคล่องตัว
10. ศิลปะของการเพิ่มปริมาณงานที่ไม่ต้องทำให้มากที่สุดเป็นสิ่งจำเป็น
11. สถาปัตยกรรม ความต้องการ และการออกแบบที่ดีที่สุด มาจากทีมที่บริหารตนเองได้
12. ในช่วงเวลาที่เหมาะสม ทีมจะสะท้อนว่าทำอย่างไรจึงจะมีประสิทธิภาพยิ่งขึ้น ต่อจากนั้นจึงปรับปรุงพฤติกรรมให้เหมาะสม

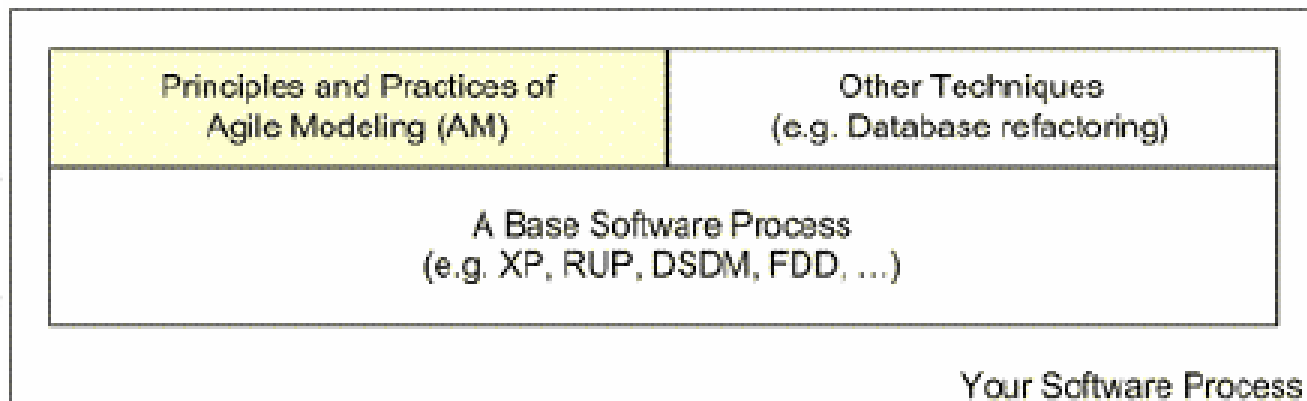
หมายเหตุ : การทำงานในขั้นแรก ก็อาจมีการส่งมอบของได้เป็น หน้าจอ, Prototype, infrastructure โดยขั้นแรกอาจมองว่า Progress เราเท่ากับ 0 เปอร์เซนต์ (เพราะยังไม่มีซอฟต์แวร์เกิดขึ้น)

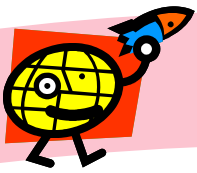




# โมเดลของ Agile

- ❑ เลือกบางหลักการมาทำ
- ❑ เป็นวิธีหนึ่งที่จะเอาหลักการของ Agile มาจัดการกับเอกสารและระบบเดิมที่มีอยู่ได้
- ❑ Agile ประกอบด้วย
  - 1) value ผลลัพธ์
  - 2) principle หลักการ
  - 3) practices วิธีปฏิบัติ
- ❑ ทั้งสามอย่างนี้เป็นส่วนหนึ่งในโมเดล Agile ที่สามารถนำมาพัฒนาซอฟต์แวร์ให้มีประสิทธิภาพและเกิด overhead น้อย
- ❑ ให้มอง Agile เป็นส่วนขยายของกระบวนการพัฒนาซอฟต์แวร์แบบเดิมได้
  - o นำ Agile เข้าไปกำกับ ดูว่าของเดิมที่มีอยู่อันไหนสำคัญก็ทำ ไม่สำคัญก็จะ
  - o นำ Agile มาจัดลำดับความสำคัญ ดูว่ากิจกรรมไหน ควรทำ ไม่ควร





# โมเดลของ Agile

## 1) Agile Model value

- o เน้นติดต่อสื่อสาร
- o เน้นความง่าย ไม่ซับซ้อน
- o เน้น feedback จากลูกค้า
- o เน้นความกล้าตัดสินใจ
- o เน้นความเคารพกันและกัน

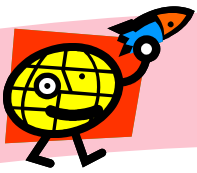
## 2) Agile Model Core principle

- o ง่าย ไม่เวอร์เกิน
- o รับ requirement พร้อมเปลี่ยนแปลงได้ตลอดเวลา
- o เน้นปัจจุบันเป็นหลัก
- o ทำ Model ตามความจำเป็นเท่านั้น
- o พยายามใช้ multiple model มองหลายๆมุมมอง
- o มีการตอบกลับเร็ว
- o SW ถือเป็นจุดมุ่งหมายหลัก
- o ให้แบกสัมภาระเบา

## AM Supplement Practices

- o ทำให้เป็นมาตรฐาน
- o ค่อยๆสร้างให้มีรูปแบบ เมื่อถึงเวลาค่อยใช้
- o โมเดลไม่ใช่ ให้โยนทิ้งไปเลย เพราะจะได้ไม่เสียเวลามากดูแล
- o เน้น contract (สัญญาระหว่างระบบที่สัมพันธ์กันอยู่) พยายามจัด contract ให้เป็นทางการ เช่น web service มี signature อะไรบ้างใน function call
- o การ update code เฉพาะตอนที่มีปัญหา





# โมเดลของ Agile

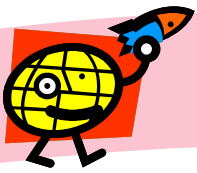
## 3) Agile Model Core Practice

1. จัดประชุม รวบรวม Active stakeholder เท่านั้น บางทีอาจมี none-stakeholder เข้ามาฟังได้ แต่ห้ามออกความคิดเห็น ห้ามถาม ห้ามติดต่อ ห้ามแสดงโอเคเดียว
2. นำ Artifact มาใช้ให้ถูกต้อง

**Note :** Artifact คือชิ้นส่วนของงานที่เราทำระหว่างการพัฒนา ระบบเช่น อีเมลล์, source code, จดหมาย, ใบเชิญประชุม ถ้า Artifact ใดถูกเลือกมาใช้ในการทำงาน เรียกว่า "work products" และถ้า work products นี้ ถูกส่งมอบให้ลูกค้าเรียกว่า "Deliverable"

3. พยายามให้ทุกคนเป็นเจ้าของงาน สามารถทำงานแทนกันและกันได้
4. พยายามใช้โมเดลแบบคู่ขนาน จะได้มองต่างมุม เพื่อเก็บรายละเอียดของระบบให้ครบ
5. ทำให้เนื้อหาง่าย
6. พยายามวาดรูปไม่ให้ซับซ้อน
7. พยายามให้โมเดลเข้าถึงได้ทุกคน
8. สามารถเปลี่ยน Artifact หนึ่งไปอีกอันได้
9. ใช้โมเดลแบบเล็กก่อนค่อยขยาย
10. พยายามให้ผู้อื่นมีส่วนร่วมในการทำโมเดล
11. พิสูจน์ด้วยการลงเขียน code ดู (จาก code เริ่มต้นตั้งแต่แรก)
12. ใช้เครื่องมือต่างๆในการทำงาน เช่น กระดาษ, กระดานดำ





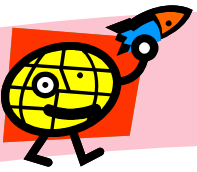
# เทคนิคการพัฒนาแบบ Agile

- ❑ Agile model driven development (AMDD)
- ❑ Code Refactor : เป็นการ redesign code คือให้แก้ code เพื่อให้มีโครงสร้างที่เข้าใจได้ง่ายขึ้น และสมบูรณ์ขึ้น
- ❑ Pair Programming : จับทีมทำงานเป็นคู่ 2 คนทำงานร่วมกัน ทำที่เดียวกัน ให้เครื่องเดียว 2 คน,แชร์กันใช้,คนนึงทำ-คนนึงดู (มีการตรวจสอบกันไปด้วย)
- ❑ Test Driven Development(TDD) : เป็นเทคนิคในการเขียน Test case เขียน test case ก่อนและค่อยทำการ implement code

## ตัวอย่างการเขียน

| Test case No. ▼ | Desc.         | Inputs  | Expected Outputs | Actual Outputs | Remark |
|-----------------|---------------|---------|------------------|----------------|--------|
| 1.              | ชื่อ pathname | A=5,B=2 | X=5              | X=-5           | ✗      |
| 2.              | ชื่อ pathname | C=8     | X=2              | X=2            | ✓      |





# Methodologies Agile

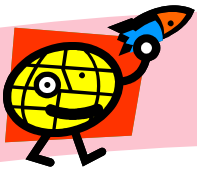
1) Agile UP (Unified Process)

2) XP (eXtream Programming)

3) FDD (Feature Driven Development)

4) Scrum



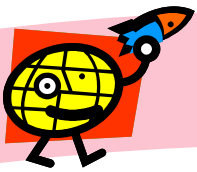


# Methodologies Agile

## 1) Agile UP (Unified Process)

- เป็นกระบวนการความพยายามที่จะดึงเอาลักษณะที่ดีที่สุดของกระบวนการดั้งเดิมออกมา แต่ปรับใช้ในลักษณะ Implement
- วิธีการเชิงวัตถุและภาษาเชิงวัตถุได้รับความนิยมในหมู่วิศวกรซอฟต์แวร์ และนำแบบจำลองและการพัฒนาระบบเชิงวัตถุที่เรียกว่า **UML** (Unified Modeling Languages)
- ให้ความสำคัญกับการสื่อสารกับลูกค้า และ Use case มาสร้างระบบ เพื่อให้มองเห็นเป้าหมายที่ถูกต้อง
- สามารถทำความเข้าใจ ปรับเปลี่ยนได้ในอนาคต และนำกลับมาใช้ใหม่ได้
- กระแสของกระบวนการเป็นแบบทำวนซ้ำ และค่อยเพิ่มขึ้น อันเป็นลักษณะของเชิงวิวัฒน์ที่พบในการพัฒนาซอฟต์แวร์สมัยใหม่





# Methodologies Agile

## 1) Agile UP (Unified Process)

กิจกรรมรอบงานประกอบด้วย 5 เฟส และโยงเข้ากับกิจกรรมทั่วไป

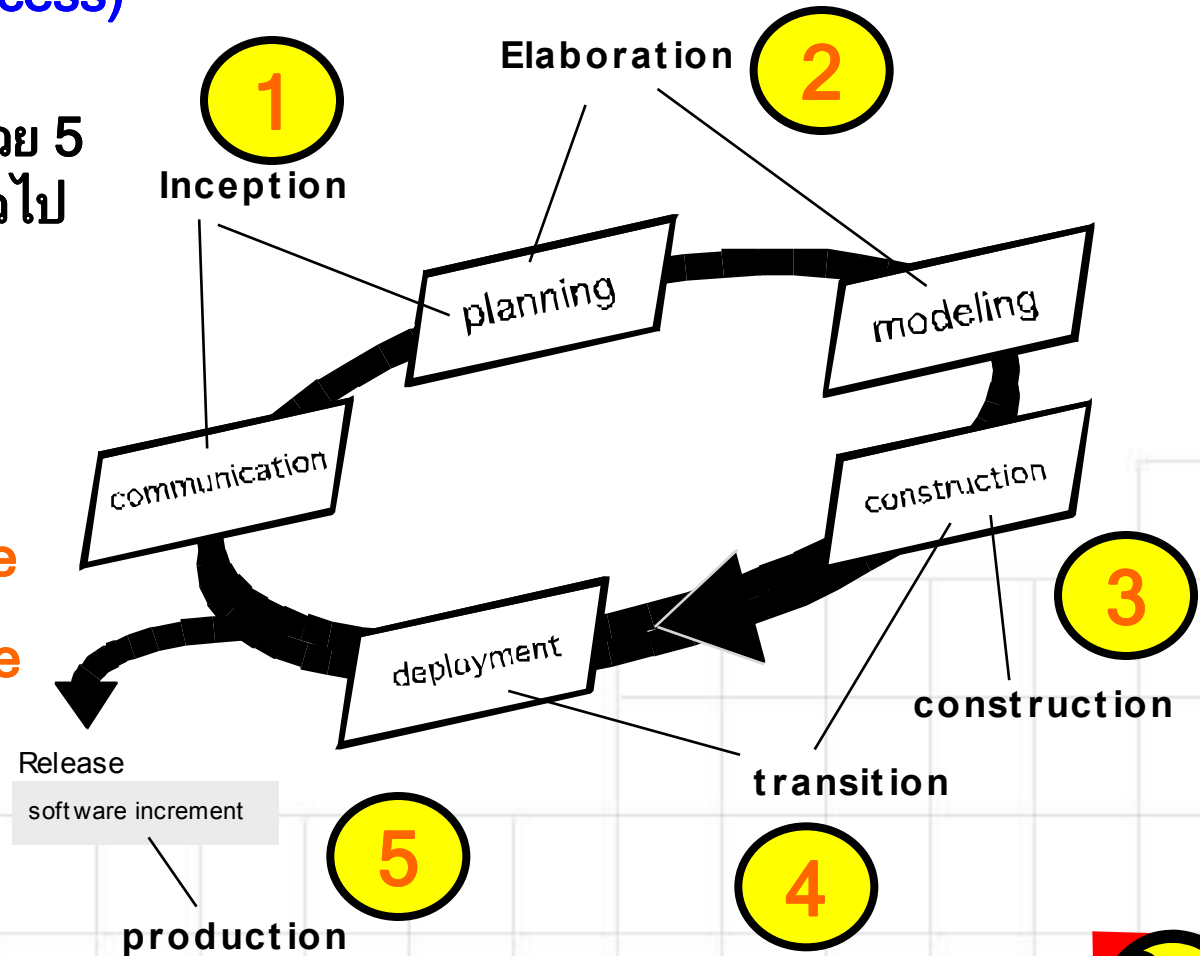
(1) Inception Phase

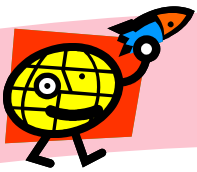
(2) Elaboration Phase

(3) Construction Phase

(4) The Transition Phase

(5) The Production Phase





# Methodologies Agile

## 1) Agile UP (Unified Process)

### o ระยะเริ่มต้น (Inception Phase)

- การสื่อสารกับลูกค้า และกิจกรรมการวางแผนงาน ซึ่งการพูดคุยจะทำให้สามารถ

(1) ระบุความต้องการทางธุรกิจ

(2) นำเสนอสถาปัตยกรรมอย่างคร่าว ๆ

(3) วางแผนโครงการในลักษณะทำวนซ้ำ หรือ ค่อยเพิ่มขึ้นได้

- การนำเสนอด้วย Use Case รุ่นแรก เหมือนสถาปัตยกรรมที่บ่งบอก

**Actor** (1) บุคคล เครื่องจักร หรือระบบอื่น ๆ ที่เข้ามามีปฏิสัมพันธ์กับระบบ

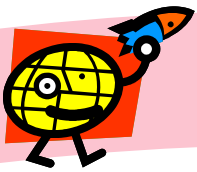
**Use Case** (2) หน้าที่ ที่ระบบต้องทำ เป็นสิ่งที่อยู่ในระบบ สามารถถูกเพิ่มเติมในภายหลัง

**Boundary** (3) ขอบเขตระหว่างระบบกับผู้กระทำต่อระบบ

**Relationship** (4) ความสัมพันธ์ของวัตถุ กิจกรรมที่เกิดขึ้นในระบบ

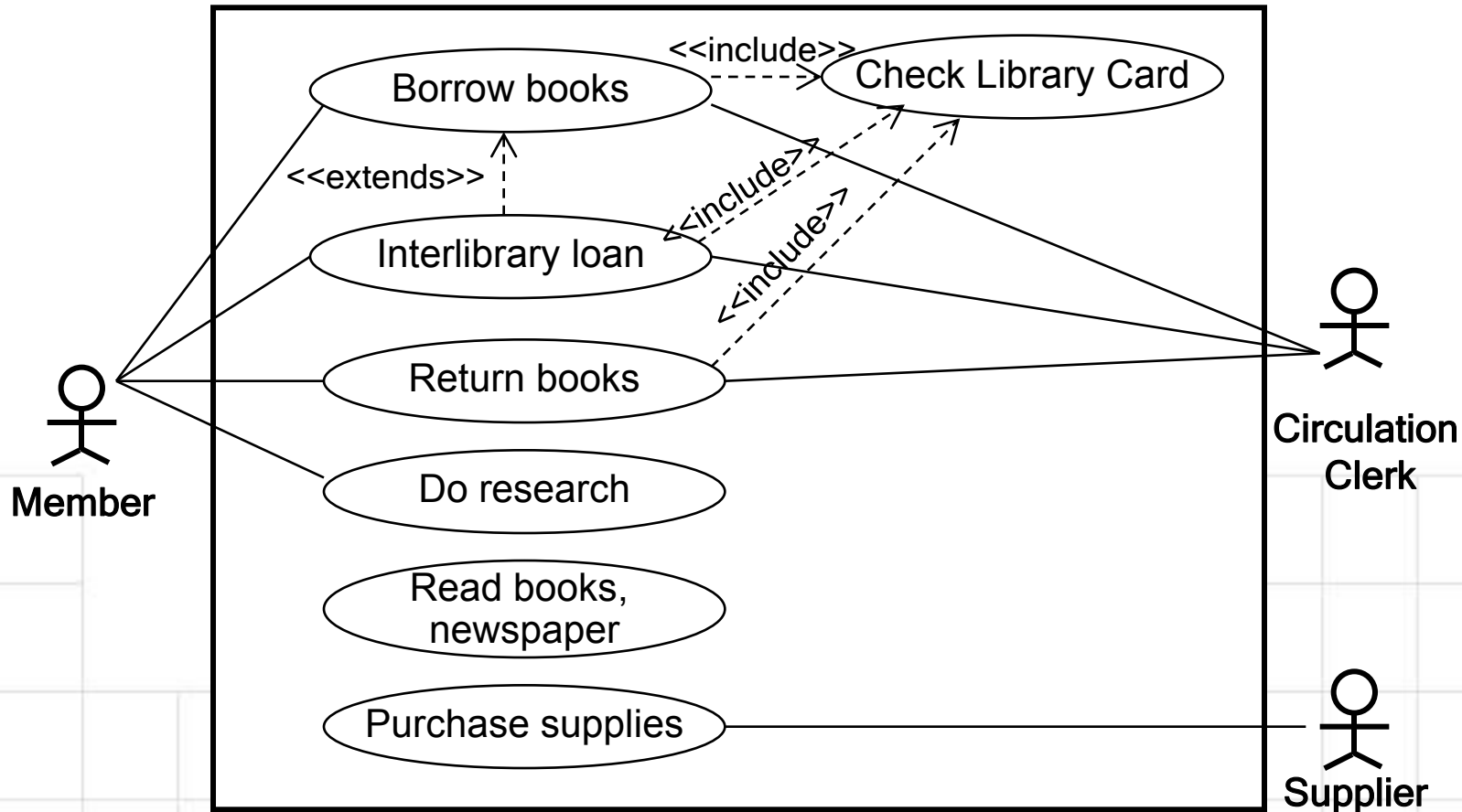
- การวางแผน ต้องระบุทรัพยากรที่ต้องใช้ ประเมินความเสี่ยงหลัก ๆ กำหนด  
ตารางเวลา และสร้างพื้นฐานสำหรับเฟสต่าง ๆ พัฒนาแบบค่อยเพิ่มขึ้น

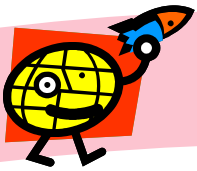




# Methodologies Agile

## 1) Agile UP (Unified Process) o Inception Phase : Use case Library





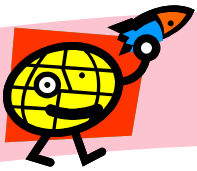
# Methodologies Agile

## 1) Agile UP (Unified Process)

### o ระยะขยายรายละเอียด (Elaboration Phase)

- ประกอบด้วยการสื่อสารกับลูกค้า และกิจกรรมสร้างแบบจำลอง การขยายรายละเอียด
- ปรับปรุง Use case ในระยะเบื้องต้น ให้ครอบคลุมสถาปัตยกรรมทั้ง 5 ของซอฟต์แวร์ ได้แก่
  - (1) แบบจำลอง Use case
  - (2) แบบจำลองการวิเคราะห์
  - (3) แบบจำลองการออกแบบ
  - (4) แบบจำลองการพัฒนา
  - (5) แบบจำลองการนำไปใช้งาน
- ทบทวนการสร้างแผนงานอย่างละเอียดในตอนท้าย เพื่อให้มั่นใจในขอบเขตงาน ความเสี่ยง และกำหนดวันส่งมอบ ว่ายังทำได้ตามกำหนด





# Methodologies Agile

## 1) Agile UP (Unified Process)

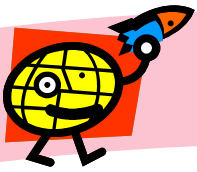
### o ระยะเวลาสร้าง (Construction Phase)

- มีการพัฒนาหรือจัดหาคอมโพเนนท์ที่ทำงานตาม Use case
- แบบจำลองการวิเคราะห์และออกแบบจะเสร็จสมบูรณ์ในเฟสนี้
- มีการพัฒนาลักษณะและหน้าที่ของซอฟต์แวร์ด้วย Source code
- มีการทดสอบระดับหน่วยสำหรับแต่ละคอมโพเนนท์
- และอาจมีการประกอบคอมโพเนนท์และทดสอบการเชื่อมต่อกัน
- การทดสอบมักสร้างชุดทดสอบ เพื่อให้เกิดการยอมรับก่อนถึงเฟสระดับถัดไป

### o ระยะเวลาส่งมอบ (The Transitive Phase)

- เป็นระยะท้าย ๆ ของกิจกรรมการก่อสร้าง และระยะต้นของกิจกรรมการใช้งาน
- ผู้ใช้งานจะได้รับซอฟต์แวร์เพื่อทดสอบ และรายงานจุดบกพร่อง และส่วนที่จำเป็นต้องเปลี่ยนแปลง
- ทีมงานจะสร้างข้อมูลสนับสนุนที่จำเป็น ได้แก่ คู่มือการใช้งาน คู่มือการติดตั้ง ก่อนซอฟต์แวร์จะถูกส่งมอบ หรือวางตลาด





# Methodologies Agile

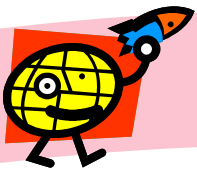
## 1) Agile UP (Unified Process)

### o ะยะการทำงาน (The Production Phase)

- กิจกรรมการใช้งานทั่วไป มีการเฝ้าดูการใช้งาน การสนับสนุนช่วยเหลือ และการรายงานจุดบกพร่อง

เฟสก่อสร้าง เฟสการส่งมอบ และเฟสการทำงาน อาจเกิดขึ้นได้ในเวลาเดียวกัน เนื่องจากการเริ่มงานสำหรับซอฟต์แวร์รุ่นถัดไป เพราะกระบวนการนี้ไม่ได้เรียงลำดับขั้น แต่เกิดขึ้นพร้อมกันเป็นขั้นซ้อน ๆ กัน

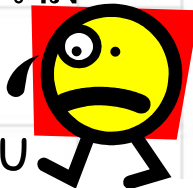


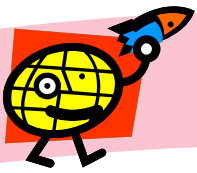


# Methodologies Agile

## 2) XP (eXtream Programming)

- o แนวคิดและวิธีการเอ็กซ์ทรีมโปรแกรมมิ่งเกิดขึ้นระหว่างปลายทศวรรษที่ 1980
- o ใช้แนวทางเชิงวัตถุในการพัฒนาระบบ โดยมีชุดของกฎและข้อปฏิบัติที่ต้องทำระหว่างกิจกรรมกรอบงาน 4 อย่าง (หัวใจหลัก) คือ
  - (1) การวางแผน (Planning)
  - (2) การออกแบบ (Design)
  - (3) การเขียนโค้ด (Coding)
  - (4) การทดสอบ (Test)
- o ปัจจัยพื้นฐาน
  - communication : เน้นเรื่องการพบปะพูดคุย (หลักการ Agile)
  - Simplicity : ออกแบบและเขียนโปรแกรมให้ง่าย ไม่เน้น performance มากนัก เน้นเรื่องแก้ไขให้ถูกต้อง
  - Feedback : เน้นเรื่องลูกค้า feedback เราเปลี่ยนได้เรื่อยๆ โดยใช้ refactor
  - Courage : เราต้องสามารถตัดสินใจเองได้ โปรแกรมเมอร์มีความกล้าในการตัดสินใจ





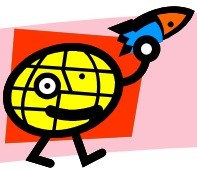
# Methodologies Agile

## 2) XP (eXtream Programming)

### (1) การวางแผน (Planning)

- ทีมพัฒนาจะเป็นผู้ทำหน้าที่เก็บรวบรวมข้อมูลความต้องการและวิเคราะห์ความต้องการของผู้ใช้เอง แล้วจัดทำเป็น User Stories
- โดยแต่ละเรื่องเล่าจะถูกจัดลำดับความสำคัญ และถูกนำมาพิจารณาว่าแต่ละเรื่องต้องใช้ระยะเวลาและต้นทุนเท่าใด
- จากนั้นทีมพัฒนาจะให้ผู้เลือกใช้เรื่องเล่าที่ต้องการให้พัฒนามากที่สุด
- จัดเป็นข้อดี เนื่องจากเป็นการเลือกโดยผู้ใช้ภายใต้ระยะเวลาที่ผู้ใช้อยอมต้องเป็นผู้กำหนดเอง ทำให้สามารถตัดทอนส่วนที่ไม่จำเป็นลงไปได้
- จากนั้นทีมงานนำมาวิเคราะห์ต้นทุนและผลตอบแทน (Cost-benefit Analysis) ตามระยะเวลาที่กำหนด พร้อมทั้งจัดแบ่งกิจกรรมต่าง ๆ ออกเป็นรอบตามจำนวนรอบที่เหมาะสม





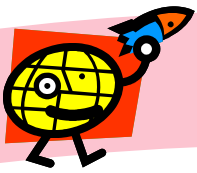
# Methodologies Agile

## 2) XP (eXtream Programming)

### (2) การออกแบบ (Design)

- หลักการออกแบบจะยึดหลัก KIS (Keep it Simple) คือ ทำให้ง่ายที่สุด แม้ว่าระบบจะซับซ้อนมากก็ตาม โดยการจัดทำต้นแบบ
- นอกจากนี้ การออกแบบยังได้กำหนดรายละเอียดที่เอื้อประโยชน์ในการเขียนโปรแกรมอีกด้วย โดยเพิ่มฟังก์ชันที่คาดว่าผู้ใช้ต้องการไว้ให้
- ไม่สนับสนุนให้มีการออกแบบหน้าที่ยื่น ๆ นอกเหนือจากเรื่องเล่า
- ถ้ามีปัญหาในการออกแบบที่ยากลำบากสำหรับเรื่องเล่าใด แนะนำให้สร้างต้นแบบที่ทำงานได้จริง เรียกว่า คำตอบสไปค์ (Spike Solution) เพื่อลดความเสี่ยงเมื่อมีการพัฒนาจริง





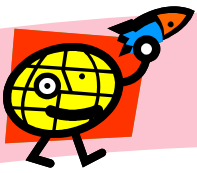
# Methodologies Agile

## 2) XP (eXtream Programming)

### (3) การเขียนโค้ด (Coding)

- เมื่อออกแบบขั้นต้นแล้ว ไม่ควรเริ่มเขียนโค้ดในทันที แต่ควรพัฒนาชุดทดสอบระดับหน่วย ที่จะทำงานกับเรื่องเล่าแต่ละเรื่องที่สร้างขึ้นก่อน จึงเริ่มเขียนโค้ดให้ทำงานผ่านการทดสอบ (ควรทดสอบโดยทันทีเมื่อเขียนโค้ดเสร็จ)
- แนวคิดที่สำคัญในการทำงานแบบเอ็กซ์พี เรียกว่า โปรแกรมเป็นคู่ (Pair Programming) คือการแนะนำให้คนสองคนทำงานร่วมกัน สำหรับเรื่องเล่าเรื่องหนึ่ง เครื่องคอมพิวเตอร์เดียวกัน เป็นกลไกที่ช่วยกันแก้ปัญหาและควบคุมคุณภาพเฉพาะหน้า
- เมื่อคู่โปรแกรมเมอร์ทำงานเสร็จ โค้ดจะถูกรวมเข้ากับโค้ดของโปรแกรมอื่น ซึ่งบางครั้งเกิดขึ้นเป็นประจำ เรียกว่า รวมกันอย่างต่อเนื่อง (Continuous Integration) เป็นกลยุทธ์ที่ช่วยป้องกันการ ทำงานเข้ากันไม่ได้ของแต่ละส่วน



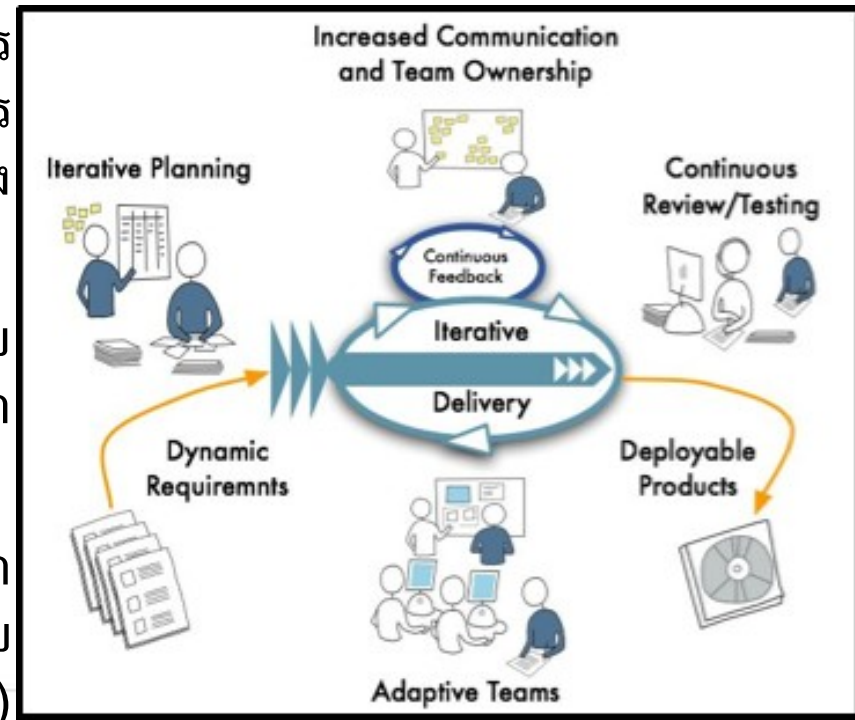


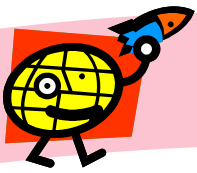
# Methodologies Agile

## 2) XP (eXtream Programming)

### (4) การทดสอบ (Test)

- การทดสอบแบบ Unit Test โดยมีการสร้าง Unit Test Case ไว้ก่อนการเขียนโปรแกรมภายใต้กรอบการสร้างงานทดสอบ
- ทำให้สามารถทดสอบโปรแกรมได้โดยอัตโนมัติ และทำให้ง่ายต่อการทดสอบซ้ำเมื่อต้องแก้ไขโปรแกรม
- จากนั้น จะนำไปให้ลูกค้าทดสอบ เรียกว่า Customer Test ซึ่งก็คือ การทดสอบการยอมรับ (Acceptance Test) นั้นเอง



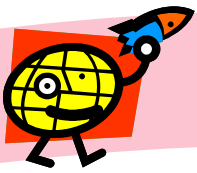


# Methodologies Agile

## 3) FDD (Feature Driven Development)

- o เริ่มจากการค้นค้นของ Peter Cord และคณะ ในปี 1999 จากหนังสือ “Java model in color with UML” เพื่อใช้เป็นกระบวนการเชิงปฏิบัติของวิศวกรรมซอฟต์แวร์เชิงวัตถุ
- o จากนั้นจึงมีขยายและเพิ่มงานที่สามารถประยุกต์ใช้กับโครงการขนาดกลางและขนาดใหญ่ขึ้น
- o ปกติเวลาเริ่มต้นโครงการเราต้องกำหนด stakeholder ว่ามีใครบ้าง จากนั้นก็เก็บ business goals (BG) กับ stakeholder เหล่านั้น ซึ่ง BG ปัญหาและเป้าหมายของเขา ซึ่งทางทีมพัฒนาฯ ต้องทำความเข้าใจ
- o และที่สำคัญควรมีอีกทีมที่ทำหน้าที่ทำความเข้าใจและตีความ BG เหล่านั้น เพราะหลายครั้งมักพบว่าทีมพัฒนาฯ ขาดทักษะด้านธุรกิจหรือไม่พอทำให้ตีความผิดและเข้าใจคลาดเคลื่อน สมัยนี้หลายองค์กรจึงมีคนมารับบทบาท เช่น solution architect, business IT strategist, ฝ่ายการตลาด เป็นต้น เพื่อทำความเข้าใจกับ BG เหล่านั้น



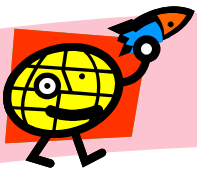


# Methodologies Agile

## 3) FDD (Feature Driven Development)

- o แล้วมาวิเคราะห์โอกาส (และอีกหลายอย่าง) เพื่อหาว่าควรต้องมีระบบฯ หรือ แอปพลิเคชันใดบ้าง และควรมีความสามารถ (Feature) ใดบ้าง
- o เมื่อระบุ feature เรียบร้อยแล้วก็เสนอลูกค้าหรือผู้บริหารครับ เมื่อตกลงยืนยันกันตามนี้ feature ก็ถือเป็น commitment ระหว่างทีมพัฒนาฯ และผู้บริหาร (หรือลูกค้า) การบริหารโครงการก็ต้อง base on feature
- o จากนั้นก็เริ่มเก็บ requirements กัน ไม่ว่าจะเป็นด้าน functional requirements, non-functional requirements ซึ่งแต่ละความต้องการต้องระบุด้วยว่าสัมพันธ์กับ feature ใด (requirements traceability) โดย software artifact ทุกประเภทต้องระบุว่าสัมพันธ์กับความต้องการใดและ feature ใด ไม่ว่าจะเป็นโมเดล diagram เอกสาร ซอร์สโค้ด test case ฯ ต้องระบุให้หมด เพราะทุกอย่างต้องสอดคล้องกับ feature ซึ่งในขณะเดียวกัน feature เองก็ต้องระบุว่าสัมพันธ์กับ BG ใดบ้าง เพื่อประโยชน์ต่อการตรวจสอบย้อนกลับหรือไปข้างหน้า (Traceability)



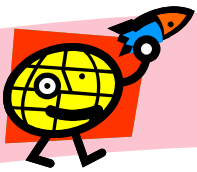


# Methodologies Agile

## 3) FDD (Feature Driven Development)

- o คุณลักษณะ “หน้าที่ที่ลูกค้าเห็นว่ามีคุณค่า ที่สามารถพัฒนาได้ภายในเวลา 2 - 4 สัปดาห์” ดังนี้
  - (1) เนื่องจากคุณลักษณะเป็นส่วนเล็ก ๆ ของซอฟต์แวร์ที่ทำงานได้ จึงสามารถอธิบายและเข้าใจความสัมพันธ์ระหว่างกันได้ง่ายกว่า และสามารถทบทวนได้ดีกว่า หากคลุมเครือข้อผิดพลาด หรือหลงลืม
  - (2) คุณลักษณะอาจถูกจัดระเบียบ เป็นกลุ่มลำดับชั้นที่มีความสัมพันธ์ทางธุรกิจ
  - (3) คุณลักษณะซอฟต์แวร์ที่ต้องส่งมอบได้ในการพัฒนาแบบ FDD ทีมงานจะมุ่งพัฒนาซอฟต์แวร์ให้มีคุณลักษณะใหม่ ๆ ที่ทำงานได้ทุก ๆ 2 สัปดาห์
  - (4) คุณลักษณะมีขนาดเล็ก ตัวแบบและโค้ดจึงง่ายต่อการตรวจทานอย่างละเอียด
  - (5) การวางแผน จัดตารางงาน ติดตามการขับเคลื่อนด้วยคุณลักษณะตามลำดับ ซึ่งดีกว่าการใช้ชุดงานย่อยที่เลือกมาแบบสุ่ม



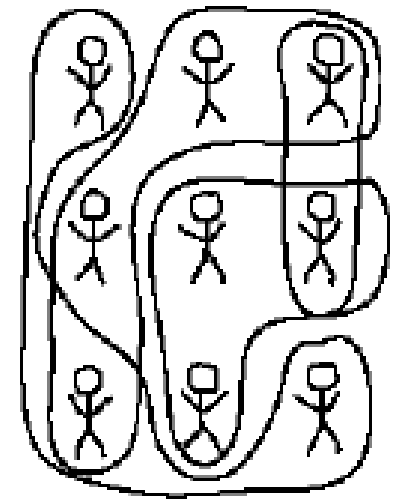
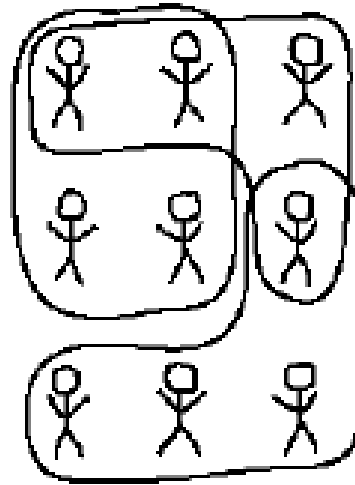


# Methodologies Agile

## 3) FDD (Feature Driven Development)

o บทบาทและหน้าที่ที่จำเป็นต้องมีเพื่อสนับสนุนให้เกิดความกระบวนกรที่ถูกต้อง

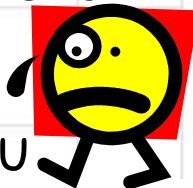
1. Domain Manager
2. Release Manager
3. Language Guru
4. Build Engineer
5. Toolsmith
6. System Administrator
7. Tester
8. Deployer
9. Technical Writer

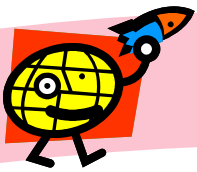


Each member of a feature team contributes to the design and implementation of a feature under the guidance of a skilled, experience developer. This reduces the risk of reliance on key developers or owners of specific classes.

Class Owners may find themselves members of multiple feature teams at the same time.

Chief Programmers are also Class Owners and take part in feature teams led by other Chief Programmers

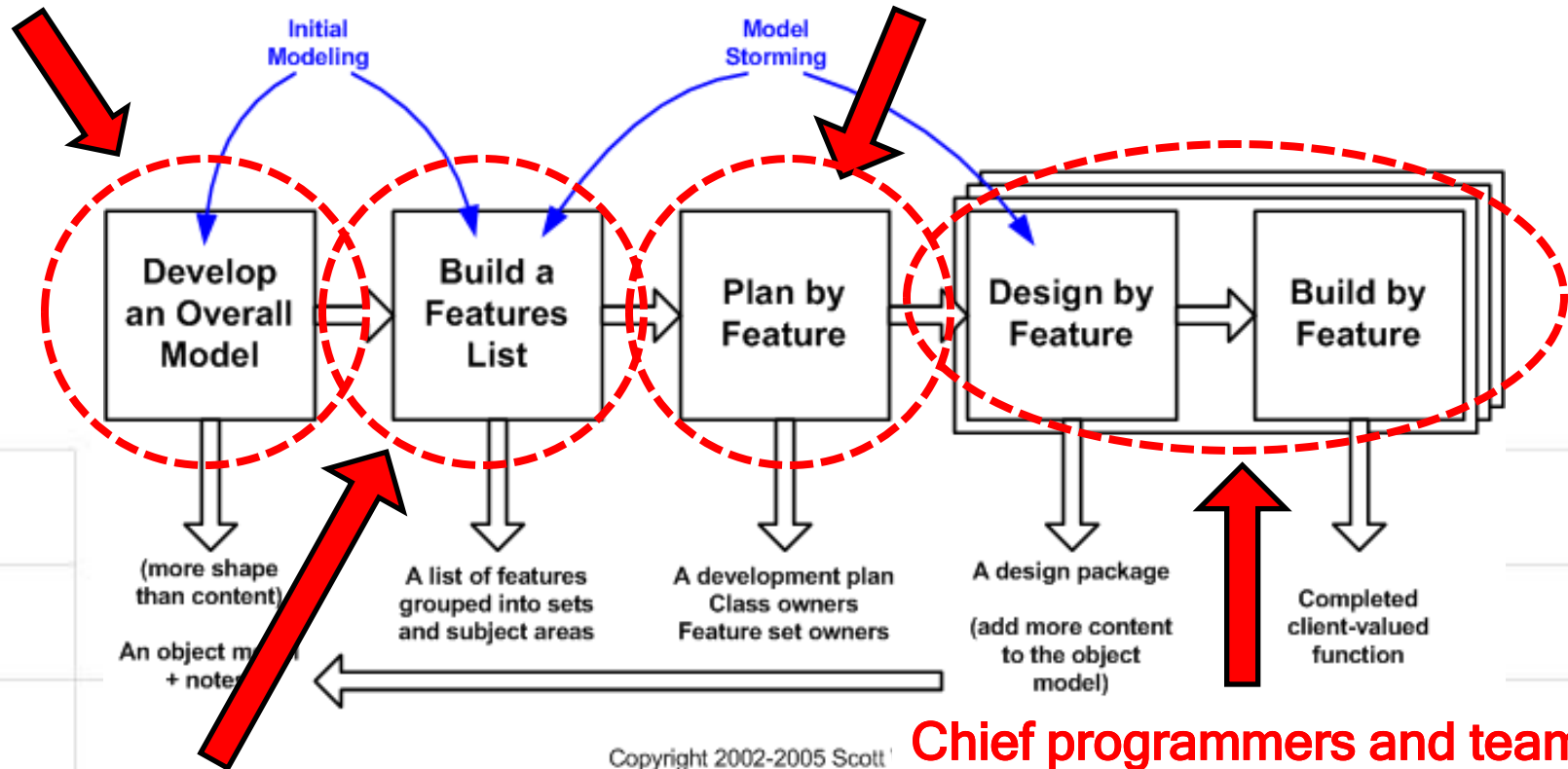




# Methodologies Agile

## FDD Process กระบวนการร่วมมือ 5 ระดับของ FDD

**Doma project Manager, Development Manager, and Chief Programmers**

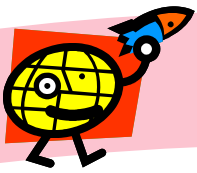


Copyright 2002-2005 Scott  
Copyright S. R. Palmer & J.M. Feising

**Chief programmers and team**

**Chief programmers and team**

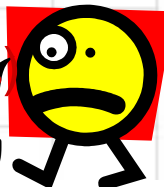


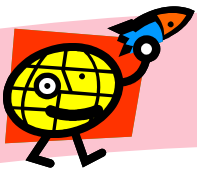


# Methodologies Agile

## 4) Scrum

- สครัมมาจากกิจกรรมในการแข่งขันรักบี้ เป็นกระบวนการเอาใจใส่ที่พัฒนาโดย Jeff Sutherland และทีมงาน เมื่อต้นทศวรรษ 1990
- หลักการของสครัมมีดังต่อไปนี้
  - (1) จัดตั้งทีมงานขนาดเล็กที่ “เกิดการสื่อสาร การแบ่งปันเทคนิค และข่าวสารที่ไม่เป็นทางการให้มากที่สุด ขณะที่ลดค่าใช้จ่ายส่วนเกินให้น้อยที่สุด”
  - (2) กระบวนการต้องสามารถปรับเข้ากับการเปลี่ยนแปลงทางธุรกิจและเทคนิคได้ เพื่อผลิตผลงานให้ดีที่สุด
  - (3) กระบวนการต้องผลิตรุ่นซอฟต์แวร์ออกมาบ่อย ๆ เพื่อตรวจสอบ ปรับแต่ง ทดสอบ บั๊กและต่อยอดได้
  - (4) งานที่พัฒนาและนักพัฒนาจะแบ่งออกเป็นแพ็คเกจหรือพาร์ติชันที่สุดและขึ้นแก่กันน้อยที่สุด
  - (5) มีการทดสอบและบั๊กเอกสารอย่างสม่ำเสมอขณะสร้างผลิตภัณฑ์
  - (6) กระบวนการสครัมจะต้อง “สามารถแจ้งว่า พัฒนาผลิตภัณฑ์เสร็จแล้ว” เมื่อใดก็ตามที่ต้องการ (มีการแข่งขันสูง/บ.ต้องการเงิน/ลค.ต้องการใช้งาน)



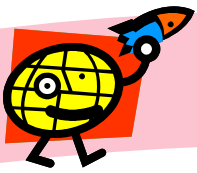


# Methodologies Agile

## 4) Scrum

- หลักการสครัมใช้นำทางกิจกรรมพัฒนา ภายใต้กระบวนการที่รวมเอากิจกรรมรอบงานต่อไปนี้ คือ ความต้องการ การวิเคราะห์ การออกแบบ การวิวัฒน์ และการส่งมอบ ที่พิสูจน์แล้วว่าได้รับผลดี
- สำหรับโครงการที่มีเวลาจำกัด มีการเปลี่ยนแปลงความต้องการ และมีความสำคัญอย่างยิ่งยวดต่อธุรกิจ แบบรูปกระบวนการแต่ละแบบนิยามชุดกิจกรรมพัฒนาระบบต่อไปนี้
  - **แบ็กล็อก (Backlog)** รายการความต้องการหรือลักษณะที่ให้คุณค่าทางธุรกิจแก่ลูกค้าที่เรียงลำดับความสำคัญแล้ว รายการอาจเพิ่มเข้ามาใหม่ได้ โดยผู้จัดการต้องประเมินและปรับปรุงลำดับความสำคัญตามความเหมาะสม
  - **สปรีน (Sprints)** ประกอบด้วยหน่วยของงาน ต้องทำให้เสร็จตามความต้องการที่นิยามโดยแบ็กล็อก โดยหน่วยของงานหนึ่ง ๆ ต้องทำให้เสร็จได้จริงตามกรอบเวลาที่กำหนดไว้แล้วล่วงหน้า (ปกติ 30 วัน) ระหว่างสปรีน รายการแบ็กล็อกใดที่กำหนดสปรีนอยู่จะถูกหยุดการเปลี่ยนแปลงไว้ก่อน



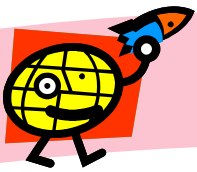


# Methodologies Agile

## 4) Scrum

- **การพบปะของสครัม (Scrum meeting)** การประชุมสั้น ๆ ประมาณ 15 นาที ของทีมสครัมจะมีทุก ๆ วัน เพื่อถามและตอบคำถามสามข้อ คือ
  - (1) คุณได้ทำอะไรไปแล้วหลังจากการประชุมครั้งที่แล้ว
  - (2) มีอุปสรรคอะไรหรือไม่ที่พบ
  - (3) คุณวางแผนจะทำอะไรให้เสร็จก่อนการประชุมคราวหน้าหัวหน้าทีมสครัม (Scrum Master) จะนำการประชุมและประเมินการตอบสนองของแต่ละบุคคล การประชุมสครัมช่วยให้ทีมค้นพบปัญหาเร็วที่สุดเท่าที่จะทำได้ นอกจากนี้ยังเป็นการส่งเสริมการเรียนรู้ และช่วยก่อให้เกิดโครงสร้างทีมจัดระเบียบตนเองได้ด้วย
- **สาธิต (Demos)** ส่งมอบรุ่นซอฟต์แวร์แก่ลูกค้าเพื่อสาธิตและประเมินหน้าที่การทำงานที่ได้พัฒนาแล้ว การสาธิตไม่จำเป็นต้องมีหน้าที่ครบถ้วนตามแผนที่วางไว้ แต่ต้องมีหน้าที่ที่ส่งมอบได้ภายในกรอบเวลาที่กำหนด





# Methodologies Agile

## 4) Scrum

### The Agile: Scrum Framework at a glance

Inputs from Executives,  
Team, Stakeholders,  
Customers, Users



Product Owner



The Team



Product  
Backlog

Team selects  
starting at top  
as much as it  
can commit  
to deliver by  
end of Sprint

Sprint  
Planning  
Meeting



Sprint  
Backlog



1-4 Week  
Sprint

Sprint end date and  
team deliverable  
do not change



Scrum  
Master



Burndown/up  
Charts

Every  
24 Hours



Daily Scrum  
Meeting



Sprint Review



Finished Work



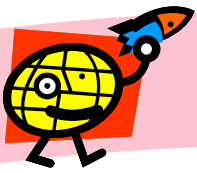
Sprint  
Retrospective

neon rain  
interactive [AGILE  
FOR ALL]

สกรัมทีมจะถูกปลูกจิตสำนึกว่า “We’re all in this together”

A typical Scrum team is 6-10 people but Jeff Sutherland has scaled Scrum up to over 500 people





# Methodologies Agile

## บทสรุป

Agile อาจเป็นเพียงวิธีการแบบหนึ่ง เป็นวิธีที่ฉีกแนวจากความคิดในการกระทำแบบเดิม ซึ่งมีข้อดีอยู่มาก แต่ยังมีหลายด้านที่เป็นข้อด้อย

Agile ถือเป็นเรื่องใหม่ ขณะนี้จะมองว่าเป็นไปตามสมัยนิยม (fashion) หรืออนาคตก็ได้ทั้งสองอย่าง แนวคิดอย่าง Agile กำลังเริ่มต้นเท่านั้น ยังรอคอยการลองผิดลองถูก รูปแบบแอปพลิเคชันในปัจจุบันก็แตกต่างกันมากขึ้นทุกวัน เครื่องมือการพัฒนาซอฟต์แวร์ก็พัฒนาไปมาก เพียงรูปแบบแอปพลิเคชันในอนาคตยังคาดการณ์ไม่ได้ วิธีการพัฒนาก็คงจะเป็นเช่นเดียวกัน และเป็นการยากที่จะบอกว่า Agile ดีกว่าแบบอื่น ๆ เพราะขึ้นอยู่กับสถานการณ์ที่จะนำไปประยุกต์ใช้และวัฒนธรรมขององค์กรเป็นสำคัญ

