Agile Software Development

Week 3

ดัดแปลงจาก slides ของหนังสือ Software Engineering [1]

หัวข้อที่จะศึกษา

- วิธีการแบบอไจล์ (Agile methods)
- เทคนิคอไจล์ (Agile development techniques)
- การบริการโครงการอไจล์ (Agile project management)
- วิธีการอไจล์แบบปรับสเกล (Scaling agile methods)

Rapid software development

- ความสำคัญของการพัฒนาและส่งมอบซอฟต์แวร์ที่รวดเร็ว
  - การพัฒนาทางธุรกิจ ทำให้ความต้องการซอฟต์แวร์ที่พัฒนาอย่างรวดเร็ว
  - การรอให้ requirement อยู่ตัว อาจไม่ทันต่อการแข่งขันทางธุรกิจ
  - ซอฟต์แวร์ที่ดี ต้องตอบสนองต่อความต้องการที่เปลี่ยนแปลงไปอย่างรวดเร็วและไม่หยุดยั้ง
  - นักพัฒนาซอฟต์แวร์ ต้องยอมรับความจริงในข้อนี้ และ เตรียมตัวให้พร้อม สำหรับการเปลี่ยนแปลงทุกรูปแบบ
  - กระบวนการพัฒนาซอฟต์แวร์ที่ไม่รองรับการเปลี่ยนแปลงอย่างว่องไว ย่อมทำให้ซอฟท์แวร์ไม่เป็นที่ต้องการของตลาด

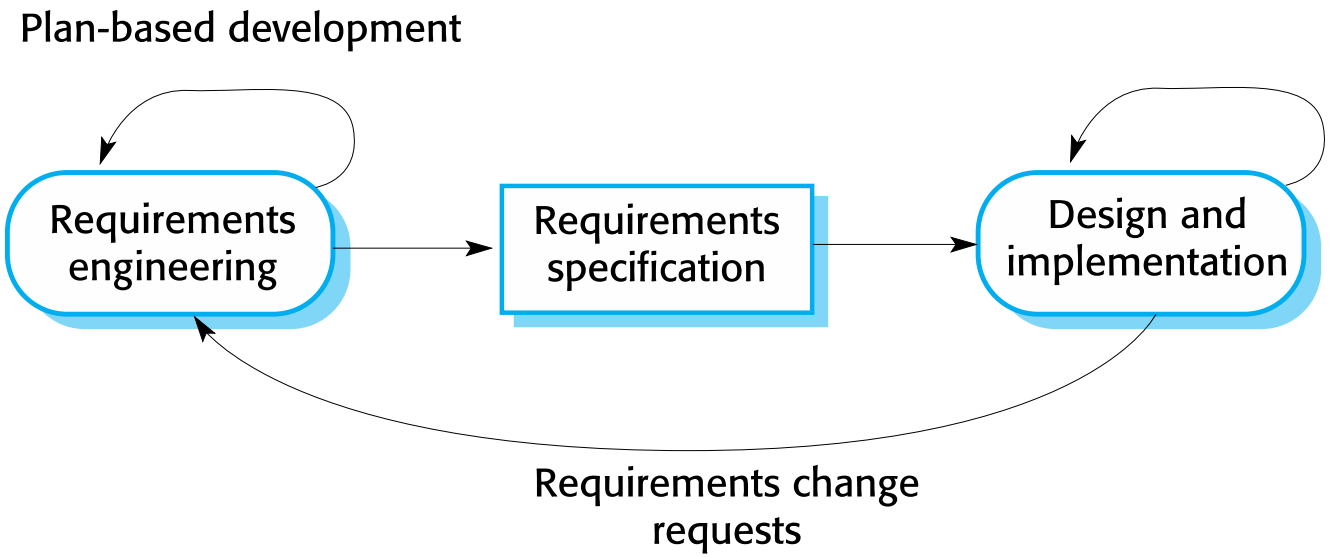
Rapid software development

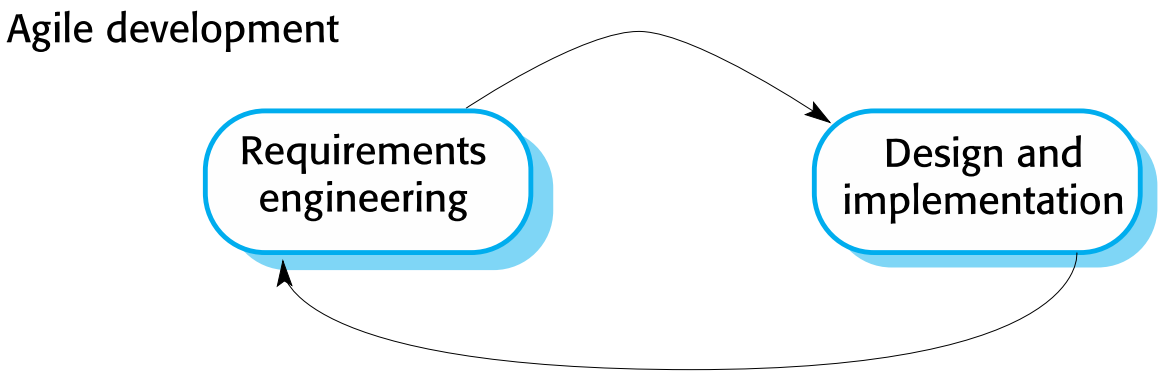
- การพัฒนาซอฟต์แวร์ ตามระเบียบแบบแผน (plan-driven) อาจช่วยในหลายเรื่อง แต่ไม่ยืดหยุ่นและไม่รองรับความต้องการที่เปลี่ยนแปลงอย่างรวดเร็ว
  - กว่าจะครบทุกขั้นตอนในการพัฒนา อาจจะใช้เวลาเป็นเดือนหรือปี ทำให้ไม่สามารถตอบสนองต่อความต้องการได้ทันท่วงที
- วิธีการพัฒนาแบบ **agile** กำเนิดขึ้นในยุค 1990
  - เพื่อลดระยะเวลาในการพัฒนาและส่งมอบซอฟต์แวร์ให้สั้นลง
  - วิธีการง่ายๆ คือ ปล่อยมาเป็นรุ่นย่อย ๆ (พัฒนาเป็น incremental)

ลักษณะเฉพาะของ Agile development

- การออกข้อกำหนดโปรแกรม (Program specification) การออกแบบและสร้าง (design and implementation) จะทำควบคู่กันไป
- การพัฒนาจะทำเป็นรุ่นๆ โดยปล่อยออกมาเป็น incremental
  - ในการพัฒนาแต่ละรุ่น จะมี stakeholders เข้ามาร่วมออก specification และทำการ evaluation
- ปล่อยรุ่นล่าสุดออกมา ให้บ่อยที่สุด เท่าที่จะเป็นไปได้
- ใช้เครื่องมือช่วยในการพัฒนาให้มากที่สุด (เช่น automated testing tools)
- ทำเอกสารน้อย ๆ
  - เน้นที่ working code (เขียนโค้ดให้มีความเป็น document ในตัว เช่นการตั้งชื่อตัวแปรที่สื่อความหมาย)

Plan-driven vs agile development





Plan-driven and agile development

- **Plan-driven development**
  - การพัฒนาซอฟต์แวร์แบบ plan-driven จะมีการแบ่ง development stages อย่างชัดเจน
  - outputs ที่ได้จากแต่ละ stages จะเป็นตัวขับเคลื่อน ให้การพัฒนาดำเนินต่อไปได้
  - ไม่ใช่เพียงแต่ใน waterfall model เท่านั้น incremental development เอง ก็ทำให้เกิดรูปแบบ plan-driven ได้เช่นกัน
- **Agile development**
  - Specification, design, implementation and testing จะของแต่ละรุ่นจะเกิดขึ้นขนานกันไป
  - outputs ของการพัฒนา จะอยู่ที่การคุยกันระหว่างทีมพัฒนาและ stakeholder

ทำไมต้อง Agile process

- เป็นการยาก ที่จะพยากรณ์ว่า ความต้องการไหนของซอฟต์แวร์ ที่จะคงอยู่ หรือเปลี่ยนแปลงไป
- เป็นการยาก ที่จะพยากรณ์ว่า ในขณะที่การพัฒนากำลังดำเนินไป user จะเปลี่ยนความต้องการที่จุดไหนบ้าง
- เป็นการยาก ที่จะบอกได้ว่าต้อง design มากแค่ไหน (ทั้งปริมาณและคุณภาพ) ถึงจะตอบโจทย์ requirement ได้อย่างเหมาะสม
  - บาง design จะให้ผลลัพธ์เมื่อสร้างเสร็จแล้วเท่านั้น
- เป็นการยากที่จะกำหนดกรอบเวลาหรือความสำเร็จของการ วิเคราะห์ ออกแบบ สร้าง และทดสอบซอฟต์แวร์ ตั้งแต่เริ่มโครงการ

Agile methods

- ข้อผิดพลาดในการพัฒนาซอฟต์แวร์ในยุค 80 และ 90 คือ การมี overhead มากเกินไป นำมาสู่การพัฒนาแบบ agile ซึ่งมีลักษณะเฉพาะคือ
  - เน้นที่การเขียน code มากกว่าที่จะเน้นที่การออกแบบ (design)
  - เน้นที่การพัฒนาซอฟต์แวร์แบบ iterative approach
- มุ่งเน้นในการส่งมอบซอฟต์แวร์ที่ใช้งานได้จริงและตรงตามความต้องการที่เปลี่ยนไปอย่างรวดเร็ว
- สิ่งที่เรียกว่า overhead ได้แก่ เอกสารจากขั้นตอนต่างๆ ที่เกิดขึ้นในกระบวนการพัฒนา
- เป้าหมายของ agile methods คือ
  - การลด overheads โดยการจำกัดปริมาณเอกสารที่ต้องทำ
  - มุ่งเน้นที่การตอบสนองต่อความต้องการของลูกค้ามากกว่าการทำงานที่ไม่จำเป็นและสิ้นเปลือง

คำประกาศของอไจล์ (Agile manifesto)

- **พัฒนาซอฟต์แวร์ด้วยความเอื้อเฟื้อซึ่งกันและกัน**

สิ่งที่ต้องให้ความสำคัญ	สิ่งที่ไม่จำเป็นต้องทำ
บุคคลและการปฏิสัมพันธ์	เครื่องมือและกระบวนการ
ซอฟต์แวร์ที่ใช้ได้จริง	เอกสาร
ความร่วมมือจากผู้ใช้	การต่อรอง
พร้อมที่จะเปลี่ยนแปลง	ทำตามแผน (อย่างเคร่งครัด)

The principles of agile methods (a)

Principle	Description
การมีส่วนร่วมจากผู้ใช้ (Customer involvement)	• Customers ควรมีส่วนร่วมอย่างใกล้ชิดตลอดระยะเวลาในการพัฒนา • หน้าที่ของ customer คือ การระบุและจัดลำดับความสำคัญของ requirements รวมทั้งต้องทำหน้าที่ทดสอบด้วย
การทยอยส่งมอบ (Incremental delivery)	• ซอฟต์แวร์จะต้องทยอยพัฒนาและส่งมอบเป็นรุ่น ๆ ตามข้อกำหนดของผู้ใช้
เน้นคน ไม่เน้นกระบวนการ (People not process)	• ดึงทักษะของคนในทีมมาใช้อย่างเต็มศักยภาพ • ปล่อยให้สมาชิกในทีมทำงานอย่างอิสระ ไม่ต้องหากระบวนการใด ๆ มากำหนดการทำงาน

The principles of agile methods (b)

Principle	Description
น้อมรับการเปลี่ยนแปลง (Embrace change)	• ให้ระลึกและตระหนักว่า system requirements จะมีการเปลี่ยนแปลงอยู่เสมอ • ออกแบบซอฟต์แวร์ให้ตอบโจทย์ต่อการเปลี่ยนแปลงที่จะเกิดขึ้น
รักษาความเรียบง่าย (Maintain simplicity)	• มุ่งเน้นความเรียบง่าย ทั้งตัวซอฟต์แวร์ที่จะพัฒนา และกระบวนการที่จะใช้ • ให้กำจัดความยุ่งยากซับซ้อนออกไปทันทีที่เป็นไปได้

ใช้งาน Agile method เมื่อใด?

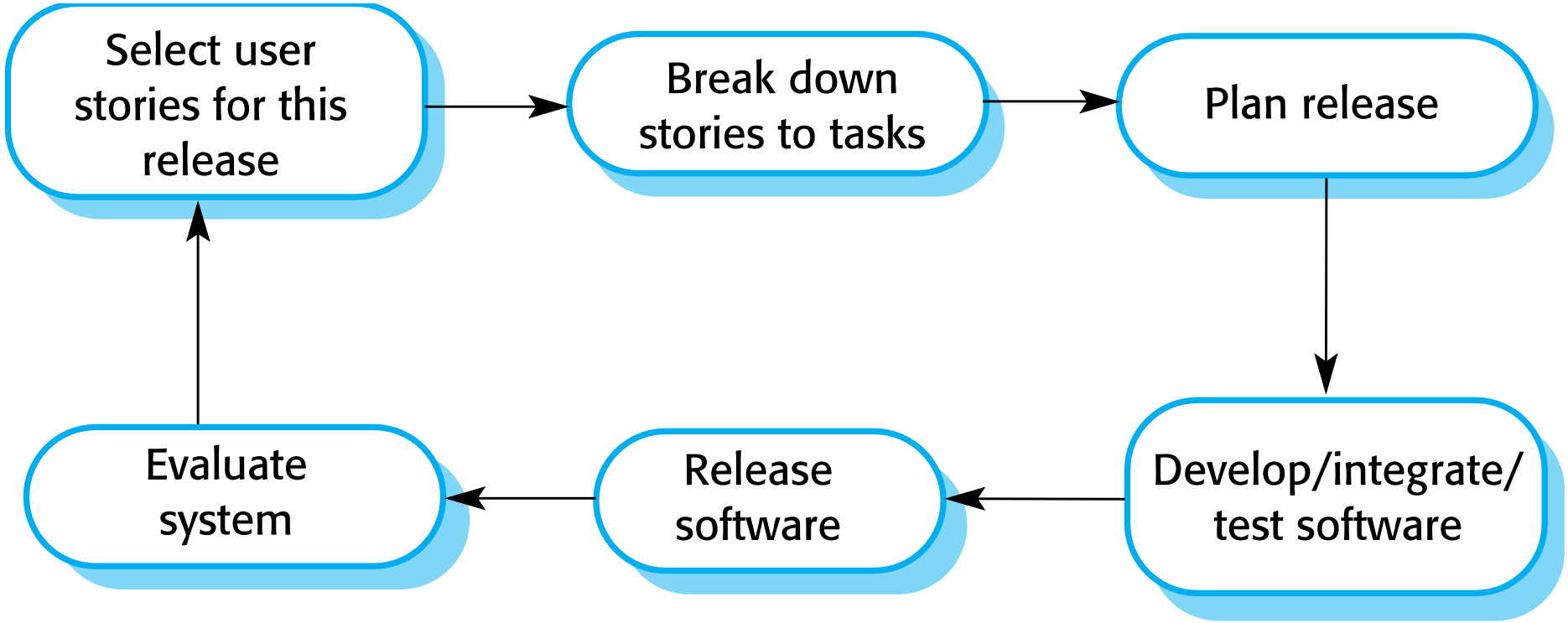
- ใช้ในการพัฒนาซอฟต์แวร์ขนาดเล็กถึงขนาดกลาง
  - แต่ดูเหมือนว่า ซอฟต์แวร์และแอพส์แทบทั้งหมดในปัจจุบัน จะพัฒนาโดยใช้ agile
- การพัฒนาซอฟต์แวร์ใช้งานภายในองค์กร
  - มี user ที่ตั้งใจจะเข้ามาร่วมทีมพัฒนา (ทำหน้าที่ตามคำประกาศของ agile)
  - ไม่มีการกำหนดกฎเกณฑ์จากภายนอก ที่จะกระทบต่อการพัฒนาซอฟต์แวร์นั้น หรือถ้ามีก็ให้น้อยที่สุด

Agile development techniques

Extreme programming (XP)

- การพัฒนาแบบ agile ได้รับความนิยมในช่วงปลายยุค 1990s และมีการคิดค้นรูปแบบออกมาอย่างมากมาย
- แต่มีรูปแบบหนึ่งที่สำคัญ คิดค้นโดย **Kent Beck** ถูกเรียกว่า Extreme Programming (XP) ใช้การพัฒนาแบบ iterative เหตุที่เรียก extreme เนื่องจาก
  - มีการออกรุ่นใหม่บ่อยมาก อาจจะหลายรุ่นในวันเดียว
  - แต่ละรุ่น จะส่งไปยังผู้ใช้ทุก ๆ 2 สัปดาห์
  - มีการทดสอบ (tests) กับทุก build
    - แต่ละ build จะถูกยอมรับ (accepted) ถ้าผ่านการรัน tests ได้อย่างสมบูรณ์เท่านั้น

The extreme programming release cycle



Extreme programming practices (a)

Principle or practice	Description
(การวางแผนแบบ Incremental) Incremental planning	• Requirements จะถูกเขียนลงบน story cards • Story ที่จะถูกเพิ่มใน release จะถูกเลือกจากระยะเวลาที่ต้องใช้และลำดับความสำคัญของ story • Developers จะกระจาย stories เหล่านั้นออกเป็น development 'Tasks'
ปล่อยรุ่นเล็ก ๆ (Small releases)	• เริ่มต้น จะพัฒนาส่วนเล็ก ๆ ที่มีผลตอบแทนสูงสุดก่อน • ปล่อยรุ่นถัดมาอย่างสม่ำเสมอและเพิ่มความสามารถจาก release แรก ๆ
ออกแบบให้ง่ายเข้าไว้ (Simple design)	• ออกแบบ (design) แค่พอให้ตอบสนองต่อ requirement ปัจจุบันเท่านั้น • อย่าเยอะ

Extreme programming practices (b)

Principle or practice	Description
พัฒนาแบบ ทดสอบก่อนเสมอ (Test-first development)	• ใช้ automated unit test framework เพื่อทดสอบฟังก์ชันใหม่ ๆ ที่พัฒนาขึ้น • เขียน test ก่อนเขียน code ที่จะใช้งานจริงเสมอ
Refactoring	• นักพัฒนาทุกคน ต้องทำ refactoring อย่างสม่ำเสมอ และทำทันทีที่พบว่าสามารถทำได้ • จะช่วยให้ทำความเข้าใจ code และบำรุงรักษาได้ง่าย
มีบัดดี้ในการเขียนโปรแกรม (Pair programming)	• ทำงานเป็นคู่ • ตรวจสอบซึ่งกันและกัน • ช่วยกันพัฒนาความสามารถของบัดดี้

Extreme programming practices (c)

Principle or practice	Description
มีความเป็นเจ้าของร่วมกัน (Collective ownership)	• นักพัฒนาทำงานเป็นคู่ ในทุกส่วนของระบบ ไม่มีใครเก่งกว่าใคร • ทุกคนในทีมมีความเป็นเจ้าของ code ร่วมกัน • ทุกคนสามารถแก้ไข เปลี่ยนแปลงได้ ทุกส่วนของระบบ
บูรณาการอย่างต่อเนื่อง (Continuous integration)	• ทันทีที่ work ตาม task เสร็จสมบูรณ์ จะถูกนำไปรวม (บูรณาการ) เข้ากับส่วนอื่น ๆ ของระบบ • หลังจากบูรณาการสำเร็จ จะต้องทำการทดสอบทั้งระบบจนผ่าน
ก้าวย่างที่ยั่งยืน (Sustainable pace)	ถึงจะมีความเร่งรีบในการพัฒนา แต่การทำงานมากจนเกินกำลัง ไม่ส่งผลดีนัก เนื่องจากจะทำให้ code ที่ได้มีความด้อยคุณภาพลงy

Extreme programming practices (d)

Principle or practice	Description
ลูกค้าร่วมเป็นทีมพัฒนา (On-site customer)	• ลูกค้าหรือ end-user ของระบบควรเข้ามาร่วมทีมอย่างเต็มเวลา (full time) กับ XP team. • ในกระบวนการพัฒนาแบบ extreme programming นั้น customer ถือเป็นบุคลากรคนหนึ่งในทีมพัฒนา มีหน้าที่นำ system requirements มาให้ทีมทำการสร้างซอฟต์แวร์