SOFTWARE ENGINEERING

Agile Software Development

Dr. Rathachai Chawuthai

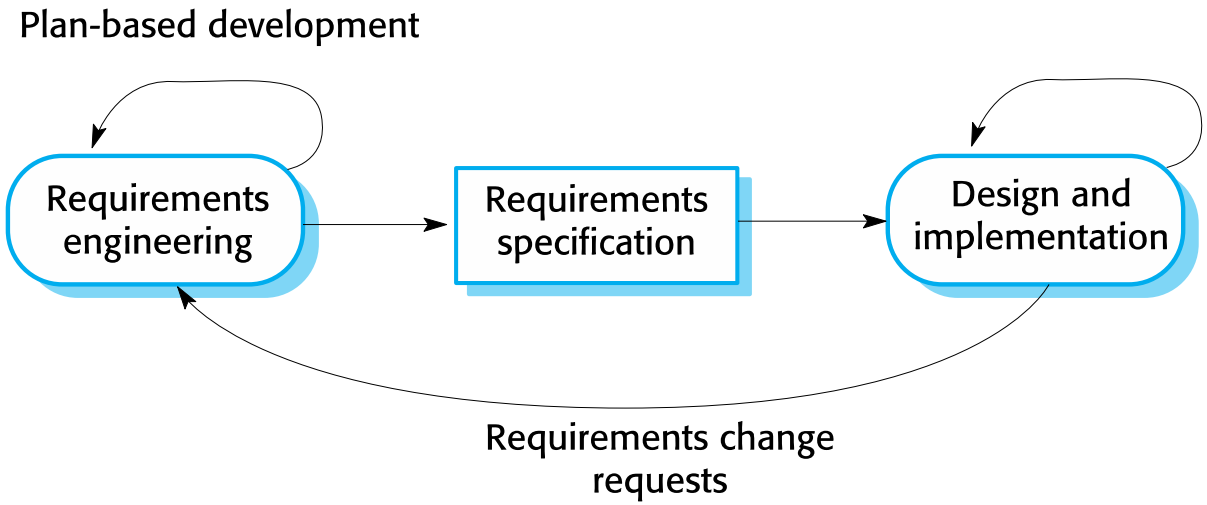
Department of Computer Engineering
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang

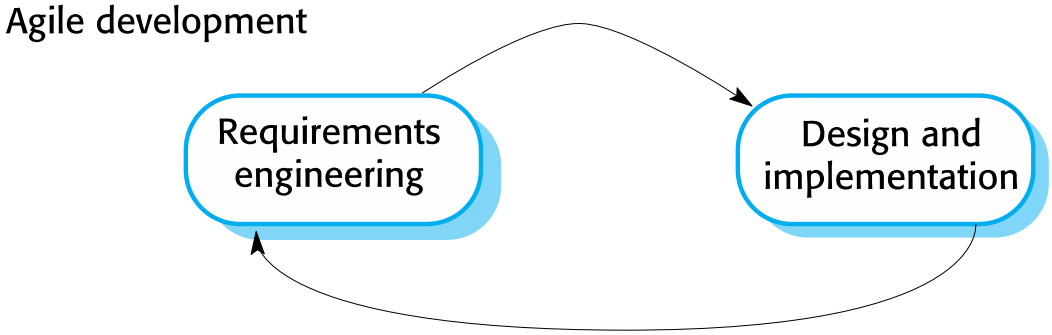
Agenda

- Agile Methods
- Agile Development Techniques
- Agile Project Management
- Scaling Agile Methods

Agile Methods

Plan-Driven vs. Agile Development





Ref: • (image) Ian Somerville; "Software Engineering", 10th Edition, Pearson (2016)

Agile

Ref: • (image) https://svsg.co/agile-for-smes/

Agile Methods

Focus on the code rather than the design

Are based on an iterative approach to software development

Are intended to deliver working software quickly and evolve this quickly to meet changing requirements.

Ref: • (image) Ian Somerville; “Software Engineering”, 10th Edition, Pearson (2016)

Agile Manifesto

Individuals and Interactions

over processes and tools

Working Software

over comprehensive documents

Responding to change

over following a plan

Ref: • (image) Ian Somerville; “Software Engineering”, 10th Edition, Pearson (2016)

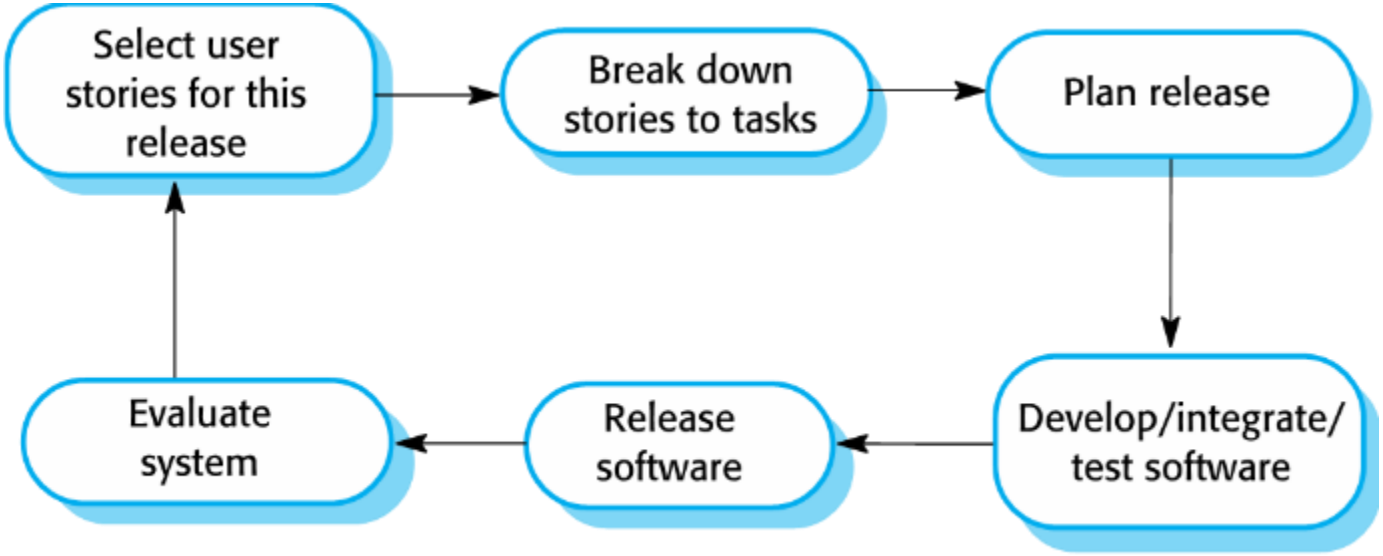
The Principles of Agile Methods

- Customer Involvement
  - ลูกค้าเข้ามาร่วมกระบวนการอย่างใกล้ชิด และช่วยโหวตเลือก requirement ในแต่ละ Iteration
- Incremental Delivery
  - ส่งมอบซอฟต์แวร์ให้ลูกค้าได้ลองใช้ feature ที่เสร็จแล้ว บ่อยๆ
- People not Process
  - เน้นพัฒนาทักษะด้าน development (รวมถึง communication) มากกว่าเรื่อง process
- Embrace Change
  - ออกแบบและพัฒนาระบบให้ยืดหยุ่นต่อการเปลี่ยนแปลงที่ลูกค้าต้องการ
- Maintain Simplicity
  - ทำการพัฒนาและกระบวนการให้เรียบง่าย

Agile Development Techniques

Extreme Programming (XP)

- The extreme programming release cycle



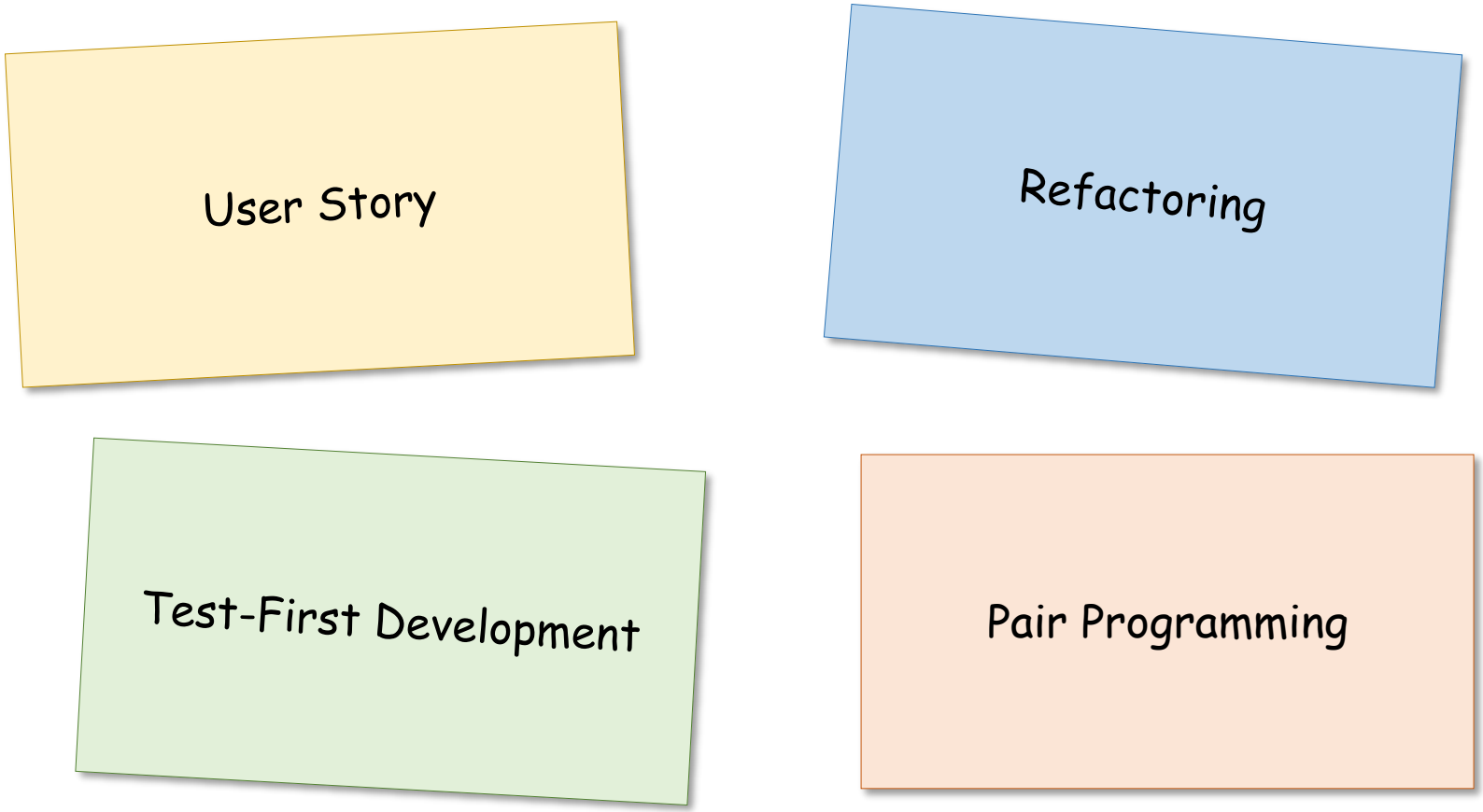
- วันหนึ่งๆ อาจจะมีการ build ประมาณ 2-3 versions
- ส่งมอบงานให้ลูกค้าบ่อยๆ ประมาณ 1-2 สัปดาห์ต่อครั้ง
- ต้องทดสอบซอฟต์แวร์ทุกครั้งที่ build และแต่ละ build จะถูกยอมรับก็ต่อเมื่อทดสอบผ่านทุกเงื่อนไขที่วางแผนไว้

Ref: • (image) Ian Somerville; “Software Engineering”, 10th Edition, Pearson (2016)

XP and Agile Principle

- Incremental Development
  - พัฒนาซอฟต์แวร์ครั้งละน้อยๆ เลือกที่สำคัญ และส่งมอบให้ลูกค้าลองได้ลองใช้บ่อยๆ
- Customer Involvement
  - มี full-time customer มาร่วมงานกับทีมพัฒนา
- People not Process
  - พัฒนาด้วยวิธี pair-programming (ใช้นักพัฒนาสองคนมาช่วยกันคิดช่วยกันเขียนโปรแกรม), collective ownership (แต่ละคู่ของนักพัฒนาต้องรู้และทำหลายๆ ส่วนของระบบ เพื่อให้ทำงานแทนกันได้อย่างมีประสิทธิภาพ), และ มีการเน้นที่ process น้อยๆ
- Change
  - ยอมรับการประเมินจากลูกค้าและนำมาพัฒนาในทุกๆ release
- Maintaining Simplicity
  - เขียน code มีมาตรฐาน เข้าใจง่าย นักพัฒนาอื่นๆ สามารถเข้าใจและพัฒนาต่อได้อย่างง่ายดาย

Key XP Practices



User Stories

- ลูกค้า (customer) หรือ ผู้ใช้ (user) เป็นส่วนหนึ่งของ XP team และมีส่วนร่วมในการตัดสินใจในแต่ละ requirement
- Requirement เขียนในรูป User Story
- เขียน User Story ในบัตรกระดาษ (card) แล้วทีมพัฒนาจะแบ่งงานตามตารางเวลาและ cost ที่ประเมิน
- ลูกค้าเลือก user story สำหรับใช้ในการพัฒนาของ iteration ต่อไป โดยเลือกจากลำดับความสำคัญและตารางเวลาที่เหมาะสม

As a *<user role>*,
I want *<desired feature>*,
so that *<value/benefit>*

ในฐานะ *<บทบาทผู้ใช้>*,
ฉันต้องการ *<สิ่งที่อยากให้ระบบทำได้>*,
ดังนั้น *<สิ่งที่ฉันได้รับ เช่น ง่ายขึ้น เร็วขึ้น ดีขึ้น>*

As a *student*,
I want *to have an online transcript*,
so that *I can show my GPA to anyone in the world*

User Story

- หลักการ I.N.V.E.S.T.
  - I = **Independent** (เสร็จได้ในตัวของมันเอง ไม่ต้องรออีก story นึง)
  - N = **Negotiable** (มา discuss พูดคุยกันได้ เปลี่ยนแปลงได้)
  - V = **Valuable** (มีประโยชน์ ให้ value)
  - E = **Estimable** (สามารถประมาณ effort ที่จะใช้ในการทำ story ได้)
  - S = **Small** (มีขนาดเล็ก จะได้ทำเสร็จเร็ว ได้ feedback เร็ว)
  - T = **Testable** (test ได้ มีตัววัดว่าเสร็จแล้วจริง)
- ตัวอย่างที่ดี
  - As a user, I want to sort videos by views, so that I can see popular videos.
- ตัวอย่างที่ไม่ดี
  - As a user, I want to have an online payment, so that I can make a payment online.

Ref: • (content) https://tanjai.me/user-story-ac199d775bd3

Refactoring

- ปรับปรุงรูปแบบการเขียน code เข้าใจง่ายขึ้น โดยที่ซอฟต์แวร์ยังทำงานได้คงเดิม ทำให้นักพัฒนาสามารถเข้าใจและต่อยอดซอฟต์แวร์จาก code ที่มีอยู่ได้สะดวกสบายขึ้น และพึ่งพาเอกสารน้อยลง
- ถ้ามีการเปลี่ยนโครงสร้างของไฟล์หรือโครงสร้างทางสถาปัตยกรรมของซอฟต์แวร์ อาจจะทำให้เกิด cost ที่สูง
- ตัวอย่างเช่น
  - แก้ไขโครงสร้างของ class โดยลบส่วนที่ซ้ำกันออกแล้วมา reuse แทน
  - เปลี่ยนชื่อ ตัวแปร และ methods ให้สื่อกับการใช้งาน
  - นำ lines of code ที่ใช้ซ้ำๆ กัน ไปเป็น method แทน แล้วเรียก method นั้น

Test-First Development

- เขียน test ก่อนที่จะเขียน code
- Test ต้องเป็นโปรแกรมที่ทำงานอัตโนมัติ (ไม่ใช่แค่เพียง data)
  - ตัวอย่าง testing framework คือ JUnit, NUnit, Mercury เป็นต้น
- เวลามีการเพิ่ม function ของซอฟต์แวร์ ต้อง test ทุก function ใหม่อีกครั้ง เพื่อให้มั่นใจว่าสิ่งที่สร้างขึ้นใหม่ไม่กระทบกับของเดิมที่ทำเสร็จแล้ว
- ปัญหา
  - นักพัฒนาชอบเขียน code แล้วค่อยเขียน test คือถ้าให้เขียน test ก่อน ก็จะเขียน test ไม่ดีพอ
  - ทำ unit test กับ User Interface เป็นเรื่องยาก
  - ยากที่จะทำให้นักพัฒนาซอฟต์แวร์เขียน test ได้ครอบคลุม

Test Case Description for Dose Checking

Test 4: Dose checking

Input:
1. A number in mg representing a single dose of the drug.
2. A number representing the number of single doses per day.

Tests:
1. Test for inputs where the single dose is correct but the frequency is too high.
2. Test for inputs where the single dose is too high and too low.
3. Test for inputs where the single dose * frequency is too high and too low.
4. Test for inputs where single dose * frequency is in the permitted range.

Output:
OK or error message indicating that the dose is outside the safe range.

Ref: • (image) Ian Somerville; "Software Engineering", 10th Edition, Pearson (2016)

Pair Programming

- นักพัฒนา 2 คนมานั่งเขียนโปรแกรมคู่กัน ที่คอมพิวเตอร์เครื่องเดียวกัน
- ทำให้การเขียน code มีคุณภาพดีขึ้นเพราะมีอีกคนช่วยเป็นหูเป็นตาให้
- ทำให้นักพัฒนามีทักษะการเขียน code ที่เป็นระบบระเบียบมากขึ้น (refactoring)
- ทำให้เกิดการกระจายความรู้ไปยังทีม
- ลดความเสี่ยงลงในกรณีที่มีนักพัฒนาลาออก (อย่างน้อยก็มีคนเข้าใจ code อีกหนึ่งคน)
- การทำ pair programming ไม่ได้บอกว่าจะมีประสิทธิผลมากขึ้น แต่ก็พอจะมีตัวอย่างที่ว่าทำ pair programming แล้วให้ผลที่ดีกว่าการให้นักพัฒนาสองคนทำงานแยกกัน

Agile Project Management

Scrum

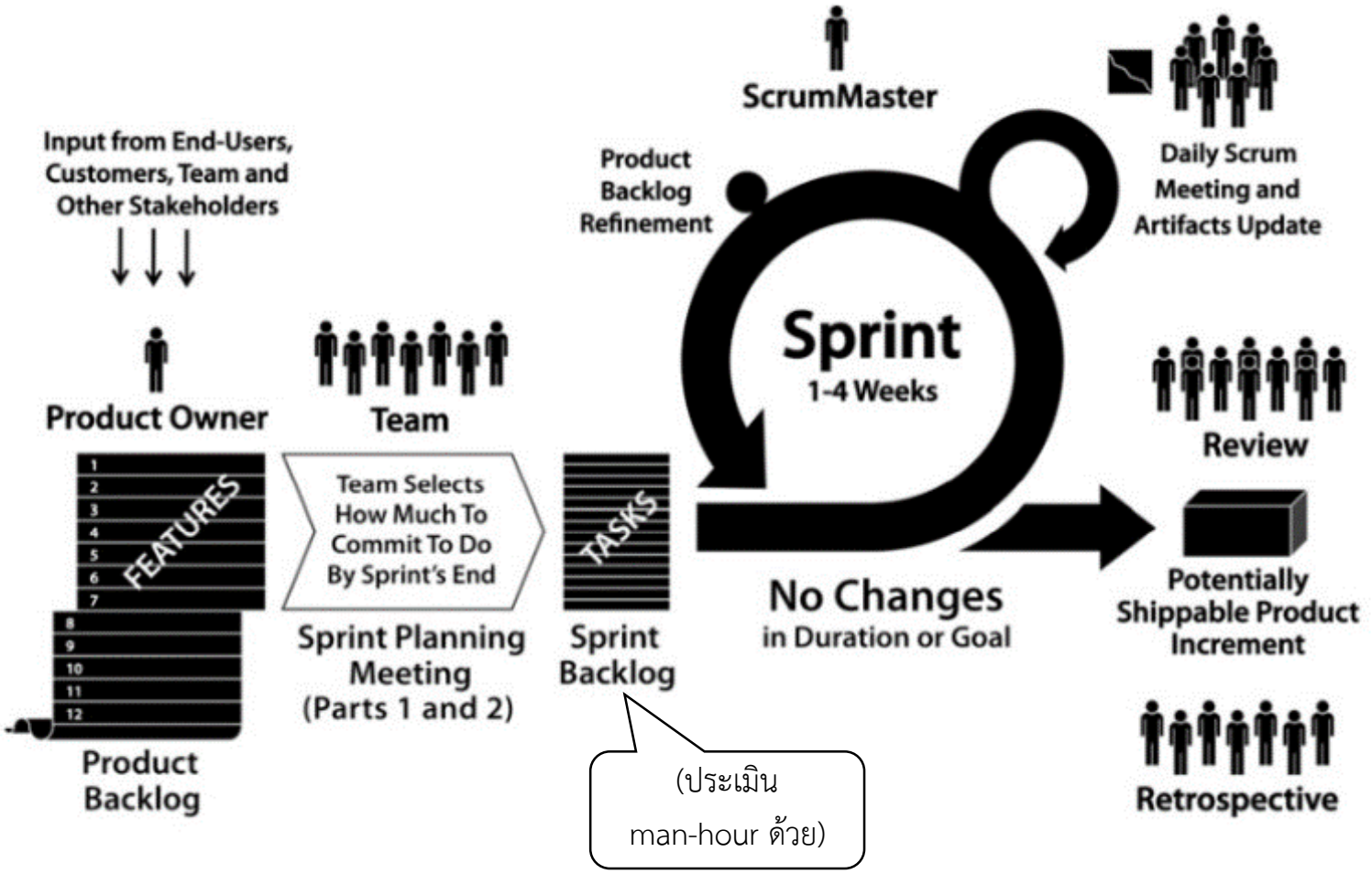
- เป็นวิธีหนึ่งที่ได้รับความนิยม โดยมุ่งที่การพัฒนาเป็นรอบๆ
- Scrum มี 3 phases
  - Initial Phase
    - เริ่มต้นจากการตั้งจุดประสงค์ของโครงการและออกแบบสถาปัตยกรรมทั้งระบบ
  - Series of Sprint Cycles
    - รอบการการพัฒนาที่จะมีไปเรื่อยๆ จนเสร็จสิ้น
  - Project Closure Phase
    - จบโครงการ ทำเอกสาร คู่มือ อบรมการใช้งาน รวมถึงการประชุมสรุปงานและบทเรียนที่พบ

Ref: • (image) http://news.bbc.co.uk/sport2/hi/rugby_union/7048733.stm

Scrum Terminology

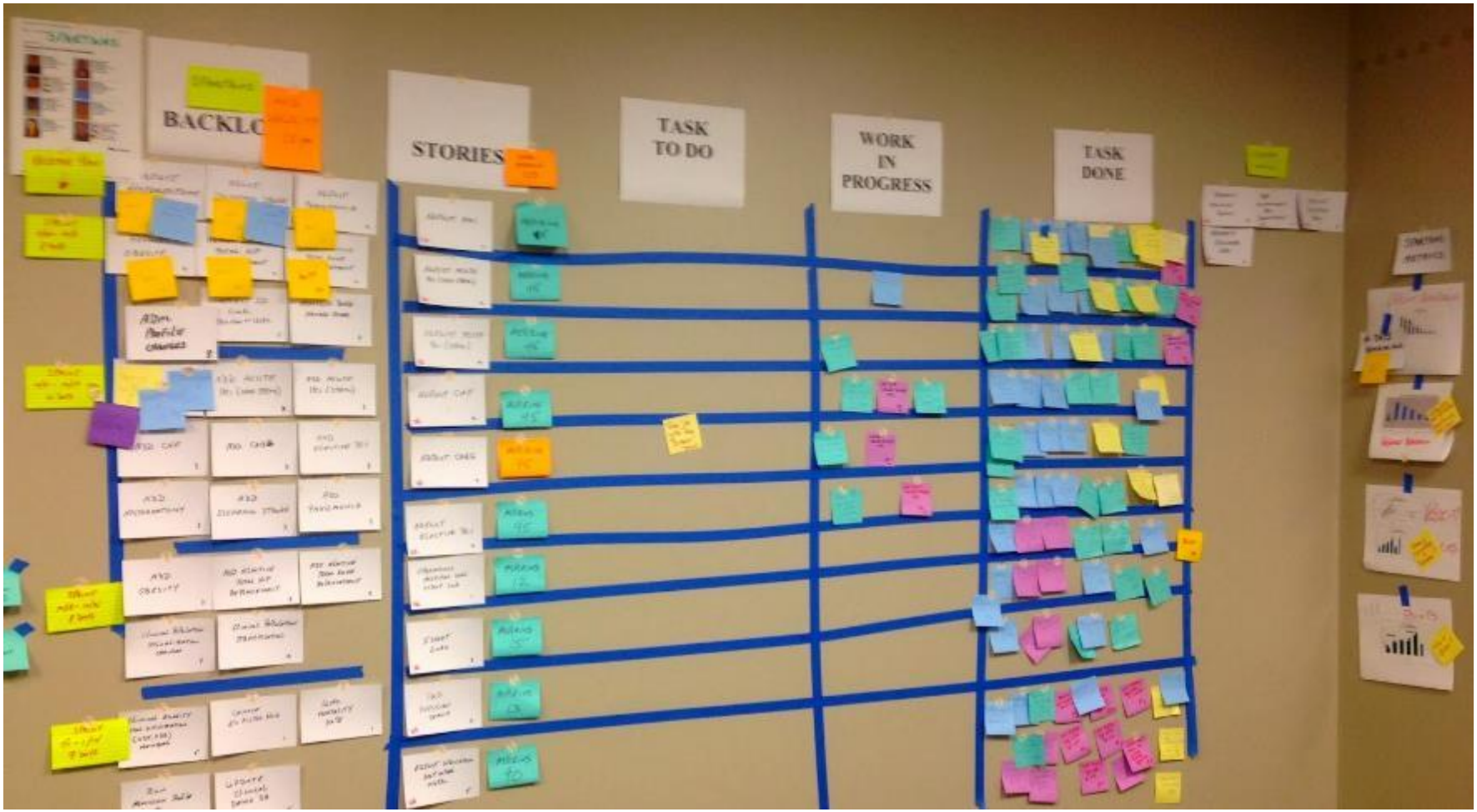
- **Development team**
  - นักพัฒนาในทีมที่ควรมีจำนวนไม่มาก (ไม่เกิน 7 คน)
- **Sprint**
  - รอบของการพัฒนา ประมาณ 2-4 สัปดาห์
- **Potentially Shippable Product Incensement**
  - ซอฟต์แวร์เวอร์ชันที่เกิดจาก sprint
- **Product Backlog**
  - งานที่ยังเหลือใน “to do” (ใช้ติดตามความคืบหน้าของการพัฒนา)
- **Product Owner**
  - บุคคลคนหรือกลุ่มคน ที่กำหนดลักษณะของซอฟต์แวร์ที่จะใช้พัฒนา
- **Scrum**
  - การประชุมทีมรายวันเพื่อติดตามความคืบหน้าและกำหนดสิ่งที่จะทำในวันนั้น
- **ScrumMaster**
  - คนที่นำให้ scrum process เข้าที่เข้าทาง
- **Velocity**
  - การประเมินว่าทีมงานสามารถรับผิดชอบ product backlog ได้เท่าไรในแต่ละ sprint

Scrum Sprint Cycle



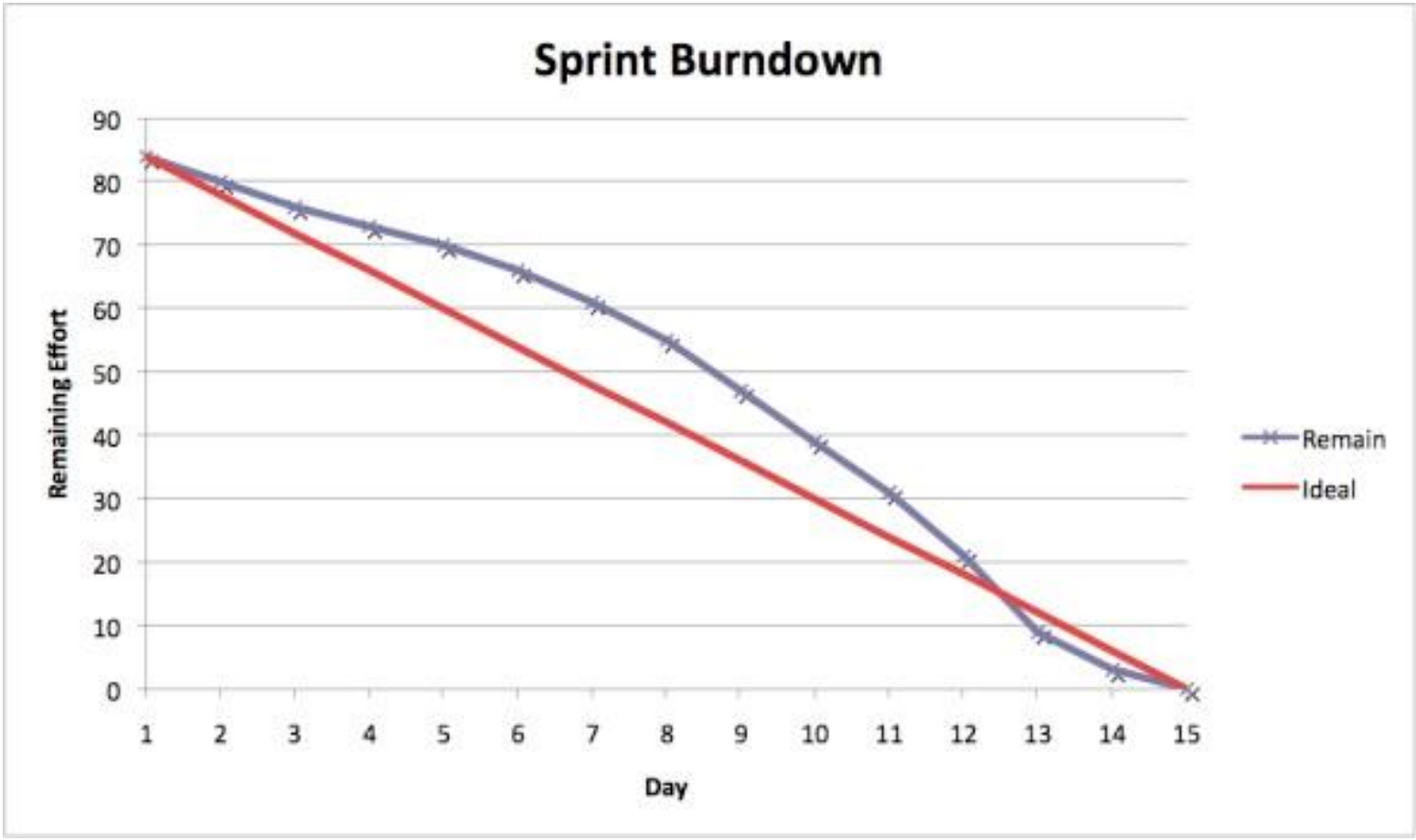
Ref: • (image) Pete Deemer, Gabrielle Benefield, Craig Larman, Bas Vodde (2010)

Scrum Board



Ref: • (image) http://agilecomplexificationinverter.blogspot.com/2013/11/elements-of-effective-scrum-task-board.html

Burndown Chart



Ref: • (image) http://www.scrum-institute.org/Sprint_Burndown_Reports.php

Teamwork in Scrum

- ScrumMaster จะเป็นคนดำเนินการช่วย
  - จัด Standup meeting ทุกวัน
  - ตามงานที่อยู่ใน backlog ให้เสร็จ
  - จดบันทึกการตัดสินใจต่างๆ ที่เกิดขึ้นในแต่ละวัน
  - วัดความก้าวหน้าเทียบกับ backlog
  - คุยกับลูกค้า หรือผู้บริหารนอกทีม
- Standup Meeting
  - ยืนประชุมกัน (ใช้คำว่ายืน เพื่อให้ประชุมไม่ต้องใช้เวลานาน) และดู scrum board ร่วมกัน
  - อธิบายความคืบหน้าจากเมื่อวาน
  - แจ้งปัญหาที่พบ และปรึกษาหาคำชี้แนะ
  - แจ้งว่าวันนี้จะทำอะไร

Scrum Benefits

- ผลิตภัณฑ์ซอฟต์แวร์ถูกแบ่งย่อยจนเป็นชิ้นส่วนที่สามารถเข้าใจและบริหารได้ง่าย
- ทั้งทีมเห็นความคืบหน้าของทีมด้วยกันผ่าน standup meeting และ scrum board ทำให้ทีมคุยกันง่าย เข้าใจกัน และพัฒนาตัวเองไปด้วยกัน
- ลูกค้าเห็นงานที่ทำอยู่ และสามารถประเมินและให้คำแนะนำตลอดเวลา
- เกิดความเชื่อถือระหว่างลูกค้าและนักพัฒนา ซึ่งเป็นวัฒนธรรมที่ดีที่จะทำให้โครงการประสบความสำเร็จ

Scaling Agile Methods

Scaling Agile Method

- Agile เหมาะกับงานขนาดเล็กและขนาดกลาง ที่ใช้ทีมงานจำนวนไม่มากและนั่งทำงานอยู่ใกล้กัน
- “Scaling Up” คือ การทำให้สามารถใช้วิธีการใน Agile กับ โครงการขนาดใหญ่ ที่ใช้ทีมงานจำนวนมากทำ
- “Scaling Out” คือ การทำให้สามารถใช้วิธีการของ Agile ในการทำงานระหว่างทีม ที่อยู่ในองค์กรขนาดใหญ่ซึ่งมีประสบการทำงานมาหลายปี
- เมื่อทำ scaling agile ต้องให้ความสำคัญกับการรักษาแนวทางของ agile ไว้ด้วย
  - วางแผนให้ยืดหยุ่น, ออก release บ่อยๆ, continue integration, test-driven development, และ สื่อสารในทีมให้ดี

Practical Problem with Agile Method

- Agile Development เหมาะกับการพัฒนาระบบใหม่ แต่ไม่เหมาะกับการการทำ maintain
- เนื่องจากกระบวนการ Agile นั้นการทำงานค่อนข้างเป็นกันเอง ทำให้กิจกรรมบางอย่างไม่สอดคล้องกับกระบวนการทางกฎหมายเกี่ยวกับเรื่องทำสัญญาจัดซื้อจัดจ้าง
- การทำสัญญาต้องออกแบบระบบไว้ทั้งหมดและไม่ยืดหยุ่นกับการแก้ไข
- สัญญาส่วนใหญ่คิดเงินตาม deadline แต่ Agile ให้ความสำคัญกับฟังก์ชั่นที่พัฒนาในแต่ละรอบการพัฒนา
- Agile เหมาะกับทีมที่ทำงานใกล้ๆ กัน (co-located team) จะได้คุยกันง่ายขึ้น แต่ปัจจุบันทีมพัฒนากระจายกันอยู่ทั่วโลก
- เอกสารมีไม่มาก จะลำบากต่อการรื้อฟื้นความรู้ไปทำ maintanance

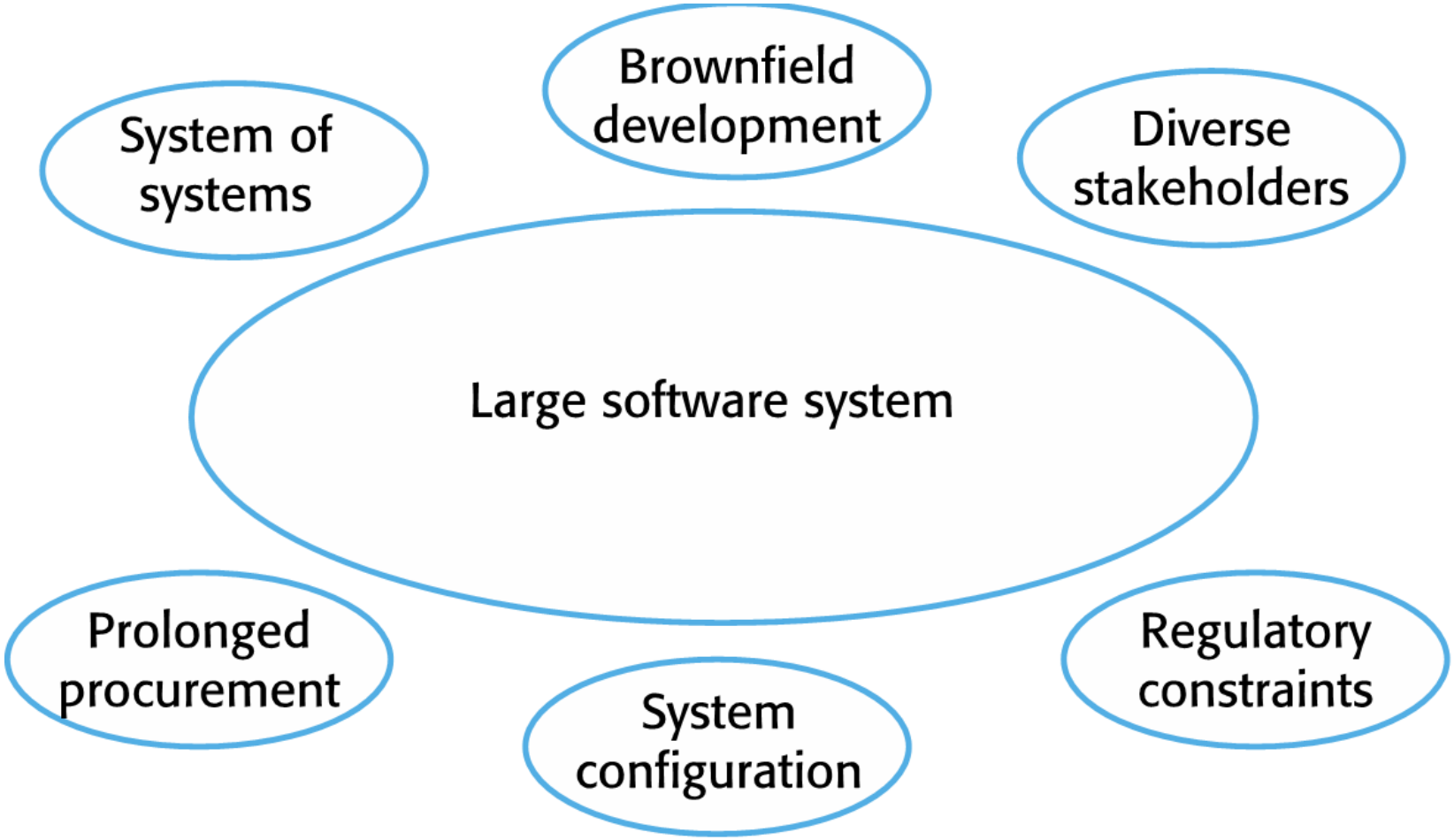
Agile Issues

- System Issues
  - **ทำระบบได้ใหญ่แค่ไหน?**
    - เหมาะกับทีมขนาดเล็กที่คุยกันได้
  - **ชนิดของระบบที่ทำได้?**
    - ระบบที่ต้องมีการออกแบบเพื่อวิเคราะห์ก่อนที่จะพัฒนา
- People and Teams
  - **ทีมงานต้องเก่งแค่ไหน?**
    - ต้องสื่อสารกันได้ มีความรับผิดชอบ (process น้อย ต้องมั่นใจว่าทีมงานมีวินัย)
  - **จะจัดการทีมพัฒนาอย่างไร?**
    - ต้องทำเอกสารการออกแบบ (design document) ถ้าทีมงานไม่ได้อยู่ร่วมกัน (distributed teams)

Large Systems

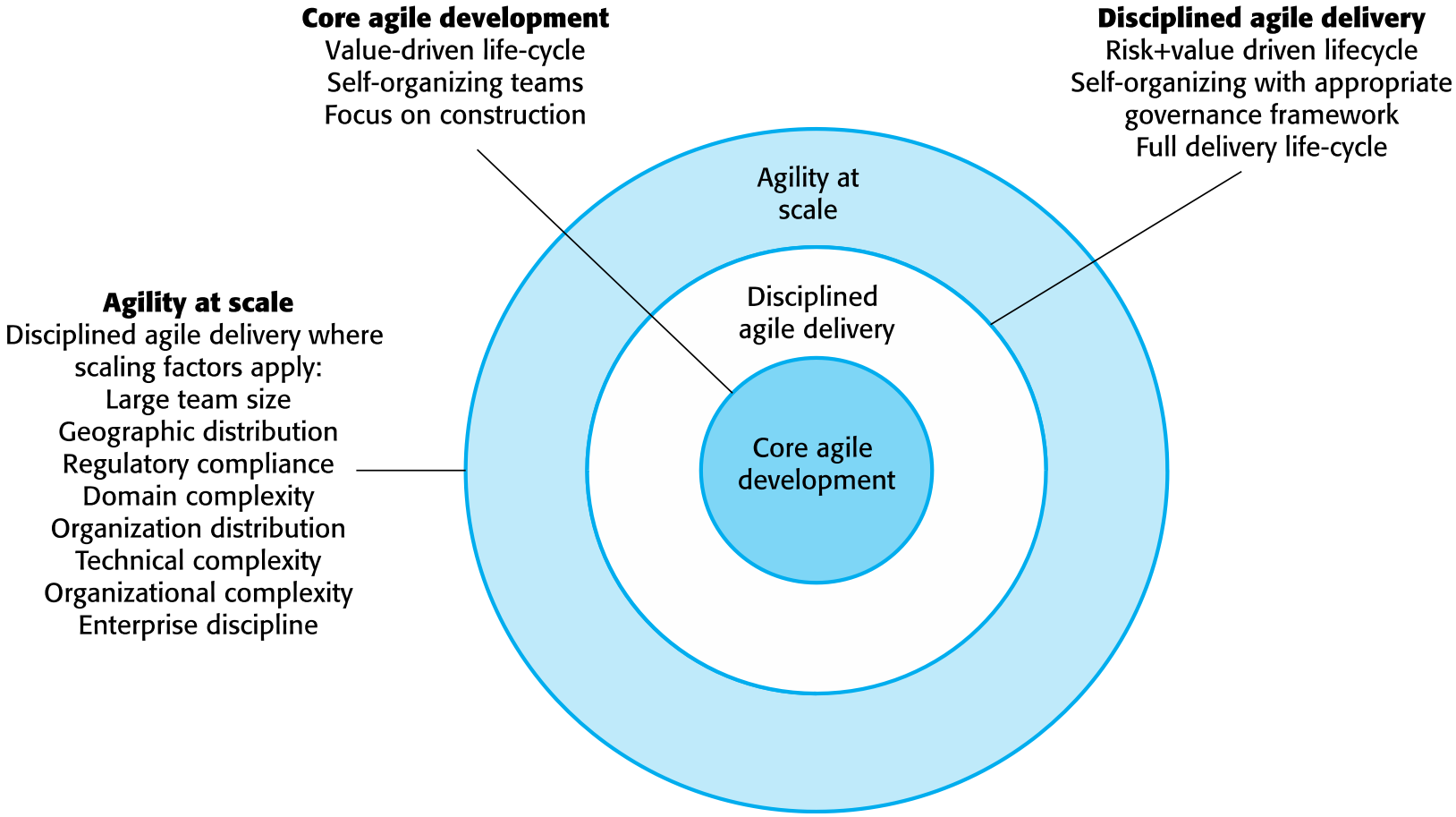
- ระบบขนาดใหญ่ คือระบบที่มี component จำนวนมาก และทีมงานอยู่ต่างสถานที่กัน บางทีก็อยู่ต่าง time zone กัน
- ระบบขนาดใหญ่ อาจเป็น “brownfield systems” ที่ต้องประกอบด้วย หรือต้องติดต่อกับระบบเก่าๆ (legacy system) หลายๆ ระบบ
- ระบบขนาดใหญ่ มีผู้มาเกี่ยวข้อง (stakeholders) จำนวนมาก ทั้ง นักพัฒนา นักออกแบบ นักทดสอบ ฯลฯ ทำให้ระบบขนาดใหญ่ยากที่จะรวบรวมคนเหล่านี้ให้มา stand up meeting ได้

Factors in Large Systems



Ref: • (image) Ian Somerville; "Software Engineering", 10th Edition, Pearson (2016)

IBM's Agility at Scale Model



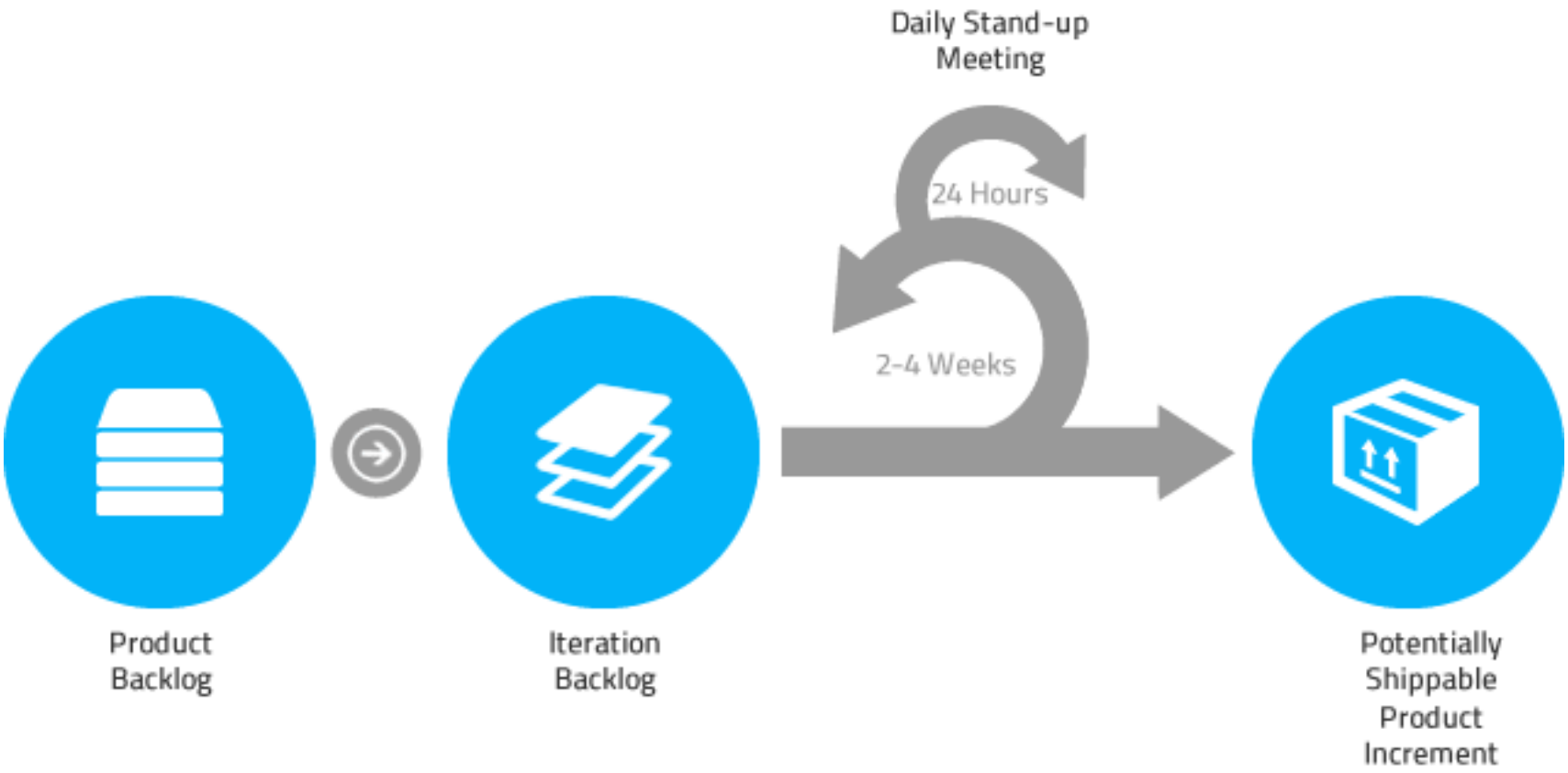
Ref: • (image) Ian Somerville; "Software Engineering", 10th Edition, Pearson (2016)

Multi-Team Scrum

- **Role Replication**
  - แต่ละทีมมี Product Owner และ ScrumMaster
- **Product Architects**
  - มีการออกแบบสถาปัตยกรรมซอฟต์แวร์โดยรวม
- **Release Alignment**
  - มีการกำหนดวัน release ซอฟต์แวร์ให้ตรงกัน
- **Scrum of Scrums**
  - คือมีตัวแทนทีมไปประชุม scrum

Summary

Key Points



Ref: • (image) https://www.mera.com/services/processes/agile

Key Points

Ref: • (image) http://www.innovation-series.com/2013/08/15/oddi-gone-agile/index.html

“

Success is almost impossible
without an environment of
organizational safety and trust.

”

Ray Arell

つづく