CS 361 – Software Engineering

Plan-Driven and Agile Process

Kamonphop Srisopha

Churee Techawut

Adapted from: Mark Sherriff and Will McBurney, 2020, Slide presentation in Advanced Software Development

Faculty of Science, Chiang Mai University

คณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่

Agenda

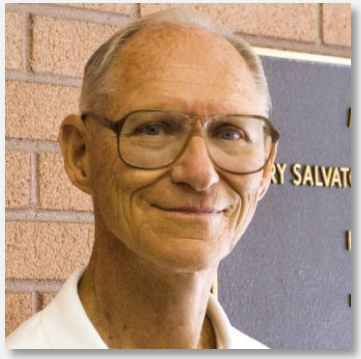
- Plan-driven methodologies
  - Rational Unified Process
- Agile manifesto
- Agile development techniques
  - Extreme programming
  - Scrum
- Polar chart
- Plan-driven vs Agile development

Plan-Driven Methodologies

Plan-driven methodologies

"Plan-driven methods work best when developers can determine the requirements in advance … and when the requirements remain relatively stable, with change rates on the order of one percent per month."

Barry Boehm



Plan-driven methodologies

- Remember: “**continuum**”
- No methodology is locked on one side or the other of the spectrum
- Depending on how they are implemented, they can be more plan-driven or more agile
- Some methodologies are just more traditionally associated with one end of the spectrum than the other

Plan-driven methodologies

- Focus on repeatability and predictability
- Defined, standardized, and incrementally improving processes
- Thorough documentation
- A defined software system architecture defined up-front
- Detailed plans, workflow, roles, responsibilities, and work product descriptions
- Ongoing risk management
- Focus on verification and validation

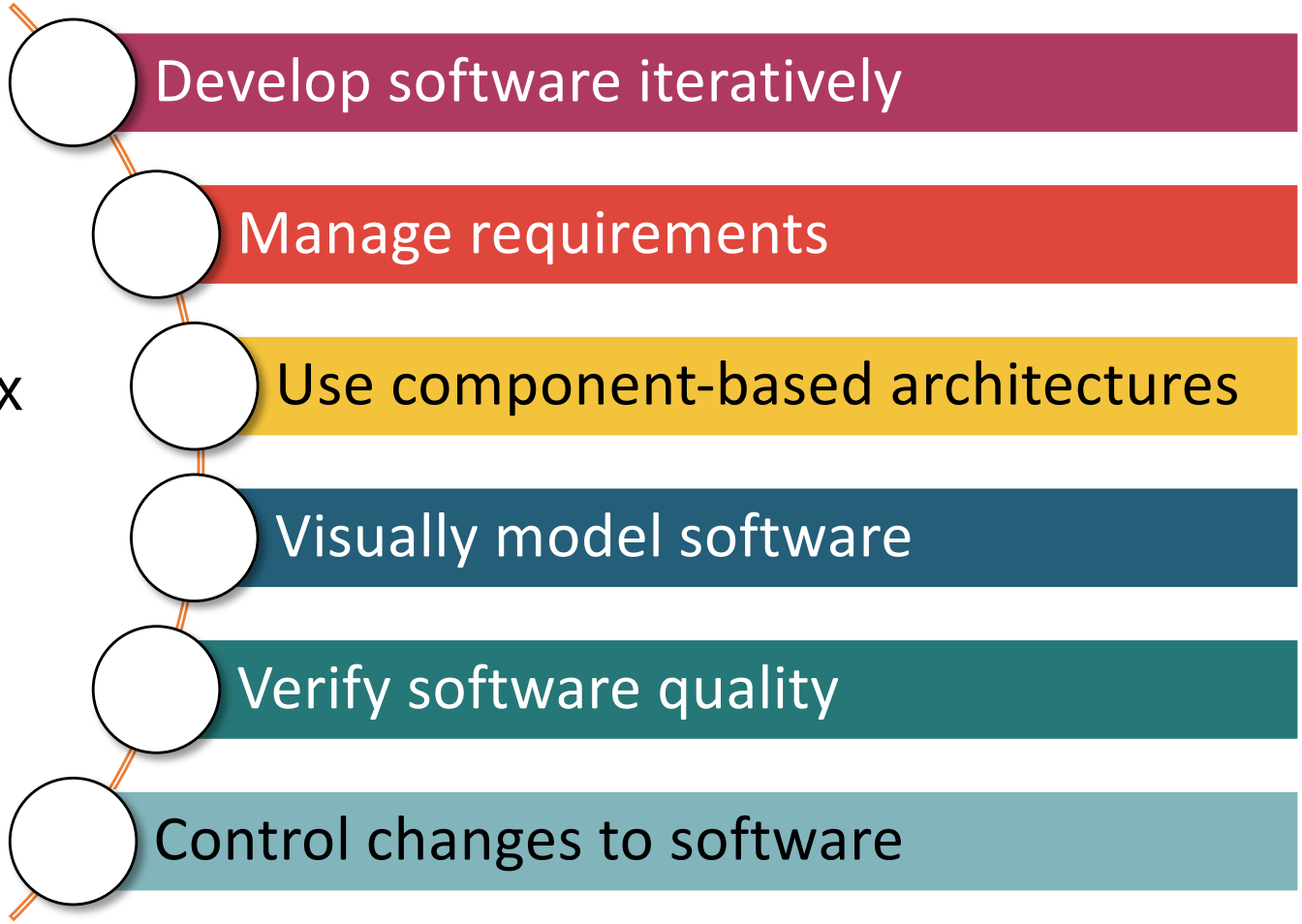
Rational Unified Process (RUP)

- The Rational Unified Process (RUP) is interesting because it unto itself is a product
- Originally created in the late 1980s by Ivar Jacobson
- His company was bought by a company called Rational (who made software development tools)
- Rational was bought by IBM (which incorporated those tools into their own systems)
- Built around the **Unified Modeling Language (UML)**
- Toolsets were created to help build UML diagrams and then translate them into code
- These toolsets were incorporated into various IDEs (one was a version of Eclipse…)

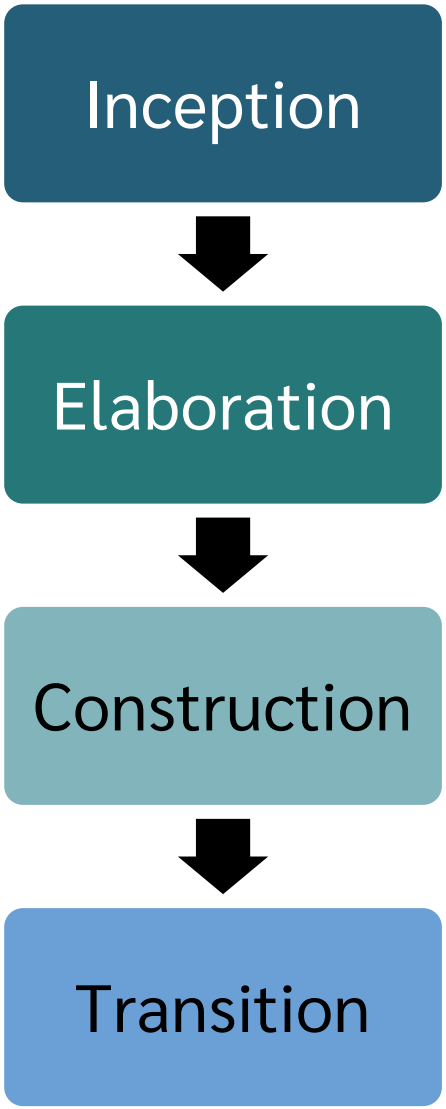
Rational Unified Process

The process itself is configurable

Built around six best practices



Rational Unified Process – 4 Phases

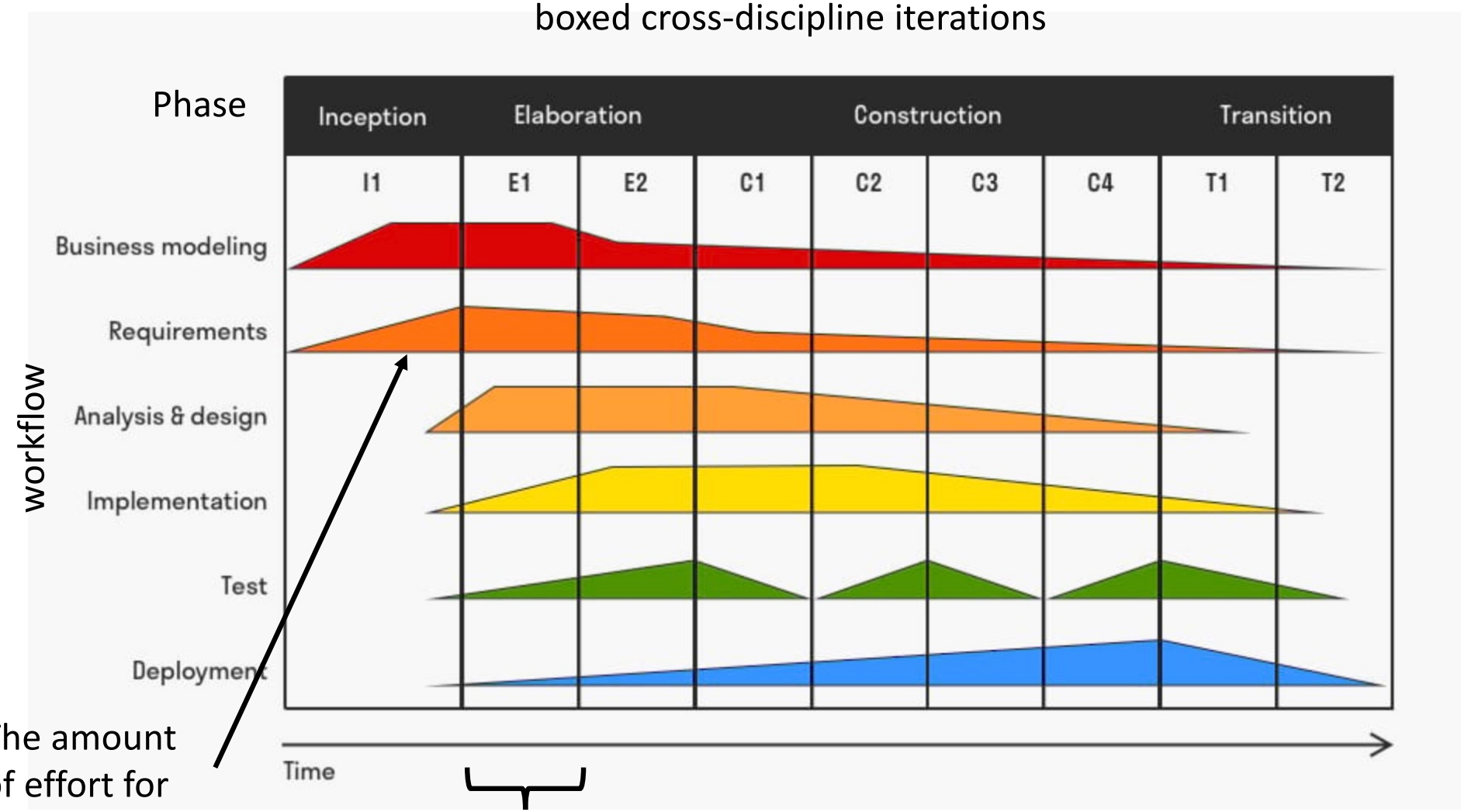


- Each phase has a distinct transition point into the next phase
- The four phases could be iterated on
- The transition phase could be to an internal build
- Within the phases are workflows with particular tasks that increase and decrease in activity
- Members of the team are given specific roles to follow within each workflow

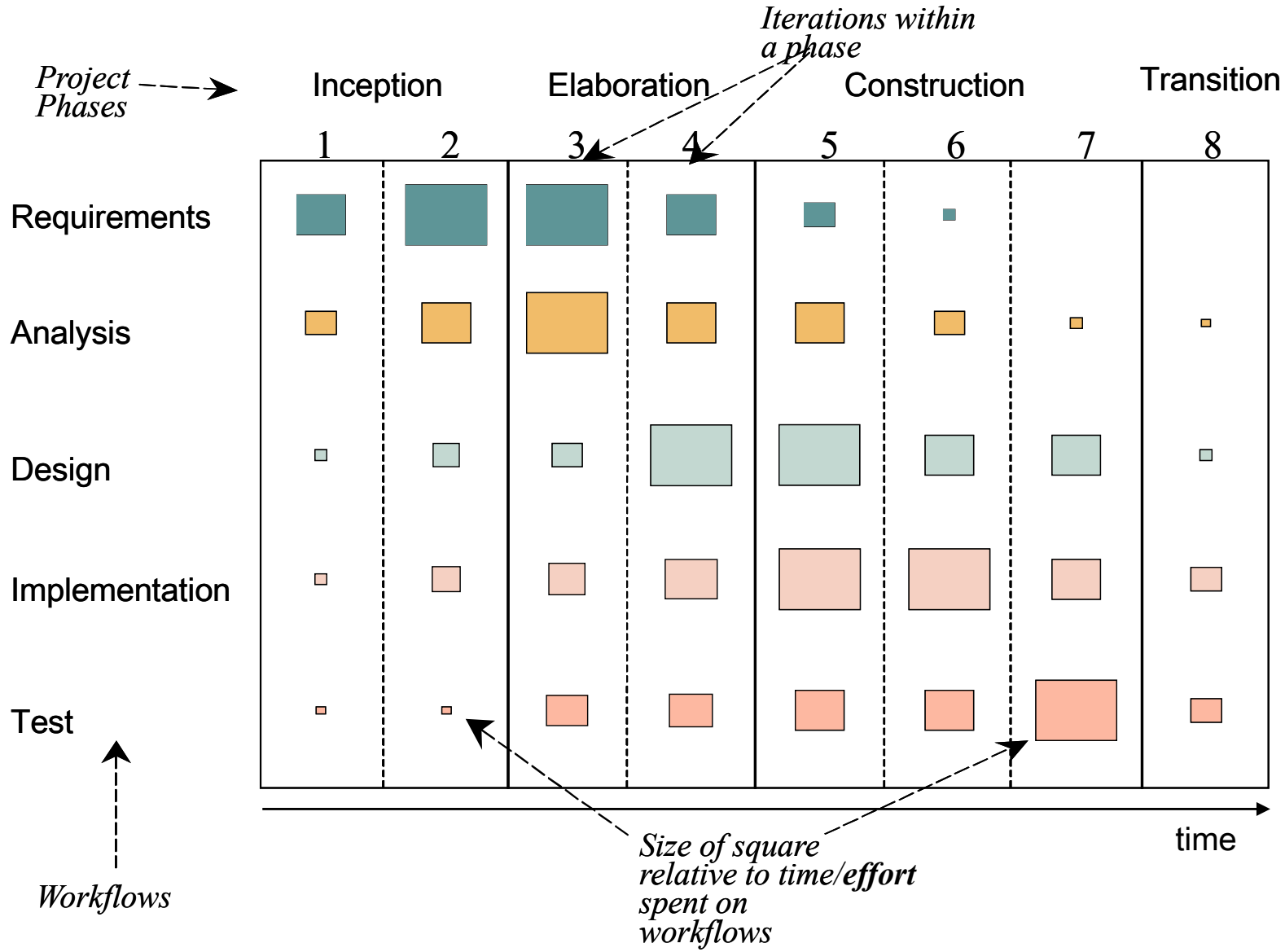
Rational Unified Process

Iterative Development

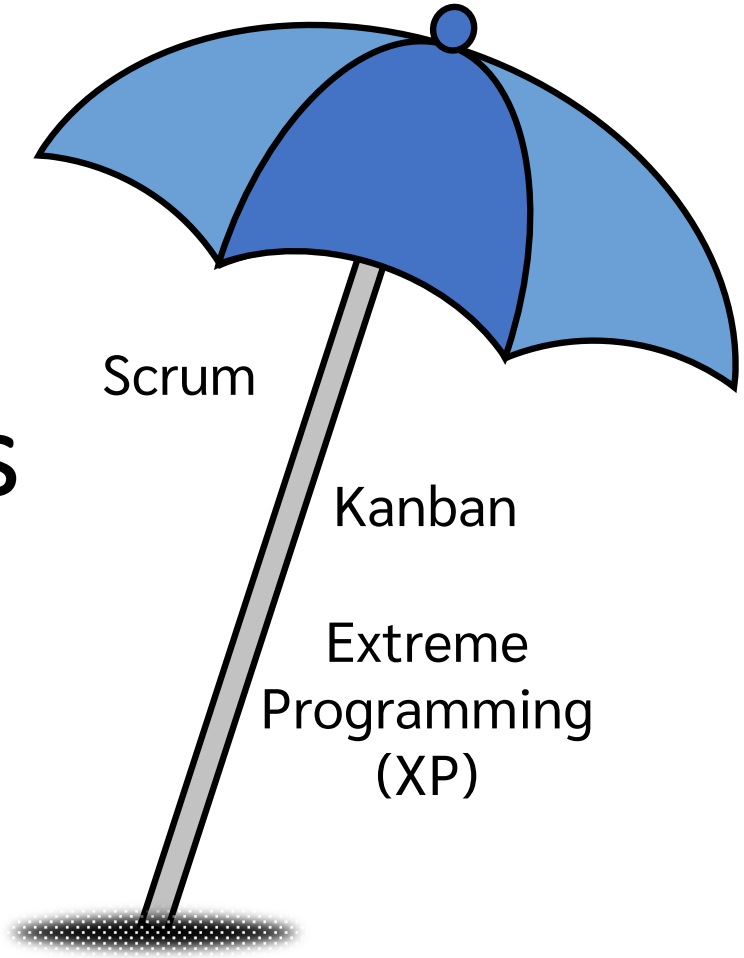
Business value is delivered incrementally in time-boxed cross-discipline iterations



A Simplified View



Agile Methodologies



Agile

ไม่ใช่กระบวนการแต่เป็น**แนวคิด**ของการทำงานซึ่งสามารถนำไปประยุกต์ใช้ได้หลายวิธี

Agile development

- Rapid development and delivery is now often the most important requirement for software systems
- Fast changing software requirements. Quick evolving to reflect changing business needs.
- The system is developed as a series of versions or increments with stakeholders involved in version specification and evaluation
- Frequent delivery of new versions for evaluation
- Extensive tool support (e.g. automated testing tools) used to support development.
- Minimal documentation – focus on working code

Agile method

- Dissatisfaction with the overheads involved in software design methods of the 1980s and 1990s led to the creation of agile methods. These 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.
- The aim of agile methods is to reduce overheads in the software process (e.g. by limiting documentation) and to be able to respond quickly to changing requirements without excessive rework.

Agile Manifesto (คำแถลงอุดมการณ์ Agile)

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

The creation of the *Manifesto for Agile Software Development*

- February 2001
- Snowbird, Utah (เขาไปเล่นสกีกัน)
- Seventeen of the biggest names in this emerging field met to come up with a set of principles and concepts for all agile processes

"แม้ว่าเราจะเห็นความสำคัญของสิ่งที่กล่าวด้านล่าง แต่เราเห็นความสำคัญของสิ่งด้านบน มากกว่า"

Read more: https://agilemanifesto.org

12 Principles of Agile

1. Our highest priority is to **satisfy the customer** through early and continuous delivery of valuable software.
2. **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
3. **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must **work together daily** throughout the project.
5. Build projects around **motivated individuals**. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.

Read more: https://agilemanifesto.org/principles.html

12 Principles of Agile

7. **Working software** is the primary measure of progress.
8. Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace (of work) indefinitely.
9. Continuous attention to **technical excellence** and good design enhances agility.
10. **Simplicity-** the art of maximizing the amount of work not done--is essential (don't overcomplicate or over-engineer things)
11. The best architectures, requirements, and designs emerge from **self-organizing teams**.
12. At regular intervals, the team **reflects** on how to become more effective, then tunes and adjusts its behavior accordingly.

Read more: https://agilemanifesto.org/principles.html

Extreme Programming (XP)

Extreme programming (XP)

- Extreme Programming (XP) was probably the first well-known agile process, created by Beck and Cunningham
- Rarely does a team follow XP strictly now (or even before)
- XP was built around the same ideas in the manifesto (it actually predates the manifesto) with 12 core practices
- Each practice could be adjusted for a given team / project
- Today, the practices survive as core to agile development if not explicitly as a part of XP

The 12 XP Practices

- **The Planning Game**
  - Use user stories and story points at the beginning of each iteration to plan
- **Whole Team** (a.k.a. On-site customer)
  - Everyone (from business to coders to QA to customer) is part of one team and is co-located as much as possible to promote cohesion
- **Sustainable Pace** (a.k.a. The 40-hour work week)
  - Team members that work too much burn out quicker and are much more likely to introduce defects into the system

The 12 XP Practices

- **Small Releases**
  - Get customer feedback early and often to prevent building the wrong solution
- **Coding Standards**
  - Following good coding standards and naming mitigates the need for extensive documentation
  - The code should always be understandable and an artifact unto itself
- **Pair Programming**
  - An early investment in learning how to pair yields significant results in reducing the number of defects injected into a system

The 12 XP Practices

- **Test-Driven Development**
  - Test first, then write the code that passes the test
  - Make tests automated and add them to the overall code base
- **Refactor Mercilessly**
  - Continuously improve your codebase and don’t leave anything stagnant
- **Collective Code Ownership**
  - No one “owns” any part of the system
  - Anyone can update any part of the code when needed

The 12 XP Practices

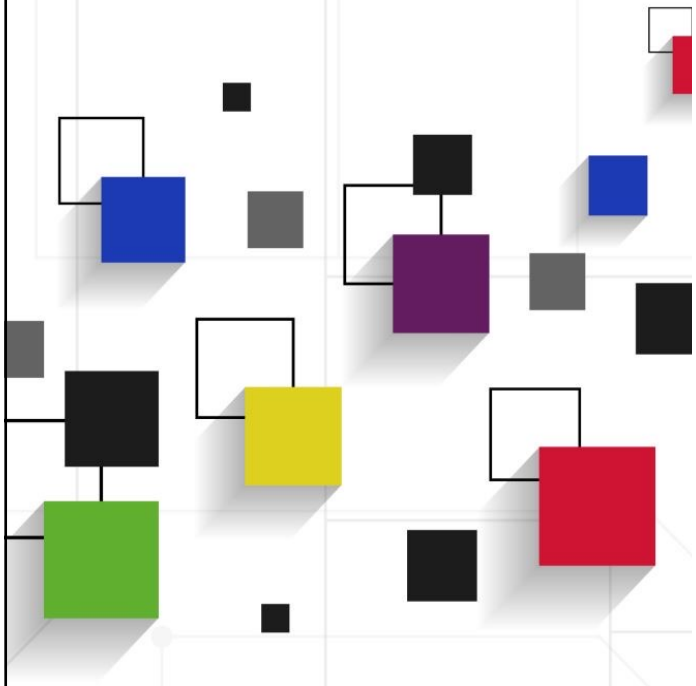
- **Continuous Integration**
  - Constantly build the code
  - When code is checked in, run all tests and rebuild
  - Always have a working build
- **Simple Design**
  - Build the simplest thing that works
- **Metaphor Over Architecture**
  - Have a narrative / description that can be used to communicate design decisions

The “13th” XP Practice

- **Stand-up Meeting**
  - The start of every workday begins with a meeting in which all team members must stand in a circle, giving updates and what they will work on that day
  - Why do they stand? To prevent the meeting from going too long…

Scrum

Scrum



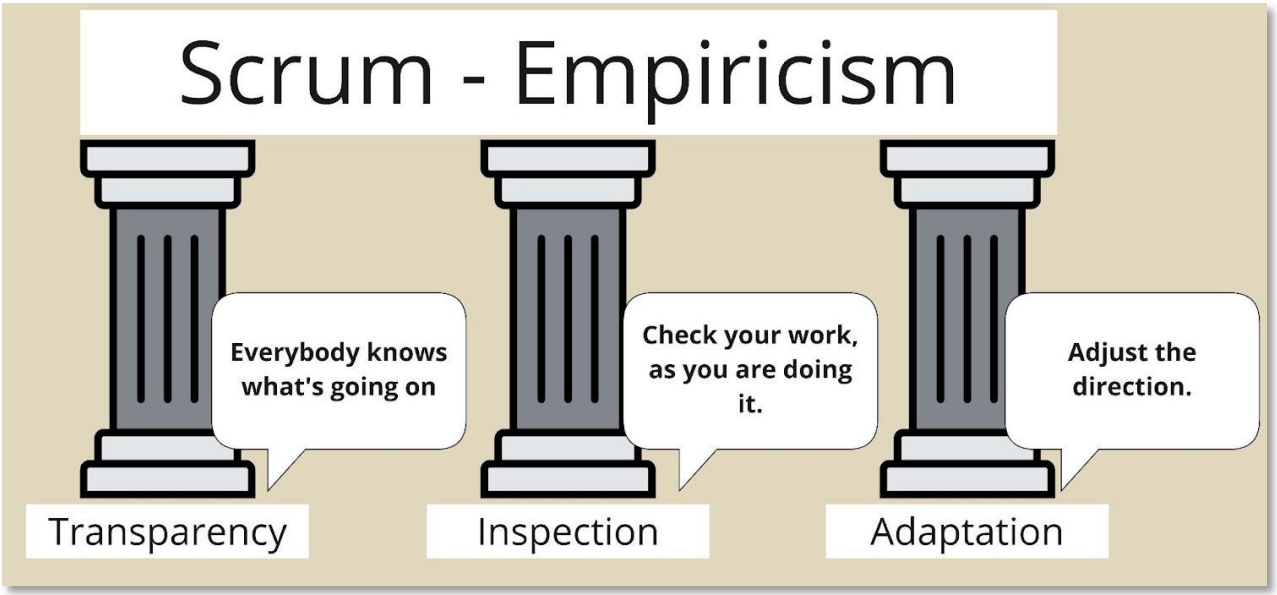
- Originally created in the mid-90s by Sutherland and Schwaber
- **Focuses on:**
  - Simplicity
  - ANY complex project (not just software!)
  - Constant feedback (both with team and customer)
  - Sprint iterations always end with a potentially shippable product

Scrum (Rugby)

A team packing closely together to strategically move forward to gain the possession of the ball
(ทีมต้องโอบกันให้แน่น เพื่อช่วยกันดันตัวไปข้างหน้า เพื่อให้ทีมของตนได้ครองบอล)

3 Pillars of Scrum

เน้นการนำความรู้จากประสบการณ์จากการลงมือทำจริง (Empiricism) มาพัฒนาการดำเนินงานในปัจจุบันให้ดียิ่งขึ้น ประกอบด้วย 3 ส่วน



ความโปร่งใส

ทีมจะต้องเห็นภาพชัดเจนและเข้าใจตรงกัน มาตรฐานเดียวกัน ไม่ตีความหมายต่างกัน รู้ว่าใครทำอะไร

การตรวจสอบ

สามารถนำผลลัพธ์การดำเนินกิจกรรมต่างๆ มาตรวจสอบและวัดผลว่าเป็นไปตามวัตถุประสงค์ที่วางไว้หรือไม่ ได้อยู่เสมอ

การปรับตัว

หากพบว่ามีการทำงานที่ไม่เป็นไปตามแผน จะต้องมีการปรับปรุงการดำเนินการต่างๆ

https://resources.scrumalliance.org/Article/three-pillars-of-scrum

3 Key Roles of Scrum

Product Owner (The visionary)

รับผิดชอบ: communication

- เป็นคนที่ต้องเข้าใจตัวงานอย่างถ่องแท้
- บริหารจัดการ product backlog ให้มี requrements ที่สอดคล้องกับวิสัยทัศน์และเป้าหมายของ product
- ตรวจสอบ product increments ว่าตรงตามเป้าหมายหรือไม่

Scrum Master (The coach)

รับผิดชอบ: Facilitate Teamwork

- เข้าใจกระบวนการของ scrum ทำให้ทีมนำ scrum ไปใช้ได้อย่างถูกต้อง
- ช่วยเหลือแก้ไขปัญหาหรืออุปสรรคที่เกี่ยวกับการดำเนินการต่างๆ

Development Team (The creators)

รับผิดชอบ: Deliver Increment

- ทำงานร่วมกันในการสร้างและส่งมอบ product increments ในแต่ละ sprint
- รู้หน้าที่ของตนว่าต้องทำอะไร รับผิดชอบส่วนไหน cross-functional

3 Key Artifacts

Product Backlog:

- รายการของสิ่งต่างๆที่จำเป็นในการสร้างหรือปรับปรุง product เช่น ความต้องการ (requirements) และ bug reports โดยส่วนใหญ่ความสำคัญของสิ่งต่างๆในรายงานจะถูกกำหนดโดย Product Owner
- มีการเปลี่ยนแปลงอยู่เรื่อยๆ (เพิ่ม/ลด item, เพิ่ม/ลด ความสำคัญ)

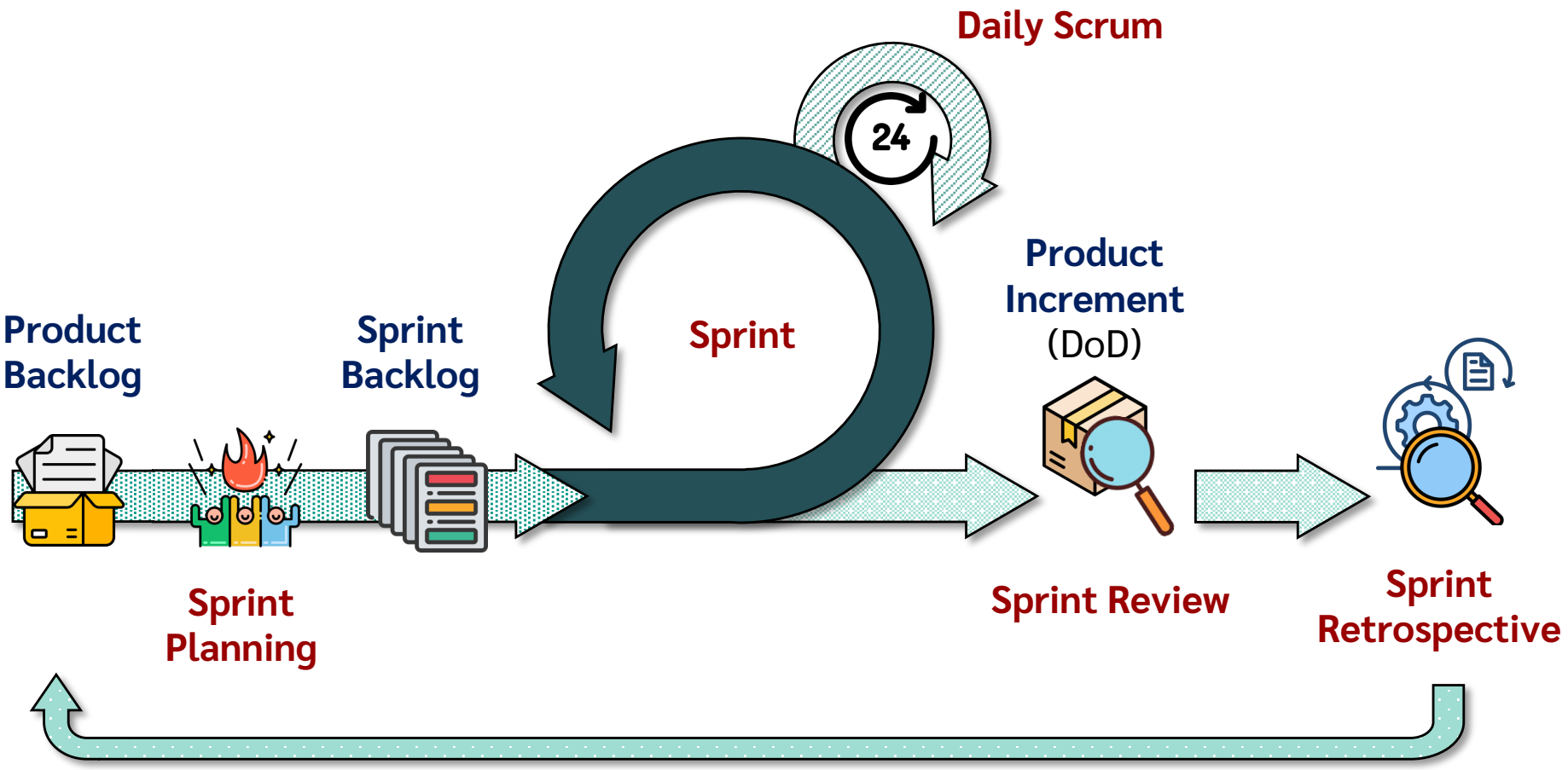
Sprint Backlog:

- เป็นรายการของงานต่างๆ ที่ Product Owner และ Development Team ตกลงกันว่าจะทำใน 1 Sprint โดยเลือกจาก Product backlog ในตอนทำกิจกรรม Sprint Planning (เลือกงานขึ้นอยู่กับหลายปัจจัย เช่น ประสิทธิภาพของทีมใน sprint ก่อนหน้า, กรอบเวลาของ sprint)

Increments:

- งานที่ทำเสร็จพร้อมสำหรับส่งมอบให้แก่ผู้ใช้งาน (อาจเป็นคนส่วนหนึ่งแต่ต้องทำงานได้) โดยหมายถึงงานที่เสร็จใน Sprint ปัจจุบันและรวมงานของ Sprint ที่ผ่านมา
- Increments ควรมีความสมบูรณ์ตามความต้องการของ Definition of Done (DoD) ที่ทีมกำหนดขึ้น

Scrum Framework



Scrum Events

Artifacts

5 Scrum Events

Sprint Planning (การวางแผน Sprint)

- **Objective:**
  - กิจกรรมที่เอาไว้กำหนดวัตถุประสงค์ของ sprint ทำให้ทีมเห็นเป้าหมายตรงกัน รู้ว่าต้องทำงานอะไรบ้าง และรู้ว่าต้องทำงานชิ้นนี้ไปเพื่ออะไร **(why)**
  - ทีมเลือกงานจาก product backlog ว่าจะทำงานอะไร **(what)** จากปัจจัยหลายๆอย่าง และ ต้องคุยกับว่าจะทำงานๆนั้นๆอย่างไร (เช่น แบ่งงานอะไรยังไง, ใช้ tools อะไร, ใครเป็นผู้รับผิดชอบหลัก) **(how)**
  - ทีมร่วมกัน Prioritize (จัดเรียงลำดับความสำคัญ)
- **Timebox:** 2- 4 ชั่วโมง
- **Frequency:** หนึ่งครั้งตอนต้นของ sprint

5 Scrum Events

Daily Scrum

- **Objective:**
    - การประชุมประจำวัน (Daily Meeting) หรืออาจจะเรียกว่า Standup Meeting เพราะเป็นการล้อมวงยืนประชุมในเวลาสั้นๆ เน้นให้ทุกคนในทีม แจ้งความคืบหน้าในการพัฒนางานแก่กัน เช่นว่าเมื่อวานทำอะไรไป วันนี้จะทำอะไร เจออุปสรรคอะไรบ้าง

- **Timebox:** 15 นาที
- **Frequency:** ทุกวัน

5 Scrum Events

Sprint Review

- **Objective:**
    - ตรวจสอบงาน (increments) ที่ทีมสร้างขึ้นหรือปรับปรุงขึ้นใน sprint
    - แสดงผลลัพธ์ของงาน (demo) ให้แก่ผู้มีส่วนได้ส่วนเสีย (Stakeholders) เพื่อรับฟังข้อเสนอแนะ
    - ตรวจสอบ Definition of Done (DoD) และ Acceptance Criteria ของ Increments
    - ทบทวน Product/Sprint Backlog ดูงานที่เหลือและเอางานที่ยังไม่เสร็จกลับไปใส่ แปลงความคิดเห็นมาใส่ใน product backlog
- **Timebox:** 2-4 ชั่วโมง (เวลาส่วนมากมาจาก engagement ของ stakeholders)
- **Frequency:** หลังจากจบ Sprint

5 Scrum Events

Sprint Retrospective

- **Objective:**
  - เป็นกิจกรรมที่เกิดขึ้นหลัง Sprint Review เพื่อให้คนในทีมร่วมกันตรวจสอบว่า Sprint ที่ผ่านมา เป็นอย่างไรบ้าง เป็นการตรวจสอบเพื่อพัฒนาตัวกระบวนการ (process improvement) เช่น ความสัมพันธ์ภายในทีม การสื่อสาร สภาพแวดล้อม กระบวนการและเครื่องมือต่าง ๆ พร้อมทั้งสร้างแผนปรับปรุงเพื่อทำให้ Sprint ต่อไปได้ผลออกมาดีกว่าเดิม

- **Timebox:** ไม่เกิน 3 ชั่วโมง
- **Frequency:** หลังจากจบ Sprint

5 Scrum Events

Sprint

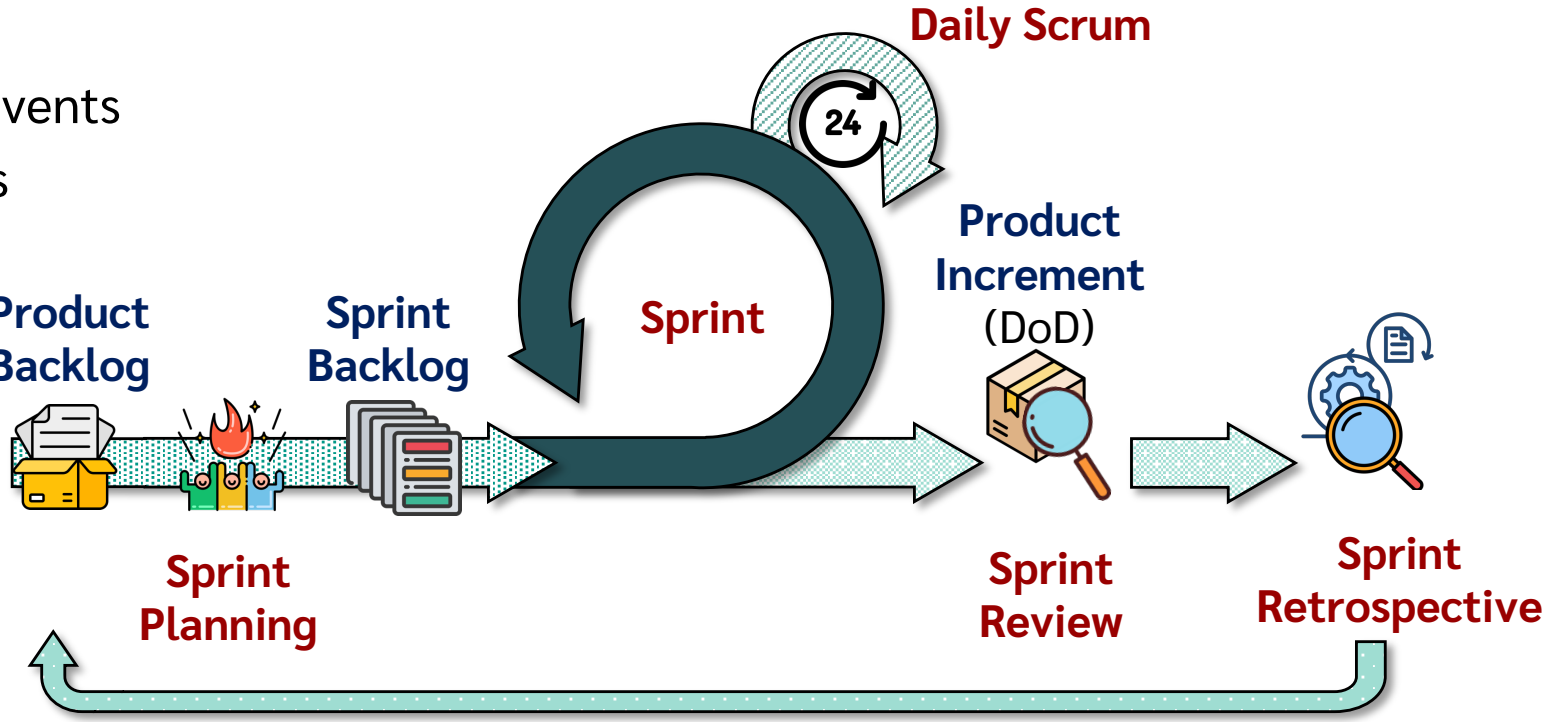
Sprint คือระยะเวลาหรือรอบการทำงานที่กำหนดไว้ (มักจะเป็นระยะเวลา 1-4 สัปดาห์) ในการใช้เวลาทำงานเพื่อสร้าง "Increment" หรือส่วนของผลิตภัณฑ์ที่สามารถใช้งานได้ เมื่อจบ Sprint จะต้องได้รับชิ้นงานตามที่วางแผนไว้และสามารถนำไปส่งมอบให้แก่ผู้ใช้ ทำให้ผู้ใช้ไม่จำเป็นต้องรอจนงานเสร็จทั้งหมดก่อนจึงจะได้รับ แต่สามารถนำบางส่วนไปใช้งานก่อนได้เลยและส่วนอื่นๆ ทยอยตามมาเพิ่มเติมในภายหลัง

Sprint หนึ่งๆอาจเรียกได้ว่าเป็น container ที่รวมกิจกรรมที่กล่าวมาก่อนหน้าทั้งหมดไว้: Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective

Scrum Framework

Scrum Events

Artifacts



- **ในตอนเริ่มของ Sprint**, ทีมจะมีการประชุม Sprint Planning โดยที่ทีมจะระบุว่าจะทำงานอะไรใน Sprint นี้ งานเหล่านี้จะถูกเลือกมาจาก Product Backlog และไปสู่ Sprint Backlog
- **ระหว่าง Sprint**, ทีมจะทำงานร่วมกันเพื่อสร้าง Increment. ทีมจะมีการประชุมแบบ Daily Scrum ทุกวัน เพื่ออัพเดทงานและปัญหาที่เกิดขึ้น
- **เมื่อสิ้นสุด Sprint**, ทีมจะมีการประชุม Sprint Review เพื่อทบทวนผลของงาน และการประชุม Sprint Retrospective เพื่อทบทวนกระบวนการทำงานของทีมและวางแผนเพื่อปรับปรุงในรอบ Sprint ถัดไป.

A product owner creates a prioritized wish list called a **product backlog**

During **sprint planning**, the team pulls a small chunk from the top of that wish list, a **sprint backlog**, and decides how to implement those pieces

The team has a certain amount of time — **a sprint** (usually two to four weeks) — to complete its work, but it meets each day to assess its progress, **daily Scrum**

Along the way, the Scrum Master keeps the team focused on its goal

At the end of the sprint, the work should be potentially shippable **(increments)**

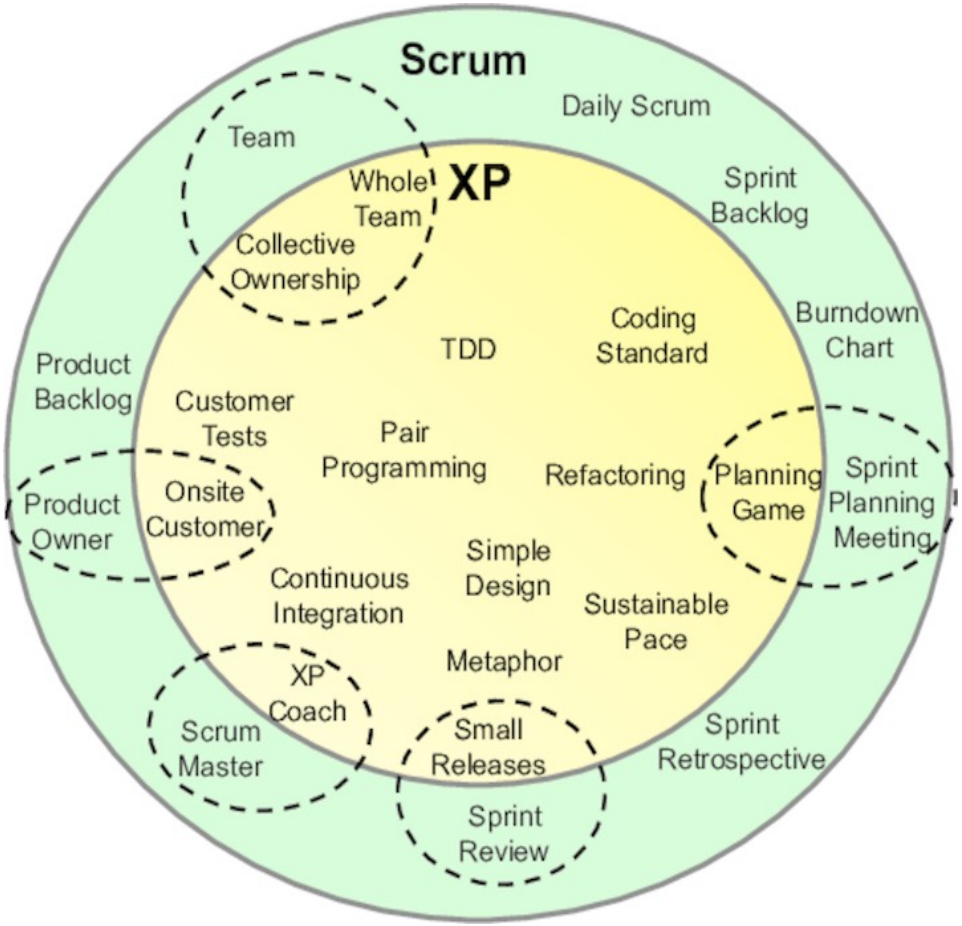
The sprint ends with a **sprint review** and **sprint retrospective**

As the next sprint begins, the team chooses another chunk of the product backlog and begins working again

สรุป Scrum Framework

- Scrum Events
- Artifacts

Scrum vs XP



Scrum concepts only has

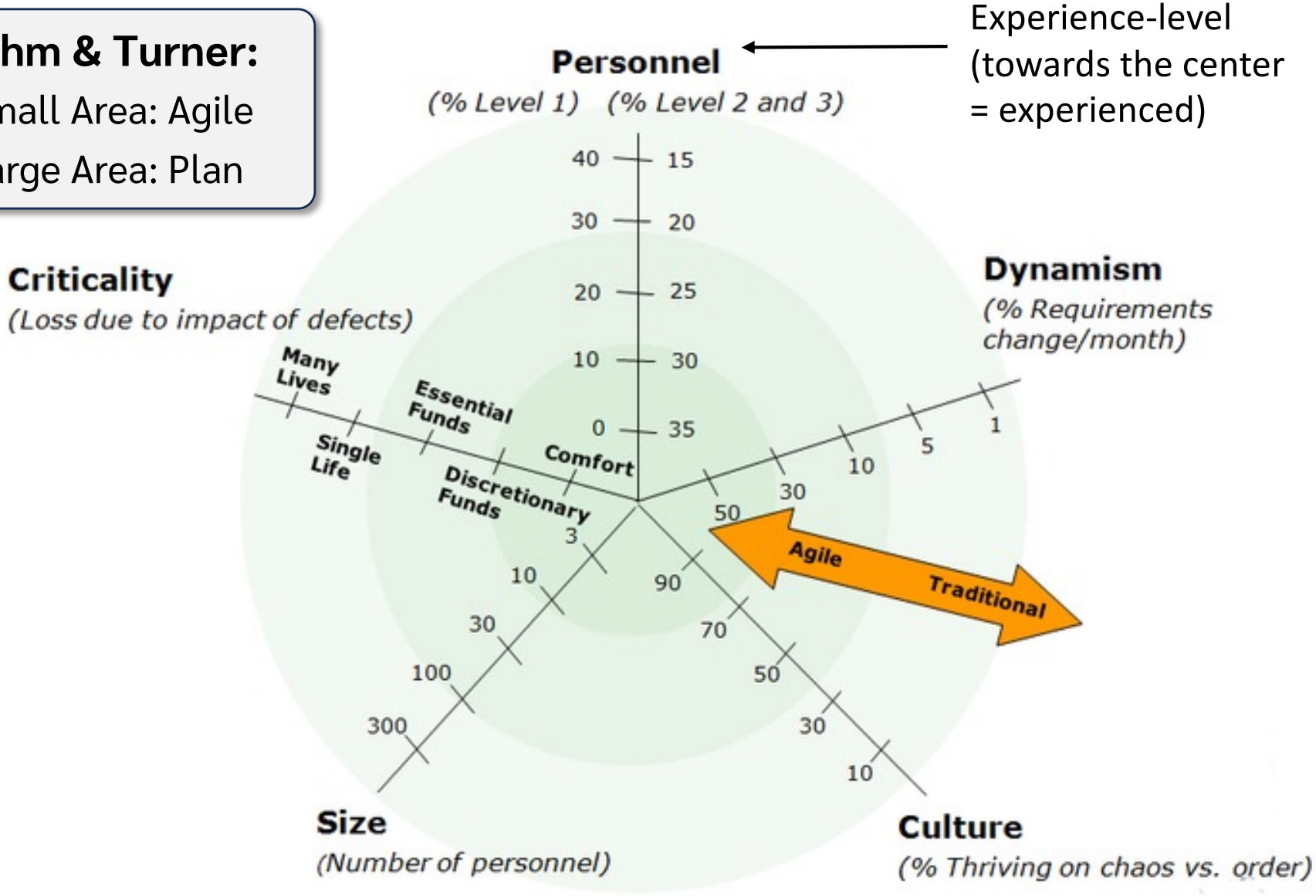
- 5 Events
  - Daily scrum
  - Sprint Planning
  - Sprint Review
  - Sprint Retrospective
  - *Sprint (container)*
- 3 artifacts
  - Product Backlog
  - Sprint Backlog
  - Potentially shippable product
- 3 roles
  - Product Owner
  - Scrum Master
  - Development Team

XP has more technical practices

Should we be more Agile or more Plan-driven?

Boehm & Turner:

- Small Area: Agile
- Large Area: Plan



Plan-driven vs Agile development

- **Plan-driven development**
  - A plan-driven approach to software engineering is based around separate development stages with the outputs to be produced at each of these stages planned in advance.
  - Not necessarily waterfall model – plan-driven, incremental development is possible (e.g., Rational Unified Process)
  - Iteration occurs within activities.
- **Agile development**
  - Specification, design, implementation and testing are inter-leaved and the outputs from the development process are decided through a process of negotiation during the software development process.

Questions ?