Software Engineering COMP1035

Lecture 16

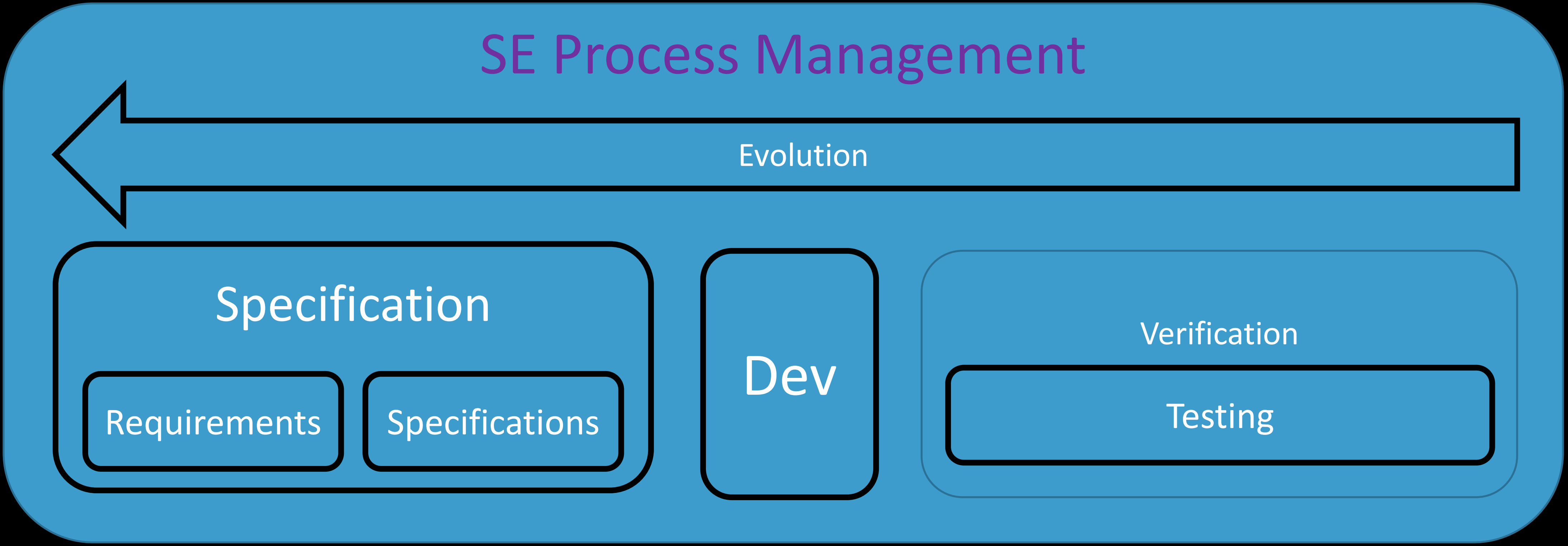
Agile Method

Outline

- Agile History
- Agile Principles
- Agile Frameworks
- Agile Survey

Where We Are In the Process

Keeping Track of SE Module



Process/Project Management

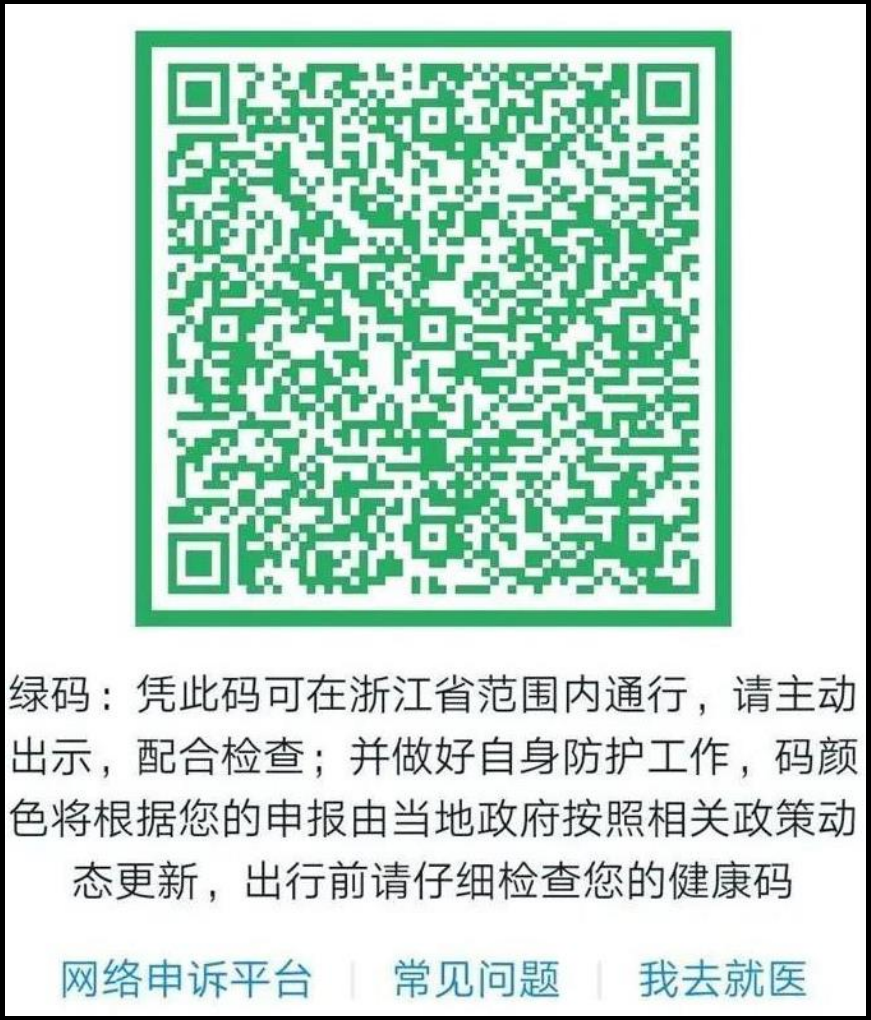
- Deliver good quality product
- **Use good reliable methods**
- Reducing as much risk as possible
- Plan / budget / manage a project based on these

Agile

- Everything we've seen so far, **is an approach, method, technique**
  - or standard, or good practice
  - but they have a lot of presumptions
  - like a big team, big budget, and lots of time
  - or a small team, with expertise, etc
- Agile methods exist **to be flexible** about the SE stages
- **There are times when it's good to plan**
  - **and times when good to be agile.**

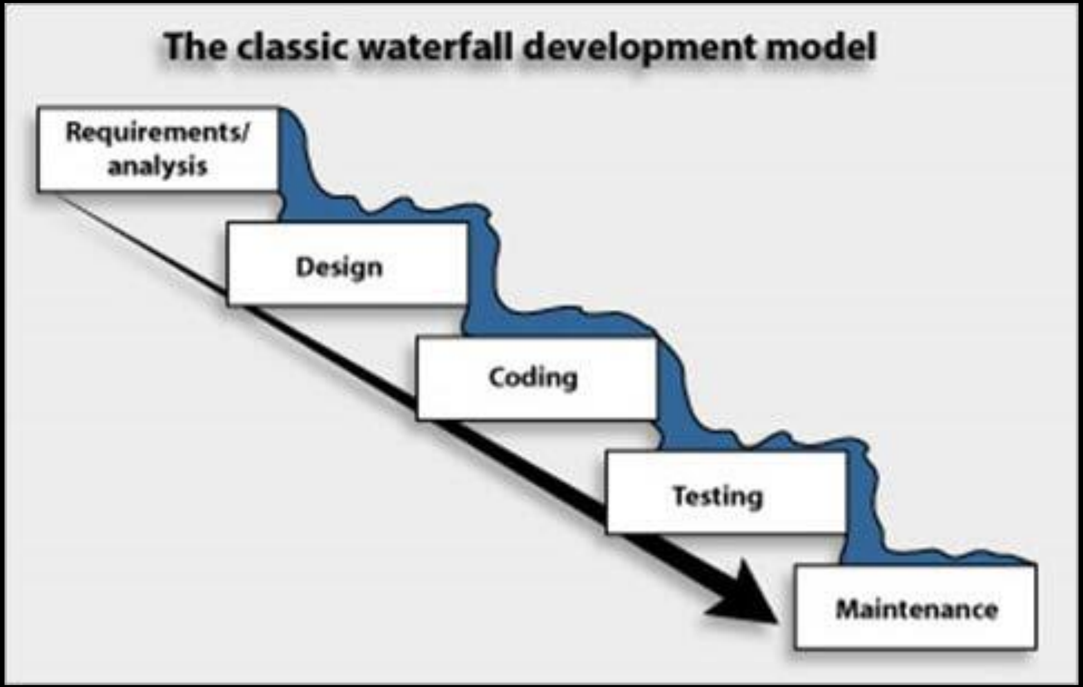
University of Nottingham
UK | CHINA | MALAYSIA

Real World Scenario…


绿码：凭此码可在浙江省范围内通行，请主动出示，配合检查；并做好自身防护工作，码颜色将根据您的申报由当地政府按照相关政策动态更新，出行前请仔细检查您的健康码

Traditional Methods

- Waterfall method: in this approach, software is developed in distinct phases, each leading to the next phase in a sequence resembling a waterfall. Once a phase is complete, the process moves on to the next phase and there is no turning back.



1990s - Concern with Traditional Methods

- Rapid Software Development was getting more important
  - global business was changing quickly
  - more SE projects were changing mid-process
- Waterfall model (and others) are heavy top-down approaches
  - do all the planning, then all the work, then all the testing
  - expensive, slow, etc
- Many limitations
  - depends on lots of documentation to be successful
  - lots of waiting times between different teams
  - a really expensive way to fix buggy code
  - not much freedom for developers etc
  - hard to manage change during a project
  - difficult to measure the progress within each phase

University of Nottingham
UK | CHINA | MALAYSIA

2001 - The Agile Alliance

- Feb 2001, 17 people met at a ski resort, to improve SE ideas
- These people had already come up with SCRUM, TDD, etc
- They are not saying ‘SE Methodologies are bad’
  - they are saying that SE methodologies can be **adaptable**
  - in order to be **effective for *the situation***
  - and to **avoid unnecessary bureaucracy** where it’s not needed

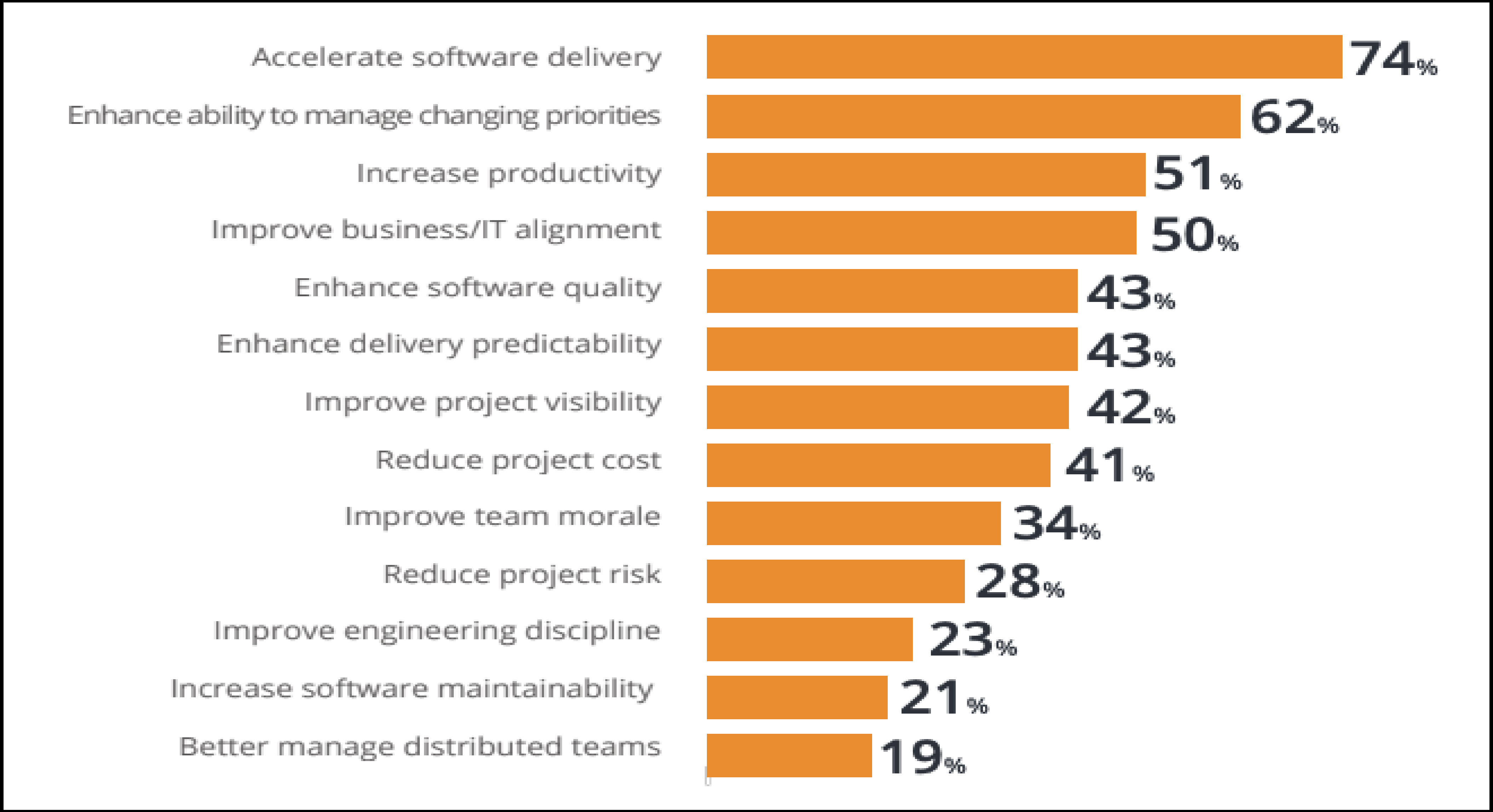
Agile Principles

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**.

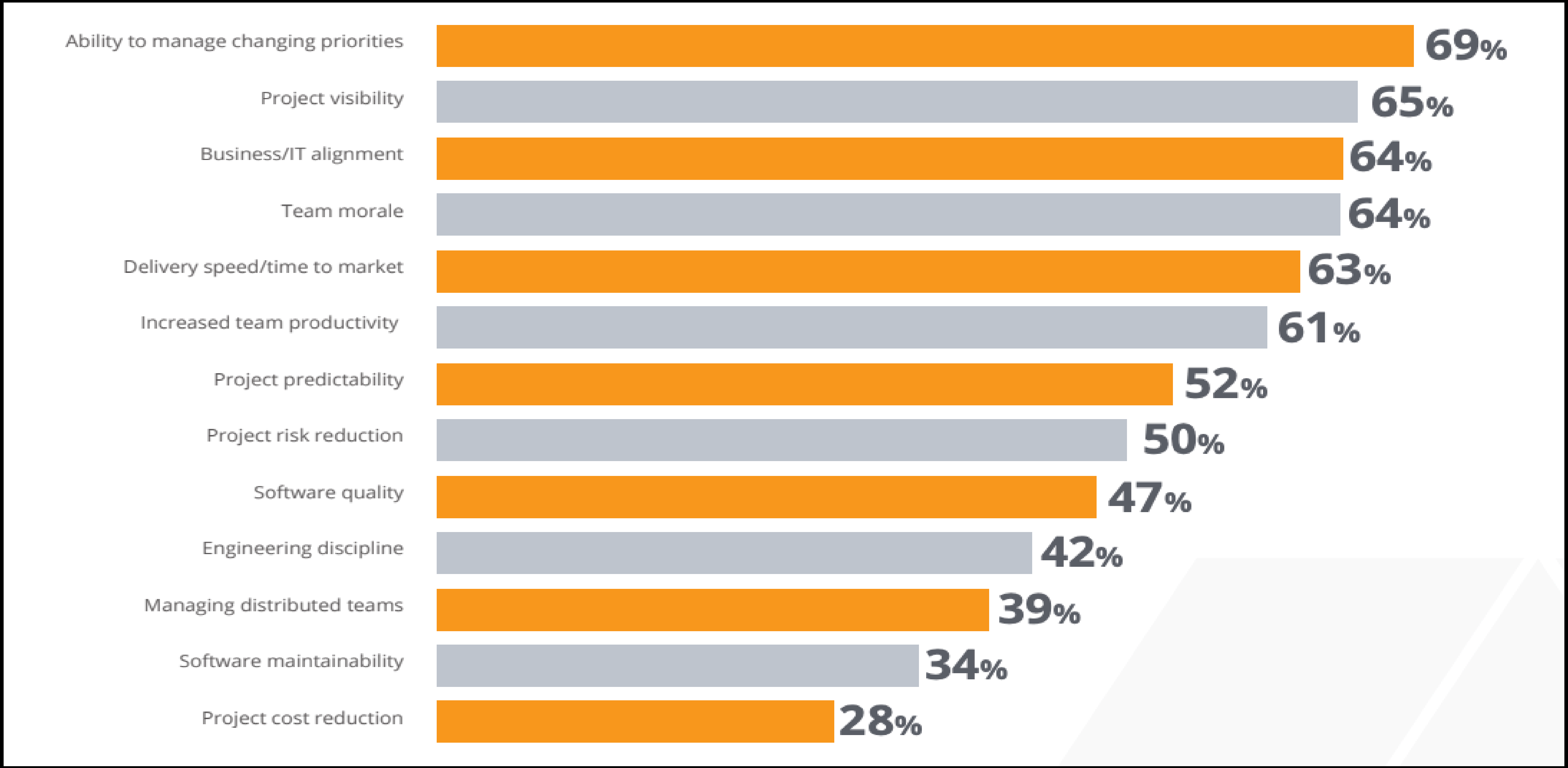
Agile Principles

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 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.
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.

2019 Survey - Why choose agile

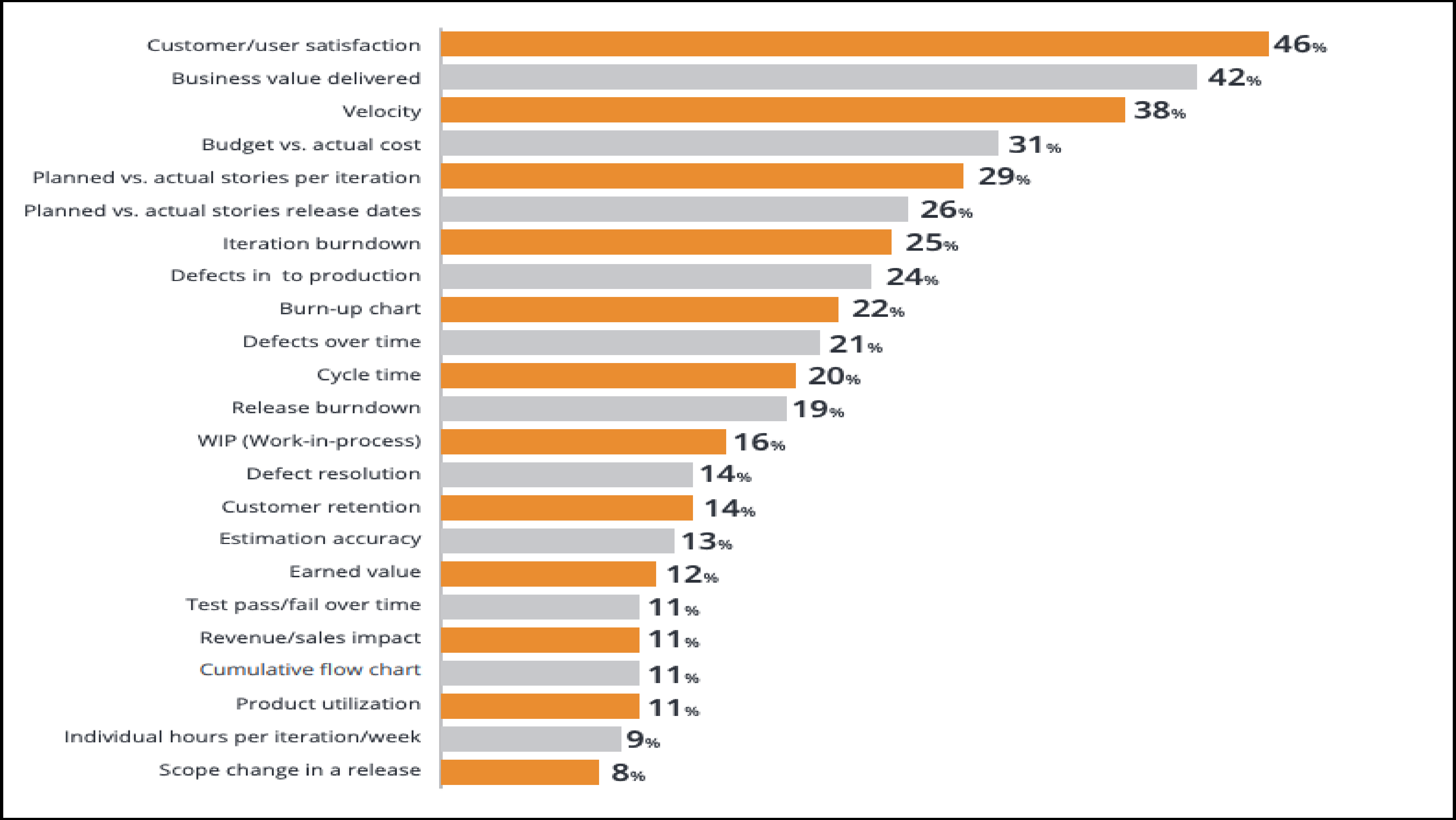


2019 Survey – Actual benefit



University of Nottingham
UK | CHINA | MALAYSIA

2019 Survey – How agile is measured

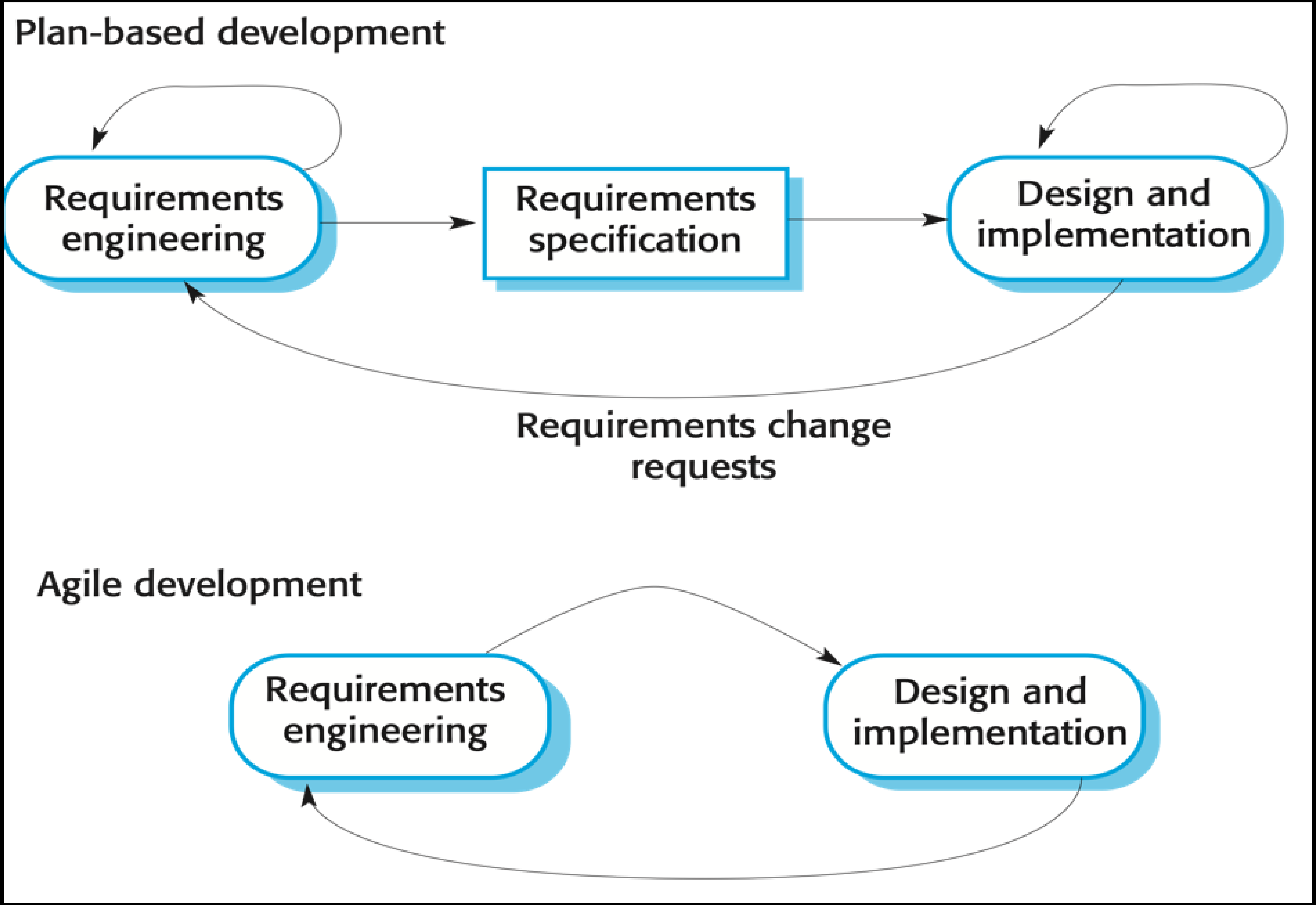


How to do Agile?

- So what can you make Agile?

- Each Stage and Activity
  - **agile requirements** - **user stories** not req docs
  - **agile design** - **prototypes** not spec docs
  - **agile implementation** - **test driven paired development**

Requirements: Plan vs Agile


Plan-based development
Requirements engineering
Requirements specification
Design and implementation
Requirements change requests
Agile development
Requirements engineering
Design and implementation

Scrum vs Kanban vs XP

- All three **framework** are designed to uphold the Agile Principles
  - they just have different focuses
- Scrum is about optimising by: time and delivery
- Kanban is about speed of delivery
- XP is about team effectiveness

eXtreme Programming

- XP is a fast ‘extreme’ approach to iterative development
  - New versions may be built everyday, or more often
  - Increments are delivered to customers every 2 weeks
  - All tests are applied before a build is accepted
  - Builds are only released when they pass all tests
- It takes many of the “faster techniques”
  - from every SE stage

eXtreme Programming

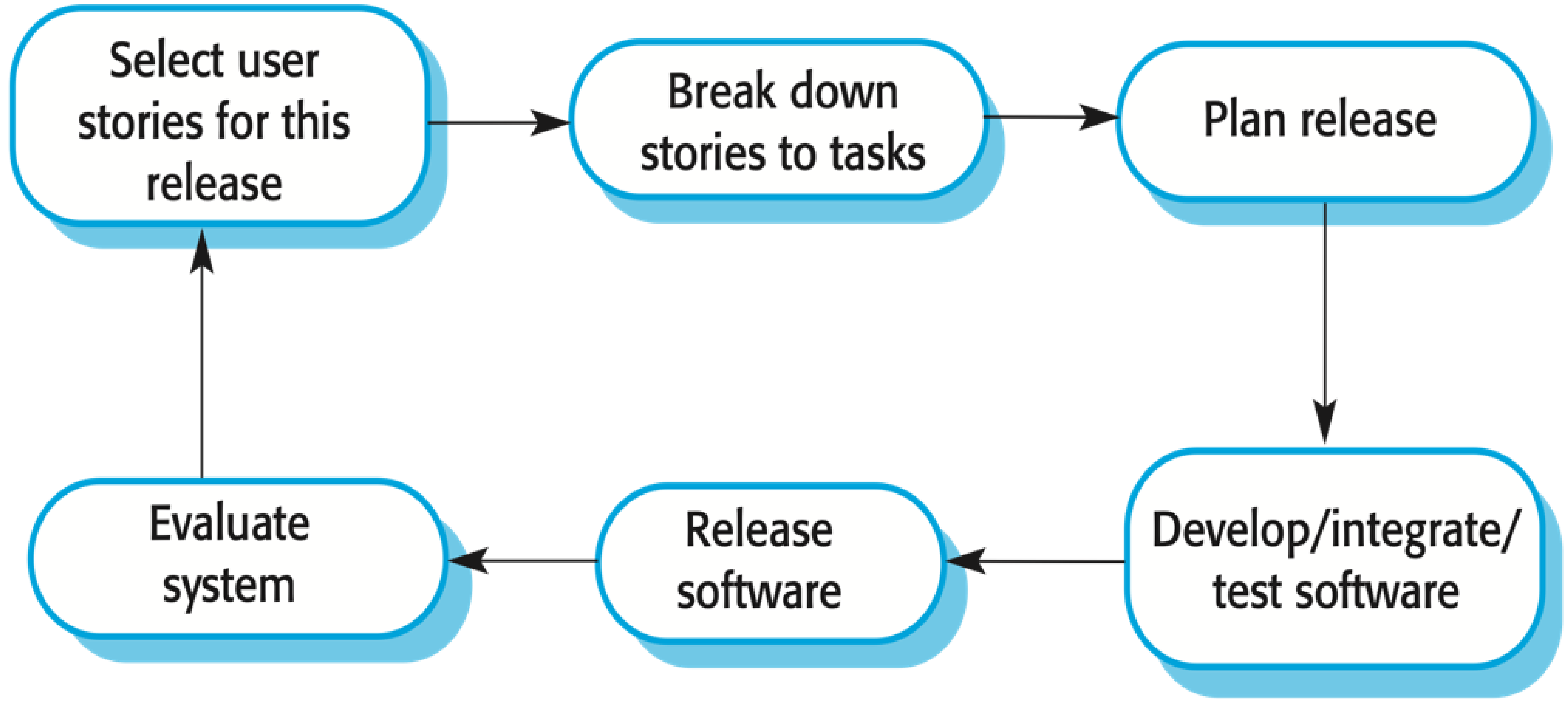

Select user stories for this release
Break down stories to tasks
Plan release
Develop/integrate/ test software
Release software
Evaluate system

Figure 3.3 - The extreme programming release cycle

eXtreme Programming

Principle or practice	Description
Incremental planning	Requirements are recorded on story cards and the stories to be included in a release are determined by the time available and their relative priority. The developers break these stories into development 'Tasks'. See Figures 3.5 and 3.6.
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.

eXtreme Programming

Pair programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
On-site customer	A representative of the end-user of the system (the customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

XP meets Agile Values?

- **Customer collaboration** means full-time customer engagement with the team.
- **Working software** is supported through small, frequent system releases.
- **Individuals and interactions** through pair programming, collective ownership and a process that avoids long working hours.
- **Responding to change** through regular system releases.

Scrum

- A lightweight framework that helps people, teams and organizations generate value through adaptive solutions for complex problems.
- Concentrate particularly on how to manage tasks within a **team-based** development environment.

University of Nottingham
UK | CHINA | MALAYSIA

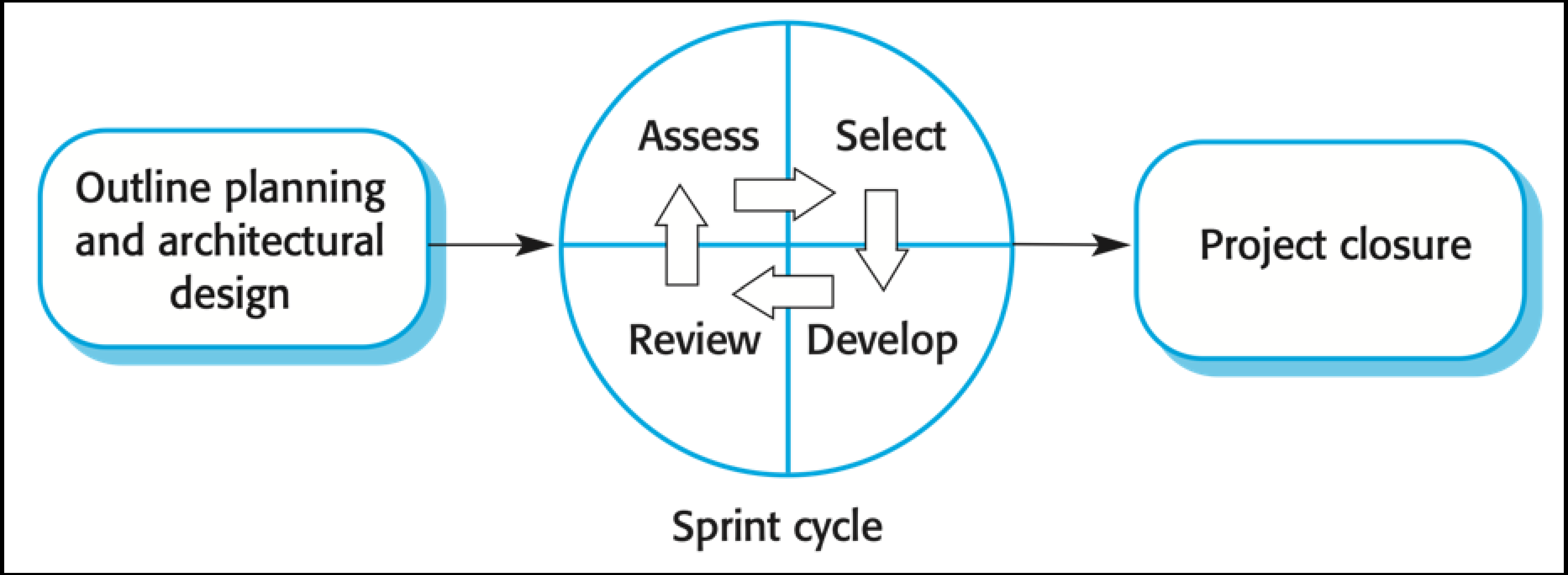
Scrum Roles

- The Scrum Team consists of **three** roles, namely a **ScrumMaster**, a **Product Owner**, and the **Team**
- **ScrumMaster**: the keeper of the scrum process, responsible for
  - Making the process run smoothly
  - Removing obstacles that impact productivity
  - Organizing and facilitating the critical meetings
- **Product Owner**: the sole person responsible for managing the Product Backlog
  - Expressing Product Backlog items clearly
  - Ordering the Product Backlog items to best achieve goals and missions
  - Optimizing the value of the work the Team performs
  - Ensuring that the Product Backlog is visible, transparent, and clear to all, and shows what the Team will work on further
  - Ensuring that the Team understands items in the Product Backlog to the level needed
- **Team:** comprises of analysts, designers, developers, tester, as relevant to the project

Scrum Artifact

- **Product Backlog:**
  - An ordered list of features that are needed as part of the end product
  - Can be updated at any time by the Product Owner
- **Sprint Backlog:**
  - Set of Product Backlog items selected for the Sprint, plus a plan for delivering the product Increment and realizing the Sprint Goal
- **Increment:**
  - Sum of all the Product Backlog items completed during a Sprint combined with the increments of all previous Sprints.

Scrum Method



- Phase 1) Plan and outline what you will achieve
- Phase 2) Set a series of ‘sprint cycles’ doing the work
- Phase 3) Document the current phase

Scrum Events

- **Sprint**
  - A sprint is **fixed time period** to achieve the current plan
    - e.g. 2-4 weeks
  - Sprints cycles are intermittent points
    - what will we achieve by the next release point?
  - Sprints involve continuous group communication
    - e.g. daily meetings
  - Sprints are managed by a Scrum Master
- **Daily Scrum Meetings**
  - A quick meeting conducted daily to quickly understand the work since the last Daily Scrum Meeting and create a plan for the next 24 hours
  - During the meeting, each team member explains:
    - What did he/she do yesterday that helped the Team meet the Sprint Goal?
    - What will he/she do today to help the Team meet the Sprint Goal?

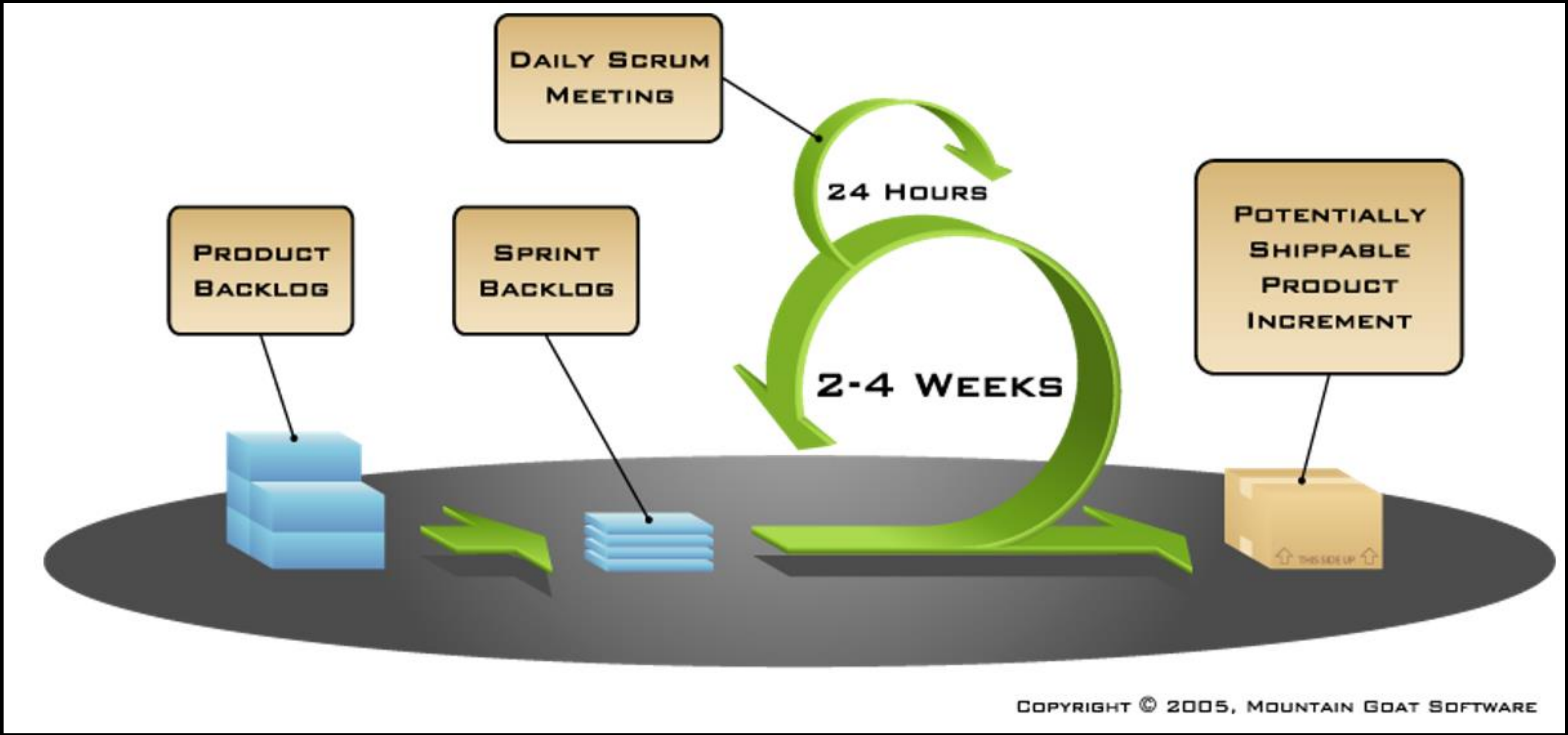
Scrum Events

- **Sprint Review:** held at the end of every Sprint
  - Attendees include the Scrum Team and key stakeholders, as invited by the Product Owner
  - The Product Owner explains what Product Backlog items have been completed during the sprint and what has not been completed
  - The Team discusses what went well during the Sprint, what problems it ran into, and how those problems were solved
  - The Team demonstrates the work that it has completed and answers questions, if any, about the Increment
  - The entire group then discusses on what to do next
  - The Scrum Team then reviews the timeline, budget, potential capabilities
  - The outcome of the Sprint Review is an **updated Product Backlog**

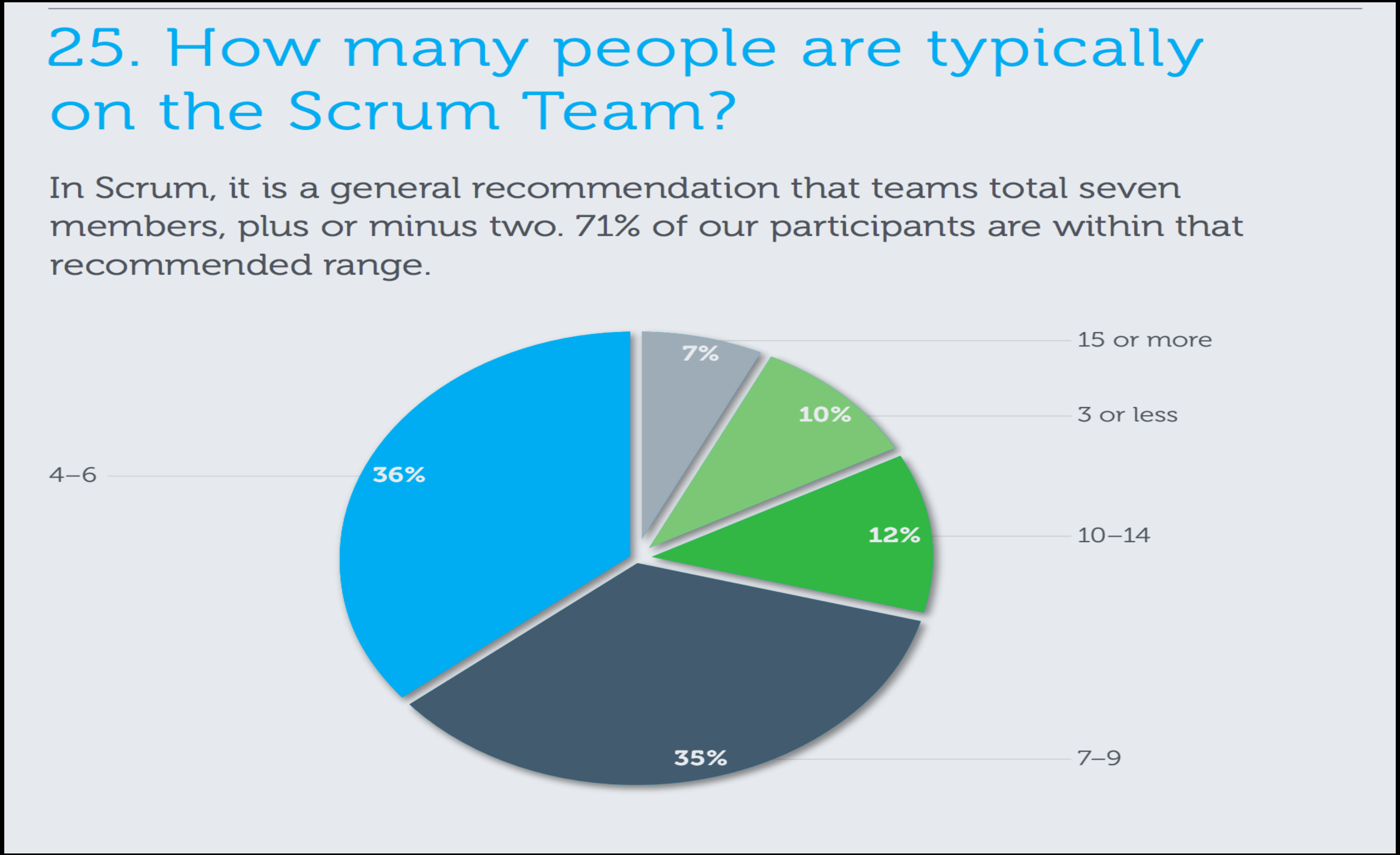
Scrum Events

- **Sprint Retrospective:** occurs after the Sprint Review
    - Combine the learnings from the last Sprint, with regards to people, relationships, process, and tools
    - Identify the major items that went well and potential improvements
    - Creation of a plan for implementing improvements to increase product quality

Scrum

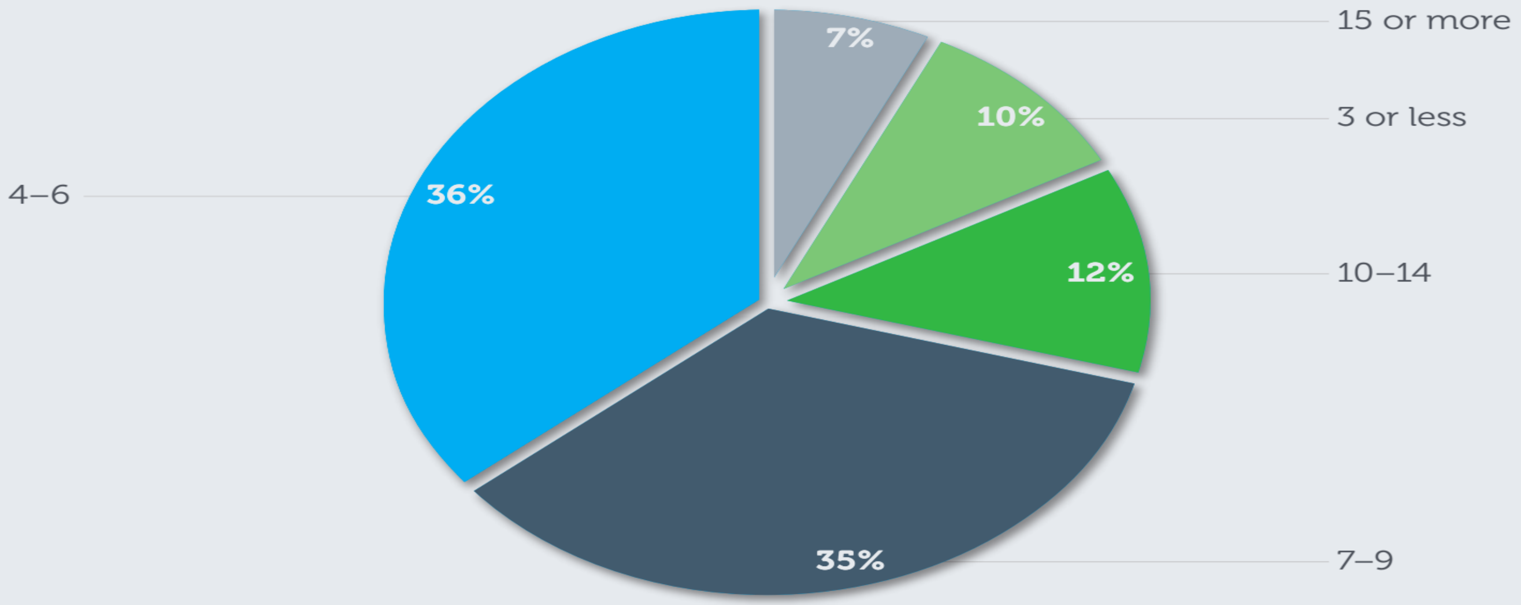

Daily Scrum Meeting
24 Hours
Product Backlog
Sprint Backlog
2-4 Weeks
Potentially Shippable Product Increment
This side up
Copyright © 2005, Mountain Goat Software

Scrum

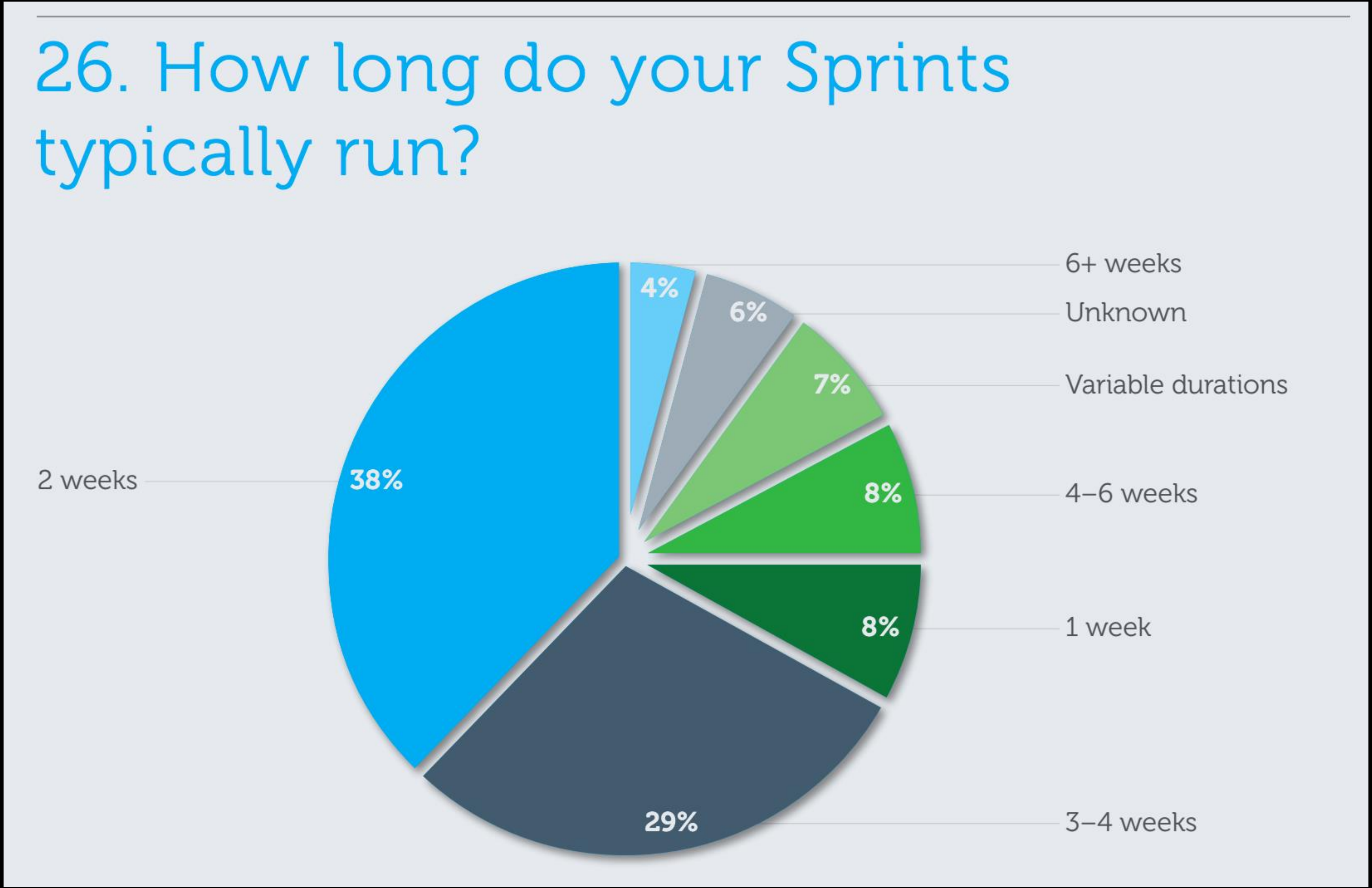


25. How many people are typically on the Scrum Team?

In Scrum, it is a general recommendation that teams total seven members, plus or minus two. 71% of our participants are within that recommended range.

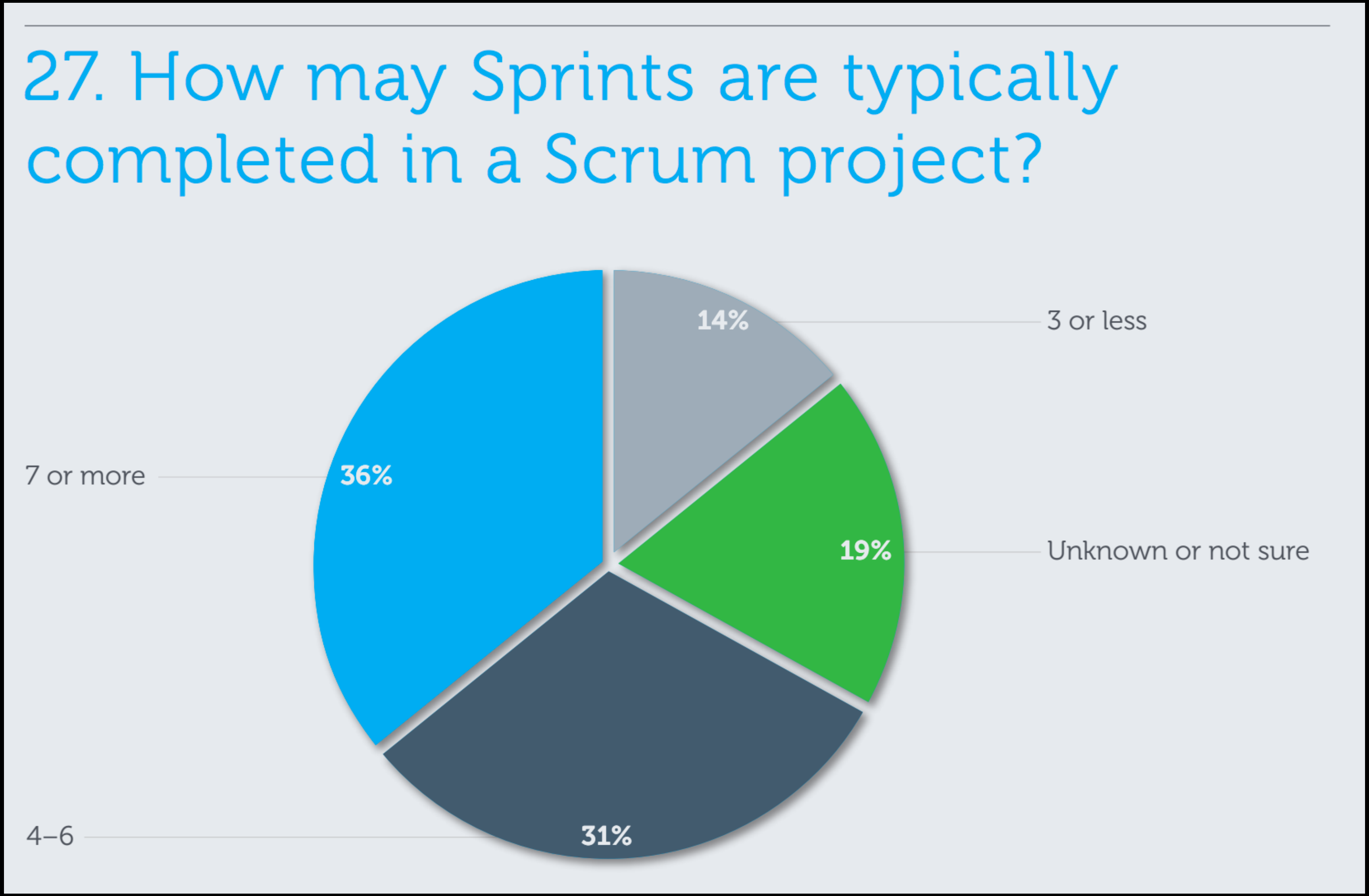


Scrum

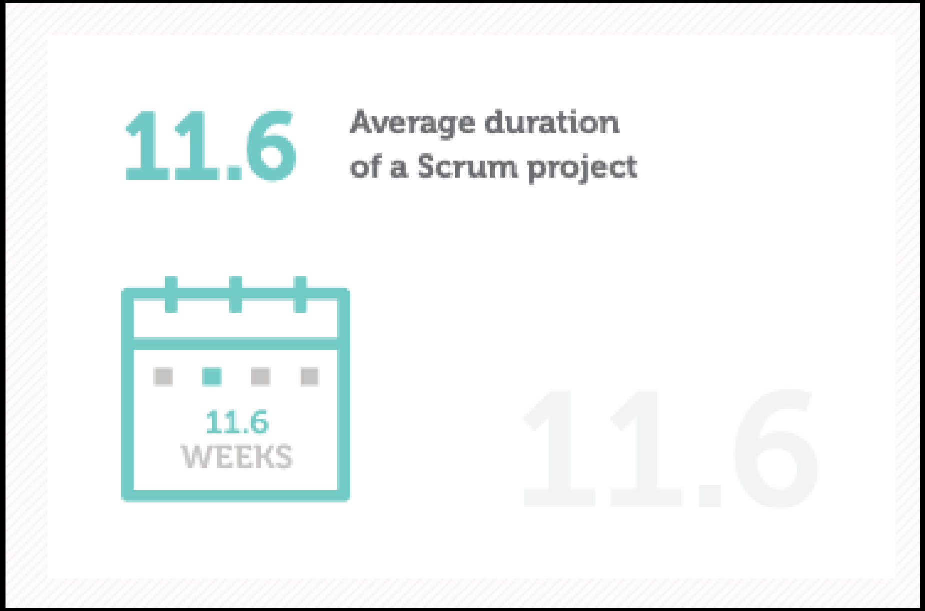

26. How long do your Sprints typically run?
6+ weeks
4%
Unknown
6%
Variable durations
7%
4–6 weeks
8%
1 week
8%
3–4 weeks
29%
2 weeks
38%

University of Nottingham
UK | CHINA | MALAYSIA

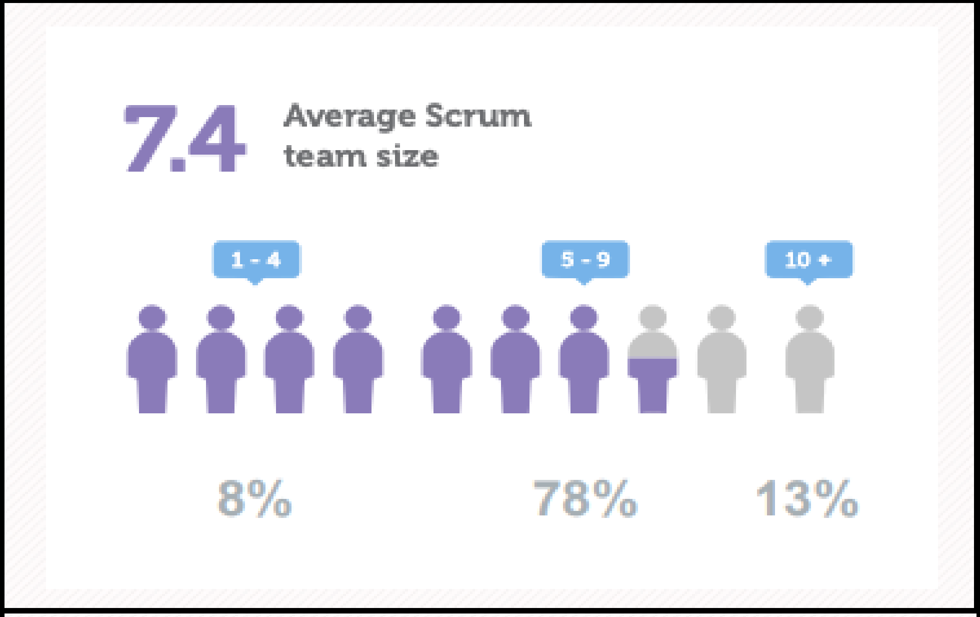
Scrum


27. How may Sprints are typically completed in a Scrum project?

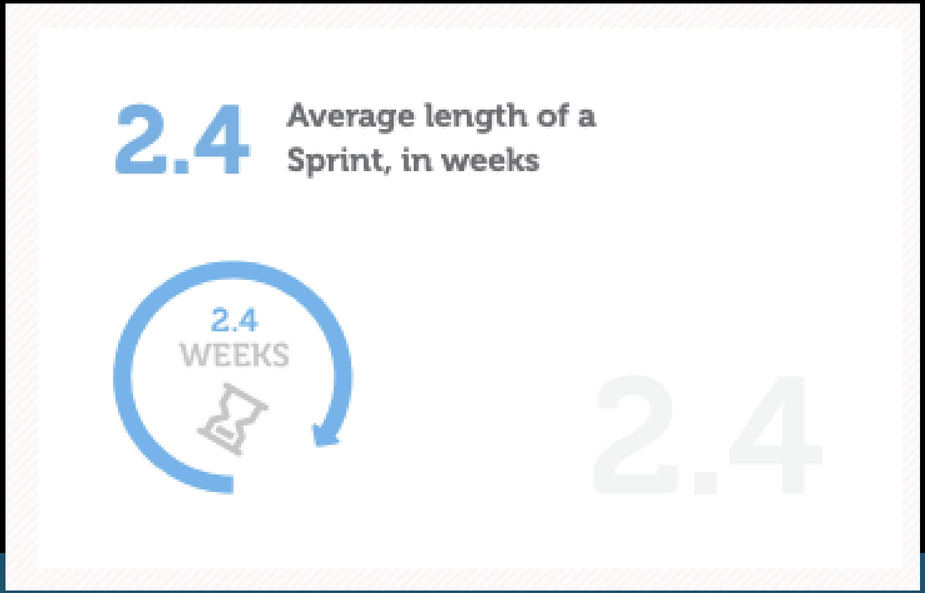
From the 2018 Scrum Survey



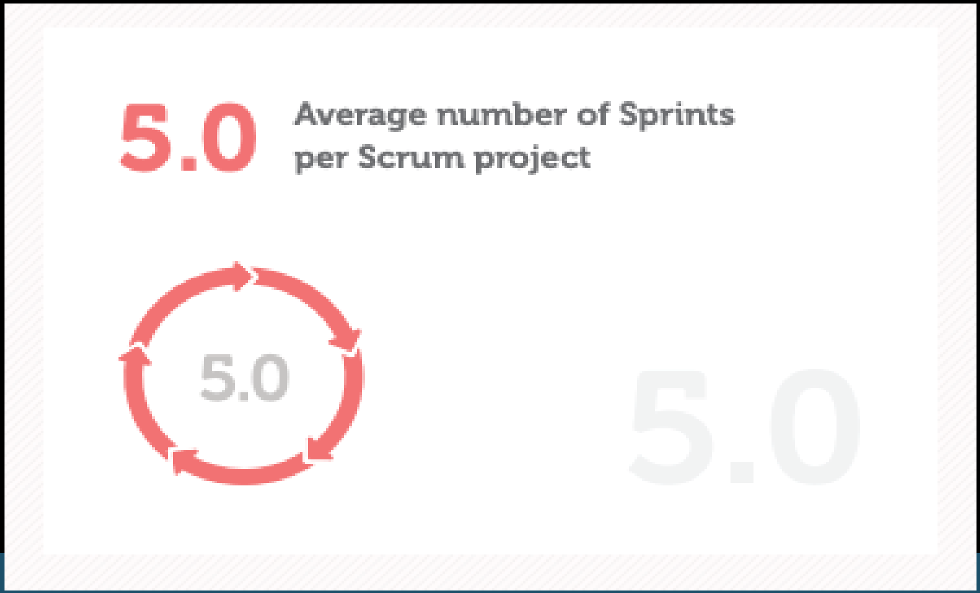
11.6 Average duration of a Scrum project



7.4 Average Scrum team size



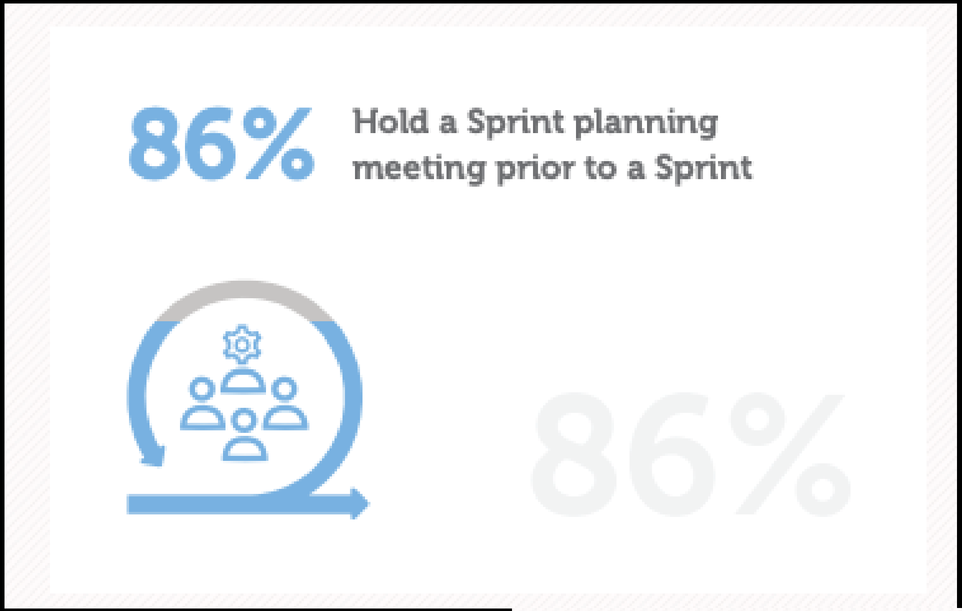
2.4 Average length of a Sprint, in weeks

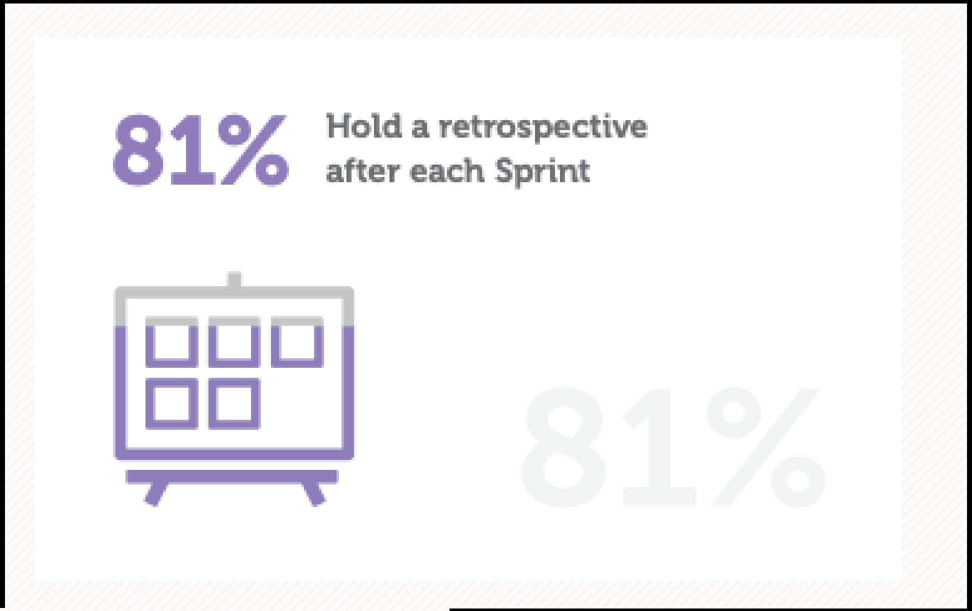


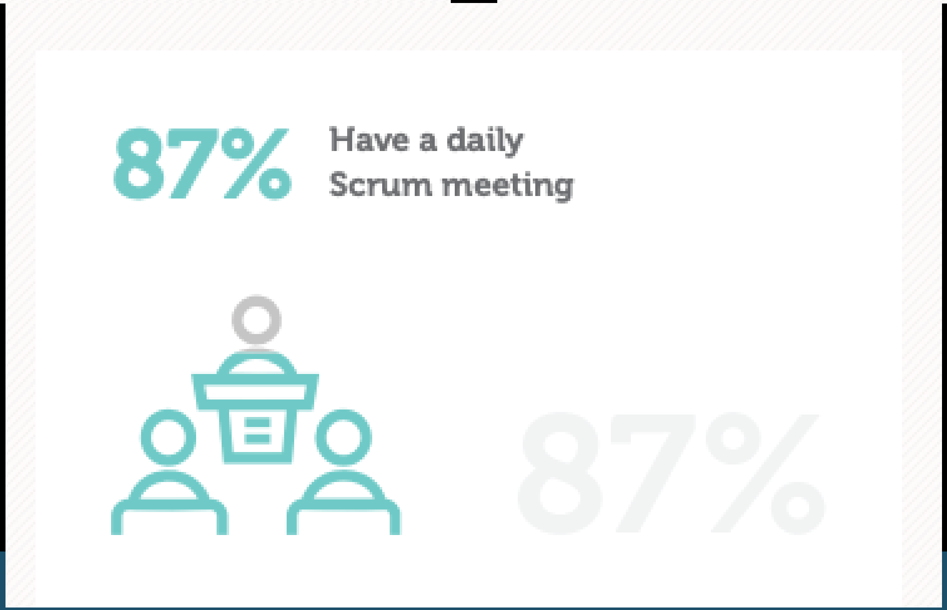
5.0 Average number of Sprints per Scrum project

University of Nottingham
UK | CHINA | MALAYSIA

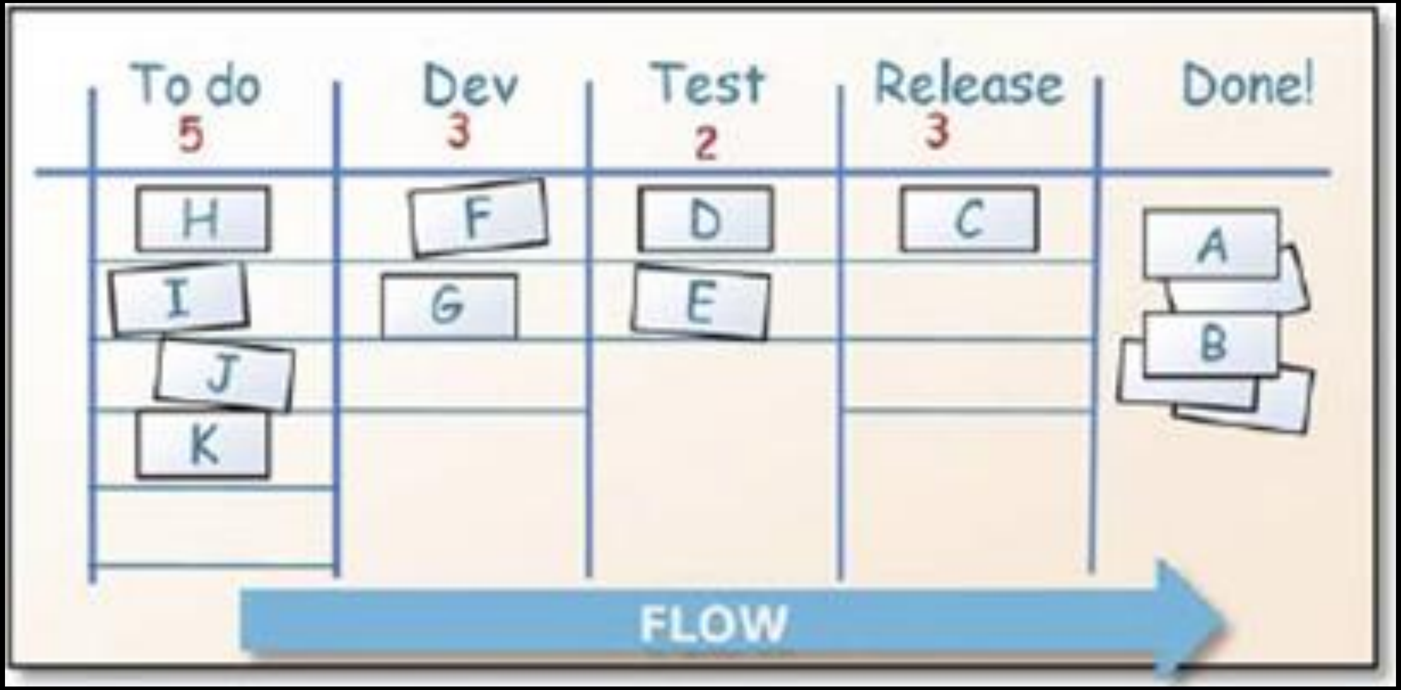
From the 2018 Scrum Survey


86% Hold a Sprint planning meeting prior to a Sprint


81% Hold a retrospective after each Sprint

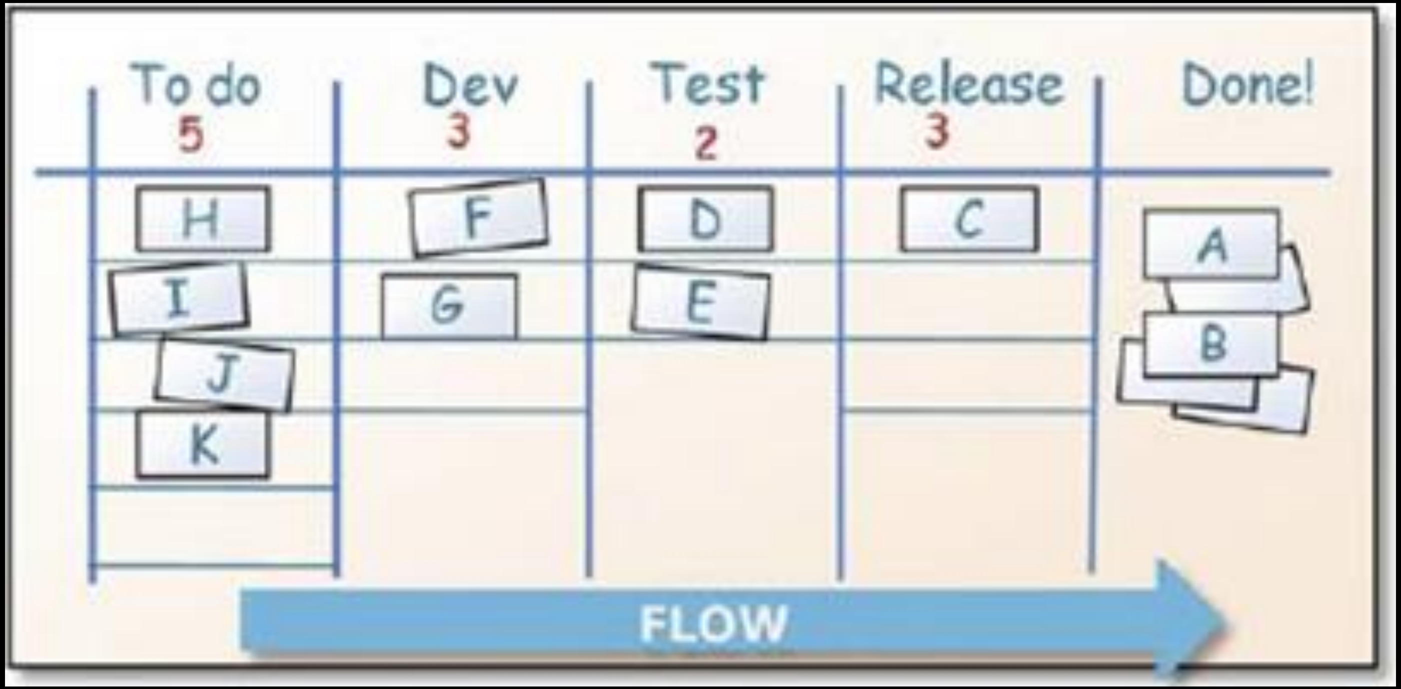

87% Have a daily Scrum meeting

Kanban



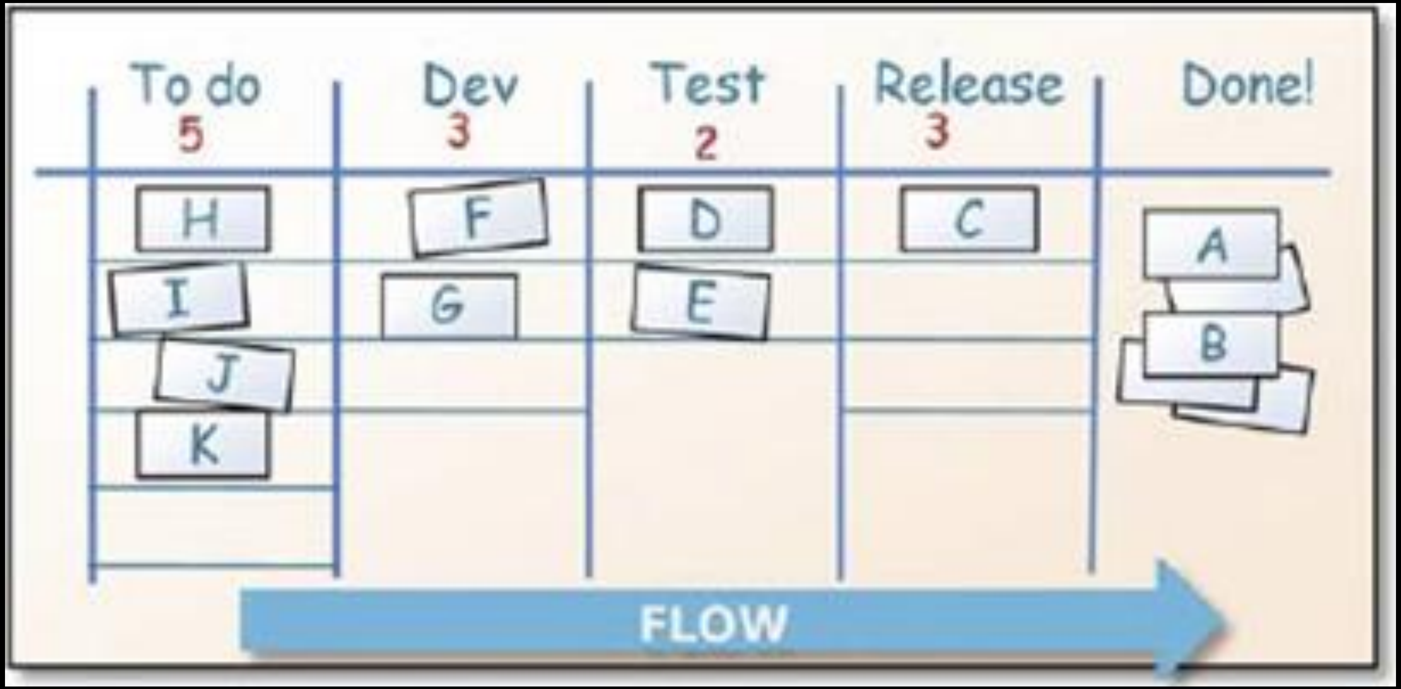
- Rather than what will we achieve in this cycle Kanban is - what shall we build next (passing jobs on)
  - the work is driven by an **updating priority list**
- You use a kanban board to manage current work

Kanban



- All features to build are kept in a list
  - items and their priority can be updated at any time
- When 1 is completed, it’s integrated into product, and released
  - Then you pick the next one

Kanban

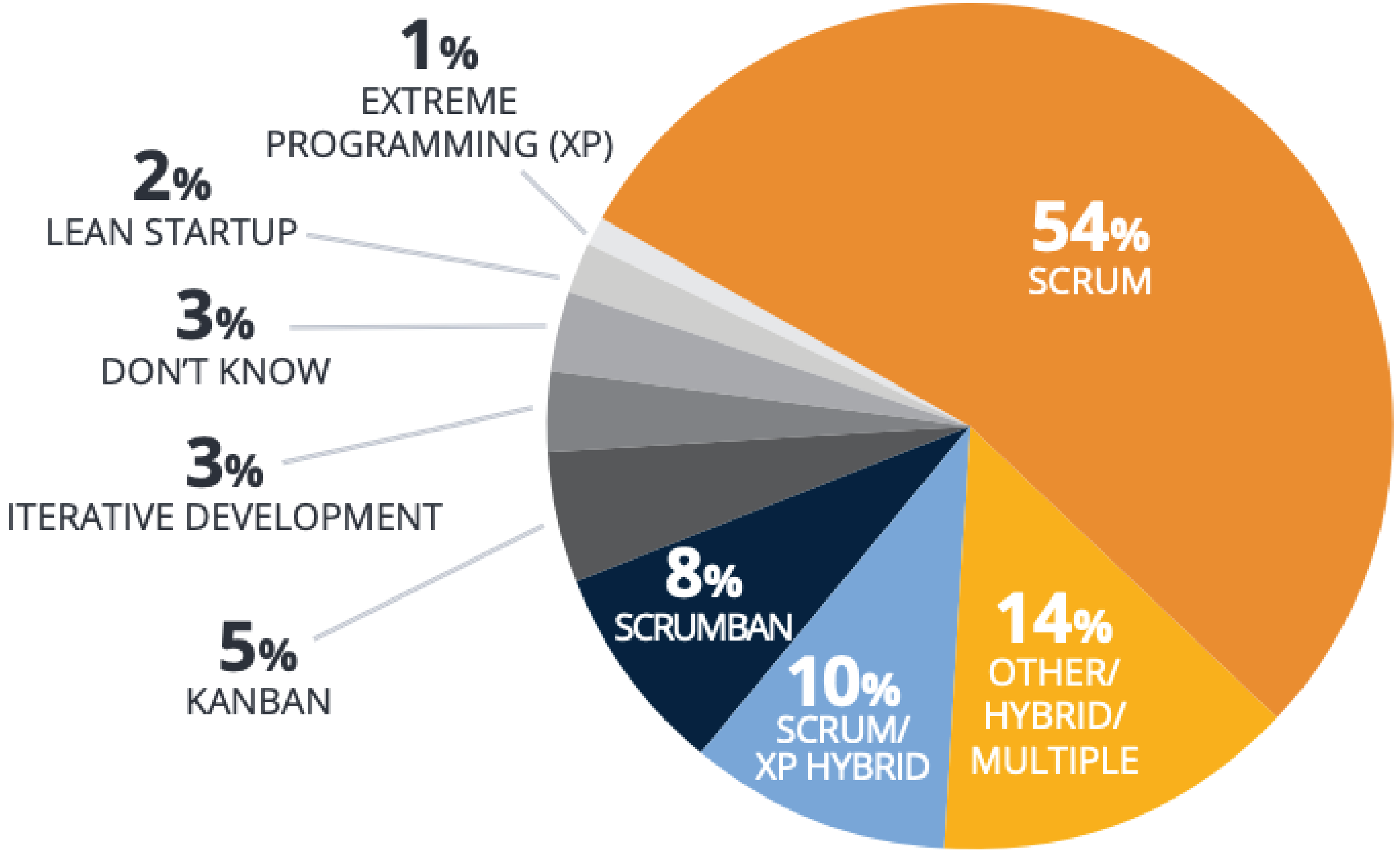


- Team effectiveness is managed by limiting the numbers in cols
- Release Cycles is a measure of time taken to build features
  - rather than a prescription of iterations in Scrum

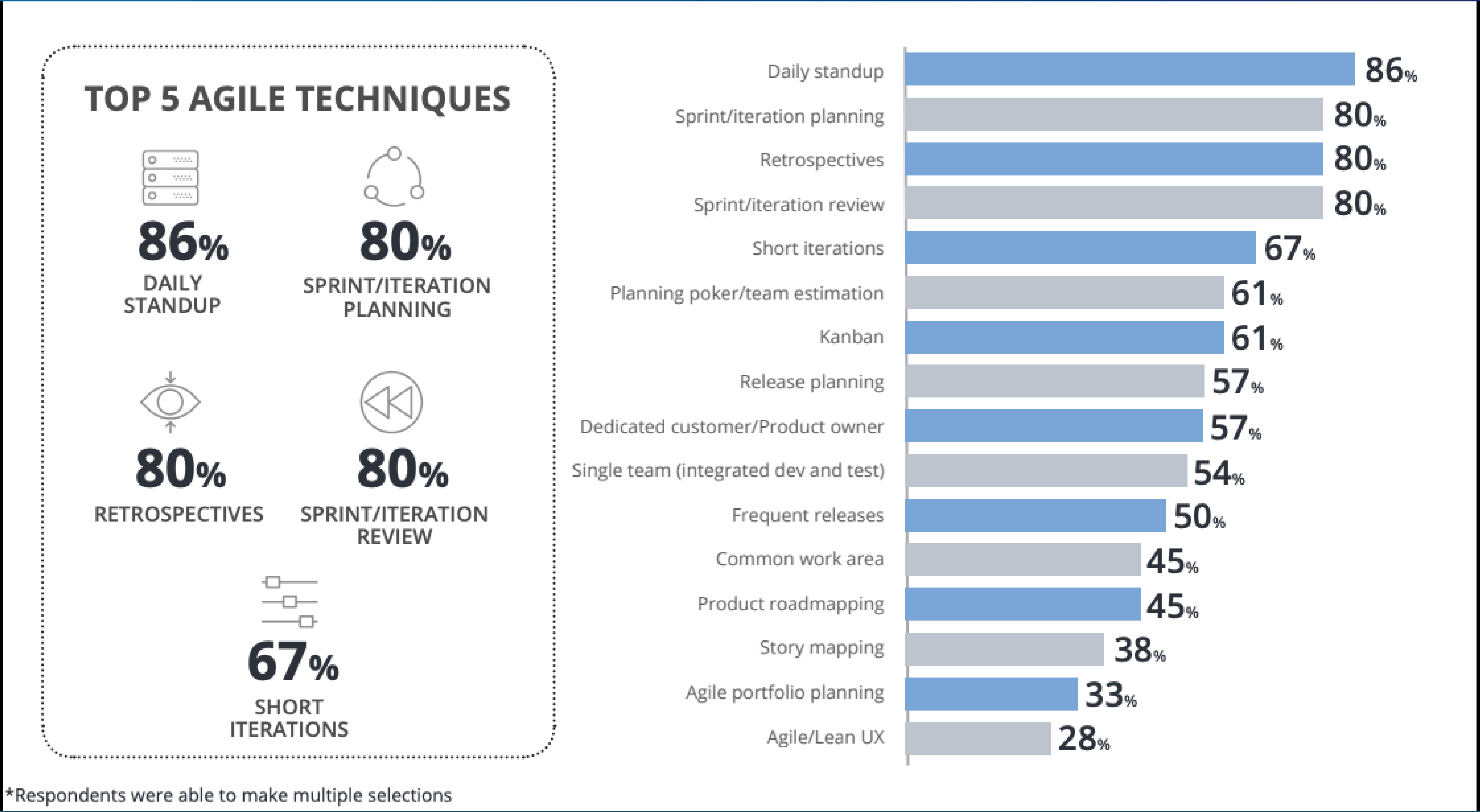
Combining Agile Methods

- Clearly these techniques aren't mutually exclusive
  - you can do paired programming in Scrum
  - you can use TDD in Kanban, Scrum, or XP
- Naturally mixtures exist
  - Scrumban
- Companies tend to pick a technique that works
  - based on team size, skill set, project clarity, etc

2019 Survey - Agile Approach



2019 Survey - Agile techniques



TOP 5 AGILE TECHNIQUES

86% DAILY STANDUP

80% SPRINT/ITERATION PLANNING

80% RETROSPECTIVES

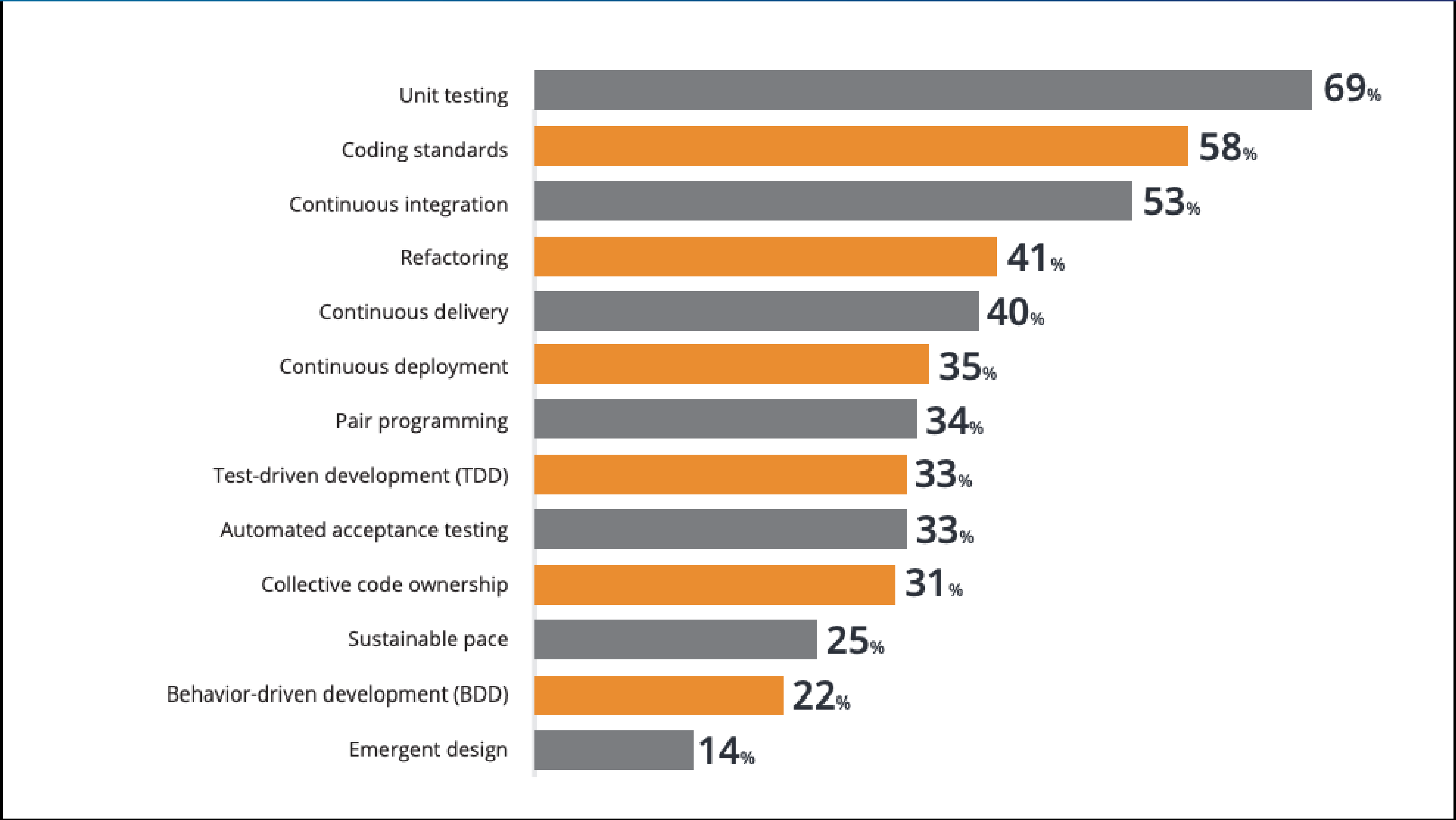
80% SPRINT/ITERATION REVIEW

67% SHORT ITERATIONS

Daily standup	86%
Sprint/iteration planning	80%
Retrospectives	80%
Sprint/iteration review	80%
Short iterations	67%
Planning poker/team estimation	61%
Kanban	61%
Release planning	57%
Dedicated customer/Product owner	57%
Single team (integrated dev and test)	54%
Frequent releases	50%
Common work area	45%
Product roadmapping	45%
Story mapping	38%
Agile portfolio planning	33%
Agile/Lean UX	28%

*Respondents were able to make multiple selections

2019 Survey - methods used



Agile vs Traditional - Comparison

Traditional Methods	**Agile Methods**
Waterfall, V-Model	Spiral, Iterative Models
Requirements Tables	User Stories, Personas
Detailed UML/Specs	Rapid Prototyping
Integration Testing Phase	TDD, Paired Coding
Formal User Testing	Frequent Client Interaction
Separate Skill Teams	Integrated skills teams

University of Nottingham
UK | CHINA | MALAYSIA

Waterfall Pros

- Forward & backward **planning and implementation is easy**.
- The waterfall model is simple to use and easy to understand. It does not require any special training for project managers or employees. It has an **easy learning curve**.
- Being rigid in nature, it is **easy to manage** the waterfall cycle. Each phase has fixed deliverables and a review process.
- **Less complexity** as the phases does not overlap.
- Works **well for small projects** where we have fixed and crystal-clear requirements.
- Processes and results are **well-documented**.
- It is **easy to measure the progress** as the start and endpoints of each phase are predetermined.
- There are **no financial surprises**. Once the requirements are fixed, the final cost can be calculated before starting the development.

Waterfall Cons

- As all the requirements must be clearly known before starting the development, it **delays the project**.
- **Requires extensive research** into the user needs.
- Low flexibility makes it **difficult to accommodate any such changes**, especially when the product needs to be re-engineered to a large extent.
- **Slow delivery times**. The customer is not able to see the product until it is fully completed.
- The **client is not informed** well about the health of the project.
- **High risk and uncertainty** are involved in the waterfall model as there is too much room for issues to remain unnoticed until the project comes near to completion.
- It is difficult to **measure the progress** within each phase.

Agile Pros

- **Great adaptability.** Agile is very flexible in dealing with the changes in customer needs and priorities.
- Allows to **constantly refine and re-prioritize the overall product backlog**
- Agile methodology offers a great degree of **stakeholder engagement**.
- Fixed schedule sprints of one to four weeks allow for **early and predictable delivery**.
- Valuable **customer feedback** is gained **early** in the project and changes to the product can be made as required.
- By producing **frequent builds**, any misalignment with the customer requirements can also be detected and fixed early.
- Agile **promotes teamwork**.
- In an agile project, there is room for **continuous improvement**.

University of Nottingham
UK | CHINA | MALAYSIA

Agile Cons

- It is often seen that Agile teams have a **tendency to neglect documentation**.
- If the initial architecture and design are weak, then **frequent refactoring** is required.
- When compared to the waterfall, Agile is **difficult to practice**. The team members must be well versed in Agile concepts.
- Due to time-boxed delivery and frequent re-prioritization, there are chances for a few features to not get delivered in the allocated timeline. This can lead to **additional sprints and additional costs**.

Guide to Agile Methods

- A guide on how to do lots of SE activities in agile ways

  - http://guide.agilealliance.org/guide/user-stories.html
  - http://guide.agilealliance.org/guide/simple-design.html
  - http://guide.agilealliance.org/guide/tdd.html
  - http://guide.agilealliance.org/guide/acceptance.html

- Also in team/project management

  - http://guide.agilealliance.org/guide/estimation.html
  - http://guide.agilealliance.org/guide/leadtime.html
  - http://guide.agilealliance.org/guide/daily.html
  - http://guide.agilealliance.org/guide/kanban.html

Agile Methods

- Especially valuable for
  - small or medium sized projects for release/sale
  - custom system development with an involved customer
- BUT
  - it's not always easy to have an involved customer long term
  - team members don't always have the right personalities
  - important tasks take time, and can be forgotten
  - company change can be slow

University of Nottingham
UK | CHINA | MALAYSIA

Agile or Plan?

- Is it important to have a very detailed specification and design before moving to implementation? Plan-driven

- Is an incremental delivery strategy, where you deliver the software to customers and get rapid feedback from them, realistic? Agile

- Is the software small enough to be built by a small co-located team? Or can be broken down so? Agile

Agile or Plan?

- Does the system require lots of careful analysis before the implementation can begin?

Plan-driven

- Does the stakeholder need a detailed documentation?

Plan-driven

- Are your tools/IDEs good for Agile development (keeping track of change)?

Agile

University of Nottingham
UK | CHINA | MALAYSIA

Agile or Plan?

- Do you need clear documentation to communicate between project partners, or remote colleagues?

Plan-driven

- Is the current culture, and all other projects, very process oriented?

Plan-driven

- Does your team have all the right requirements, design, testing, skill-sets?

Agile

- Is your project or system subject to external regulation or formal (e.g. safety critical) standards?

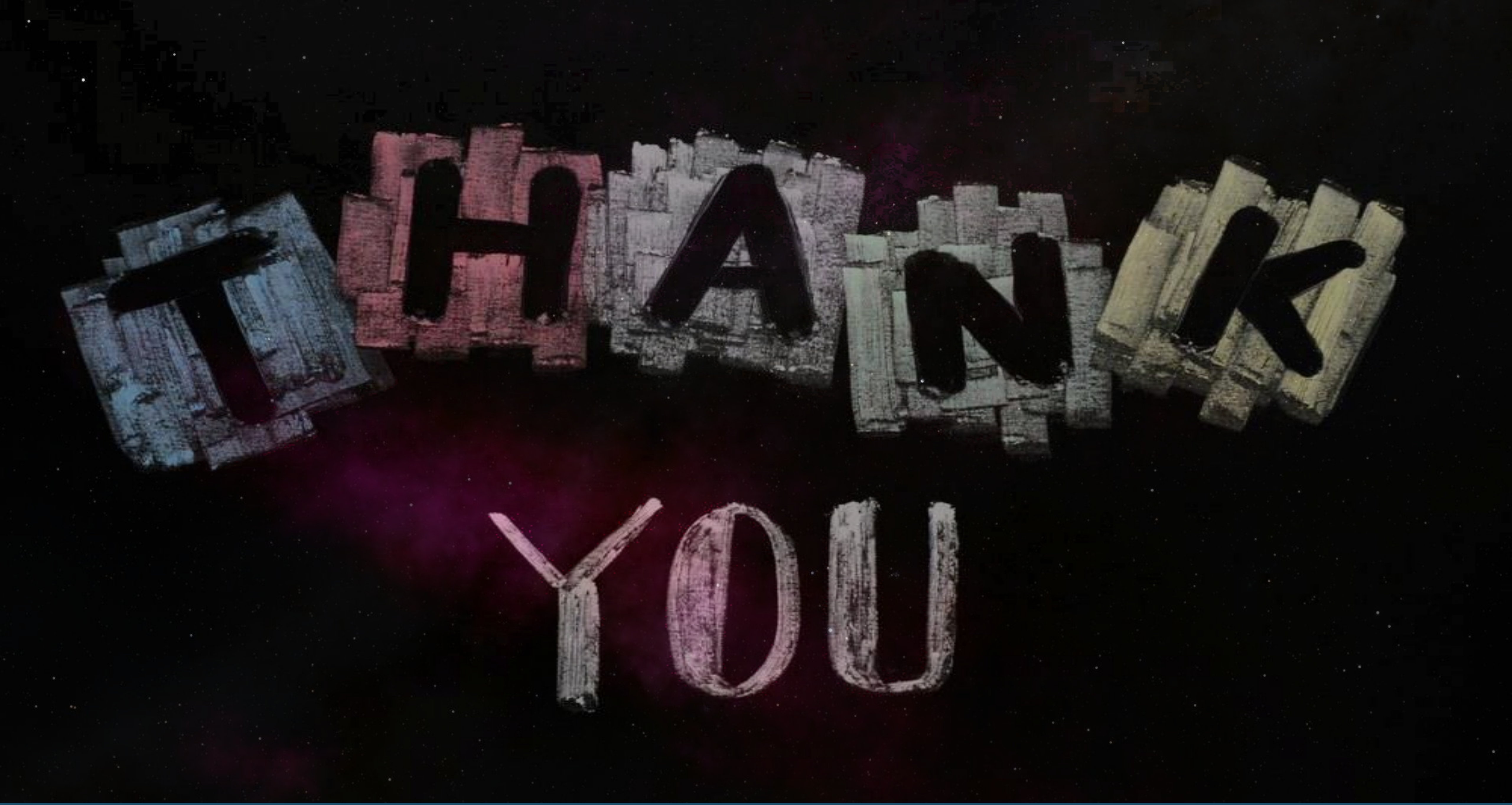
Plan-driven

Remarks

- It’s not Agile or Plan - overall
- It’s Agile or Plan - for each stage
  - » Overall, you might mix things
- What can be done agile? What needs formal approaches?
- Is Agile faster and cheaper than rigorous plan approaches?

Summary

- Agile techniques were designed because all the rigour in SE became too big and too slow
- Agile techniques **still have rigour**, but in a fast adaptive way that is responsive to change
- All sorts of ways to be Agile - not just a big overall approach
- **Following a strict Agile method misses the point about being Agile**


THANK
YOU