ICT393
Advanced Business Analysis and Design

Topic 2

Agile Development Methodologies

Murdoch
UNIVERSITY

Readings and Resources

- Fowler, M. (2005) *The New Methodology.* Available from: http://www.martinfowler.com/articles/newMethodology.html
- James, M. and Walter, L. (2017) *Scrum Reference Card*. Available from https://www.collab.net/sites/default/files/uploads/CollabNet_scrumreferencecard.pdf
- Online video: *Introduction to SCRUM*. Available from: http://scrumtrainingseries.com/Intro_to_Scrum/index.html

Learning Objectives

After completing this topic you should be able to:

- Understand what agile development is
- Describe how agile development approaches relate to traditional system development methodologies
- Discuss the potential problems with agile development
- Describe several examples of agile development approaches

What is Agile Development?

Agile development refers to a group of software development methodologies that are based on iterative development, where requirements and solutions evolve through collaboration between self-organising cross functional teams.

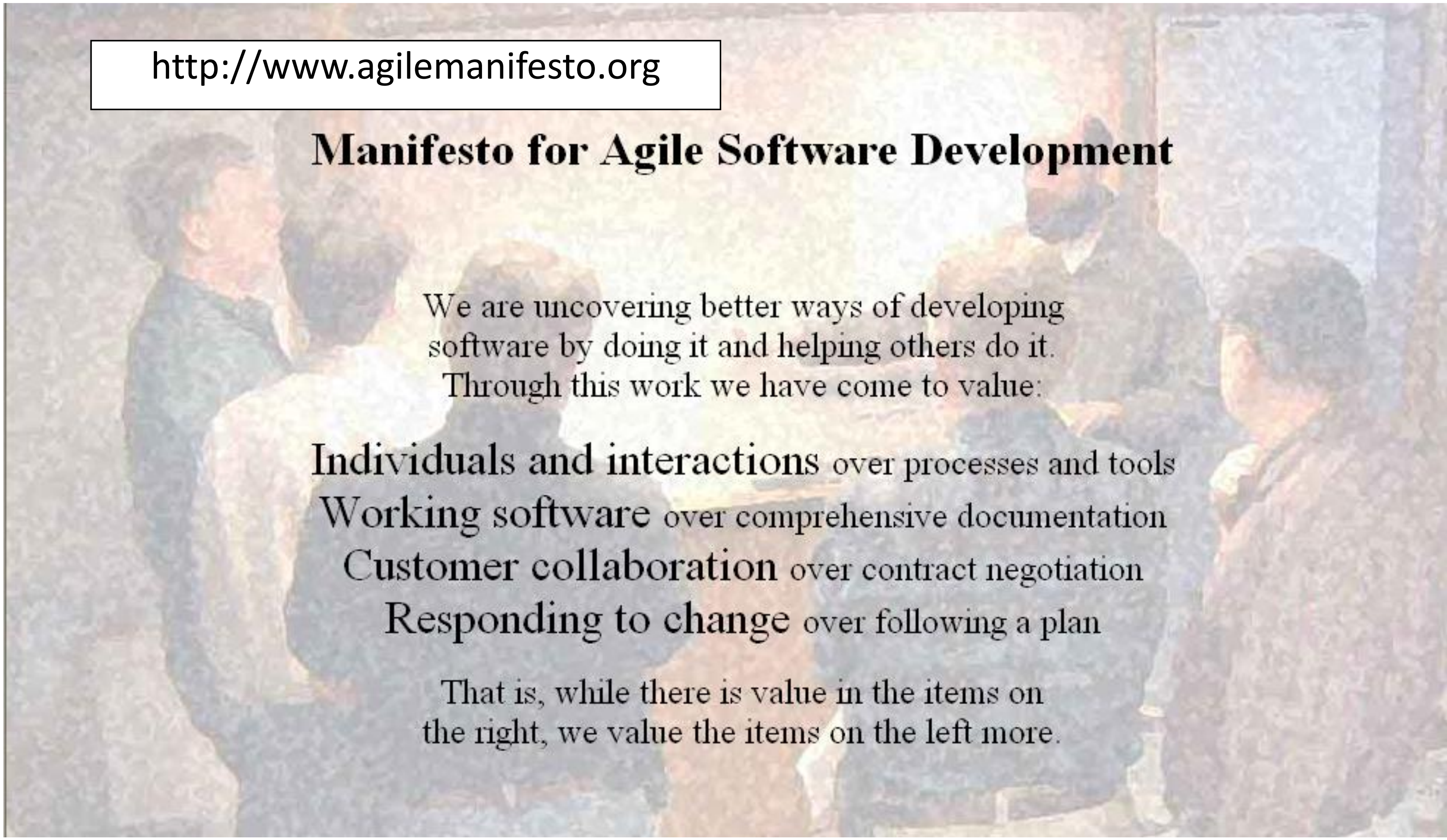
- **What are self-organising cross functional teams?**

What is Agile Development?

Characteristics of agile development include:

- A project management process that encourages frequent inspection and adaptation
- A leadership philosophy that encourages teamwork, self-organisation and accountability
- Development practices that allow for rapid delivery of high-quality software
- A business approach that aligns development with customer needs and company goals

The Agile Manifesto



http://www.agilemanifesto.org

Manifesto for Agile Software Development

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

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

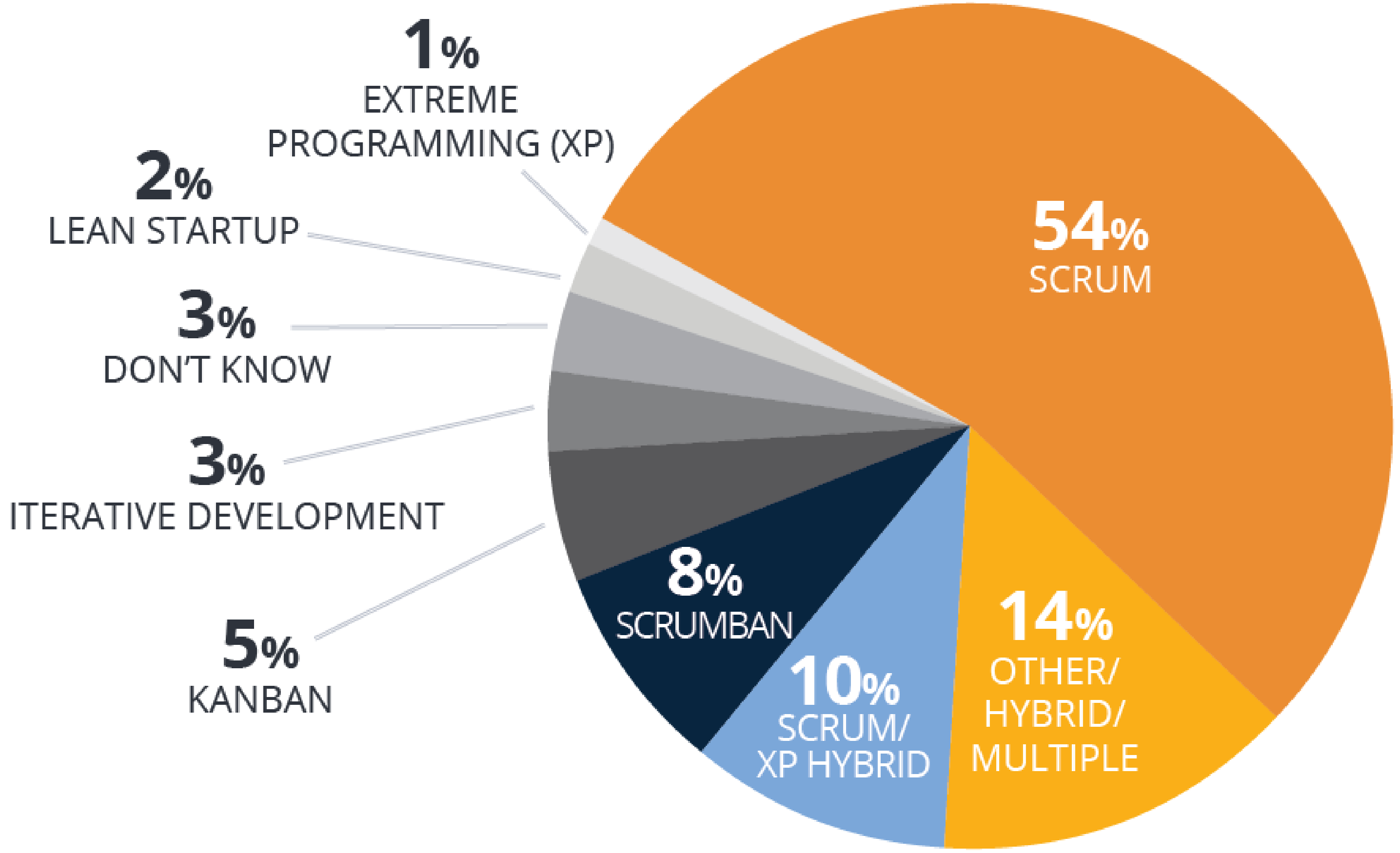
What Makes a Method Agile?

- Incremental (small releases, rapid cycles)
- Cooperative (communications between developers and customers)
- Straightforward (method is easy to learn and modify, well documented)
- Adaptive (embrace changes, even at last moment)

Examples of Agile Approaches

- Scrum
- Extreme Programming (XP)
- Agile Unified Process (AUP)
- Dynamic Systems Development Method (DSDM)
- Crystal family of methodologies
- Lean Software Development
- Internet-Speed Development (ISD)

Scrum is the most commonly used agile approach



CollabNet VersionOne (2019) *The 13th Annual State of Agile Report*.

Scrum

- A quick, adaptive, and self-organizing agile methodology – used for software development and more broadly for other kinds of projects
- Concentrates on the management aspects of software development. Development is divided into iterations called sprints (often 2 to 4 weeks)
- Focuses primarily on the team level - team exerts total control over its own organisation and work processes
- Uses a product backlog as the basic control mechanism - prioritised list of user requirements used to choose work to be done during a Scrum project

Scrum Organisation

Main roles include:

- Product owner:
  - The client stakeholder for whom a system is being built
  - Maintains the product backlog list
- Scrum master (c.f. project manager) - person in charge of a Scrum project
- Scrum team or teams:
  - Small group of developers (approx 7)
  - Set their own goals and distribute work among themselves

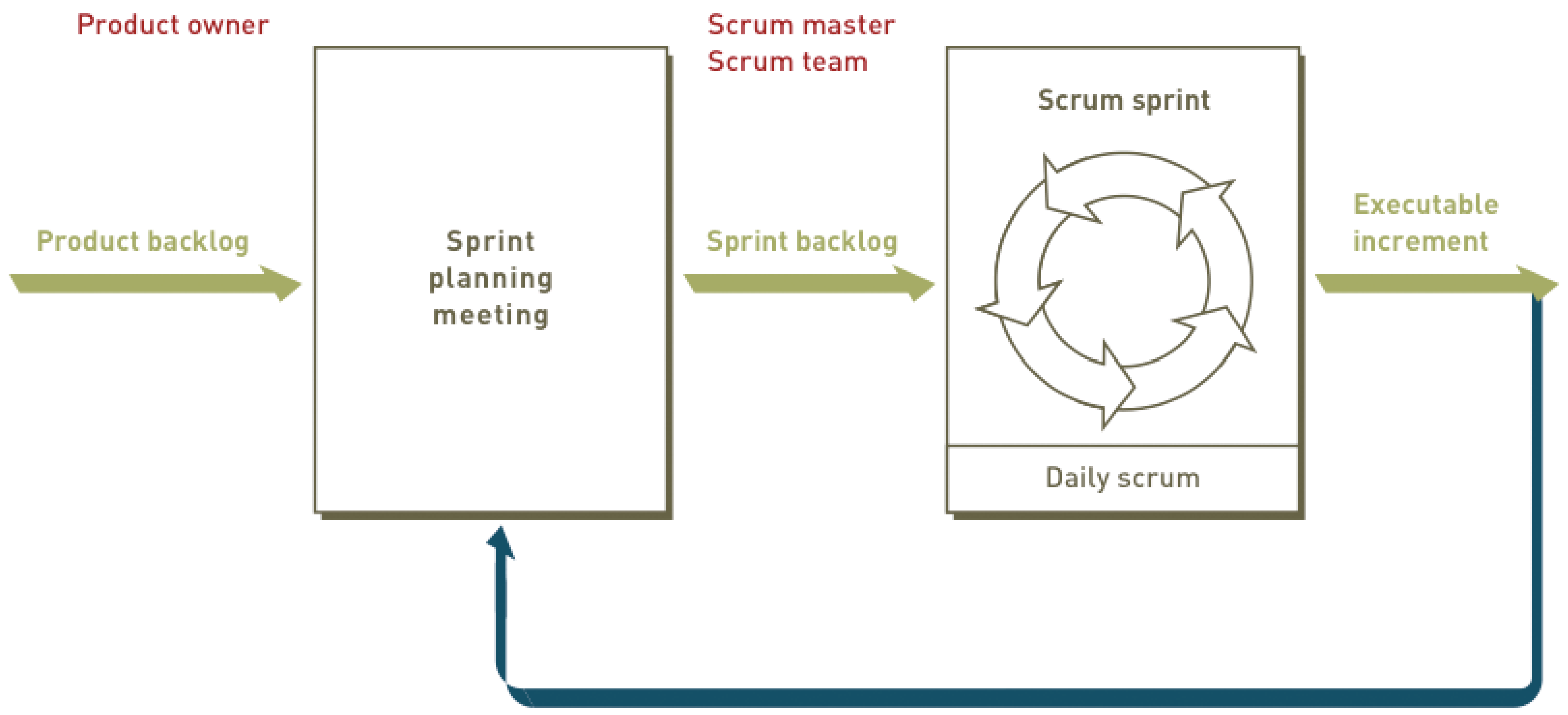
Scrum Practices

- Sprint
  - The basic work process in Scrum
  - A time-controlled mini-project
  - Firm time box with a specific goal or deliverable

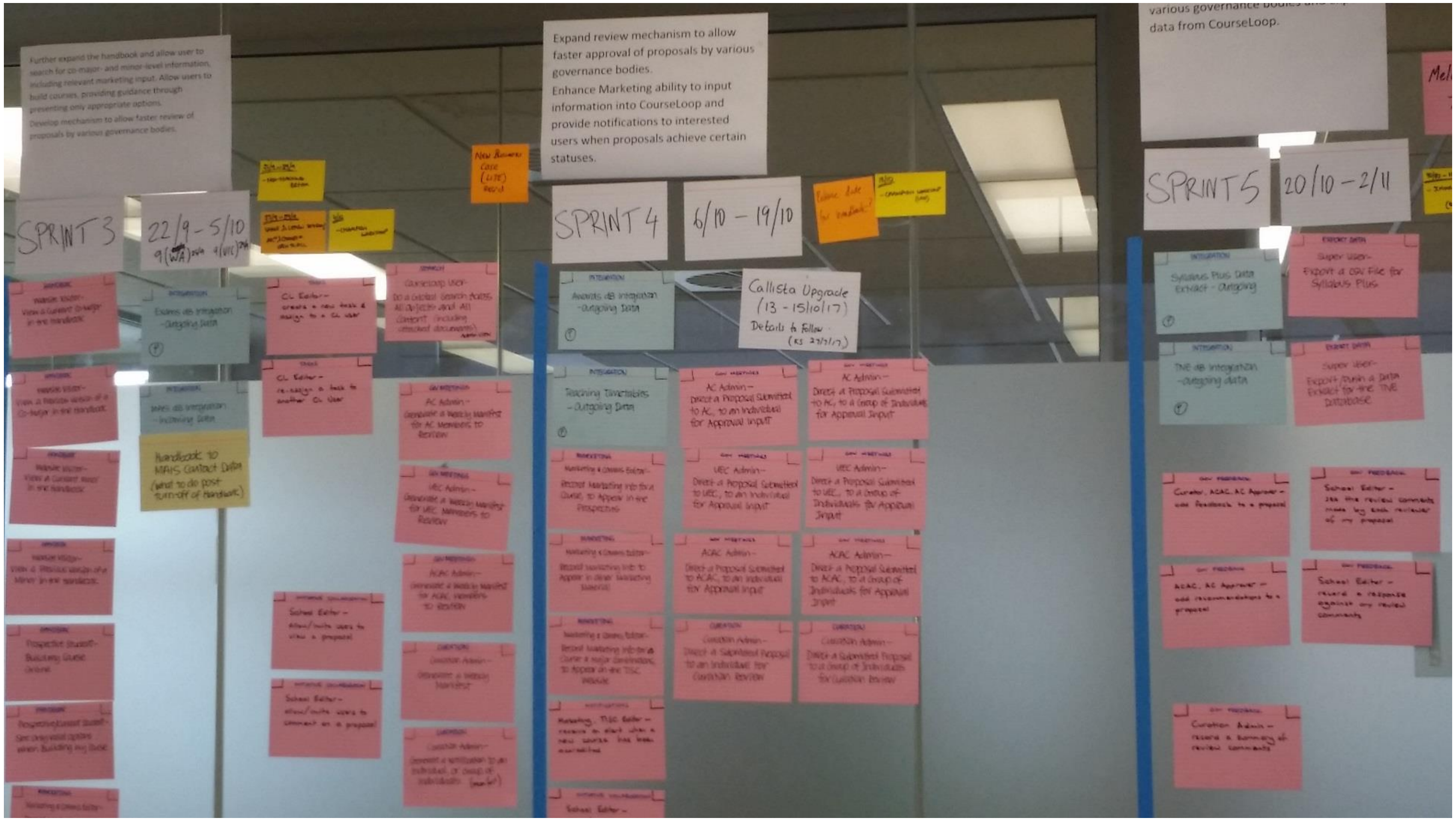
- Parts of a sprint
  - Begins with a one-day planning session
  - A short daily Scrum meeting to report progress
  - Ends with a final half-day review

NOTE: See James and Walter (2017) for more detail

Scrum Development Process



Scrum Use at Murdoch


Expand review mechanism to allow faster approval of proposals by various governance bodies.
Enhance Marketing ability to input information into CourseLoop and provide notifications to interested users when proposals achieve certain statuses.
data from CourseLoop.
SPRINT 3
22/9 – 5/10
SPRINT 4
6/10 – 19/10
SPRINT 5
20/10 – 2/11
New Business Case (LITE) Recd
Callista Upgrade (13 – 15/10/17) Details to Follow
Handbook to MAIS Contact Data
Awards dB Integration – Outgoing Data
Teaching Timetables – Outgoing Data
Syllabus Plus Data Extract – Outgoing
Super User – Export a CSV File for Syllabus Plus
AC Admin – Direct a Proposal Submitted to AC, to an Individual for Approval Input
UEC Admin – Direct a Proposal Submitted to UEC, to an Individual for Approval Input
ACAC Admin – Direct a Proposal Submitted to ACAC, to an Individual for Approval Input
Curation Admin – Direct a Submitted Proposal to an Individual for Curation Review
School Editor – see the review comments made by each reviewer of my proposal
Curation Admin – receive a summary of review comments

Extreme Programming (XP)

- XP has been a popular agile methodology
- It takes proven industry best practices and focuses on them intensely and combines them in a new way
- XP has 5 values:
  - **Communication** - with open, frequent verbal discussions
  - **Simplicity** - in designing and implementing solutions
  - **Feedback** - on functionality, requirements, designs and code
  - **Courage** - in facing choices such as throwing away bad code or standing up to a too-tight schedule
  - **Respect**

Some XP Practices

- Planning - users develop a set of stories (called user stories) to describe what the system needs to do
- Testing - tests are written before solutions are implemented
- Pair programming - 2 programmers work together on designing, coding, and testing
- Simple designs - “KISS” and design continuously

User Stories

- User stories serve the same purpose as use cases. They are used instead of a large requirements document
- User stories are written by customers to describe the things the system needs to do for them
- They are used to create time estimates for release planning
- They are in the format of several sentences of text written by the customer in the customer’s terminology: E.g. **As <persona> , I want <what?> so that <why?>**

User Story Example

Murdoch UNIVERSITY

- *As a sales representative, I want to search for my customers by their first and last name so that I have maximum flexibility*

Question: What would be a user story in this format for students buying parking permits?

Pair Programming

WE'RE GOING TO TRY SOMETHING CALLED EXTREME PROGRAM-MING.
FIRST, PICK A PARTNER. THE TWO OF YOU WILL WORK AT ONE COMPUTER FOR FORTY HOURS A WEEK.
THE NEW SYSTEM IS A MINUTE OLD AND I ALREADY HATE EVERYONE.
scottadams@aol.com
www.dilbert.com
© 2002 United Feature Syndicate, Inc.
1/9/03

Adams, S. (2003) [Cartoon] Retrieved from: http://dilbert.com/strip/2003-01-09

Some XP Practices (ctd)

- Refactoring - improving code without changing what it does
- Owning the code collectively - anyone can modify any piece of code

Question: What are the possible negative implications of this practice?

- Continuous integration - small pieces of code are integrated into the system daily or more often
- System metaphor - guides members towards a vision of the system

Some XP Practices (ctd)

- On-site customer - intensive user/customer interaction required
- Small releases - produce small and frequent releases to user/customer
- Forty-hour work week - project should be managed to avoid burnout
- Coding standards - follow coding standards to ensure consistency and ease of refactoring

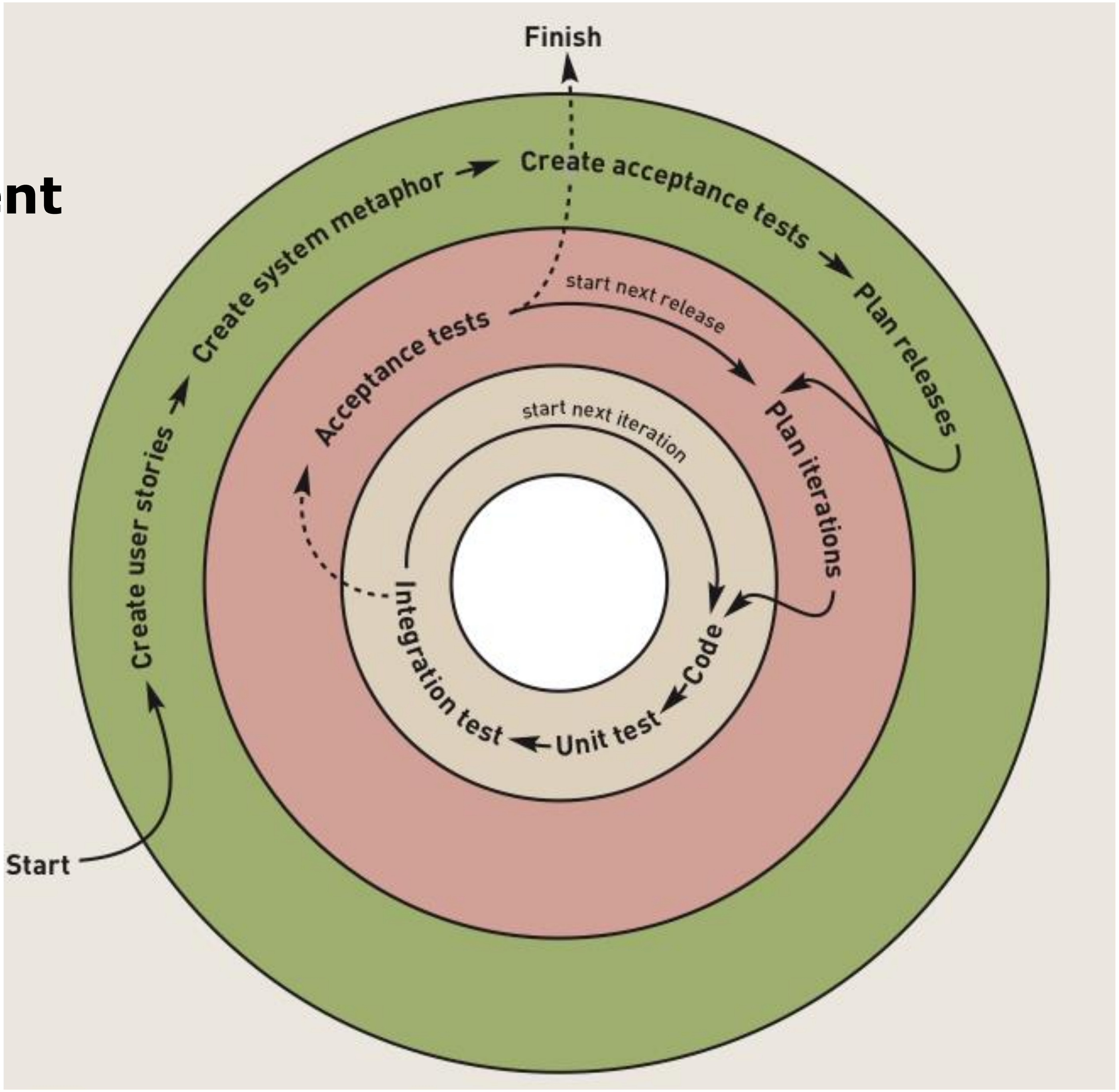
XP Project Activities

- System-level activities:
  - Occur **once** during each development project
  - Involve creating **user stories** and planning releases

- Release-level activities:
  - Cycle multiple times – once for each release
  - Releases are developed and tested in a period of no more than a few weeks or months

- Iteration-level activities:
  - Code and test a specific functional subset in a few days or weeks

XP Development Approach



Possible Limitations of Agile Approaches

Agile approaches provide limited support for:

- Projects with distributed development teams and resources - the emphasis on co-location and face-to-face communication doesn’t fit well with distributed projects
- Outsourcing – as outsourcing of software development tasks is often based on contracts that precisely stipulate what is required
- Projects involving large teams - management processes tailored for small teams. May be communication problems

Possible Limitations of Agile Approaches (ctd)

Limited support for:

- Building or using reusable artifacts - focus on building software to solve specific problems rather than generalised solutions
- Development of large software systems - assumption that code refactoring removes need to design for change may not hold for large complex systems
- Development of safety-critical software systems - quality control processes haven’t been shown to be adequate

Agile Project Success Rates

Murdoch UNIVERSITY

What do these Chaos Report figures suggest about the value of agile development approaches?

Size	Approach	Successful	Challenged	Failed
All	Agile	39%	52%	9%
	Waterfall	11%	60%	20%
Large	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

Source: https://www.infoq.com/articles/standish-chaos-2015

Project Management and Agile Approaches

Project management of adaptive approaches differs from project management of traditional approaches. Consider the differences in some of the main areas of project management:

- **Project time management**
  - Smaller scope and focused on each iteration
  - More realistic work schedules
- **Project scope management**
  - Users and clients are more responsible for scope
  - Scope control consists of controlling the number of iterations
- **Project cost management**
  - More difficult to predict because of unknowns

Project Management (ctd)

- **Project communication management**
  - Critical because of open verbal communication and collaborative work
- **Project quality management**
  - Continual testing and refactoring must be scheduled
- **Project risk management**
  - High-risk aspects usually addressed in early iterations
- **Project human resource management**
  - Teams organise themselves

Question

- Many organisations are attempting to use both traditional approaches and agile approaches

- **Why do you think this is so? What benefits do you think organisations can obtain from allowing different development approaches to co-exist? What problems do you think can arise?**

Learning Objectives Revisited

- What are the characteristics of agile development?
- How do agile development approaches differ from traditional system development methodologies?
- What are the potential problems with agile development?
- Can you describe several different agile development approaches?

Additional References

- CollabNet VersionOne (2019) *The 13th Annual State of Agile Report*. Available from https://www.stateofagile.com/#ufh-i-521251909-13th-annual-state-of-agile-report/473508
- Turk, D., France, R., & Rumpe, B. (2002). Limitations of agile software process. In *Proceedings of the Third International Conference on eXtreme Programming and Agile Processes in Software Engineering* (pp. 43-46) Sardina, Italy. http://www4.in.tum.de/publ/papers/XP02.Limitations.pdf
- Vinekar, V., Slinkman, C. W., & Nerur, S. (2006). Can agile and traditional systems development approaches coexist? An ambidextrous view. *Information Systems Management*, 23(3), 31-42.
- Williams, L. (2012) What agile teams think of agile principles. *Communications of the ACM* 55(4), 71-76