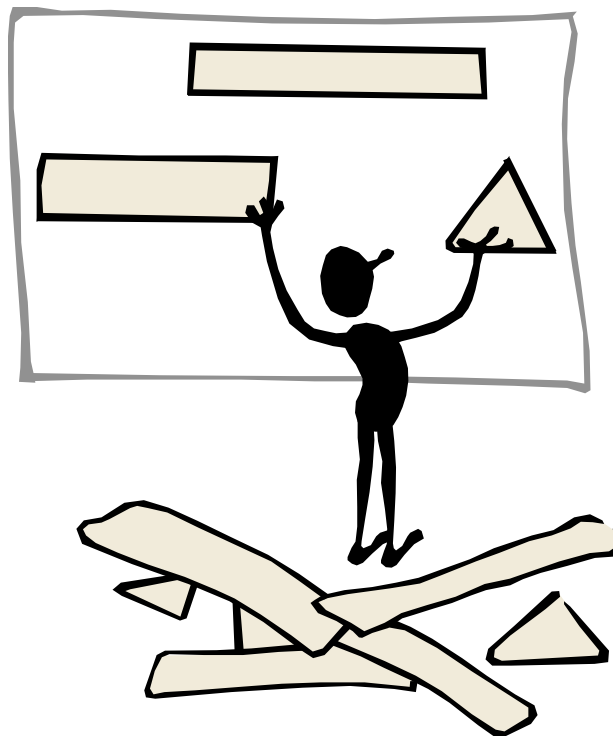


COMP33711

Agile Software Engineering
(Agile Methods Theme)

Introduction: the Agile Lego Game

2017/2018 Academic Session



School of Computer Science
University of Manchester
Oxford Road
Manchester M13 9PL
U.K.

Session 1. The Agile Lego Game

This first session of the course unit introduces some of the basic concepts underlying agile software engineering, to provide the context for the rest of the course. Rather than being talked at for two hours about agile approaches, you will instead gain experience of what agile methods are like in practice, by playing the Agile Lego Game, devised by Sam Newman, Dan North and Mike Hill, of Thoughtworks.

Games are an important concept in agile software engineering, both as a means of conveying the key concepts of agile and also as a means of actually doing agile development. In this game, you will work as part of an agile team to develop a new product to meet a customer's needs, using the medium of Lego bricks.

The University of Manchester	<div>MANCHESTER 1824</div> <div>School of Computer Science</div>	Notes
	<h3 data-bbox="331 844 807 880">Agile Methods Theme: Overview</h3> <ul data-bbox="331 896 847 1279" style="list-style-type: none"><li data-bbox="331 896 847 931">• Builds on agile ideas introduced in 2nd year<li data-bbox="331 967 847 1104">• This semester:<ul data-bbox="368 1003 847 1104" style="list-style-type: none"><li data-bbox="368 1003 847 1032">– COMP33711: Agile Software Engineering<li data-bbox="368 1037 847 1066">– Agility during software development<li data-bbox="368 1070 847 1104">– Suzanne Embury & Christos Kotselidis<li data-bbox="331 1142 847 1279">• Next semester:<ul data-bbox="368 1178 847 1279" style="list-style-type: none"><li data-bbox="368 1178 847 1207">– COMP33812: Software Evolution<li data-bbox="368 1211 847 1240">– Agility after deployment<li data-bbox="368 1245 847 1279">– Andy Carpenter & Sarah Clinch	
The University of Manchester	<div data-bbox="292 1391 424 1429">MANCHESTER 1824</div> <div data-bbox="831 1391 951 1424">School of Computer Science</div> <h3 data-bbox="331 1464 938 1500">COMP33711: Agile Software Engineering</h3> <ul data-bbox="331 1518 810 1863" style="list-style-type: none"><li data-bbox="331 1518 810 1655">• Before reading week:<ul data-bbox="368 1554 810 1655" style="list-style-type: none"><li data-bbox="368 1554 810 1583">– Introduction to Agile<li data-bbox="368 1588 810 1617">– Agile Requirements Gathering<li data-bbox="368 1621 810 1655">– Agile Planning<li data-bbox="331 1675 810 1812">• After reading week:<ul data-bbox="368 1711 810 1812" style="list-style-type: none"><li data-bbox="368 1711 810 1740">– Evolutionary Design<li data-bbox="368 1744 810 1774">– Test-Driven Development<li data-bbox="368 1778 810 1812">– Acceptance Test-Driven Development<li data-bbox="331 1832 810 1863">• January Exam	

See Blackboard for lots more information about the course:

- Electronic copies of handouts
- Pointers to additional reading to support lectures
- Self-tests for some of the topics covered
- Information about the exam, plus material to support revision.

Plus, you can find out about the content of each session, and its location. All the lectures before reading week will be held in IT407, but after reading week we'll divide our time between IT407 and the 3rd year project lab. We'll remind you in lectures and also on Blackboard whenever we're not in IT407.


MANCHESTER
1824

The University
of Manchester

School of
Computer Science

Today's Lecture: The Agile Lego Game

- Experience agile approach in less than 2 hours
- Just a flavour – not a complete presentation
- Based on the XP Lego Game by Sam Newman, Dan North and Mike Hill (© ThoughtWorks)
- Yes, we're going to be using real Lego!
- But first, what happened at the start of this lecture?



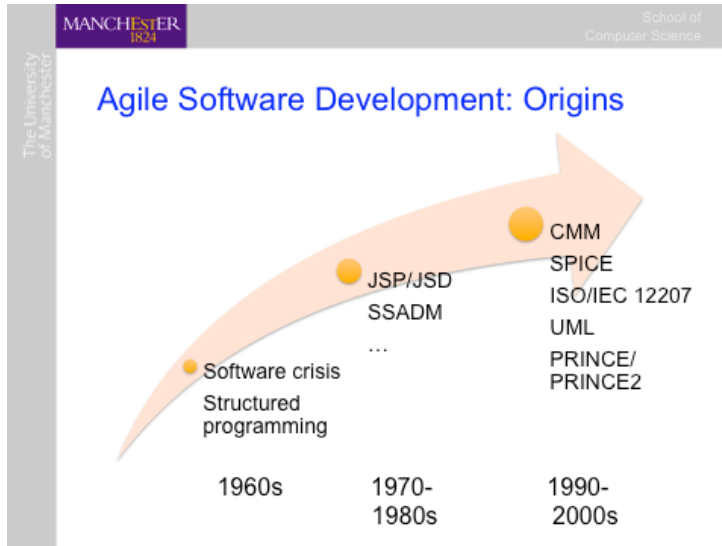
MANCHESTER
1824

The University
of Manchester

School of
Computer Science

Connection Activities

- A quick (fun?) activity at the start of the session, to help you connect with the topic to be covered.
- Expect a connection activity at every lecture.
- Don't wait to be told, just get started on the activity as soon as you come in.
- Try to take an agile approach:
 - Aim for "good enough" quickly, not perfection
- Many thanks to Abeer Shahid, Iliada Eleftheriou, Caroline Jay and Raluca Puichilita



Notes

MANCHESTER 1824 School of Computer Science

Big Software Process: Did it work?

- UK study of 1,027 projects: 13% succeeded.
 - “waterfall-style scope management” named as number 1 problem in 82% of projects
- 1995 US DoD study of \$37 billion+ projects:
 - 46% failed to meet user needs, never successfully used
 - 20% required extensive rework to be usable
- Study of 400+ waterfall projects:
 - only 10% of developed code deployed
 - only 2% of developed code actually used
- 2015 Standish Chaos Report
 - 52% challenged, 19% failed

These figures come from [Larman 2003] and the Standish Chaos report. Chapter 6 of Larman’s book contains more figures and evidence of our difficulties in producing software on time, to budget and to meet customers’ needs.

The 2011 Chaos report figures are quoted from a short article by John Curtis at:

<http://quotient.net/blog/2012/6/25/the-importance-of-a-great-project-manager/>

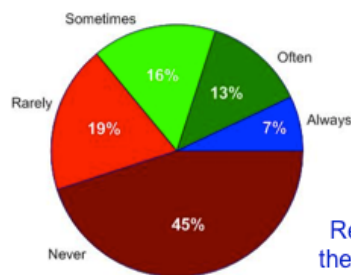
and are unverified. (It should also be noted that there are a number of questionable aspects regarding the methodology used by the Standish Group in their surveys. Most notable of these is the fact that the raw data from which these data are generated is not made publicly available at any time. This means we have no way to independently verify the details of the analysis which produced these conclusions.)

Chaos?

- "The project was two years late and three years in development. We had thirty people on the project. We delivered an application the user didn't need. They had stopped selling the product over a year before." [Standish Chaos Report 2005, focus group comment from project manager at insurance company]
- "We found that both satisfaction and value are greater when the features and functions delivered are much less than originally specified and only meet obvious needs." [Standish Chaos Report, 2015]

What's Going Wrong?

Average percentage of delivered functionality actually used when a serial approach to requirements elicitation and documentation is taken on a "successful" information technology project.



Results from
the connection
activity?

Source: Chaos Report v3, Standish Group.
Copyright 2005-2006 Scott W. Ambler

The Agile Manifesto (2001)

"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."

Taken from: agilemanifesto.org

Aim of Software Development

- Deliver value
- With less resource than the value to be delivered
- In time for the value to be realised

Agile Solution

- Trust
 - Self-organising team
 - Empower developers to take key decisions
 - Customer is a member of the team
- Simplicity (of both process and product)
 - “Do the simplest thing that can possibly work” – Ward Cunningham
 - YAGNI
- Feedback
 - Fail fast
 - 360° feedback



Naming of Parts

In this course unit, and in the industry, you will hear people talking about agile processes, agile practices, agile values and agile principle. Here is a quick guide to what each of these terms mean.

- There are 4 agile values, and they come directly from the agile manifesto. When people talk about an agile value, they mean one of the four statements from the manifesto, saying what elements of a software development method is preferred, in agile terms.
- There are also 12 agile principles. These are short statements describing the underlying principles that should be followed by anyone doing agile development. They are very general and high level. We will look at them again later in the course unit.
- Based on the agile values and principles, industry practitioners have developed a number of agile practices. These are individual techniques that can be applied to a specific aspect of software development, to help carry it out without waste and with a focus on value. Examples are: user stories, task boards, daily stand-up meetings and test-driven development.
- Last, there are a number of agile processes (sometimes called agile methodologies). Well known examples of such processes are Scrum and XP. A process selects a number of agile practices, and describes how and when they should be used on a development project.

The University of Manchester
MANCHESTER 1824

School of Computer Science

The Agile Lego Game

- Get into teams of 4-6
- You are working for AnimalCo
- Project Aim: to develop a new kind of animal
- Use an agile approach

Notes

The University of Manchester
MANCHESTER 1824

School of Computer Science

Waterfall Approach

```

graph LR
    A[Gather requirements] --> B[Plan]
    B --> C[Design]
    C --> D[Implement and Unit Test]
    D --> E[Integrate and System Test]
    E --> F[Deploy]
  
```

The University of Manchester
MANCHESTER 1824

School of Computer Science

Simple Agile Process

- Iterative & incremental
- Short iterations

```

graph TD
    A[Gather some requirements] --> B[Estimate]
    B --> C[Plan]
    C --> D[Develop]
    D --> E[Deploy?]
    E --> F[Reflect]
    F --> A
  
```


The University of Manchester
MANCHESTER 1824

School of Computer Science

Agile Practice: User Stories

- What does a typical requirements spec contain?
- Agile takes a different approach:
 - Requirements gathered as "user stories"
 - Example:

Story: As a student, I want to know which course units I'm not reaching my target average mark in.			
Business value:	200		
Effort:	Small	Medium	Large

Notes

The University of Manchester
MANCHESTER 1824

School of Computer Science

Estimation – Iteration 1

- You will be given a set of stories.
- Before starting to build, you need to decide what you can commit to:
 - Business value is given (not realistic!)
 - You need to estimate effort
- 3 minutes** to estimate effort for all your stories
 - 3 point scale: small, medium, large
 - N.B. **relative** estimation, not absolute!
 - Aim for consistency across the stories

The University of Manchester
MANCHESTER 1824

School of Computer Science

Planning – Iteration 1


- Next, select the stories from your backlog that you will implement in this iteration
- Must balance
 - Business value achieved
 - Effort expended (limited resources)
 - Dependencies between stories
 - Learning gained from implementation effort
- How will we document the plan?
 - Big Visible Chart practice
 - Task Board practice



Notes

Taken (without permission) from <http://www.richard-banks.org/2008/06/another-reason-for-story-boards.html>

The University of Manchester




MANCHESTER 1824

School of Computer Science

Planning – Iteration 1

- **3 minutes** to select stories from your backlog that you will implement in this iteration
 - You'll get 5 minutes for implementation
- Use your task board to record your selection
 - Put selected stories in “doing” column
 - All others in “to do” column

The University of Manchester




MANCHESTER 1824

School of Computer Science

Development – Iteration 1

- 5 minutes to implement as many of your selected stories as you can

The University of Manchester



MANCHESTER 1824

School of Computer Science

Showcase – Iteration 1

- Show your customer what you achieved!
- The customer will “sign off” those stories that have been completed satisfactorily
- Remember:
 - It is unlikely you will have got it all perfect 1st time
 - Lack of sign-off = good (you learn something)

Retrospectives

- Delivering in small chunks means more opportunities to get **feedback**
 - From the customer
 - Showcase (and many other opportunities)
 - From the development team
 - Iteration Retrospective (and many other opportunities)
- Delivering in small chunks means more opportunities to **change** in response to feedback

Retrospective – Iteration 1

- 5 minutes to answer these 3 questions about your team's process:
 - What went well?
 - What could have been done better?
 - What was confusing?
- Each team will present one idea for how they will work differently as a team in the next iteration.



A Note on Retrospectives:

A lot of students struggle with the idea of retrospectives, so don't worry if your team finds it difficult at first. The important point to remember is that the aim is to examine and improve *your team's process*, and not the specific work you have done in the last iteration. You are not being asked: "which stories did you implement well and which could have been better implemented?" That is decided at the showcase. Instead, you should consider what working practices your team put in place that worked well, and what practices could have been better. From this, you can think up some small but hopefully significant changes to try, that might improve your team's performance in the next iteration.

An example (brief) retrospective is given below:

- Julia: We worked really well in small groups on the individual components, but then fitting them together didn't go so well, because we had all assumed different sizes for the animal.
- Sam: I was confused about how many legs we were going for.

- Jameel: Next time, we could try to agree rough sizes for the connecting parts before we begin to build.
- Sam: Yes, and the number of legs!
- Julia: That's a good idea. But how can we stop ourselves from getting bogged down in a long discussion about all the details, when we should be getting on and building?
- Sian: We could give ourselves one minute to agree the big picture, and then get building. I could set the alarm on my phone.
- Jameel: Good idea. Let's try that.

Notice that the focus in this example is on how the team performed as a whole, and not on criticism of any individual team member or their work. The team is asked to think about how they could do their work even better next time, rather than thinking about blame and whose fault the problem is. This is to encourage team members to be honest in examining their work, without worrying that they will be criticised for any mistakes. We want to encourage the team to admit their mistakes as soon as they happen, so that we can try to fix them straight away, rather than having them swept under the carpet until the costs of a mistake are so high it can no longer be ignored.

MANCHESTER
1824

School of
Computer Science

The University
of Manchester

Feedback from Teams

- Each team should tell us about 1 change it plans to make in the next iteration

MANCHESTER
1824

School of
Computer Science

The University
of Manchester

Feedback from Your Customers

The University of Manchester

MANCHESTER
1824

School of
Computer Science

Iteration 2

- Now we're going to leave you to run through the process again
- **10 minutes** to:
 - Estimate the new stories
 - Plan what you will deliver for the next iteration
 - Put these cards in the “doing” column
 - All other unfinished cards in the “to do” column
 - Develop the stories in your “doing” column

Notes

The University of Manchester

MANCHESTER
1824

School of
Computer Science

Showcase – iteration 2

- Show us what you achieved!
- Get your customer to sign off completed stories.

The University of Manchester

MANCHESTER
1824

School of
Computer Science

Debrief

- Trust
- Simplicity
- Feedback

The University of Manchester
MANCHESTER 1824
School of Computer Science

Conclusions

- From this exercise, agile software engineering is...?
- Many agile ideas/practices have been around for a long time (e.g. iterative/incremental development)
- Significance is the particular combinations of complementary techniques
 - AND the change in mindset/philosophy

Notes

The University of Manchester
MANCHESTER 1824
School of Computer Science

Reflection

- Collect a reflection exercise on your way out
- See the self-test on Moodle for this week, for more reflection
- Coming up next week:
 - The Trouble With Big-Upfront Requirements Gathering (and Planning and Design...)

The University of Manchester
MANCHESTER 1824
School of Computer Science

End-of-Lecture Retrospective

- Finally, please comment on how this lecture went, using the post-it notes provided
- Write your comment on the post-it notes
- Stick on the wall beside the question your comment addresses

References

- [Larman 2003] Agile and Iterative Development: a Manager's Guide, by Craig Larman, Addison-Wesley Publishers, August 2003.
ISBN: 978-0131111554