CSC3045 & CSC3052
AKA: “The Agile Modules”

Project Information & Team Organisation

School of Electronics, Electrical Engineering & Computer Science

Queen's University, Belfast

Dr Darryl Stewart

Outline

- Why do a team project?
- Very brief description of a *typical* Agile Project
- The 'Story' behind your project
- The phases of the project
  - Getting up to speed
  - Estimation and planning
  - Sprinting
- Deliverables
- Assessments
- Important Stuff
- Questions?

How much is it worth?

Why do a team project?

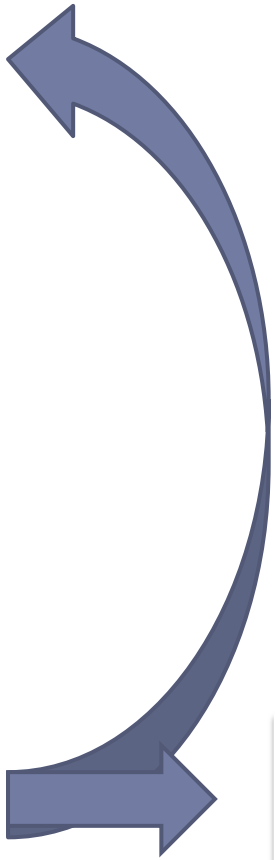
- Agile development is about ***how to do successful team projects***
  - Learnt by experiencing a project
  - Assessed by seeing you do a project
- Could be done individually
  - You could apply agile methods to your individual software engineering projects
- Various agile practices and tools are specifically designed to:
  1. Take advantage of teamwork
  2. To make teamwork easier/better
- You can get to experience these **only** in the context of a **team** project

A ***typical*** Agile Project

- *Customer* provides current set of requirements with associated business value **for this project**
- Face-to-face discussions to clarify and refine requirements
- Acceptance tests are specified

- Team plans what can/will be done in next iteration (**the iteration requirements**)

- Team does a development iteration
- *Team and Customer review what has been achieved*
- *Team reviews how they have performed*
- More requirements can be added **<u>to the project</u>** or changed (Business value updated) during the sprint
- Customer decides if this version of the system should be released

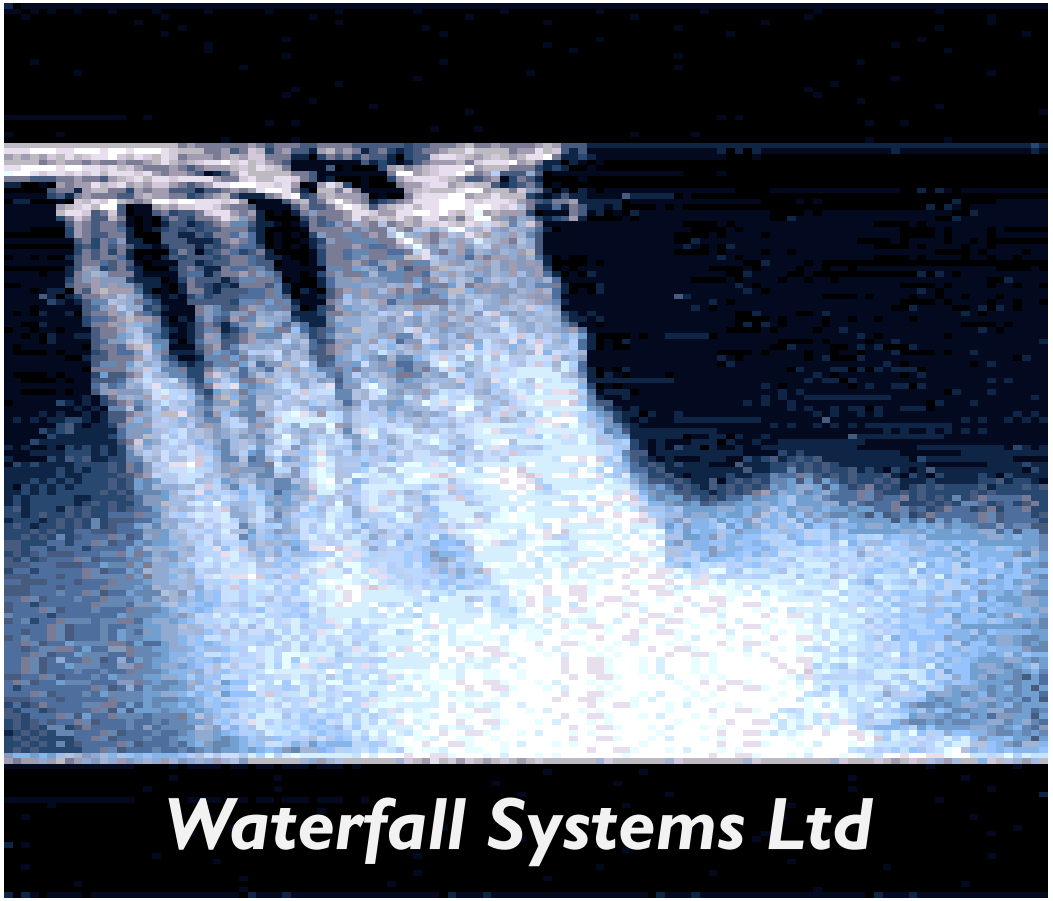


Repeat until project complete

Finished Project

The *'Story'* behind your project

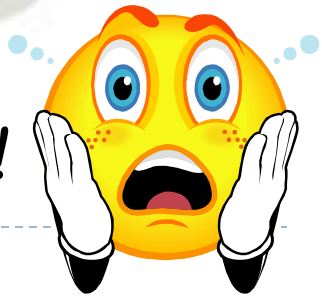
- Imagine you work for ***Waterfall Systems Ltd***



The ***'Story'*** behind your project

- *They are having problems*

They ask you to sort it out!



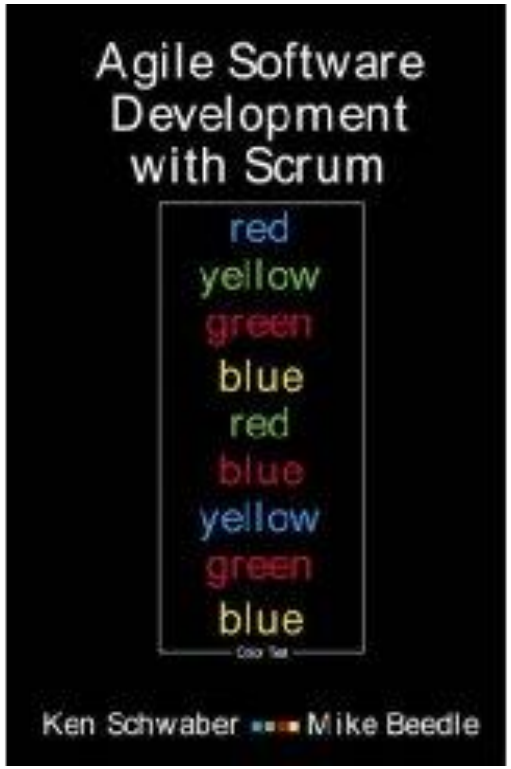
The ***'Story'*** behind your project

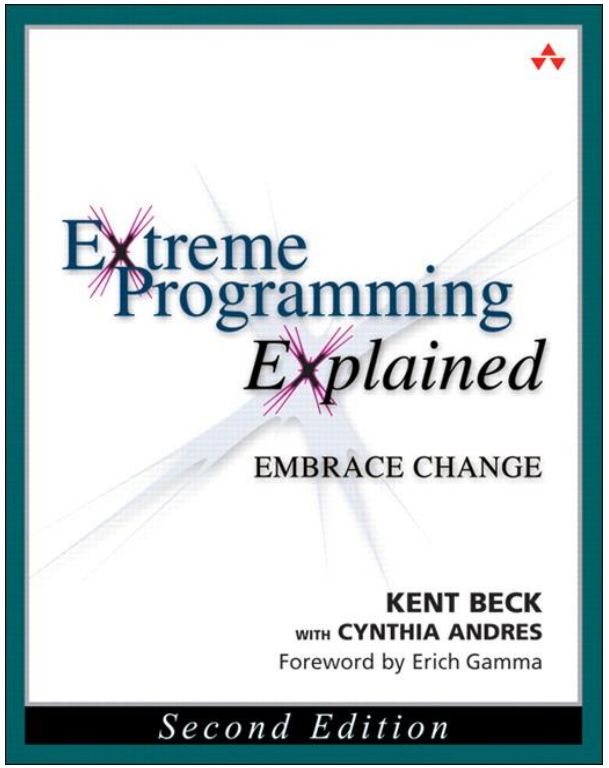
You suggest "Agile Development is the solution!"

They say "Lets give it a try!"

The *'Story'* behind your project

- *You suggest using Scrum and elements of Extreme Programming in the next project*



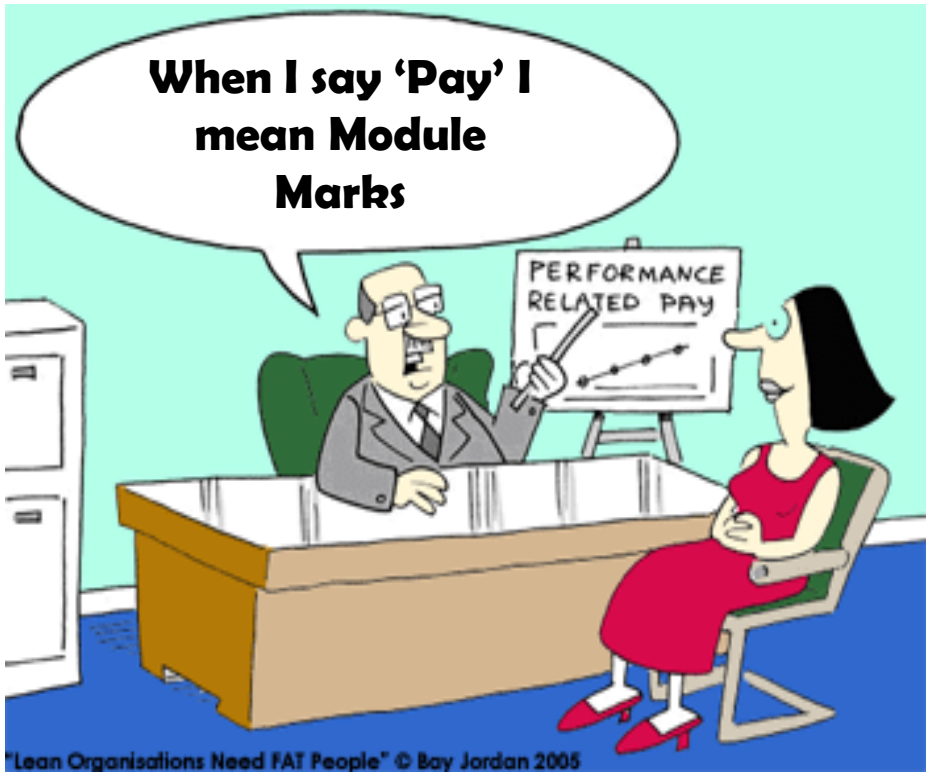


The ***'Story'*** behind your project

- *They say they will monitor your teams performance to see if your suggestion is good*

The *'Story'* behind your project

- *And your pay will be performance related…*



The next Customer Arrives…

The *'Story'* behind your project

- The customer will provide the specific requirements for the system over the next 2 weeks
- You must apply the Scrum process learned in the lectures to manage the project
- You should apply the XP practices learned to develop the system
- The clients for the projects are:

System A

- This project will be implemented by **teams on CSC3052**
- I am your client (Darryl Stewart)

System B

- This project will be implemented by **CS/SE/SESE** teams on **CSC3045**
- I am your client (Darryl Stewart)

System C

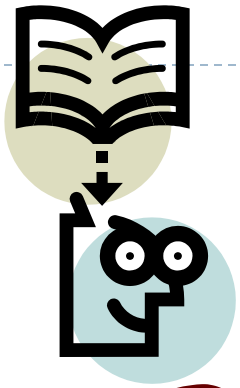
- This project will be implemented by the **Games** teams on **CSC3045**
- Your client is Phil Hanna

Phases in your project

- Requirements elicitation & Backlog creation
- Sprint 1estimation and planning
- Sprint 1
- Sprint 1 Review
- Sprint 2 estimation and planning
- Sprint 2
- Sprint 2 Review

Getting up to speed – Weeks 1 to 4

- Learn about Scrum and XP practices in lectures
- Independently learn the language/technologies for your project
- Investigate how to implement the known system components and test everything possible in the labs (**keep evidence of this work in order to possibly demonstrate later**)
- Practice the Agile process/practices and get the team organised
- Learn how to use the tools (VStudio, NUnit, SVN etc.)
- Attend things (team mates will monitor this)
- Learn what you can realistically get done in a given amount of time – **very important** for accurate requirement and task estimation

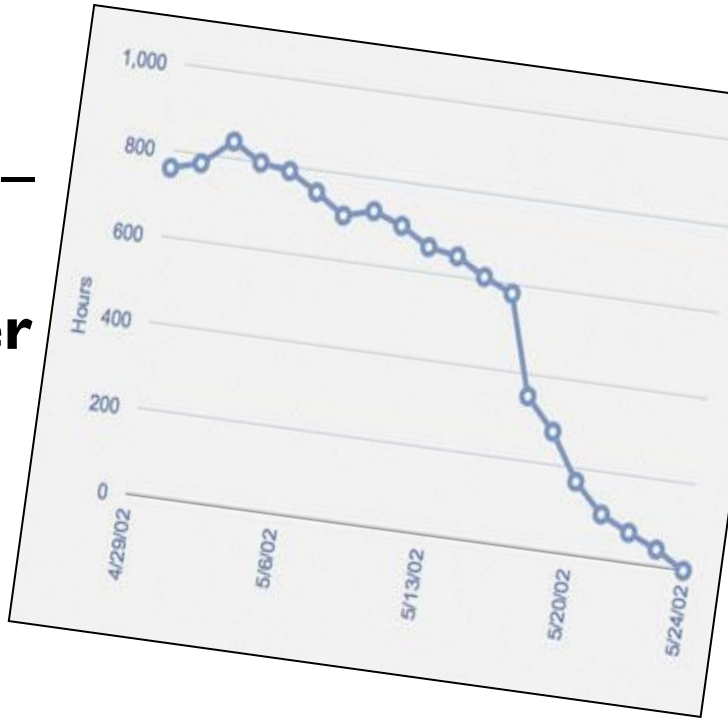


Initial estimation and planning

- Learn as much as possible about your project (requirements and priorities) during the lab sessions
- You are running your project so you should apply the correct agile process
  - Produce a Product Backlog
  - Do planning poker
  - Goal setting
  - Sprint planning
  - Sprint Backlog creation
- You should produce the correct artifacts of the process at the correct times and use them appropriately
- They should be held in the team repository and could be inspected at any stage

During a Sprint

- Work consistently according to your Sprint plans to implement the features you have committed to
- Work in the labs together when possible – **the most successful teams on the modules so far have worked together consistently in the labs**
- Follow the Scrum guidelines
- Apply the XP practices
- Produce clean code
- Send daily emails to me and your team **(More on this later…)**



After a Sprint

- Prepare videos/individual reports/peer assessments
- Have a Sprint Review with me and potentialy others
- Do a Retrospective individually and collectively
- Prepare group reports
- Plan next Sprint

Deliverables – for each Sprint

Team Deliverables

1. **Current Working System & Code Repository**
   - The current working source code and build of the system in your teams' SVN code repository.
   - The repository should be well organized, tidy and with good clear comments.
   - The source and working build should be clearly labeled and easy to find.
   - Each time a user story is completed (*done done*) then the SVN version should be clearly labeled for subsequent review.

Deliverables – for each Sprint

Team Deliverables

2. Audio Recordings & Videos & Minutes

A. Daily scrum meetings should be recorded (audio only) and added daily to the SVN repository (use a mobile or similar to record these). **Otherwise** type up minutes of the meeting. Uploaded daily.

B. Video screen capture or video presentation detailing/ analyzing how the sprint was managed and the sprint's level of success **(Max 5 minutes) (One video)**

- Show and discuss any artifacts/ processes /practices followed
- Significant problems faced/issues encountered/blockers etc.
- Should demonstrate knowledge of correct Agile processes

Deliverables – for each Sprint

Team Deliverables

2. Videos (contd.)

C. Video screen capture of the current working software with someone providing audio description
(**Max 8 minutes for Sprint 1 and Max 15 minutes for Sprint 2**)

- This should demonstrate the '*done done*' user stories in the most up to date working system at the end of the sprint
- Highlight significant **known** bugs/issues in system

Deliverables – for each Sprint

Team Deliverables

2. Videos (contd.)

D. Video screen capture of code walkthrough with audio description (**Max 5 minutes)**

This should demonstrate :

- ☐ Code structure
- ☐ Coding standards (Code that follows them and any Code that doesn't!)
- ☐ Test code (Show code, test runner output and mention estimated coverage)
- ☐ A description of any code/reference material which has been used directly or as inspiration during the development.
- ☐ A few slides at the start to highlight and justify the significant choices made in the code and during development (highlight anything you feel is particularly good)

Deliverables – for each Sprint

Team Deliverables

2. Videos (contd.)

For videos B, C and D

- They must be good enough quality to see text, code etc.
- Use screen capture software, e.g. Jing
- Place them in the repository in a folder called **Videos** located in the root folder for each sprint
- Zip them up together and email to me via DropBox
- **Have them ready and delivered by 9pm on the Tuesday after each sprint**

Deliverables – Week 12 delivery

Team Deliverable

Team Report (Max 30 pages - don't pad it out!)

- Artifacts from the scrum process for both sprints
  - Sprint planning activities
  - Sprint management activities
  - Sprint review and retrospective process and results
- System design description (UML)
- Code quality analysis
- Testing documentation
- Attribution of work done to individuals in team
- A disclaimer concisely detailing any code/reference material which has been used directly or as inspiration during the development.
- Known bugs list
- Critical reflection on the teams work (the output of the retrospective)
- Other significant information or things of note
- PDF document submitted electronically - details regarding submission outlined later

Deliverables – Week 12 delivery

Individual Deliverable

Individual report (Max 10 pages)

- Critical reflection on team deliverables
  - Quality of the process followed
  - Quantity of working code delivered
  - Quality of the code delivered
- Summary of your contributions to sprints
  - Management process
  - Code delivered
    - ☐ Identify your specific contributions
  - Critical reflection on your own performance (don't leave blind spots)
- Brief reflection on contributions made by members of your team
  - Min 1 page - max 3 pages (of the 10)
- PDF document submitted electronically - details regarding submission outlined later

Deliverables – Week 12 delivery

Team Deliverables

3. Peer Assessment

- Specific instructions to follow
- This will just be one form of evidence which I will examine
- It will not be used directly to allocate the final marks

Deliverables – **every weekday** during sprints

Individual email update

- Daily email sent when all work has been completed for the day (before midnight)
- **Send to me and all your team mates**
- Use the following **specific Subject Line** as below:
  - "CSC30*xx* – Team C*xx* – *Agile Daily Update*"

    E.g. "CSC3052 – Team CIT3 – Agile Daily Update"

Deliverables – **every weekday** during sprints

Individual email update – Example content

- **Time 1**: *10am-11am*
  - *Attended scrum meeting.*
  - *Added class X to do…*
- **Time 2:** *2pm-4pm*
  - *Pair programmed with Jane to fix…*
  - *User Story C is done done.*
  - *Found issue with class C in method M…*
  - *Updated the spreadsheet…*

- **Be brief and factual and honest**
- **Read all emails sent to you and let me know at the end of each week if there is any significant false accounting going on!**

Assessments

1. Assess how much **VALUE** was added to the system from the customers perspective

2. Assess the **QUALITY** of the software produced from a Software Engineering perspective

3. Assess the quality of the **PROCESS** followed including the effectiveness of Project Management, Teamwork & Communication displayed

Measuring **Value Added to System**

- How much of the product backlog did the team and each individual burn down? Are the requirements (user stories) fully complete (done done)?
- Are the Acceptance Tests passed?
- Is it ***functional, reliable, useable*** *and* ***efficient***?
- Focus on **Quality**
- This will determine the proportion of marks awarded for "**Value Added to System**"
- Conceptual marking scheme used

Measuring **Quality of Software**

- *Suitability for purpose* and *maintainability* are major aspects I will be assessing
  - Coding style and standards
  - System Design
    - Component Cohesion/Coupling
    - Refactoring
    - Clarity
  - Testing
    - Testability and coverage
- **I may use code analysis tools to gain some measures of this along with code inspections**

Measuring **Quality of Process**

Application of Scrum

- Backlog preparation/management
- Scrum Spreadsheets
  - Detail, correct usage, proactive management
- Communication with Product Owner
  - Timely, efficient, complete, professional,…
- Daily scrum meetings
  - 2 selected at random and assessed
- Story by story progress?
- Sprint retrospective notes show evidence of reflection and plans for improvement?
  - Thoughtful, complete, knowledgeable
  - Clearly identifies Highlights and *Lowlights* of all aspects of the teams performance

Feedback & Marks

- I will give each team feedback during the sprints and specifically during the Sprint Reviews
- The marking process takes a considerable amount of time so final marks for the project will not be known until January

Important stuff

- Is code re-use allowed?
  - **Yes**, however…
    - It must be free and publicly available code
    - Code developed by one team **should not be shared** with another team
    - You must clearly indicate the source or inspiration for any such code in your teams' **DISCLAIMER** information even if it has been changed a lot compared to the original
    - It must be 'refactored' to fit in properly with your project design, coding standards etc. and all unit tests must be written for it – usually not available from source
    - If it turns out that lots more code is available than originally thought (by me or by the team) and much less work is required than estimated then the onus is on the team to seek more requirements to ensure they are giving 'good value for money'

Important stuff

- Plagiarism
  - The project may be submitted to an on-line plagiarism test.
  - Incredibly important that you identify which bits, if any, draw upon other sources
  - Using code is not a problem if you've referenced the source
  - Please also be aware that other teams may use the same source for some code and if they reference it correctly and you do not then this will set off alarm bells
- Vivas
  - If there is suspected plagiarism then vivas will be held – if found guilty then it is a major academic offence
- Tracking
  - The code repository will show a trace of every development step

Any Questions?

Take home messages

- Start forming teams of **compatible** people now
- Minimum expected lab work each week
- Project is assessed on:
  - Value of Functionality
  - Quality of Software Engineering
  - Quality of Process
- Lots of specific deliverables needed during and after the sprints and in Week 12