

COM3005J: Agile Processes

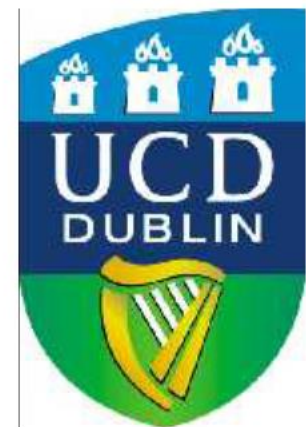
Agile Practices

Dr Soumyabrata DEV

<https://soumyabrata.dev/>

School of Computer Science and Informatics
University College Dublin

Beijing-Dublin International College



Agile Practices

1: Meetings

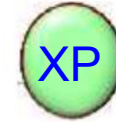
2: Development

3: Release

4: Testing and quality

5: Management and others

Daily Meeting



Held every morning

Goal: set the day's work, in the broader context of the project

Time-limited, usually 15 minutes (“stand-up meeting”)

Involves all team members, with special role for “committed” (over just “involved”)

Focus:

- Defining commitments
- Uncovering impediments

Resolution will take place outside of meeting

Daily Meeting: The Three Questions



The daily meeting requires every team member to answer three questions:

- What did you do yesterday?
- What will you do today?
- Are there any impediments in your way?

Planning Meeting

Scrum



At beginning of every sprint

Goal: define work for sprint

Outcome: Sprint Backlog, with time estimate for every task

8-hour time limit

- 1st half, product owner + team: prioritize product backlog
- 2nd half, team only: plan for Sprint, producing sprint backlog

Retrospective



All team members reflect on past sprint

Make continuous process improvements

Two main questions:

- What went well?
- What could be improved?

3-hour time limit

Normally does not involve customers.

Retrospective : Prep



Book a meeting room for at least an hour: 15 minutes to set up, 30 minutes for the session, and 15 minutes to take photos and clean up.

Come 15 minutes ahead of the rest of the team, and bring all supplies. Draw the headings "**What did we do well?**" and "**What should have we done better?**" up on the whiteboard.

Retrospective : Step 1



Set the stage (5 min)

Welcome everyone to the retrospective meeting and establish the rules of engagement:

- Embrace a positive spirit of continuous improvement and share whatever you think will help the team improve.
- Don't make it personal, don't take it personally.
- Listen with an open mind, and remember that everyone's experience is valid (even those you don't share).
- Set the boundary of your discussion – is it that last sprint? the last quarter? since the project started? Be clear how far back you're going to go.
- Encourage the team to embrace an improvement mindset, away from blame.

Retrospective : Step 2



What went well? (10 min)

Start the session on a positive note. Have each team member use **green sticky notes** to write down what they feel went well (one idea per sticky). As people post their stickies on the whiteboard, the facilitator should group similar or duplicate ideas together.

Discuss your ideas briefly as a team.

Retrospective : Step 3



What needs improvement? (10 min)

Same structure as above, but using **pink or red stickies**.

Remind your team that this is about actions and outcomes
– not about specific people.

Retrospective : Step 4



Next steps (5 min)

Having identified what didn't go so well, what concrete actions can the team take to improve those things? Have your team use **blue sticky notes to place ideas** on the board. Group them and then discuss as a team, agree to which actions you will take, assign owners and a due date to get them DONE.

Thank everyone for their involvement and their honesty. Quickly run through the list of follow-up items, their owners and due dates.

Review Meeting



Review work:

- Completed
- Not completed

Present and demo completed work to stakeholders

Incomplete work cannot be demonstrated

4-hour time limit

Includes the customer in the discussion

Reflective Improvement



Developers must take breaks from regular development to look for ways to improve the process

Iterations help with this by providing feedback on whether or not the current process is working

Practices

Part D: Practices 1: Meetings

What we have seen:

Agile development is characterized by a set of well-defined meetings

Most important is the Daily Meeting

In Scrum: planning and review meetings

Require adaptations for distributed teams

Agile Practices

1: Meetings

2: Development

3: Release

4: Testing and quality

5: Management and others

Pair Programming



Source: Beck 2005

Two programmers sitting at one machine

Thinking out loud

Goals:

- Make thinking process explicit
- Keep each other on task
- Brainstorm refinements to system
- Clarify ideas
- Take initiative when other stuck, lowering frustration
- Hold each other accountable to team practices

Single Code Base



Maintain a single code base: avoid branching, even if permitted by configuration management system

Shared Code



Agile methods reject code ownership in favor of code whose responsibility is shared by entire team

Rationale:

- Most non-trivial features extend across many layers in the application
- Code ownership creates unnecessary dependencies between team members and delays
- What counts is implemented features, not personal responsibility
- Avoid blame game
- Avoid specialization
- Minimize risk (team members leaving)

Leave Optimization till Last

Source: Wallace 02



Wait until you have finished a story and run your tests before you try to optimize your work

Only then can you analyze what exactly it is that needs optimizing

Do not make work for yourself by trying to anticipate problems before they exist

Simple Design



Produce the simplest design that works

Refactor as needed

Incremental Design

Source: Shore 08



Developers work in small steps, validating each before moving to the next
Three parts:

- Start by creating the simplest design that could possibly work
- Incrementally add to it as the needs of the software evolve
- Continuously improve design by reflecting on its strengths and weaknesses

“When you first create a design element, be completely specific. Create a simple design that solves only the problem you face, no matter how easy it may seem to solve more general problems.

“The ability to think in abstractions is often a sign of a good programmer. Coding for one specific scenario will seem strange, even unprofessional. Waiting to create abstractions will enable you to create designs that are simple and powerful. Do it anyway.”

System Metaphor



Source: Tomayko 03

“A metaphor is meant to be agreed upon by all members of a project as a means of simply explaining the purpose of the project and thus guide the structure of the architecture”

Benefits:

- Communication, including between customers & developers
- Clarify project, explain functionality
- Favors simple design
- Helps find common vocabulary

Example Metaphors



Source: Tomayko 03



Project	Metaphor	Explanation
Wrist camera	Portrait studio	The software has the capability for transferring images from one device (e.g., PDA, PC) to another, and some image processing capabilities. It is much like a portrait studio, where a camera takes a picture, which is developed, retouched, printed, and distributed.
Wrist camera	Cities and Towns	(same assignment as above). Larger, more capable devices are like cities, in which many services are available. Smaller, less capable devices are like small cities, or even villages, where fewer services are available. Transfer of files is like a train moving from one municipality to another.
Variable transparency window	Chameleon	The window will use nanotechnology to vary opaqueness depending on sensor readings, e.g., temperature. It can also generate decorative patterns. This is the architecture project.

Refactoring



Source: Fowler



“Disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior“

Example refactoring techniques:

- Encapsulate attribute (field) into function
- Replace conditional with dynamic binding
- Extract routine (method)
- Rename routine or attribute
- Move routine or attribute to another class
- Pull up, pull down

Used in agile methods as a substitute for upfront design

Practices

Part D: Practices 2: Development

What we have seen:

Practices that have exerted significant influence on the way
We develop software: pair programming, refactoring
.. They are complements, not replacements, for more
upfront-style techniques



Thank you



For general enquiries, contact:

Please contact the Head Teaching Assistant: Xingyu Pan (Star), Xingyu.Pan@ucdconnect.ie