COM3005J: Agile Processes
Agile Practices

Dr. Anca Jurcut
E-mail: anca.jurcut@ucd.ie

School of Computer Science and Informatics
University College Dublin

Beijing-Dublin International College

Agile Practices

1: Meetings

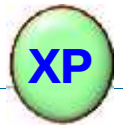
2: Development

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

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

Review Meeting

Review work:

- Completed
- Not completed

Present and demo completed work to stakeholders

Incomplete work cannot be demonstrated

4-hour time limit

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

XP

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 senor 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

Agile Practices

1: Meetings

2: Development

3: Release

4: Testing and quality

5: Management and others