Agile Software Development

Pressman

agile

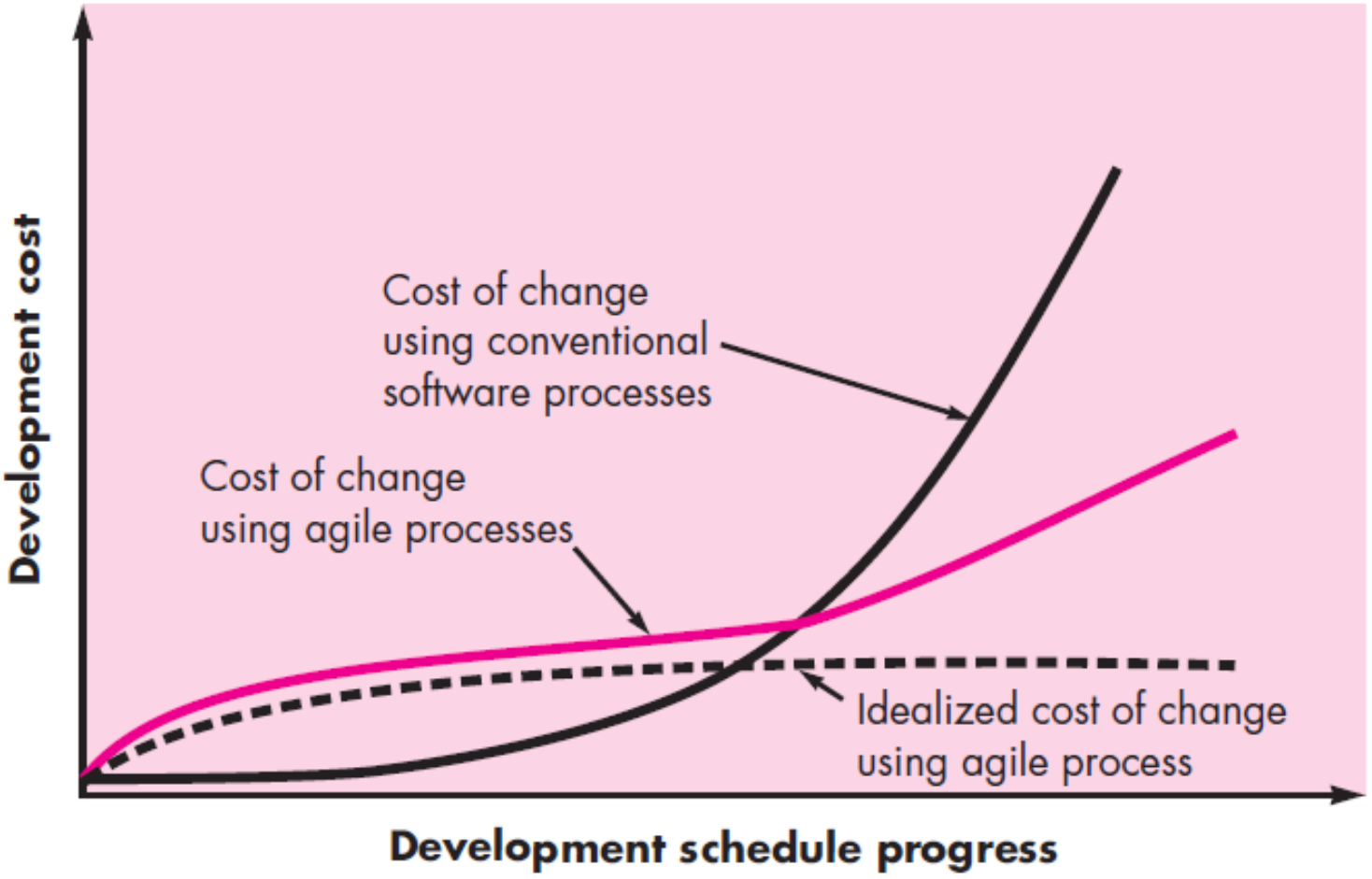
- able to move quickly and easily

- 2001 – Kent Back

There is value in the items on the right, we value the items on the left more.

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

Change costs as a function of time in development



Assumptions addressed by an Agile process

- It is difficult to predict in advance which **software requirements** will persist and which will change.
- For many types of software, **design and construction** are interleaved, they must be performed in tandem.
- **Analysis, design, construction, and testing** are not as predictable, as we like.

Principles of agile methods

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

12 agility principles – Agile Alliance

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

12 agility principles – Agile Alliance

- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

12 agility principles – Agile Alliance

- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.

12 agility principles – Agile Alliance

- Simplicity - the art of maximizing the amount of work not done - is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Jim Highsmith

- “Traditional methodologists are a bunch of stick-in-the-muds who’d rather produce flawless documentation than a working system that meets business needs.”

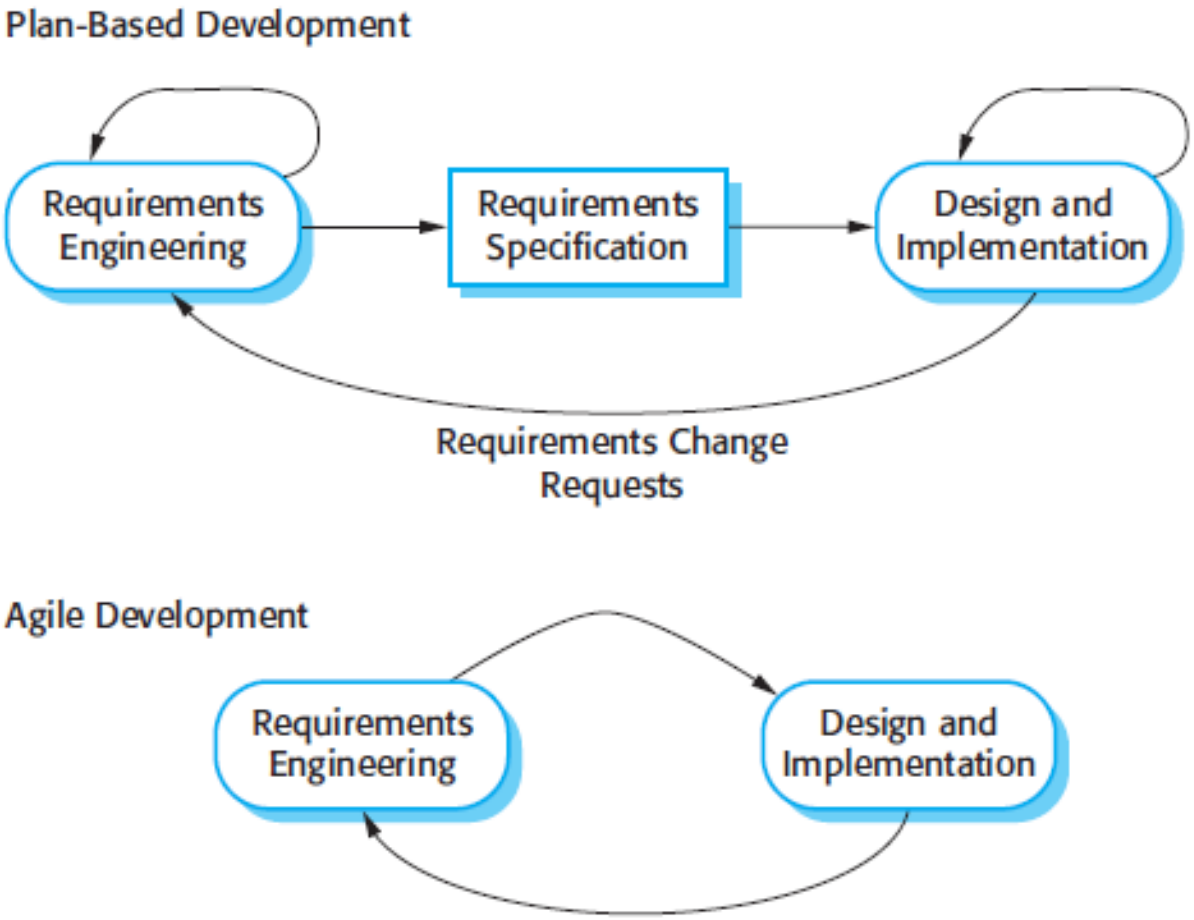
Cockburn and Highsmith

- *Agile process molds to the needs of the people and team, not the other way around.*

What key traits must exist among the people on an effective software team?

- Competence
- Common Focus
- Collaboration
- Decision making ability
- **Fuzzy problem solving ability**
- Mutual trust and respect
- Self-organization

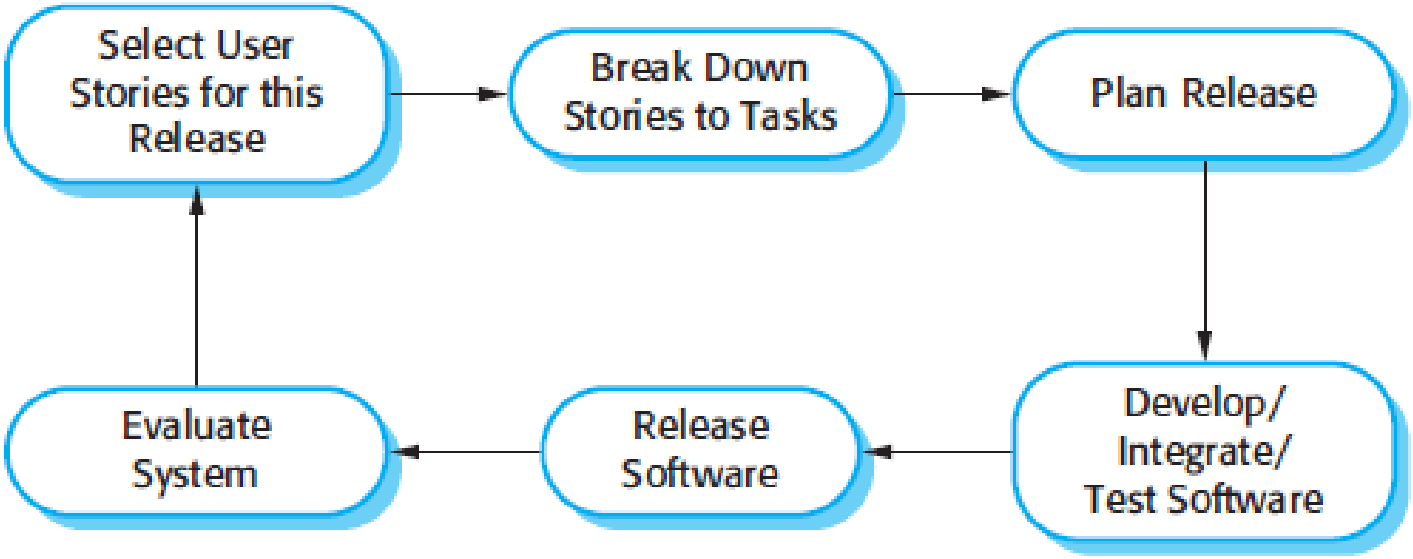
Plan driven and agile specification



Extreme Programming (XP)

- Most widely used approach to agile software development.
- Kent Beck
- a variant of XP, called *Industrial XP (IXP)*
  - targets the agile process specifically for use within large organizations.

XP release cycle



Story card

- The requirements are not specified as lists of required system functions.
- The system customer is part of the development team and discusses scenarios with other team members.
- Together, they develop a '**story card**' that encapsulates the customer needs.
- The development team then aims to implement that scenario in a future release of the software.

Example Story Card for the mental health care patient management system

- This is a short description of a scenario for prescribing medication for a patient.

Prescribing Medication

Kate is a doctor who wishes to prescribe medication for a patient attending a clinic. The patient record is already displayed on her computer so she clicks on the medication field and can select current medication', 'new medication' or 'formulary'.

If she selects 'current medication', the system asks her to check the dose. If she wants to change the dose, she enters the dose and then confirms the prescription.

If she chooses 'new medication', the system assumes that she knows which medication to prescribe. She types the first few letters of the drug name. The system displays a list of possible drugs starting with these letters. She chooses the required medication and the system responds by asking her to check that the medication selected is correct. She enters the dose and then confirms the prescription.

If she chooses 'formulary', the system displays a search box for the approved formulary. She can then search for the drug required. She selects a drug and is asked to check that the medication is correct. She enters the dose and then confirms the prescription.

The system always checks that the dose is within the approved range. If it isn't, Kate is asked to change the dose.

After Kate has confirmed the prescription, it will be displayed for checking. She either clicks 'OK' or 'Change'. If she clicks 'OK', the prescription is recorded on the audit database. If she clicks on 'Change', she reenters the 'Prescribing medication' process.

Task 1: Change Dose of Prescribed Drug

Task 2: Formulary Selection

Task 3: Dose Checking

Dose checking is a safety precaution to check that the doctor has not prescribed a dangerously small or large dose.

Using the formulary ID for the generic drug name, look up the formulary and retrieve the recommended maximum and minimum dose.

Check the prescribed dose against the minimum and maximum. If outside the range, issue an error message saying that the dose is too high or too low.

If within the range, enable the 'Confirm' button.

Extreme Programming Practices

Principle or practice	Description
Incremental planning	Requirements are recorded on Story Cards and the Stories to be included in a release are determined by the time available and their relative priority. The developers break these Stories into development 'Tasks'. See Figures 3.5 and 3.6.
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.

Extreme Programming Practices

Pair programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
On-site customer	A representative of the end-user of the system (the Customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

Degrade the software

- A general problem with incremental development is that it tends to degrade the software structure, so changes to the software become harder and harder to implement.

Refactoring

- Extreme programming tackles this problem by suggesting that the software should be constantly **refactored**.
- This means that the programming team look for possible improvements to the software and implement them immediately.

Examples of refactoring

- reorganization of a class hierarchy to remove duplicate code.
- the tidying up and renaming of attributes and methods.
- the replacement of code with calls to methods defined in a program library.

Testing in XP

- **Test-first development** - you write the tests before you write the code. This means that you can run the test as the code is being written and discover problems during development.

Testing in XP

- **Incremental test development from scenarios**
  - The customer who is part of the team writes tests as development proceeds.

Testing in XP

- **user involvement in the test development and validation** - The role of the customer in the testing process is to help develop acceptance tests for the stories that are to be implemented in the next release of the system.

Testing in XP

- **the use of automated testing frameworks** - Test automation is essential for test-first development. **Junit** is a widely used example of an automated testing framework.

Test case description for ‘dose checking’

Test 4: Dose Checking

Input:

1. A number in mg representing a single dose of the drug.
2. A number representing the number of single doses per day.

Tests:

1. Test for inputs where the single dose is correct but the frequency is too high.
2. Test for inputs where the single dose is too high and too low.
3. Test for inputs where the single dose × frequency is too high and too low.
4. Test for inputs where single dose × frequency is in the permitted range.

Output:

OK or error message indicating that the dose is outside the safe range.

Pair Programming

- An innovative practice that has been introduced in XP is that programmers work in pairs to develop the software.
- They actually sit together at the same workstation to develop the software.
- However, the same pairs do not always program together.
- Rather, pairs are created dynamically so that all team members work with each other during the development process.

Advantages of pair programming

- **Egoless programming** - It supports the idea of collective ownership and responsibility for the system.

- **Cheaper inspection process** - It acts as an informal review process because each line of code is looked at by at least two people.

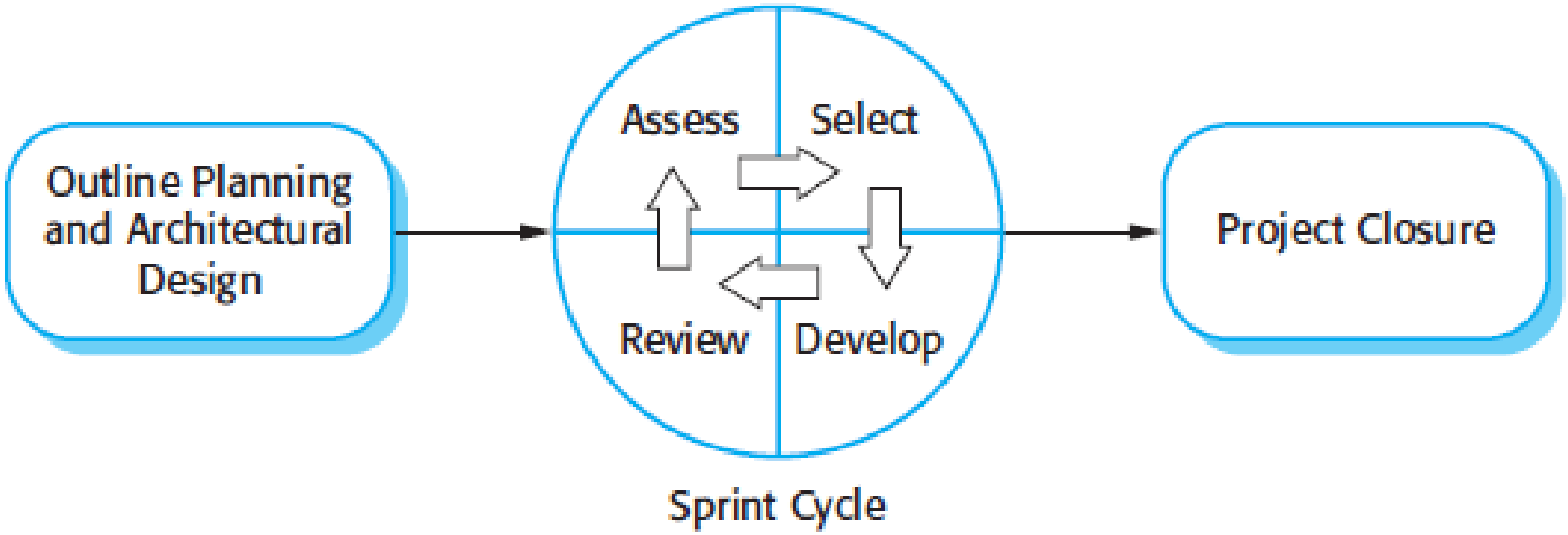
Advantages of pair programming

- It helps support refactoring.
- The sharing of knowledge that happens during pair programming is very important as it reduces the overall risks to a project when team members leave.
- The pairs discuss the software before development so probably have fewer false starts and less rework.

Agile project development

- The **Scrum approach** is a general agile method but its focus is on managing iterative development rather than specific technical approaches to agile software engineering.

Scrum management process



Scrum management process

- There are three phases in Scrum.

1. The first is an outline planning phase where you establish the general objectives for the project and design the software architecture.

2. This is followed by a series of sprint cycles, where each cycle develops an increment of the system.

3. Finally, the project closure phase wraps up the project, completes required documentation such as system help frames and user manuals, and assesses the lessons learned from the project.

Scrum Sprint

- The innovative feature of Scrum is its central phase, namely the sprint cycles.
- A Scrum sprint is a planning unit in which the work to be done is assessed, features are selected for development, and the software is implemented.
- At the end of a sprint, the completed functionality is delivered to stakeholders.

Characteristics of Scrum process

- Sprints are fixed length, normally 2–4 weeks. They correspond to the development of a release of the system in XP.