Computers II Lesson 4

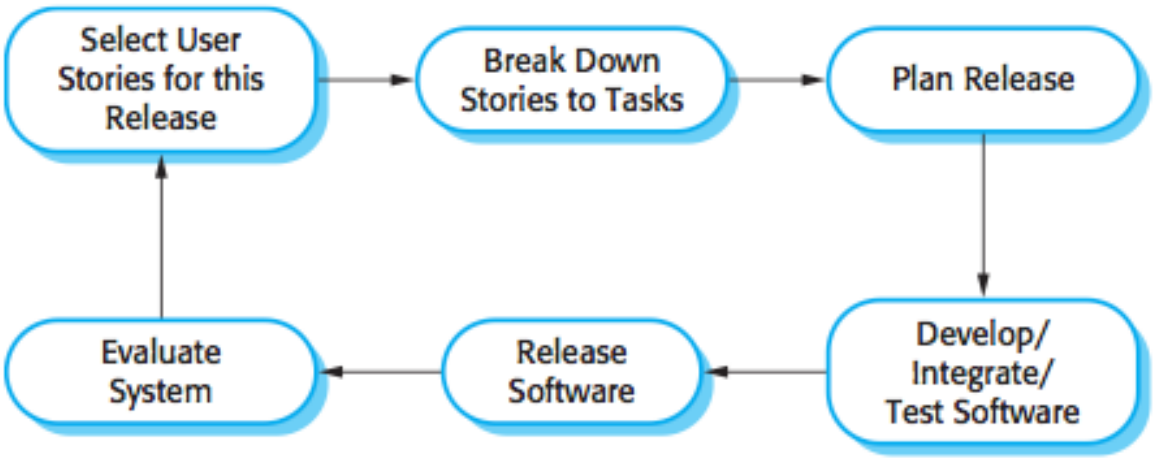
4.0 Extreme Programming

Extreme programming (XP) is best known and most widely used of the agile methods.

It was named extreme programming because the approach was developed by pushing recognized good practices to extreme levels.

In XP, several new versions of a system may be developed by different programmers, integrated and tested in a day.

- In extreme programming, requirements are expressed as scenarios called user stories, which are implemented directly as a series of tasks.
- Programmers work in pairs and develop tests for each task before writing the code.
- All tests must be successfully executed when new code is integrated into the system.
- There is a short time gap between releases of the system.



1. Incremental development is supported through small, frequent releases of the system. Requirements are based on simple customer stories or scenarios that are used as a basis for deciding what functionality should be included in a system increment.

2. Customer involvement is supported through the continuous engagement of the customer in the development team. The customer representative takes part in the development and is responsible for defining acceptance tests for the system.

3. People, not process, are supported through pair programming, collective ownership of the system code, and a sustainable development process that does not involve excessively long working hours.

4. Change is embraced through regular system releases to customers, test-first development, refactoring to avoid code degeneration, and continuous integration of new functionality.

5. Maintaining simplicity is supported by constant refactoring that improves code quality and by using simple designs that do not unnecessarily anticipate future changes to the system.

- In an XP process, customers are intimately involved in specifying and prioritizing system requirements.

- 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 story cards are the 'planning game'

- The development team then aims to implement that scenario in a future release of the software.

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.
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.

- 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.

- Essentially, the development proceeds by finding workarounds to problems, with the result that code is often duplicated, parts of the software are reused in inappropriate ways, and the overall structure degrades as code is added to the system.

- Extreme programming tackles this problem by suggesting that the software should be constantly refactored.

- The programming team looks for possible improvements to the software and implement them immediately. When a team member sees code that can be improved, they make these improvements even in situations where there is no immediate need for them.

4.1 Story Cards

- When story cards have been developed, the development team breaks them down into tasks and estimates the effort and resources required for implementation.

- This usually involves discussions with the customer to refine the requirements.

- The customer then prioritizes the stories for implementation, choosing those stories that can be used immediately to deliver useful business support.

- The intention is to identify useful functionality that can be implemented in about two weeks, when the next release of the system is made available to the customer.

- As requirements change, the unimplemented stories change or may be discarded. If changes are required for completed system, new story cards are

developed and the customer decides whether these changes should have priority over new functionality.

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.

4.2 XP Testing

- Test-first development is one of the most important innovations in XP.
- XP includes an approach to testing that reduces the chances of introducing undiscovered errors into the current version of the system.
- Instead of writing some code and then writing tests for that code, 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.

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.

An edge case is a problem or situation that occurs only at an extreme (maximum or minimum) operating parameter.

For example:

- A function that divides two numbers might be tested using both very large and very small numbers. This assumes that if it works for both ends of the magnitude spectrum, it should work correctly in between.

Boundary case refers to the behavior of a system when one of its inputs is at or just beyond its maximum or minimum limits.

If an input field is meant to accept only integer values 0–100, entering the values -1, 0, 100, and 101 would represent the boundary cases.

For example:

- A common technique for testing boundary cases is with three tests: one on the boundary and one on either side of it. So for the previous example that would be -1, 0, 1, 99, 100, and 101.