Extreme Programming

Chapter 3.3

CMPT 276
© Dr. B. Fraser
Based on slides from Software Engineering 9th ed, Sommerville.

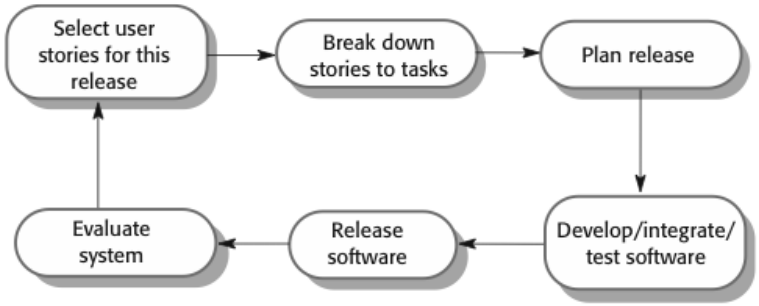
Topics

1) What is Extreme Programming (XP)?
2) How are programs tested in XP?

Extreme programming

A well known agile method.

- Extreme Programming (XP) takes an 'extreme' approach to...
  - New versions may be built several times per day;
  - Increments delivered to customers every 2 weeks;
  - All tests (automated) must be pass for...



XP practices

Principle or practice	Description
	Requirements are recorded on Story Cards. Stories included in increments based on priority and available time. Developers break Stories down into Tasks.
	Start small: First develop minimal useful set of functions which deliver business value. Release often: Releases are frequent and incrementally add functionality.
Simple design	Design is only done to support... not any possible future ones.
	Automated unit tests are written for an object before the object is written.
	Code kept simple and maintainable by continuous refactoring by all developers.

XP practices cont.

Principle or practice	Description
	Developers work in pairs, always checking each other's work and providing support.
	All developers work on all parts of the code. Shared responsibility for the code, and no one developer has all knowledge about an area.
	Changes integrated into system as soon as the are completed.
	Large amounts of overtime are discouraged: would compromise productivity and code quality.
	Customer representative is a full time member of the development team: brings the team requirements and priorities.

Requirements scenarios

- Customer or user is part of the XP team.
  - Responsible for..
- User requirements:
  - Written on cards as scenarios or...
  - Development team breaks them down into..
  - Tasks are the basis of schedule and cost estimates.
- Customer chooses the stories for inclusion in the next release based on their priorities and schedule estimates.

Story: Prescribing medication

Prescribing medication

The record of the patient must be open for input. Click on the medication field and select either 'current medication', 'new medication' or 'formulary'.

If you select 'current medication', you will be asked to check the dose; If you wish to change the dose, enter the new dose then confirm the prescription.

If you choose, 'new medication', the system assumes that you know which medication you wish to prescribe. Type the first few letters of the drug name. You will then see a list of possible drugs starting with these letters. Choose the required medication. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

If you choose 'formulary', you will be presented with a search box for the approved formulary. Search for the drug required then select it. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

In all cases, the system will check that the dose is within the approved range and will ask you to change it if it is outside the range of recommended doses.

After you have confirmed the prescription, it will be displayed for checking. Either click 'OK' or 'Change'. If you click 'OK', your prescription will be recorded on the audit database. If you click 'Change', you reenter the 'Prescribing medication' process.

Task cards for Prescribing medication

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, lookup 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.

User stories are broken down into tasks:

XP and change

- Conventional wisdom:
  - It is worth spending time and effort anticipating changes as this reduces costs later in the life cycle.
- XP's view:
  - XP uses refactoring to constantly improve code.
  - This makes changes easier when they have to be implemented.

Refactoring

- Developers look for possible code improvements and make these improvements
  - This improves the understandability of the code and reduces the need for documentation.
- Changes are easier to make because...
  - However, some changes require architecture refactoring and this is much more expensive.

Examples of refactoring

- Refactoring Examples:
  - to make its purpose clearer.
  - to make a long function shorter or reduce duplicate code.
  - to split a class which does 2 things into two classes.
  - to explicitly test a required condition.
- Single Responsibility Principle (SRP)
  -
  - It should have only one reason to change.

Pair programming

- Developers work in pairs..
  - Pairs change so everyone works together.
- Fosters of code and spreads knowledge across the team.
  - Reduce problem when key developers leave.
  - No one person blamed for bugs.
- Informal review process:
- Encourages refactoring:
  - Whole team gets benefit of clean code.
- Productivity with P.P. ≈ two people working independently.

XP Testing

Overview of testing in XP

- Testing is central to XP:
  -
- XP testing features:
  - Test-first development.
  - Incremental test development from scenarios.
  - User involvement in test development and validation.
  -

    used to run all component tests each time a release is built.

Test-first development

- Tests are executable programs
  - Usually rely on a testing framework such as JUnit.
- All tests (old and new) are run after every change:
  - Ensures new change doesn't..
- Test-first Advantages:
  -
  - Promotes good test coverage of code.
  - Ensures all code has up-to-date tests (no lag).
  - Ensures interfaces are testable.

Customer involvement

- Customer is part of development team:
  -

    for the stories being implemented in next release.
  - Writes tests as development proceeds.
    - All new code is therefore validated to ensure that it is what the customer needs.
- Challenge:
  - Customer may have limited time available, so may be reluctant to get involved in the testing process:

    "I gave you the requirements, can't you figure out if your code meets them?"

Test case description: 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.

Test automation

-
  before the task is implemented.
  - Automated testing framework (JUnit) runs stand-alone tests which simulate input and check result.
- Tests run whenever new functionality added.
  - Tests can be run quickly and easily.
  - Catch problems immediately.
- Automated verification:
  - Gives developers the confidence and security of knowing nothing broke.
  –

Extreme Programming Summary

- Well known agile method.
- Uses a range of good programming practices:
  - Frequent releases;
  - Continuous code improvement (refactoring);
  - Customer participation
- Testing:
  - Requires implementing tests first.
  - All automated tests must pass when an increment is integrated into a system.