SE202
Introduction to Software Engineering

Lecture 7.1

Software testing

Levels of testing

Questions??

I ask you to develop a software for Mr. A's project.

How would you do to make sure that he will like it??

Ans: software testing

How many of number of bugs line of code (in KLOC) does the industry estimate?

Ans: 15-50

Software Testing

- Testing is the process of exercising a program (a piece of code) with the specific intent of finding errors prior to delivery to the end user.
  - Errors can be that:
    - Part of the program or even a whole program does not meet the requirement OR/AND
    - The program does not work correctly under all conditions
- Testing goal?
  - Is always removing every single bug from a program a testing goal?
  - a *bug* is a flaw in a program that causes it to produce an incorrect result or to behave unexpectedly.

Reasons Bugs never die

- Deadlines
  - Deadlines are driven by management, competitors or marketing department
  - No time to fix every bugs
  - If you try to fix every bug, you'll never release anything
- Consequences
  - Sometimes a bug fix might have undesirable consequences
  - E.g. fixing saving the spiral's color in a drawing application affects the format of saving picture files.
- Obsolescence
  - Over time, some features may become less useful
  - E.g. if an operating system has a bug in a floppy drive controller that limits its performance

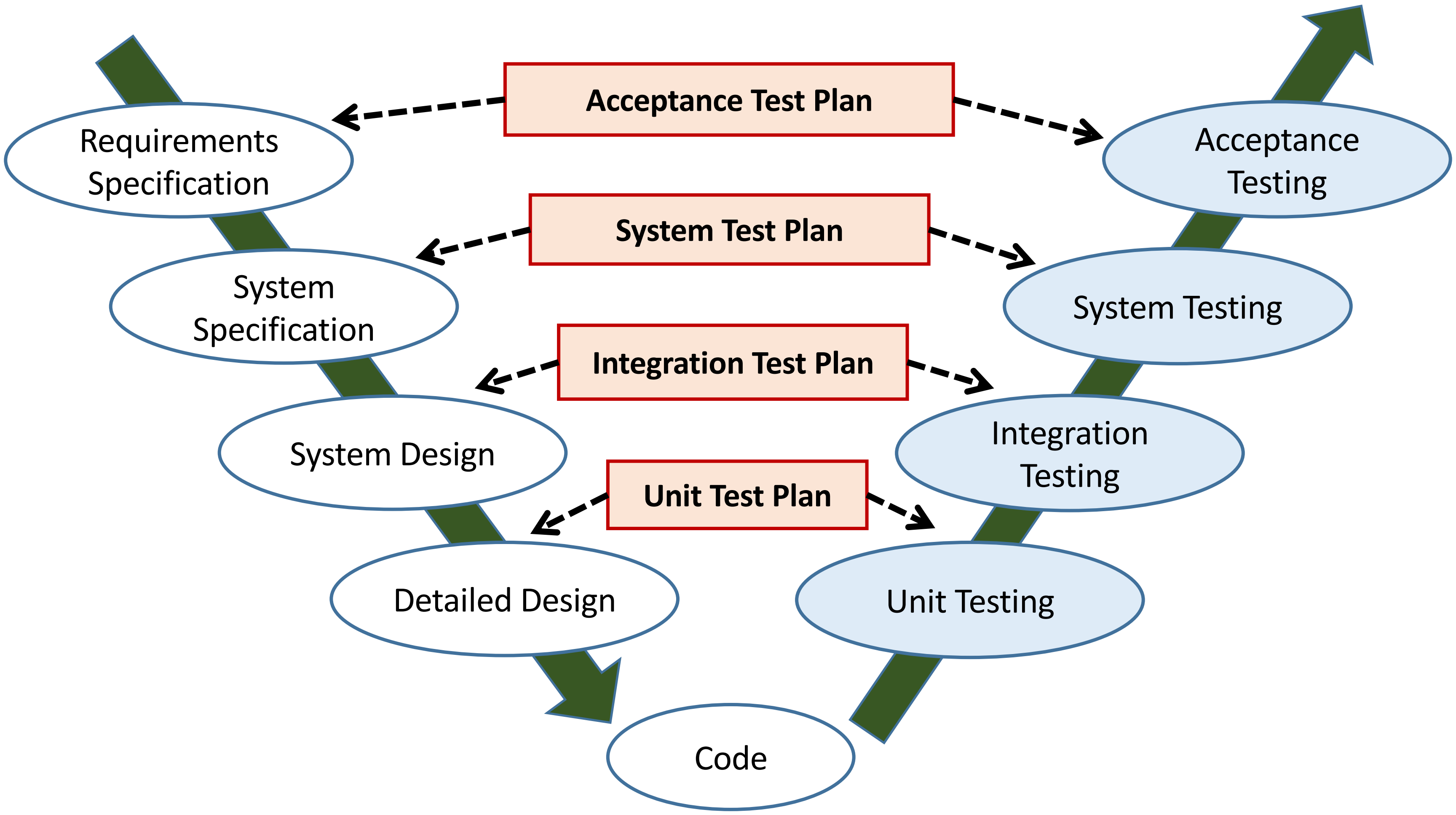
Reasons Bugs never die

- It’s Not a Bug
  - Sometimes users think a feature is a bug when actually they just don’t understand what the program is supposed to do.
  - If the documentation is incomplete or unclear, this is a “documentation bug”
- Fixing Bugs Is Dangerous
  - When you fix a bug, there’s a chance that you’ll fix it incorrectly, so your work doesn’t actually help.
  - There’s also a chance that you’ll introduce one or more new bugs when you fix a bug.

V & V

- Verification refers to the set of tasks that ensure that software correctly implements a specific function.
- Validation refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements. Boehm [Boe81] states this another way:
  - Verification: "Are we building the product right?"
  - Validation: "Are we building the right product?"

V-model of Development



Who Tests the Software?

developer

independent tester

Understands the system

but, will test "gently"

and, is driven by "delivery"

Must learn about the system,

but, will attempt to break it

and, is driven by quality

Testing Strategy

- We begin by ‘testing-in-the-small’ and move toward ‘testing-in-the-large’
- For conventional software
    - The module (component) is our initial focus
    - Integration of modules follows
- For OO software
    - our focus when “testing in the small” changes from an individual module (the conventional view) to an OO class that encompasses attributes and operations and implies communication and collaboration

Levels of Testing

1. **Development testing**: The system is tested ***during*** *development* to discover *bugs and defects*.
   - A bug is the result of a coding error
   - A defect is a deviation from the requirements
2. **Release testing**: A separate testing team test a complete version of the system ***before*** *it is released to users*.
3. **User testing**: Users or potential users of a system test the system in *their own environment*.

1. Development Testing

Includes all testing activities that are carried out *by the team developing* the system.

- Unit testing
- Integration testing
- Regression testing
- Automated testing
- Component interface testing
- System testing

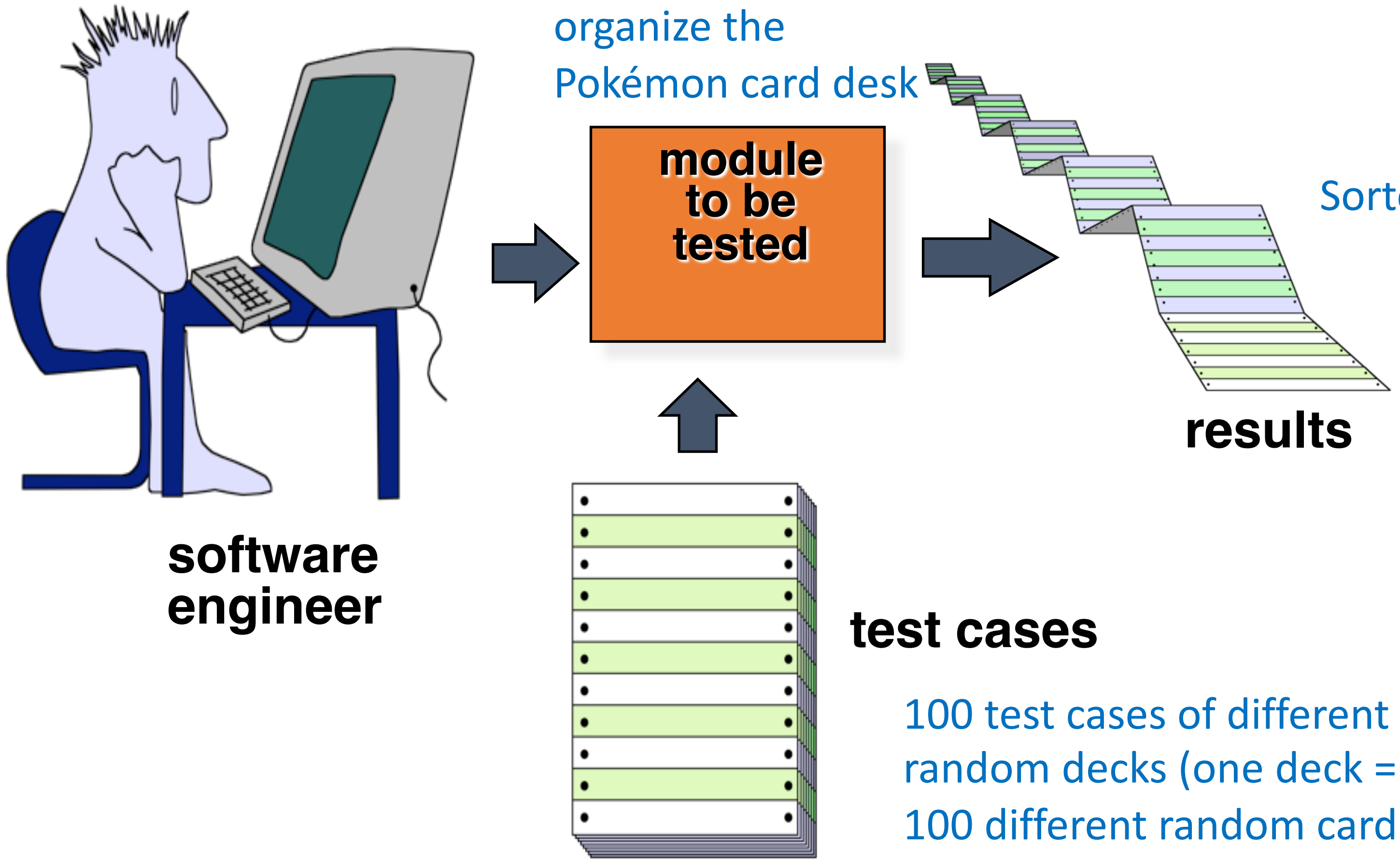
Unit Testing

- The process of verifying the correctness of a specific piece of code in isolation
- Units may be:
  - Individual *functions or methods* within an object
  - *Object classes* with several attributes and methods: all operations, all attributes, exercising the object in all possible states (Inheritance makes it difficult to design tests…)
  - *Composite components with defined interfaces* used to access their functionality.
- Test unit as thoroughly as possible because the unit will get harder to fix later!!

Unit Testing

Typically, a test is another piece of code that invokes the code you are trying to test and then validates the result

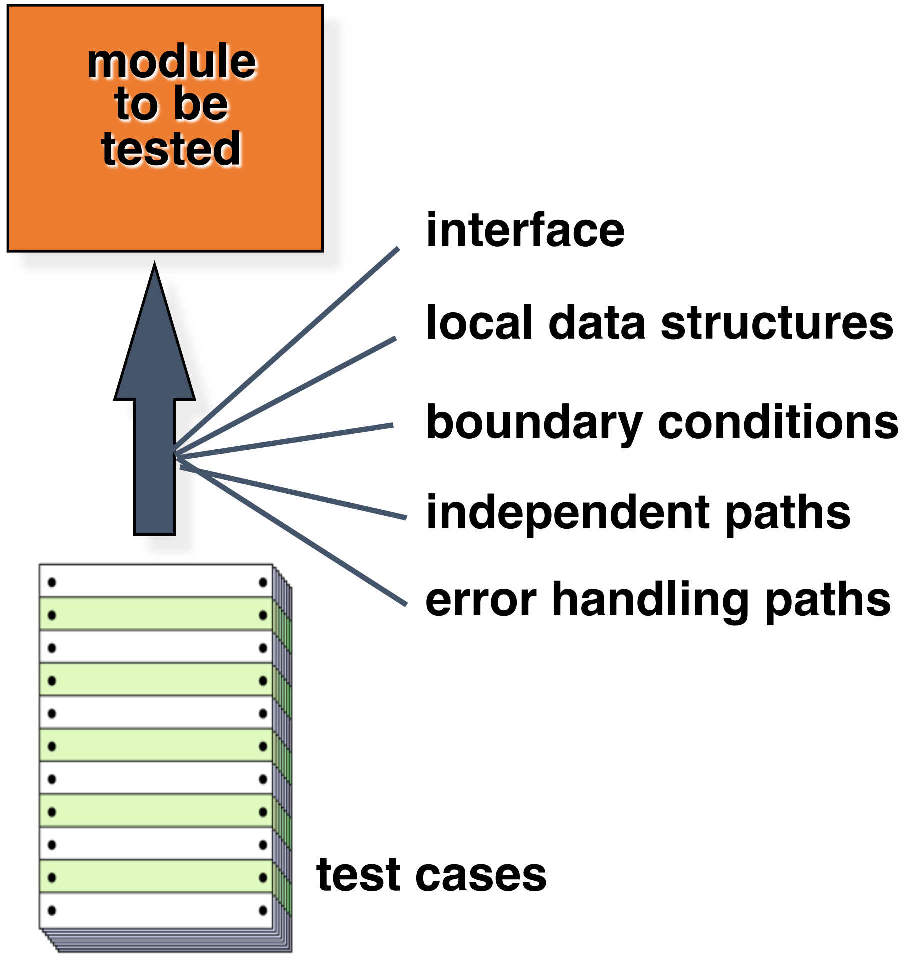
E.g. Method that organize the Pokémon card desk



Sorted deck results

100 test cases of different random decks (one deck = 100 different random card)

Unit Testing

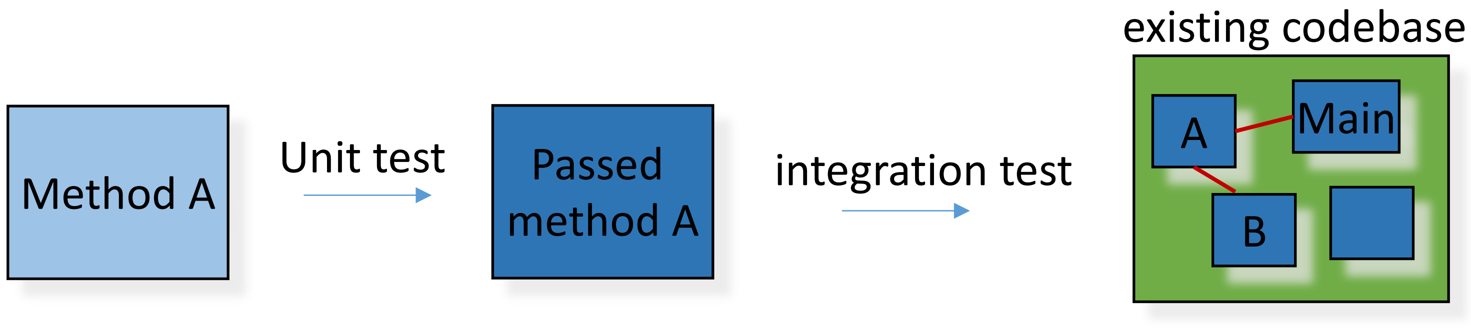


Integration testing

- The process of verifying that the new method works and plays well with others.
- After you write a method and use unit tests to verify the method, it’s time to integrate it into the existing codebase.
- It checks that existing code calls the new method correctly, and that the new method can call other methods correctly.

Integration testing

- Example: Program that helps to do duct type projects



Method A of listing duct type info for a particular project

the main program can call the method successfully, and the new method A can call method B of fetching duct tape roll lengths and prices.

Integration testing consequences

- May introduce one or more new bugs
- E.g. Use the previous example, the integration the method A to the existing codebase affect the program to new duct online.
  - The method A might open the pricing database and accidentally leave a price record locked. You might not notice this during unit testing and integration testing, but the tape ordering part of the program won’t work if that record is locked.

Regression Testing

- The process of testing a system to check that **changes have not ‘broken’ previously working code**.
- In a manual testing process, regression testing is expensive but, with automated testing, it is simple and straightforward.
- All tests are rerun every time a change is made to the program.
- Recommended steps
    1. Finish unit testing a piece of code
    2. Then integration testing to make sure it fits in where it should and that it didn’t break anything obvious.
    3. Then regression testing to see if it broke something non-obvious.

Automated Testing

- A testing tool that do the testing (unit testing and integration testing) for you.
  - Automated testing tools let you define tests and the results they should produce.
  - After running a test, the testing tool can compare the results it got with expected results.
- Unit testing should be automated whenever possible.
- Make use of a **test automation framework** (such as Junit, Emma) to write and run your program tests.
- Unit testing frameworks provide generic test classes that can be extended, run all of the tests implemented and report on the result of the tests.

Component interface testing

- The process of testing all interactions between components
- Focus is on showing that the **component interface** behaves according to its specification.
- **Interface Testing** aims at detecting faults due to interface errors or invalid assumptions about interfaces.

Interface Types	Common Errors
parameter interfaces, shared memory interfaces, procedural interfaces (calls), message passing interfaces	interface misuse (e.g. parameters in the wrong order), interface misunderstanding, timing errors (e.g. out-of-date info)

System Testing

- The process of testing an end-to-end run-through of the whole system.
- Explore many possible paths of interaction
  - compatible (*right data type/format*)
  - interacting correctly
  - transferring the *right data*
  - transferring at the *right time* across their interfaces.

System Testing Example

Suppose you're writing a program to manage office chair information like type, color, amount available, and size. Also suppose the program includes only a login screen and a single form that uses a grid to display office chair information.

- The possible operations for the system testing
  - Start the program and click Cancel on the login screen.
  - Start the program, enter invalid login, click OK, verify that you get an error message, and finally click Cancel to close the login screen.
  - Start the program, enter invalid login, click OK, verify that you get an error message, enter valid login information, and click OK. Verify that you can log in.
  - Log in, view saved information, and close the program. Log in again and verify that the information is unchanged.
  - Log in, add new dirt information, and close the program. Login in again and verify that the information was saved.
  - Log in, edit some dirt information, and close the program. Login in again and verify that the changes were saved.
  - Log in, delete a dirt information entry, and close the program. Login in again and verify that the changes were saved.

2. Release Testing

- The process of **testing a particular release of a system** that is intended for use outside of the development team.
- The *primary goal* is to convince the supplier of the system that it is good enough for use.
  - e.g. delivering specified functionality, performance and dependability, and not failing during normal use
- Usually a **black-box testing** process where tests are only derived from the system specification.

Release Testing vs. System Testing

Release testing is a form of system testing.

Release Testing	**System Testing**
• Conducted by a separate team that has not been involved in development • Focusing on checking that the system meets its requirements and is good enough for external use (validation testing)	• Conducted by the development team • Focusing on discovering bugs in the system (defect testing)

Performance Testing

- Part of release testing may involve testing performance and reliability of a system.
- Tests should reflect the profile of use of the system.
- Performance tests usually involve planning a series of tests where **the load is steadily** increased until the system performance becomes unacceptable.
- **Stress testing** is a form of performance testing where the system is **deliberately overloaded** to test its failure behavior.

3. User Testing

- A stage in the testing process where users or customers provide input and advice on system testing.
- User testing is essential, even when comprehensive system and release testing have been carried out.
- **Influences from the user’s working environment** have a major effect on the reliability, performance, usability and robustness of a system. These cannot be replicated in a testing environment.

Types of User Testing

- **Alpha testing**

  Users of the software *work with the development team* to test the software at the developer's site.

- **Beta testing**

  Users experiment *a release of the software* and raise problems that they discover with the system developers.

- **Acceptance testing**

  Customers test a system to decide whether or not it is *ready to deployed in the customer environment*. Primarily for custom systems.

Summary

- Software testing includes development testing, release testing, and user testing.
- **Development testing** includes unit testing, component testing, and system testing
- **Release testing** checks a particular release of the system is good enough for use and may involve performance testing (or stress testing).
- **User testing** may include acceptance testing where customers decide if the software is good enough to be deployed and used in its operational environment.