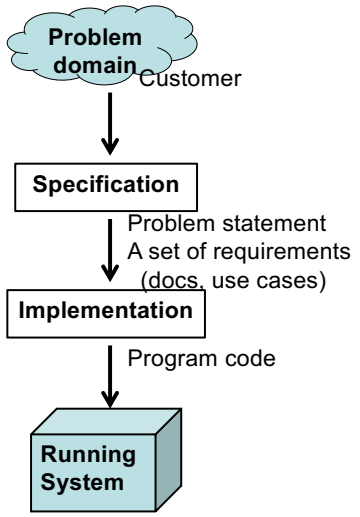
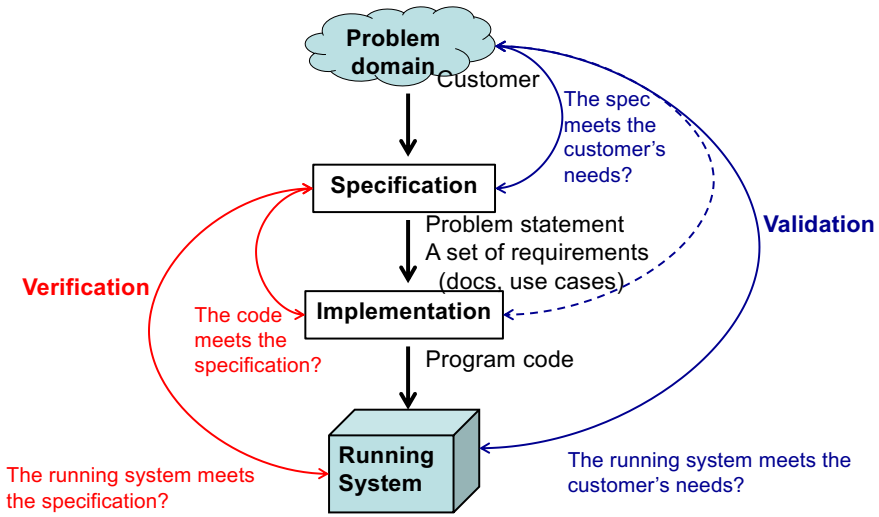
Software Testing: Verification and Validation

Software Development



Verification and Validation (V&V)

- Verification
  - Testing whether a system is developed in accordance with its specification (i.e., a set of gathered requirements).
    - Ensures you built it right.
- Validation
  - Testing whether a system meets your customer's needs.
    - Ensures you built the right thing.



Defects in V&V

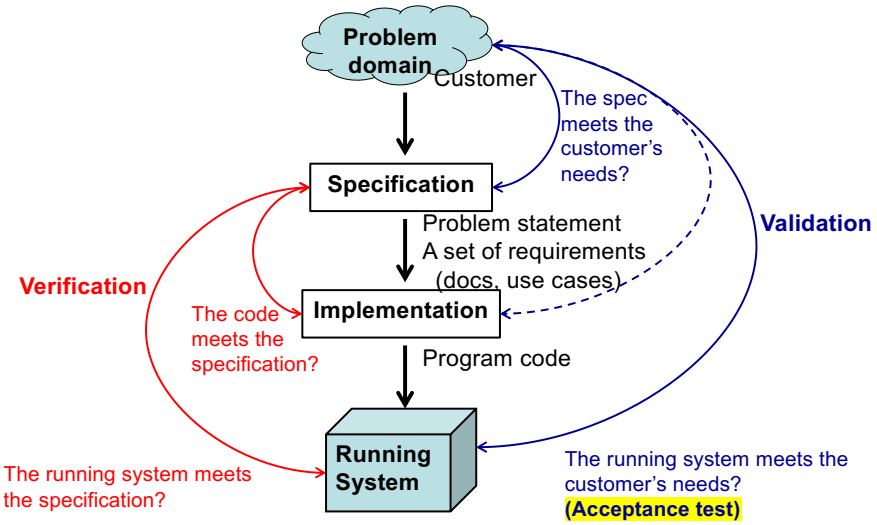
- Defects found in verification
  - Found when the implementation and/or running system fail to meet the specification.
    - e.g., The spec. of a printer's firmware states that the printer stops printing when its paper tray is empty.
      - However, the firmware doesn't stop the printer when a tray is empty.

- Defects found in validation
  - Found when the specification is wrong or misses the customer's needs.
    - e.g., The firmware's spec. states nothing about what the printer should do when its tray is empty.
      - Thus, the firmware does not stop printing when a tray gets empty.
      - However, the customer wants the printer to stop.

Importance of Validation

- Need to correctly define requirements in the specification, so…
  - Developers can clearly tell what needs/features to implement and how to implement them.

- However, it is not always easy to make the specification sufficiently comprehensive, so..
  - Developers do not miss the customer's needs.
  - Requires numerous "what-if" discussion.
    - What if a tray gets empty?
      - The current on-going print job should stop immediately?
      - What if another tray has papers?
      - Can the printer still accept extra print jobs from computers?
  - Requires "acceptance test" by the customer



An Example Defect in Validation

- Firmware for Boeing 787's generator control unit (GCU)
  - Does periodic "status check" every 10 milliseconds.
  - Had a counter (timestamp) with signed (!) 32-bit integer.
    - 2^31=2,147,483,648 (> 2B)
    - 10 msec * 2,147,483,648 = 248.551 days
    - An integer overflow occurs once GCU has continuously operated for 248.551 days.

Counter	Status
0	G
1	G
2	G
⋮	⋮

- Firmware for 787's generator control unit (GCU)
  - Had a counter (timestamp) with signed (!) 32-bit integer.
    - An integer overflow occurs once GCU has continuously operated for 248.551 days.
- GCU fails if it is powered on for 248+ days.

Counter	Status
0	G
1	G
2	G
⋮	⋮

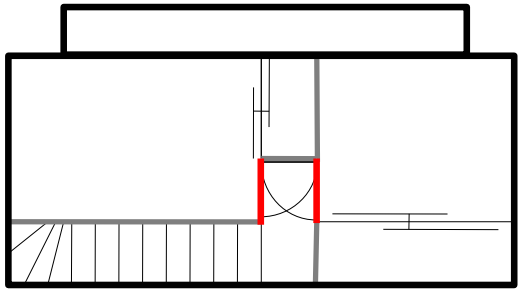
- A 787 aircraft has 4 GCUs.
  - If all of them are powered on at the same time, the aircraft can lose its control in 248+ days.

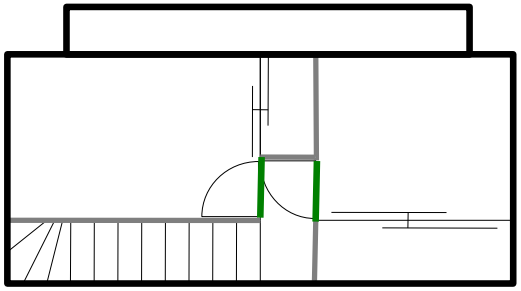
- Customer
  - Didn't consider and wasn't asked how long a GCU should be able to keep running if it is not turned off.
    - Status check might look like a minor feature in GCU development.
  - Did consider or was asked about it, but it was not stated in the firmware specification.
- Developer
  - Didn't consider and wasn't instructed (by the specification) about up to how long a GCU should run if it is not turned off.
  - Decided to use the simplest data type for the counter and didn't have a chance to re-visit the decision.
    - Status check might look like a minor feature in firmware development.

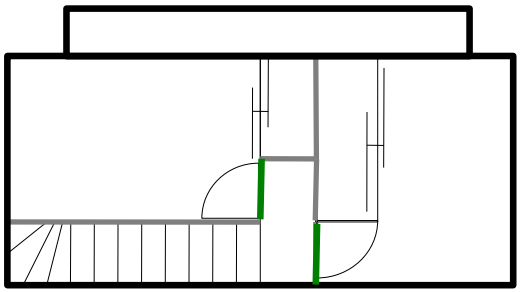
X-day Problems

- 248-day problem
- 494-day problem
  - Occurs if a counter/timer relies on an unsigned 32-bit integer
    - Server OSes, WiFi routers, network switches, etc. etc.
- 24-day and 49-day problems
  - Occur if a counter/timer relies on an signed/unsigned 32-bit integer and its counting/timing resolution is 1 millisecond.
- 830-day problem
  - Occurs if a counter/timer relies on an unsigned 32-bit integer and its counting/timing resolution is 60 Hz (1/60 second; 16.67 msec)
- Year 2038 problem (Unix millennium problem)
  - Many OSes have a timer that counts time in second from 1970/1/1 0:00:00, using a signed integer. The timer will overflow at January 19 in 2038.

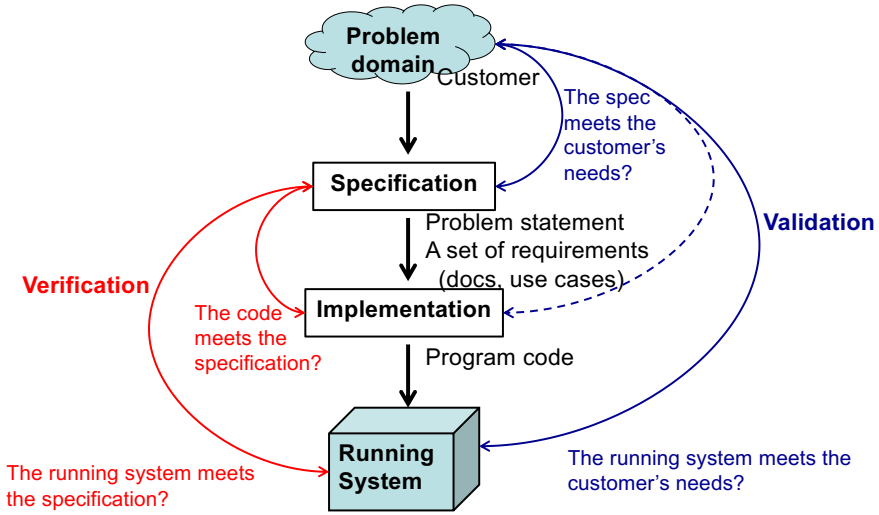
When I was a kid...





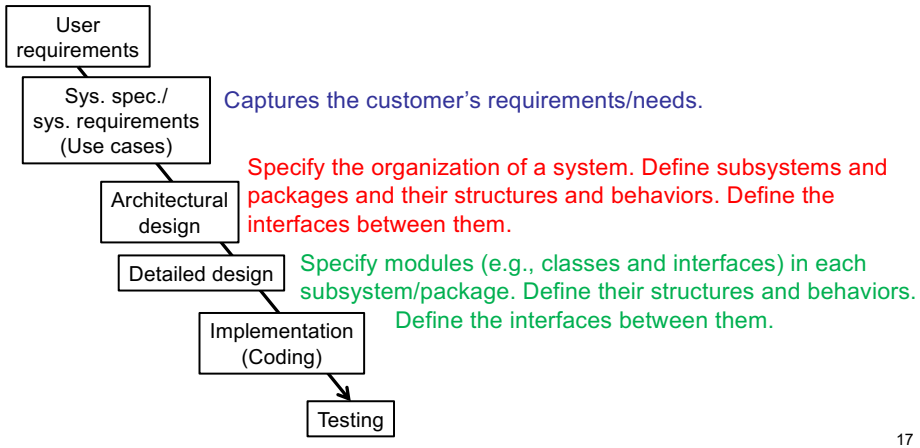


Software Development



Waterfall Process Model

- One of the earliest models to describe development processes.

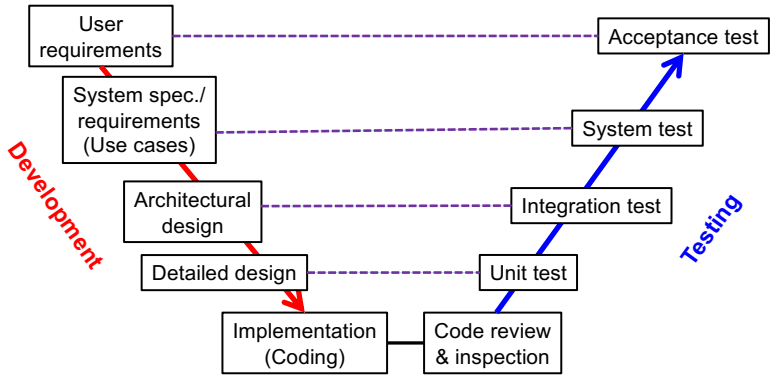


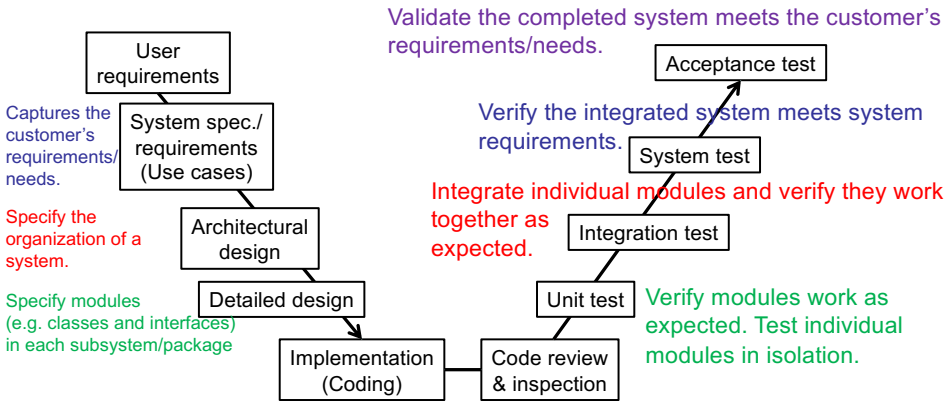
Problems in Waterfall Process

- Defects are found at the end of the project.
  - Testing does not take place until the end of the project.
- It is often too late and too expensive to push feedback up the waterfall.

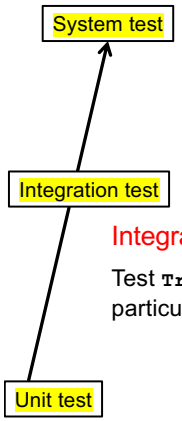
V-Model

- Extends the waterfall model.
  - Testing phase is expanded
- Explicitly states which testing phase corresponds to which development phase.





Example Tests



Verify the integrated system meets system requirements.

Verify that…
- Paper feeding stops when a particular tray becomes empty.
- A error notification is sent to OS's printing facility (e.g. a print queue)
  OR
- Paper feeding re-starts with another tray if it is not empty.

Integrate individual modules and verify they work together as expected.

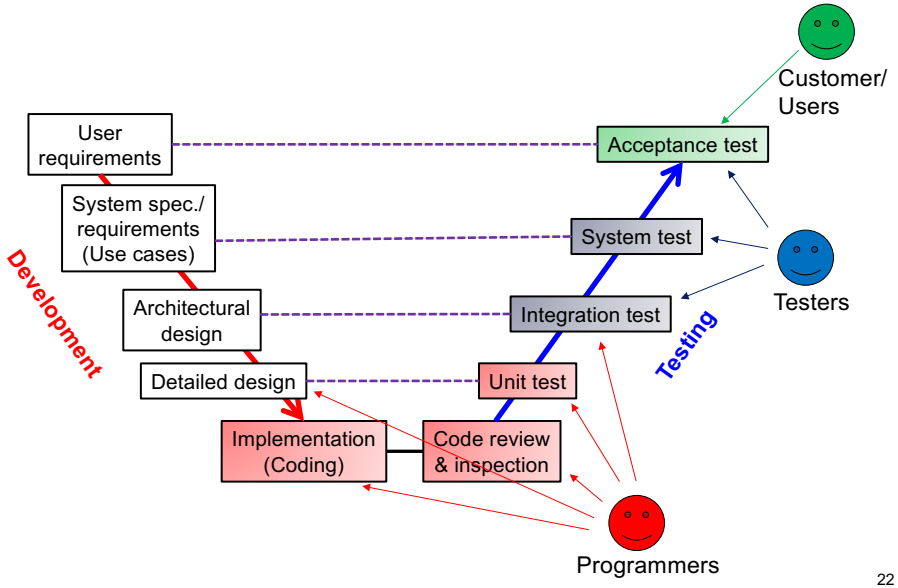
Test `Tray` and `PaperFeeder` together to verify that paper feeding stops when a particular tray becomes empty:

```
if(!aTray.isEmpty()){ aPaperFeeder.stop(...); }
```

Verify modules work as expected. Test individual modules in isolation.

Test `Tray` to verify its `isEmpty()` returns true when a particular tray is empty.
Test `PaperFeeder` to verify its `stop()` stops paper feeding.

Division of Responsibilities



Test Levels and Test Types

- Test level
  - Corresponds to a "development level."
    - e.g., unit test, integration test, system test and acceptance test.
  - A group of test activities that are organized and managed together.
- Test type
  - Focuses on a particular test objective.
    - e.g. functional test, non-functional test, structural test, confirmation test, etc.
  - Takes place at one test level or at multiple levels.

- Test levels and test types are orthogonal.

	Functional test	Non-functional test	Structural test	Confirmation test
Acceptance test				
System test				
Integration test				
Unit test				
Code rev&insp.				

- Different projects have difference policies on which test types involve in which levels.

- For example…

	Functional test	Non-functional test	Structural test	Confirmation test
Acceptance test	X	X		
System test	X	X		X
Integration test	X	?	X	X
Unit test	X	?	X	X
Code rev&insp.	X	?	X	X