Overview and current status of embedded system

- What's Embedded system?
- What's Computer?
- What's Software?
- What's Software quality?
- What's Debugging?
- What's Testing?

Debugging and Testing

- **Debugging** : A process to prove that a program works correctly
  - Done by programmer
- **Testing** : A process to prove that a program DOES NOT work correctly
  - Done by QA

Debugging and Testing Con’t

- Debugging : Find bug and debug (Daily exercise)
  - (like find guilty and correct)
- Testing : Find just one bug that program can’t work, So REJECT!! (Final exam)
  - (like find guilty and Vo la! → reject)

Debugging and Testing Con't

- Both start from what's our purpose to test or debug?

- Generate test cases and test it.

Test cases

- General rule 1 test case / 10 LOC

  (job of project manager)

Such as

- Func A 200 LOC/20 test cases
- Func B 500 LOC/50 test cases

If they built

- Func A 50 test cases → less bugs
- Func B 5 test cases → Huge bugs

Test cases con’t

If they built

- Func A 50 test cases → less bugs
  - Know Func A very well, So they built plenty of test case
- Func B 5 test cases → Huge bugs
  - He don’t know function very good enought

Project management

If Program A

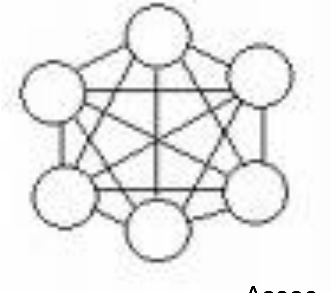
- 10 Engineers, 1 Manager, 1 Year project
- But 2 months delay

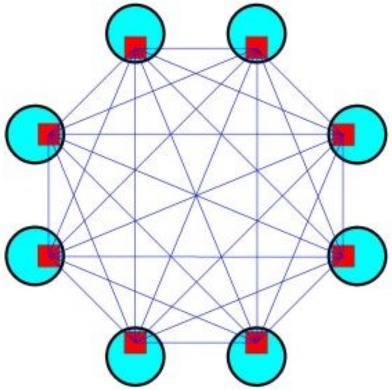
- 5 Engineers add more →
    - The project will be bigger delay, Why???

Project management con’t

- 5 Engineers add more →
  - The project will be bigger delay, Why???

- Educational problem
  - Some engineers have to teach the new ones.
- Communication complex
  - More people, more complex





Test case guide line

- A approx 1 test case/ 5 – 15 LOC
  - Language processor : 1 / 8 – 12 LOC
  - Online system : 1 / 5 – 10 LOC
  - Batch system : 1 / 10 – 15 LOC
- B
  - Normal cases : 60%
  - Abnormal cases : 15% (error case)
  - Boundary cases : 10% (0-10, test -1,11)
  - Environment cases : 15% (other driver, OS)

Admission fee

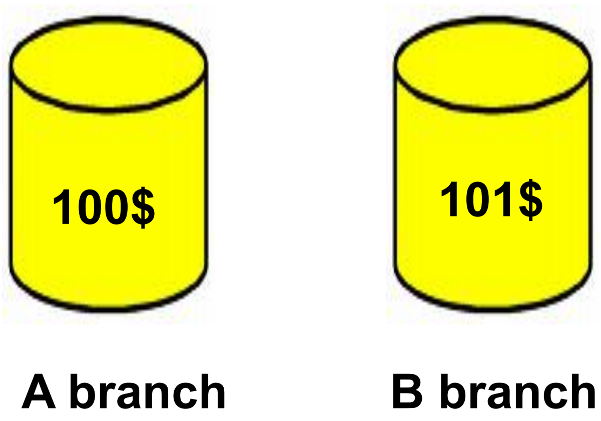
- Below 6 years old : Free
- 7 – 12 years old : 500 yen
- 13 – 18 years old : 800 yen
- 19 years and older : 1,000 yen
    - ***Do you find any bug?***

Bug collected

- Developers must report all the bugs.
- What must be reported?
  - When detected
    - ID number, Symptom of the bug, Who find out, When, Seriousness
  - When fixed
    - Which modules, Explain of the bug, Who fixed, When and What was test, By whom, Code before and after.

Example of bug

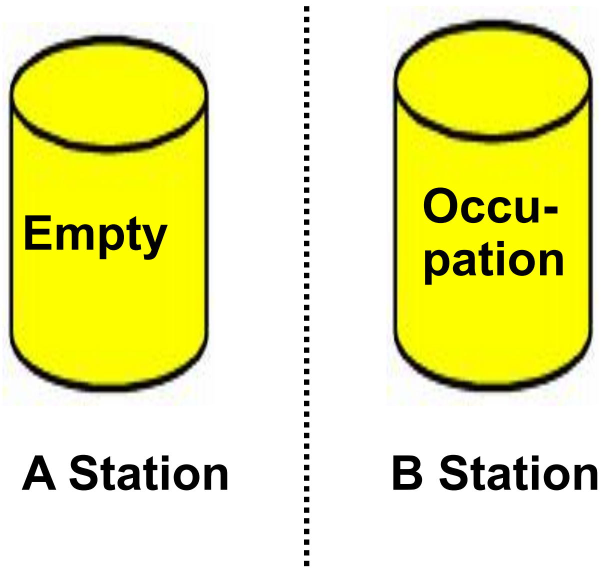
- Banking system,
- Crashing in 1 system is ok,
  - But difference system is also crash cos that bug is TERRIBLE. (outta world)



unmatched money

Hitachi railway database

Mirror disk of booking system



Empty

Occu-
pation

A Station

B Station

Seat C27, Train 0749

- Stop system and analyze for 6 hours and fix for 2 hours
  - Loss million$, cos stop system → no business
- Or else ?

When program ready for release?

- All test case are tested.
- The growth of bugs get flat.
- All the bugs are fixed.
- 48 hours continuous operation success fully complete

Independent QA engineer

- Around 8% of all engineers.
- They test (requirement spec, design doc, program, test case, etc.)
- A product CAN NOT be released without their “Go ahead” (So powerful)
- From customer view point, They design, Write and test the test cases.
  - (completely difference from the developer, not use the test cases from developer.)

The cost of the quality

- Assumption 12 months project with 10 developer on C with 100 KLOC

Phases

Requirement spec	:	2 Months
Designing	:	3 Months
Coding	:	2 Months
Debugging	:	3 Months
Testing (By QA)	:	2 Months

Testing phase (3M)

- 100 KLOC need 10K test cases (1K/Develpoer)
- 10 days to design 10K test cases
- 10 days to check 1K test cases in code inspection
- 40 days to check 10K test cases in machine testing

We built software for product

3 main concerns

- Quality → must be good
- Schedule → must be quick
- Cost → must be cheap

- Quality is the big problem to obtain.

Engineering VS Science

- Eng : built something that really worth to use
- Sci : Just possibility.

Example : Built a little gold from huge sea water

- Eng : Don’t worth enough → so much cost
- Sci : Great new way of technology

Student VS Professional Programming

Student program

- **No quality assurance/control**
- **Profitability** : 1 LOC = 50 – 100&

  1 person-month = 1KLOC

  student don’t care, is it worth enough, or how many day to built, just program for fun
- **Estimation** : How many days, money, people need? Cost, schedule, person/month

Student VS Professional Programming con't

- **Project management** : student don't care
- **Risk management** : such as heart control in hospital, The worst case, how to manage

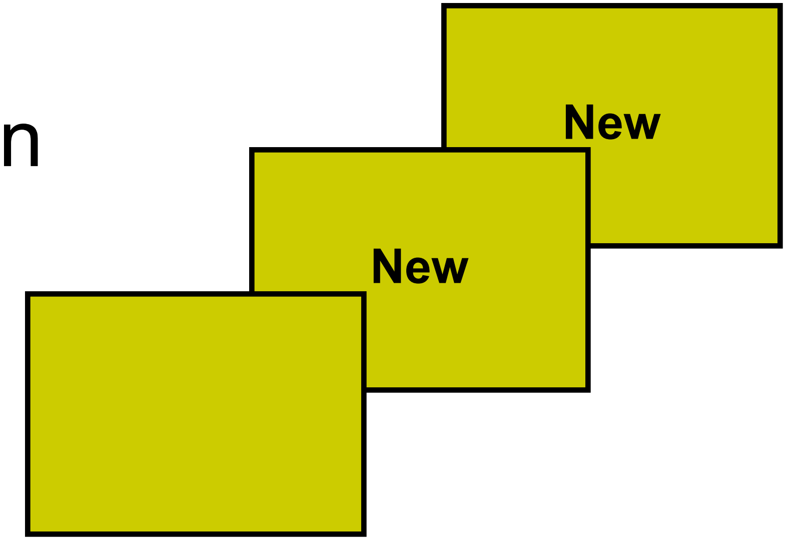
  or one day top of project manager quit, how to run project.
- **Error cases** : boundary, abnormal

Student VS Professional Programming con't

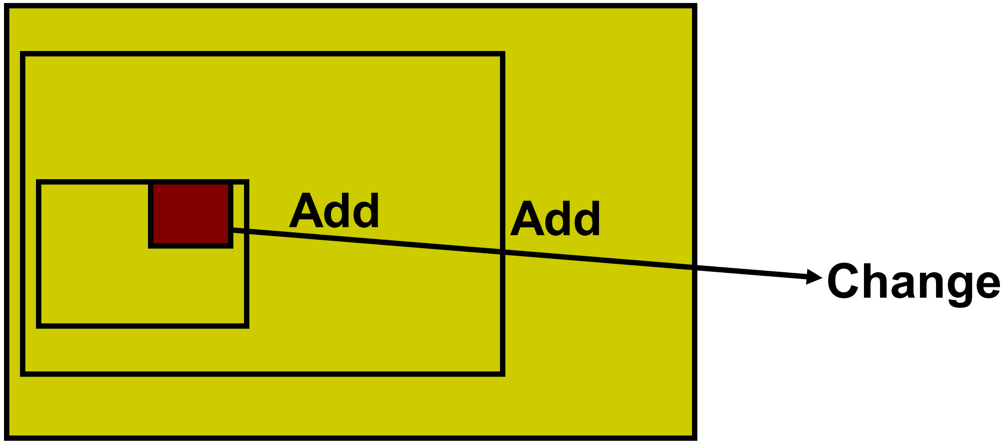
- **Reusability** : (make sure not much more modify, < 20%)
- **GUI** : This is real direct of user (non comp expert)
- **Documentation** : Student always jump to coding But professional do for reusable later, Adding, Modifying.

Reusability concept

- Old fashion



- New fashion : 1 time code, 100 modify



Resource usage

- 1 programmer = 1 KLOC
- 1 line = 50 – 100 $
- 1 programmer = 100,000$ (Japan - US)
- 10 – 20 programmers is ok
  - More than this project may be fail.
- 1 year project

Req.	Spec.	Design	Code	Debug	Test
2M	1.5M	1.5M	3M	2M	2M

Fact.

- Software engineering/ Embedded system
  - The most important is not coding (you can use any fresh graduate school)
  - But Requirement spec.
  - Specification
  - Design

Iceberg Tip

Outcome from user view point

Outcome actually

Source code

Req Spec Func spec Prj plan

General design Detail design

Interface spec Bug report

Bug analysis Test items

Schedule manage Test result

Operation manage Test result analysis

Code report

Popular models

- Waterfall model
  - Advantage : Finish phase and go on
  - Disadvantage : No outcome in early stage

    User always know what they want

    In prototype use (final stage)

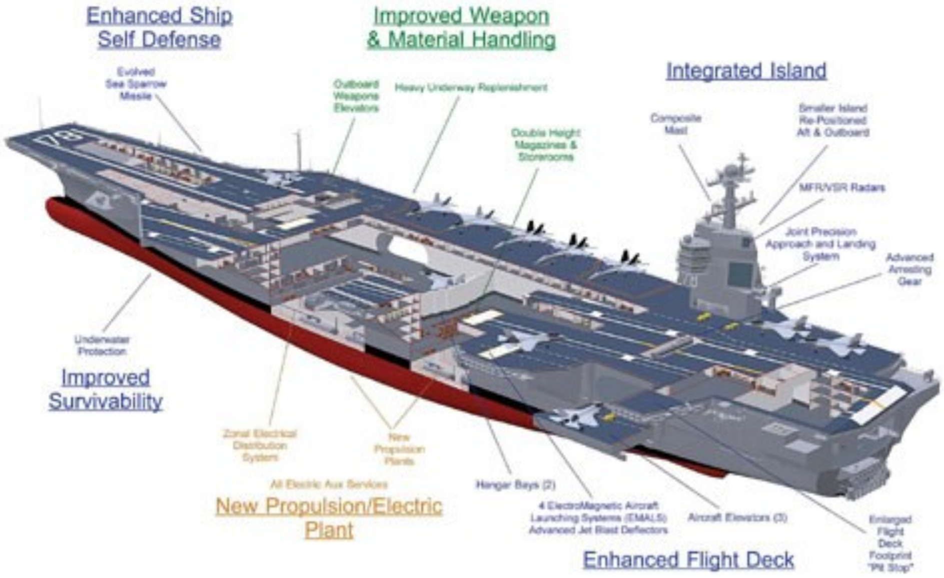
    If it not ok, So waste time
- Spiral model
  - Disadvantage : First version always make bugs, so hard to add function

Software Quality

- Software is the far more complex artificial thing that human ever built.
- 4 things make software very complex

4 complex attributes

1. Software is HUGE!


Enhanced Ship Self Defense
Evolved Sea Sparrow Missile
Improved Weapon & Material Handling
Outboard Weapons Elevators
Heavy Underway Replenishment
Double Height Magazines & Storerooms
Integrated Island
Composite Mast
Smaller Island Re-Positioned Aft & Outboard
MFR/VSR Radars
Joint Precision Approach and Landing System
Advanced Arresting Gear
Underwater Protection
Improved Survivability
Zonal Electrical Distribution System
New Propulsion Plants
All Electric Aux Services
New Propulsion/Electric Plant
Hangar Bays (2)
4 ElectroMagnetic Aircraft Launching Systems (EMALS) Advanced Jet Blast Deflectors
Aircraft Elevators (3)
Enhanced Flight Deck
Enlarged Flight Deck Footprint "Pit Stop"

Easy to built

So hard

Software is Huge!

VS

Software is Huge!

vs

How big is the software?

- Mobile phone → 10 MLOC
- Xerox multi function → 100 MLOC

- Example

1 line of page = 1 LOC, So 10MLOC is so big library

4 complex attributes con’t

2. Software is Invisible/intangible/in touchable

   Can’t feel the hard

   Non-programmer think it is easy to built

3. Software is easy to change

   If it hard to change such as building, Developer will pay more attention to build

   → But software is not, so the careless lead to many bugs.

4 complex attributes con’t

4. Software has too much Flexibility

   If you think ok, There are plenty of ways to code.

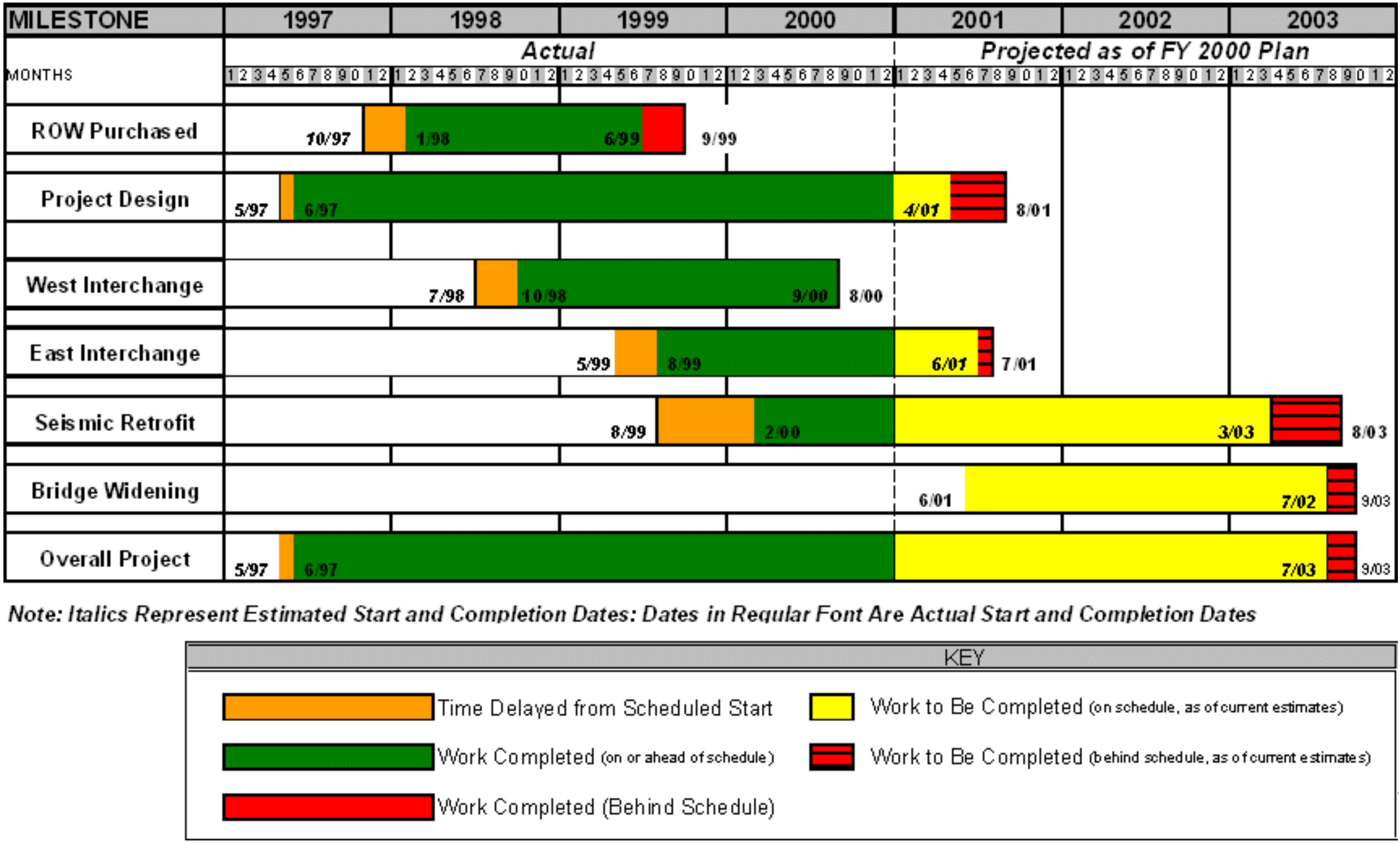
   No rules/regulation like Ohm’s law

Software Development Criteria's

- Schedule → Quicker
- Cost → Cheaper
- Quality → Better

Schedule

- Easy, Just plan and mark actual



Note: Italics Represent Estimated Start and Completion Dates: Dates in Regular Font Are Actual Start and Completion Dates

Cost

- Easy too, Just add people to do job and cost

  In software : Almost money go to salary of man power (a little for PC, Network)

Quality

- What is software quality? → hard to measure

  (It is difficult to tell as What's life/happiness?)

  A program without bugs → not good answer

  A program with 1KLOC should be bug free if you have 100 years to test, BUT you have to make it in 2 – 4 weeks, HOW CAN?

Software compare to real life

- In peaceful society
  - Arrest criminals → debugging
  - Avoidance → design for avoid bug

    (teach young to do good)

Software Quality

= Customer satisfaction? (partial true)

Customer criteria

Developer criteria

Depend on domain

- Beat maker
- Cell phone
- Xerox
- Games
- Watch
- Vending mach.
- TV
- Car

If you have to launch, cos customer request

So **Beta** release

Schedule delay

- If your plan delay, But customer said “We need it on that day, No delay!”

- How to manage?
  - Dead line is shifter
  - Reduce function

ISO 9126

1. Functionality → User require, Bugs
2. Reliability → Continue work after face bugs
3. Usability → Easy to use or not (GUI)
4. Efficiency → Performance, Resource usage, Network
5. Maintainability → How easy you can change, add func.?
6. Portability → How easy to change environment?

Exercise

What is a good restaurant?

In ISO 9126