Software's Chronic Crisis

– W. Wayt Gibbs –

4조

Overview

1. Software Chronic Crisis
2. Shifting Sands
3. Mayday Mayday
4. Proceed of Process
5. Mathematical Recreations
6. No Silver Bullet
7. Just add water

☐ **Denver's airport case**

- Twice the size of Manhattan
- 21 miles of steel track
- 4000 independent telecar
- 5000 electric eyes
- 400 radio receiver

☐ **Error in the software**

- 193-million$ BAE automated System
- red ink 1.1-million$ a day
- not predict when airport to open

☐ **rate of success**

- about 33% of projects are canceled
- overshooted shcedule by half
- 75% systems have operating failure

- Many code is handcrafted
- Very little interchangeability
- Maximum of craftsmanship

Need something…

☐ **Software Engineering**

- 1968 NATO Science Committee

- systematic, disciplined, quantifiable approach to the development, operation, maintenance of software

- most industry concern "interchangeable, reusable"

☐ **Break traditional programming**

-doubled code
-difficult to find error in real-time system
-must change assumptions

☐ **Distributed system**

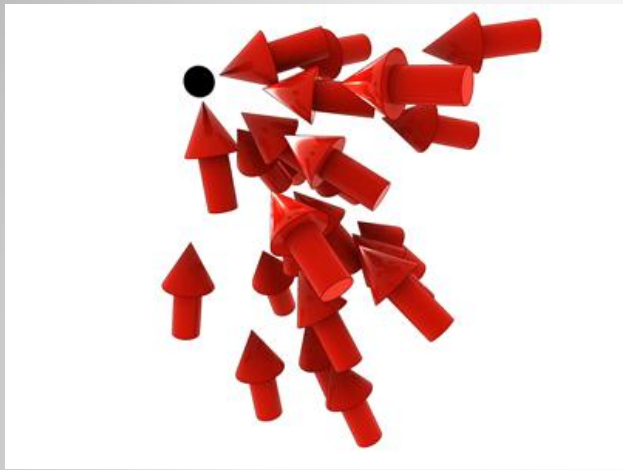
-run cooperatively on networked com

☐ **System integration**

-share data, user interface
-difficult to modify and repair

- **Dept of Motor Vehicle in California**

  - merging driver & motor registration
  - 6.5 times expected cost
  - they pulled plug remain investment
  - can’t build skyscrapers using carpenter



☐ **Complex System**

- if manager can't manage entire system, traditional process will break

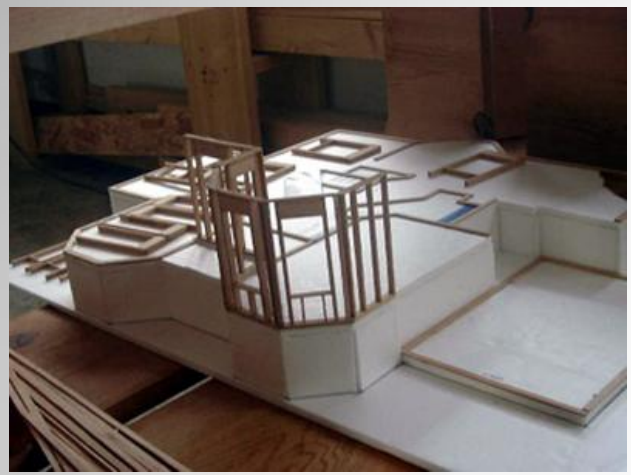
☐ **Be Engineering**

- how to measure consistently, quantitatively
- understanding densiry of errors and stagnation of productivity

☐ **Focus to Process**

- Emphasizing concentrate on process
- Grading ability of programming team

Not early bugs, but final can be devastrated



☐ **Mass Market Sortware**

- Release the faulty s/w as “beta”
- Tested by ‘volunteers’

☐ **Prototype**

- clear up misunderstanding between programmer and customer
- only can catch outer seeing bugs

☐ **Formal Method**

- rely on mathematical analysis to predict
- difficult to translate computer to mathmatical universe
- but ‘Formal Method’ can do

☐ **Clean room process**

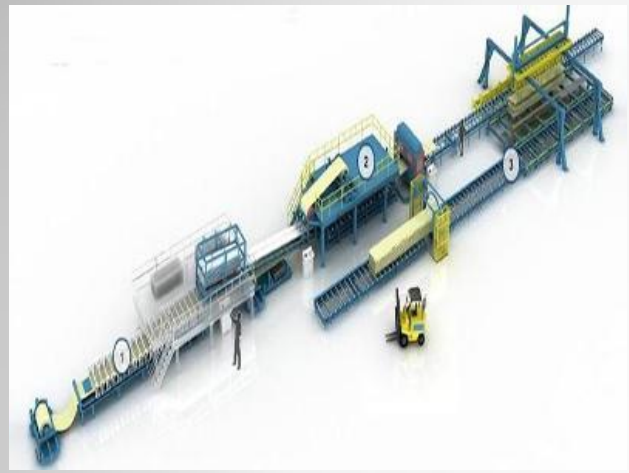
- "Safety” concerned
- Only quality proved function attach to system
- testing entire events in real world

☐ **To improve productivity**

- Object-Orient, CASE, 3th, 4th, 5th generation langauge

☐ **None knows productivity of S/W develop**

- Few programmer count their bugs
- No standards for measure
- Personal difference



☐ **Library**

- Reuse, no more rewrite.
- No standards.

☐ **Components**

- Assemble components to make software
- match any environment
- Give recipe for each components

☐ **Engineer do not spontaneously generate**

- educated in university
- trained out of habits developed by craftsmen

Q & A