SOFTWARE CRISIS SOLUTIONS?

Software Engineering Process

- Specification
- Design
- Development
- Verification
- Validation
- Management

Software Engineering Activities

- Software specification
  - Customers and engineers define the software and it's operational constraints
- Software development
  - Software is designed and programmed
- Software validation
  - Software (and specification) is checked against requirements
- Software evolution
  - Software modified to meet new customer and market requirements

Failures of process

- Specification
  - Missing functions specification
  - Ambiguity
- Attempts to improve
  - Formalization
  - Linking specification to tests Cucumber

Importance of test design in specification

- Requirements are tied to tests
- Every requirement will have linked set of tests
  - R1
    - T1 T2 T3 T4 T5
- Test parameters defined in Cucumber
- Tests generated in JUnit

What is software engineering? (Fred Brooks)

"…the fashioning of complex conceptual structures that compose the abstract software entity, and accidental tasks, the representation of these abstract entities in programming languages, and the mapping of these onto machine languages within space and speed constraints."

First published as: Brooks,F.P. (1986) "No Silver Bullets", Proceedings of the IFIP Tenth World Computing Conference, (ed.) H.-J. Kugler, pp 1069-79]

"…become a monster of missed schedules, blown budgets, flawed products …", in short "a werewolf" the solution to which is a "silver bullet" that " …makes software costs drop as rapidly as computer hardware costs …".

First published as: Brooks,F.P. (1986) "No Silver Bullets", Proceedings of the IFIP Tenth World Computing Conference, (ed.) H.-J. Kugler, pp 1069-79]

Software crisis

- We noted that software engineering is hard
- Why?
  - It must perform
  - It is boxed (time, money, size)
  - It is constrained by hardware, designs, use
  - It is obsolete very quickly
  - It is complex

Essential Difficulties

- Complexity
  - Because of size in terms of elements involved which may interact
  - An essential property, not accidental
  - e.g. natural language processing, image processing, legal systems
- Conformity
  - Interfaces are defined
  - Standards are imposed

Complexity

- Not
  - Measured in lines of code
  - Time to develop
- More to do with
  - Complexity of problem
    - Google self drive car
    - Any complex AI problem
  - Complexity of requirements
    - Redundancy
    - Data security

Essential Difficulties

- Changeability
  - Same product, many modifications
  - It's easy to request modifications
  - It needs to evolve
- Invisibility
  - Software is nebulous without geometry
  - Not visualisable

Computer Hardware

- Designed once, made many times
  - Economy of scale in design
- Simple goals
  - Increase instruction rate
  - Increase memory capacity
  - Improve reliability
- Performance not always with complexity increase
  - Multi core
  - Wider data paths
  - Increased clock rate

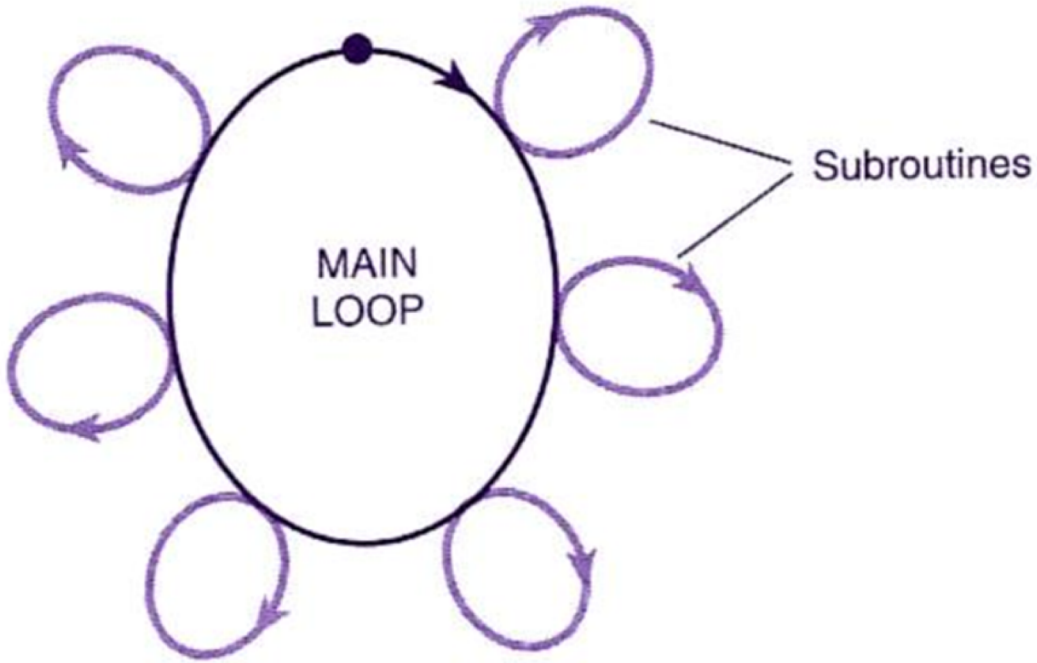
Important advances to 1986

- High level languages
  - Most important productivity development
  - Reduces accidental complexity
- Time sharing and development interactivity
  - Immediacy allows concentration
- Unified programming environments
  - e.g. Unix, provides a workbench and tools

Rules of thumb (in 1986)

- Exploit what exists (reuse)
- Use rapid prototyping for establishing software requirements
- Grow software organically, adding more functionality as they are run and tested
- Identify and develop the best conceptual designers of the new generation

Incremental organic growth



Silver Bullets?

- Better HLL ?
- Object Oriented programming ?
- Artificial intelligence

Silver Bullets?

- Expert systems
- “Automatic” programming
- Graphical programming

Silver Bullets?

- Program verification
- Environment and tools
- Workstations

No Silver Bullets

- Brooks concluded that in 1986:
  - there seemed to be no silver bullets
- 10 years later he reviewed the situation
  - still no silver bullets
- 20 years later
  - still no silver bullets

Conclusions

- Software engineering failures
  - Problem not solved
  - No mechanical answer
- Approaches
  - Based on best practice and experience
  - Still remains a skills based industry