

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

Application of AI to product specification

- Products such as ChatGPT
- Benefits...
 - Quickly generates a detailed list of specifications which are clearly written
 - Good at coverage of core requirements
- Disadvantages
 - It doesn't understand what its producing
 - Any innovations are just pirated from existing products

Example ChatGPT

- software specification for innovative bean to cup coffee machine

Innovative Bean-to-Cup Coffee Machine Software Specification

1. Intuitive User Interface:

- High-resolution touchscreen display with vibrant visuals.
- Intuitive icons and animations for easy navigation.

2. Coffee Selection:

- Wide variety of coffee options, including espresso, Americano, latte, cappuccino, and more.
- Customizable settings for coffee strength, milk froth level, and temperature.

3. Personalized Profiles:

- Multiple user profiles to save individual preferences.
- Store preferred coffee recipes, milk ratios, and grind preferences.

4. Coffee Bean Recognition:

- Built-in camera to recognize coffee bean bags.
- Automatically adjusts grinder settings based on the recognized bean type for optimal flavor extraction.

5. Brewing Time and Temperature Control:

- Fine-tune brewing time and temperature for each coffee type.
- Advanced temperature stability for consistent extraction.

6. Flavor Infusion:

- Option to infuse coffee with various flavors (vanilla, caramel, etc.).
- Adjustable flavor intensity for a personalized experience.

7. Smart Milk Frothing:

- Integrated milk frothing system with adjustable froth texture and temperature.
- Precise control over milk steaming for latte art enthusiasts.

8. Coffee Art Display:

- Visualization of coffee extraction process on the display.

software specification for innovative
bean to cup coffee machine with
payment handling

7. Payment Integration:

Integration with payment gateways
(credit/debit cards, mobile payments) for
secure transactions.

NFC payment capabilities for contactless
payments.

software specification for innovative bean to cup coffee machine with cash handling

- **7. Cash Handling Integration:**
 - Secure cash handling mechanism for accepting and storing banknotes.
 - Accurate note validation and acceptance for seamless payments.
- Missing
 - Internationalization (currencies)
 - Cash handling detail (change return)
 - Accounting

Using AI to generate code

- Good for boilerplate code but...
- Code will be often based on code from web, good or bad
- Try this with ChatGPT
 - write a class in Java using the open closed principle
- Produced a class which was open, but did not show how closure was implemented

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/Tesla 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

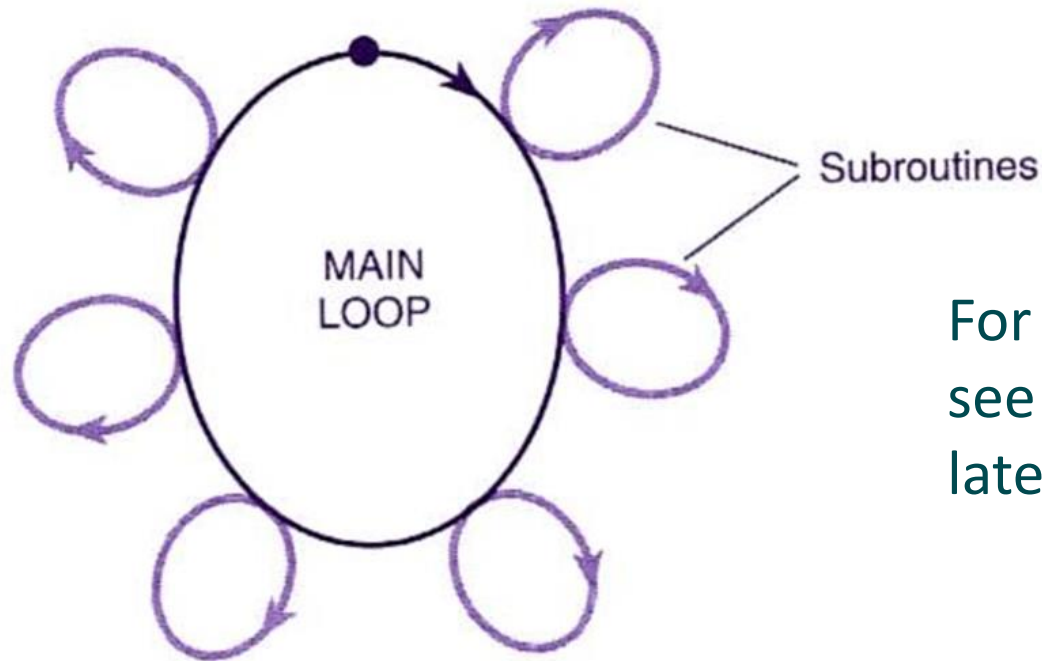
Important advances to 1986

- 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) (see OOD/OOP)
- 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



For OO sub-classes
see dependency inversion
later on

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
 - AI provides as many issues as answers (see Tesla self-drive)
- Approaches
 - Based on best practice and experience
 - Software still remains a skills-based industry