Fundamental Challenges

Cambiamento, Complessità, Difetti

Outline

- Principles and practices for dealing with fundamental challenges:
  - Requirements can change: apply the right **Process**!
  - Software is intrinsically complex: **Design** for easy maintenance!
  - Defects are inevitable: **Test** the software!
- These challenges have implications for what to build and how to build it

Requirements can Change

A user requirement describes a user need, want, or goal

A Scenario:

- Developers talk to customers and write user requirements
- Which they use to design and build a product
- But, the completed product fails to meet user expectations!

- Is it the process?

$$\begin{tikzcd}
 & \text{Customer Needs} \arrow[ld, bend right, "\textit{drive}"'] & \\
\text{Process} \arrow[rr, "\textit{build}"] & & \text{Product}
\end{tikzcd}$$

Software is Intrinsically Complex

Why?

- Let us say software is complex if it is hard to understand and modify
- Accidental (or Incidental) complexity is tied to an implementation
  - It can be removed by cleaning up the code, changing the language, etc.
- Intrinsic complexity remains even if all accidental complexity is removed
  - It can be managed by good designing that makes run-time behaviour more predictable

Intrinsic Complexity: Run-Time Behaviour is Invisible

- The static (compile time) source text is visible
  - –We can read and write code
- But, the dynamic (run-time) behaviour is invisible
  - –We have to imagine what the code does when it runs
- The primary source of intrinsic complexity
  - –The predictability (or not) of system behaviour from the source code
  - –In short, the invisibility of behaviour is the primary source

Dealing with Complexity

- Avoid Accidental (or Incidental) complexity
  - –Use the right languages, frameworks, and tools for the job
  - –Avoid technical debt, which accumulates, and makes code brittle
  - –Technical debt is the conceptual cost of cleaning up a "quick fix"
- Manage Intrinsic complexity
  - –**Design modular systems**, with relatively independent parts
  - –Low coupling, high cohesion
  - –Refactor to improve the design
  - –Apply well-known design patterns

Defects are Inevitable

Why?

- From George Miller's Experiments in the 1950s
  - –People can keep track of bout 7 units: bits, words, colors, tastes, tomes
  - –Beyond about 7 chunks of information, confusion and errors set in
- To err is human

Faults versus Failures

- Faults are in the static source code or in documents
  - –Defect, bug, and anomaly are synonyms for faults
- Failures are triggered at run time (dynamically)
  - –A failure is unexpected behavior, with crashes being dramatic failures
- Example: When an app crashes some of the time
  - –There must be a fault or faults in the app's code
  - –But, failures occur only some of the time; e.g., when there is a crash