Introduction to Software Engineering

Fritz Solms

February 2, 2015
Highly sought after discipline
- Software Engineer & Software Architect = top jobs in job surveys
- salary, job satisfaction, profession outlook, ...

However
- plagued by high failure rates
  - $\approx 50\%$ of projects over time/budget or abandoned
- Not acceptable for an engineering discipline.

Often complex projects requiring
- planning and synchronization across sub-projects, &
- communication and management across potentially large teams.
Origins of Software Engineering

- **1950’s/60s** ⇒ Software Engineering term used loosely
- **1968/9** ⇒ Formal creation of discipline
  - Bauer, NATO sponsored Software Engineering Conference
  - Early in period of perceived software crisis
    - 1000 developer projects fail on architecture (OS/360)
  - Advent of formal methods (slow growth over decades).
- **1980’s** ⇒ Professional discipline:
  - degree courses,
  - codes of conduct,
  - software engineering standards,
  - waterfall & other processes
- **2000’s** ⇒ resistance to engineering approach
  - Spawning of plethora of methodologies.
  - Many explicitly light weight.
- **2004** ⇒ SWEBOK
- Next??
What is Engineering

- Simplification of def from *Engineers’ Council for Professional Development*, (1947)
- Creative application of maths & science to
  - develop machines, . . . , or to
  - utilize them singly or in combination.
Characteristics/Requirements of Engineering

- Predictable outcome $\Rightarrow$ **Repeatability**
- Based on **theory** (mathematical models)
- Design **metrics** with defined construction **tolerances**
- **Failure tolerance**
- Component **reuse**
- **Separation of design** from implementation
- Controlled quality **quality attribute trade-offs**
Work Approach in Engineering

- **Professionalism**
  - code of conduct
  - legal accountability
  - Perhaps sub-conscious resistance to professionalism

- **Standards** and methods.

- **Division of work**
  - **Roles** with defined
    - responsibilities and
    - skills requirements

- **Quality assurance/Mathematical proof**
  - based on theory

- **Tool support**
  - for all aspects of the development and maintenance processes
What is Software Engineering?

- Application of engineering to software systems
- Systematic design & development of software products and management of software processes.
Quality Requirements for Software Engineering Processes

- cost,
- repeatability/predictability/estimatability,
- risk,
- performance (time-to-market),
- measurability/observability/auditability
- manageability
What is Quality Software?

Software which meets all requirements including

- functional requirements
  - (what the software must do)
- and non-functional requirements
  - reliability
  - scalability
  - performance
  - security
  - integrability
  - reusability
  - maintainability
  - flexibility
  - auditability/monitorability
  - ...
Responsibilities of Software Engineering (1/2)

- **Requirements engineering**
  - elicitation, analysis, specification, validation

- **Software design**
  - architecture design & application design

- **Software construction**
  - Coding, verification, unit testing, debugging, building, distribution for reuse, documentation

- **Quality assurance**
  - Unit testing, integration testing, quality testing, standards compliance testing, peer reviews

- **Software maintenance**
  - requirements change management, architecture evolution, application evolution

- **Software configuration management**
  - identification of configuration as a function of version/branch/time.
Responsibilities of Software Engineering (2/2)

- **Process management**
  - process definition, execution, management, measurement, reporting, improvement

- **Resource management**
  - manage skills & use and evolve skills effectively, estimation, ...

- **Risk management**
  - identify risks, remove risks, manage risks

- **Tools**
  - Development tools, testing tools, build tools, configuration management tools, installation tools, ...
Roles within a Software Engineering Process

- **Requirements specialist**
  - Functional & non-functional requirements
  - Technology-neutral process designs
  - Services and/or component contracts

- **Software architect**
  - Design, validation, documentation, enforcement of
    - software infrastructure addressing non-functional concerns
    - software infrastructure for integration
    - technologies, frameworks, high-level components, . . .

- **Software developer**
  - Low level technical design and mapping onto implementation technology/architecture

- **Quality assurance**
  - Enforce standards (documentation, code, tests, . . .)
  - Testing (functional incl unit & integration), non-functional

- **Project management**
  - estimation, planning, risk management, resource management, configuration management, reporting