“Beyond”

Software Engineering

Guest Lecture, University of Toronto

Homy Dayani-Fard, PhD
Consultant, Technology Strategy
IBM Business Consulting Services

Summary

- Software engineering is a new and fast growing field, which has grappled with its identity: from using the word engineering to definition of the term, to educational needs, to professional certification.
- A personal, somewhat historical perspective, on software engineering: from education, to practice, and beyond.

A short biography

- Consultant, Technology Strategy
- Quality advisor, DB2 UDB development
- Release analyst, DB2 UDB development
- Research officer, Centre for Advanced Studies
- Adjunct at University of Toronto, York University, and Queen’s University
- PhD, MSc from Queen’s University
- BSc, University of Toronto
- Service technician, Olivetti

Questions

- What is software?
- What is software engineering?
- What makes a software engineer?
- [What is engineering?]

Goal of software engineering

- To build software
  - Catches
    - Meets the specification
    - High quality
    - Cost and schedule control
  - $$$
- Software = program?
- Who are software engineers?

History

- 1968 NATO conference
  - Software crisis
  - Software engineering
  - Need for a formal discipline
- Holy grails
  - Automatic programming
  - Formal methods
  - Reuse
  - “Better” management

Automatic programming

- A system that “automatically” generates programs.
- If the system is “reliable”, so are its resulting programs
- Examples:
  - Compilers
  - 4GL
  - Application generators (e.g., Draco, KBEmacs, Programmer’s Apprentice)

Personal

- A new compiler was being developed that would radically change compilation. There was only once catch: converting make files to a standard configuration file.
- Result: Failed!

Formal methods

- Two camps:
  - Verification
    - Create formal specifications and demonstrate that the implementation is consistent with the specification
  - Refinement
    - Using mathematical techniques step-by-step refine the specification until it is “executable”
- Examples
  - Z, VDM, CSP,
  - Darlington, Paris Metro

Personal

- Developed a small size distributed real-time system. Developed formal specifications, formally “proved” that the implementation was consistent with its specification. A group of five reviewed and approved the implementation.
- Result: Failed!

Reuse

- Build software from components:
  - Like hardware design, put together IC’s
- Early success
  - Fortran, C libraries
- Challenges
  - Indexing and searching
  - Generality of code
  - Performance
  - NIH
  - Architectural mismatches

Personal: second hand

- A large development group set a goal of creating reusable modules. Developers had to contribute to a central repository. They also received bonus points if they used modules from the library.
- Result: Failed!

Reuse … continued

- Later success (or otherwise)
  - COM, DCOM, CORBA, RMI, Java class libraries
- Higher level reuse (and successes)
  - Architectural patterns, e.g. n-tier, pipeline
  - Design patterns, e.g. MVC, Command, Facade
  - Frameworks, e.g. Struts
- Future : Web services, SOAP, MDA

Management

- **Software life cycles**
  - Control
  - Traceability
  - Parallel development
  - Risk management
- **Examples:**
  - Waterfall (and variations), Iterative (and variations), Process oriented (RUP), people oriented (XP)
  - Configuration management

Management … continued

- Certification: showing off our abilities to customers (raise their level of confidence)
  - CMMI
  - SPICE, ISO 9000
  - Other mandated government agencies, e.g., FDA

Personal

- A model driven approach built on top of a commercial framework generating web services definitions.
  - Process modeling
  - Use case modeling
  - Object modeling
  - Design
  - XML generation
- Results: jury is out!

Personal

- A CIO of a financial institution asked us if he could receive the same level of benefits (ROI ~ 20-40%) by investing in maturity. In particular, going from level 2 to 3 on CMMI.
- Result: No!

Software engineering characterization

- Large
  - Number of people
  - Number of features
  - Number of dependencies
- *Soft*
  - Changing requirement
  - Changing environment
  - Changing people

Aside:
What is computer science?

- If I had to summarize the entire field of computing, it would be:
  - Building hierarchies of abstractions for solving [repetitive] problems

Software engineering characterization … continued

- Repetition
  - Problems solved will come back nastier
  - Number of features
  - Number of dependencies
- Mosaic
  - Art: creativity, vision
  - Scientific: fact-based, hypothesis driven
  - Engineering: control, repetition of success
  - Management: team work, communication, decision making

Final thought

- Objective of software engineering is to solve a problem.
- Size matters. Scalability is a must!
- Time goes on. History will repeat itself!

Final thought … continued

- Whatever software engineering is, it helps if you have, on top of all your technical and conceptual skills
  - Communication skills: influencing
  - Team work: negotiation, compromise
  - Vision: see beyond the technical solution

Thank you!

Questions?

Categorization of software

- Commercial shrink-wrap
  - Vertical vs. horizontal (middle-ware)
- Custom applications
- Government
- Safety critical
- Embedded