"Beyond" Software Engineering

Guest Lecture, University of Toronto

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

A short biography

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

Summary

- n 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.

Questions

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

Goal of software engineering

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

History

- n 1968 NATO conference
 - $_{\mbox{\tiny G}}$ Software crisis
 - $_{\mbox{\tiny q}}$ Software engineering
 - $_{\mbox{\tiny q}}$ Need for a formal discipline
- n Holy grails
 - $_{\mbox{\tiny q}}$ Automatic programming
 - $_{\mbox{\tiny g}}$ Formal methods
 - $_{\text{q}}$ Reuse
 - $_{\mbox{\scriptsize q}}$ "Better" management

Automatic programming

- A system that "automatically" generates programs.
- If the system is "reliable", so are its resulting programs
- n Examples:
 - $_{\mbox{\tiny G}}$ Compilers
 - g 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.

n Result: Failed!

Formal methods

- n Two camps:
 - $_{\mbox{\tiny q}}$ Verification
 - ⁿ Create formal specifications and demonstrate that the implementation is consistent with the specification
 - g Refinement
 - Using mathematical techniques step-by-step refine the specification until it is "executable"
- n Examples
 - $_{\mbox{\scriptsize q}}$ Z, VDM, CSP,
 - $_{\mbox{\scriptsize g}}$ 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.
- n Result: Failed!

Reuse

- n Build software from components:
 - $_{\mbox{\tiny \ensuremath{\mathbb{T}}}}$ Like hardware design, put together IC's
- n Early success
 - $_{\mbox{\tiny G}}$ Fortran, C libraries

n Challenges

- $_{\mbox{\tiny q}}$ Indexing and searching
- ${}_{\mbox{\tiny g}}$ Generality of code
- g Performance
- g NIH
- g 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.
- n Result: Failed!

Reuse

... continued

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

Management

n Software life cycles

- g Control
- g Traceability
- Parallel development
- g Risk management

n Examples:

- ^d Waterfall (and variations), Iterative (and variations), Process oriented (RUP), people oriented (XP)
- g Configuration management

Management

... continued

- Certification: showing off our abilities to customers (raise their level of confidence)
 - $_{\tt q}$ CMMI
 - g SPICE, ISO 9000
 - $_{\mbox{\tiny g}}$ Other mandated government agencies, e.g., FDA

Personal

- A model driven approach built on top of a commercial framework generating web services definitions.
 - $_{\mbox{\tiny q}}$ Process modeling
 - $_{\mbox{\tiny q}}$ Use case modeling
 - $_{\mbox{\tiny G}}$ Object modeling
 - $_{\rm q}$ Design
 - $_{\mbox{\scriptsize g}}$ XML generation
- n Results: jury is out!

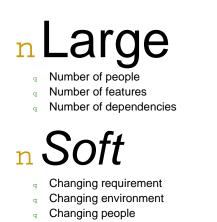
Personal

- 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.
- n Result: No!

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



Software engineering characterization ... continued

n Repetition

- g Problems solved will come back nastier
- n Number of features
- $_{\mbox{\tiny g}}$ $\,$ Number of dependencies

n Mosaic

- G Art: creativity, vision
- ^a Scientific: fact-based, hypothesis driven
- g Engineering: control, repetition of success
- Management: team work, communication, decision making

Final thought

- Dbjective of software engineering is to solve a problem.
- n Size matters. Scalability is a must!
- n 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
 - g Communication skills: influencing
 - $_{\mbox{\scriptsize g}}$ Team work: negotiation, compromise
 - $_{\mbox{\tiny g}}$ Vision: see beyond the technical solution

Thank you!

Questions?

Categorization of software

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