Brief History of SE

1. Intro
   1. Software is everywhere
      1. Buying, driving, washing
      2. Synonyms: programs, applications
   2. People, who develop the software
      1. SE, SD, Programmer
2. Difficulties
   1. Accidental difficulties
      1. Current/past/future technologies
      2. Sudden change of the OS, compilers, language, process
   2. Essential difficulties of software
      1. Studied by Fred Brooks
      2. Subset of essential (defining) properties
      3. No easy answer to essential difficulties
3. Essential difficulties
   1. Invisibility
   2. Complexity
   3. Changeability
   4. Conformity
      1. Large system
      2. More complexity
   5. Discontinuity
      1. People easily understand linear or semi-linear system
         1. Shower
      2. Software is discontinuous
         1. Small change on input can result in huge change of output
            1. Password
4. Software engineering
   1. Set of recommendations how to develop software
   2. A discipline with a considerable body of knowledge and considerable importance in both academia and industry
   3. First app appeared in 1950’s
      1. 50 more years history
5. Beginning of Software
   1. Separated from the hardware in 1950
      1. Emerged as a distinct technology
      2. Became independent product
   2. Original programmers recruited from the ranks of hardware engineers and mathematician
      1. Use ad-hoc technique from their former fields
6. Paradigm
   1. Thomas S. Kuhm
   2. Paradigm
      1. Coherent tradition of scientific research
      2. Current overused
7. Anomaly
   1. Anomaly is an important fact that directly contradicts the old paradigm
   2. Dilemma: disregard anomaly or change paradigm
   3. To change paradigm means to abandon large part of investment
   4. The anomaly must be compelling
   5. Change
      1. Discontinuity in the development of discipline (revolution)
         1. Kuhn collected extensive historical data on paradigm change
         2. Phlogiston 🡪 Oxygen in 1770
8. Resistance to paradigm change
   1. Advantages of the new paradigm is in dispute
      1. Attempts are made to extend old paradigm to accommodate anomalies
      2. Band-Aid approaches try to fix old paradigm
   2. Knowledge and investment accumulated up to that point may lose its significance
      1. Some knowledges may be completely lost (knowledge of color of the chemical compounds)
   3. Final victory of the new paradigm guaranteed by a generation change
   4. Unsuccessful attempts at paradigm change
9. Paradigm change of ~ 1970
   1. Anomaly
      1. Previous techniques did not scale up
      2. Brooks: “mythical Man-month”
      3. OS/360 by IBM
   2. Paradigm change established discipline of software engineering
      1. Dealt with complexity of software
      2. Software design established as an important consideration
      3. Introduced the waterfall metaphor
10. Waterfall metaphor (linear process)
    1. Used in construction and manufacturing
       1. Collect the requirements
       2. Create a design
       3. Follow the design during the entire construction
       4. Transfer finished product to the user
       5. Solve residual problems through maintenance
    2. Intuitively appealing metaphor
       1. Good design avoids the expensive late rework
       2. Waterfall became the dominant paradigm
    3. Model
       1. Requirement
       2. Design
       3. Implementation
       4. To the user
       5. Maintenance
    4. Paradigm
       1. Elaborate up-front activities
          1. BBDUF (big design up front)
          2. Value may turn out to be small
       2. Textbooks
          1. Still largely based on the waterfall
11. Anomaly of requirements volatility
    1. 30% or more requirements may change during development
       1. this is the direct result of the team’s learning process and software interoperability
    2. Caper-Jones: requirements for IT change at a rate 2-3% per month
12. Standish group anomaly
    1. In 1995
       1. …
    2. the waterfall metaphor did not solve the problems of software development
13. Band-Aid: anticipation of changes
    1. If changes can be anticipated at design time, they can be controlled by a parameterization, encapsulations, etc.
       1. Waterfall model still can be used
    2. Experience confirms
       1. Many changes are not anticipated by the original designer
       2. Inability to change software quickly and reliably means that business opportunities are lost
       3. Only a band-aid solution
14. Band-Aid prototyping
    1. Another band-aid attempt to extend waterfall
    2. Create a prototype to capture requirements
    3. Problem: volatility continues after porotype has been completed
15. Paradigm change of ~ 2000
    1. New life-span model emphasizes software evolution
       1. Staged model of software lifespan
       2. Based on data from long-lived system
    2. Iterative development
       1. Rational unified process
    3. Agile development
       1. SCRUM
       2. Extreme programming
16. All three paradigms currently coexist
    1. Ad hoc still used by some single programmers
    2. Waterfall works if there is no volatility
       1. Small or short-lived project
    3. New paradigm is the new mainstream