CS 390 – Lecture 1

- Course webpage: http://csserver.evansville.edu/~hwang/f08-courses/cs390.html
- Course textbook: Schach, Object-Oriented & Classical Software Engineering, 7/e

Course Structure

- Textbook, case studies, guest lectures, discussion
- Short reaction responses, some textbook problems
- 5-page formal reaction paper
- Software project – implementation documentation, presentation

Chapter 1: The Scope of Software Engineering – Historical Aspects

- 1969 NATO Conference
- Software crisis – quality unacceptably low and deadlines and budgets not met
- Use engineering to solve
- Still with us: Figure 1.1 – Outcomes of over 9000 projects completed in 2004

Cutter Consortium Data

- 2002 survey of information technology organizations
  - 78% have been involved in disputes ending in litigation
- For the organizations that entered into litigation:
  - In 67% of the disputes, the functionality of the information system as delivered did not meet up to the claims of the developers
  - In 56% of the disputes, the promised delivery date slipped several times
  - In 45% of the disputes, the defects were so severe that the information system was unusable

Economic Aspects

- Coding method CM<sub>new</sub> is 10% faster than currently used method CM<sub>old</sub>. Should it be used?
- Common sense answer
  - Of course!
- Software Engineering answer
  - Consider the cost of training
  - Consider the impact of introducing a new technology
  - Consider the effect of CM<sub>new</sub> on maintenance

Maintenance Aspects

- Life-cycle model
  - The steps (phases) to follow when building software
  - A theoretical description of what should be done
- Life cycle
  - The actual steps performed on a specific product
Classical Model Phases

- Requirements phase
  - Explore the concept
  - Elicit the client’s requirements
- Analysis (specification) phase
  - Analyze the client’s requirements
  - Draw up the specification document
  - Draw up the software project management plan
  - “What the product is supposed to do”

Classical Model Phases (2)

- Design phase
  - Architectural design, followed by
  - Detailed design
  - “How the product does it”
- Implementation phase
  - Coding
  - Unit testing
  - Integration
  - Acceptance testing

Classical Model Phases (3)

- Postdelivery maintenance
  - Corrective maintenance
  - Perfective maintenance
  - Adaptive maintenance
- Retirement

Classical Maintenance

- Defined in terms of the time at which the activity is performed
  - A fault is detected and corrected one day before installation => development
  - Same fault is detected and corrected one day after the software product was installed => (corrective) maintenance

Classical Maintenance (2)

- Another example
  - The client wants increase in functionality to be made just before installation ("moving target problem") => development
  - The client wants its functionality to be increased after installation => (perfective) maintenance
- Other issues
  - Building software from scratch is rare today
  - Reuse is widespread

Modern Maintenance

- Now usually defined operationally (rather than temporally)
  - The process that occurs when a software artifact is modified because of a problem or because of a need for improvement or adaptation
- Postdelivery maintenance happens after installation
Importance of Postdelivery Maintenance

- Bad software is discarded
- Good software is maintained – 10, 20 years or more
- Software is a model of reality, which is constantly changing

Time (= Cost) of Postdelivery Maintenance (Figure 1.3)

(a) Between 1976 and 1981
(b) Between 1992 and 1998

Consequence of Relative Costs of Phases

- Surprisingly, the costs of the classical phases have hardly changed
- Return to CM_{old} and CM_{new}
- Reducing the coding cost by 10% yields at most a 0.85% reduction in total costs
  - Consider the expenses and disruption incurred
- Reducing postdelivery maintenance cost by 10% yields a 7.5% reduction in overall costs