Software Maintenance and Evolution

Thomas LaToza

1/27/2005

What makes good research in software engineering?

Area Overview

- Traditional up front view
  - Anticipate design decisions changing at inception and act accordingly
  - Information hiding, interchangeable components, requirements analysis, architectural analysis, apply appropriate problem frames, ...
- Challenges for the up front architect
  - Successful systems endure
  - Office, Windows, SESS telephone switch, ...
  - Software context changes
  - Security, OS, databases, web services, PDAs, ...
  - User expectations change
  - Full text search anywhere, viewable from website, ...
  - Construction techniques change
  - OO, refactoring, design patterns, ADLs, program analysis, ...
- Central Problem
  - How to change existing software for requirements and technology that were unknown at inception


- Empirical study of OS/360 development over 12 years, 20 releases
- “Laws” of program evolution
  - Law of continuing change
    - A system that is used undergoes continuing change until it is judged more cost effective to freeze and recreate it.
  - Law of increasing entropy
    - The entropy of a system (its unstructuredness) increases with time, unless specific work is executed to maintain or reduce it.
  - Law of statistically smooth growth.
    - Growth trend measures of global system attributes may appear to be stochastic locally in time and space, but, statistically, they are cyclically self regulating, with well-defined long-range trends.

Results
Current Approaches

- Agile
- Refactoring
- Empirical studies
- Program comprehension
- Reverse Engineering
- Reengineering