Software Re-Engineering: Research Paradigm

Marwan Abi-Antoun

17-939A: What Makes Good Research in Software Engineering?
Professor Mary Shaw
Spring 2005
Software Re-Engineering as a Planned Activity

- **Why:**
  - Benefits
- **What**
  - Code, design (class or transition diagrams), architecture, …
- **When:**
  - During development or maintenance, as input to recoding, …
- **Who:**
  - Fully automated, semi-automated, manual
- **Where:**
  - Find the “bad smells”
- **How:**
  - Preserving certain properties, introducing new ones, …
- **How Much:**
Reengineering can improve maintainability by 20%

- Empirical studies or controlled experiments
  - Structural differences do impact maintainability
  - Restructuring programs to simply adhere to certain rules does not improve maintainability
  - Disclaimers: small sample; toy program (< 500 SLOC)

- Sources:
Current State of Affairs

- Not using real world examples
  - Toy programs < 1000 SLOC
  - Written by researchers or students (not legacy code!)
- Prototype tools without commercial promise
- “The whole field of software engineering is hobbled by a lack of empirical studies. The state of affairs in reverse engineering is even worse”
- Sources:
Current Research

- **Formalisms**
  - Program slicing
  - Graph transformations
  - **Software metrics** (including statistical techniques)
  - Concept analysis
  - Refinement techniques

- **Research questions:**
  - Which formalisms are most suited?
  - How do we preserve certain properties?
    • Input/outputs, behavior, non-functional quality attributes, ...
  - How do we introduce certain properties?
  - How do we guarantee consistency?
Software Metrics

• Before
  – Measure internal/external quality

• After
  – Measurement improvement of the quality

• Early work started that way
  – # of GOTO statements
  – Modularity measures

• More objective measure
  – eXtreme Programming: “Bad Smells”
  – “In our experience, no set of metrics rivals informed human intuition” (Fowler, M – Refactoring: improving the design of code, p. 75)
Design Metrics: an empirical analysis
How to check restructuring is correct?

• Meaning-preserving
  – Pre/Post-conditions, Invariants
    • Pre and Post-conditions logically equivalent
  – Dependencies
  – Heuristics
Are we talking about module view or runtime view?

Relax constraints

- Non-behavior preserving
  - Perturb local structure
  - But maintain overall consistency
References