Definitions (1)

- **Reverse Engineering**
  - process of analyzing a system to:
    - Identify the components and interrelationships
    - Create representations at a higher level of abstraction
  - Redocumentation – Creation or revision of a semantically equivalent representation within the same relative abstraction level
  - Design Recovery – reverse engineering where domain knowledge, fuzzy reasoning, and other external information is used

Definitions (2)

- **Restructuring** – transformation from one representation to another on the same semantic level without changing functionality of the system
- **Reengineering** – Examination and alteration of a system to recompose it in a new form and the process of realizing this new form
- **Program Comprehension** – Process of acquiring knowledge about a computer program

Relationships Between Terms

*Figure 1. Relationship between terms. Reverse engineering and related processes are transformations between or within abstraction levels, represented here in terms of life-cycle phases.*
What is Reverse Engineering Research?

- **Methods**
  - Visualization – Verified by implementation
  - Analyses
- **Models**
  - Cognitive – Verified by Empirical studies and Controlled experiments
    - Top-down (R. Brooks)
    - Bottom-up (Soloway)
    - Both (Pennington, Von Mayrhauser and Vans)
- **Tools**
  - Often "proof by existence"
  - Sometimes report from real world experience

Tools

- What does a Reverse Engineering tool have?
  - Parser-generator/s
  - Repository
    - Information Retrieval-based
    - Relational
    - Intermediate representation (ASTs, etc.)
  - Analysis tools/functionality
  - Visualization capabilities (UI)

Methods

- **Static Analysis** – e.g. effects analysis
- **Dynamic Analysis** – e.g. deadlock detection
- **Concept Recognition**
- **Pattern (Cliché) Recognition**
  - Programming language idioms
  - Domain-based
  - General programming idioms (patterns, stacks, etc.)
- **Graphs/Charts/Visualization**
  - Control flow, data flow, star diagrams, trees, UML diagrams, animations, etc.

Goals and Hindrances (1)

- Aid human ability to comprehend complex software systems in order to perform maintenance activities
  - Correction, enhancement, evolution
- Extract important information and deposit in permanent memory store for later use
  - Needs to be able to support multiple languages and types of information
- Facilitate all methods of learning about software
  - Ways of looking: browsing, searching
  - Method of understanding: top-down, bottom-up
Goals and Hindrances (2)

- Provide facilities for abstraction (and visualization)
  - Design vs. Specification
  - Traceability throughout the system
  - Means to translate from one result type to another
  - Managing level of detail
- Tool interoperability
  - This is one of the major issues in Reverse Engineering
  - Each tool has slightly different inputs and outputs
  - Research coalitions produce “tool suites” that contain all the tools they think are necessary for understanding

Fun quotes

- “If programmers would just understand code the way they ought to (i.e. the way tools work), the code comprehension problem would be solved”
  - Von Mayrhauser and Vans
- “The key to program understanding is deciding what to look for and what to ignore”
  - M. Shaw

Classifying the Fields

- Abstraction
  - Design Recovery
  - Redocumentation
  - Program Comprehension
  - Restructuring
  - Debugging

Change to the System

Maturation

- 1974: ADTs (Uskov)
- 1980: Program Slicing (M. Weiser)
- 1990: Design Recovery (Biggerstaff)
- 1992: IWPC
- 1993: WCRE
- FUJABA
- COPRO
- UML 1.0
- GUPRO
- PROUST
- MORALE
- MORALE
- OMT/Booch/etc.
- DESIRE
- GUPRO
- FUJABA
- Star Diagrams (Griswold)
- 2001
### Measuring Computer Program Quality and Comprehension (B. Schneiderman)

- Memorization/Recall measure for program comprehension (second experiment in the paper)
  - Still used widely today as "basis" measure
- Theory is that in order to memorize programs, semantic structures can be extracted and then there is less to memorize
- Shown by the "exact lines" versus "functionally correct" lines recalled
- Recall improves when subjects are given a task to perform with the code before they are to recall it

### Design Recovery for Maintenance and Reuse (T. Biggerstaff)

- Definition of "design recovery"
  - Recreates design from code, existing documentation, personal experience, and domain knowledge (which is what Schneiderman calls "semantic" knowledge)
  - Extract overall architecture
  - Group things together, recognize common tasks (performed by humans – relate things in conceptual abstractions)
- DESIRE system = impl. of domain model w/ idioms
  - Linguistic = syntactic
  - Semantic = relationship-based
- Domain model typifies expected relationships with built-in queries
  - Only partial mapping; partially human-directed
Tool Support for Planning the Restructuring of Data Abstractions ... (Griswold, et. al.)

- Using existing method for different purpose (star diagrams for planning restructuring and measuring/keeping track of progress)
- Hypothesize that too much overhead was needed to use existing technology (guaranteed transformations), simplify!
- Different teams used the technology in diverse (and unanticipated) ways
- Concept of elision
- Post-mortem design tradeoff analysis