

## Welcome to SENG 371 Software Evolution Spring 2013 A Core Course of the BEng Program

Hausi A. Müller, PhD PEng  
Professor, Department of Computer Science  
Associate Dean Research, Faculty of Engineering  
University of Victoria

## Announcements

- Marking
  - Midterm and A1 graded
  - Marks posted
  - **Today** office hours reserved for marking questions —1:30-2:30 ECS 660
- Course website
  - <http://www.engr.uvic.ca/~seng371>
  - **Lecture notes posted**
  - **Lab slides and activities are posted**
- Assignment 2
  - Due March 11 — revised
  - Reverse engineering and program understanding
    - Part I—Summarize three papers
    - Part II—Define terms
    - Part III—Reverse engineer a C program (gawk)
  - Cite your sources
  - Submit by e-mail to [seng371@uvic.ca](mailto:seng371@uvic.ca)

2

## Reading assignments

- Chikofsky, Cross: Reverse Engineering and Design Recovery: A Taxonomy, *IEEE Software* 7(1):13-17 (1990)  
[http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=43044](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=43044)
- Kienle, Müller: Rigi—An Environment for Software Reverse Engineering, Exploration, Visualization, and Redocumentation, *Science of Computer Programming* 75(4):247-263, Elsevier, Apr. 2010.  
<http://www.sciencedirect.com/science/article/pii/S016764230900149X>
- Müller, Jahnke, Smith, Storey, Tilley, Wong, Reverse Engineering: A Roadmap, in *The Future of Software Engineering, ICSE 2000 Millennium Celebration*, 2000.  
<http://dl.acm.org/citation.cfm?id=336526>

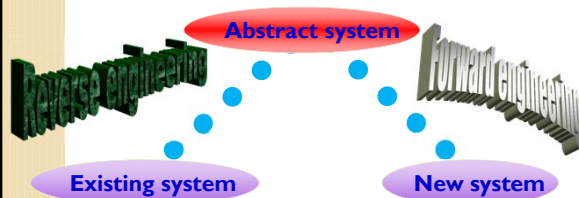
3

## Software reverse engineering

- **Def.** A two-step process
  - Information extraction
  - Information abstraction
- **Def.** A three-step process [Tilley95]
  - Information gathering
  - Knowledge organization
  - Information navigation, analysis, and presentation
- **Def.** Analyzing subject system [CC90]
  - to identify its current components and their dependencies
  - to extract and create system abstractions and design information
- The subject system is not altered; however, additional knowledge about the system is produced

4

## The Horseshoe Model of Software Migration

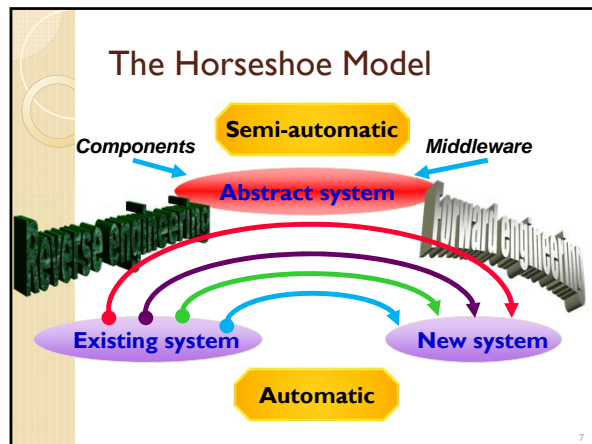


5

## Reengineering Categories

- Automatic restructuring
- Automatic transformation
- Semi-automatic transformation
- Design recovery and reimplementaion
- Code reverse engineering and forward engineering
- Data reverse engineering and schema migration
- Migration of legacy systems to modern platforms

6



### Reengineering Categories...

- Automatic restructuring
  - to obtain more readable source code
  - enforce coding standards
- Automatic transformation
  - to obtain better source code
  - HTML'izing of source code
  - simplify control flow (e.g., dead code, goto's)
  - refactoring and re-modularizing
  - Y2K remediation

### Reengineering Categories...

- Semi-automatic transformation
  - to obtain better engineered system (e.g., re-architect code and data)
  - semi-automatic construction of structural, functional, and behavioral abstractions
  - re-architecting or re-implementing the subject system from these abstractions

### Design Recovery Levels of Abstractions

- Application
  - Concepts, business rules, policies
- Function
  - Logical and functional specifications, non-functional requirements
- Structure
  - Data and control flow, dependency graphs
  - Structure and subsystem charts
  - [Software Architectures](#)
- Implementation
  - AST's, symbol tables, source text

### Synthesizing Concepts


- Build multiple hierarchical mental models
- Subsystems based on SE principles
  - classes, modules, directories, cohesion, data & control flows, slices
- Design and change patterns
- Business and technology models
- Function, system, and application architectures
- Common services and infrastructure

### How do you document software architecture?

- Documenting the relevant views one at a time and then adding information that applies to more than one view
- Modules or subsystems and how they compose or decompose into code units
- Processes and how they synchronize
- Programs and how they invoke each other or send data to each other
- Partition of system into work assignments
- How components and connectors work at run time

### Organizational axes Abstraction hierarchies

- Aggregation hierarchies
  - part-of relationships
- Generalization / specialization hierarchies
  - is-a relationships
  - inheritance
- Grouping
  - arbitrary
- Classification
  - category, instances
  - type, variables
  - class, objects

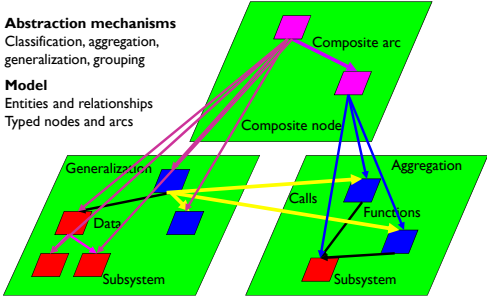


13

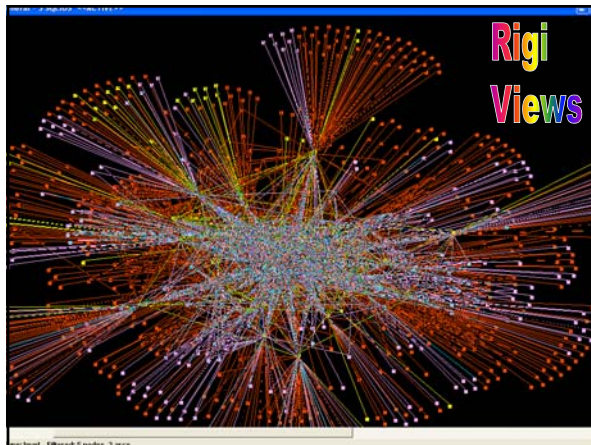
### The ubiquitous graph model

**Abstraction mechanisms**  
Classification, aggregation, generalization, grouping

**Model**  
Entities and relationships  
Typed nodes and arcs



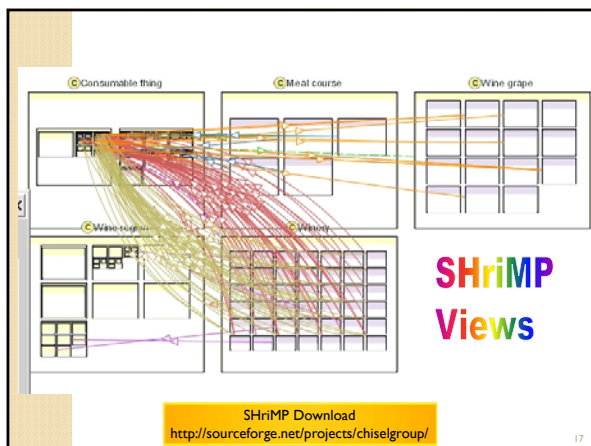
14



### Rigi System

- Website
  - <http://www.program-transformation.org/Transform/RigiSystem>
- Installation
  - <http://www.program-transformation.org/Transform/RigiInstall>
- Publications
  - <http://www.program-transformation.org/Transform/RigiPublications>

16



SHriMP Views


SHriMP Download  
<http://sourceforge.net/projects/chiselgroup/>

17

### How do you document software architecture?

- Box and arrow diagrams
- UML diagrams
- Class diagrams in Rational Rose

**Multiple views**



- Software architectures are complicated—typically too complicated to view all at once

18

## Views

- 2009—UML 2.2
  - <http://www.omg.org/spec/UML/2.2/>
  - Seven structural modeling diagrams
  - Seven behavioral modeling diagrams
- 2003—SEI Views
  - Documenting Software Architectures by Clemens, Bachmann, Bass, Garlan, Little, Nord, Stafford
  - [http://www.sei.cmu.edu/ata/C4ISR\\_03/C4ISR\\_03\\_1.htm](http://www.sei.cmu.edu/ata/C4ISR_03/C4ISR_03_1.htm)
- 2003—UML 2.0
- 2000—Siemens Views
  - Applied Software Architecture
  - by Hofmeister, Nord and Soni, Siemens
- 1997—UML 1.0
- 1995—Rational Views
  - Also referred to as 4+1 view model of software architecture
  - by Kruchten, Rational
- 1980 Software Cost Reduction (SCR) Method
  - by Parnas et al.
  - Module view, Uses view, Process View
- Programming languages as design notation

