

# *Reverse Engineering A Roadmap*

*Hausi A. Müller*

*Jens Jahnke*

*Dennis Smith*

*Peggy Storey*

*Scott Tilley*

*Kenny Wong*

*ICSE 2000 FoSE Track  
Limerick, Ireland, June 7, 2000*

1

## *Outline*

- *Brief history*
- *Code reverse engineering*
- *Data reverse engineering*
- *Reverse engineering tools*
- *Evaluating reverse engineering tools*
- *Key research pointers*
- *Conclusions*

## The Early Days

### ■ 1980's

*There will always be old software.*

- *Laws of Software Evolution [Lehman74,78,80]*
- *Fundamental strategies for program comprehension [Brooks83, Letovsky86, Pennington87]*
- *Taxonomy for reverse engineering [Chikofsky&Cross90]*
- *Up to 90% of software evolution involves program understanding [Standish84]*
- *Long-term software evolution is as important as software construction—in fact, probably more important today from an economic point of view*

ICSE 2000 Roadmap

3

## An Emerging Discipline

### ■ 1990's

- *Research challenges [IEEE Software90]*
- *Infrastructures and tools for the three major components of a reverse engineering system*
- *WCRE—Working Conf. on Reverse Engineering*
- *IWPC—Int. Workshop on Program Comprehension*
- *FEAST—Int. Workshop on Feedback and Evolution in Software and Business Processes*
- *WES —Int. Workshop on Web Site Evolution*
- *Ph.D. Theses*

*Discover abstractions in large software systems and transfer this understanding into the minds of software engineers for maintenance, evolution, and reengineering purposes.*

4

## Recent Reverse Engineering PhD Theses

- *Domain retargetable reverse eng. [Tilley95]*
- *Cognitive design elements for software exploration tools [Storey98]*
- *Management of uncertainty and inconsistency in database reengineering [Jahnke99]*
- *Continuous understanding—Reverse Engineering Notebook [Wong99]*
- *Integrating static and dynamic reverse engineering models [Systa2000]*
- *Architectural component detection for program understanding [Koschke2000]*

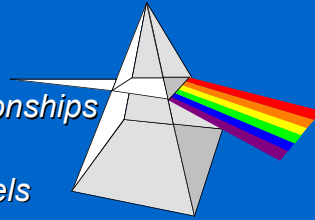


## Reverse Engineering Strategies

- **1990's**
  - *Reverse engineering strategies for specific program understanding and reengineering scenarios*
    - *Synthesizing concepts*
    - *Composing subsystem structures*
    - *Design, program, change pattern matching*
    - *Program slicing and dicing*
    - *Analysis of static and dynamic dependencies*
    - *Software exploration and visualization*
  - *Industrial-strength reverse engineering and transformation tools*

## Reverse Engineering Tasks

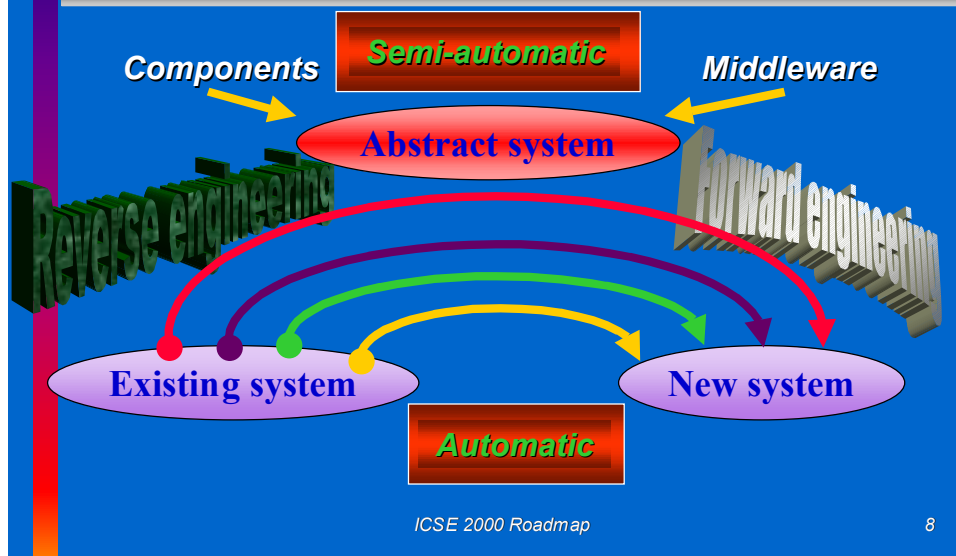
- **Information extraction**
  - Artifact gathering from many sources, knowledge organization, analysis
- **Information abstraction**
  - Aggregate components, relationships
  - Synthesize abstractions
  - Build hierarchical mental models
  - Subject system is not altered; additional knowledge about the system is produced
- **Every feedback loop in life cycle models is opportunity for reverse engineering**



ICSE 2000 Roadmap

7

## The Horseshoe Model



ICSE 2000 Roadmap

8

## ***Levels of Abstractions***

- ***Application***
  - *Application concepts, business rules, policies*
- ***Function***
  - *Logical and functional specifications, non-functional requirements*
- ***Structure***
  - *Data and control flow, dependency graphs*
  - *Structure and subsystem charts*
  - *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***

## ***Data Reverse Engineering***

- *Reverse engineering community has mostly concentrated on code reverse engineering*
- *Data are often more important than the code*
- *Data reverse engineering will be a key research area for the next decade*
- *Workshop on Data Reverse Engineering*

## ***Data Reverse Engineering Process***

- *Data analysis*
  - *Dealing with imperfect knowledge*
  - *Customizability*
- *Conceptual abstraction*
  - *Iteration*
  - *Bidirectional mapping process*
- *Many lessons from code reverse engineering directly apply*

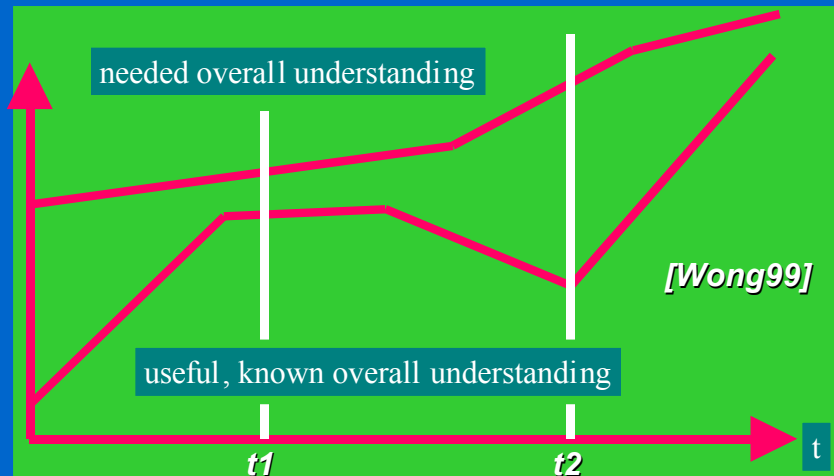
## ***Evaluating Reverse Engineering Tools***

- *The purpose of most reverse engineering tools is to increase the understanding an engineer has of the subject system*
- *No agreed-upon definition or test of understanding*
- *But several types of empirical studies that are appropriate for studying the benefits of reverse engineering tools*

## ***The Big-Bang Reverse Engineering Problem***

- *What can we do during evolution to ease future understanding and migration of information systems?*
- *We know the kind of knowledge we need but it is difficult to obtain from scratch*
- *“Big-bang” reverse engineering when the system becomes “critical” is high-risk*
- *Analysis paralysis*

## The Understanding Gap



ICSE 2000 Roadmap

15

## Continuous Reverse Engineering

- Apply program understanding continuously and incrementally during evolution of the software system
- Use software reverse engineering to re-document existing software
- Insert reverse engineering techniques into development [Wong99]
- Symbiosis: models and code [Jackson00]

ICSE 2000 Roadmap

16

## ***Key Research Pointers***

- *Investigate infrastructure, methods, and tools for continuous program understanding to support the entire evolution of a software system from the early design stages to the long-term legacy stages*
  - *Reverse engineering notebook*

## ***Key Research Pointers ...***

- *Instrument design architecture to ease extraction of understanding architecture*
- *Store architecture artifacts in schema-based repository and as unstructured or Web-based text to ease searching*
- *Allow for incomplete semantics and partial extraction of artifacts*

## ***Key Research Pointers ...***

- *Allow user to build virtual, multiple architectures, perspectives, and views*
- *Provide tools to compare virtual and code-centric architectures (e.g., reflection models [Murphy98])*
- *Make architecture extraction tools end-user programmable and extensible*

## ***Key Research Pointers ...***

- *Teach reverse engineering, program understanding, and software analysis in CS, CE, and SE programs*
  - *Carefully balance software analysis and software construction efforts in both research and education*
  - *Restructure curricula to teach both fresh creation and evolutionary change*
  - *Teach reverse engineering and program understanding methods and tools in 1st year*

## ***Key Research Pointers ...***

- *Develop methods and technology for computer-aided data and database reverse engineering*
  - *Integrate code and data reverse engineering methods and tools*
  - *Leverage synergy between code and data reverse engineering communities*

## ***Key Research Pointers ...***

- *Develop tools that provide better support for human reasoning in an incremental and evolutionary reverse engineering process that can be customized to different application contexts*
  - *End-user programmable tools*
  - *Domain retargetable reverse engineering*

## ***Key Research Pointers ...***

- *Concentrate on the tool adoption problem by improving the usability and end-user programmability of reverse engineering tools to ease their integration into actual development processes*
  - *Start with a web-based user interface*
  - *Conduct user studies*

## ***Conclusions***

- *Mission statement*
  - *Researchers in software design and formal methods should concentrate on software evolution rather than construction*
  - *Program understanding and analysis experts should teach their methods in 1st-year*
- *Plenty of research problems*
- *Wonderful case studies*
- *Exciting research!!!!*

# *An Invitation*

