

# ELEC 876: Software Reengineering

**Dr. Ying Zou**

Department of Electrical & Computer Engineering  
Queen's University



“Software is a ***mathematical product***; mathematics doesn't decay with time. If the theorem was correct 200 years ago, it will be correct tomorrow. If a ***program*** is correct today, it will be correct 100 years from now. If it is wrong 100 years from now, it must have been wrong when it was written. It makes no sense to talk about ***software aging***.”

## Software Aging

- Software aging occurs when
  - Existing software no longer meets the owners needs, or
  - Changes made to an existing system make it even hard to maintain
- Causes of software aging
  - Lack of movement
    - Developer fails to modify it to meet change requirements
    - Work well in the past, but no body uses it today
  - Ignorant surgery
    - Changes made by developers who do not understand the system
    - Difficult to understand the system after many changes

## Costs of Software Aging

- Owners of aging software found it increasingly hard to keep up with the market and lose customers to newer products
- Aging software often degrades in its space/time performance as a result of a gradually deteriorating structure
- Aging software often becomes “buggy” because of errors introduced when changes are made

***We can't prevent aging, but we can understand its causes, take steps limits its effects, temporarily reverse some of the damage it has caused, and prepare for the day when the software is no longer viable***  
***-- David Parnas, ICSE 1994***

## Course Objective and Focus

- Examine techniques that allow for a smooth transformation of older systems to ones that use more appropriate and robust advanced technologies
- Study the techniques for program analysis and understanding
- Gain hands-on experience with several state-of-the-art tools (e.g., Rigi, Shrimp, Swagkits, Bunch)

## Marking Schema

- The final mark will be a composition of
  - Midterm (10%)
  - Project Proposal (10%)
  - Project Progress Report (20%)
  - Final Report (50%)
  - Class participation (10%)

## Course Outline

- This course is broken into three major parts:
  - Part I: software reengineering terminology and processes
  - Part II: program analysis techniques
  - Part III: state-of-the-art research in software reengineering

## Course Schedule

- **Topic 1** — Reengineering: The role of software maintenance in a product's life cycle. An overview of software reengineering. Basic process models and real life examples
- **Topic 2**— Program analysis: Provide a review of the state of the art in Program Representation schemes and techniques for program analysis

## Course Schedule (con't)

- **Topic 3** — Software reengineering economics: Evaluate risks and costs involved software reengineering projects
- **Topic 4** — Software metrics & software quality: Provide an overview of software metrics used in reengineering and maintenance tasks and introduce the techniques to improve software quality
- **Topic 5** — Software design patterns, antipatterns and refactoring

## Course Schedule (con't)

- **Topic 6** — Software architecture & styles  
software architecture styles, software integration and interoperability
- **Topic 7** — Business process reengineering:  
Make software to fit with the organization, and provide the automated software to reflect the rethinking of business processes

## References

- “Software reengineering” Robert S. Arnold, IEEE Computer Society Press, 1993
- “Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices”, Robert C. Seacord Daniel Plakosh, Grace A. Lewis, ISBN: 0-321-11884-7, Addison Wesley Professional, 2003
- Advanced Compiler Design and Implementation, Steven Muchnick, Morgan Kaufmann, 1997