Abstract

- Focuses on reverse engineering strategies for software migration.
- Reviews the state-of-the-art in reverse engineering techniques.

Introduction

- Need for maintaining and improving existing software has risen dramatically in the last decade.
- Has been recognized a critical problem.
- Migration/reengineering involves: capturing, preserving and extending knowledge about software, analyzing and understanding software, and change, improving and evolving software.

Software Evolution

- Evolution is a natural phenomenon for systems, since requirements change with time.
- Managing long-term software evolution is critical
- Knowledge embedded in these systems constitutes corporate assets totaling billions of dollars.
- Mitigate by acquiring expertise in software reengineering and invest in migration technologies.

Reengineering Strategies

- Software reengineering: the process of examining and altering a subject system to reconstitute it in a new form.
- Includes: redocumentation, restructuring of source code, transformation of source code, abstraction recovery, and reimplemention.

Migration Strategies

- Migration difficulties: scale, incomplete specifications, inconsistent documentation, aging implementation technologies.
- Minimization of risk is key. Incremental approach helps.
- See Migrating to Object Technologies (I. Graham) for techniques than can help.
- Reverse Engineering Strategies.
Reverse Engineering: the process of generating new information about software such as synthesizing abstractions and producing different views.

Tools: Analysis tools, program understanding environments and reverse engineering systems.

The Year 2000 Problem

- Gartner Group estimated total cost of $400-600 Billion, with about 10% of existing code needing changes.

Conclusions

- It's critical we deal effectively with software evolution.