Software Understanding

W. Lewis Johnson
USC / Information Sciences Institute

Mar. 12, 1996

more information available at:
http://www.isi.edu/isd/I-DOC/I-doc.html

Software Understanding

- Techniques for helping people to understand software
- USC’s investigations:
  - Empirical studies of the software understanding process
  - Question-answering tool for explaining software: I-Doc
Intelligent Documentation (I-Doc)

- Generate software documentation on demand
- Focus presentation to user task, expertise

Problems with Documentation

- Focus is on amassing information
  - relevant information is hard to find
- Tends to fall out of date
- Needs of particular readers, e.g., maintainers, not satisfied well
Task-Dependent Content

- Information required depends on reader's activity
- Examples:
  - Designers want hierarchical decomposition, descriptions of system's environment and domain
  - Application programmers want to know external functions, inputs, outputs
  - Maintainers want:
    - Explanation of observed system behavior, if debugging
    - Explanation of machine/OS dependencies, if porting
  - Users want to know how to perform specific tasks
- No single document meets all needs

Dependencies on Expertise

- Presentation content depends on reader's familiarity:
  - Novices want function, experts want structure (Paris 1988)
  - Component classes vs. component instances
Empirical Investigations

- Studied inquiry episodes on newsgroup: Tcl users
- Provides examples of queries from software installers, application programmers
- Shows some biases
- Questions investigated:
  - How do user task influence the information provided?
  - How can the desired information be categorized?
  - How easily is the required information obtainable from conventional sources (e.g., code, requirements docs, etc.)

Task Classifications

- Tasks can be classified at two levels:
  - Global task (install, use in application, etc.)
    - defines relationship to system as a whole
  - Local task or goal (e.g., add item to menu)
    - defines relationship to a system component
- Types of global tasks:
  - Installer
  - User
  - System Integrator (with programs, networks)
  - GUI developer
  - Other programmers, designers
Question Classifications

- Goal Oriented
  - Procedural - how do I do x?
  - Goal achievement - can I do x?
- Problem Oriented - why did x happen?
- System Oriented
  - Motivational - why does x behave as it does?
  - Conceptual - What is x?
  - Explanatory - How does x work?

Distribution of Questions
Variance with Expertise

I-Doc Approach

- Build repository of code, semi-structured annotations
- Select information dynamically
- Compose presentation dynamically
- Generate NL fragments as needed
- Use hypertext as interface metaphor
- Advantages:
  - Repository can be built incrementally
  - Annotations are propagated and reused
Architecture Components

- Mosaic hypertext viewer
- Document server (HTTPD)
  - programmed to generate pages as needed
- Software repository (REFINE)
  - enhanced to respond to queries from document server

Architecture Diagram
Example Test Case

- I-Doc is being used to document the Advanced Multi-Purpose Support Environment (AMPSE)
  - Delivered to Wright Patterson AFB
  - Implemented in Ada

- Includes:
  - Software Design Document
  - Commented Ada code

- I-Doc currently provides:
  - dynamic documentation in place of commented code
  - Links into design document