Lecture 13: Documentation

CS190: Software System Design

April 3, 2002
Steven P. Reiss

I. Today's Class

A. Documentation

1. **What is documentation?**
2. **How can I talk for 50 minutes on documenation?**
3. **Is documentation important and why?**
4. **How does documentation fit into XP?**

II. What is documentation

A. Description of how to use the system

1. **User's manual for the overall system**
   a) Reference manual
   b) Tutorial
   c) Descriptive manual
   d) Different types of users might have different manuals
      (1) Database user vs. database administrator
2. **Programmer's manual for libraries**
3. **Problem reference manual**
4. **FAQ detailing primary questions users might have**

B. Description of the system itself

1. **Requirements documentation**
2. **Specifications documentation**
3. **Design documentation**
4. **Detailed design documentation (pseudocode)**
5. **Technical reports & papers describing the ideas/methods behind the system**

C. Internal documentation

1. **Javadoc style comments describing methods and classes**

2. **In line comments describing the code and data**
3. **Test cases**
4. **The code itself**

D. Online Documentation

1. **Help files**
2. **Balloon help, tool tips**
3. **Microsoft assistant**

III. Is Documentation Important

A. What documentation do you read

1. **Programmer documentation (java descriptions, C++ reference manual, ...)**
   a) As needed or in toto
2. **User's manuals, FAQs, quick start manuals**
3. **Online help**
4. **Design documents, etc.**
5. **External papers**

B. Better questions

1. **What don't you read**
2. **Why don't you read it**
   a) Because you can figure it out quicker directly
   b) Because it is inaccurate
   c) Because it is not needed
   d) Because it is irrelevant
   e) Because it is unreadable
   f) Because its model differs from the users
      (1) Different vocabulary
      (2) Different notion of what the system does

C. Documentation is important if it is needed

1. **But then it should be done right**
2. **If it is not needed, why bother doing it**

IV. Doing documentation right

A. User Documentation

1. Know the audience and the purpose

a) Target these specifically

b) Most documentation is used for answering questions

c) Design the documentation accordingly

2. Step back and come in slowly

B. Program Documentation

1. Know the audience and the purpose

a) Is it for a newcomer to the project

b) Is it for an experienced programmer

c) Is it for an overview or specific questions

2. Understand what questions will be asked

C. On-line Documentation

1. Tool-tips

a) Prevalent today, quite useful and supported

b) How to put useful information in a tool tip

c) How much information goes in a tool tip

2. Help

a) When is online help useful

(1) When it answers questions

(2) When it is accurate and complete

b) What should be in on-line help

(1) User manual

(2) Reference manual

(3) Question-oriented

c) Indices and tables of contents

3. Note help systems abound (Java help, for example)

D. Documentation guidelines

1. Documentation should be available

2. Documentation should be complete

3. **Documentation should be accessible**
4. **Documentation should be valid**
5. **Documentation should match its audience and purpose**

E. If you are going to do documentation

1. **Ensure that it is complete**
2. **Ensure you keep it up to date**
3. **Ensure that it is readable (not translated Chinese)**

V. Avoiding Documentation

A. XP

1. **Little emphasis on documentation**
2. **No emphasis on design or program documentation**
3. **The code is the documentation**

B. Non-XP

1. **Most programmers view the code as the documentation**
   a) Why -- accurate, up-to-date, ...

C. Combining documentation and code

1. **Goal -- keep the documentation accurate & accessible**
   a) Documentation should be with the code
   b) Then if one is changed, both will be
   c) Also, its right where the programmer needs it to understand the code
2. **Many ways of doing this**
   a) Comments
   b) Literate programming
   c) Javadoc
3. **Where to put high-level documentation**
   a) How to use multiple classes together
   b) How a package is to be used
   c) Technical notes and FAQs should be maintained
   d) Library of questions asked and answers

D. The code is the documentation

1. **Assembly language comments**
2. **Programmers understand code**
3. **However code must be readable**
   a) Code should be obvious
   b) Code should be simple
   c) Code should be well formatted, etc.
   d) Code should be readable
4. **Write code as if you were writing a paper**
   a) For others to read
   b) For you to be proud of
   c) For posterity

E. Make the program self-evident

1. **User shouldn't need a manual to figure it out**
2. **What is self-evident to the programmer is not necessarily self-evident to the user**
3. **Program use should be intuitive**
   a) Follow established guidelines and conventions
   b) Actions (drag/drop, select, draw line, ...) should do what the user expects
4. **Program should be forgiving**
   a) This encourages/allows experimentation
   b) Everything is undoable (or warned if not)
   c) Can't get into a bad state
5. **User experimentation (not the programmer) helps**
   a) Get the customer to help
   b) Find target users and ask them to try things
   c) Controlled vs. uncontrolled experiments
   d) Track problems, comments, etc.
6. **Built-in tutorials, help, etc. should be directed and lead the user**