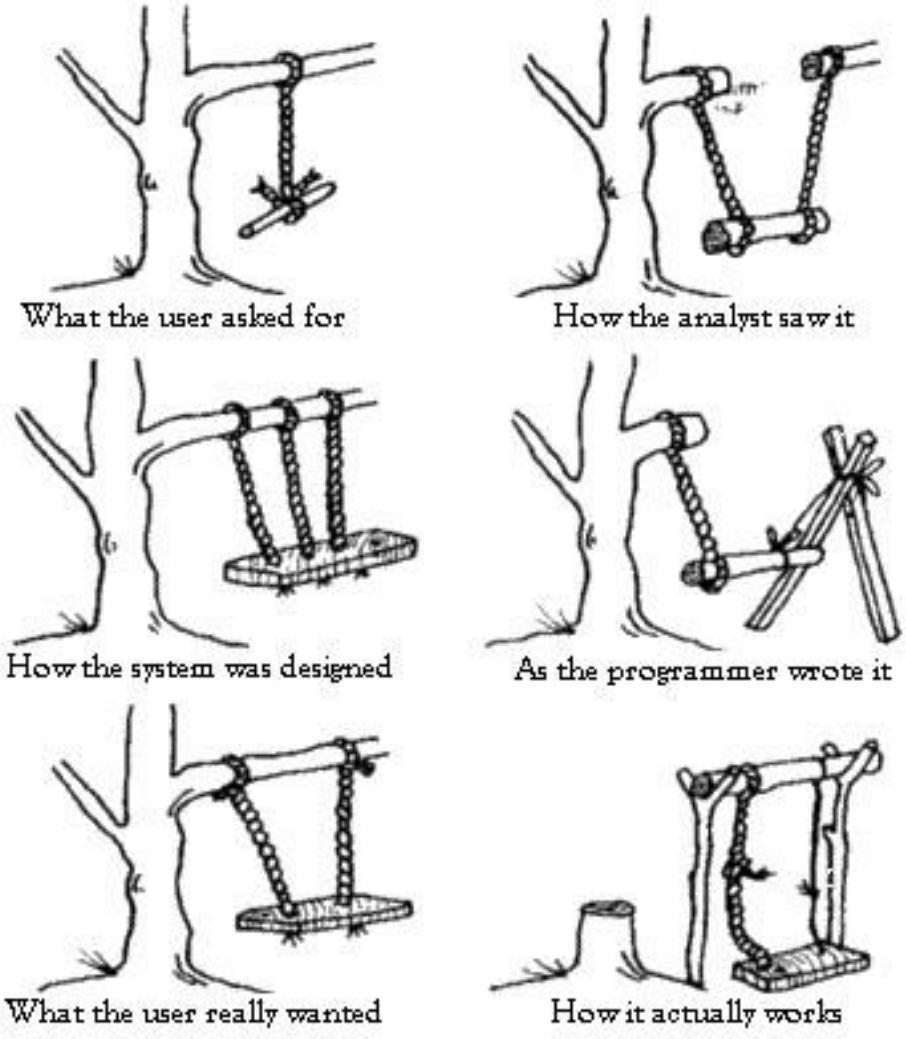
Documentation & Maintenance

Jana Kostičová, 16.5.2016

Documentation

Why documentation?

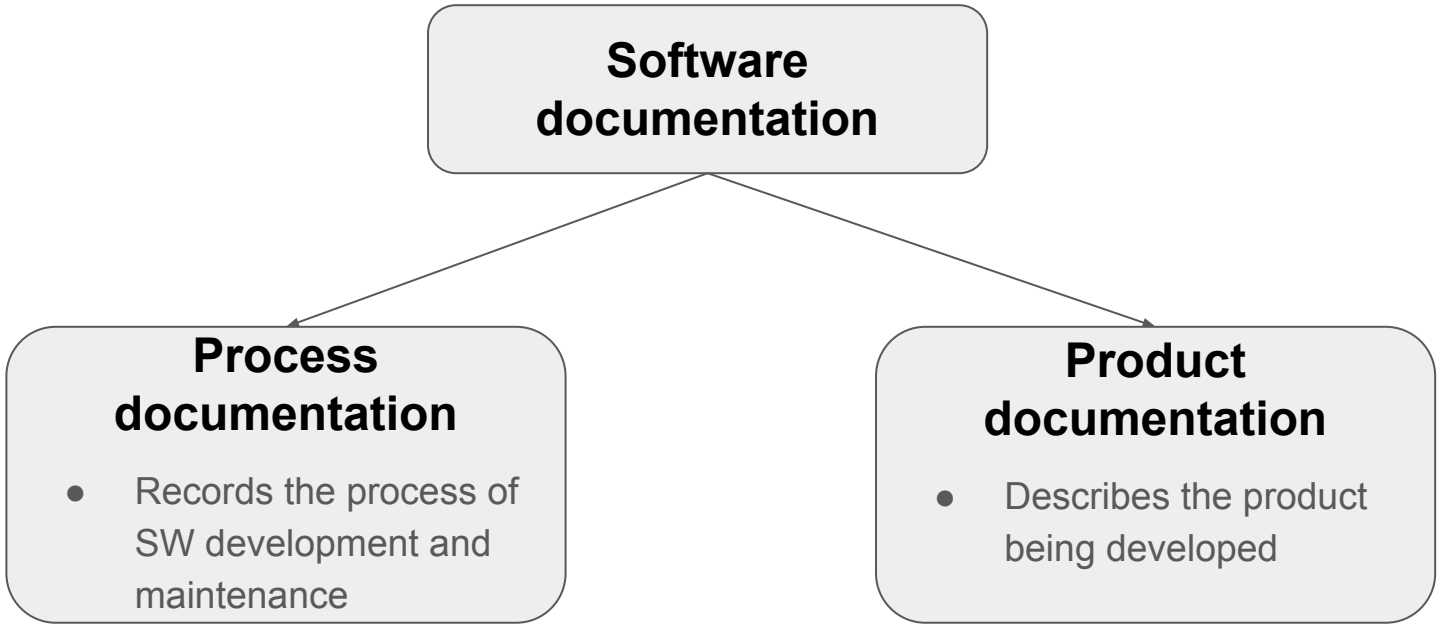
1. Facilitates communication
   - Within the development team itself
   - Between the development team and the project management
   - With customer
2. Records contracts and agreements



Why documentation?

3. Provides information
    - For users and system administrators
    - For future maintenance
    - For project management

Classification



Process documentation:

-> Records the process of SW development and maintenance

- PM documents
  - Predict and control the SW process
  - E.g., schedules, plans, estimates
- Reports
  - Report resource usage during the SW process
  - E.g., amount of man-days consumed, deadlines satisfaction
- Standards
  - Set out how the SW process is to be implemented
  - E.g., coding standards, documentation standards
- Communication documents
  - Record the details of everyday communication
  - E.g., memos, e-mail discussions, meeting minutes, version history

➢ These documents becomes quickly **out of date**!

Product documentation

➢ These documents have **longer** lifespan

-> Describes the product being developed

1. **System documentation** -> Describes how the system works
   - Requirements
   - Architecture & design
   - Source code listings (commented)
   - Validation & verification documents (testing,..)
   - Maintenance documents (List of Known Bugs, HW and SW dependencies, ..)
2. **User documentation** -> Tells user how to use the SW product
   - Should take into account all relevant user classes
     - E.g., end users vs. system administrators
   - Should take into account various levels of expertise
     - E.g., beginners -> screenshots, tutorials, use cases/scenarios
       advanced users -> function reference, detailed description

User documentation - typical documents

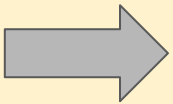
- Functional description
  - Provides system overview - a brief description of services provided
- System installation document
  - Describes how to install the system in a particular environment
- Introductory manual
  - Informal introduction to the system - describes its “normal” usage
- System reference manual
  - Describes all system functions/services, error messages and error recovery methods in detail
- System administrator’s guide
  - Documents the system’s faults and advanced configuration options
- But also: Context help, On-line help, FAQs, Knowledge Base, Tutorials, Mailing lists, Forums, Blogs, ..

➢ Note that it might be difficult to change already published API documentation (why?)

Document form

- Documents - Office, PDF, text, HTML,..
- Diagrams - UML, ..
- Wiki
- Document/content management systems
- E-mail messages
- Bug/issue reports
- Version history
- …

Each (product) document provides a separate view of the system and these views overlap



It is important to keep all documents up-to-date and mutually consistent !

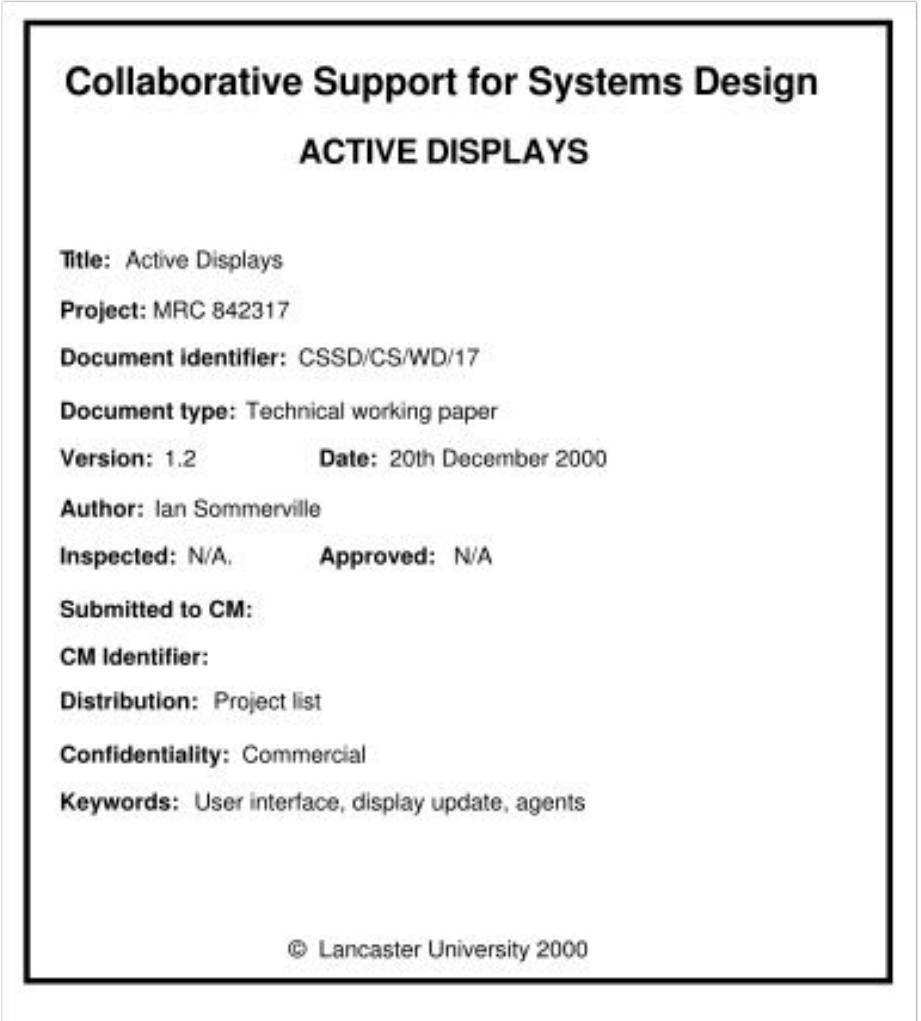
Document structure

- All documents for a given product should have a similar structure
  - A good reason for product standards
- The IEEE Standards list such a structure
  - It is a superset of what most documents need

Common “best practices”:

- Project identification
- Document identification
- Author, Approver
- Document type
- Current version, history
- Distribution list
- Confidentiality class
- Abstract, Keywords
- Copyright notice
- List of notions and abbreviations
- Table of contents, division into chapters and subchapters
- Index

Cover page example



Collaborative Support for Systems Design

ACTIVE DISPLAYS

Title: Active Displays

Project: MRC 842317

Document identifier: CSSD/CS/WD/17

Document type: Technical working paper

Version: 1.2 **Date:** 20th December 2000

Author: Ian Sommerville

Inspected: N/A. **Approved:** N/A

Submitted to CM:

CM Identifier:

Distribution: Project list

Confidentiality: Commercial

Keywords: User interface, display update, agents

(Sommerville, 2002)

Documentation management

- Amount of documentation grows quickly
- It is inevitable to manage the documentation efficiently
  - Use predefined document templates
  - Specify location for each type of document clearly
  - Use document (content) management systems
  - Use version control
  - Generate documentation automatically (Doxygen, NDoc, javadoc, EiffelStudio, Sandcastle,...)
- Recommended minimal system documentation:
  - Requirements specification, architecture/design documents, commented source code

➢ It is better to provide a minimal but up-to-date and consistent documentation than a comprehensive but poorly maintained documentation

Maintenance

Software maintenance

= Modifying a system after it has been put into use

✓ Modifies existing components
✓ Adds new components to the system

x (Normally) does not significantly change the system's architecture

Maintenance is not only bug fixing!

But also: adapting the software to changing requirements, changing environment, ...

➢ In fact, corrections represent only about 25% of all maintenance tasks

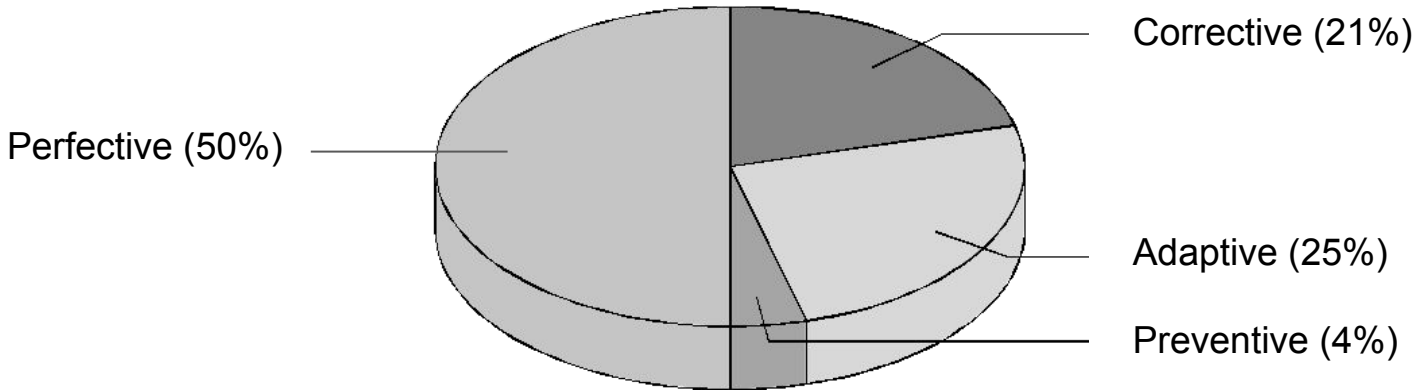
Why software maintenance?

Environment of system operation changes in time

-> Requirements on system changes in time (and new requirements emerge)
-> Systems MUST be maintained if they are to remain useful in an environment

Types of maintenance (ISO/IEC 14764)

1. Adaptive: Modifying the system to cope with environment changes (computer, OS, etc.)
2. Perfective: Modifying the system to satisfy new or modified requirements
3. Corrective: Correcting discovered problems
4. Preventive: Detecting and correcting latent faults before they become effective faults



(Lientz and Swanson,, 1980)

Why does software maintenance cost so much?

- It is usually more expensive to add functionality after a system has been developed rather than design this into the system
- Expensive activity: To figure out
  - WHAT part of code to modify and
  - HOW to modify it

- Overall maintenance costs:
  - Usually 2* to 100* greater than development costs (Ian Sommerville, 2000)
  - Affected by both technical and non-technical factors

Factors affecting maintenance cost

- Team stability
  - Costs are reduced if the same staff are involved for some time
  - In case of staff turnover, "cultural" knowledge of the software is lost
- Contractual responsibility
  - The developers of a system may have no contractual responsibility for maintenance so there is no incentive to design for future change
- Staff skills
  - Maintenance staff are often inexperienced and have limited domain knowledge
  - Maintenance is generally considered as an unglamorous task and is typically assigned to the team newcomers
- Inadequate configuration management
  - Different representations of a system are out of step

Factors affecting maintenance cost

- Inadequate documentation
  - Insufficient, incomplete, inconsistent or out-of-date documentation makes it more difficult to understand the system
  - Reverse engineering may help
- Inflexible design/architecture
  - The architecture and/or design of the system is too rigid to allow for simple implementation of requested changes
  - Costs are increased especially in case that significant changes in original software design are not allowed
- Program age and structure
  - Programs are poorly structured already during the initial development
  - As programs age, their structure is degraded and they become harder to understand and change (e.g., old languages, compilers, programming styles, design patterns)
  - Maintenance corrupts the software structure so makes further maintenance more difficult.

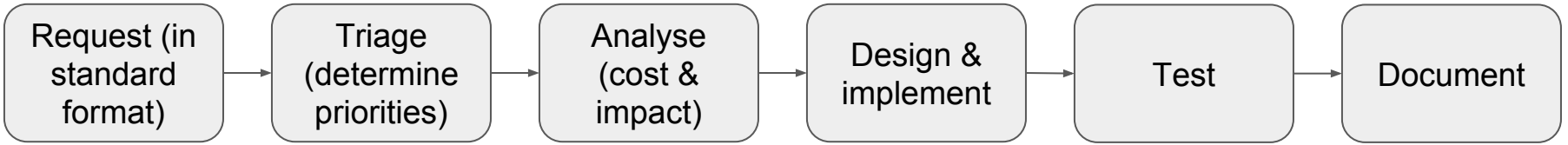
Well-known maintenance examples

- Y2K (1.1.2000) - worldwide
- SKK -> EUR (1.1.2009) - Slovakia

-> Many systems had to be updated
-> In both cases, complex analysis was needed
(find where changes need to be made)

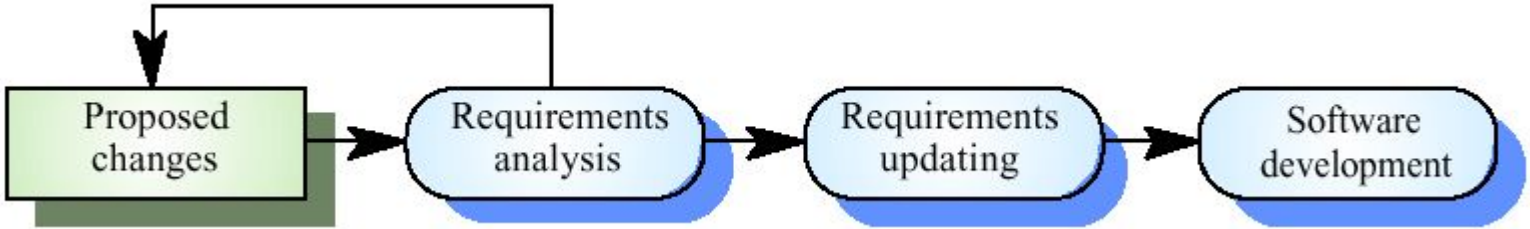
Maintenance process

- Complex and varied (depends on type of maintenance)
- In general:

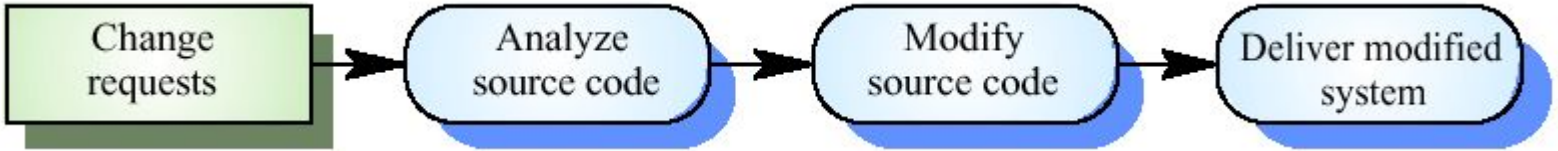


Structured vs. unstructured maintenance

Structured maintenance:



Unstructured maintenance (emergency repair):



Clearly structured maintenance is a more reliable and (usually) a more efficient process - unfortunately, it's not always possible

Maintainable software

- Good initial design
- Understandable software structure
- Accurate documentation
- Good configuration management
- Use of standards (design, language, coding, etc.)

References

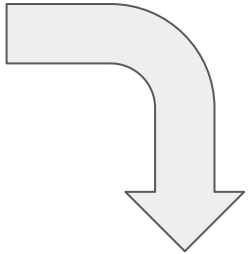
Documentation

- Ian Sommerville: Software documentation, In Software Engineering, Vol 2: The Supporting Processes. R. H. Thayer and M. I. Christensen (eds), Wiley-IEEE Press.[Book chapter], 2002.
- ISO/IEC/IEEE 15289:2015: Systems and software engineering - Content of life-cycle information items (documentation), 2015
- IEEE 829-2008: Standard for Software and System Test Documentation, 2008
- IEEE 1063-2001: Standard for Software User Documentation, 2001
- Google Style Guides: https://github.com/google/styleguide

Maintenance

- Ian Sommerville: Software Engineering (10th edition), 2015
- Pigoski, Thomas. Chapter 6: Software Maintenance (PDF). SWEBOK. IEEE. 2001, http://sce.uhcl.edu/helm/SWEBOK_IEEE/data/swebok_chapter_06.pdf
- Lientz B., Swanson E.: Software Maintenance Management. Addison Wesley, Reading, MA, 1980
- ISO/IEC 14764:2006: Software Engineering - Software Life Cycle Processes - Maintenance, 2006
- IEEE STD 1219-1998: Standard for Software Maintenance, 1998

Modified requirement:
Product code consists of 1-8 digits.
Rationale:
Product code currently consists of 1-6 digits. Starting from 1.1.2017 the e-shop will offer third-party products so that their original product codes will be used. These third-party product codes consist of 8 digits.



Analyze documentation

User documentation:

Field “Product code”: Contains product code. It must consist of 1-6 digits. Duplicate product codes are not allowed.

Requirements specification:

NFR-34: Product code consists of 1-6 digits.
NFR-35: Make extension of product code up to 10 digits easy to implement in the future.

Code with comments:

```
/** Check product code consists of 1-6
digits. Check code is unique in DB.

@param product code to be checked
@throws ValidationException
*/
public void verifyProductCode(String
code) throws ValidationException {
...
```

Architecture & design documents/models:

Table PRODUCT in local DB:

PRODUCT	
CODE	VARCHAR2(10) <pk>
NAME	VARCHAR2(20)
…	...

Table PRODUCT in interface eshop <-> order system

PRODUCT	
PRODUCT_CODE	VARCHAR2(20) <pk>
PRODUCT_NAME	VARCHAR2(20)
...	...