Software Engineering Group Project

COMP2043.GRP

Session 08: Documentation

Acknowledgements

- Some of the materials we use may come directly from previous teachers of this module, and other sources …
- Thank you to (amongst others):

  Dr Julie Greensmith

University of Nottingham
UK | CHINA | MALAYSIA

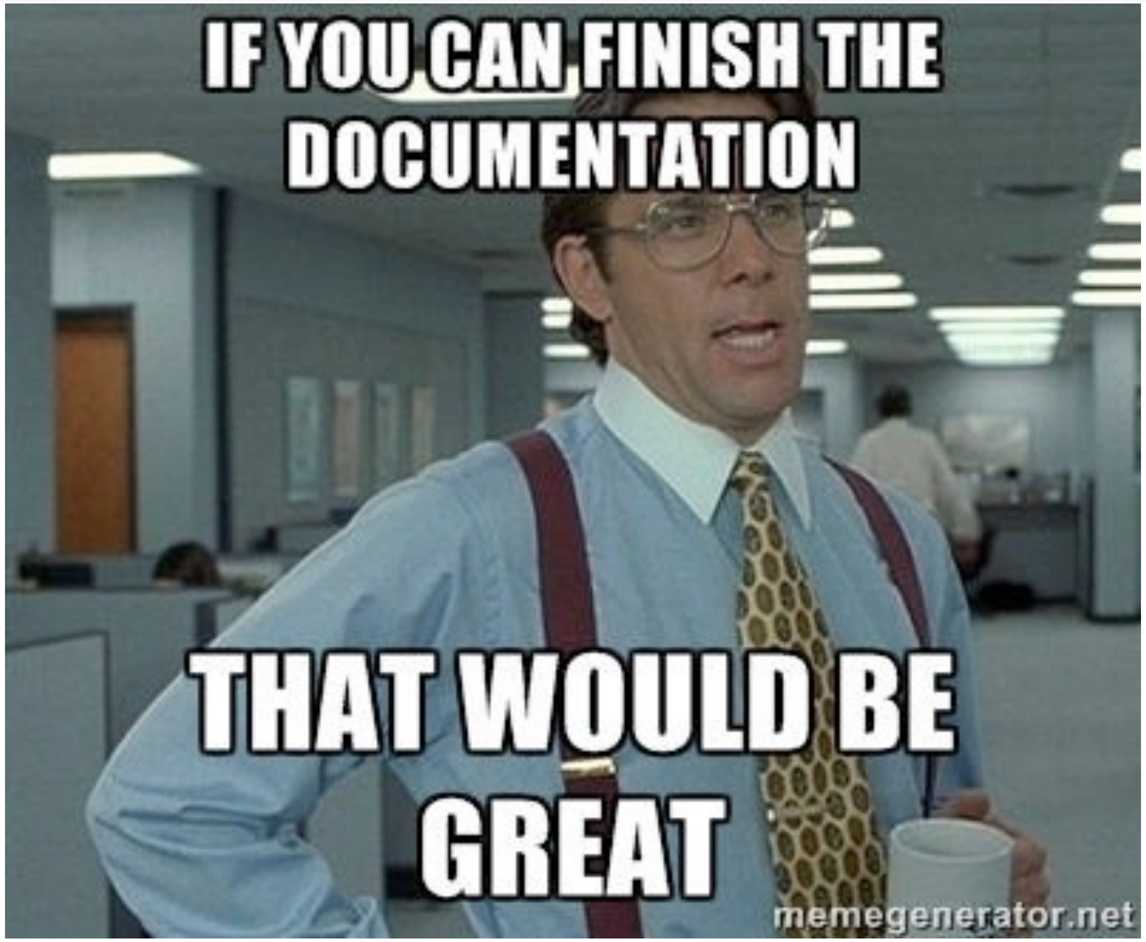
Overview

- Best Practice in Software Documentation
- Tools for Documentation
- General writing for computer science

Learning Outcomes

- Know how to write self-documenting code
- Know what type of documentation is appropriate or important
- Have an awareness of the range of tools available for documentation
- Understand the differences between standard and technical writing

Best Practice in Software Documentation



Documentation when needed

- No documentation is just as bad as excessive documentation
- Types of documents which are useful:
  - code documentation
  - requirements specifications
  - design and test documents
  - user manuals
- Documents need to be managed and shared so version control on documentation is also important

Self-documenting Code

- Use meaningful names
  - classes, variables, functions, packages
  - code version names/testing files/results files
- Short comment lines where function is not obvious
- Can use a tool like Javadoc to autocomment
- Systematic way of managing the basics of a software development project

Short Design Documents

- Can use UML to enhance documentation
- Include documents showing overall architecture, modularisation and high level activity diagrams
  - but must be kept short
- User guide documents also useful
  - but must be kept short

Tracking Tools

- Report on the status of project elements to form documentation
- Many applications:
  - system and software requirements
  - issues and bugs
  - backlog item
  - function points
- Can be very effective

University of Nottingham UK | CHINA | MALAYSIA

Test Tracking

- Keep track of the progress of testing
  - Not just the status of the tests
  - Testing data files
  - Testing results
- Especially important in TDD where you need to ensure 100% testing coverage
- Can monitor the functional status or measure by code completion

Documentation is live!

- If you produce some documents, it is a good practice to update or produce new versions
  - This continues until the end of the development process
- If a document is not up-to-date, it is useless
- Reflects the current state of the project
- Acts as a communication medium for distributed development

Versioning of (Text) Files

- For each new version of a document, a new version number
- Apply a 'stable versioning strategy'
  - decide on a general purpose format
  - e.g. GRP.08.XXX.20241105
  - A version tracking table with change info may be included for better tracking

Templates for Consistency and Speed

- Having a consistent document template is important for companies
  - clear communication
  - saves time
- Header, footer, headings, font sizes must be consistent in and among documents for improved readability
- It is also a good practice to have a cover page, table of contents, table of pictures and a glossary for formal documents - a template saves time

Write Right

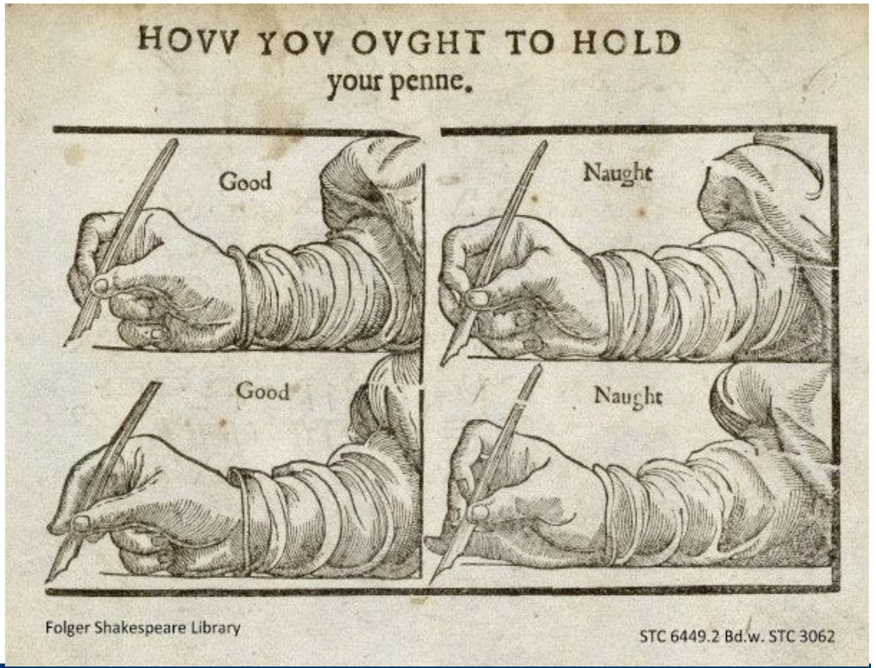
- We might not think of writing as being that important
  - compared to coding
- Projects require good communication
  - written communication is a primary tool
- A framework for our ideas and abstract thoughts
- Pay attention to:
  - formatting, language usage and typing, size, etc.
- Distractions:
  - Typing errors, inconsistent table sizes, indentations

Retrospectives

- Save “lessons-learned” info in a shared place
- Those items are some useful experiences, information or code/configuration sections
- May not be included in standard process documents
- Get the current developers or future developers up to speed about some challenging and repeated operations

Documentation Tools

- Speed up with Autogenerate
- A wide array of different tools used to manage and create documentation
  - Doxygen, NDoc, javadoc, EiffelStudio, Visual studio for C#, ...


HOVV YOV OVGHT TO HOLD your penne.
Good
Naught
Good
Naught
Folger Shakespeare Library
STC 6449.2 Bd.w. STC 3062

Wikis

- Piece of server software that allows users to freely create and edit Web page content using any Web browser
- Wiki supports hyperlinks and has a simple text syntax for creating new pages and crosslinks between internal dynamically
- Wiki is unusual among group communication mechanisms in that it allows the organization of contributions to be edited in addition to the content itself
- Standard users can create and edit content

Wiki Software

- There are *lots* of them!
- A favourite is tiddlywiki
  - http://en.wikipedia.org/wiki/TiddlyWiki
  - http://tiddlywiki.com/
  - especially useful for creative work
- MediaWiki for new media stuff
- You should explore, and find your own favourite

Full written documentation

- Occasionally documents need to be longer
  - e.g. government project justifications
  - e.g. preparing evidence for certification
- Not always best to choose word processor
- Regardless of the editor you use to construct the document

LaTeX

- Used for technical and research documentation
- Mark up language for documentation
- Extremely powerful ☺
- Compiled from source
  - can go into version control repositories
- Separation of content from presentation
- BibTeX is the related bibliography manager

LaTeX IDEs

- Base install of MiKTeX
  - or equivalent
- Many different development environments
  - macros to add markup patterns
  - auto- preview of compiled document
- Excellent for managing very large documents
- e.g. TeXworks, WinEdt, Latexian, TeXshop (Mac), TeXstudio, ShareLaTeX, Overleaf …

Writing For Computer Science

Why Improve?

- Writing well gives you an “unfair advantage”
- Writing well matters in getting your work published in top venues and helps communicate your ideas and work
  - Who do you think is a good writer?
  - Where can you get samples of this kind of writing?

10 Tips

1. Distinguish real grammatical rules from folklore
   - don't write ghetto innit
2. Use subjects to name the characters in your story
3. Use verbs to name their important actions
4. Open your sentences with familiar units of information
5. Begin sentences constituting a passage with consistent topics/subjects

10 Tips

6. Get to the main verb quickly: avoid long introductory phrases and clauses
7. Push new, complex units of information to the end of the sentence
8. Be concise and cut meaningless and repeated works (and the obvious)
9. Keep in the active voice
10. Write as something that ***you*** would like to read

The Three B's

- Brevity
- Balance
- Benefit

University of Nottingham
UK | CHINA | MALAYSIA

Here is an introduction to a paper - what do you think it is about?

In 2005, when women made up of 15% of computer science undergraduates, Harvard president and economist Larry Summers suggested that gender differences in "overall IQ, mathematical ability, scientific ability" kept women out of engineering and science fields.

One year later, Michael Nettles and Catherine Millet reported in their book "Three Magic Letters" that of all surveyed doctoral students in mathematics and engineering, African Americans were more than three times less likely than whites to publish and had lower completion rates than either white students or international students.

Brevity

- Write less
- Use short sentences
- Avoid passive voice
- Write one thing per sentence
- Avoid too many buzzwords or jargon
  - technical language is *less* complex than standard English
- Group ideas into paragraphs
- Use subheadings as signposts

Balance

- Read technical documentation to get an idea of the style
- Never write colloquially
- Balance the formal with the informal
  - keep it lively in the active voice
- Provide a mix of formal definitions with prose
  - both equally important

Benefit

- Think about who cares
- Who will read it?
- What is my big idea?
- Is this open to interpretation or have I communicated precisely what I intended?
- Maybe present the problem first, then your solution to give the best lead in

Writing is iterative, like agile SEMs

- Give yourself time to reflect, write, review, refine ...
- Give others a chance to read/review and provide feedback
- Get a reader's point of view
- Find a friend to help proofread
- Edit, edit, edit ...

Summary

- As computer scientists, we need words just as much as any other discipline
  - code documentation
  - requirements documentation
  - testing and user manuals
- Many different tools to autogenerate code
- Collaborative documentation with Wikis
- Aim to improve the quality of your writing