SOFTWARE ENGINEERING

DOCUMENTATION

FACETS OF SOFTWARE ENGINEERING

- Requirements
- Design & Architecture
- Implementation
- Quality Assurance
- Documentation
- Packaging & Delivery
- Maintenance & Support

Team Process

OVERVIEW

- Any mechanism used to teach users how to interact with an application
  - Hard copy manual
  - Website
  - Tutorials (written walkthrough, video)
  - Tooltips / Balloon Help
  - Game Introductory Segments
  - Kiosk Menus

OVERVIEW

- Product v. Project
- Product documentation commonly written by a documentation team
  - Professional, trained technical writers
    - Requires close interaction with developers (don’t always get it)
- Project documentation usually provided by the developers
  - Often isn’t very good
- Versioned (“since version X”)

CONSIDERATIONS (1 OF 2)

- Intended audience
  - Administrators (“admins”)
  - Users
  - Deployers (person installing/configuring the product)

CONSIDERATIONS (2 OF 2)

- Level of prerequisite knowledge required
  - Don’t make assumptions about your readers’ experience
- Accessibility considerations
  - Visually Impaired (e.g. color blind)
  - Physically Impaired

TYPES OF DOCUMENTATION

INSTALLATION GUIDE

- Provide both clean installation and upgrade instructions
- List prerequisites
  - Hardware specs
  - Networking specs
  - Base operating system
  - Dependencies
- Differentiate between different levels of configuration
  - Basic (can be similar to a “Quick Start” guide)
  - Advanced
  - Security
- Example: http://docs.pulpproject.org/en/2.11/user-guide/installation/index.html

RELEASE NOTES

- Describe the differences between application versions
- Highlight new features
- Typically list fixed bugs
- List current “known issues”
- Include installation notes and upgrade concerns (if any)
- Example: http://docs.pulpproject.org/en/2.11/user-guide/release-notes/index.html

USER DOCUMENTATION (1 OF 2)

- Benefits of web-based over hard copy
  - Can be written after code freeze
  - Can be updated after release
- Typically scoped to a user type (Administrator, User, etc.)

USER DOCUMENTATION (2 OF 2)

- Often found in the application (the "Help" menu)
- Typically
  - Formal
  - Professionally written
  - Undergo an approval process
- Example: http://docs.pulpproject.org/en/2.11/user-guide/

DEVELOPER GUIDES

- Describe how to interact programmatically with the application
  - How to use a library
  - How to write a plugin
- Explains how to contribute to the code (open or closed source)
  - Environment setup
  - Style guidelines
  - Contribution policies
- Example: http://docs.pulpproject.org/en/2.11/dev-guide/index.html

API DOCUMENTATION

- Describes how to use an external service
- Should describe inputs and outputs to the call
  - Should include example data (IMO)
- Example APIs:
  - Integration with Facebook for authentication
  - Mobile apps for a service (Twitter, Instagram, etc.)
  - Loose coupling of internal systems
- Example: http://docs.pulpproject.org/en/2.11/dev-guide/integration/rest-api/index.html

CODE DOCUMENTATION

- Each language has a specific format
  - Java: // or /* */
  - Python: # or """<text>"""
- Types:
  - In code comments (why something behaves a particular way)
  - Class / method documentation
- Tools can generate API documentation by inspecting comments
- Example: http://okaara.readthedocs.org/en/latest/#api-documentation

CODE DOCUMENTATION

CODE DOCUMENTATION OVERVIEW (1 OF 2)

- In-code documentation to describe how to use the code's modules, classes, methods, and variables
- Written for other developers (ignored by the compiler in most cases)
- Structured format built on top of the language's built-in comment syntax

CODE DOCUMENTATION OVERVIEW (2 OF 2)

- External tool used to generate a formatted version (typically HTML)
- IDE's typically have support to render and display
- Often nowhere near as detailed as it should be

JAVADOC

- Included with the JDK
- Reads comments that begin with /** (**not** /*)
- Tags are indicated using the @ symbol
- Comments may include HTML tags
  - <code> and <ul> are the most common
- First sentence of the method documentation is used as the summary

WHAT TO DOCUMENT (1 OF 4)

- Package
  - Description of the scope of the functionality found in the package
  - Not commonly done

WHAT TO DOCUMENT (2 OF 4)

- Class / Module
  - Description of the purpose of the class, how to work with it, and any external requirements for using it
- Tags
  - @author <name>
    - typically frowned upon
  - @version <version>
    - typically not used in favor of tracking through version control

WHAT TO DOCUMENT (3 OF 4)

- Method
  - Description of what the method does, if there are side effects, and any preconditions that must exist before calling it
  - Tags
    - @param <name> <description>
      - describes a single parameter passed when calling the method
      - should indicate valid and invalid values
      - specify once per parameter
    - @return <description>
      - describes what the returned value represents
      - should indicate if `null` may be returned
      - only specified once (most languages are single return)

DOCUMENTATION

- Method (continued)
  - Tags
    - @throws <exception_class> <description>
      - describes a single exception that may be thrown
      - should indicate the conditions under which it is thrown
      - specified once per *likely* exception
        - no need to document every single possibility (such as OutOfMemoryError)
    - @deprecated <optional_description>
      - indicates if the method should no longer be used (no longer supported, has known bugs, etc.)
      - should specify what to use as a replacement

WHAT TO DOCUMENT (4 OF 4)

- Variable
  - Description of what the variable is used for and any criteria about it's possible values
  - Example criteria:
    - expected range
    - units
- Documentation of public v. private variables will differ by project

RESOURCES & EXAMPLES

- Java API
  - https://docs.oracle.com/javase/7/docs/api/
- Liferay Javadoc Guidelines
  - https://dev.liferay.com/participate/javadoc-guidelines?_ga=1.163421628.1056016496.1459185780
- JUnit API (assertions in particular)
  - http://junit.sourceforge.net/javadoc/org/junit/Assert.html

HOMEWORK

- Quiz