Software Engineering (CSE3004)

Coding

Puneet Kumar Jain

CSE Department

National Institute of Technology Rourkela

3/21/2022

Reference

- R. Mall, Fundamentals of Software Engineering, Fifth Edition, PHI Learning Pvt Ltd., 2018.

Coding Phase

- Coding is undertaken once design phase is complete.

- The input to the coding phase is the design document.

- During coding phase:
  - every module identified in the design document is coded and unit tested.

- Objective of coding phase:
  - transform design into code
  - unit test the code.

Coding Standards and Guidelines

- Provide general suggestions regarding coding style to be followed:
- Good software development organizations require their programmers to adhere to some standard style of coding
  - called coding standards.
- Advantage of adhering to a standard style of coding:
  - it gives a uniform appearance to the codes written by different engineers,
  - it enhances code understanding,
  - encourages good programming practices.
- Good organizations usually develop their own coding standards and guidelines:
  - depending on what best suits their organization.

Representative Coding Standards

- Rules for limiting the use of globals:
  - what types of data can be declared global and what can not.
- Naming conventions for
  - global variables (GlobalData)
  - local variables (localData)
  - constant identifiers (CONSTDATA)
- Contents of headers for different modules:
  - The headers of different modules should be standard for an organization.
  - The exact format for header information is usually specified.

Representative Coding Standards

- Header data should contain the following information:
  - Name of the module,
  - date on which the module was created,
  - author's name,
  - modification history,
  - synopsis of the module,
  - different functions supported, along with their input/output parameters,
  - global variables accessed/modified by the module.

- Error return conventions and exception handling mechanisms.
  - the way error and exception conditions are handled should be standard within an organization.
  - For example, when different functions encounter error conditions
    - should either return a 0 or 1 consistently.

Representative Coding guidelines

- Do not use too clever and difficult to understand coding style.
  - Code should be easy to understand.

- Many inexperienced engineers actually take pride:
  - in writing cryptic and incomprehensible code.

- Clever coding can obscure meaning of the code:
  - hampers understanding.
  - makes later maintenance difficult.

- Avoid obscure side effects.
  - An obscure side effect is one that is not obvious from a casual examination of the code.
  - For example: if a global variable is changed obscurely in a called module, it becomes difficult for anybody trying to understand the code.

Representative Coding guidelines

- Do not use an variable for multiple purposes.
- The rationale given by programmers for such use:
  - memory efficiency: same variable used in three different ways uses just one memory location.
- Leads to confusion and annoyance
  - Also makes future maintenance difficult.
- Each variable should be given a name indicating its purpose:
  - This is not possible if an identifier is used for multiple purposes.

Representative Coding guidelines

- Code should be well-documented.

- Rules of thumb:
  - on the average there must be at least one comment line for every three source lines.

- Do not make lengthy functions:
  - The length of any function should not exceed 10 source lines.
  - Probably do too many different things and hence beocmes very difficult to understand

- Do not use goto statements.

- Use of goto statements:
  - make a program unstructured
  - make it very difficult to understand.

Code inspection and code walk through

- After a module has been coded:
  - Code inspection and code walk through are carried out to ensures that coding standards are followed
- Detect as many errors as possible during inspection and walkthrough:
  - detected errors require less effort for correction
    - much higher effort needed if errors were to be detected during integration or system testing.

Code Walk Through

- An informal code analysis technique:
  - Undertaken after coding of a module is complete.
- A few members of the development team select some test cases:
  - simulate execution of the code by hand using these test cases.
- Even though an informal technique:
  - several guidelines have evolved over the years making this naive but useful analysis technique more effective.
  - These guidelines are based on
    - personal experience, common sense, and several subjective factors.

Code Walk Through

- The guidelines should be considered as examples:
  - rather than accepted as rules to be applied dogmatically.
- The team performing code walk through should not be either too big or too small.
  - Ideally, it should consist of between three to seven members.
- Discussion should focus on discovery of errors:
  - and not on how to fix the discovered errors.
- To foster cooperation:
  - avoid the feeling among engineers that they are being evaluated in the code walk through meeting,
  - managers should not attend the walk through meetings.

Code Inspection

- In contrast to code walk throughs,
  - code inspection aims mainly at discovery of commonly made errors.
- During code inspection:
  - the code is examined for the presence of certain kinds of errors,
  - in contrast to the hand simulation of code execution done in code walk throughs.
- For instance, consider:
  - classical error of writing a procedure that modifies a formal parameter
  - while the calling routine calls the procedure with a constant actual parameter.

Code Inspection

- Good software development companies:
  - collect statistics of errors committed by their engineers
- A list of common errors:
  - Use of uninitialized variables.
  - Nonterminating loops.
  - Array indices out of bounds.
  - Incompatible assignments.
  - Improper storage allocation and deallocation.
  - Actual and formal parameter mismatch in procedure calls.
  - Use of incorrect logical operators or incorrect precedence among operators.
  - Improper modification of loop variables.
  - Comparison of equality of floating point values, etc.
  - Also during code inspection, adherence to coding standards is checked.

Software Documentation

- When developing a software product we develop various kinds of documents, in addition to the source code:
  - users' manual,
  - software requirements specification (SRS) document,
  - design document, test document,
  - installation manual, etc.

- All these documents are a vital part of good software development practice.

- Good documents enhance understandability and maintainability of a software product.

Internal Documentation

- Different types of software documents can be classified into:
  - internal documentation,
  - external documentation (supporting documents).

- **Internal documentation:**
  - documentation provided in the source code itself.

- **External documentation:**
  - documentation other than those present in the source code.

Internal Documentation

- Internal documentation provided through:
  - use of meaningful variable names,
  - code indentation,
  - code structuring,
  - use of enumerated types and constant identifiers,
  - use of user-defined data types, etc.
  - module headers and comments
- Good software development organizations:
  - ensure good internal documentation through coding standards and coding guidelines.
- Careful experimentation suggests:
  - meaningful variable names is the most useful internal documentation.

External Documentation

- Users' manual,
- Software requirements specification document,
- Design document,
- Test documents,
- Installation instructions, etc.
- All the documents for a product should be up-to-date:
  - Even a few out-of-date documents can create severe confusion.
- An important feature of good documentation is consistency.

Textual Documents

- Readability is an important attribute of textual documents.

- Readability determines understandability
  - hence determines maintainability.

- A well-known readability measure of text documents:
  - Gunning's Fog Index.

Gunning’s Fog Index

$$F = 0.4 \left(\frac{\text{Number of Words}}{\text{Number of Sentences}} + \text{Percentage of words of 3 or more syllables} \right)$$

- F corresponds to the number of years of schooling to easily understand the document.
- syllable is a group of words that can be independently pronounced. For example, the word “sentence” has three syllables (“sen”, “ten”, and “ce”).

Gunning’s Fog Index

- Consider the following sentence:

“The Gunning’s fog index is based on the **premise** that use of short **sentences** and simple words makes a **document** easy to **understand**.”

Calculate its Fog index.

- The fog index of the above example sentence is
- 0.4 x (23/1) + (4/23) x 100 = 26

- End of Chapter

Thanks