Software Design And Development (CM2010)

Course Notes

Felipe Balbi

October 12, 2020

Contents

Week 1 **3**
Module Introduction 3
1.0202 Reference points 3
1.0203 SWEBOK guide and IEEE vocab 4
1.0204 What is a module? 4
1.0206 What is module complexity? 4
1.0208 Complexity references 5

Week 1

Key Concepts

- Define the terms module and module complexity in terms of computer programs and systems.
- Identify the modules present in computer programs and systems.
- Analyse program code in terms of its complexity.

Module Introduction

The objectives of the module are:

1. Write programs using control flow, variables, and functions
2. Use defensive coding and exception handling techniques to prevent processing of invalid data and to handle unexpected events
3. Use Version Control tools to manage a codebase individually and collaboratively (`git`)
4. Define Test-Driven Development and write Unit Tests
5. Assign different categories of module coupling and cohesion to a given program
6. Describe how User Testing can be carried out and evaluated

We will use three different languages throughout the course. They are: C++, Python, and JavaScript.

1.0202 Reference points

Two main references will be used during this course: the *SWEBOK* and ISO/IEC/IEEE 24765:2010 - IEEE Systems And Software Engineering Vocabulary.

SWEBOK is the Software Engineering Body Of Knowledge. From this reference material, we focus on the topic of *Design*.

Design is concerned with the Design fundamentals, key issues of software, software structure and architecture, user interface design, software design quality analysis and evaluation, software design notations, software design strategies and methods, and software design tools.

ISO/IEC/IEEE 24765:2010 is a sort of *dictionary* defining common terms.

1.0203 SWEBOK guide and IEEE vocab

- ISO/IEC/IEEE International standard – Systems and software engineering – Vocabulary, ISO/IEC/IEEE 24765:2010(E) Dec 2010, pp.1–418.
- R.E.D. Fairley, P. Bourque and J. Keppler, The impact of SWEBOK Version 3 on software engineering education and training in 2014 IEEE 27th Conference on Software Engineering Education and Training (CSEE&T). (Klagenfurt, Austria: IEEE, 2014).

1.0204 What is a module?

"... a mechanism for improving the flexibility and comprehensibility of a system while allowing the shortening of its development time". Parnas, 1972.

Once software has been modularized, different parts can be replaced and/or used in different software. Moreover, modularity makes software easier to understand because each small piece can be studied and understood in isolation.

During this course, we define a module as:

- program unit that is discrete and identifiable with respect to **compiling**, **combining** with other units, and **loading**;
- logically separable part of a program;
- set of source code files under version control that can be manipulated as one;
- collection of both data and the routines that act on it.

1.0206 What is module complexity?

Complexity is defined as:

1. The degree to which a system's design or code is **difficult to understand** because of numerous components or relationships among components;
2. Pertaining to any of a set of structure-based **metrics** that measures the attributes in (1);
3. The degree to which a system or component has a design or implementation that is difficult to understand and **verify**.

Simplicity is defined as:

1. The degree to which a system or component has a design or implementation that is **straightforward** and **easy** to understand;
2. Software attributes that provide implementation of functions in the most understandable manner.

1.0208 Complexity references

Please read the following articles. The first is a paper about structural complexity and how it changes over time. The second is the classic McCabe paper on module complexity.

- R.S. Sangwan, P. Vercellone-Smith and P.A. Laplante 'Structural epochs in the complexity of software over time', IEEE Software 25(4) Jul-Aug 2008, pp.66–73.
- T.J. McCabe 'A complexity measure', IEEE Transactions on Software Engineering SE-2(4) Dec 1976, pp.308–320.