Quantitative assessment of good programming practices impact on software maintenance

Experiment by:
Miguel Esteban Ramos Montilla

Software Maintenance

In software engineering, software maintenance refers to modifications made to the software product after delivery to correct faults, to improve performance or other attributes.

Software Maintainability

The software maintainability may be defined qualitatively as the easiness to understand, to correct, to adapt and/or to improve the software product.

The problem

- Often there are people working in the software maintenance process who were not involved in the software development process

- It is hard to them perform the system maintenance if the system doesn't have a good level of maintainability.

Base Statement

- The implementation of good programing practices improves the software maintainability.

Question

Every good-programming-practice influences the software maintainability equally?

- Commening
- Space and consistent indentation
- Consistent Naming Scheme
- DRY Principle and tasks divided in functions

Objective

- The goal of the proposed experiment is to demonstrate through numerical results which are the advantages of using good programming practices for the software maintenance process.

- The experiment also seeks to find out, into a group of good programming practices selected, which one is the most helpful when maintaining software is needed.

Hypothesis

- One of the selected good programming practices is more helpful than the others in the software mainteinance process.

Variable Selection

- The metrics selected to measure the code maintainability were: **problem recognition time** and **problem analysis time**.

- **Independent variable:** GPP (Good Programing Practice) to evaluate.

- **Dependent variable:** Problem solution time: It's implicit the problem recognition time and the problem analysis time.

Experiment Setup

- Select the GPPs
- To build an easy program to be maintained
- The original program has to implement all the GPP to be evaluated.
- To build another 4 versions of the program where every version doesn't implement one of the GPP.
- To choose an easy-to-work IDE in order to compile the program (Code::Blocks).
- To write the experiment instructions.

Experiment Setup

1) Commening
2) Space and consistent indentation
3) Consistent Naming Scheme
4) DRY Principle and tasks divided in functions

Versions/GPPs	1	2	3	4
Version 1		X	X	X
Version 2	X		X	X
Version 3	X	X		X
Version 4	X	X	X	
Original	X	X	X	X

Experiment Instructions

- Software maintenance concept.
- Program description.
- Using pointers in C information
- Maintenance tasks instructions and descriptions

Maintenance Tasks Composition

- Problem description.
- Modification request.
- Request to register the modification starting time.
- Tests to verify the modification correctness.
- Request to register the modification ending time

Measures collected

- **Task 1 - Changing the division symbol**
  - Starting time __ : __
  - Finish time __ : __
  - Was it accomplished?
- **Task 2 - Error when printing the negative results**
  - Starting time __ : __
  - Finish time __ : __
  - Was it accomplished?
- **Task 3 - Error when subtracting equivalent fractions**
  - Starting time __ : __
  - Finish time __ : __
  - Was it accomplished?
- **Task 4 - Format error not captured**
  - Starting time __ : __
  - Finish time __ : __
  - Was it accomplished?

Experiment Execution

- The experiment was executed with two diferent groups:
- Group 1: 21
- Group 2: 12
- The experiment was executed in two hours.
- The participants were requested to send the modified code by e-mail.

Thanks!