ECAM
BRUSSELS ENGINEERING SCHOOL

Fall 2019
Sébastien Combéfis

I402A Software Architecture and Quality Assessment

Quizz 2: Software metric

This assessment evaluates the following competencies:

- *SA201 – Define what is software quality and explain how it can be ensured*
- *SA204 – Understand how the effort, the structure and the information flow can be measured thanks to metrics*
- *SA301 – Understand the TDD cycle and write a software following this methodology*
- *SA302 – Understand refactoring and use it to improve the quality of a code*
- *SA303 – Choose a suitable metric and use it to evaluate a given quality criterion*
- *SA206 – Compare several metrics to measure a given quality criterion and argue about the best choice*

Three affirmations are given for each assessed competency. For each of them, you have to decide whether it is true or false. To get a star for the competency, you must have the correct answer for the three affirmations.

SA201	**True**	**False**
Just by choosing a good software architecture, you can ensure any quality criterion.	☐	☐
The response time of a software cannot be considered as a quality criterion.	☐	☐
Having redundancy in a software system increases its availability.	☐	☐

SA204	**True**	**False**
The McCabe cyclomatic complexity can be used to measure the required effort to understand the code of a software.	☐	☐
The number of logical lines of source code may be used, combined with other metrics, to measure the required effort to understand the code of a software.	☐	☐
With Halstead software metrics, you have a precise and indisputable measure of the effort that will be needed to write a software, as a number of seconds.	☐	☐

SA301	**True**	**False**
Following the TDD cycle, you first write your code and then you write tests to check whether the code is correct.	☐	☐
Following the TDD cycle helps improving some software quality criteria.	☐	☐
It is impossible to use the TDD methodology with some programming languages because it requires specialised frameworks.	☐	☐

SA302	**True**	**False**
Refactoring is always applied on the machine code of a software after compilation succeeded.	☐	☐
Refactoring transforms the code while keeping the same features and observable behaviour.	☐	☐
Refactoring a code without having any tests written is a good practice.	☐	☐

SA303	**True**	**False**
The source line of code (SLOC) metric may be used to (partially) measure the reliability of a software.	☐	☐
The distance from main sequence metric may be used to (partially) measure the flexibility of a software framework, such as Symfony, for example.	☐	☐
The code coverage metric may be used to (partially) measure the maintainability of a software.	☐	☐

SA206	**True**	**False**
To measure the maintainability of a software, it is better to use the source line of code (SLOC) metric than the code coverage metric.	☐	☐
To measure the reliability of a software, it is better to use the number of coding rule violation metric than the afferent (Ca) coupling metric.	☐	☐
To measure the information flow of a software module, it is better to use the Henry and Kafura fan-in fan-out metric than the McCabe cyclomatic complexity metric.	☐	☐