Correctness

Hao Zhe Kok, Ramsey Towell, Binbin Wu

Correctness

- Definition: If the specifications of the systems are available, correctness assumes that it is possible to determine unambiguously whether a program meets the specifications.

- A program is functionally correct if it behaves according to its stated functional specifications. But may be missing other specifications such as performance and scalability.

- Basically if the software solves the problem if the problem is sufficiently defined.

- Must be unambiguous, can be assessed. Is a mathematical property that establishes the equivalence between the software and its specification.

Correctness

- Specifications are usually specified by people informally using natural language.

- This presents ambiguity because natural language is not consistent.

- Despite this ambiguity, specifications are useful to determine the approximate desirable goals of a given software system.

Correctness

- Correctness can be enhanced by using appropriate tools.

- Such as high-level languages - particularly those supporting extensive static analysis - testing using tools without running the program.

- Likewise, correctness can be improved by using standard proven algorithms or built-in libraries, instead of reinventing the wheel.

- Using proven methodologies and processes.

Application to Project

- Using Cucumber (utilizes Gherkin, which is practically natural language).

- Cucumber provides a way for developers and users to reconcile natural language and actually making the project correct.

- By using behaviour-driven development, the project is tested to ensure its correctness - provided the Gherkin specifications are defined and understood correctly.

Separation Of Concerns

Separation of concerns allows us to deal with different aspects of a problem, so that we can concentrate on each individually.

Separation of concerns is commonsense practice that we try to follow in our everyday life to overcome the difficulties we encounter. The principle should be applied also in software development.

Separation Of Concerns

There are many decisions that must be made in the development of a software product.
Some of them concern features of the product: functions to offer, expected reliability, efficiency with respect to space and time, user interfaces, etc.

Some people concern the development process: the development environment, the organization and structure of the teams, scheduling, control procedures, design strategies, error recovery mechanisms, etc.

Others concern economic and financial matters.

Separation Of Concerns

The only way to master the complexity of a project is to separate the different concerns. First of all, we should try to isolate issues that are not so closely related to the others. Then, we consider issues separately, together with only one relevant details of related issues.

First type: one can separate them in Time.
Second type: one can be separated in terms of qualities.
Third type: separation of concerns allows different views of the software to be analyzed separately.
Fourth type: separation of concerns allows us to deal with parts of the same system separately; here, separation is in terms of size.

Separation Of Concerns

There is an inherent disadvantage in separation of concerns: By separating two or more issues, we might miss some global optimization that would be possible by tackling them together. But our ability to make “optimized” decisions in the face of complexity is rather limited. If we consider too many concerns simultaneously, we are likely to be overwhelmed by the amount of detail and complexity we face.

System designers and architects often face such trade-offs.

Separation Of Concerns

Note that if two issues associated with one problem are intrinsically intertwined, it is often possible to make some overall design decisions first and then effectively separate the different issues.

The most important application of separation of concerns is to separate problem-domain concerns from implementation-domain concerns. problem-domain properties hold in general, regardless of the implementation environment.

Final remark, notice that separation of concerns may result in separation of responsibilities in dealing with separate issues. Thus, the principle is the basis for dividing the work on a complex problem into specific assignments, possibly for different people with different skills.