Object-Oriented Programming

Software Design Characteristics

Michael L. Collard, Ph.D.

Department of Computer Science, The University of Akron

Software Design Characteristics

- Compatibility (Interoperability)
- Fault-tolerance
- Reliability
- Usability
- Performance (Efficiency)
- Scalability
- Robustness
- Extensibility
- Modularity
- Maintainability
- Reusability
- Security
- Portability

Compatibility (Interoperability)

The software can operate with other products designed for interoperability

For example, a piece of software may be backward-compatible with an older version of the same software

- Scenario: Upgrade to version 2
- *Are version 1 data files readable?*
  - *Directly readable or a first-time conversion?*
- *Can version 2 data files be used with version 1?*

Extensibility

Developers can add new capabilities to the software without significant changes to the underlying architecture

- *Are new input formats quickly added?*

- Changing the existing source code - *white-box extensibility*

- Changing nothing about the existing source code - *black-box extensibility*

Modularity

The software comprises well-defined, independent components

This independence leads to better maintainability.

The components are implemented and tested in isolation before being integrated to form the desired software system

Isolation before integration allows the division of work in a software development project.

- Units of modularity depend on the level of design

- For our purposes, think of classes. Can we test each class individually?

- *What is in each class?*

Fault-tolerance

The software is resistant to and able to recover from component failure

- *Can we continue operations even with diminished resources?*

Maintainability

A measure of how easily developers can fix bugs or functional modifications

High maintainability can be the product of modularity and extensibility

- *Can we easily maintain the current functionality?*

Reliability (Software durability)

The software can perform a required function under stated conditions for a specified time

- Every time we run it, we expect a result in the time expected

- Counterexample: Every time we run the program, we get a different result

Reusability

Some or all aspects of the software are usable in other projects with little to no modification

- Often focused on, but not always the most crucial characteristic

- Much of *over design* is from too much focus on reusability

Robustness

The software can operate under stress or tolerate unpredictable or invalid input

For example, a design can be resilient to low memory conditions

- Programs depend on resources: memory, disk space, file handles, etc.
- *Can it continue to operate with reduced resources?*

Security

The software can withstand and resist hostile acts and influences

- Most common "hostile acts" are through input
- *Does the design allow for the inputs to be verified?*

Usability

The software user interface must be usable for its target user/audience

Default values for the parameters are a good choice for most users

- This is not just the end-user interface but even our APIs

Performance

The software performs its tasks within a time frame acceptable for the user and does not require too much memory

- Considers both time and space (memory used)
- "Acceptable" time is based on requirements

Portability

The software should be usable across many different conditions and environments

- If the software is too tightly bound to a particular operating environment, it might not work in others

Scalability

The software adapts well to increasing data or the number of users

- Vertical Scaling, *scale up*, is handling an increase in demand by increasing resources in an instance

- Horizontal Scaling, *scale out*, is handling an increase in demand by increasing the number of instances

Purpose of a Taxonomy

Compatibility

Fault-tolerance

Reliability

Usability

Performance

Scalability

Robustness

Extensibility

Modularity

Maintainability

Reusability

Security

Portability

- Common, shared terminology
- Check that we are not overlooking anything
- Shows that there is not just one viewpoint
- Some characteristics may be potential tradeoffs with other characteristics
- Some may be highly related as equal partners
- Some may be dependencies
- All of these have costs, and therefore, scoring high for a particular characteristic may not be worth it in certain circumstances