System Testing

What, Why & How.

Maximilian Meffert

A Motivational Question

What is the deliverable
of a software development process?

A Motivational Question

What is the deliverable of a software development process?

a) A collection of artifacts

b) UX and behavior

My Answer

What is the deliverable of a software development process?

a) ~~A collection of artifacts~~ **Just the Medium!**

b) **UX and behavior**

My Little Testing Glossary

White Box Testing

The investigation from an *internal perspective* whether a program works as expected.

Examines the source code of a program, e.g. control flow, data flow, coverage, etc.

Black Box Testing

The investigation from an *external perspective* whether a program works as expected.

Examines the functionality of a program, i.e. whether it is fit to fulfill its purpose.

My Little Testing Glossary

Unit Testing

The investigation whether an *individual program unit* works as expected.

Examines units independently, i.e. in isolation.

Units may be sets of one or more routines:

- procedures, functions or modules (in Procedural and Functional Programming)
- methods, class or interface signatures (in Object Oriented Programming)

Integration & Integrated Testing

The investigation whether *multiple program units in combination* work as expected.

An informal distinction:

- *Integration Testing* involves third party units.
- *Integrated Testing* does not involve third party units.

System Testing

The investigation whether *all program units in combination* (the entire system) work as expected

My Little Testing Glossary

Acceptance Testing

The investigation whether all requirements of a specification are met.

Regression Testing

The investigation whether all requirements of a specification are still met after a change was introduced.

Principles of SOLID Design

5 Design Principles for creating understandable and maintainable software.

Principles of SOLID Design

Single Responsibility Principle

A class should only have one responsibility, i.e. one *“reason to change”* [Robert C. Martin a.k.a. Uncle Bob].

Open/Closed Principle

“Modules should be both open (for extension) and closed (for modification).” [Bertrand Meyer]

Clients of interface methods or abstract methods of abstract base classes are closed for modification but still open for extension through implementation of such methods.

Liskov Substitution Principle

The overall behavior of a program regarding one type should not differ or change for any of its sub-types.

Given a program P with a type T and its sub-type S:

Let q(P,x) be a provable property in P for all instances x of T, then q(P,y) should be true in P for all instances y of S. [Barbara H. Liskov, Jeannette M. Wing]

Interface Segregation Principle

Interfaces should be client-specific, only exposing methods necessary for the client to know.

Dependency Inversion Principle

A business logic should only interact with concrete environment or third party logic (e.g. file system) through abstraction; see Interface Segregation.

1) *“High-level modules should not depend on low-level modules. Both should depend on abstractions.”*
2) *“Abstractions should not depend on details. Details should depend on abstractions.”*

A Blind Spot

Focus on SOLID Design and Unit Testing ensures quality of the medium but not of the deliverable, i.e. behavior of the software.

Behavior Driven Development