SENG321: Requirements Engineering

REQUIREMENTS VALIDATION AND VERIFICATION

Dr. Daniela Damian

thesegalgroup.org

danielad@uvic.ca

Outline

Definitions of V&V activities
V&V activities throughout the development life-cycle

Definitions

*Software **Verification and Validation** (V&V) is the process of ensuring that software being developed or changed will satisfy functional and other requirements (**validation**) and each step in the process of building the software yields the right products (**verification**)*

Validation and Verification

Validation:

"Are we building the right product?"

The software should do what the users/stakeholders need

Verification:

"Are we building the product right?"

The software should conform to its specification

Validation and Verification Criteria

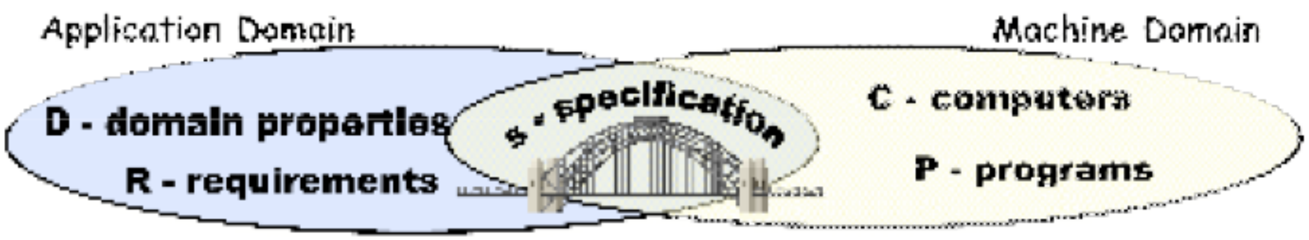


Figure 4: Application domain versus machine domain (Adapted from Jackson 1997)

- Validation criteria:
  - Discover and understand all **domain properties**
  - Discover and understand all **requirements**

- Verification criteria:
  - The **program** satisfies the **specification**
  - The **specification**, given the **domain properties**, satisfies the **requirements**

A V&V Plan

Major **V&V activities:**

Validation: Reviews, inspections and walkthroughs

At all stages

Verification: Testing

1) On the code but also 2) in relation to requirements

The **V&V Plan** should cover all V&V activities during all phases of the life cycle

V&V: On and about Requirements

Validation techniques: reviews, walkthroughs, inspections and prototypes

Inspections of the requirements document for:

- Completeness and consistency
- Conformance to standards
- Conflicts
- Ambiguous requirements

Verification techniques: creating the **Acceptance Testing Plan** and **Verification Matrix** (traceability requirement -> test case)

Development Stages and V&V activities

Requirements. Requirements must be reviewed with the client; rapid prototyping can refine requirements and accommodate changing requirements.

Specification. The specifications document must be checked for *feasibility*, *traceability*, *completeness*, and absence of *contradictions* and *ambiguities.* Specification reviews (walkthroughs or inspections) are especially effective

Design. Design reviews are similar to specification reviews, but more technical. The design must be checked for *logic faults*, *interface* faults, lack of *exception handling*, and *nonconformance* to specifications.

Implementation. Code modules are informally tested by the programmer while they are being implemented (*desk checking*). Thereafter, formal testing of modules is done methodically by a testing team. This formal testing can include *nonexecution-based methods* (code inspections and walkthroughs) and *execution-based methods* (black-box testing, white-box testing).

Integration. Integration testing is performed to ensure that the modules combine together correctly to achieve a product that meets its specifications. Particular care must be given to the interfaces between modules.

Product Testing. The functionality of the product as a whole is checked against its specifications. Test cases are derived directly from the specifications document. The product is also tested for robustness (error-handling capabilities and stress tests). All source code and documentation are checked for completeness and consistency.

Acceptance Testing. The software is delivered to the client, who tests the software on the actual hardware, using actual data instead of test data. A product cannot be considered to satisfy its specifications until it has passed an acceptance test. Commercial off-the-shelf or software downloads software usually undergo *alpha* and *beta* testing as a form of acceptance test.

V&V: Software Architecture (preliminary design) Phase

Updating the preliminary version of the **Acceptance Test Plan** and the **verification matrix.**

Reviews and walkthroughs or inspections of the preliminary designs.

V&V: Software Detailed Design Phase

Complete the **Acceptance Test Plan** and the **verification matrix**, including test specifications and unit test plans.

Conduct **informal reviews and walkthroughs or inspections** of the detailed software

Development Stages and V&V activities

Requirements. Requirements must be reviewed with the client; rapid prototyping can refine requirements and accommodate changing requirements.

Specification. The specifications document must be checked for *feasibility*, *traceability*, *completeness*, and absence of *contradictions* and *ambiguities.* Specification reviews (walkthroughs or inspections) are especially effective

Design. Design reviews are similar to specification reviews, but more technical. The design must be checked for *logic faults*, *interface* faults, lack of *exception handling*, and *nonconformance* to specifications.

Implementation. Code modules are informally tested by the programmer while they are being implemented (*desk checking*). Thereafter, formal testing of modules is done methodically by a testing team. This formal testing can include *nonexecution-based methods* (code inspections and walkthroughs) and *execution-based methods* (black-box testing, white-box testing).

Integration. Integration testing is performed to ensure that the modules combine together correctly to achieve a product that meets its specifications. Particular care must be given to the interfaces between modules.

Product Testing. The functionality of the product as a whole is checked against its specifications. Test cases are derived directly from the specifications document. The product is also tested for robustness (error-handling capabilities and stress tests). All source code and documentation are checked for completeness and consistency.

Acceptance Testing. The software is delivered to the client, who tests the software on the actual hardware, using actual data instead of test data. A product cannot be considered to satisfy its specifications until it has passed an acceptance test. Commercial off-the-shelf or software downloads software usually undergo *alpha* and *beta* testing as a form of acceptance test.

V&V: Implementation Phase

Code inspections and/or walkthroughs.

Unit testing software and data structures.

Locating, correcting, and retesting errors.

Development of **detailed integration and product test procedures**

Code Testing

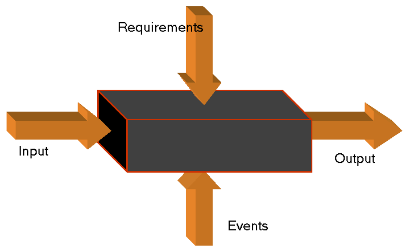
There are two basic types of code testing:

nonexecution-based testing: the module is reviewed by a team:
inspections, walkthroughs

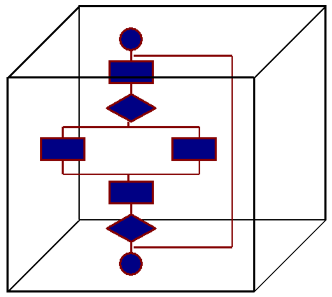
execution-based testing: the module is run against test cases.

Execution-based (automated) Testing

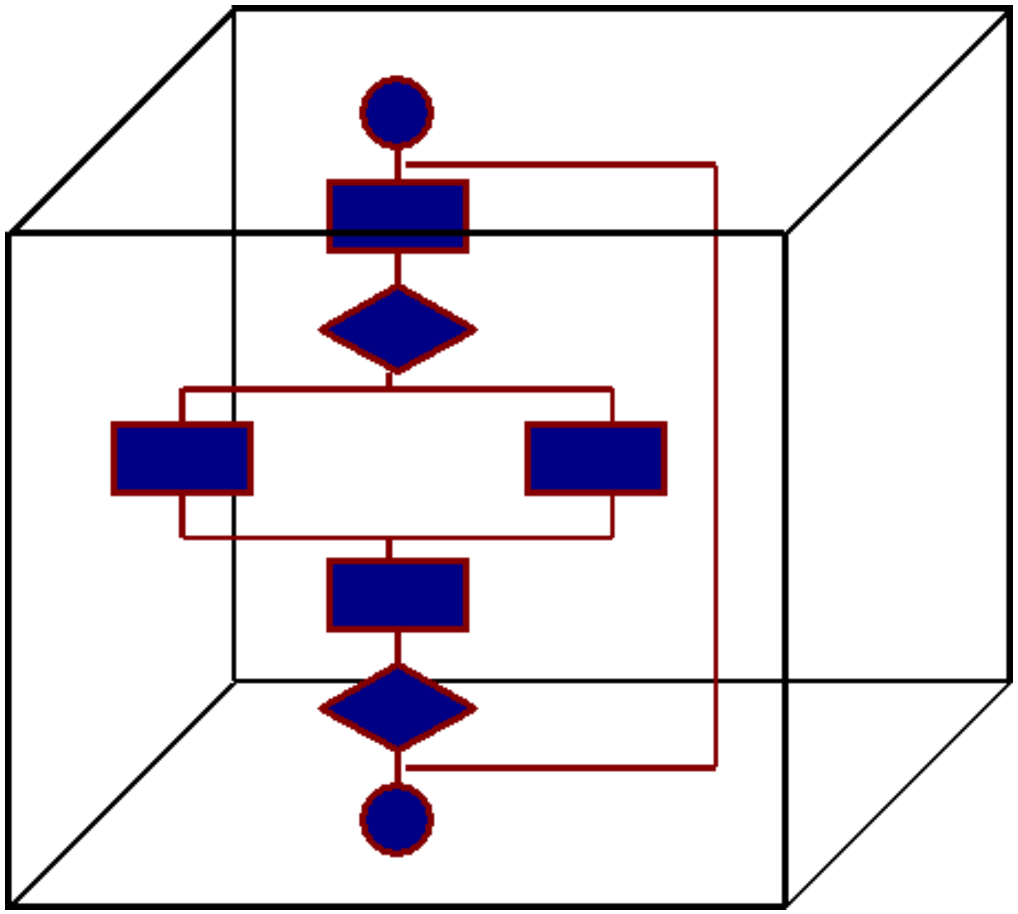
Black-Box Testing: The code itself is ignored; the only information used in designing test cases is the specification document.



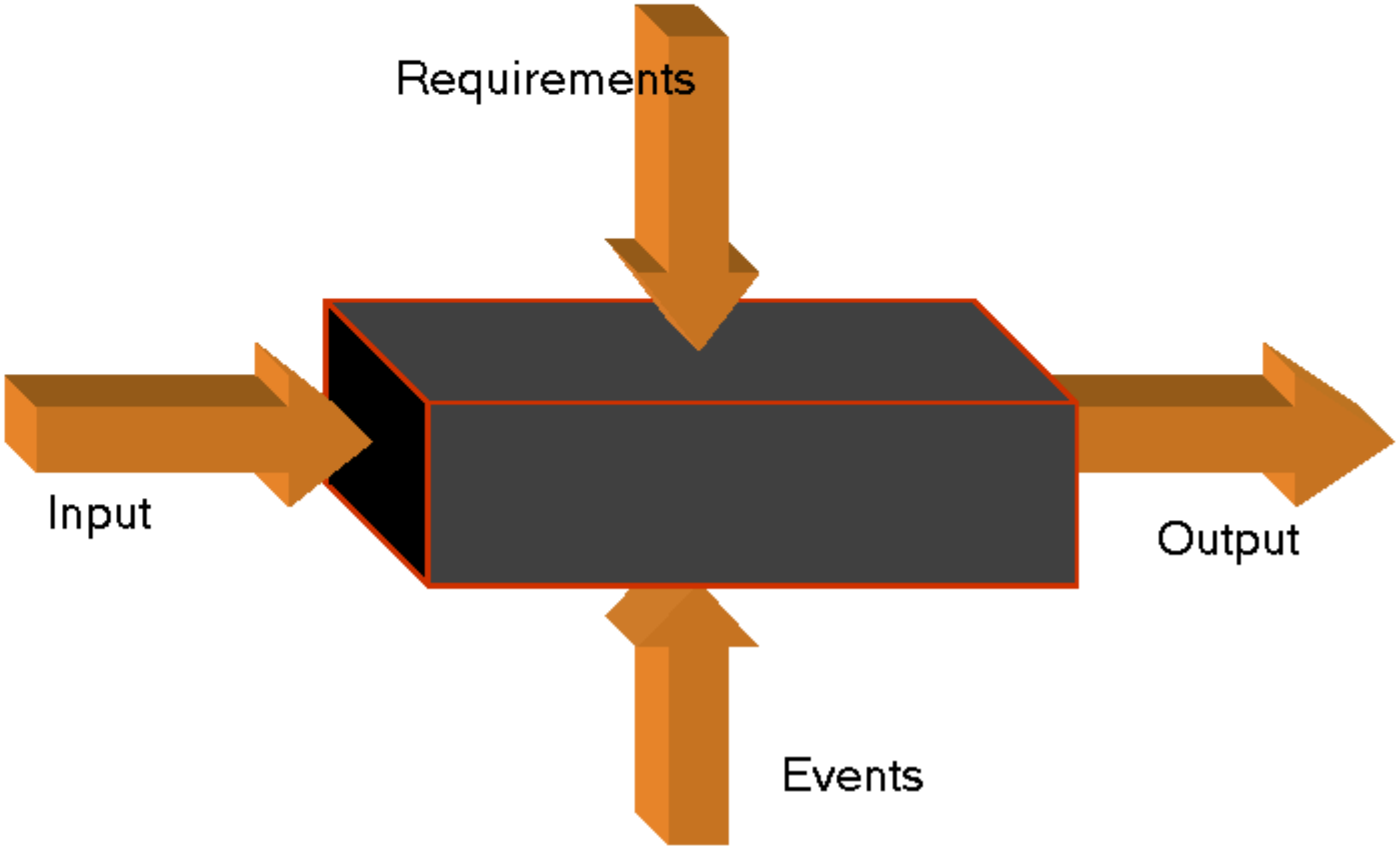
White-Box Testing: The code itself is tested, without regard of the specifications.



White-box Testing



Black-box Testing



Black-box Testing

Exhaustive black-box testing is usually either impossible or unreasonable. The art of testing is to design a small, manageable set of test cases so as to maximize the chances of detecting a fault while minimizing the redundancy amongst of the cases.

Equivalence Testing and Boundary Value Analysis
Equivalence testing, combined with boundary value analysis, is a black-box technique of selecting test cases in such a way that new cases are chosen to detect previously undetected faults. An *equivalence class* is a set of test cases such that any one member of the class is representative of any other member of the class.

Suppose the specification for a database product state that the product must be able to handle any number of records from 1 through 16,383. For this product, there are three **equivalence classes**:

Equivalence class 1: less than one record.

Equivalence class 2: from 1 to 16,383 records.

Equivalence class 3: more than 16,383 records.

Testing the database product then requires that one test class from each equivalence class be selected.

Equivalence Testing and Boundary Value Analysis

A successful test case is one that detects a previously undetected fault. In order to maximize the chances of finding a new fault, a high-payoff technique is *boundary-value analysis*.

Experience has shown that when a test case on or just to one side of a boundary of an equivalence class is selected, the probability of detecting a fault increases.

Testing then should include members of the equivalence class, boundary elements and adjacent members.

V&V: Integration Phase

Conducting tests per test procedures.

Documenting test performance, test completion, and conformance of test results versus expected results.

Providing a **test report** that includes a summary of nonconformances found during testing.

Locating, recording, correcting, and retesting nonconformances.

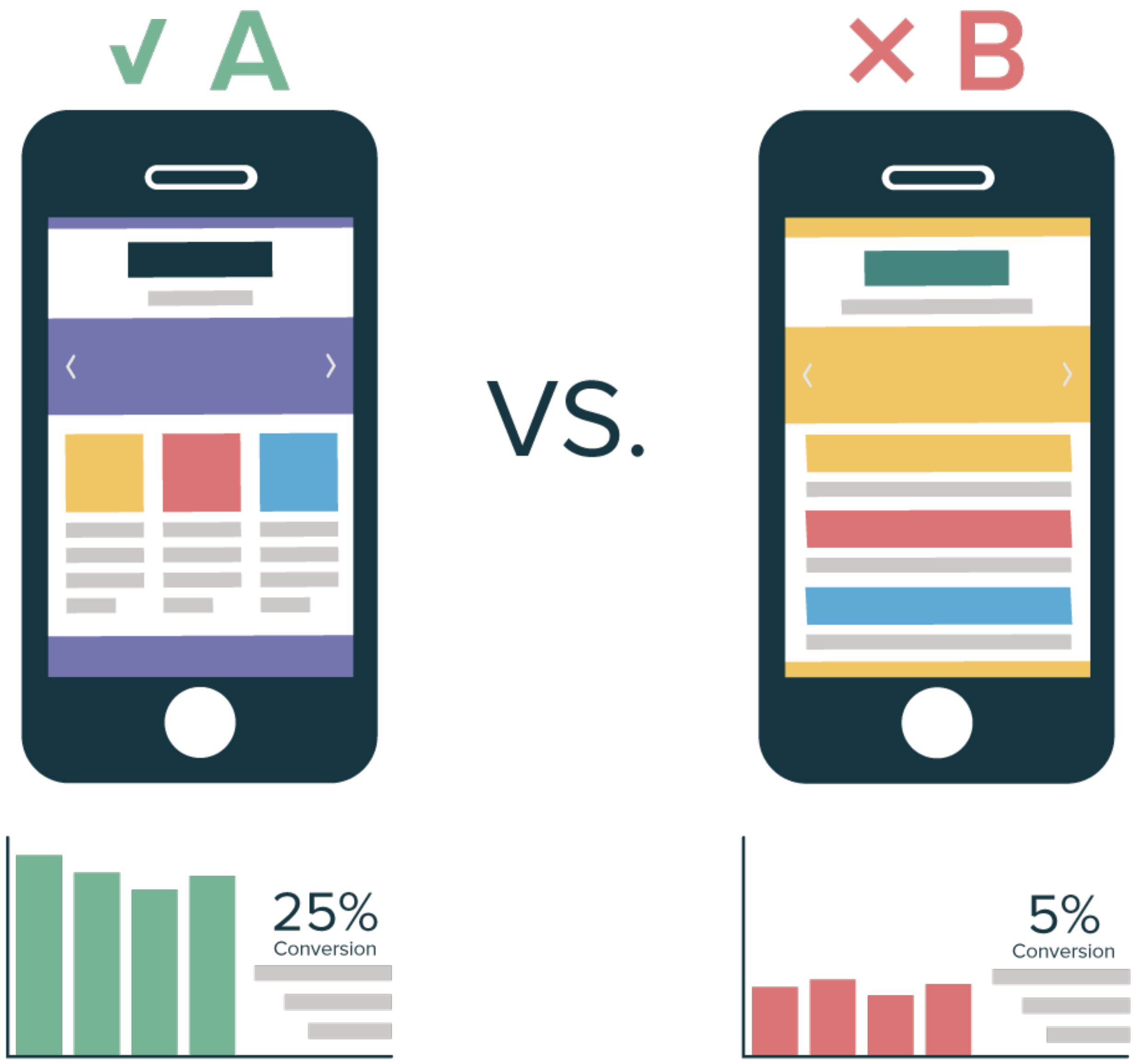
V&V: Product Testing Phase

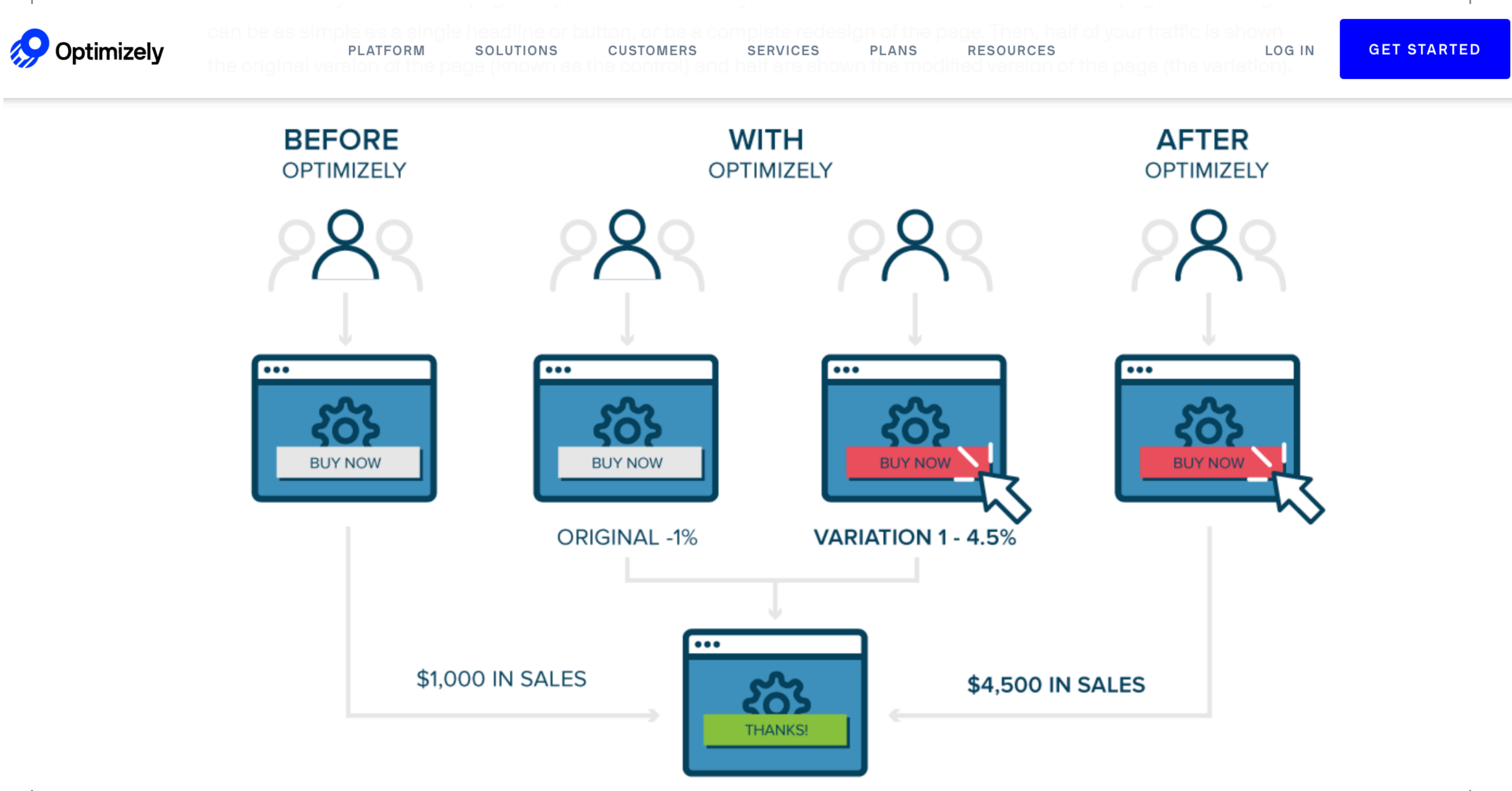
A/B Testing: Multiple versions of a product are deployed to different users without their awareness to gain insight as to which design choices are better received.

Continuous Experimentation: Continuously making incremental changes to a product, monitoring and collecting user data, which informs future changes.

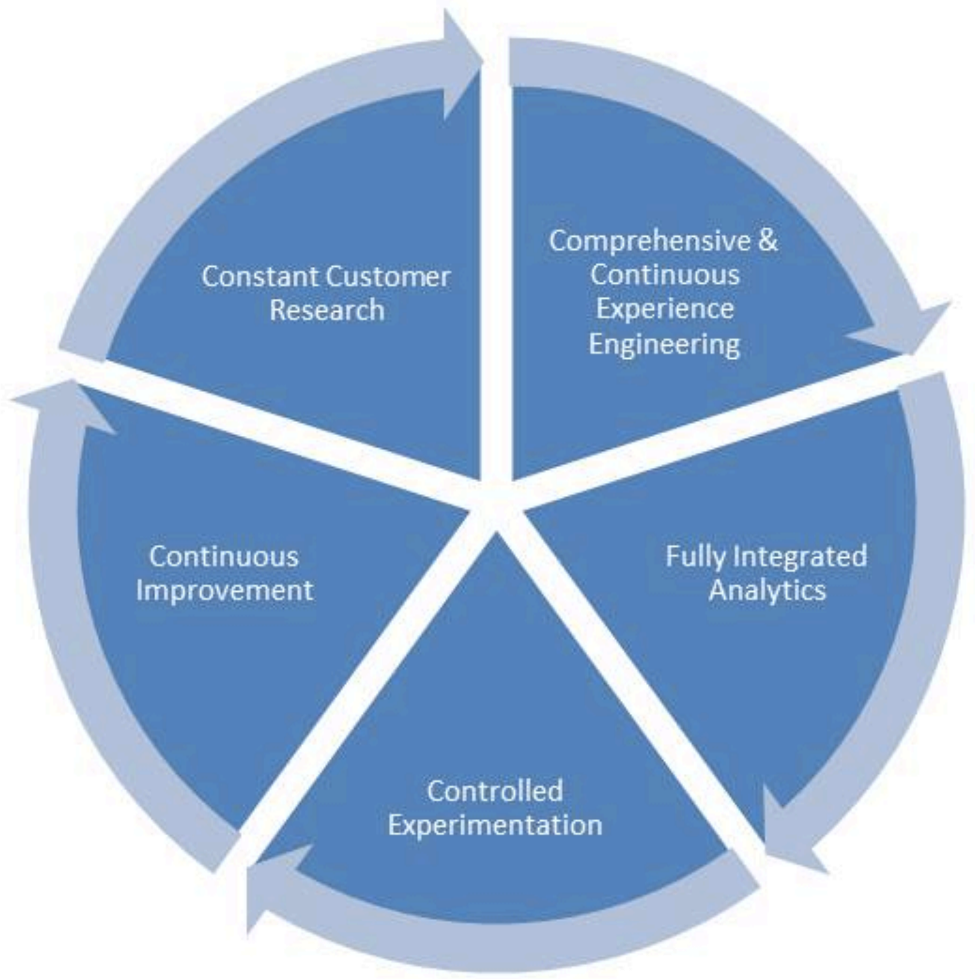
A/B Testing falls under Continuous Experimentation.

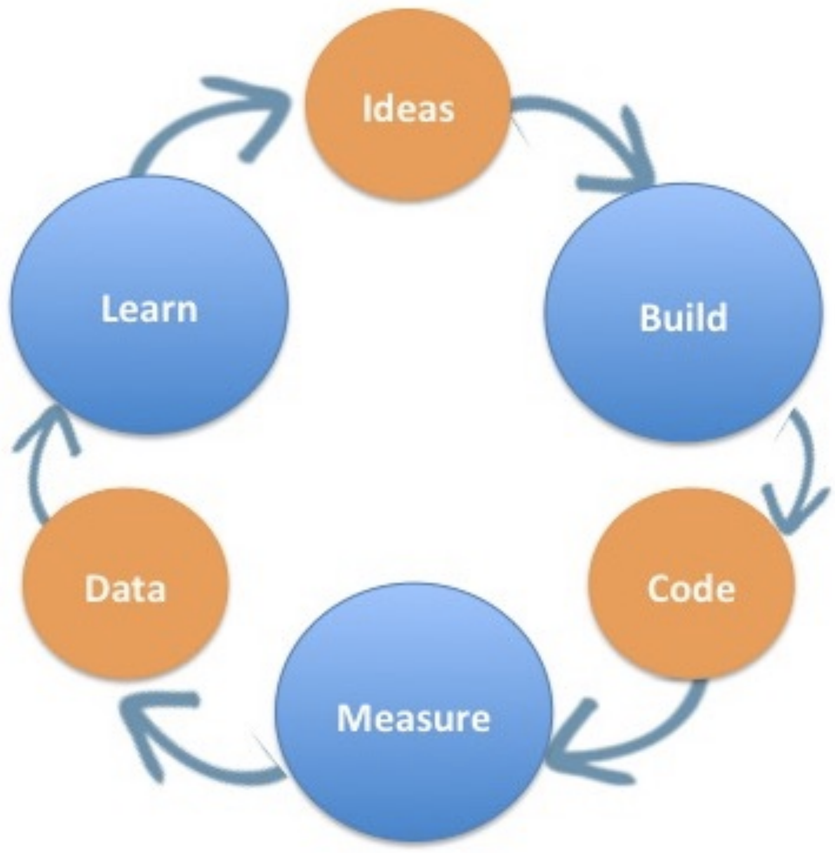
A/B Testing




can be as simple as a single headline or button, or be a complete redesign of the page. Then, half of your traffic is shown the original version of the page (known as the control) and half are shown the modified version of the page (the variation).

Continuous Experimentation


Constant Customer Research
Comprehensive & Continuous Experience Engineering
Fully Integrated Analytics
Controlled Experimentation
Continuous Improvement



V&V: Software Acceptance and Delivery Phase

Test, analyze, inspect and demonstrate that the developed system meets its functional and non-functional requirements in house (**product** testing) and in the operational environment (**acceptance** testing)

Locate, correct and re-test nonconformances (regression testing)

References

For more information on Continuous Experimentation: See "Continuous Verification/Continuous Testing", "Continuous Run-Time Monitoring", and "Continuous Innovation" in:
https://dl.acm.org/citation.cfm?doid=2593812.2593813

For more information on A/B Testing, Google or read more at:
https://www.optimizely.com/optimization-glossary/ab-testing/

A study done mapping Continuous Experimentation and A/B Testing:
https://dl.acm.org/citation.cfm?id=3194766