Facultad de
ESTADÍSTICA E INFORMÁTICA

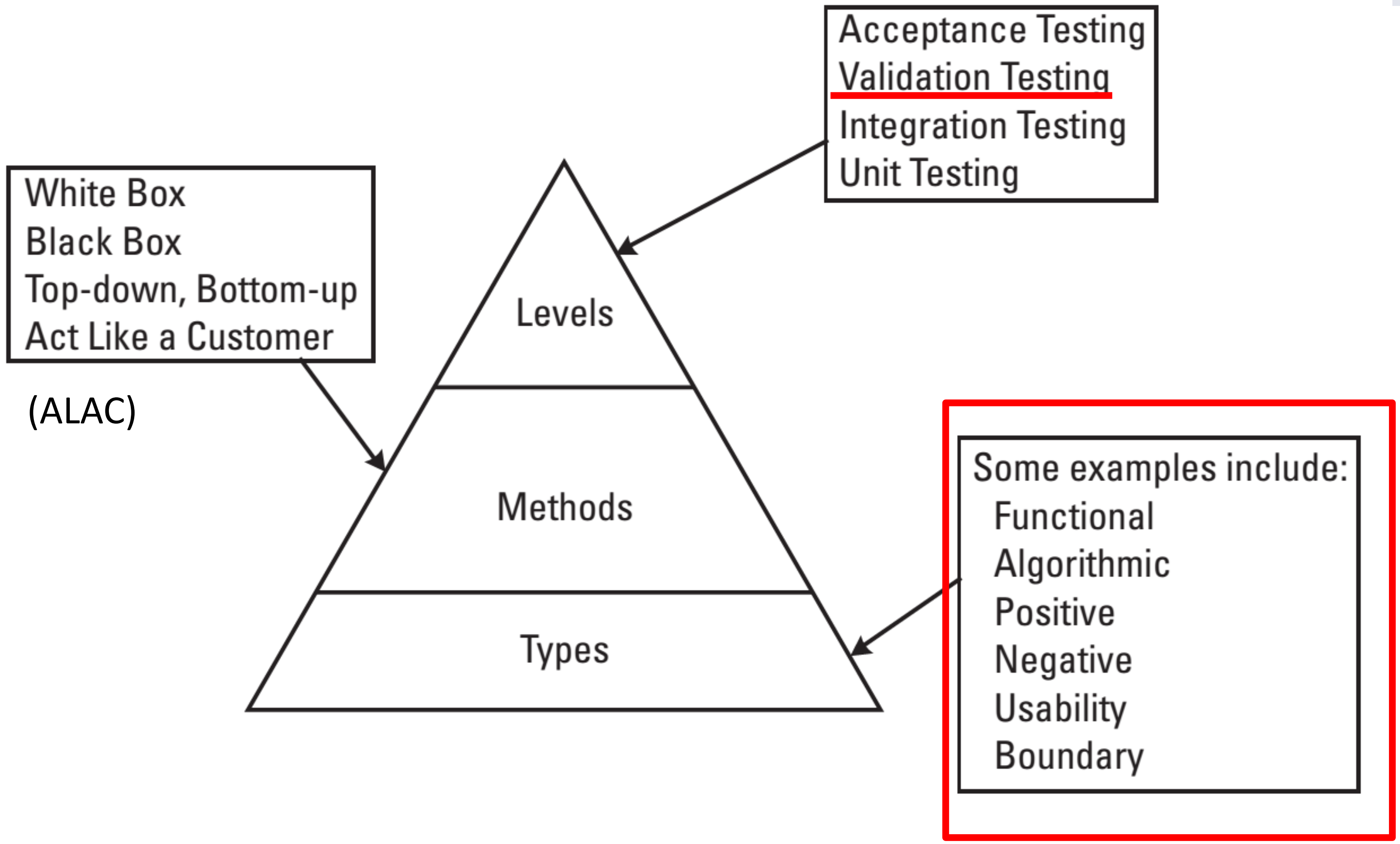
LIS
Licenciatura en Ingeniería Software

V & V

Validation techniques

Dra. Lizbeth Alejandra Hernández González
lizhernandez@uv.mx

Testing hierarchy



Validation

- “The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements”.

[IEEE Standard 610.12-1990,

Standard Glossary of Software Engineering Terminology]

- **verification and validation (V&V)** ***1.*** process of determining whether the requirements for a system or component are complete and correct, **the products of each development phase fulfill the requirements or conditions imposed by the previous phase**, and the final system or component complies with specified requirements *cf.* independent verification and validation.
- EVALUATION. ***1.*** systematic determination of the extent to which an entity meets its specified criteria.
- VALIDATION. ***1.*** **confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled**. ***5.*** the assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers.
- VERIFICATION. 1. **confirmation, through the provision of objective evidence, that specified requirements have been fulfilled**. ***3.*** process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.

Validation testing

- The objectives of validation testing are to determine if the software meets all of its requirements as defined in the SRS.

- Regression testing entails selectively rerunning validation tests to ensure that bug fixes have not introduced new bugs.

- Some organizations actively involve customers in testing by providing prerelease software for customer evaluation through what is commonly called alpha and beta testing.

- Acceptance testing is similar to validation testing, with one difference—the customer is actively involved.
  - Can be a repeat of (or subset of) the same tests used for validation testing or can employ tests developed entirely by customers.
  - Regression testing is performed to determine if the software still meets all of its requirements in light of changes and modifications made to the software.
  - Involves selectively repeating existing validation tests, not developing new tests.
- Is important that the objectives of these testing activities be precisely defined to minimize confusion and maximize return on investment.

Facultad de ESTADÍSTICA E INFORMÁTICA

LIS Licenciatura en Ingeniería Software

Alpha and Beta Testing

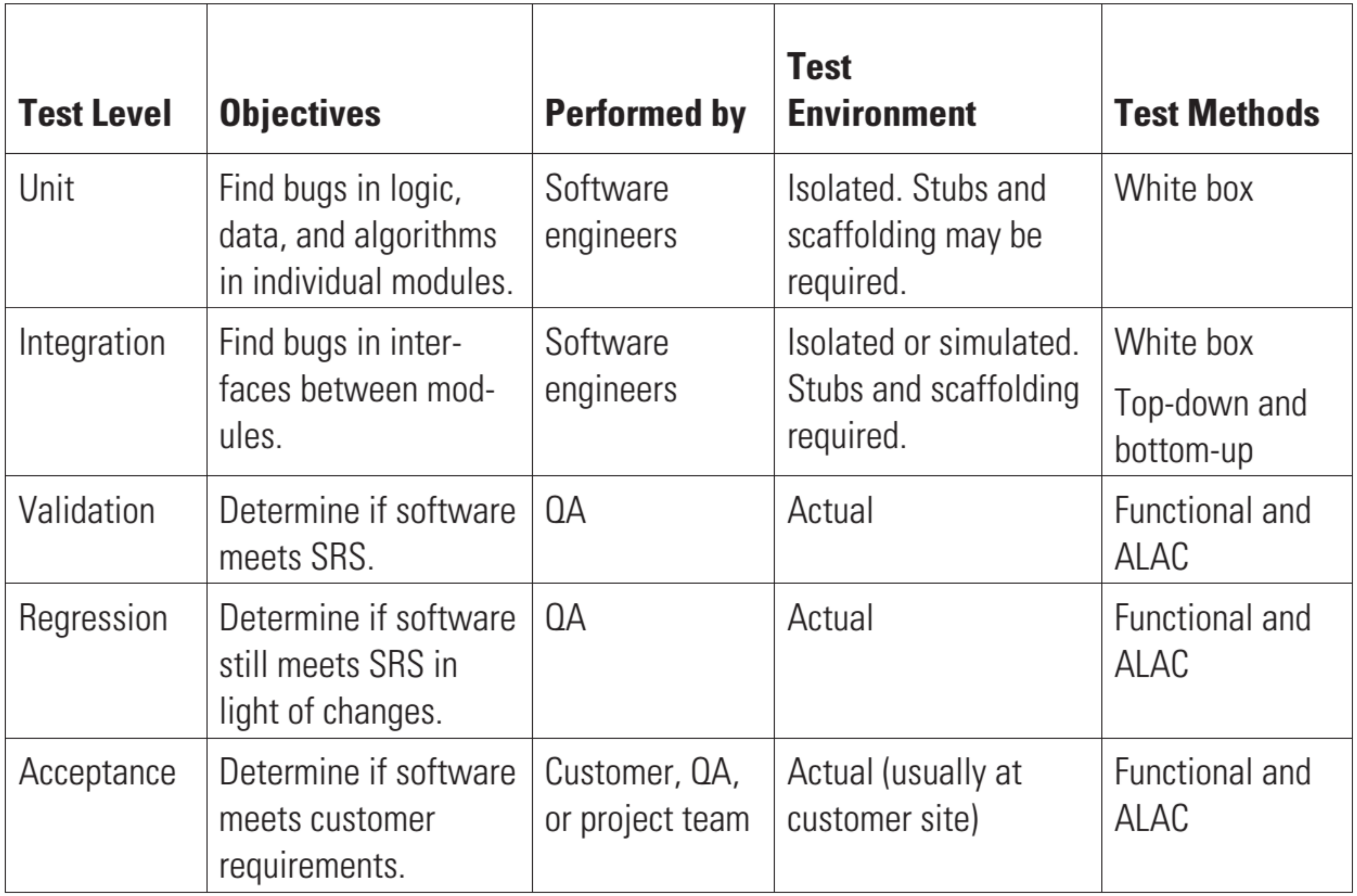
1. For alpha and beta testing to be most effective, you should provide your customers with an outline of the things that you would like them to focus on and specific test scenarios for them to execute.

   In this way, you can get very specific and valuable feedback from key customers.

2. Provide customers who are actively involved in your alpha and beta testing program with a commitment to fix defects that they find when they perform the test scenarios you identify for them.

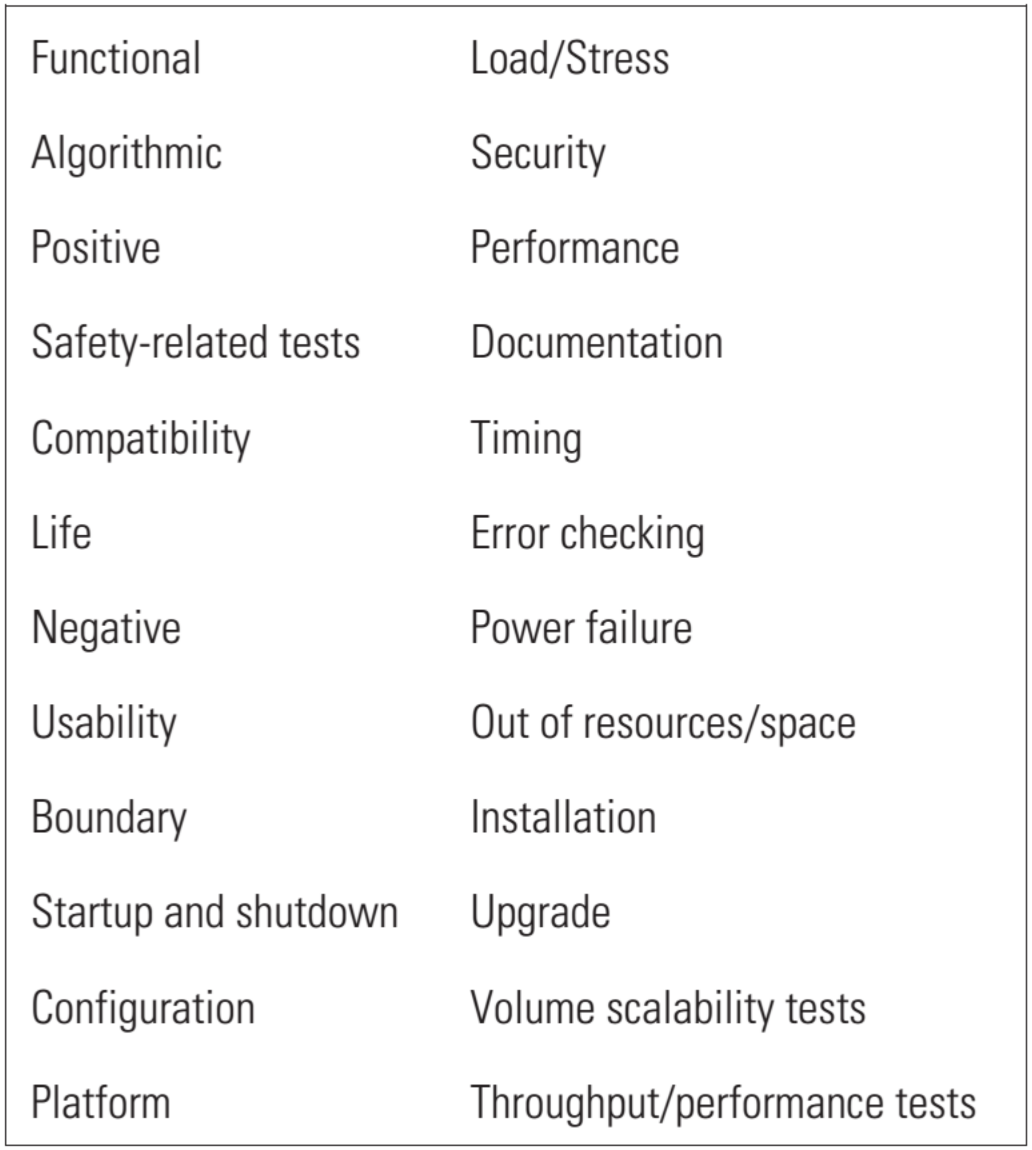
   This provides the motivation customers need to commit resources to alpha and beta testing your software.

Testing Levels and Test Methods



Test Level	Objectives	Performed by	Test Environment	Test Methods
Unit	Find bugs in logic, data, and algorithms in individual modules.	Software engineers	Isolated. Stubs and scaffolding may be required.	White box
Integration	Find bugs in inter-faces between mod-ules.	Software engineers	Isolated or simulated. Stubs and scaffolding required.	White box Top-down and bottom-up
Validation	Determine if software meets SRS.	QA	Actual	Functional and ALAC
Regression	Determine if software still meets SRS in light of changes.	QA	Actual	Functional and ALAC
Acceptance	Determine if software meets customer requirements.	Customer, QA, or project team	Actual (usually at customer site)	Functional and ALAC

Test types



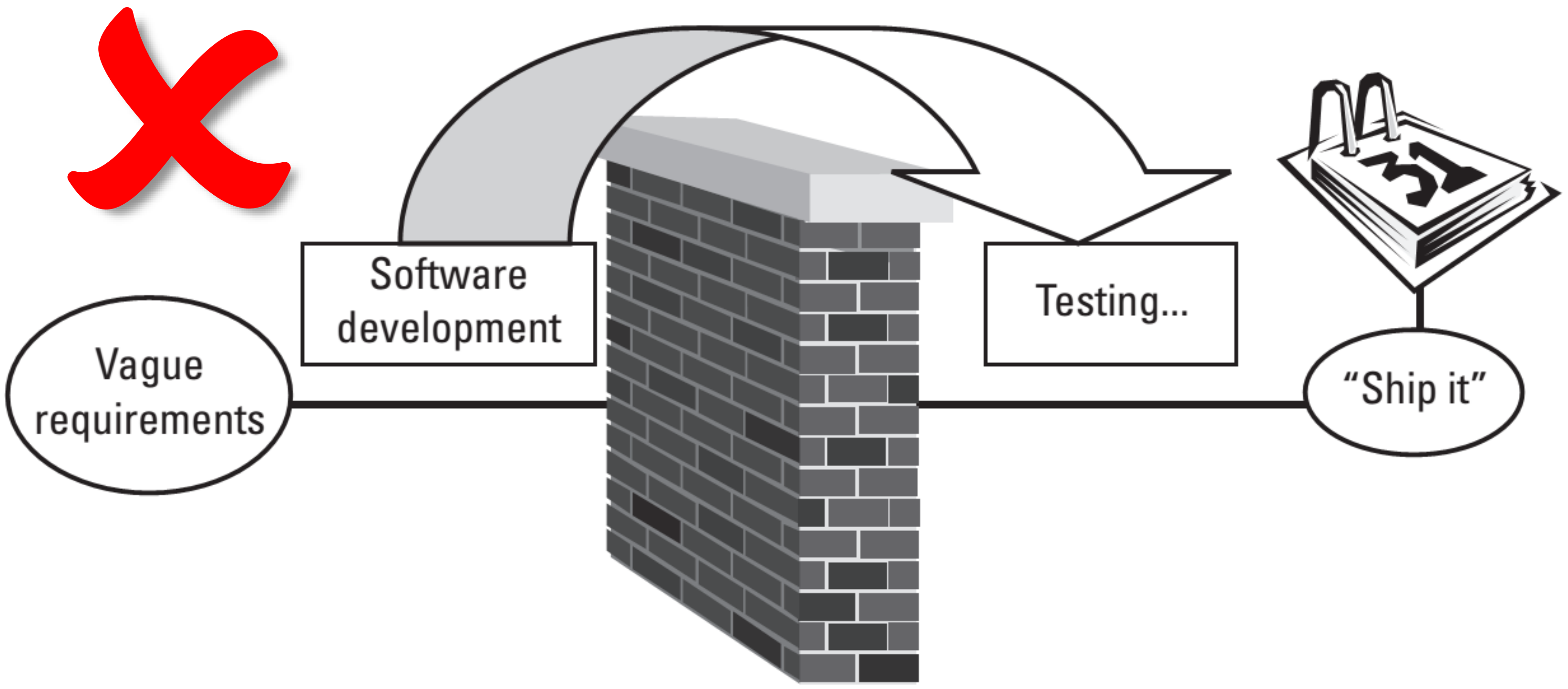
Functional	Load/Stress
Algorithmic	Security
Positive	Performance
Safety-related tests	Documentation
Compatibility	Timing
Life	Error checking
Negative	Power failure
Usability	Out of resources/space
Boundary	Installation
Startup and shutdown	Upgrade
Configuration	Volume scalability tests
Platform	Throughput/performance tests

Some of the more common test types are:

- ***Functional.***
  - Tests designed to determine if specific functions/features work as specified.
- ***Algorithmic.***
  - Tests designed to determine if specific algorithms have been implemented correctly.
- ***Positive tests.***
  - Tests designed to determine if a feature produces results that are consistent with the stated requirements when the software is used properly.
- ***Negative tests.***
  - Tests designed to determine if the software behaves reasonably when faced with invalid inputs or unexpected operator actions.
- ***Usability tests*****.**
  - Tests that exercise specific user interface features in order to determine if the software behaves as would be expected by trained/untrained users.

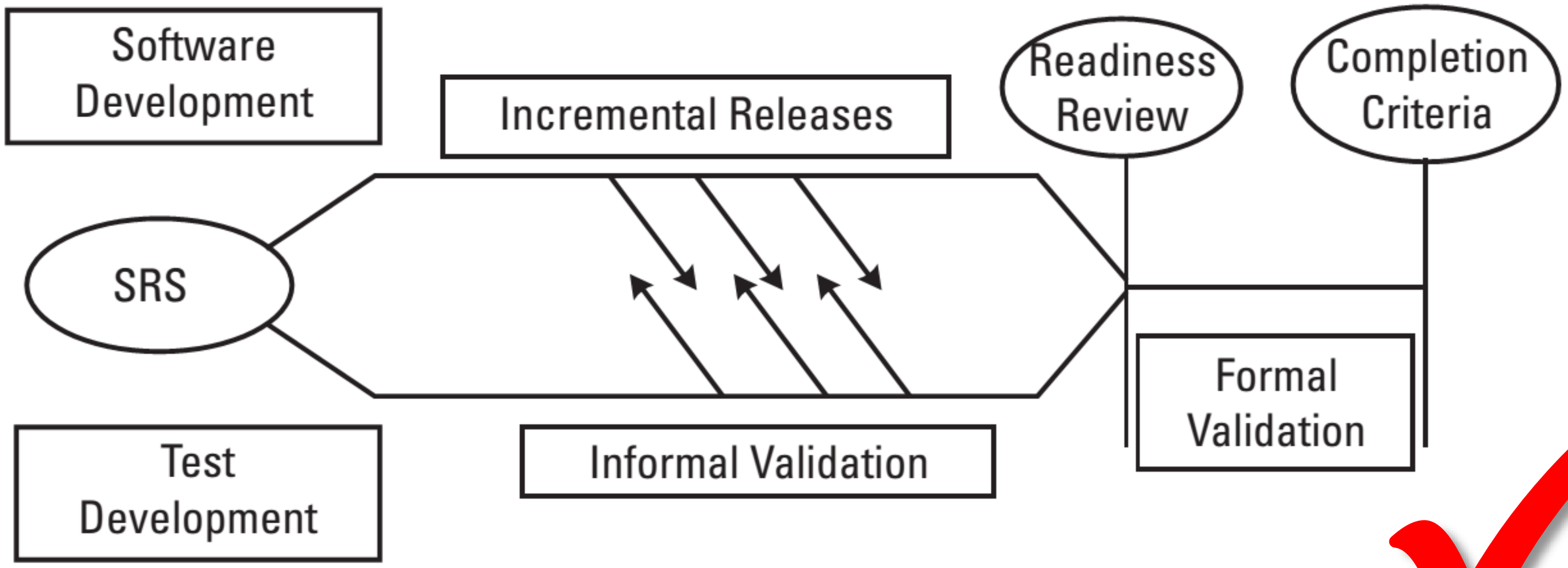
- ***Boundary tests.***
  - Tests that exercise specific limitations of the product, such as minimum and maximum values, to determine if the software behaves reasonably.
- ***Startup/shutdown tests.***
  - Tests designed to determine if startup and shutdown functions have been implemented correctly.
- ***Configuration tests.***
  - Tests designed to determine if a feature produces results that are consistent with the stated requirements when the software is used properly.
- ***Platform tests.***
  - Tests designed to determine if the software works properly on all supported platforms/operating systems.
- ***Load/stress tests.***
  - Tests that exercise the product under stated or expected load conditions.

Concurrent Development / Validation Testing Model



Typical validation testing process.

Concurrent Development /Validation Testing Model V&V



Informal validation

- Provides an opportunity for validation tests to be developed and debugged early in the software development process;
- Provides early feedback to software engineers;

Results in formal validation being less eventful, since most of the problems have already been found and fixed.

Once coding is completed and most all of the validation tests have been run at least once, the project team is ready for the validation readiness review.

Validation Readiness Review

- The purpose of this review is to ensure that everything is in place before beginning formal validation.

- Main differences between informal validation and formal validation :
  - During **informal validation** developers can make any changes needed in order to comply with the SRS.
  - During **informal validation** QA runs tests and makes changes as necessary in order for the tests to comply with the SRS.

  - During **formal validation** the only changes that can be made to the code are bug fixes in response to bugs reported during formal validation testing.
    - No new features can be added at this time.
  - During **formal validation** the same set of tests run during informal validation is run again.
    - No new tests are added.

- **Configuration management** is essential for increasing the effectiveness of validation testing.

- The test plan should define the criteria that should be met before formal validation can begin.

Example criteria

1. Software development is completed.
2. The test plan has been reviewed, approved, and is under document control.
3. **A requirements inspection has been performed on the SRS.**
4. **Design inspections have been performed on the SDDs.**
5. **Code inspections have been performed on all “critical modules.”**
6. All test scripts are completed and the software validation test procedure document has been reviewed, approved, and placed under document control.
7. Selected test scripts have been reviewed.
8. All test scripts have been executed at least once.
9. **CM tools are in place and all source code is under configuration control.**
10. Software problem reporting procedures are in place.
11. Validation testing completion criteria have been developed, reviewed, and approved.

Formal Validation

- At this point in the project, software changes are restricted to changes required to fix bugs.

- No new functionality can be added.

Activities

1. The same tests that were run during informal validation are executed again and results recorded.
2. Software Problem Reports (SPRs) are submitted for each test that fails.
3. SPR tracking is performed and includes the status of all SPRs
    1. (i.e., open, fixed, verified, deferred, not a bug).
4. For each bug that is fixed, the SPR identifies the modules that were changed to fix the bug.
5. Baseline change assessment is used to ensure that only those modules that should have changed actually have changed and that no new features have slipped in.

6. Informal code reviews (not formal inspections) are selectively conducted on changed modules to ensure that new bugs are not being introduced.
7. Time required to find and fix bugs (*find-fix cycle time*) is tracked.
8. Regression testing is performed using the following guidelines:
   a. Use complexity measures to help determine which modules may need additional testing.
   b. Use judgment in deciding which tests need to be rerun.
   c. Base decision on knowledge of software design and past history.
9. Track test status

   (i.e., passed, failed, or not run).
10. Record cumulative test time

    (cumulative hours of actual testing) for software reliability growth tracking .

Completion criteria

- A key element of an effective testing effort is knowing when to stop testing.

Examples of completion criteria:

- All test scripts have been executed.
- All SPRs have been satisfactorily resolved
  - bugs being fixed, deferred to a later release, determined not to be bugs, and so on.
  - All parties must agree to the resolution.
  - This criterion could be further defined to state that all high-priority bugs must be fixed, while lower-priority bugs can be handled on a case by-case basis.
- All changes made as a result of SPRs have been tested.
- All documentation associated with the software have been updated to reflect changes made during validation testing.
- The test report has been reviewed and approved.

Bibliografía

- Steven R. Rakitin (2001), *Software Verification and Validation for Practitioners and Managers,* Artech House