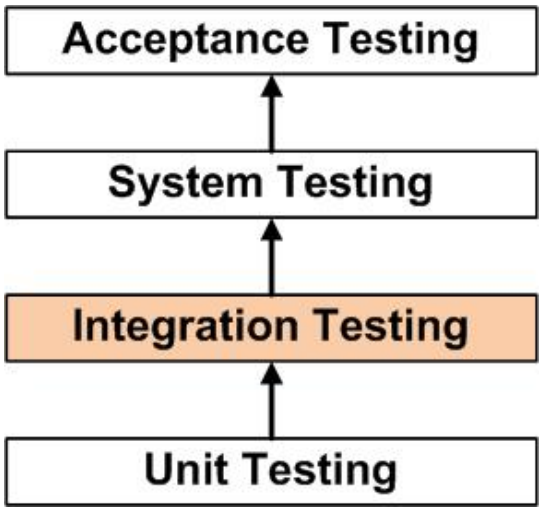
Integration Testing

INTEGRATION TESTING is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.



Definition by ISTQB

- **integration testing:** Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems. See also c*omponent integration testing, system integration testing*.
- **component integration testing:** Testing performed to expose defects in the interfaces and interaction between integrated components.
- **system integration testing:** Testing the integration of systems and packages; testing interfaces to external organizations (e.g. Electronic Data Interchange, Internet).

Analogy

During the process of manufacturing a ballpoint pen, the cap, the body, the tail and clip, the ink cartridge and the ballpoint are produced separately and unit tested separately. When two or more units are ready, they are assembled and Integration Testing is performed. For example, whether the cap fits into the body or not.

Method

Any of Black Box Testing (http://softwaretestingfundamentals.com/black-box-testing/), White Box Testing (http://softwaretestingfundamentals.com/white-box-testing/) and Gray Box Testing (http://softwaretestingfundamentals.com/gray-box-testing/) methods can be used. Normally, the method depends on your definition of 'unit'.

Tasks

- Integration Test Plan
  - Prepare
  - Review
  - Rework
  - Baseline
- Integration Test Cases/Scripts
  - Prepare
  - Review
  - Rework
  - Baseline
- Integration Test
  - Perform

When is Integration Testing performed?

Integration Testing is the second level of testing (http://softwaretestingfundamentals.com/software-testing-levels/) performed after Unit Testing (http://softwaretestingfundamentals.com/unit-testing/) and before System Testing (http://softwaretestingfundamentals.com/system-testing/).

Who performs Integration Testing?

Developers themselves or independent testers perform Integration Testing.

Approaches

- *Big Bang* is an approach to Integration Testing where all or most of the units are combined together and tested at one go. This approach is taken when the testing team receives the entire software in a bundle. So what is the difference between Big Bang Integration Testing and System Testing? Well, the former tests only the interactions between the units while the latter tests the entire system.
- *Top Down* is an approach to Integration Testing where top-level units are tested first and lower level units are tested step by step after that. This approach is taken when top-down

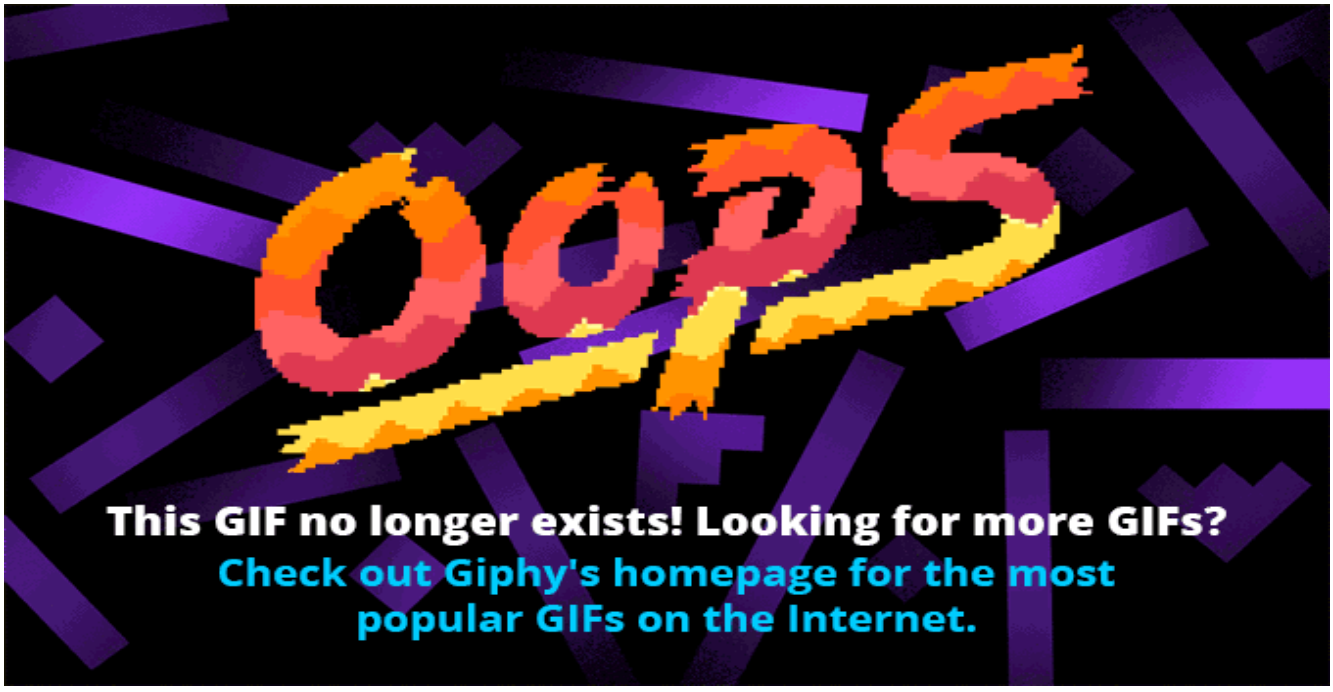
development approach is followed. Test Stubs are needed to simulate lower level units which may not be available during the initial phases.

- *Bottom Up* is an approach to Integration Testing where bottom level units are tested first and upper-level units step by step after that. This approach is taken when bottom-up development approach is followed. Test Drivers are needed to simulate higher level units which may not be available during the initial phases.
- *Sandwich/Hybrid* is an approach to Integration Testing which is a combination of Top Down and Bottom Up approaches.

Tips

- Ensure that you have a proper Detail Design document where interactions between each unit are clearly defined. In fact, you will not be able to perform Integration Testing without this information.
- Ensure that you have a robust Software Configuration Management system in place. Or else, you will have a tough time tracking the right version of each unit, especially if the number of units to be integrated is huge.
- Make sure that each unit is unit tested before you start Integration Testing.
- As far as possible, automate your tests, especially when you use the Top Down or Bottom Up approach, since regression testing is important each time you integrate a unit, and manual regression testing can be inefficient.

Lack of Integration Test



‹ System Testing (http://softwaretestingfundamentals.com/system-testing/)

Unit Testing › (http://softwaretestingfundamentals.com/unit-testing/)

Search...

Enter your email address:

Subscribe

If you do not find the verification email, check your spam folder

Delivered by FeedBurner (https://feedburner.google.com)

Categories

- Artifacts (http://softwaretestingfundamentals.com/category/artifacts/)
- Basics (http://softwaretestingfundamentals.com/category/basics/)
- Career (http://softwaretestingfundamentals.com/category/career/)
- Defects (http://softwaretestingfundamentals.com/category/defects/)
- Levels (http://softwaretestingfundamentals.com/category/levels/)
- Methods (http://softwaretestingfundamentals.com/category/methods/)
- Metrics (http://softwaretestingfundamentals.com/category/metrics/)
- Resources (http://softwaretestingfundamentals.com/category/resources/)
- Types (http://softwaretestingfundamentals.com/category/types/)

Skim

Disclaimer (http://softwaretestingfundamentals.com/disclaimer/)

Privacy Policy (http://softwaretestingfundamentals.com/privacy-policy/)

Engage

Facebook Page (https://www.facebook.com/softwareTestingFundamentals/)

Advertise (http://softwaretestingfundamentals.com/advertise/)

Contact (http://softwaretestingfundamentals.com/contact/)

Ponder

Weinberg's Second Law: If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would have destroyed civilization.