

Testing any software is essential, but is an activity that is sometimes overlooked, often badly planned and managed and almost always under resourced. Formal testing requires developing a test plan, establishing a test lab, staffing, and overcoming corporate challenges and hurdles.

A test strategy helps focus on how testing will be conducted by laying out the environments, roles and responsibilities of all parties, not just the testers. Comprehensive defect management, with a well defined procedure which can be followed by both developers and testers goes a long way to help alleviate mistakes often introduced by poor management of the testing process.

Traditional software development follows what is known as the waterfall method, where each step follows the next, as if in a waterfall. As each step in the process is completed a set of outputs are produced from which test specifications can be created and allowing the next step in the process to proceed.

Requirements

The client, with the help of a Business Analyst, details his requirements as to what the system shall achieve. The specification can cover both functional and non-functional requirements.

Functional Specification

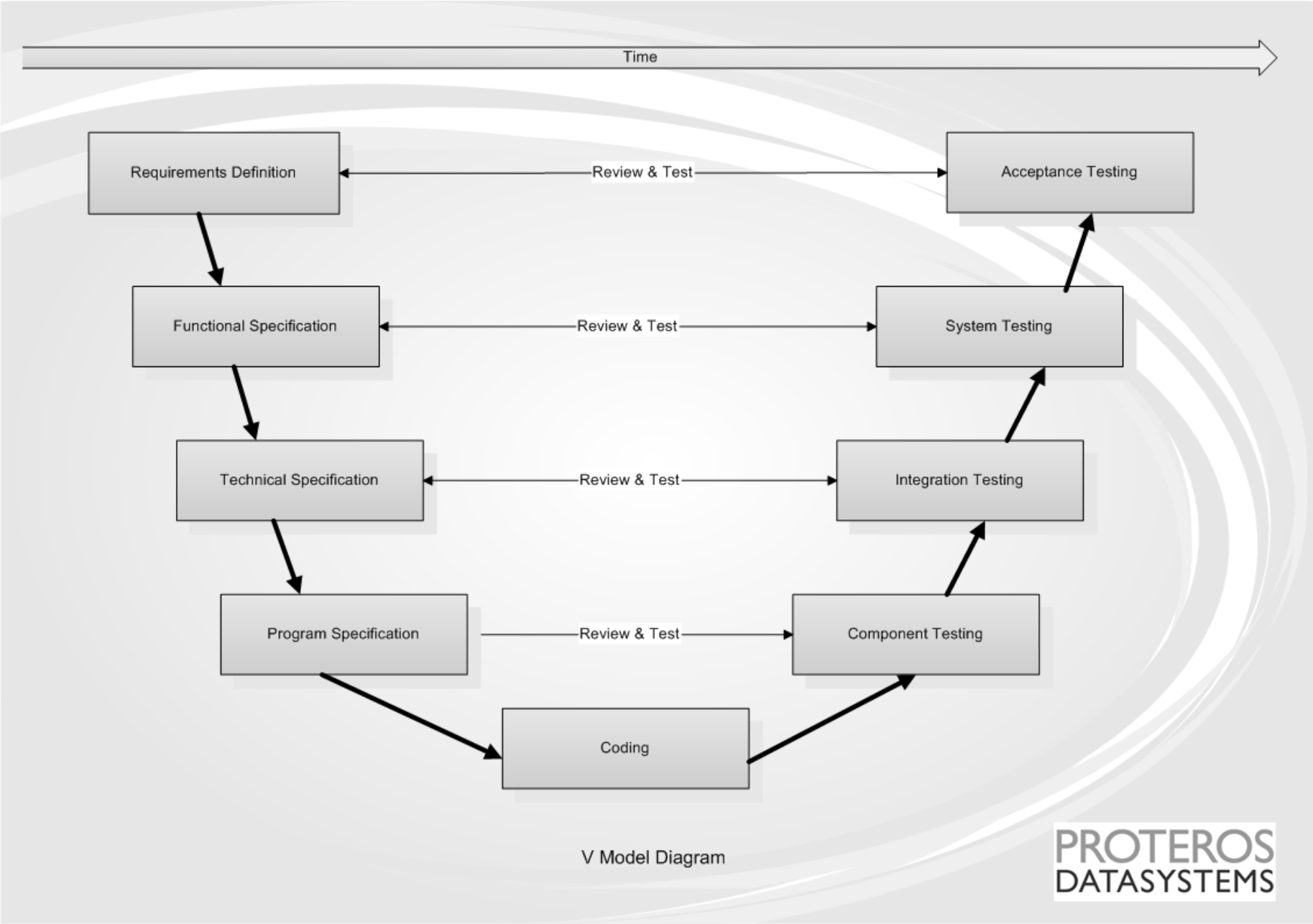
The requirements are then passed to the Systems Analysts to formalise into a functional specification, in computing terms, which then is reviewed and signed-off by the client.

Technical Specification

The Senior Programmer then takes the functional specification and defines what the system shall achieve and how it will be achieved. It details the features required, maps them to various components and defines the relationships between these components.

Program Specification

Each component then has a program specification which describes, in detail, exactly how it will perform its piece of processing.



Component Testing

component Testing involves checking that each feature specified in the program specification has been implemented in the particular unit. In theory, an independent tester should do this, but in practice the developer usually does.

Interface Testing

As each of the units/components are developed and tested they are then linked together to check if they work with each other and tests are organised to check all the interfaces, until all the components have been built and interfaced to each other producing the whole system, which is tested against the technical specification.

System Testing

Once the entire system has been built and all interfaces have been verified, it then has to be tested against the functional specification to check if it delivers the features required.

Acceptance Testing

Acceptance Testing checks the system against the business requirements, it is similar to systems testing, in that the whole system is checked but the important difference is the change in focus. Systems testing checks that the system that was specified has been delivered. Acceptance Testing checks that the system delivers what was requested.

Regression Tests

Once the system has been verified and the changes have been applied it is important to ensure that subsequent changes do not adversely affect the changes that have just been tested and signed-off.

To guard against this, regression testing must be performed, usually at the end of system testing, to ensure that the parts of the system that have not changed continue to work as originally specified and that the only changes that can be detected are due to the current ones.