In agile and spiral software development methodologies, software products are developed using incremental integration so that new features are added to the existing software products continuously. This process poses a grave risk because the existing functionalities of the product may be broken in the new versions. This happens because when new features are added to the existing software product, many times, the existing source code will be changed to accommodate the new features. In fact, sometimes a lot of existing source codes are rewritten to create a new version because the old source codes were not scalable to allow the addition of new features. Similarly, any old source code can be refactored to make it scalable for the addition of new features. This results in changes to the old (i.e., existing) source code of the product. In these scenarios, once the new features are added, many old features may not work at all or may work very differently from what they are supposed to do. It is important that the old features continue to function in the same way for the existing customers of the software product. Because of this, during software testing, old features are also tested when the new software product is released or when new features are added to the existing product. This kind of testing is known as regression testing.

When we think of software testing, we often forget that it is the most profound activity in software projects. Each line of source code must be tested to determine if it is syntactically correct and the source code compiles cleanly. Otherwise, a compilation error will be raised. In addition, the source code should be checked for functional correctness. Developers should always ask themselves these questions: Is the source code doing the calculations right? Is the source code of a software unit integrated perfectly with the other units? Is a code module of the software product integrated perfectly with the other modules? Is the entire system working as per the requirement specifications? Here, we see that the software needs to be tested at various levels, including when the source code is written.

As shown in Figure 9.3, software testing can be done at many levels. Different types of tests are conducted at different levels. Unit testing, integration testing, system testing, and user acceptance testing are all part of "levels of software testing" (validation) and they are discussed later in this chapter.

9.5.1 V Model

The V model is another way of looking at the levels of testing. It connects the artifacts that are developed during a software development project with the level of testing that

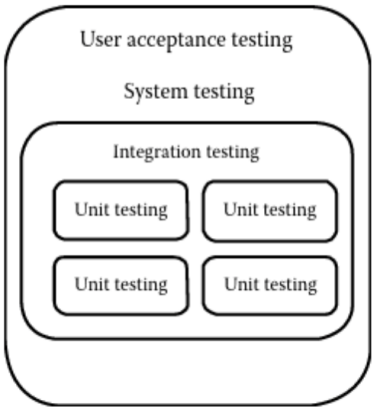


Figure 9.3 Levels of software testing.

is performed on those artifacts. The V model is used to capture the essence of system testing and user acceptance testing. Different levels and types of testing that are carried out during these testing activities are best depicted by the V model developed by Barry Boehm.

The V model relates the major phases in the software development life cycle with the validation that needs to be done on the artifacts that are developed in those major phases. Unfortunately, the V model does not cover the verification part of testing.

The V model is not based on time-phased life cycle activities. Instead, it only relates the validation activities to be done on a phase after that phase is completed. Figure 9.4 depicts the V model. This figure indicates that user acceptance testing (the last activity in the software development life cycle) is done based on the requirement specifications because user acceptance testing and requirement specification are at the same level in the figure. As shown in the figure, system testing is done based on the software design because these two are also at the same level. Similarly, the other activities that are at the same level are associated with each other.

If we observe closely, we can see that the validation activities maintain a reverse chronological relation with the phases of the software development life cycle. In Figure 9.4, you can see two long arrows, one of which is pointing down and the other pointing up. These arrows indicate the chronological order in which the phases and validation activities are carried out in the software development life cycle. Even here, this chronological order is not followed strictly for unit and integration testing because, in this figure, unit testing occurs after integration testing. Still, the V model

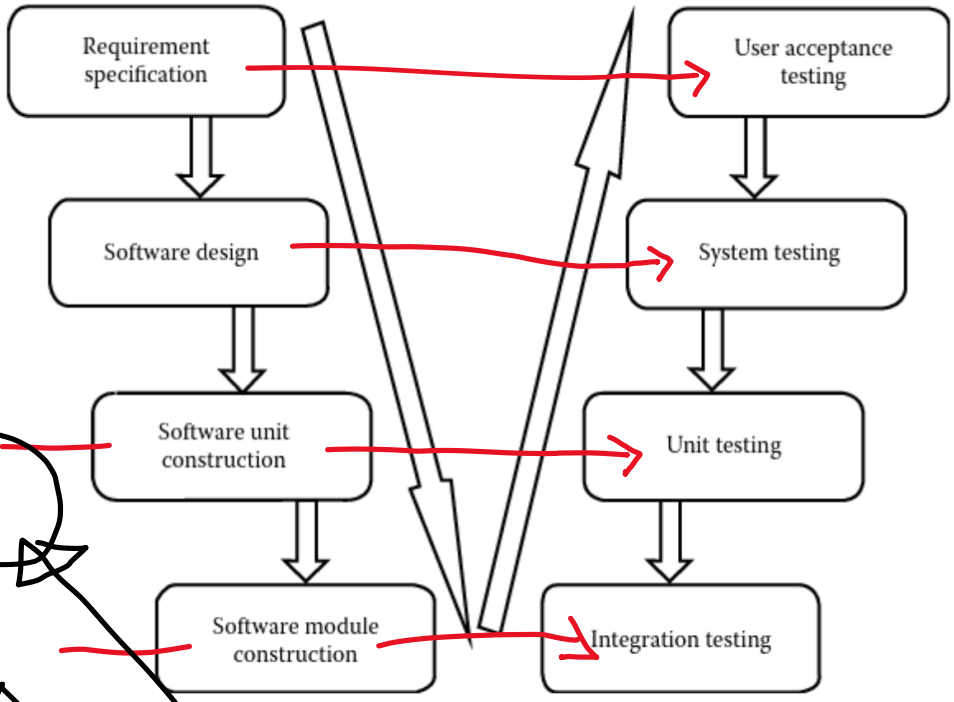
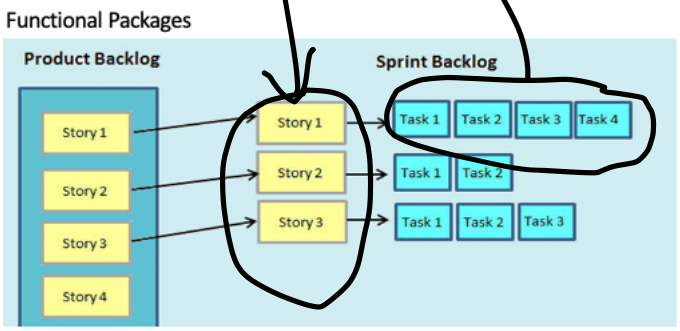


Figure 9.4 V model.

Ahmed, Ashfaque, and Bhanu Prasad. *Foundations of Software Engineering*, Auerbach Publishers, Incorporated, 2016.

ProQuest Ebook Central, https://ebookcentral.proquest.com/lib/monash/detail.action?docID=5165107.



Quality = User acceptance criteria (eg what the client expects from the product)

Reliability = does the product meet the standards for operations – AND what the client expects

- Quality examples – Abdullah likes a quality coffee from “Black Velvet” in Exhibition street Melbourne. Often regarded as the finest coffee in Australia

YET

Phil thinks 7 – 11 (a convenience store) makes coffee that is OK and the “quality” is fine.

Remember QUALITY is in the eye of the beholder. For your IE project you need to ask and understand the clients “Acceptable minimum standard”

- Reliability standards; in a landing page of a website, what do you expect the traffic numbers to be expected? If you design too low, the landing page will crash making the website “unreliable” and not trusted. Too much allowance for traffic may create a large cost and time requirement, which is not needed.