User-Interface Testing

Testing systems that include user interfaces (UTs) can be challenging. This remains true for most types of test objectives whether it be verifying that a UI meets its requirements or checking that a user interface meets key nonfunctional metrics, for example, usability and performance [37, 41]. To this end, we discuss some of the issues that are encountered during user-interface testing and how the UTP can be used to address some of these concerns.

9.1 Issues in User-Interface Testing

Most systems include at least one UI. Yet testing systems with these interfaces remains a challenge for test automation. A UI can encompass many different forms of media, for example physical (i.e., buttons/switches), graphical interfaces, speech, audio, biometrics. This variation introduces a number of issues. For example, the level of abstraction that is used to control and observe a graphical UI will depend on the level of integration between the test system and the application and/or underlying graphics system. If no integration is provided, then test control and observation are based on the rendered inputs and outputs of the system, for example, captured images of the graphical interface. However, if some integration is provided, then test control and observation can be abstracted away from rendered images, thereby improving the resilience of tests and reducing test maintenance. For example, if the requirements for the rendered graphical interface are modified and if no abstract integration is provided for control and observation, then tests will have to be updated accordingly. Therefore, understanding the level of abstraction for specifying both requirements and tests is very important. In addition to abstraction, there are other issues that affect UI testing. For example, we may implement a system that supports different localization options, such as different languages.

Another consideration when testing UIs is the verification of nonfunctional requirements, such as usability and performance. For example, we may want
Tip 11 User-interface abstraction can reduce test maintenance

Abstracting the way in which user interfaces can be controlled and observed can improve the resilience of test specifications as well as reduce the effort required for test specification maintenance.

to enforce specific usability design principles, such as “when the user presses the ‘exit’ key they always return to the previous screen,” or measure the time the system takes in responding to a user request. Unfortunately, these aspects are often overlooked when system requirements are defined.

9.2 Planning UI Test Activities

In this section, we describe how we can approach test specification in a manner that attempts to address some of the issues presented above. For our example, we do this by defining a conceptual framework for test specification that separates the various concerns relating to UI testing to minimize the impact of changes and maintenance for test specification—see Figure 9.1.

In the following sections, we describe the different aspects illustrated in Figure 9.1.

![Fig. 9.1. Example of how different concerns can be separated during user-interface test specification](image-url)