

the associated application is called to provide user functionality. For example, when objects of type sound are accessed, the audio player is called to process them.

The main advantage of this approach is that a lot of application functionality can be implemented quickly at a very low cost. Users who are already familiar with the applications making up the system do not have to learn how to use new features. However, if they do not know how to use the applications, learning may be difficult, especially as they may be confused by application functionality that isn't necessary. There may also be performance problems with the application because of the need to switch from one application system to another. The switching overhead depends on the operating system support that is provided.

17.4 Software prototyping

As I discussed in the introduction to this chapter, there are some circumstances where, for practical or contractual reasons, an incremental software delivery process cannot be used. In those situations, a statement of the system requirements is completed and is used by the development team as the basis for the system software. As I explained, you can get some of the benefits of an incremental development process by creating a prototype of the software. This approach is sometimes called throw-away prototyping because the prototype is not delivered to the customer or maintained by the developer.

A prototype is an initial version of a software system that is used to demonstrate concepts, try out design options and, generally, to find out more about the problem and its possible solutions. Rapid, iterative development of the prototype is essential so that costs are controlled and system stakeholders can experiment with the prototype early in the software process.

A software prototype can be used in a software development process in several ways:

1. In the requirements engineering process, a prototype can help with the elicitation and validation of system requirements.
2. In the system design process, a prototype can be used to explore particular software solutions and to support user interface design.
3. In the testing process, a prototype can be used to run back-to-back tests with the system that will be delivered to the customer.

System prototypes allow users to see how well the system supports their work. They may get new ideas for requirements and find areas of strength and weakness in the software. They may then propose new system requirements. Furthermore, as the prototype is developed, it may reveal errors and omissions in the requirements

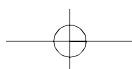
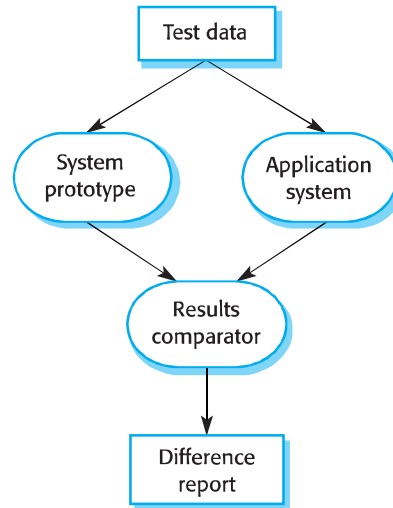


Figure 17.12 Back-to-back testing



that have been proposed. A function described in a specification may seem useful and well-defined. However, when that function is combined with other functions, users often find that their initial view was incorrect or incomplete. The system specification may then be modified to reflect their changed understanding of the requirements.

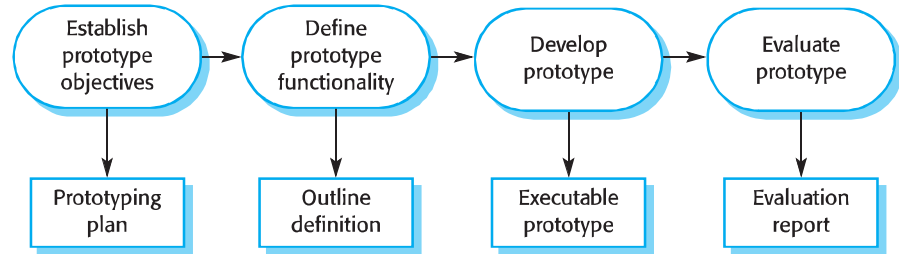
A system prototype may be used while the system is being designed to carry out design experiments to check the feasibility of a proposed design. For example, a database design may be prototyped and tested to check that it allows for the most efficient data access for the most common user queries. Prototyping is also an essential part of the user interface design process. Because of the dynamic nature of user interfaces, textual descriptions and diagrams are not good enough for expressing the user interface requirements. Therefore, rapid prototyping with end-user involvement is the only sensible way to develop graphical user interfaces for software systems.

A major problem in system testing is test validation where you have to check whether the results of a test are what you expect. When a system prototype is available, you can reduce the effort involved in result checking by running back-to-back tests (Figure 17.12). The same test cases are submitted to the prototype and to the system under test. If both systems give the same result, the test case has probably not detected a fault. If the results differ, it may mean that there is a system fault and the reasons for the difference should be investigated.

Finally, as well as supporting software process activities, prototypes can be used to reduce the time required to develop user documentation and to train users with the system. A working, albeit limited, system is available quickly to demonstrate the feasibility and usefulness of the application to management.

In a study of 39 prototyping projects, Gordon and Bieman (Gordon and Bieman, 1995) found that the benefits of using prototyping were:

Figure 17.13 The process of prototype development



1. Improved system usability
2. A closer match of the system to users' needs
3. Improved design quality
4. Improved maintainability
5. Reduced development effort

Their study suggests that the improvements in usability and better user requirements that stem from using a prototype do not necessarily mean an overall increase in system development costs. Prototyping usually increases costs in the early stages of the software process but reduces costs later in the development process. The main reason for this is that rework during development is avoided because customers request fewer system changes. However, Gordon and Bieman found that overall system performance is sometimes degraded if inefficient prototype code is reused.

A process model for prototype development is shown in Figure 17.13. The objectives of prototyping should be made explicit from the start of the process. These may be to develop a system to prototype the user interface, to develop a system to validate functional system requirements or to develop a system to demonstrate the feasibility of the application to management. The same prototype cannot meet all objectives. If the objectives are left unstated, management or end-users may misunderstand the function of the prototype. Consequently, they may not get the benefits that they expected from the prototype development.

The next stage in the process is to decide what to put into and, perhaps more importantly, what to leave out of the prototype system. To reduce prototyping costs and accelerate the delivery schedule, you may leave some functionality out of the prototype. You may decide to relax non-functional requirements such as response time and memory utilisation. Error handling and management may be ignored or may be rudimentary unless the objective of the prototype is to establish a user interface. Standards of reliability and program quality may be reduced.

The final stage of the process is prototype evaluation. Provision must be made during this stage for user training, and the prototype objectives should be used to derive a plan for evaluation. Users need time to become comfortable with a new system and to settle into a normal pattern of usage. Once they are using the system normally, they then discover requirements errors and omissions.

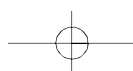
A general problem with developing an executable, throw-away prototype is that the mode of use of the prototype may not correspond with how the final delivered system is used. The tester of the prototype may not be typical of system users. The training time during prototype evaluation may be insufficient. If the prototype is slow, the evaluators may adjust their way of working and avoid those system features that have slow response times. When provided with better response in the final system, they may use it in a different way.

Developers are sometimes pressured by managers to deliver throw-away prototypes, particularly when there are delays in delivering the final version of the software. Rather than face up to delays in the project the manager may believe that delivering an incomplete or poor quality system is better than nothing. However, this is usually unwise for the following reasons:



KEY POINTS

- As pressure grows for the rapid delivery of software, an iterative approach to software development is becoming increasingly used as the standard development technique for small and medium-sized systems, especially in the business domain.
- Agile methods are iterative development methods that focus on incremental specification, design and system implementation. They involve the customer directly in the development process. Reducing development overhead can make faster software development possible.
- Extreme programming is a well-known agile method that integrates a range of good programming practices such as systematic testing, continuous software improvement and customer participation in the development team.
- A particular strength of extreme programming is the development of automated tests before a program feature is created. All tests must successfully execute when an increment is integrated into a system.
- Rapid application development involves using development environments that include powerful tools to support system production. These include database programming languages, form and report generators, and links to office applications.
- Throw-away prototyping is an iterative development process where a prototype system is used to explore the requirements and design options. This prototype is not intended for deployment by the system customer.
- When implementing a throw-away prototype, you first develop the parts of the system you understand least; by contrast, in an incremental development approach, you begin by developing the parts of the system you understand best.



This document was created with Win2PDF available at <http://www.win2pdf.com>.
The unregistered version of Win2PDF is for evaluation or non-commercial use only.
This page will not be added after purchasing Win2PDF.