1 THE IMPORTANCE OF UAT

The introduction presented some general concepts that help to get a better understanding of UAT and introduced examples of high-profile project failures that, if not caused by UAT, were certainly not prevented by it. Chapter 1 provides an overview of UAT, its purpose and its relationship to an implementation project and the people who take part in it. You will find out why UAT is different from other types of testing and yet often uses the same processes – one of which is the fundamental test process. Finally you will discover what the different types of UAT are, who the stakeholders of UAT are and what each role has to offer to and to gain from the UAT process.

Topics covered in this chapter

- What is UAT?
- Why test information systems?
- Business vulnerability
- The UAT process
- From UAT to service delivery
- UAT and contracts
- Stakeholders in UAT

WHAT IS UAT?

UAT stands for user acceptance testing and is commonly used to refer to the end-user software testing carried out prior to a new information system (IS) being introduced to an organisation. The primary objective of UAT is to ensure the new system does what it set out to do and meets the requirements the business has of it. Here is a definition of the term UAT taken from the ISTQB Glossary of Testing Terms:

> Formal testing with respect to user needs, requirements, and business processes, conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system.

THE INTERNATIONAL SOFTWARE TESTING QUALIFICATIONS BOARD (ISTQB)

ISTQB provides certification at Foundation, Advanced and Expert levels for software testers. The certificates are supported by a glossary available free from www.istqb.org/downloads/glossary.html. We will use the ISTQB Glossary throughout this book because it is widely used in the testing community and in tester certification programmes.

You may also find the following related books, which also use the ISTQB Glossary, helpful if you want to increase your knowledge of testing: *Software Testing: an ISTQB-ISEB Foundation Guide* by Brian Hambling (Ed), Peter Morgan, Angelina Samaroo, Geoff Thompson and Peter Williams; *Software Testing: An ISEB Intermediate Certificate* by Brian Hambling and Angelina Samaroo.

Three aspects of this definition are important and will drive what we do in preparing and implementing UAT:

1. UAT requires 'formal testing', which means that tests should be designed and executed in a structured way that provides objective evidence of the acceptability or otherwise of the system.
2. The definition speaks of testing with respect to 'user needs, requirements, and business processes'. It does not mention any particular specification document but it does draw attention to what users need and it goes beyond testing software to include business processes.
3. The definition speaks of satisfying 'acceptance criteria', which define what is acceptable to the users.

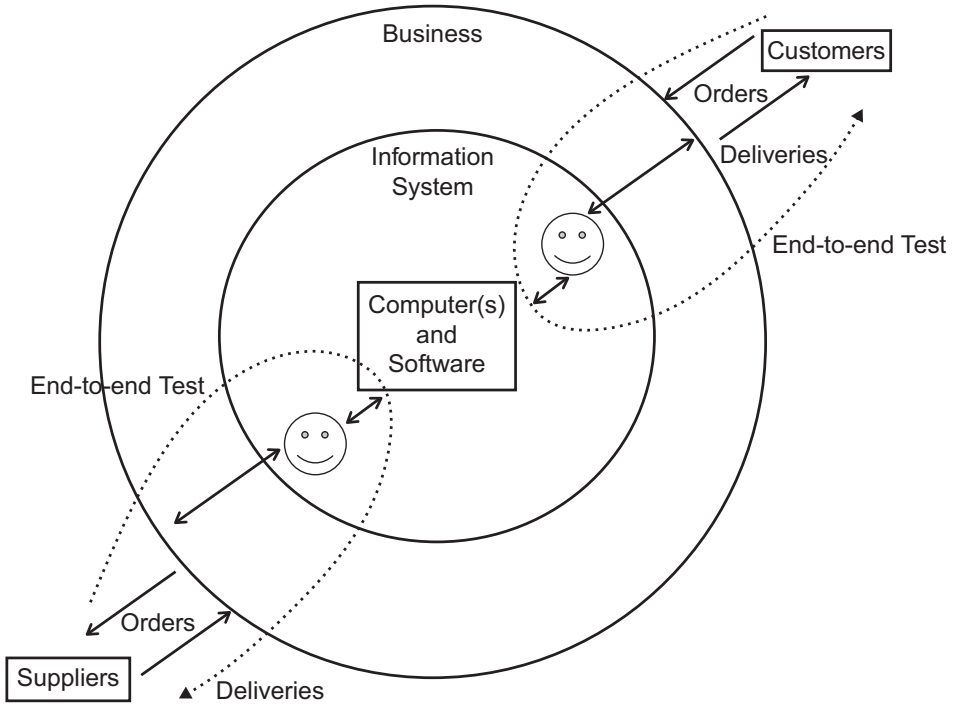
We will explore all three of these key ideas in the book and we will be developing our UAT around acceptance criteria and formal testing of user needs and business processes.

Although other testing takes place before UAT is carried out, UAT is unique in that it is the first time the completed system is tested by users against user needs. Testing during development is based on ensuring that what is delivered matches the corresponding technical specification, but the fit between an IS and its business users is more intimate and it is the unique user perspective that can identify the problems of fitness for purpose in a business context. Without it a system that has been tested by developers and professional testers, and which has passed every test, can still fail at the first hurdle of real use.

That is why UAT needs to include 'end-to-end' testing that exercises complete business processes, including the computer system, rather than testing the computer system in isolation (Figure 1.1). For example in an online retail system, a test of the ordering process must check that an order placed by a customer is accepted and that it triggers identifying, packaging and shipping of the correct product. But these processes must trigger other processes to complete a transaction, for example that the order generates and sends a correct invoice and that payment received is correctly matched with the invoice to close the transaction. Although the scenario is simple, this is a fairly complex set of interacting processes and there is also a time lapse between the initial order and the payment of the invoice, so a full end-to-end test of the transaction will need to follow

an order through the system and over time to ensure all the necessary interactions happen correctly and produce a successful result.

Figure 1.1 End-to-end testing of a transaction in an online retail system



WHY TEST ISs?

Software is one of the greatest examples of human creativity and endeavour. The growth in what can be achieved with computers and software in recent decades has been phenomenal and the trend is likely to continue. Software and software-based systems such as IS are becoming ever more endemic in business life. As the complexity and subtlety of the relationship between IS and business increase, so the impact of failure, by which we mean any mismatch between what an IS actually does and what we need it to do, becomes greater.

This increase in the risk of using an IS must be balanced by increased attention to risk management. There are many facets of risk management but, in the world of software and IS, testing remains one of the key risk management disciplines. As software has become more complex, testing has had to become more effective at identifying problems and potential problems so that they can be resolved before a system fails. As we will explain in Chapter 3, where the sophistication of development activity

has increased, testing has also had to become more sophisticated. Why? Because with greater complexity comes greater risk of human mistakes in design and in implementation; because greater integration between human activity and the systems that enable it brings more opportunities for mismatch; and because the many essential characteristics of ISs (such as reliability, usability and security) all need to be assured before the system can be trusted to deliver its services in a real business situation.

The increased dependence of business on IS has also made the relationship between systems and business processes more critical to business success, and in many cases the desired behaviour of an IS cannot be adequately described in technical specifications. Most testing during development is based on ensuring that what is delivered matches the corresponding technical specification, but the fit between an IS and its business users is more intimate and more critical than the fit between craftspeople and their tools; it is rather more like the fit of a garment that is 'made to measure' so that an individual can work comfortably and effectively in it all day and every day. Whether that kind of fit has been achieved is not something that can be determined by anyone other than the user. That is why UAT is different from all other kinds of testing and also why users must take part in it. It is the unique user perspective that can identify the problems of fitness for purpose in a business context and, without it, a system that has been tested by developers and professional testers, and which has passed every test, can still fail at the first hurdle of real use.

REASONS WHY WE NEED TO TEST

The fact that we live in an imperfect world is, at the most basic level, why we may have trouble creating a perfect IS and why it is important to test them.

There may be many reasons why requirements do not exist at all or are not complete and up to date:

- It is often hard for the sponsor or end-users to imagine what their requirements are going to be. Ideas about what the system should become change over time and in the context of development.
- Even when the requirements are clear in the minds of the sponsor or end-user, they may have trouble communicating the requirement. What is written in the requirements document may therefore not reflect what was intended.
- If the requirements reflect what was intended by the sponsor or end-user, the developers may interpret requirements incorrectly because they will bring their own assumptions to the process.
- Even if correctly imagined, communicated, written and interpreted, developers may make mistakes when writing the code that creates the functionality described in the requirements because they are human and therefore fallible.

All of these have been cited as significant issues in studies of why systems fail.

BUSINESS VULNERABILITY

The relationship we have with IS in business is a unique one. In most cases the system does not just make our lives a little easier; it actually carries on some or all of our business processes in a way that may be difficult or even impossible to do manually once the system is in place. The intimate relationship between businesses and the ISs that serve them makes us particularly vulnerable when a system is first introduced or when it is changed or updated. The term 'business critical' is often used for systems that have a significant impact on the way a business operates and delivers its services. To be clear, this is not just about avoiding high-profile failure; it is about ensuring that each and every time we initiate a new IS or change the relationship between a business and its IS, we take care that the new relationship will be as we expected and planned it to be.

The UAT process is the last, and arguably the most important check, before an IS is rolled out. ISs are becoming more and more embedded in business processes and in delivery of business value and, as a result, they are also becoming more complex. The combination of the complexity and business criticality of ISs leaves little margin for error.

THE CHAOS REPORT – THE STANDISH GROUP

The Standish Group was formed in1985 by a group of IT professionals whose aim was to collect case information on real-life IT failures in order to improve IT project success rates. The original CHAOS report, published in 1994, examined 8,380 projects in order to determine whether they were successful and, if not, what the causes of the failures were. The report found the following percentages of projects were successful, unsuccessful or impaired:

- Successful projects completed on time, within budget, containing all the features and functions initially specified – 16 per cent.
- Challenged projects completed and working, but over budget, over time and offering fewer features and functions than initially specified – 53 per cent.
- Impaired projects cancelled at any point during the development cycle or not used post-completion – 31 per cent.

The report also identified the reasons for the failures, listed from the most often occurring to the least often occurring:

1. incomplete requirements;
2. lack of user involvement;
3. lack of resources;
4. unrealistic expectations;
5. other;
6. lack of executive support;

7. changing requirements and specifications;
8. lack of planning;
9. did not need it any longer;
10. lack of IT management;
11. technology illiteracy.

The 2011 edition of the CHAOS Manifesto, an annual report from The Standish Group that examines trends in software project success, found that in a marked improvement on the 1994 figures 37 per cent of all projects succeeded, 42 per cent of projects were challenged and the remaining 21 per cent were considered a failure. Notably, however, the majority of projects are still challenged or impaired according to their findings.

Not all the reasons listed are relevant to UAT, but those that are relevant will be at the forefront of our thinking when planning and executing UAT activities.

The massive disruption to NatWest Group customers following a routine update to its software is an example of what can happen when systems do not perform as expected in a business-critical environment such as banking. The introduction of a new baggage-handling system for Heathrow Airport's new Terminal 5 is another recent example.

THE HEATHROW TERMINAL 5 OPENING

In March 2008 Willie Walsh, CEO of British Airways (BA), gave evidence to the House of Commons Transport Committee about the problems with the opening of London Heathrow Airport's Terminal 5. In his evidence he said:

> ... we had compromised on the testing regime as a result of delays in completing the building programme and the fact that we compromised on the testing of the building did impact on the operation of T5 in the first few days after its opening.

In the first five days of operation the cost to BA of the operational problems was estimated at £16 million and a total of 23,205 bags were 'misconnected'. The extensive delays experienced by passengers, the sight of the unfinished buildings and the problems caused by delays and unprepared staff also caused embarrassment to BA, the British Airports Authority (BAA) and the UK government (House of Commons Transport Committee 2008). One of our main aims in this book is to explain why these kinds of outcomes happen and how they can be avoided.

Meticulous planning, use of highly experienced and competent practitioners and good project management are all essential to success, but they are not enough. We need

to know exactly what will happen from the point when the first real user logs on to a system, and no amount of time and money spent on modelling, problem solving, focus groups, contingency planning or any other discipline will tell us what we need to know as effectively as a well-planned, well-structured and well-designed UAT. The immediate cost of failure (£16 million in a few days in BA's case) and the ongoing impact on a business of even a small interruption in a key service, such as paying bills through a bank, have a huge impact – significant enough to cause even large companies to fail.

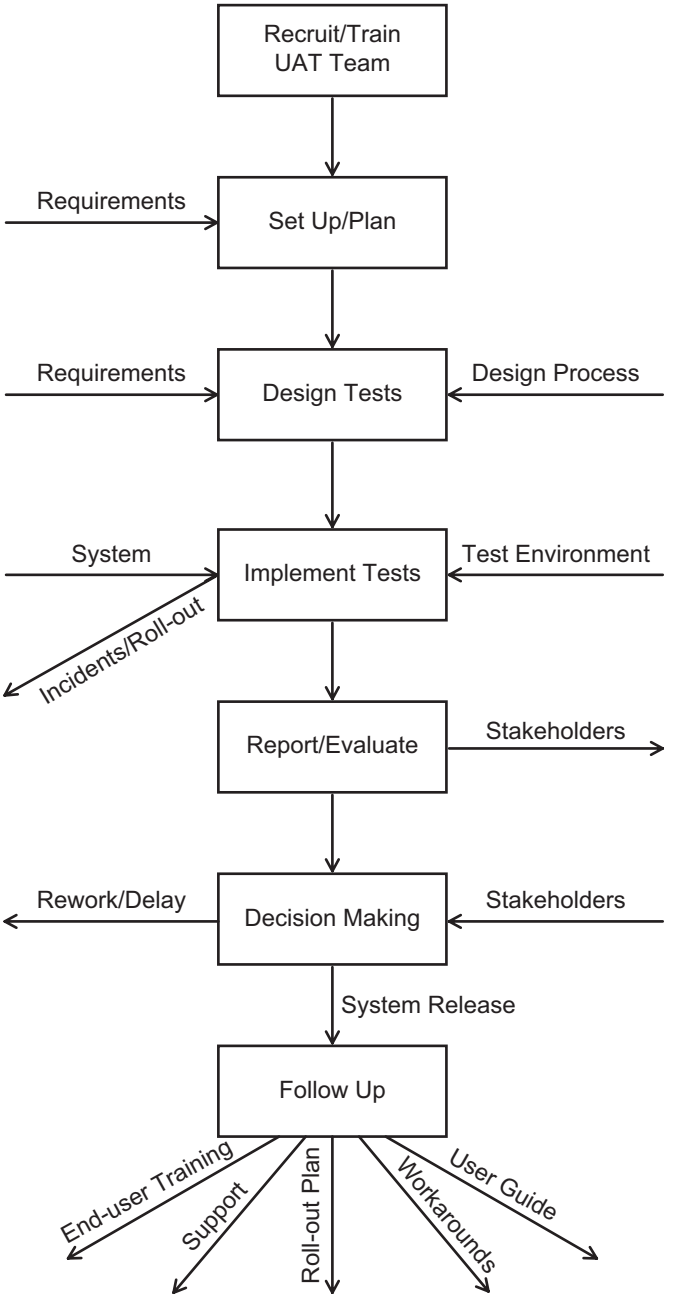
It is important to understand at this stage that although UAT has so far been described as the activity that can verify the usefulness of the new IS and save a project from going horribly wrong, UAT may not be perceived as such by others. UAT is sometimes perceived as an expensive evil that interrupts 'business as usual', diverts important staff, and costs a disproportionate amount of time and money. The number of high-profile failures associated with releasing software systems that were not ready ensures that most organisations make at least some attempt at an acceptance test, often by pressing reluctant users into an activity in which they feel well outside their 'comfort zone'. Does this have to be the case? Can we make UAT an exercise that is independent of the IT community that built the system yet in partnership with it to achieve the best possible result? The premise of the book as a whole is that we can.

Before we embark on UAT we need to be sure that the cost and effort are worthwhile and we also need to understand the activity well enough to do the best job with the fewest resources. If we understand the activities of UAT well enough to carry them out effectively, the parties involved in UAT should be well prepared, the testing should prove the usefulness of the system and little testing time should be wasted or subject to delay.

THE UAT PROCESS

The process we will describe in this book will take you through each stage of achieving successful UAT in a step-by-step fashion. We will first explain enough background to enable you to see how the parts of the process fit together and we will provide you with tools and techniques that you will need to do your testing. Next we will describe how to recruit and prepare an effective UAT team. Finally we will take you through the stages of a UAT project, again step by step, so that you can apply all the tools and techniques at the right time and in the right way. The basic process is shown in Figure 1.2 and it is based around the fundamental test process as defined in the ISTQB Glossary so that you can compare it with other testing processes you may read about elsewhere.

Figure 1.2 The process of UAT



The fundamental test process (FTP) has five stages.

The fundamental test process (FTP)

The ISTQB Glossary describes the FTP as comprising five test stages:

1. test planning and control;
2. test analysis and design;
3. test implementation and execution;
4. evaluating exit criteria and reporting;
5. test closure activities.

Each stage represents a number of key tasks that need to be carried out in order to complete the formal testing as described in the ISTQB definition of UAT. The stages also represent the order in which these tasks are carried out, although we will see that in real life we will probably need to move between these stages, especially when additional functionality is identified that needs to be tested or faults are repaired that need to be retested. The FTP can be applied to any testing level or to an overall testing project.

For our UAT process we will add an additional stage at the beginning for the unique element of UAT of recruiting and forming a UAT team – unique because UAT is conducted by end-users rather than test professionals; it does not simply follow on as one more testing stage for the professional testers to do. UAT team recruitment, development and training are explained in Chapter 4.

The second stage aligns with the FTP's 'test planning and control' stage; it entails setting up the UAT project, identifying its purpose and goals, ensuring that we have a sound basis for testing, and planning all the remaining stages. The key input is requirements, which can be problematic, and Chapter 2 explains how we ensure we have the right requirements to work from. Chapter 6 explains in detail what we have to do at this stage.

The third stage aligns with the FTP's 'test analysis and design' stage; it entails generating tests from the requirements, using well-defined processes that we will explain in Chapter 3. The output from this step will be a complete set of tests that we need to implement to achieve our overall goal; this is explained in detail in Chapter 7.

The fourth stage aligns with the FTP's 'test implementation and execution' stage. At this point we need the system itself and any test environments needed to enable all the tests to be successfully executed. As we test we will generate data about the status of the system that we can report to stakeholders and we report any problems we find to the development team for investigation and, if appropriate, correction. This is explained in detail in Chapter 8.

Next comes a step that aligns with the FTP's 'evaluating exit criteria and reporting' stage. In our UAT process this is the point at which we gather together all the information from all the testing to determine whether the acceptance criteria have been met and, if not, how far short the system falls and what could be done about any problems. This vital information enables the final acceptance decision to be taken and guides the follow-up from UAT to system deployment. This is explained in detail in Chapter 9.

The final step aligns with the FTP's 'test closure activities' stage and extends it to consider the implications of what was found in UAT and what was decided about deployment. Valuable experience and knowledge gained in UAT can now be used to identify any 'workaround' solutions needed to overcome shortcomings of the system, guide end-user training and the user documentation needed to support users, shape the kind and level of technical support needed, and plan deployment mechanisms and timescales. This is all explained in Chapter 10.

THE MANY FACES OF UAT

Contract acceptance testing

Testing to determine whether contractual conditions have been met, so UAT is based on the requirements defined in the original contract for a system acquired from a third-party supplier.

Factory/site acceptance testing

For systems requiring installation after build, there may be a factory acceptance test to establish that the system meets requirements on completion of factory build before installation. Installation may have its own contractual conditions and there may be stage payments for completion of each stage, especially if installation involves shipping overseas. In this case factory acceptance will normally be based on requirements in the contract and site acceptance will be based on achieving the same set of requirements in an installed location.

Alpha/beta testing

When requirements are difficult to define or deliberately open-ended, a conventional contract acceptance may not be possible. In this case some form of 'testing in use' may be done. Alpha testing would be conducted at the developers' premises and beta testing would be at the customers' premises. The actual testing may be based on specific activities but more often is left to the users' discretion. For obvious reasons this is seldom used for custom-built systems, but may be used for releases of commercial products to the marketplace.

Field testing

Field testing is needed for systems that are deployed when in use, such as a control system for fire services. The system elements that are used in deployed locations are tested in their deployed environment to ensure that they are fit for purpose and sufficiently robust, and that communications with base facilities are effective.

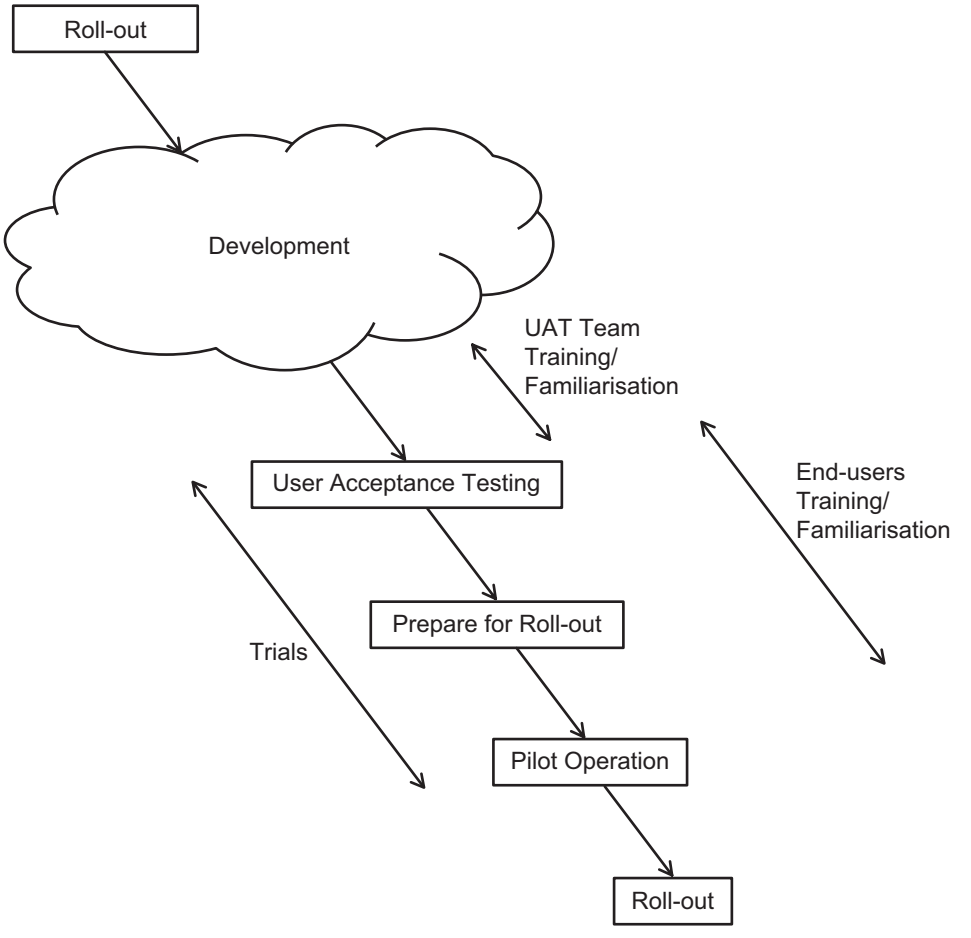
FROM UAT TO SERVICE DELIVERY

The importance of testing throughout the development of an IS should not be underestimated and, as The Standish Group has shown, there is a strong argument for

getting the eventual users involved as early as possible and for keeping them engaged throughout development. Some systems are built this way, but many more are not – and however we build a system, at some point it has to be transitioned from development to use. It is at this transition that most problems with a system become visible (though the root cause may have occurred much earlier). The choice is between unearthing the problems before and during the transition or living with them after the transition. Given that, for most systems, the transition involves some kind of handover from development to users that involves money or some other form of exchange, we need a mechanism for making the transition.

Figure 1.3 shows the transition phase at the end of development. It may have several steps, but UAT will be a critical step that will enable others. The transition may take many forms in different organisations and the UAT step will have to be shaped to the overall transition process, so UAT can also take many forms.

Figure 1.3 The transition to live use



The tested system will then move on through other steps in transition that might involve running a pilot project to ensure that the system is capable of meeting the business need on a small scale before roll-out. At some point users will need training and familiarisation with the new system. Roll-out may need to be phased to reduce the impact on existing systems and processes. There may be many factors that can affect the transition mechanism, some of which are outlined below. The choice of transition mechanism will affect the nature of UAT and the way we plan it.

MANAGING THE TRANSITION TO LIVE USE

Pilot projects

A pilot project is a scaled-down implementation, usually in a controlled environment and with specially selected staff. The aim is to demonstrate that the system is ready for implementation and to identify any problems that might affect a full-scale roll-out.

Phased transitions

Following a successful pilot a phased roll-out can be used to gradually increase the scale of operations. The aim in this case is to provide an opportunity to confirm what was found in the pilot and to manage any issues that arise related to scale, for example logistics.

Training and familiarisation

All users will need training before roll-out is complete. This may be incremental with a phased implementation. Users will also need to become familiar with the system following training so that they do not lose the knowledge and skills acquired during training and so that they are reasonably proficient when the system goes into live operation. This is a separate exercise from UAT, although UAT will certainly enhance the system knowledge of the testers. If UAT is used for training or familiarisation, the effect is not only to slow down the testing but also to compromise the quality of testing because testers not trained on the system will be likely to make errors in designing and/or executing tests.

UAT AND CONTRACTS

In the development of an IS we typically see requirements evolve throughout development, so that the system delivered to UAT is typically rather different from that specified in the initial requirements specification.

Where a contractual agreement exists, however, the contract is normally based on an agreed requirements specification, and that specification will have been defined at the beginning of the contract and, therefore, also at the beginning of the project. Even if some allowances are made through the contractual mechanism for changes, it is almost inevitable that the contractual requirements (on which contractual acceptance must be based) will not completely describe the evolved business and user expectations

at the end of the project. UAT must therefore embed contract acceptance within a wider UAT exercise.

We need to make a distinction between the acquisition of an IS and the development of an IS. We speak of developing an IS when we build the system in-house using our own resources so a contractual relationship is not necessary (although some organisations may use an internal contract to enable one part of the business to buy services from another). Even though the relationship between sponsor, business managers, end-users and developers may be relatively formal, the possibility of coming to a final agreement about what is an acceptable outcome always exists. If an IS is acquired from a third party, or if a third party is instrumental in developing and delivering it, the scope for flexibility in the end result is very limited.

FOUR WAYS TO ACQUIRE SOFTWARE FOR AN IS

There are at least four different ways we can acquire software:

1. Build the system in-house (with our own resources).
2. Outsource the development to a third party (who could be offshore).
3. Acquire a commercial off-the-shelf (COTS) solution configured to our specific needs.
4. License software for our use (software as a service).

An IS may use any of these to acquire the software part of the system and this will have an effect on how UAT is performed.

Build in-house

Within this category of systems, requirements may be documented before development begins or they may have evolved incrementally during development. In extreme cases, requirements may never have been documented at all or not documented in any detail. In all these cases some work will need to be done to match what has been captured in requirements with current user expectations.

The main advantage of building software in-house is that the key stakeholders – developers, sponsor(s), manager(s) and end-users – should all be available for consultation.

Outsource

A system acquired via outsourcing development will most likely have been designed and developed in a place remote from the location where it will be used. The requirements in this case may have been written by the acquirer, the developer or in some form of collaborative effort, but the likelihood is that they will be similar to the requirements we would get from an in-house development. The development process may also be similar to that used in-house.

Where the similarity ends is in the availability of information and the accessibility of key stakeholders. Sponsor, manager(s) and end-users should all be available, but developers will be off-site, making communication about requirements, testing and other aspects of the development activity more difficult. Essentially this acquisition is similar to in-house development but communication channels will be potentially less reliable, communication lines will typically be longer and information may be harder to get and take longer to acquire.

COTS

COTS software covers a wide range of systems from simple productivity tools to sophisticated decision support systems. COTS solutions are already defined and built in modular form so implementation involves selecting modules, configuring individual modules, where this is feasible, and installation. In this case there is no access to the requirements from which the modules were built and the purchaser (sponsor) must make any changes necessary to make the solution work by changing business processes.

UAT for a COTS system is, in some respects, very simple because the modules will typically be in use in other organisations and will have been in use for some time. There should be very few, if any, issues with the way the software functions. On the other hand, the fit with the business processes is likely to be less than ideal and assessing how well the solution will work in the business context and how well users will be able to adjust to its approach will be much larger concerns. UAT in this case is much more about assessing a solution than about testing software.

Software as a service

Software as a service (SaaS) may be used to provide services that the purchasing organisation does not wish to develop or host on its own systems. The service is provided under a licensing arrangement and is usually a 'standard' service that will be in use in many organisations; as a result, it should be stable and reliable in use. In some cases the service can be customised to generate changed or additional functionality to fit a particular business need.

In the case of SaaS solutions the main focus of UAT will need to be the fit of the solution to the organisation's business processes.

Whenever we do business with a third party we enter into some form of contract and, where a contract is in place, contract acceptance will be essential to enable completion of the contract and any payment associated with that event. However, contract acceptance does not fulfil all the objectives of UAT so we will need to ensure that UAT includes an agreed contract acceptance. That does not mean that UAT will be limited solely to contract acceptance; we will still need to plan to achieve all the other outcomes that give us benefits from UAT.

STAKEHOLDERS IN UAT

There are many stakeholders in an implementation, drawn from the business, its staff and its customers, the development organisation, any suppliers and potentially a host of others. In an ideal world all these stakeholders would be identified, consulted and informed of progress at every stage. In the real world this is hardly ever the case, but we still need to take account of the expressed needs of at least some stakeholders.

For the purposes of UAT the most important stakeholder groups will be:

- the sponsor, which means the person or group that commissioned the system (and defined the business intent);
- the manager(s) who will be responsible for delivering expected business benefits from the IS implementation;
- the end-user(s) who will actually operate the system;
- developers and other technical staff who are responsible for implementation and who will support the UAT effort.

Each of these stakeholder groups has a unique viewpoint that will influence UAT and each group has a set of responsibilities. Each group will be expected to deliver certain information and services and each will have information needs to enable it to carry out its role successfully.

Sponsors

The sponsor is the person who signs the cheques. In a small company the sponsor will typically be the owner; in a larger company a sponsor will typically be an executive at some level who has a budget to acquire software for the organisation. In both cases the sponsor's interest is in getting value for money by ensuring that the acquired software is fit for purpose, usable by the people for whom it has been acquired and meets the acceptance criteria.

The sponsor will normally initiate the risk analysis on which test prioritisation is based and subsequently be the person who makes the release decision. Unless the company is very small (sole trader or partnership), the sponsor is unlikely to be the user who will conduct the tests but may be involved in planning the UAT exercise.

The sponsor's main concern is with achieving the expected business benefits, so their focus of attention will be on identifying potential risks and barriers to success. Following risk analysis the sponsor will be most interested in defining test scenarios that will exercise the software in a realistic environment and take some measurements that give comfort in the performance of the software and its effectiveness in driving the key business success parameters, such as revenue, profit and cost.

Business managers

Business managers are those who will be commissioning the software and delivering the expected benefits so they will need reassurance that the software is usable and

capable of achieving the claimed benefits. For this group the scope of interest is likely to be wider than simple fitness for purpose and the tests defined will need to enable the business processes designed around the software to be exercised by users in realistic scenarios. The scenarios will be based on the success criteria set by the sponsor and will ensure that users interact with the software and the business processes at a realistic level over a realistic time frame.

An example might be to carry out a few days' transactions, using saved data from real processing using the existing system, to ensure that the results are better than those achieved when the transactions were first exercised. This kind of exercise would need the original transaction details to be saved and stored, and it would need the complete system to be set up, with users suitably prepared, to repeat the set of transactions with the new system and measure the results. This is an exercise that may go beyond UAT but will necessarily include a UAT component that can be designed to achieve the desired test coverage and measure outcomes against acceptance criteria. The overall exercise, a form of trial or pilot, can achieve effective UAT as well as a benchmark of business performance for the new system.

There are also other management roles that will be involved in the UAT exercise. One of the most important of these may be a quality manager, who would be primarily interested in the overall quality of the delivered system, or a test manager, who would be responsible for the overall quality and effectiveness of testing and would advise and guide the UAT team to enable them to achieve the best possible UAT with the resources available.

End-users

End-users are those who will directly interact with the system, either inside the organisation (staff) or outside the organisation (suppliers and customers). So a group of end-users is not necessarily a homogeneous group with a single set of expectations; there may be multiple viewpoints and sets of expectations to be satisfied. Since end-users directly interact with the system it is clearly appropriate for this group to carry out the actual testing at the user interface and a team made up of as many end-user viewpoints as possible should be formed for that purpose. The primary role of this team will be to assist in designing the tests and then to execute them and report on the results. A vital secondary role, however, is to raise any issues or insights that arise as they carry out the tests so that the user perception of the system is as fully explored and documented as possible.

For end-users the nature of testing will be in a practical context, so test cases should replicate realistic scenarios and use realistic data to enable the testers to operate the software as it will be utilised to achieve the IS's overall purpose.

End-users of software may be drawn from any specialism or none. They may have lots of previous computer experience or none. Their common characteristic is that they will be expected to use the new system to achieve their own business targets. Depending on the level of previous experience with computers and especially with systems of the kind being tested, users can be encouraged to identify scenarios that will entail use of the software to achieve a particular business outcome. Those with significant experience will be able to suggest scenarios of interest in testing, such as those that will

be particularly challenging for the system, those that will be particularly prone to errors and those that will involve high levels of user interaction. These scenarios will ensure that the testing is based on real business activity, while other tests can be defined to provide adequate test coverage and ensure that all the higher risk areas of operation are adequately exercised.

While end-users are expected to make up the body of a UAT team, it is important that business managers and sponsors are aware that the UAT exercise should also involve exercising their expected interaction with the system. In a well-balanced UAT team all three roles will have some part to play directly in the UAT exercise, but even if this is not possible the dependencies between the roles need to be well understood.

Developers and technical support staff

Although the developers' main work should be completed by the time UAT begins and the work of technical support will not yet have started in earnest, these two groups can make a vital contribution to successful UAT. Developers have an important role in helping UA testers to gain familiarisation with the built system and they will be assessing any incident reports raised during UAT to determine what remedial work may be required. Getting this work done promptly can have a significant impact on the time it takes to complete UAT. Technical support staff will be responsible for managing any test environments put in place for UAT and for keeping the system under test in a serviceable and usable condition. Without them the whole exercise could grind to a halt before it gets properly started.

CHAPTER SUMMARY

The main purpose of this chapter was to provide an overall view of what UAT is about at the broadest level. It has provided a basic understanding of what UAT is and outlined its unique role and purpose in preparation for the content covered in the rest of the book.

The definition of UAT identified the need for formal testing and the importance of acceptance criteria and of understanding the users' real needs rather than relying on a specification written at the beginning of the development project. The basic concept of the business requirement has been introduced and this will be expanded in Chapter 2, but we need to remember that business requirements represent the reason a system is being constructed and this may be quite different to what was originally captured in the requirements specification.

We have shown where UAT fits into the overall process of implementing ISs, where UAT fits into the sequence of testing (regardless of the model the project follows), who carries out UAT and whose needs are primarily considered during UAT. We have also identified how UAT may be affected by the way a system has been built or acquired.

We have considered the evidence that ISs often fail at introduction or after change and we have examined the characteristics of ISs that make them vulnerable to potential mishaps. An analysis of the CHAOS report provides surprising and interesting information about how closely ISs reflect the original business

requirements, and important conclusions can be drawn about the value of keeping stakeholders involved throughout the development process.

The way UAT is carried out to avoid damaging failures represents UAT as a form of risk management, and UAT also has an important role to play in building user confidence, assessing the readiness of the system for deployment, and preparing for roll-out.

Finally we have identified the key stakeholders in UAT and their roles and we have outlined the main areas of cost associated with UAT.

After reading Chapter 1 you should understand the purpose and the benefits of UAT, and who is involved in UAT, and understand and justify its costs.

What have you learned?

Test your knowledge of Chapter 1 by answering the following questions. The correct answers can be found in Appendix B.

1. In what way does UAT differ from other testing carried out during development?
    - **A.** UAT takes longer to do than other kinds of testing
    - **B.** UAT must be carried out by end-users
    - **C.** UAT evaluates the system against the requirements specification
    - **D.** UAT can be informal testing as long as it is done by users

2. Which **three** of the following are used as the basis of UAT?
    - **A.** Technical specifications
    - **B.** User needs
    - **C.** Requirements specification
    - **D.** Acceptance criteria
    - **E.** Business processes
    - **F.** Tests developed for system testing
    - **G.** Automated tests designed by the developers

3. Why is it vital for users to carry out UAT? Select **three** options.
    - **A.** Because they have more time available than developers and testers
    - **B.** Because they are not experts on the technical performance of the system
    - **C.** Because the users' ability to use the system as a tool to achieve business benefit is critical to success
    - **D.** Because only users will be able to recognise some situations in which the system's behaviour may be counterproductive
    - **E.** Because developer testing has less value than UAT
    - **F.** Because the system must be evaluated against user needs and use actual business processes.

Some questions to consider (our responses are in Appendix B)

1. What questions would you ask if your organisation asked you to carry out UAT on a new piece of software that is being introduced to support the sales activity in your business?
2. How would you react if your boss told you that the development project for which you will be doing UAT is running late and he wants you to do UAT in parallel with the developers completing the development?
3. Why not just get professional testers to do UAT? After all they have experience of formal testing and know all the techniques.