16) Testing

General remarks

Any **product must be tested** before it is made available to the users it is intended for.

It is very likely that a product that has not been sufficiently tested will feature major defects. Such defects may "**kill**" a product (sometimes even people...) and convey a very negative image of the developer and/or of the project owner (publisher, etc.), even if the product is free of charge (eg an institutional website or a website with adverts).

Testing is therefore a **fundamental** activity of any project. The **PM**, as **guarantor of the product's quality**, must ensure that testing is well planned, well organized and performed in depth and exhaustively. Furthermore, the PM must **ensure that problems revealed by the tests are reported and solved** by whomever is concerned (content provider, developer, graphic designer...).

In principle, **developers should test the code they have written and debug it** before their software is made available to the project owner.
However, as a rule, **the PM cannot rely on developers to thoroughly test the result of their work**. Indeed, developers tend to expedite testing, even if it is organized and controlled by a Development subproject manager.
Testing by developers must therefore be supplemented with testing organized and controlled by the PM, which is the focus of this chapter. The perspective is that of an overall PM working for the project owner.

Test plan – Test cases

A **test plan** and "**test cases**" (or "test scripts") need to be prepared for each of the **major deliverables in the project execution process**.

The test plan should include a testing schedule and it should specify "who tests what".
The test cases should describe in detail the elements to be tested, what the tests should consist of, as well as the expected results of each test.

Note that preparing a test plan and writing test cases are tasks that should be explicitly featured in the **overall project plan**, along with the actual testing tasks, as part of a "**Testing work package**" for each deliverable.

Obviously, **adequate resources** need to be assigned to testing, which is generally very time-consuming. The time and resources (mainly people) required for testing are **often underestimated**.
Note that trainees (interns) may be assigned to parts of the testing effort if the PM is certain they will do a good job (some guidance and control may however be necessary).

Internal testing vs external testing

The PM may subcontract testing to a **service provider**, in particular, in the case of software, to check that the product works properly on a broad variety of platforms not available in-house. A service provider may also be used for performance testing, load testing and for accessibility testing (which usually requires the involvement of visually impaired persons).

If an external testing service is to be used, the scope of the tests to be subcontracted should be extremely well defined, in particular with precise test cases. The service should be limited to testing that cannot be performed in-house, unless the PM believes that some degree of duplication may be useful and that the corresponding cost is justified.

Experience shows that external testing, though extremely useful and even absolutely necessary in many cases, is generally not sufficient, in particular as regards editorial testing and even functional testing of complex products. Successive versions of software must therefore also be tested **internally** (in-house), **not only by the developers but also by the project owner**.

Types of test

There are several **types of test**, in particular:

- editorial tests,
- content integration tests,
- user interface tests,
- functional tests,
- technical tests,
- performance tests,
- documentation tests,
- accessibility tests,
- free tests.

For all types of test, with the exception of free tests, **compliance** of the tested elements **with the requirements and functional specifications** needs to be checked. The specification documents, in total or in part, must therefore be made available to testers for reference as needed, unless the test scripts provide sufficient information.

In certain cases, **automated testing software** may be used to perform automatic tests of successive releases of a product. Such software must obviously be designed and developed with extreme rigour, and, of course, it needs to be thoroughly tested!

The expression "**black-box testing**" is sometimes used to refer to functional (or behavioural) testing, which does not require any knowledge of the software's internal structure and coding, as opposed to "**white-box testing**" (or structural testing, aka clear-box testing, glass-box testing, transparent testing), which requires internal knowledge of the software and therefore can be performed only by the developers themselves or technical persons specialized in such testing.

Editorial tests

Editorial tests concern the product's **content**: texts, images (and captions) and other multimedia assets, such as animations, for which compliance with the script must be checked. Editorial tests should also include testing links between elements of content, in particular those that have been established manually.

Editorial testing also applies to user-interface text elements, as well as user manuals, Help, tutorials, "Read me" documents, etc. These items will be addressed in more detail further on in this chapter.

Editorial testers must be able to **read and write properly** (in at least one of the languages in which the content is written), so they should preferably be professional proofreaders or editors.
Note that **developers and technical people in general are seldom good writers** (they tend to make spelling and grammatical errors) **and are** therefore usually **poor proofreaders**. The same is true of testing service providers, unless of course they are specialized in editorial testing.

Editorial tests may be extremely tedious for products with a substantial volume of content, but they must be performed in a rigorous and exhaustive fashion.

For certain products, **content may be tested independently of the "container"**: for example proofreading of texts and testing of animations may be done before the content is integrated with the software.

In the case of a website, it is generally possible to change content after the site has been released, so editorial bugs that have not been corrected before the end of the product creation process may nevertheless be fixed at a later stage, in principle without too much effort.
Post-release editorial bug fixing is obviously made easier with **"back-office" tools** that enable editors to change content without any intervention on the part of developers.

Content integration tests

As of the first alpha version of the software, it is advisable to check that **content** has been **correctly integrated** and that its **appearance and layout** comply with the specifications, in particular as regards the **style sheets** and, more generally, the **style guide** (or style manual).
Although content integration testing can be performed as early as the first alpha version of the software, it cannot be done exhaustively until the **first beta version** is delivered by the developer, since alpha versions do not include all of the product's content.

A typical problem to look out for is the **degradation of special characters** (accented characters such as "É" or ligatures such as "œ"), and even apostrophes. Such degradation on a website may however be due to a bad choice of character encoding…
Note that special characters often pose a problem in search functions, so searching for words featuring such characters should also be thoroughly tested (this is particularly important for products featuring a full hypertext capability, where a click (or double-click) on a word triggers a search for that word).

Another potential problem to be submitted to testing is the **degradation of content due to digitization, reformatting, compression** or any other processes that may not have been performed properly. So content integration tests should also cover the **rendering of multimedia assets**: sharpness of images, fluidity of animations and videos, quality of sound.

Content integration tests may be very tedious but they should be carried out systematically and with extreme care.

User interface tests

These tests apply to all of the elements that make up the user interface of a product: its overall architecture (from the user's viewpoint), navigation tools, ergonomic features, menus, buttons, checkboxes (tickboxes), tooltips (or "bubble help"), error and warning messages, graphic elements, etc.

All of the text elements of the user interface must be proofread for each language by testers capable of detecting grammatical and spelling errors and inconsistencies (eg a mixture of verbs and nouns in a list of menu items, lack of consistency in the use of capital letters).

In this respect, it is **not advisable to let developers provide any user interface text items** (since they will generally be misspelled!). They should be delivered by the project owner as early as possible in the development cycle, in order to avoid having developers create temporary elements that might remain "as is" in the final product.

If testing reveals major **problems in terms of ease of use** of the product, due to incomplete or imprecise requirements or poor design, it may be necessary to **negotiate design changes** with the developer, which is generally easier to achieve with an in-house development team…

Functional tests

The purpose of functional testing is to check that **all specified functions and features** of a product (including back-office tools) **have been implemented and work properly**. Tests must be systematically performed for each function and each feature, even the most mundane such as "login", "logout", "copy & paste" and "print".

As mentioned above, in order to check the compliance of the product's functions and features with its specifications, testers need to have access to them or, even better, they should be provided with **precise scenarios** (test scripts) describing the specific tests to be carried out and the expected result of each test.

For certain functions, a **special test version** may be required. For example a test version of an online payment system should be used in order to avoid any actual disbursement on the part of the testers.

Technical tests

Technical tests are intended to check that all of the software's functions and features work properly on all **platforms** (computers, servers, smartphones, tablets, etc.), in all **environments** (operating systems, browsers, plug-ins, etc.) and with all minimum **configurations** documented in the requirements specification.

As mentioned above, exhaustive technical testing, in particular configuration testing, generally requires the involvement of a service provider in addition to internal testers.

Technical tests should be conducted neither too soon nor too late. A good choice might be the **first beta version** of the software. If serious problems are detected, additional rounds of tests should be done on the next beta versions.

Waiting for the first beta version of the software to do exhaustive technical testing does not preclude conducting **partial tests** at an earlier stage, eg with an **alpha version**. It is indeed advisable to **identify and report major technical problems as soon as possible**.

Performance and load tests

For certain applications, it is important to check that performance (speed, response time, throughput, etc.) is acceptable with minimum configurations and maximum loads as documented in the requirements specification: speed of search, speed of display, overall performance of client-server software or of a website with a large number of simultaneous users, etc.

Like technical testing, performance testing may be done on a few configurations as soon as the **first alpha version** of the software is available. Full-scale performance testing should be done with **a beta version** of the product and generally requires the use of a specialized service provider and/or of automated performance and load testing tools.

Documentation tests

A product generally includes documentation (for example a Help feature, an installation and operation manual, a "Read me" file). As any other part of a product, documentation must be tested.

Help text, which may be illustrated with screen shots and/or videos/animations, should be **in phase with the product**. This is one of the reasons why Help testing is necessary, and it also explains why Help content should be **finalized at the latest possible stage** of the development process. The schedule agreed upon by the PM and the developer should therefore reflect the need to integrate the final Help content **just before the "Release candidate"** is produced.

If the product includes a **manual**, it should also be in phase with other components of the product, and should be tested accordingly. Usually, a product's manual has to be **sent to print before the product is finalized**. The manual should therefore be **frozen as late as possible** with respect to the deadline for printing.

As regards the "**Read me**" file, if required, its primary purpose is to inform users of errors in the manual and to provide adequate corrections. It is therefore generally finalized and integrated at the "**last minute**". It should nevertheless be proofread.

Accessibility tests

Some products, in particular websites, are required to be **"accessible" to visually impaired users**. Making a website accessible involves complying with the guidelines of the **Web Accessibility Initiative (WAI)** of the WorldWide Web Consortium (W3C).

In theory, testing the accessibility of a website means checking that W3C/WAI guidelines have been followed by the developer, as well as by the content provider (some of the guidelines apply to content).

In practice, compliance with accessibility guidelines should be checked by visually impaired persons working for a specialized testing service provider or for a university lab or any other organization (which will generally be more than happy to be involved in testing, provided some form of compensation is offered).
Practical accessibility testing covers at least ease of use, in particular navigation, as well as the website's compatibility with **text-to-Braille** and **text-to-speech** software.

> For **more information on the W3C/WAI guidelines**, see www.w3.org/WAI/.

Free tests

It is advisable to supplement the above-mentioned range of tests with testing performed **by people who are not given any guidelines or specific instructions**. Such "free testers" should simply use the product as if they were "real users".

Free testing does not require any reference to the specifications or any test scripts. Free testers should however be required to **make a note** of anything they may find surprising, abnormal or suspicious in the product's content, user interface, functions & features, behaviour and performance.

Of course, feedback from free testers needs to be analyzed, sorted and consolidated by the PM (or by someone designated by the PM), and any **new bugs** that have been identified as a result of free testing should be **adequately described and entered into the bug reporting system**.

Trainees (interns), future in-house users, if any, as well as Marketing & Sales and Customer Services people may be assigned to or asked to volunteer for free testing.

Problem management

Testing is useless if the **problems** it reveals are not **rigorously documented, reported and solved**.

The PM must therefore set up a **process for problem ("bug") reporting, tracking and resolution**.

Such a process can be implemented with a "**bug management**" tool such as Mantis Bug Tracker (see www.mantisbt.org/) installed on an intranet or extranet site.

A bug management tool typically features a data base of "**issues**" and related functions. Each issue is documented in a record with an identifier, a title, a category, a description, a severity level, a priority level, and, if necessary, attached files (eg screenshots).
The person who reports an issue (the "reporter") is also identified.
Of course, the status of each issue changes as work on its resolution progresses (from "new" to "closed", with intermediate stages).
The PM, the development subproject manager, the developers, the testers and any other persons that have been given access to the bug management system may be included in a distribution list for e-mails that are automatically sent by the system whenever a new issue is created and/or when an existing issue record is changed.

Each **issue** should be described in a **clear and precise** fashion. The description should include instructions that enable the developer to **reproduce the problem**. If it is possible, and useful, the problem should be **illustrated** by one or several screenshots.

I strongly advise to deal with **one bug at a time**, ie an issue record should document a **single problem** only. Indeed, experience shows that if a record addresses several issues, there is a high risk that the person assigned to the resolution of the issues will deal with only one of them, solve it and prematurely mark the issue as being "resolved"!

A **discussion** between a developer and a tester (and/or the PM) relating to a specific issue may proceed by means of **notes** added to the issue record.

When an issue has been marked "resolved" by the developer, the PM should test (or have someone test) that it is **really resolved** before accepting the solution and closing the corresponding data base record.

Records marked "closed" are not physically deleted from the data base, so they remain accessible, which is all the more useful as **bugs that have been fixed** in a given version of the software **may reappear** in a subsequent version. In such a case, the issue record needs to be reopened and resubmitted to the person supposed to have resolved it (and/or to his/her manager…).

Note that **regressions** are unfortunately frequent and generally unavoidable. They make testing even more tedious and costly than it should be (in an ideal world…). It is however a fact that must be accepted, so the PM (and the team of testers) should watch out for regressions.

Involvement of the project manager

The **PM** is **responsible** for the **quality** of the product, so he should be directly and fully involved in the testing and problem management processes. With very large projects however, the PM may need to **delegate** some of his responsibilities to a **Testing subproject manager**. The PM should nevertheless get involved at some stage of the process, if only to check that testing has been properly planned and carried out seriously, which may require some testing by the PM!

The PM or the Testing subproject manager may decide to **let testers document issues directly** in the bug tracking system. In this case, testers should be given precise guidelines, and it is necessary to check that the guidelines are actually followed.
The PM or the Testing subproject manager may however prefer to **centralize the collection of bug descriptions** written by testers, in order to review them, to eliminate duplicates and, if necessary, to rewrite the descriptions before creating new records in the data base. This approach is generally necessary in the case of free testing and may also be applied to the findings resulting from tests performed by service providers.

> *At Hachette, I made a point of getting deeply involved in testing, particularly for major reference products such as the encyclopedia and dictionaries.*

In fact, I would not rest until I was confident that the product had reached a high level of quality, which required total dedication to the task, including spending the better part of weekends performing tests!
Furthermore, as far as the EHM was concerned, I was in charge of entering issues into the bug management system and coordinating their resolution in close cooperation with the development team.
By the way, at one point in the development of a new version of the EHM, the number of software bugs that had been identified exceeded 1,024 and, to dramatize the event, I sent a congratulations message to the development team, which some developers did not really appreciate!

Product acceptance

Acceptance of a product is a process which is usually applied to the successive versions of the product under development (alpha, beta… final). If the result of testing a given version of the product is positive, then that version is accepted, otherwise it is rejected!
The above applies to a prototype of a product as well as to the "real product".

Testing performed on behalf of the project owner is sometimes called "**user acceptance testing (UAT)**", as opposed to testing carried out by developers.

As mentioned in Chapter 8 ("Requirements specification"), in the case of business applications and, particularly, for complex information systems (IS), specific testing phases may be required and therefore specified by the project owner, such as:

- an "**operational acceptance testing (OAT)**" phase, sometimes called "**operational readiness testing**", intended to determine whether the product meets requirements in a real-life situation on a limited scale before it is fully deployed;
- an "**operational health check (OHC)**", intended to verify that the product meets requirements after being fully deployed and put into unrestricted and regular service.

As the deadline for completion of the development work gets closer, it may become apparent that some of the bugs identified as a result of testing cannot be fixed without delaying the product. In such a situation, the PM must exercise his judgment, weigh the "pros and cons", and decide either to give the developer a little more time or to release the product with "**known bugs**".

In principle, known bugs should not be promoted to the status of "feature" (applying the principle described by "if you can't fix it, feature it!" is certainly not good practice…).
The problems should be fixed for a subsequent version of the product, in agreement with the developer, which may require some form of negotiation and agreement preferably reached before the product is officially accepted in its current state.

Once the tests have been completed and a decision has been made concerning any known bugs, it is **the PM's responsibility** to promote the final "Release Candidate" to the status of "Final product"!
Depending on the context (organization, level of authority of the PM, etc.), a representative of the **project owner** other than or in addition to the PM (and in addition to the Testing subproject manager, if any) may need to be **involved in the final acceptance process**, which should be featured as an explicit task in a project plan.

At Hachette, as director of a Products division, I had been given authority and full responsibility for the acceptance of products managed within my division. So if I declared a product worthy of being released, then it was released. In some cases however, I felt it was necessary to obtain prior approval from my management, which usually involved explaining the situation and justifying the fact that there was more to gain from releasing the product than from delaying it.