

Not a matter of opinion



QBIT's web testing course presenter and PT editor

Edward Bishop believes web usability testing must be objective

For public-facing web applications which are intended to attract and retain users, good usability is as critical as correct functionality.

This article suggests a method, designed for use by testers carrying out functional testing of a public-facing website, to improve usability at the same time as, and as part of the same activity as, assuring correct functionality. It may also be of use in some non-public-facing applications, eg intranets and extranets. However usability of these is often considered to be of less importance because poor usability can be compensated for by user documentation and training and because the users generally have no or less choice as to whether or not to use the application.

What is web usability?

BS 7925 defines usability testing as *testing the ease with which users can learn and use a product*. IEEE 610 defines usability as *the ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component*. Both of these are hopelessly vague.

For a list of similar (and mostly similarly useless) definitions Google for “usability definition”. The problem with nearly all of them is their reliance on the relative term “ease” (or similar alternatives such as “user-friendly”).

Although many meanings can be applied to these terms, there are two main possibilities: “easy” can mean “straightforward” or it can mean “efficient”. This article is about the former, ie how we as testers can help to increase the ease with which users of a web site can discover how to operate it correctly.

The so-called “usability standard”, ISO 9241, is almost exclusively concerned with the latter meaning; it, and most other non-testing sources, describe what most people (including the standard itself) call “ergonomics” rather than usability. Poor ergonomic design in software, generally speaking, is obvious and detected early in development. Even if this is not the case and it falls to users to complain that they are being required to

repeat tasks or do manual work unnecessarily, such defects are usually easy to fix. However ISO 9241 does give us the best definition of usability so far: *the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use*.

A third school of thought says that usability is “whatever the user wishes to define it as”. While this is obviously an admirable sentiment, it seems of little practical use.

When our ultimate aim is to increase usability by testing, it is better to take the negative view and define what usability is not; that is, to define a usability failure. For a website, this is one of three occurrences:

- 1 The user makes a mistake
- 2 The user becomes unsure what to do, or whether what they have done is correct
- 3 The user misunderstands the meaning of information imparted by the site.

Heuristics are for designers, not testers

Many of the most often suggested approaches to usability testing involve comprising the design features of the site under test to advice given by various commentators on usability, also known as “usability gurus”. The most famous of these are Jacob Nielsen (see www.useit.com), Vincent Flanders (www.webpagesthatsuck.com) and Steve Krug (www.sensible.com). These sites, and the interesting books written by these and similar authors, contain examples of good and bad usability and advice on which navigation and other design features tend to cause usability problems. Nielsen, in particular, has listed general principles (he calls them “rules of thumb” or “heuristics”) for user interface design. All web designers should read at least some of this material.

Now, imagine yourself as a tester speaking to a designer/developer: “I’ve seen this feature on your site and Nielsen says you shouldn’t have done it. His explanation of course is a general rule of thumb (or another guru gives a specific example which refers to a different

site and is set in a different context), but as far as I can tell with my limited knowledge, understanding and practical experience of web design you’re in the wrong and should be criticized and made to redo it the way I think it should be”. The likely effect on the tester-builder relationship, project morale and the final product is obvious.

Keep your opinions to yourself

The second type of usability testing is often carried out informally throughout the development process, and usually does more harm than good. It’s natural for people to form opinions on the site; the subject may range from specific details (“users won’t understand what the wording of that link means”) to broad generalizations (“they shouldn’t have designed the navigation like that, they should have done it more like my favourite site”). These opinions, especially the broader ones, are valuable early in development, and should be taken into account by those making decisions.

Once those decisions have been made however, further expressed opinion becomes damaging because it undermines the work already done and gives rise to change - with all the loss and risk that implies - for no clearly-defined or traceable reason.

Persons who have been exposed to the design of the site, or who have knowledge of the underlying business processes, cannot predict what users will understand or do. To do so would require the ability to “pretend” that one does not know what one does know. In many years working on web sites - right from their earliest days - I have never met anyone who can do this, from the most intuitive programmer, to the most experienced interface designer, to the most analytical and methodical tester. Their knowledge of the project makes them “tainted” and their opinions on usability worthless. Even so, their hunches may in fact be correct; but there is no way to check this.

Managers need to draw a line much earlier in projects beyond which the opinions of tainted persons, including themselves, are discounted and discouraged. After this point has been



Cresta Testing – accident prevention through test driven development

Ensure that you conduct the testing of your mission critical systems early in the project lifecycle through Cresta's Structured System Testing Methodology™ (SSTM™). Reduce risk and cost through the customisable Processes, Tools and Techniques which allow you to effectively manage and execute the complete lifecycle of testing within a project.

Cresta Training – where accidents should happen

Fully exploit your investment by turning WinRunner & LoadRunner into "wealthware" through educating staff in testing, quality processes and effective test automation. Cresta provides a broad range of renowned software quality and testing courses, including ISEB Certification in Software Testing, Mercury Interactive TestDirector, QuickTest Pro, WinRunner and LoadRunner.

For more details, contact Karen Espley on +44 (0)870 1600 333, email info@cresta.net or visit www.cresta.net/testing_training.php.

reached, change should be allowed only if based on empirical evidence. Otherwise, the final design is likely to be influenced mainly by those persons with the loudest voices and/or most manipulating personalities. The dreadful end results of this syndrome can be observed at any of thousands of hard-to-use websites, many owned by very large organizations.

Usability trialling is not a test technique

One well-known way of gathering that empirical evidence is to introduce untainted persons in order to observe them using the site and ask their opinion. Some of the usability gurus advocate this, and much advice on various ways to carry out such an exercise (often called “user testing” or “usability trialling”) can be found at www.usability.gov and www.usableweb.org.

The exercise can be very enlightening and interesting. However from the testing point of view there are several fatal, and unavoidable, flaws:

- 1 There’s no way to know if the abilities of the subjects are typical of real users
- 2 Trials can’t be done until late in development when change probably means very expensive rework
- 3 Any rework carried out as a result must then be retested (see 1)

The last is the most serious. The trial is likely to report that a large proportion of subjects made mistakes, encountered difficulty or expressed dislike of certain features or points in navigation of the site. However the subjects are far less likely to agree on what change could be made to improve matters – it must be remembered that they do not have a design or IT background (they could not be representative of users otherwise) and have been given only a few minutes to think about the site. Their opinions, although they may give useful insight, must not be used as a design.

So instead the findings are given to the site designers and they are asked to act upon them. But they have already tried to produce the best and most usable design they could: their first reaction may well be to say “despite the trial’s findings, we think this part of the site is already as good as it can be” – in which case, there was no point carrying out the trial.

Otherwise, the designers must now think of even better ideas, or else readopt alternative ideas they had previously rejected. In either case, there is no way of knowing whether the new version is better than the old; the only chance of finding that out is to repeat the usability trial with new, untainted users. That will probably give rise to another, similar set of results suggesting shortcomings, perhaps in the same places as before, for reasons which are unknown: perhaps the subjects have less or different abilities than the first group, perhaps

Test 15c: To show that a bill of value more than the current balance of account cannot be paid

Entry criteria

- function 15 ‘Pay a bill’ and function 3 ‘Check balance’ released for testing

Actions

- 1 log on to use home banking
- 2 obtain the balance of the account to which you have logged on
- 3 attempt to pay a bill of value exceeding the balance of the account
- 4 check the balance of the account again

Exit criteria

- balance checked successfully at least once
- bill payment details displayed for confirmation reflect user input correctly
- bill payment can be confirmed

Expected outcomes

- after confirming bill payment, information that bill cannot be paid appears
- the displayed balance of the account remains unchanged

Figure 1: Nongranular, requirements-based web test script

they are judging the features relatively and picking out the “worst” rather than the “bad”, or perhaps the designers’ first attempt at a given feature was in fact more usable. So we now have a situation where two expensive trials have been done, and perhaps more are needed, with no certainty of conclusive results at any time.

The best way to do usability testing is to do testing

If we as testers are ever going to help to provide more usable sites to users, a test method is needed that

- 1 is more objective and does not depend upon opinion
- 2 can be done by testers without the use of untainted subjects
- 3 can be started earlier and continued for longer, improving usability gradually throughout development
- 4 is economical enough for use in web projects with short timescales and frequent change of requirements and design for reasons other than usability issues.

Objectives 3 and 4 can be achieved by performing usability testing not only in parallel with but as part of the same activity as functional testing. The key to being able to do this in the test design phase: web testing scripts should provide opportunities for testers to detect usability issues. This means writing a script whose input section (often called “actions” in web testing) describes what the person executing the test should do to exercise the functions but avoids mention of how they should go about it.

The very granular scripts used for non-web testing, featuring step-by-step instructions, confirmation of the expected outcomes at each

step, and individual traceability of every step to a business objective, are not appropriate for web testing. Users of a public web application will not have such guidance and will probably not be prepared to invest much effort in learning how to operate the interface. Writing such a script assumes that the developers and testers can predict the navigation choices and other inputs made by the user, and it is the unpredictable deviations from this prediction made by real users that give rise to severe usability failure and which testing should aim to detect.

Nongranular scripts such as that shown in figure 1 give exactly the same benefit to analytical testing: the scripts describe the system from the test analyst’s understanding of the requirements, thus providing a check that the developers’ understanding is the same, as well as helping to guide the developers in directing and prioritizing the development effort with the intention that all the tests will be passed first time. However they can also provide effective usability testing during the empirical phase which granular scripts cannot.

The starting point for a script should, of course, be a functional test objective. The most difficult part of writing a nongranular script is getting the correct level of detail. This is a skill learned quickly with practice. The script shown in figure 1 is a good example but was not written like this at the first attempt; rather, it is the result of a refinement process. Start by writing the steps as they come to mind, and then pass through them looking critically for ones which can be simplified because they give unnecessary detail or eliminated because they are not essential to the test objective. Incidentally, a similar process can be applied to the entry criteria for each test, enabling testing to be done earlier, when fewer items have been released.

If you ignore these opportunities, it's not just software you should be testing

£ATTRACTIVE + EXCELLENT BENEFITS + TRAINING + RAPID CAREER PROGRESSION

We are the world's largest and fastest growing independent IT performance assurance consulting group, with offices in 13 countries and over 550 consultants, including more than 100 in the UK. Our comprehensive, unbiased range of testing solutions – from specific expertise to full outsourcing – use automated testing tools across all platforms, applications, networks and operating systems over a range of sectors including financial services, telecoms, defence, public sector, retail and leisure as well as healthcare.

TESTING PROFESSIONALS ALL LEVELS – ANALYSTS, SENIOR TESTERS, TEST TEAM LEADERS & TEST MANAGERS

We have very ambitious plans for expansion in the UK, where we expect to grow by 94% and increase headcount by 60% in 2004 alone, and seek equally ambitious people who would thrive on working on high level projects, both locally and as part of international teams.

You will need a minimum of two years' relevant experience, good client and consultative skills, the personality and flexibility to thrive as part of a team in a global organisation as well as the ability to learn quickly and the drive to get things done. We are particularly interested in people whose background includes any of: Peoplesoft/HR Programs; SMS; MMS; WAP; LoadRunner; WinRunner; Rational Tools; Java; Biztalk; .NET; Set Top Box/Digital TV.

Our exceptional client base, variety of new projects and range of sectors mean the opportunities for personal and professional growth with Tescom are immense – and we are dedicated to rapid promotion from within.

So why not join us? And test yourself to the full. Please send your CV to jobsuk@testcom-intl.com or post it to Tescom Direct, Tescom International, 21-22 Great Sutton Street, London EC1V 0DN. www.tescom-intl.com

TESCOM

AUSTRALIA FRANCE GERMANY ISRAEL SINGAPORE UNITED KINGDOM UNITED STATES

Objectives 1 and 2 are achieved by recording every usability failure experienced during testing, in exactly the same way that a functional failure is raised. This requires a certain amount of discipline from the persons executing the tests and recording results; however it is not necessary for these persons to be untainted.

The script shown in figure 1 was created from its test objective, which was based on a description of a function, which was derived from business requirements. All this was done before any design work had taken place and so the script does not contain any assumptions about *how* the user will access that function. Such a script can be executed by its author so that in very small projects all the testing can be done by one person. Very often in web projects early screen designs are available at the same time as or even before requirements and this may enable more detailed scripts to be created which do refer to specific screens and objects by name. A discussion of which type of script is best is not in the scope of this article, but it should be understood that scripts of the second type contain an implicit assumption that the user – who does not have access to these names – will recognize the purpose and meaning of the screens and objects. To check this assumption such scripts must be executed by someone other than their author; this means that in small projects all the testing could be done by two people, sharing the test design and execution work equally. In either case, however, during the empirical phase more people can be enlisted if necessary to execute scripts.

The point of writing scripts without guidance as to how to proceed is to make the knowledge and ability of the text executors irrelevant. Obviously proficient web users, or those with a knowledge of the system under test or its underlying business processes, will tend to experience fewer usability failures. However when they do, it is very likely that less proficient or knowledgeable users will experience the same failure. A long list of errors made by inept users is of little value; those users would probably experience similar difficulties on any site, and in any case their ability is beyond our control. A shorter list of errors and other usability failures recorded by experienced testers is an accurate guide to where the most severe usability faults lie; addressing these will lead to improved usability for users of all proficiency levels.

Failure is a fact, not a feeling

When executing a test, from time to time the tester will experience a usability failure; one of the three types listed above. The tester must be familiar with this list and sufficiently disciplined to be aware of and record each failure just as they would a functional failure or anomaly.

The first type, “the user makes a mistake”, will typically include failures such as:

- user navigates to a page which was not expected
- user revisits a previously-viewed page inadvertently
- user enters invalid data to a form and is asked to amend

Failures suggesting these might be reported by the tester, respectively, as comments such as:

“When I clicked the “My account” link I expected to get a page that would let me log on and was surprised to see that it meant an explanation of the types of account available”

“I selected “recent transactions” from the dropdown list when trying to check if my standing orders were active for step 4. I didn’t expect that it would lead back to the page showing what I’d done in this session, to which I’d already been at step 2”

“I didn’t notice that I could not choose a password less than 6 characters long and my first choice was rejected”

These are obviously most likely to be noticed the first time a tester runs a given script, but sometimes testers will find by accident later in the script, when executing the script again (eg for compatibility testing), or when executing another script that they have been operating under a misconception and so experienced usability failure previously:

“I’ve just realized that up until now I’ve been going round the long way by selecting ‘music’ then ‘CDs’ then ‘popular music’ then ‘70s’ when I could have just selected ‘Great 70s Pop’ from the dropdown on the front page. I ignored this link before because I thought it was just the title of an album being promoted”

The second type, “the user becomes unsure what to do, or whether what they have done is correct”, might manifest itself as follows:

- user is not sure which link to click to see information they want
- user does not know what they should enter in a field
- user is uncertain whether they have entered a user transaction or not

which could be experienced and recorded by the tester as, respectively,

“At step 4 (order a 128MB DIMM) I was unsure whether I should select “computer upgrades” or “computer components”

“When entering my account number at step 3, I wondered whether or not I should include the dashes”

“When the ‘track your order’ screen appeared after I submitted the order form and stated ‘order not yet picked’ I was not sure

whether this meant I had placed my order successfully or not”

The third type, “the user misunderstands the meaning of information imparted by the site”, is the rarest; however if it occurs after deployment it can sometimes have severe business impact. It is usually one of the following situations:

- user selects incorrect product based on what they think the content means
- user proceeds incorrectly in the real world based on what they think the content says
- user misinterprets information or instructions on the site as advice

These are not usually reported directly by the tester; like the user, he or she can make mistakes of this type and remain unaware of the fact. However, they are sometimes prevented indirectly by the results of functional testing with a usability element; for example when examining a back-end database to ensure that transactions have been recorded correctly, it might be noticed that a tester has ordered a product which does not fit the criteria in the test script. More often, the tester realizes that they have been operating under a misconception and records something like:

“I’ve just realized that the colour of the model car is not shown in the photo but in that little box underneath. I’ve been ordering the wrong colours”, or:

“I’ve just been surprised to find out that the ‘buy this share’ button appears on every page. On most of the ones I’ve looked at it’s been scrolled off the bottom. I thought we were using it to recommend certain shares to users and might have chosen those shares as a result”

This last example actually occurred after deployment of a share dealing site and affected some users. It could not have been detected by error-guessing techniques such as checklists of site heuristics, and it is unlikely that inexperienced users performing a trial would have noted it. However usability testing of the type described in this article would have stood a chance of detecting it.

They all count

It is vitally important that testers are aware and bear in mind throughout testing that:

- 1 They must record every usability failure, even if they feel it is a result of their own lack of concentration or knowledge; users will be affected by exactly the same factors
- 2 Failures must be recorded even if the tester feels he or she could or should have noticed and recorded them earlier; testers, like users, will notice and deduce more about the site over time

3 One of the objectives of this method is to eliminate opinions and base change on empirical evidence. A tester may believe that a feature of the site will cause problems to some users and should be changed; however there is no way of knowing whether they are right in this or not. In contrast, when an error is recorded by a tester he or she is attesting to the fact that a usability failure occurred. It is conceivable (but would seem, intuitively, unlikely) that a tester with a strongly-held opinion might introduce a failure report designed to support that opinion.

If these guidelines are followed test execution will produce a potent deliverable: a stream of failures experienced during realistic use of the site, indicating precisely those points where it is important to improve usability. These must not be ignored by development; “we think this was caused by the tester, or an unlucky coincidence, and we don’t want to change it” is not an acceptable resolution. Every failure recorded should cause a change to the interface, and if necessary the underlying system, which attempts to prevent that or similar failure happening again, during testing or after deployment; and that change should not be a reversion to a previous state which, by definition, has also been shown to give rise to usability failure. Often the change will be as simple as rephrasing or increasing visual prominence of text or an object; occasionally it may involve rethink-

ing the grouping of information and form fields into pages or the navigation paths and methods offered to the user.

Sometimes a change will be made which is ineffective or actually makes the same or another usability failure more likely; if so, these failures will come to light as testing proceeds. The aim is not to attempt to maximize usability of the product in one “big bang”, but to improve it gradually as testing and development proceeds, in the same way and at the same time as functionality is improved.

Because the functional tests have been prioritized by their relationships to functions and thus to business objectives, the more important faults should come to light earliest in the testing, and as testing and rework proceeds the frequency and criticality of faults reported should gradually lessen. More importantly, there will be more time to work on the faults associated with critical and/or popular functions; if change leads to further failure reports there will tend to be time for further change and further testing until failure reports related to the changed items cease. As the project nears its close, faults found during execution of the lower-priority tests will have less time to reach stability; so the success of this approach depends upon accurate prioritization of the functional tests.

Summary

Using this method usability changes made to the site are based on empirical evidence – the fact that a usability failure was observed – not subjective opinion. The evidence is gathered during the activity which must be carried out anyway: functional test execution. The number of faults detected, and the chance of detecting those which are hard to find, is increased due to repetition of navigation and other actions and the fact that testers have more time to think about their use of the site and are relieved of the very difficult mental exercise of trying to pretend they are a user. The best use is made of the testing time available since usability testing starts at the same time as functional test execution and is automatically prioritized by its association with functional testing.

The main disadvantage of this method is that the nongranular test scripts do not lend themselves to automated execution; or, to put this another way, automated testing cannot help with usability. However it might be possible to gain the main advantage of automated testing – faster and more reliable functional regression and maintenance testing – by creating an automated test suite near the end of the first phase of test execution when the frequency of usability failure reports has fallen and further major changes to the user interface become less likely. PT

ARE YOU SURE YOU’RE MAKING THE RIGHT TESTS?



Qualified to succeed

WHEN TESTING SOFTWARE YOU NEED TO BE SURE YOU’RE WORKING TO THE HIGHEST POSSIBLE STANDARDS – EVERY TIME.

ISEB Software Testing qualifications give you all the assurance you need. The Foundation Certificate is proof of an excellent basic understanding; the Practitioner Certificate demonstrates in-depth knowledge and ability to carry out testing. Internationally recognised, they are the IT industry’s gold standard and evidence your staff and your testers work to best practice. How’s that for a competitive edge.

www.iseb.org.uk/pt



MTG/AD/856/0304

For information contact Customer Support: Tel: +44 (0)1793 417542 Email: isebenq@hq.bcs.org.uk Quoting reference number: 856/0304. The BCS is a registered charity: number 292786

VACANCIES FOR MANUAL & AUTOMATED TESTERS

SDLC Solutions, the test consultants, are winning new business in a number of industry sectors. We pride ourselves on commitment and delivery, and our clients agree. We need testers who share our vision and our passion.

We offer a full and open career path

We encourage you to develop in the way that suits you

We’ll train you in testing techniques, testing methodologies, and management

We’ll train you in automated tools

All our testers go through ISEB certification

If you’re interested in joining our fast-moving, rapidly-expanding company, email m.chan@sdclsolutions.com, quoting reference: PT05 or call us on 01625 521093

Public ISEB Practitioner course running in June. Call for details

*“SDLC’s enthusiasm and energy provides motivation that I have not come across before, and this has filtered through to full time testing staff at all levels”
(Client Test Director)*



Committed to Efficient Testing

www.sdclsolutions.com