METHODS & TOOLS

Global knowledge source for software development professionals **ISSN 1023-4918**

Summer 2000 (Volume 8 - number 2)

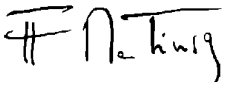
A simple challenge

Software project planning and control belong to the core activities of software engineering. The common opinion about software project management is that "failure" is more the standard than "success", even if these words have surely not the same definition for everyone. Schedule and cost overruns seem to be linked to the majority of projects. When you look at published figures on software project results you will see the usual Standish Group's CHAOS figures (go to www.standishgroup.com for more info) showing for instance that around 30% of projects are canceled before completion. Only 16% of software projects are completed on time and on budget. Our own process quality database shows that only 60% of the companies have a formal schedule estimation mechanism and 18% a formal cost estimation mechanism.

When you look however at Kathleen Peters' article and more precisely at the software estimation 101 part, software estimation does not look like rocket science. If you read a basic book about software project management like the DeMarco reference, you will find that most of the text is related to common sense. So why are so many projects bound to fail... or to "succeed" like they are failures for most of the participants?

First you have to accept the limitations of what management can achieve. Projects are launched in organizations and you cannot request too much rationale behaviors from humans. But we have also to admit that project planning and control activities are not always highly considered. Some project managers are coming from the developers' ranks and they do not like "administrative" tasks. Other come from the middle management and they can perform only administrative task without being able to see an operational meaning to the time and costs records and act accordingly. Hello Dilbert!

So perhaps the biggest challenge of project planning and management is simply there: to find qualified people and to take the time to do it properly.



Inside

Software Project Estimation

Kathleen Peters, kpeters@spc.ca
Software Productivity Center Inc., www.spc.ca

Effective software project estimation is one of the most challenging and important activities in software development. Proper project planning and control is not possible without a sound and reliable estimate. As a whole, the software industry doesn't estimate projects well and doesn't use estimates appropriately. We suffer far more than we should as a result and we need to focus some effort on improving the situation.

Under-estimating a project leads to under-staffing it (resulting in staff burnout), under-scoping the quality assurance effort (running the risk of low quality deliverables), and setting too short a schedule (resulting in loss of credibility as deadlines are missed). For those who figure on avoiding this situation by generously padding the estimate, over-estimating a project can be just about as bad for the organization! If you give a project more resources than it really needs without sufficient scope controls it will use them. The project is then likely to cost more than it should (a negative impact on the bottom line), take longer to deliver than necessary (resulting in lost opportunities), and delay the use of your resources on the next project.

Software Project Estimation 101

The four basic steps in software project estimation are:

1) Estimate the size of the development product. This generally ends up in either Lines of Code (LOC) or Function Points (FP), but there are other possible units of measure. A discussion of the pros & cons of each is discussed in some of the material referenced at the end of this report.

2) Estimate the effort in person-months or person-hours.

3) Estimate the schedule in calendar months.

4) Estimate the project cost in dollars (or local currency)

Estimating size

An accurate estimate of the size of the software to be built is the first step to an effective estimate. Your source(s) of information regarding the scope of the project should, wherever possible, start with formal descriptions of the requirements - for example, a customer's requirements specification or request for proposal, a system specification, a software requirements specification. If you are [re-]estimating a project in later phases of the project's lifecycle, design documents can be used to provide additional detail. Don't let the lack of a formal scope specification stop you from doing an initial project estimate. A verbal description or a whiteboard outline are sometimes all you have to start with. In any case, you must communicate the level of risk and uncertainty in an estimate to all concerned and you must re-estimate the project as soon as more scope information is determined.

Two main ways you can estimate product size are:

1) By analogy. Having done a similar project in the past and knowing its size, you estimate each major piece of the new project as a percentage of the size of a similar piece of the previous project. Estimate the total size of the new project by adding up the estimated sizes of each of the pieces. An

experienced estimator can produce reasonably good size estimates by analogy if accurate size values are available for the previous project and if the new project is sufficiently similar to the previous one.

2) By counting product features and using an algorithmic approach such as Function Points to convert the count into an estimate of size. Macro-level "product features" may include the number of subsystems, classes/modules, methods/functions. More detailed "product features" may include the number of screens, dialogs, files, database tables, reports, messages, and so on.

Estimating effort

Once you have an estimate of the size of your product, you can derive the effort estimate. This conversion from software size to total project effort can only be done if you have a defined software development lifecycle and development process that you follow to specify, design, develop, and test the software. A software development project involves far more than simply coding the software – in fact, coding is often the smallest part of the overall effort. Writing and reviewing documentation, implementing prototypes, designing the deliverables, and reviewing and testing the code take up the larger portion of overall project effort. The project effort estimate requires you to identify and estimate, and then sum up all the activities you must perform to build a product of the estimated size.

There are two main ways to derive effort from size:

1) The best way is to use your organization's own historical data to determine how much effort previous projects of the estimated size have taken. This, of course, assumes (a) your organization has been documenting actual results from previous projects, (b) that you have at least one past project of similar size (it is even better if you have several projects of similar size as this reinforces that you consistently need a certain level of effort to develop projects of a given size), and (c) that you will follow a similar development lifecycle, use a similar development methodology, use similar tools, and use a team with similar skills and experience for the new project.

2) If you don't have historical data from your own organization because you haven't started collecting it yet or because your new project is very different in one or more key aspects, you can use a mature and generally accepted algorithmic approach such as Barry Boehm's COCOMO model or the Putnam Methodology to convert a size estimate into an effort estimate. These models have been derived by studying a significant number of completed projects from various organizations to see how their project sizes mapped into total project effort. These "industry data" models may not be as accurate as your own historical data, but they can give you useful ballpark effort estimates.

Estimating schedule

The third step in estimating a software development project is to determine the project schedule from the effort estimate. This generally involves estimating the number of people who will work on the project, what they will work on (the Work Breakdown Structure), when they will start working on the project and when they will finish (this is the "staffing profile"). Once you have this information, you need to lay it out into a calendar schedule. Again, historical data from your organization's past projects or industry data models can be used to predict the number of people you will need for a project of a given size and how work can be broken down into a schedule.

If you have nothing else, a schedule estimation rule of thumb [McConnell 1996] can be used to

get a rough idea of the total calendar time required:

Schedule in months = 3.0 * (effort-months)$^{1/3}$

Opinions vary as to whether 2.0 or 2.5 or even 4.0 should be used in place of the 3.0 value – only by trying it out will you see what works for you.

Estimating Cost

There are many factors to consider when estimating the total cost of a project. These include labor, hardware and software purchases or rentals, travel for meeting or testing purposes, telecommunications (e.g., long-distance phone calls, video-conferences, dedicated lines for testing, etc.), training courses, office space, and so on.

Exactly how you estimate total project cost will depend on how your organization allocates costs. Some costs may not be allocated to individual projects and may be taken care of by adding an overhead value to labor rates ($ per hour). Often, a software development project manager will only estimate the labor cost and identify any additional project costs not considered "overhead" by the organization.

The simplest labor cost can be obtained by multiplying the project's effort estimate (in hours) by a general labor rate ($ per hour). A more accurate labor cost would result from using a specific labor rate for each staff position (e.g., Technical, QA, Project Management, Documentation, Support, etc.).

You would have to determine what percentage of total project effort should be allocated to each position. Again, historical data or industry data models can help.

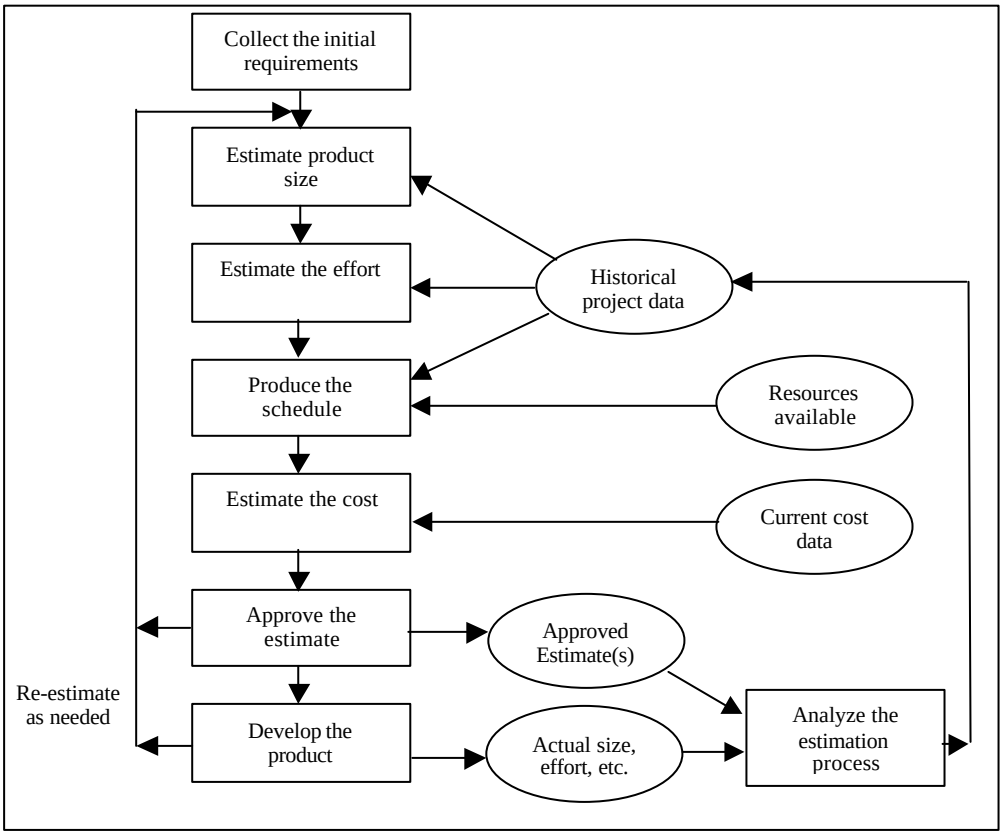


Figure 1 – The Basic Project Estimation Process

Working Backwards from Available Time

Projects often have a delivery date specified for them that isn't negotiable - "The new release has to be out in 6 months"; "The customer's new telephone switches go on-line in 12 months and our software has to be ready then". If you already know how much time you have, the only thing you can do is negotiate the set of functionality you can implement in the time available. Since there is always more to do than time available, functionality has to be prioritized and selected so that a cohesive package of software can be delivered on time.

Working backwards doesn't mean you skip any steps in the basic estimation process outlined above. You still need to size the product, although here you really do have to break it down into a number of pieces you can either select or remove from the deliverable, and you still need to estimate effort, schedule, and cost. This is where estimation tools can be really useful. Trying to fit a set of functionality into a fixed timeframe requires a number of "what if" scenarios to be generated. To do this manually would take too much time and effort. Some tools allow you to play with various options easily and quickly.

Understanding an Estimate's Accuracy

Whenever an estimate is generated, everyone wants to know how close the numbers are to reality. Well, the bottom line is that you won't know exactly until you finish the project – and you will have to live with some uncertainty. Naturally, you will want every estimate to be as accurate as possible given the data you have at the time you generate it. And of course you don't want to present an estimate in a way that inspires a false sense of confidence in the numbers.

What do we mean by an "accurate" estimate? Accuracy is an indication of how close something is to reality. Precision is an indication of how finely something is measured. For example, a size estimate of 70 to 80 KLOC might be both the most accurate and the most

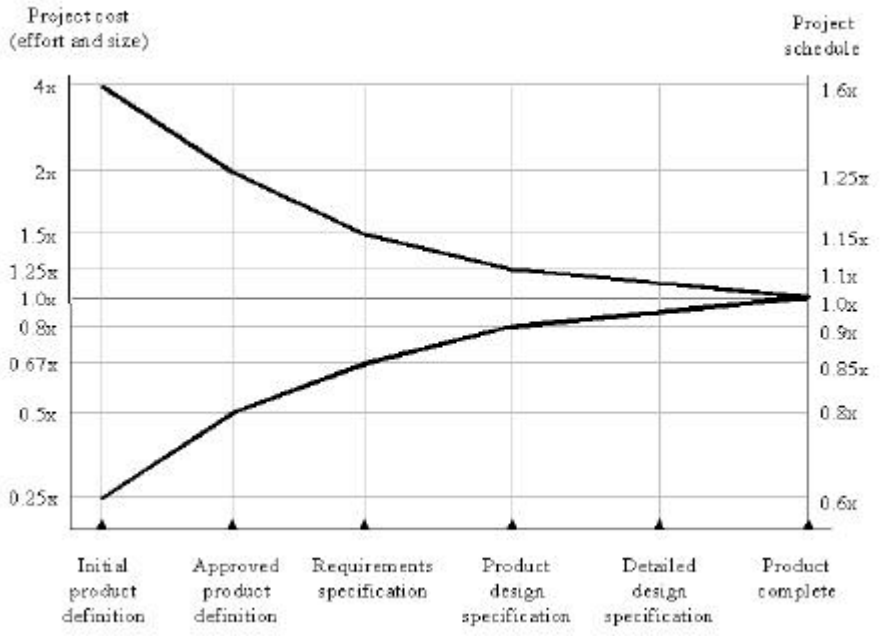


Figure 2 Estimate Convergence Graph; Source: "Rapid Development"
Adapted from "Cost Models forFuture Life Cycle Processes: COCOMO 2.0" (Boehm et al. 1995).

precise estimate you can make at the end of the requirements specification phase of a project. If you simplify your size estimate to 75000 LOC it looks more precise, but in reality it's less accurate. If you offer the size estimate as 75281 LOC, it is precise to one LOC but it can only be measured that accurately once the coding phase of the project is completed and an actual LOC count is done.

If your accurate size estimate is a range, rather than a single value, then all values calculated from it (e.g., effort, schedule, cost) should be represented as a range as well. If, over the lifetime of a project, you make several estimates as you specify the product in more detail, the range should narrow and your estimate should approach what will eventually be the actual cost values for the product or system you are developing (Figure 2).

Of course, you must also keep in mind other important factors that affect the accuracy of your estimates, such as:

- the accuracy of all the estimate's input data (the old adage, "Garbage in, Garbage out", holds true)
- the accuracy of any estimate calculations (e.g., converting between Function Points and LOC has a certain margin of error)
- how closely the historical data or industry data used to calibrate the model matches the project you are estimating
- the predictability of your organization's software development process, and
- whether or not the actual project was carefully planned, monitored and controlled, and no major surprises occurred that caused unexpected delays.

Understanding the Tradeoffs

Once you've generated a project estimate, the real work begins – finding some combination of functionality, schedule, cost and staff size that can be accepted by management and customers! This is where a solid understanding of the relationships between these variables is so important, and where being armed with different project estimates illustrating the tradeoffs is very useful for establishing what the limits are.

Here are a few facts of life you need to remember during the estimate "adjustment" phase:

- If you lengthen the schedule, you can generally reduce the overall cost and use fewer people. Sometimes you only have to lengthen the schedule by a few weeks to get a benefit. Usually management and

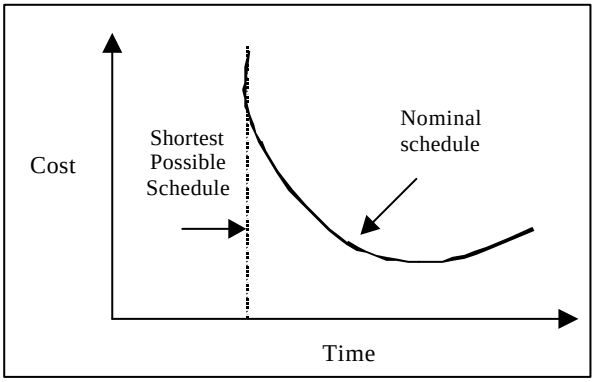


Figure 3 – Relationship between cost and schedule on a software project.
Source: "Rapid Development" (McConnell 1996). The cost of achieving the nominal schedule is much less than the cost of achieving the shortest possible schedule

customers don't want a long delay, but see how much "extra" might be acceptable. Many people don't consider generating an estimate option that lengthens the schedule to see what effect it has unless they are driven to it in an attempt to reduce cost or staff size.

- You can only shorten a schedule three ways. You can reduce the functionality (reducing the effort by doing less), increase the number of concurrent staff (but only if there are tasks you could now do in parallel to take advantage of this!), or keep the number of staff constant but get them to work overtime.

If you can't reduce the functionality, choosing one of the two remaining alternatives is going to cost you. It might cost you a lot more than you can afford to pay depending on just how much you want to shrink the schedule (see Figure 3). And it might not work! Remember the "adding people to a late project only makes it later" rule? Well, the same principle applies here – you can add more people, but the amount of work also goes up because you now have additional communication and management overhead. If you rely on a lot of overtime to reduce the schedule, you have to remember that productivity may increase in the short term but will go down over the long term because people will get tired and make more mistakes.

- There is a shortest possible schedule for any project and you have to know what it is. You can only shrink a schedule so far given the functionality you are required to implement, the minimum process you have to follow to develop and test it, and the minimum level of quality you want in the output. Don't even think of trying to beat that limit!

- The shortest possible schedule may not be achievable by you. To achieve the shortest possible schedule your project team had better all be highly skilled and experienced, your development process had better be well defined and mature, and the project itself has to go perfectly. There are not many organizations that can hope to make the shortest possible schedule, so it's better not to aim for this. Instead, you need to determine what your shortest achievable schedule is (also known as the "nominal" schedule). Data from past projects is your best source of information here.

- Always keep in mind the accuracy of the estimate you are attempting to adjust. If your schedule estimate is currently "5 to 7 months" then a small change, for example 2 weeks, either way doesn't mean much yet. You can only adjust the schedule in increments that have some significance given the accuracy of the estimate.

It's interesting to observe the reactions of people learning to estimate projects who are asked to do a number of different estimates for a project using a variety of options. When they analyze the results, most people are startled by the consequences of different tradeoffs. For example, the following tables provide 3 different estimate options for a 75 KLOC project:

The difference between the nominal schedule and the shortest schedule for the project is only a little over two months, but to achieve the shortest schedule the peak staff has to increase by almost 10 people and the cost increases by over $870,000! These results should cause someone to ask if a 2-month decrease in the schedule is worth the cost, and if 10 additional people can be found in time to help achieve it. For some projects, a schedule decrease may be required at any cost; for others, it won't be.

Not all projects have such dramatic differences between estimate options, but the size-effort-schedule-staff-cost relationship follows some basic rules that you can't circumvent. Having various options available as you discuss a project estimate ensures everyone involved can see those basic rules in action and can make properly informed decisions.

Nominal Plan

Management Metric	Planning Value
Effort (staff months)	40
Schedule (calendar months)	12.4
Cost	$605,868
Peak Staff (people)	4.8
Average Staff (people)	3.2

Shortest-Schedule Plan

Management Metric	Planning Value
Effort (staff months)	97
Schedule (calendar months)	10.0
Cost	$1,479,170
Peak Staff (people)	14.6
Average Staff (people)	9.8

Least-Cost Plan

Management Metric	Planning Value
Effort (staff months)	14
Schedule (calendar months)	16.2
Cost	$212,131
Peak Staff (people)	1.3
Average Staff (people)	0.9

Figure 4 – Three different estimates for a 75 KLOC Project

The Trouble with Estimates

While effective software project estimation is absolutely necessary, it is also one of the most difficult software development activities. Why is it so hard?

The following lists some of the things that make it hard – and they're things that we need to overcome:

- Estimating size is the most difficult (but not impossible) step intellectually, and is often skipped in favor of going directly to estimating a schedule. However, if you haven't thought through what you are being asked to build you really don't have a good base from which to predict a schedule or to evaluate how scope changes may affect the schedule.

- Customers and software developers often don't really recognize that software development is a process of gradual refinement and that estimates made early in a project lifecycle are "fuzzy". Even good estimates are only guesses, with inherent assumptions, risks, and uncertainty, and yet they are often treated as though they are cast in stone. What can help is offering estimates as a range of possible outcomes by saying, for example, that the project will take 5 to 7 months instead of stating it will be complete on June 15. Beware of committing to a range that is too narrow as that's about as bad as committing to a definite date! Alternatively, you could include uncertainty as an accompanying probability value by saying, for example, that there is an 80% probability that the project will complete on or before June 15.

- Organizations often don't collect and analyze historical data on their performance on development projects. Since the use of historical data is the best way to generate estimates for new work, it is very important to establish some fundamental project metrics that you collect for every project.

- It is often difficult to get a realistic schedule accepted by management and customers. Everyone wants things sooner rather than later, but for any project there is a shortest possible schedule that will allow you to include the required functionality and produce a quality output. You have to determine what you can do in a given period of time and educate all concerned about what is and what is not possible. Yes, the impossible has been known to happen from time to time, but it's rare and very costly,

and we count on it far more than it is prudent to do so!

Maintenance & Enhancement Projects vs. New Development

The software industry does far more maintenance and enhancement work on existing products than completely new development. Most maintenance projects are a combination of new development and adaptation of existing software. Although the estimation steps outlined above can still apply to maintenance and enhancement projects, there are some special issues that have to be considered such as:

- When sizing new development for a maintenance project you have to keep in mind that inserting this new functionality will only be feasible if the product's existing architecture can accommodate it. If it cannot, the maintenance effort must be increased to rework the architecture.

- It's tricky to attempt to size adaptation work in the same manner as new work. An experienced individual estimating maintenance effort by analogy is a more common approach than attempting to size adaptation work in LOC or Function Points and then converting size to effort (although approaches to this have been discussed; for example, see [Putnam 1992]).

- Estimation models that are calibrated to produce effort and schedule estimates for new development projects assume everything is created from scratch. This isn't the case for maintenance projects where you are modifying a certain amount of existing documentation, code, test cases, etc. Using these models may tend to over-estimate maintenance projects.

- Often, maintenance work has fixed delivery dates (e.g., a maintenance release every 6 months or once a year) and is done by a fixed number of people (i.e., an allocated maintenance team) so estimates have to deal with fitting work into a fixed timeframe with a constant staffing level.

Some existing estimation models do attempt to address maintenance concerns. At the moment though, there is a lot more support, guidance and discussion available regarding new development estimation than there is on maintenance and enhancement estimation. Hopefully this will change because so much help is needed in this area.

Estimating Small Projects

Many people work on small projects, which are generally defined as a staff size of one or two people and a schedule of less than six months. Existing industry-data project estimation models are not calibrated from small projects and so are of little or no use here unless they can be adequately adjusted using an organization's small project historical data.

Estimates for small projects are highly dependent on the capabilities of the individual(s) performing the work and so are best estimated directly by those assigned to do the work. An approach such as Watts Humphrey's Personal Software Process (PSP) [Humphrey 1995] is much more applicable for small project estimation.

Estimating a "New Domain" Project

How do you estimate a project in a new application domain where no one in your organization has any previous experience? If it's a leading-edge (or "bleeding-edge"!) project, no one else has any previous experience either. The first time you do something you are dealing with much more uncertainty and there is no way out of it except to proceed with caution and manage the project carefully. These projects are always high risk, and are generally under-estimated regardless of the estimation process

used [Vigder 1994]. Knowing these two facts, you must (a) make the risks very clear to management and customers, (b) avoid making major commitments to fixed deadlines, and (c) re-estimate as you become more familiar with the domain and as you specify the product in more detail.

Selecting a project lifecycle which best accommodates the uncertainty of new-domain projects is often a key step that is missing from the development process. An iterative life cycle such as the Incremental Release Model where delivery is done in pieces, or the Spiral Model where revisiting estimates and risk assessment is done before proceeding into each new step, are often better approaches than the more traditional Waterfall Model.

Some Estimating Tips

- Allow enough time to do a proper project estimate – rushed estimates are inaccurate, high-risk estimates! For large development projects, the estimation step should really be regarded as a mini-project.

- Where possible, use documented data from your organization's own similar past projects. It will result in the most accurate estimate. If your organization has not kept historical data, now is a good time to start collecting it.

- Use developer-based estimates. Estimates prepared by people other than those who will do the work will be less accurate.

- Use at least one software estimation tool. Estimation tools implement complex models that would take significant time to learn to apply manually. They also make sure you don't forget anything, and allow you to tune an estimate quickly and relatively painlessly.

- Use several different people to estimate and use several different estimation techniques (using an estimation tool should be considered as one of the techniques), and compare the results. Look at the convergence or spread among the estimates. Convergence tells you that you've probably got a good estimate. Spread means that there are probably things that have been overlooked and that you need to understand better. The Delphi approach or Wideband-Delphi technique [Boehm 1981] can be used to gather and discuss estimates using a group of people, the intention being to produce an accurate, unbiased estimate.

- Re-estimate the project several times throughout its lifecycle. As you specify the product in more detail, your estimates should begin to approach what you will actually use to complete the project.

- Create a standardized estimation procedure that all involved can and do buy into. That way you can't argue about the outputs, only the inputs, and your effort is spent productively understanding the scope and cost drivers for the project.

- Focus some effort on improving your organization's software project estimation process. As each project is completed, compare actuals to estimates – how well did you do in predicting the project's effort and schedule? What did you miss? Where could you improve?

Software Project Estimation Tools

Estimation tools may be stand-alone products or may be integrated into the functionality of larger project management products. Estimation tools may just support the size estimation process, or just the conversion of size to effort, schedule and cost, or both. Tools that support just size estimation include LOC counters, Function Point analysis programs, and even requirements capture and management applications. This section of this report just focuses on estimation tools that are stand-alone products and support the conversion of size to effort etc.

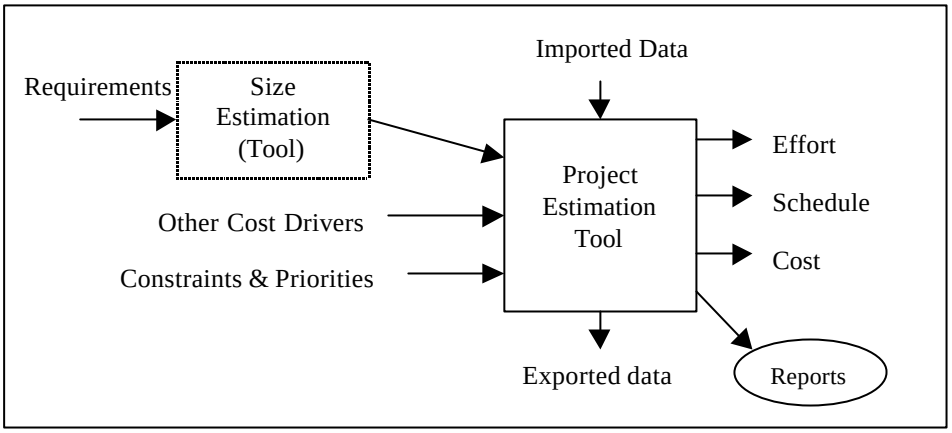


Figure 5 – Estimation Tool Context

No estimation tool is the "silver bullet" for solving your estimation problems. They can be very useful items in your estimation toolkit, and you should seriously consider using one (or more), but their output is only as good as the inputs they are given and they require you to have an estimation and software development process in place to support them. Beware of any vendor claiming their tool is able to produce estimates within +/- some small percentage of actuals unless they also highlight all the things you must be able to do in a predictable manner and what must go right during a project to ensure an estimate can be that accurate.

There are a variety of commercial and public domain estimation tools available. Searching for software project estimation tools on the web isn't as straightforward as one might expect. A mix of keywords and search engines needs to be used to discover about 80% of the tools and web-sites where tool lists were available identified the rest. Web-based information on the capabilities and pricing of the tools is variable and occasionally very superficial, so some phone calls and email should be used to augment what is gleaned from the web.

The following provides a summary of the important features and criteria you should consider when evaluating a software project estimation tool. Figure 5 provides a context for the discussion.

Price

Commercial estimation tools can be categorized as either "for rent" (you pay an annual fee) or "for purchase" (one-time fee), and they come in 3 price ranges: affordable ($1000 or less), mid-range ($1001-$5000), or expensive ($5001 to $20000 or more). Probably only larger organizations or large projects would consider the mid-range or high-priced tools.

The tools under $1000 implement non-proprietary models published by others (e.g., COCOMO) and may lack some of the functionality and extensive support of the expensive options, but can still generate more than adequate estimates.

Platform and Performance

Does it run on your current hardware/software? Does it run on more than one platform? How much RAM and disk space is required? Will its database handle the amount of historical data and the size and number of project estimates you will be entering?

Ease of Use and Documentation

Can you generate an initial project estimate easily the first time you use the tool, or do you have to study the underlying model in detail for days first learning all the acronyms and struggling with attribute definitions? Can you tailor your estimates easily? Do the user manual and help text provide an understanding how to use the tool to generate project estimates, as well as a simple list of what general functionality the tool possesses? Is sample project data available?

Networking Capability

Is there a common, shared database so that multiple users can access and add to the historical project data, and view or update estimates (assuming this is important to you)?

Upgrades

Model data shouldn't be static – as new programming languages and new development paradigms appear, as different kinds of development projects are studied, updates to the model data in the tool should be made. Does the vendor make model updates available to customers? Is the vendor committed to making continuous enhancements that offer new functionality and support new platforms etc.? What do these upgrades cost?

Support

It is important to understand that although estimation tools are becoming more affordable and easier to use, the model(s) they implement are quite complex and you may have questions or need some guidance now and then. Does the vendor provide technical support and a means for asking "how to" questions? Does the vendor offer estimation training courses that extend beyond just how to use the tool or can they recommend supporting courses and training/reading material?

How Scope/Size is Specified

Flexibility is the key – you may start out estimating a project's size a certain way, but as you learn more about the specification of a particular product or as you become more skilled at estimating and branch out into other sizing techniques, you want a tool that supports your needs.

What options does the tool provide to specify the size estimate? Can you enter either LOC or Function Points? Can you specify GUI components, number of classes/methods or modules/functions? Is the size value entered merely as a single number (e.g., 55000 LOC; 345 Function Points), a range of numbers (e.g., Low: 45000 LOC; Expected: 55000 LOC; High: 65000 LOC), or can the size estimate be divided into a number of "modules" or "work packages" that you estimate in whichever way suits that particular component?

Estimation Model(s) Supported

Some tools use one or more proprietary models where little detailed information is published; others use non-proprietary models where you can purchase a book and/or download detailed information from the web to learn more. It takes a lot of resources to develop a sophisticated model of software development so its not surprising that there is only a handful of models.

Regardless of how much you can learn about the tool's internal algorithms, what you must determine is whether or not the estimates generated by the tool are useful in estimating your organization's type of software development projects. Parametric models typically have a bias – for example, some suit a military development process while others suit commercial development.

The only way to quickly become confident that a tool can give you valuable results is to obtain an evaluation or demo copy and estimate previous projects where actuals are known.

Compare the estimates from the tool with what you know about previous projects and see if the results are "in the ballpark".

Assess whether the tool allows you to capture historical information on your past projects and how it requires you to enter it. Some tools can be calibrated to your projects only by modifying the underlying model data (i.e., you have to derive the values yourself); others allow you to simply enter project metrics like actual size, effort, schedule and then the tool derives the model data changes.

Does the tool support the estimation of maintenance & enhancement projects? Does it have support for object-oriented, COTS, software re-use or other issues important to your projects?

Other Cost Drivers

The models generally allow you to specify values for a number of cost, or productivity drivers (e.g., staff capabilities and experience, lifecycle requirements, use of tools, etc.) in order to tailor the estimate to your organization and your project's particular situation. What cost drivers are available and are the values you can set them to useful for your situation?

Constraints and Priorities

Does the tool allow you to specify constraints (e.g., Maximum Schedule=12 months; Peak Staff=10) when calculating an estimate? Does the tool allow you to specify priorities (e.g., "Shortest possible schedule has highest priority"; "Lowest number of staff has highest priority") when calculating an estimate?

Outputs Generated

Look for a tool that has functionality which shows options, probabilities and ranges. Tools using Monte Carlo simulation to generate estimates with different probabilities provide interesting insight into the volatility of the development process.

Reports should help you clearly present and discuss estimate options with customers and management. What kinds of reports are generated and are they useful to you? Can you obtain a softcopy of the reports so that you can add material to them or include them easily in other project documents you generate?

Import/Export Capability

Possible imports include things like module-by-module size estimates, historical project data, and updated model data. Possible exports include schedule, WBS, and staffing information to project management software like MS-Project or to a spreadsheet program like Excel.

Conclusion

There is no quick fix that will immediately make us better estimators and users of estimates. Effective estimates come about as a result of process definition and improvement, education and training, good project management, use of proper tools and techniques, measurement, sufficient resources, and sheer hard work.

Depending on what the situation is when you start, and how long a typical project lasts in your organization, it could be several years before you've had enough time and project cycles to establish the basics from which better estimates are consistently made. Trying to set up everything in a week is equivalent to trying to build Rome in a day.

But don't be discouraged by this! There are things you can do right now to make a difference to your current project, and there are actions you can take to make your next project better.

Assuming you are past the planning stage of your current project and you have little time to

spare, here are some important actions you can take on this project to start improving your estimation process:

- Re-estimate the project at several key stages (e.g., after completion of requirements specification, after completion of architectural design, after completion detailed design).

- At the end of the project, record the actual values (or get as close as you can) for size, effort, schedule, cost, staffing. Start your historical database.

- At the end of the project, review your estimate(s)/actuals and evaluate what you did right and how you might improve in future. Use what you learn here the next time you estimate.

Here are some important actions you can take for your next project:

- Review the current state of your software development process – is it chaotic, or does it have some order and structure that you generally follow? If it is chaotic to start with, or your process breaks down under pressure and you expect that to happen on this project, then any estimate you make had better take that into account. Even better, try to reduce the amount of chaos. Establishing a predictable development process isn't something you can do overnight, but every little bit of work you do in this area will help allow for better estimates and better project control.

- Create a first draft of an estimation procedure document and follow the procedure when estimating. See what works and what doesn't, and adjust as necessary. Note that there are templates around for creating such a document so you don't have to start from scratch.

- Allow enough time to do proper project estimates

- Decide when you should re-estimate the project and put tasks/milestones for re-estimation into the project plan

- Start the education of managers and customers regarding the accuracy of estimates. Offer estimates as ranges, and explain the uncertainty and risks associated with them.

- Follow as many of the other estimation tips given earlier in this report as you can.

Try to make time to:

- work on defining, documenting and/or improving your software development process. A clearly defined, predictable development process is required so that your project estimates can be made on a firm development foundation. There are document templates and process definition guidelines, consultants and courses available to help you in this.

- investigate project estimation tools and use one (or more).

Small improvement steps, taken with care and attention, will lead you down the road to better project estimation and planning. In fact, taking small steps is often the only way to ensure permanent change occurs.

References

General Reading (includes Estimation and Project Planning)

- DeMarco, Tom, *Controlling Software Projects*, Prentice-Hall, 1982
- Goether, Wolfhart B., Elizabeth K. Bailey, Mary B. Busby, *Software Effort and Schedule Measurement: A framework for counting Staff-hours and reporting Schedule Information*, CMU/SEI-92-TR-021, 1992, http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.021.html
- Humphrey, Watts, *A Discipline for Software Engineering*, Addison-Wesley, 1995
- McConnell, Steve, *Rapid Development – Taming Wild Software Schedules*, Microsoft Press, 1996
- McConnell, Steve, *Software Project Survival Guide*, Microsoft Press, 1998
- Vigder, M.R. & A.W. Kark, *Software Cost Estimation and Control*, 1994, http://wwwsel.iit.nrc.ca/abstracts/NRC37116.abs (full text available)

Sizing Lines of Code

- R. Park, *Software Size Measurement: A framework for counting source statements*, CMU/SEI-92-TR-20, 1992, http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.020.html

Function Points

- Dreger, Brian, *Function Point Analysis*, Prentice-Hall, 1989
- Garmus, David & David Herron, *Measuring the Software Process*, Yourdon Press, 1996
- Jones, Capers, *Applied Software Measurement: Assuring Productivity and Quality*, McGraw-Hill, 1991
- Jones, Capers, *What are Function Points*, 1997, http://www.spr.com/library/0funcmet.htm
- Symons, Charles, *Software Sizing and Estimating: Mark II Function Point Analysis*, John Wiley, 1991
- International Function Point Users Group (IFPUG) web site: http://www.ifpug.org
- A function point FAQ: http://ourworld.compuserve.com/homepages/softcomp/fpfaq.htm

Estimation Models

- Boehm, Barry, *Software Engineering Economics*, Prentice-Hall, 1981 (original COCOMO)
- Putnam, Lawrence & Ware Myers, *Measures for Excellence: Reliable Software on Time, Within Budget*, Yourdon Press, 1992
- Putnam, Lawrence & Ware Myers, *Industrial Strength Software: Effective Management using Measurement*, IEEE Computer Society, 1997
- COCOMOII web site: http://sunset.usc.edu/COCOMOII/cocomo.html

Other Estimation Tool Summaries & Discussions

- Douglis, Charles, *Cost Benefit Discussion for Knowledge-Based Estimation Tools*, 1998, http://www.spr.com/html/cost_benefit.htm
- Giles, Alan E. & Dennis Barney, *Metrics Tools: Software Cost Estimation*, 1995, http://www.stsc.hill.af.mil/CrossTalk/1995/jun/Metrics.html
- Software Cost Estimation Web-Site (SCEW), http://www.ecfc.u-net.com/cost/index.htm
- Parametric Cost Estimating Reference Manual, http://www.jsc.nasa.gov/bu2/resources.html
- DoD Data & Analysis Center for Software, http://www.dacs.dtic.mil

About the Author

Kathleen Peters is a senior consultant with Software Productivity Center Inc. (SPC) Kathleen has an M.Sc. in Computing Science and more than 15 years of industry experience developing software and managing projects. She also teaches at Simon Fraser University in British Columbia, Canada.

Kathleen is editor of SPC's free, biweekly e-newsletter *E-ssentials!*, which contains how-to's for successful project management and practical software development. Subscribe at http://www.spc.ca/resources/essentials/index.htm

Risk-Based E-Business Testing - Part 1 Risks and Test Strategy

Paul Gerrard, paulg@evolutif.co.uk Systeme Evolutif Ltd. www.evolutif.co.uk
9 Cavendish PlaceLondon W1M 9DL, UK
Tel: +44 (0)20 7636 6060, Fax: +44 (0)20 7636 6072

There are five main risk areas in E-Business (EB) system development. These risks relate to usability, performance, security, availability and functionality. These risks are not new. What is new is that functionality may not be the biggest concern and the shift in risk from functionality towards non-functional issues is dramatic. Most EB systems are simple (at least from the users' perspective) and the four non-functional areas present a challenge to business and testers.

The pace of development on the web is incredibly quick. 'Web-time' describes the hustle that is required to create and maintain EB momentum. Unfamiliar risk areas, rapid development, uncontrollable user base, new tools and techniques present the tester with a real headache.

What are the imperatives for EB testers? To adopt a rapid response attitude. To work closely with marketers, designers, programmers and of course real users to understand both user needs and the technical risks to be addressed in testing. To have a flexible test process having perhaps 20 different test types that cover each of the most likely problems. To automate as much testing as possible. These are the main planks of our EB Test Strategy.

Whether homegrown or proprietary, the essential tools are test data and transaction design; test execution using the programmer and user interfaces; incident management and control to ensure the right problems get fixed in the right order. Additional tools to validate HTML and links, measure download time and generate loads are all necessary.

To keep pace with development, wholly manual testing is no longer an option. The range of tools required is large but thankfully, most are now widely available.

This paper presents an overview of the risks of E-Business and a framework for creating an E-business Test Strategy. The second part of this two-part paper will describe a collection of E-Business techniques and tools in detail.

1. Introduction

1.1 E-Business, E-Commerce and Everything Else

Before we go any further, let's get some definitions clear. The E-phenomenon is immature and vendors are generating new E-definitions thick and fast. We've taken, as at least an independent authority, the whatis.com definitions:

E-Commerce The buying and selling of goods and services on the Internet, especially the web.

E-Business The conduct of business on the Internet which includes:

- buying and selling plus servicing customers (internet)
- collaborating with partners (extranet)
- internal work and information flow (intranet).

1.2 Current State of E-Business Testing Experience

At a recent conference (February 2000) I asked a group of 190 testers a series of questions and asked for a show of hands.

The table below summarizes the responses:

Question	*% Yes*
Use the web regularly?	100
Bought products or services online?	50
Experience of working on an E-Commerce or E-Business system that is now in use?	<5
Planning to build an E-Commerce system?	25
Planning to build an E-Business system?	30
Experienced problems when using a web site?	100

What interpretation can we put on these figures? Perhaps that the 'first generation' of E-Business systems did not involve the professional testers employed by user IT organizations. Perhaps these web sites were not tested adequately at all?

People abandon Web sites for all of the following reasons:

- Sites are difficult to use.
- Sites are too slow to respond.
- Sites were not trusted to be secure.
- Web pages broke (page not found etc.)
- Goods or services ordered were late, incorrect or never arrived.

Reference 1, Winning the On-Line Customer, Boston Consulting Group, (www.bcg.com) presents some startling statistics concerning the quality of E-Business systems, the response of users and the potential impact on suppliers. For example:

- 4 out of 5 E-Commerce purchasers have experienced failures.
- 28 percent of all online purchases fail.

These statistics make sobering reading. Not only could a poor E-Business site fail to gain more business. They could affect the business done by your bricks and mortar stores.

1.3 'Web time'

It might be regarded as hype, but there is no denying it, in some environments, 'Web time passes 5-7 times as fast'. What does this mean?

The notion of Web time could be summarized as follows:

- There are significant technology shifts every six months. The products you are using today may not have existed six months ago. The developers using these products are inexperienced.
- In some organizations, the E-Business projects may be owned and managed by non-computer experts. Software development is out; web publishing is in.
- Those working in the dotcom environments experience this phenomenon particularly acutely. The sprint to market dominates every other objective. There are many precedents that suggest that in an immature market, getting to market first means that you may dominate the market. In this kind of environment, does quality come second?
- There are obvious cultural, practical, technical and schedule obstacles to implementing thorough quality assurance and testing practices.

1.4 The E-Business Testing Challenge

Testers are used to working in time-pressured projects, without requirements, using unfamiliar technology in cultures that prefer to code first and plan second. But testers have rarely encountered all of these difficulties simultaneously.

This is the E-Business Testing challenge:

- Identify and address risks quickly and gain consensus on the overall test approach.

- Develop and implement flexible test strategies that address the risks of most concern.
- Devise new test techniques and methods that address the particular constraints of E-Business technologies.
- Make best use of test automation in every area of test activity.
- Provide thorough test evidence to help stakeholders to make the correct release decision.

1.5 Risk-Based Approach to Testing

If we believe the computer press, the E-Business revolution is here; the whole world is getting connected; that many of the small start-ups of today will become the market leaders of tomorrow; that the whole world will benefit from E-anyWordULike. The web offers a fabulous opportunity for entrepreneurs and venture capitalists to stake a claim in the new territory – E-Business. Images of the Wild West, wagons rolling, gold digging and ferocious competition over territory give the right impression of a gold rush.

Pressure to deliver quickly, using new technology, inexperienced staff, into an untested marketplace and facing uncertain risks is overwhelming. Where does all this leave the tester? In fast-moving environments, if the tester carps about lack of requirements, software stability or integration plans they will probably be trampled to death by the stampeding project team.

In high integrity environments (where the Internet has made little impact, thankfully), testers have earned the grudging respect of their peers because the risk of failure is unacceptable and testing helps to reduce or eliminate risk. In most commercial IT environments however, testers are still second-class citizens on the team. Is this perhaps because testers, too often, become anti-risk zealots? Could it be that testers don't acclimatize to risky projects because we all preach 'best practices'?

In all software projects, risks are taken. In one way, testing in high-integrity environments is easy. Every textbook process, method and technique must be used to achieve an explicit aim: to minimize risk. It's a no-brainer. In fast-moving E-Business projects, risk taking is inevitable. Balancing testing against risk is essential because we never have the time to test everything. It's tough to get it 'right'. If we don't talk to the risk-takers in their language we'll never get the testing budget approved.
So, testers must become expert in risk. They must identify failure modes and translate these into consequences to the sponsors of the project. "If xxx fails (and it is likely, if we don't test), then the consequence to you, as sponsor is..." In this way, testers, management, sponsors can reconcile the risks being taken to the testing time and effort.

How does this help the tester?

- The decision to do more or less testing is arrived at by consensus (no longer will the tester lie awake at night thinking: "am I doing enough testing?").
- The people taking the risk make the decision consciously.
- It makes explicit the tests that will not be done – the case for doing more testing was self-evident, but was consciously overruled by management.
- It makes the risks being taken by the project visible to all.

Using risk to prioritize tests means that testers can concentrate on designing effective tests to find faults and not worry about doing 'too little' testing.

What happens at the end of the test phase, when time has run out and there are outstanding incidents? If every test case and incident can be traced back to a risk, the tester can say, "At this moment, here are the risks of release". The

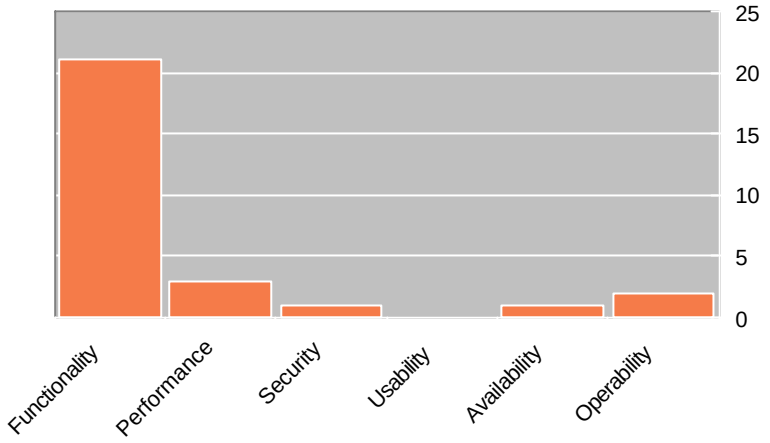


Figure 1 – Risks in a recent IT project

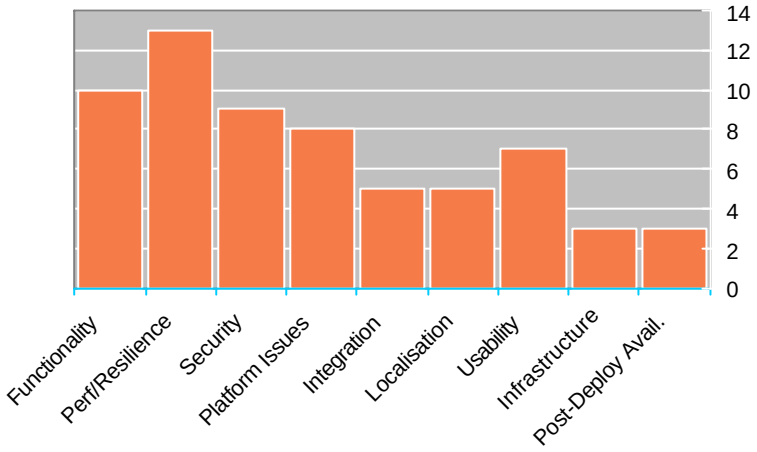


Figure 2 – Risks in a recent E-Business project

decision to release needn't be an uninformed guess. It can be based on an objective assessment of the residual risk.

Adopting a risk-based approach also changes the definition of 'good' testing. Our testing is good if it provides evidence of the benefits delivered and of the current risk of release, at an acceptable cost, in an acceptable timeframe. Our testing is good if, at any time during the test phase, we know the status of benefits, and the risk of release. No longer need we wait a year after release before we know whether our testing is perfect (or not). Who cares, one year later anyway?

1.6 The Shift in Project Risks

In a recent IT project (figure 1), we conducted an early risk assessment to help the testers prioritize the testing and we identified 28 risks of concern. Of these, 21 related to the functionality of the product. This is typical of a traditional system development. In a recent E-Business project (figure 2), we identified 63 product risks of concern., only 10 of the risks related to functionality. In all E-Business projects, the

issues of Non-Functional problems such as usability, browser configuration, performance, reliability and security dominate people's concerns. We used to think of software product risks in one dimension (functionality) and concentrate on that. The number and variety of the risks of E-Business projects forces us to take a new approach.

There has been a shift in risks associated with E-Business projects and this forces us to reconsider how we address risk in testing. We need to spend time identifying the risks to:

- **Help us prioritize what we need to do in our testing and**
- **Raise the visibility of the risks that are being taken by the project.**

2. Web Risks

2.1 Web Failures

To help us understand the nature of risks facing the E-Business project, here are a few examples of problems reported recently on both sides of the Atlantic.

- A book publisher found that its system sent out books with blank labels because the address information was not carried through to the dispatch department, although customer credit cards were being debited.
- Many web sites advertise prices that include postage and packing. There are several reports of sites that assumed customers would be based in the home country. However, anyone in the world can access these sites and place an order to be delivered half way across the planet.
- A recruitment consultancy invited applicants to post their résumés on the web site but didn't provide facilities to manage them. They received thousands, but ended up trashing them.
- An airline advertised cheap tickets, but didn't remove the web pages when the promotion ended. Customers continued to demand cheap tickets. One successfully sued the airline when they were refused the special deal.
- A health-food wholesaler offered trade prices on their web site, but failed to direct retail customers to their local outlet. How many folk want 100 boxes of dried apricots at once?

2.2 Web Site as a Retail Store

It is a convenient analogy to compare a web site to a retail store. Just like a high street store, anyone can come in to browse or buy. The store is open to all including minors, foreigners, hackers, crooks and terrorists.

If your site gives good service, customers will come back again and again. If the buying experience is good, they will remember and use your store regularly. However, they won't come back if:

- The door is locked (your servers are down).
- It is difficult to find the product required (the product catalogue is out of date, incomplete or inaccurate).
- They get easily lost in the store (the site is difficult to use).
- Service is slow; there is a queue at the checkout (performance is poor).
- They feel that they can't trust the staff (the site feels insecure).

One of the most disturbing findings of the BCG report (reference 1) was that many customers wouldn't give you a second chance. Where a user experienced problems at a web site:

- 28% said they stopped shopping at that web site.
- 23% said they stopped buying at that web site.
- 6% said they stopped buying at the company's bricks and mortar stores.

The message to E-Business companies is clear: Improve the reliability and usability of your web site and give the customer a comfortable, fast, efficient and secure buying experience.

2.3 But Many of the Risks are Outside Your Control

Here is a summary of the risks that threaten your EB system. Many are outside your control, but may affect the customers' experience.

Unlimited potential users

There are virtually an infinite number of users who could visit your web site, browse and buy your products. You servers may be perfectly able to support 1000 visitors per hour, but what if your marketing campaign is exceptionally successful? Could your technical infrastructure support 5,000 users, 10,000 or 10 million?

Dependency on Internet Service Providers (ISPs)

Many web sites are hosted on servers operated and maintained by ISPs. Can you rely on these companies to provide the support you may have grown use to in your internal data centers?

You have no control over client platforms

Most users of the web have 'traditional' hardware such as PCs, Macintosh or Unix workstations. However, many users of the Internet in future will use technology that is either available now on the horizon. WebTV, mobile phones, and PDAs are all being promoted as alternatives to existing browser based devices. Cars, fridges, Internet kiosks and many other Internet capable devices will be marketed in the next few years. Could your current web developments accommodate these future platforms?

You have no control over client configuration

Just like the new Internet-ready devices that have different screen sizes and resolutions, your web site may have to accommodate unusual user configurations such as very large or very small screens. Further, PCs and browsers can be configured with foreign character sets. Will your back-end systems and databases accept these unusual local character sets?

You have no control over client software

Visitors to your web site may have any operating system, any version and in any condition. If you have children who share your home PC, you'll know all about the problems caused by curious kids installing games, deleting files and fiddling with control panels. But beyond the state of the operating system, the user has a large choice of browser technologies. Microsoft Internet Explorer and Netscape dominate the market, but there are many, many other products available.

Visit http://browserwatch.internet.com for a comprehensive list of browsers. There are over 20 that run under Windows, some are non-graphic (e.g. Lynx). Other obscure browsers, such as Cello for example (http://www.law.cornell.edu/cello/), run on machines with extremely limited configurations (2MB of RAM on a 386SX-16!)

Your users may not have required plug-ins

Does your application demand that users have plug-ins? Common or not, not all users will have the required plug-ins (or versions) required. If your site depends on Adobe Acrobat, Flash! or an up to date version of the Java Virtual Machine to operate correctly – not all users appreciate being asked to download the software required. Some may refuse. In this case, will your application still work? How will your user know this? What happens if they refuse the download, or it fails to install?

The context of web transactions is a risk area

The HTTP protocol is stateless. What this means is that under normal conditions, every request to a web server is treated as an independent, autonomous, anonymous transaction. The server cannot link series of messages from a single client machine without some further information. Typical methods are cookies (see below) and hidden fields on HTML forms. Cookies can be rejected or time out. Web transactions can be interrupted through loss of network connection, a visit to another web domain or using the browser back and forward buttons.

Cookies can be a problem

Cookies are a common method for holding web site-specific data on the client machine. Cookies are typically small amounts of data written to the client machine's hard drive to hold personalized data or information that contains data to be carried through web pages as they are navigated. However, the mainstream browsers can be configured to warn the user when these are being requested by the web sites visited. Some users may allow and ignore them, some want to be reminded when they are invoked, some will reject them. Will your application work, if cookies are rejected?

Network connections

If your web site provides business-to-business services, it is likely that your customers access the web via a high-speed local network and the customer's site may have a direct connection to the Internet. In this case, the download speed of web pages, even large ones may never be a problem. However, business people working from home, and the vast majority of consumer customers will use a dial up connection and modem. The relatively slow speed of modems dramatically slows the pace of page downloads, particularly those containing large images. The perceived 'slow speed of the web' is usually due to over-large images that have not been optimized. This is a very serious consideration – slow web sites are unpopular and many people lose patience and leave them prematurely.

Firewalls may get in the way

There are some situations where the configuration of a customer's firewall can cause your site to be unusable or inaccessible. Whether your site is 'banned' by the customer network administrator, or the settings of your web servers conflict with the remote firewalls, your site may be unusable.

Anyone can 'walk in'

Just like a real shop, your web site is open to the public. Users can be experts or hackers but they are always unknown to you. Anyone on the web is a potential customer, but you don't see them and they don't see you. Some may be under-age. How can you tell if a 10 year old places an order with their parent's credit card? How would you know that you were selling alcohol to minors?

Usability is now a prime concern

Your site is unlikely to be documented. Your users are almost certainly untrained. If the site is hard to use, they'll go elsewhere. What is your experience of web sites? If you find it difficult to navigate, confusing, unpleasant to look at, time consuming, slow on more than one occasion, you may have just given up. Maybe you didn't even reach the functionality that was designed to help you place an order or find information. The user experiences good or appalling usability at the front door of the web site so usability is now a make-or-break issue for your EB developments.

Internationalization

Once your web site is up and running, anyone on the planet who has the technology can visit and use your site. If you are offering an international service or delivery this is a fabulous opportunity. However, people in foreign

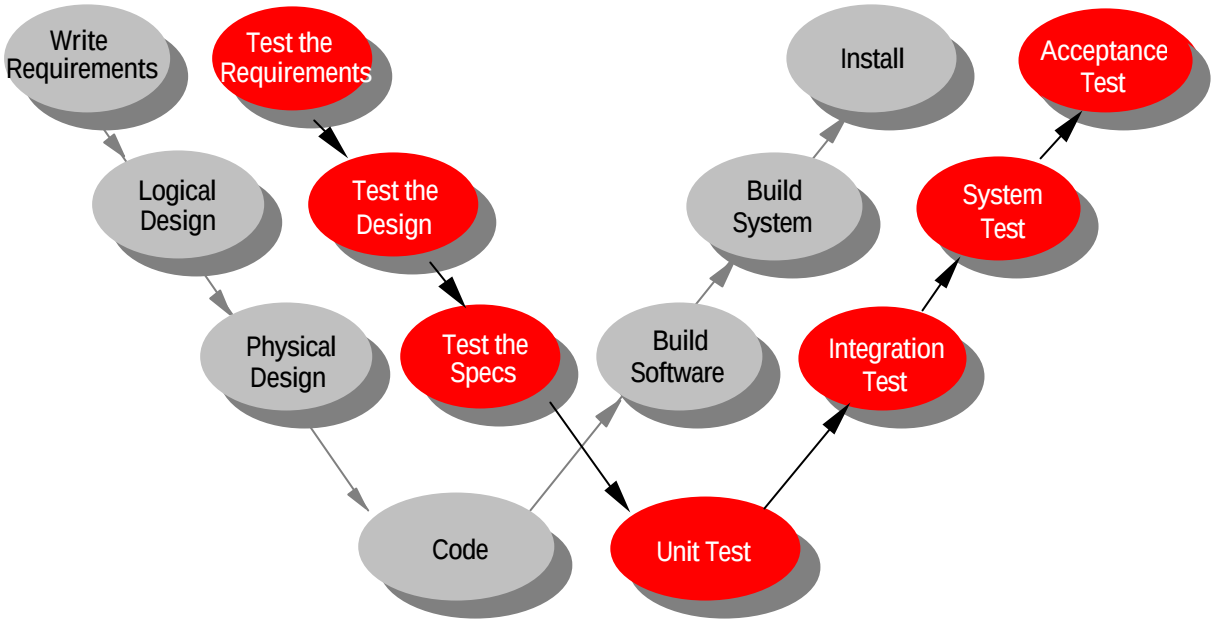


Figure 3. The W Model

countries may have to use their own language. Many will not appreciate 'obscure' address formats. They may wish to pay in local currencies. They may have their own local tax arrangements.

3. EB Test methodology and the W-Model

The EB test methodology that we will present in this paper is a natural extension of our W-Model view of testing.

To many software engineers and development managers, testing is the last phase of the development life cycle. However, waiting until after executable software has been built is the most costly and least effective way of performing testing.

The belief that tests can only be run against executable software is, frankly, wrong. There are test techniques that can be applied throughout the entire development life cycle, even as early as the requirements gathering stage. These techniques are not only cheaper than dynamic testing, but they also avoid wasting money on building the wrong system. Defect prevention is cheaper than defect correction.

Testing is least costly and most effective if it is performed throughout the whole life cycle, in parallel with every stage of development. This strand of testing in parallel with development is represented in the W model.

For those who are familiar with the V model, the W model is a natural evolution. The V model illustrates the layered and phased nature of software testing, but lists only dynamic test stages like unit and system testing. Some variations also include early preparation of test cases. The W model, by contrast, supports testing of all deliverables at every stage of the development life cycle.

The W-model promotes the idea that for every activity that generates a project deliverable, each of those deliverables should have an associated test activity. This is certainly not a new idea. Where it differs from the V-model is that it promotes both static testing of early document or code deliverables and dynamic test stages of software deliverables.

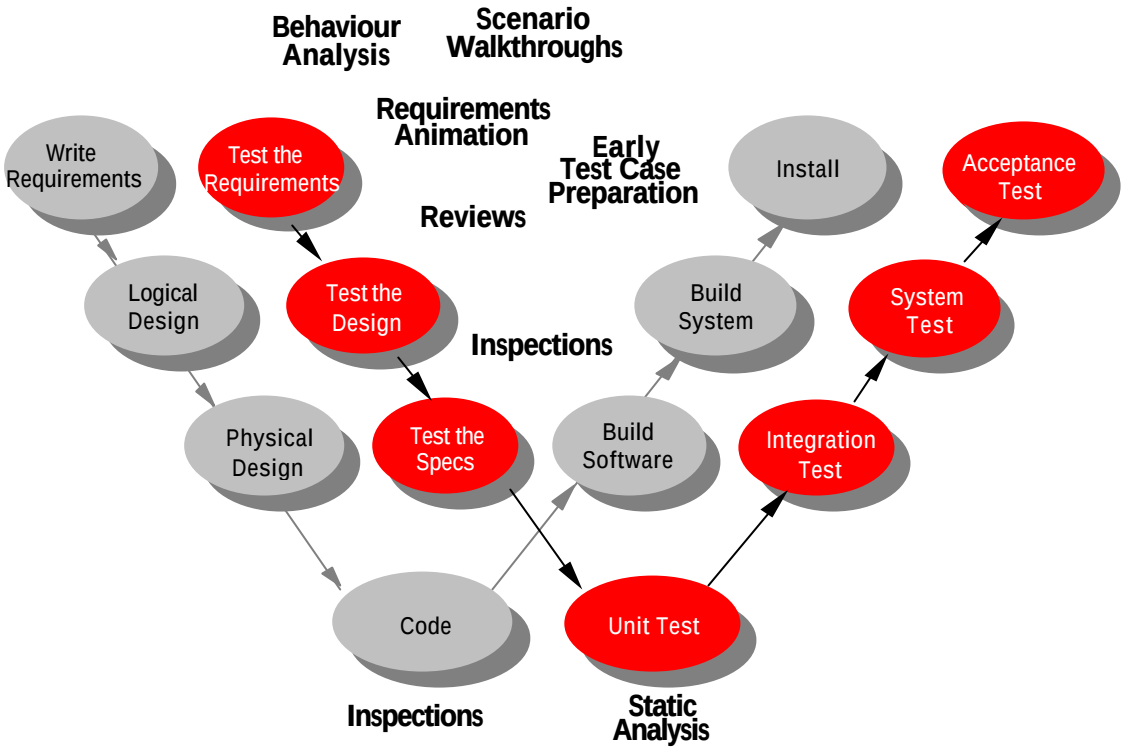


Figure 4 – Static test in the lifecycle

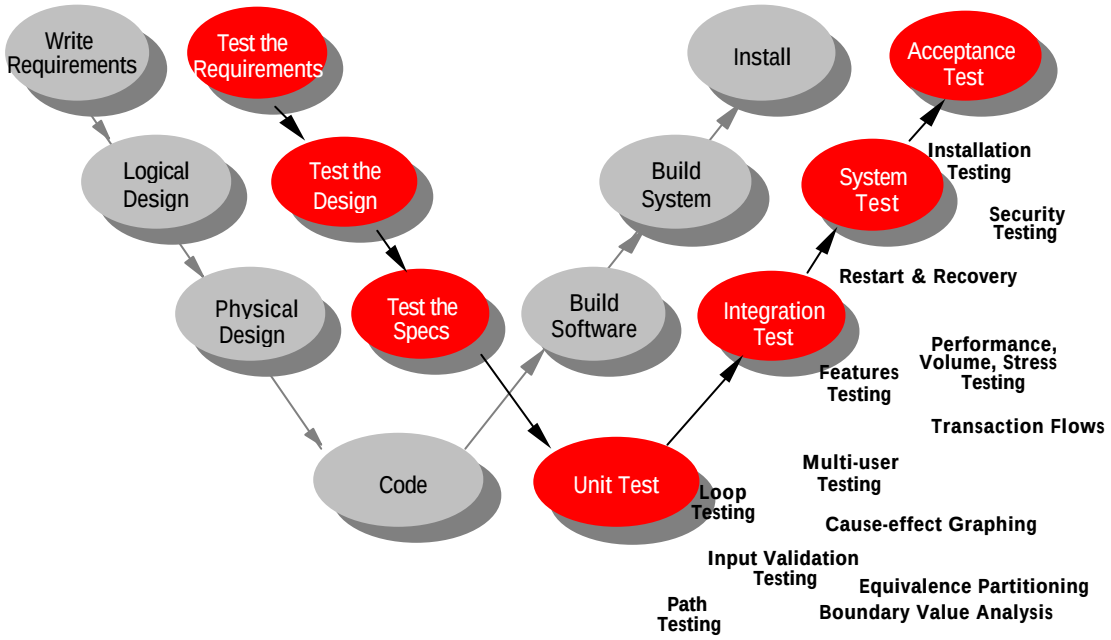


Figure 5 – Dynamic test in the lifecycle

Evolutif doesn't suggest that dynamic testing is obsolete, only that testing has to start right up front during requirements definition. The effect is a dramatic reduction in faults found in dynamic testing (the faults are removed earlier, before the wrong code is written), fewer faults found in live running, and certainty that the delivered system meets requirements.

*Total*Testing™ is Evolutif's implementation of the W model. It encompasses a range of static test techniques – like behavior analysis, requirements animation, scenario walkthroughs, document reviews and inspections – as well as dynamic testing and early preparation of dynamic test cases.

Figures 4 "Static tests in the lifecycle" identifies the most commonly available static tests that are useful in the testing of requirements, designs, code and test plans, perhaps).

Figures 5 "Dynamic tests in the lifecycle" presents the most commonly used dynamic tests that are appropriate for testing components, sub-systems or entire systems. This includes of course, the many non-functional types of testing.

The value of the W-model approach is that it focuses attention on all project deliverables. By matching the deliverables with product risks, the types of testing required to be incorporated into the test strategy can be quickly organized into the test process.

We have used the W-model approach to identify types of test for the various deliverables of E-Business projects and to build a collection of techniques structured into a generic testing framework that we believe should help testers address the E-Business testing challenge.

4. E-business test Strategy

4.1 Specialist Knowledge and the New Test Techniques

Testers need more than a minimal technical knowledge to plan, specify, execute and analyze tests of Internet based systems. The technical details of how the web works are easy to obtain – there are now hundreds of books covering HTML, CGI, Java, ASP and the rest of the many concepts, terms and technologies. The knowledge seems easy to acquire, but where does an E-Business tester's knowledge start? The more difficult question is, "where does it stop?"

The list of topics listed in are a minimum set of terms and concepts that the E-Business tester should be familiar with. Beyond the most basic issues, there are a number of more advanced topics that the tester should have at least a passing familiarity.

Some of the test techniques need a perhaps programmer's level of knowledge. This is an unfortunate consequence of working in a rapidly changing technical environment. In particular, the tester needs to know where sub-system (developer) testing stops and system testing starts.

4.2 Testing Considerations

Here are some broad issues that need to be considered when formulating the E-business test strategy. If testing is to keep pace with development, we need to squeeze the testing into the develop/release cycle.

Automation-focused approach

Design tests to be automated from the start. There won't be enough time to automate manual tests later. Plan all tests once only and reuse as regression tests. Where there is dramatic change in the user interface, consider using test drivers (below) to do the bulk of functional tests.

Consider working with developers to help them automate their tests and your own. If you are all using the same tool, the transition between development and system testing will be smoother. Developers can more easily reproduce the faults reported by later system testers if there is common test technology.

Developer testing

Consider tying in tests to developer code check-in/check-out procedures. Firstly, this will ensure that they do some testing and secondly, a

trusted set of regression tests will be maintained throughout the development.

Reference 2, Extreme Programming Explained, Kent Beck, ISBN 0-201-61641-6, promotes a set of disciplines for developers that encourage them to do significantly more and better testing. Pair programming, continuous automated testing, permanently maintained regression testing all combine to improve the development test process if (and it's a big if) the developers are prepared to work this way.

Consider asking the developers to use a variety of browsers (and versions) for their testing. Rather than using the company's chosen browser, encourage them to use Internet Explorer, Netscape or their favorite browser because using a variety of browsers can help to flush out compatibility problems across the different browser types. You may be able to eliminate later configuration testing!

Consider using test drivers

From the point of view of the user interface, Web applications are often relatively simple compared to traditional client/server or mainframe based systems. However, the user interface of Web applications is sometimes the very last thing to be developed and the integration of automated tools with browser-based applications is not always perfect, so consider using test drivers to execute functional tests of server based code.

The general recommendation with Web applications is to place as much functionality on servers, rather than the client browser. Where the Web is being used as a front-end to existing legacy systems, the testing of the back-end systems is as complicated as ever. Consider testing the back end functionality using the application programmer's interface (API). Test drivers are often very simple programs that accept test data, construct the call to the server-based code, execute the transactions and store the results for later evaluation. Some drivers can be enhanced to perform simple comparisons with previously run benchmarks.

There are a limited number of proprietary tools that can help, but you should consider writing your own in a few hundred lines of Visual Basic, Perl or C++ etc.

4.3 Using a Test Process Framework to Build Your Test Strategy

The E-Business Test Process Framework (Table 2) is an attempt to provide a framework for testers to construct their own test process from an assessment of risks faced by their project. (The risk assessment methodology that we use in our projects is not described in this paper). Based on an assessment of E-business product risks, the test types listed in the first column would be selected to address each risk in turn. The structure of the test process table is intended to help you to construct the detailed test stages from the test types.

Each risk can be transformed into a test objective to be addressed by the test types.

Test types are grouped into five categories

To address the risks of the E-Business project, we have identified 19 distinct test types. Each type of test addresses a different risk area. The purpose table 2 is to provide a framework for constructing your own project-specific test process. The test types are grouped into five main categories:

- Static testing
- Test Browsing
- Functional Testing
- Non-Functional Testing
- Large Scale Integration.

These reflect the nature of the test activities themselves. For example, static tests are those relating to inspection, review or automated static analysis of development deliverables.

Tests can be static or dynamic

Test types can be static or dynamic. There are actually only four static test types.

Essential Testing Priorities

Interesting aspects of E-Business projects are the extremes of project pace. In mature companies trying to establish a web presence, the existing IT department may be responsible for the development and they may follow their standard, staged development and test process.

However, some projects operate in Web Time. If you are working in a dotcom environment, where speed to market is the first priority, you may have to work under such pressure that the testing has to be squeezed into a few days or even a few hours. Release cycles may be daily. How can the traditional test process be fitted into this kind of schedule?

It is clear that the tester must be prepared to adopt test practices that complement these two very different scenarios. Our proposal is that a range of 'test priorities' be used to align with the project development process.

Table 1 Essential Testing Priorities presents one possible way of looking at the most important attributes of your web site. They are introduced here to give you an impression of the stark choices you may face in your projects. We suggest that you consider the following four levels of test priority.

The test types have been allocated a slot against the four test priorities mentioned earlier. Test types that fall into the Smoke testing column, are typically automated and should be the minimum set of tests considered to address the maximum number of risks in the shortest time. The tests that fall into the Functionality column are typically the tests that are most expensive and time consuming to implement and execute. The purpose of these columns is to help you to select the test types should be included in your test process.

Testing Priority	***When to Use***
1. Smoke testing	Can the application survive one user session without crashing? If you are under severe time pressure and have hours to do the testing, stick to smoke testing.
2. Usability	If your site is hard to use, the user won't hesitate to leave your site before reaching the functionality. There are several techniques involving inspection, review and heuristic evaluation.
3. Performance	If your site is usable, but slow, users may give up on the functionality and leave. Performance tests measure the speed of the site; flush out weak points, measures limits.
4. Functionality	Functionality isn't always the biggest issue. On basic E-Commerce' sites, functionality may be simple and relatively easy to 'get right'. Functionality is obviously tested in all projects, but often the functionality can be proven during development and user evaluation.

Table 1 Essential Testing Priorities

Five stages of testing

We have found that the test types do not always fit into the traditional unit, link, system and acceptance test stages. So we provide a test structure that has five test stages that group the test types by the clear technical groupings:

- Desktop development testing (broadly, what the browser executes)
- Infrastructure testing (what runs on the servers)

- System testing (of the complete system in isolation)
- Large Scale Integration (with other systems)
- Post-deployment monitoring (retaining automated tests to monitor live sites).

Tests can be automated or manual

The A and M entries in the table denote whether the test can be automated:

- M – the test can only be done manually.
- A – the test can only be done using a tool.
- A/M – the test can be done manually or using a tool.

Tools are covered later in this paper.

4.4 Guidelines for Using the Test Process Table

Not all test types are appropriate or possible in all circumstances. We are not suggesting that you must commit to doing all the test types identified in the framework. Rather that you gain a consensus view on the risks of most concern, match this to your development process and use the framework to help construct your test process.

The sequence of tasks required to construct the test process is as follows:

1. Identify the risks of concern with the project stakeholders, management, users and technologists and prioritize those that must be addressed by the testing.
2. For each risk to be addressed, identify one, or more, test types that can help to address the risk. Be sure to document those risks that will **not** be addressed by the testing.
3. Using the text of the risk description, write up test objectives (e.g. "to demonstrate that the Web application operates reliably in a range of configurations") to show how this risk will be addressed.
4. Assign the test type to a test stage and nominate the responsible party within the project.
5. Estimate the time required to plan and execute the test.
6. Review the scope of the testing (what is in scope and what is out of scope), test objectives, stage definitions, responsibilities and estimates and refine.

The framework is a guideline to be considered and adapted - tailor it to your environment.

4.5 Post-Deployment Monitoring

Post-deployment monitoring is done after implementation in a production environment. The reason for including this in your test strategy is that production E-Commerce sites can fail, but it may be that none of your prospective customers will tell you. They'll simply go elsewhere.

Further, degradation of site performance may not be noticeable day by day, but using a tool, statistics can give early warning of performance problems while there may still be time to recover the situation. Hence, we recommend that some selected automated tests be re-used to monitor the site. Services aimed at monitoring your site remotely are now becoming available, so this is an alternative worth considering.

4.6 Other Test Scoping Issues

Which platforms will your software support?

What browsers and versions will your Web site support? Do you need to worry about no-frames browsers? Do you need to consider non-graphic browsers? These questions are important because they help you to determine the scale of regression testing across a range of browsers and versions. Even if you will support only Explorer and Netscape, which versions of these browsers should you test to ensure there are no configuration problems?

	Test Priorities					Test Types Mapped to Usual Test Stages				
Test Type	***Smoke***	***Usability***	***Performance***	***Functionality***	***Static/ Dynamic***	***Desktop Development***	***Infrastructure Testing***	***System Testing***	***Integration Testing***	***Post-Deployment Monitoring***
Static Testing										
HTML testing	Y				S	A/M				
Browser syntax compatibility	Y				S	A				
Visual browser validation		Y			D	M		M		M
Test Browsing										
Link checking	Y				D			A		A
Object load and timing		Y	Y		D			A		A
Transaction verification	Y				S	A/M		A/M		
Functional Testing										
Browser page testing	Y				D	A/M				
CGI component testing	Y				D		A/M			
Transaction Testing				Y	D			A/M		
Application testing				Y	D			A/M		
Internationalisation		Y			D	A/M		A/M		
Non-Functional Testing										
Configuration testing	Y				D	M		A/M	M	
Performance			Y		D		A	A		A
Soak Testing/reliability	Y				D	A	A	A	A	
Availability					D					A
Usability		Y			S/D			M		
Security				Y	D		A/M	A/M	A/M	A
Large Scale Integration										
External links/legacy system integration				Y	D		A/M		A/M	
End to end functionality	Y				D				A/M	A

Table 2 – E-Business Test Process Framework

Propose a scope for the testing to be performed (and estimate the potential cost) and review that with the stakeholders, management and technical experts in the project. This will help to force a decision on what will actually be supported, and that may be much less than was envisaged by management. If you are going to test many of the browsers and versions, you will be committed to doing an awful lot of regression testing.

Consider and propose an alternative: that the developers work to the HTML standards supported by the chosen browsers. Tools can then be used to validate the HTML in your web pages. The number of browser platforms available is steadily growing. In the future, it looks increasingly likely that standards-based development and automated HTML validation will be a more practical approach than expensive regression testing across a large range of platforms.

Which Web conventions should be adhered to?

Although adherence to conventions is never mandatory, following Web conventions can make a big difference to your users. If your web site behaves differently to 90% of other sites, it might look good superficially, but may cause your users so much hassle they stop using it.

For example, will your application support users turning off graphics? Dial-up users regularly switch off graphics to speed things up. Will the site work if users reject cookies or the cookies time out prematurely? Must users always have the required plug-ins? As a tester, you need to establish what conventions apply to you can test against these, and ignore other contraventions. The best thing about web conventions is that they are widely documented and the browser defaults support them. Testing becomes straightforward.

The Web Accessibility Initiative (WAI), The World Wide Web Consortium, (www.w3.org) guidelines paper is an ideal document for testers to work against. Any contravention can be reported as a fault to the developers. In that way, the document is a good baseline (invaluable where other requirements may be lacking). So, get your project either to obey some or all conventions, and use a baseline to test against, or eliminate web conventions from the considerations in your test plan.

5. References

1. Winning the On-Line Customer, Boston Consulting Group, (www.bcg.com)
2. Extreme Programming Explained, Kent Beck, ISBN 0-201-61641-6
3. The Web Accessibility Initiative (WAI), The World Wide Web Consortium, (www.w3.org)

This paper is Part 1 of a two-part document. In Part 2, the details of the various test types introduced in this paper are described in detail. Automated tools that support the various test types are also covered.

APPENDIX A - ESSENTIAL INTERNET CONCEPTS FOR THE TESTER

HTTP	Hypertext Transfer Protocol - the way that client browsers and web servers communicate
URL	Unified Resource Locator e.g. http://www.evolutif.co.uk/subdir/index.html
HTML	Hypertext Markup Language Web pages comprise HTML tags and ASCII text
CGI	Common Gateway Interface – the mechanism by which HTML pages can execute programs that reside on web servers. CGI are Perl, C++ or other programs that execute on the server. Using HTML forms, users can send data and have it processed on the server
Cookies	Small quantities of data stored on the client's hard drive at the request of a web site. The web site can 'read' the cookie on a later visit. Used to carry data through Web transactions.
How a web page works	• You select a target URL in your browser by using the URL window or clicking on a link. • The browser sends a HTTP request for the HTML file to the site defined by the URL. • The server processes the request; sets up a connection, sends the HTML file and closes the connection. • Browser interprets the HTML file and displays it according to HTML standard.
JavaScript/VBScript	Programming languages (cut down versions of Java and Visual Basic) used to provide functionality within a web page on the client or on servers.
ASP	Active Server Pages – a Microsoft technology that allows VBScript or JavaScript embedded in HTML pages to be executed on servers.
Applets	Programs normally written in Java that are downloaded from servers and executed within the browser.
Servlets	Programs normally written in Java that are executed on servers.
The web is stateless	After processing an HTTP request, the server forgets everything about the transaction. Developers need to maintain context through business transactions so use cookies, hidden form fields to do this.

Companies

Corel and Inprise/Borland divorce

Before the marriage was even celebrated, Corel and Inprise/Borland announced in May that theirs merger agreement has been terminated by mutual agreement. Respecting a three-year confidentiality agreement, both companies will not comment further this divorce, but the end of the Linux market's overhyping and the subsequent drop of Corel's stock price could be a big part of the explanation.

Considering the current problems of Corel, we think that this failure is good for Inprise. It will allow the company to focus again on serving the interests of the software development community.

$9%£ Numbers

Have you failed enough?

"Question: How many datawharehouses do you have?
Answer: We have had eight.
Question: To what do you attribute so many warehouses?
Answer: Seven mistakes...."

Source: The Meta Myth, The Standish Group, www.standishgroup.com.

The text adds that the user still realized substantial benefits of the mistakes that occurred over a number of years. Yes, we have to admit imperfection... and learn from it to improve!

In Others' Words

Do consultants suck?

"**You're way to enthusiastic, and you dress funny**. This is not your fault. But you can't blame us for feeling annoyed when you show up all bushy-tailed, and you tail is in $1'200 blue pinstripes and we're in brown. Or you're wearing a power tie we haven't seen in ten years. Or your hair is too short. Or too long. There's no question you're from over the border, chum.

The boss likes you far too much. You're an institutionalized suck-up. You're there to make him look good when he shouldn't. Does he listen to the guys who are in the trenches every day? No, he does not. [...]

You don't listen. You look like you're listening, but you're not, you're just waiting to talk. Because you've got a lot to say. You've been waiting since they poured all that stuff in your ear at Wharton to download all your knowledge, and this is your chance.

The solutions you propose to our problems were baked fresh in the 1990s and have been used every year since then. You have your spiel, and you stick to it. Why not? It's why you were hired. This is particularly true of McKinsey guys, who seem to come out of one Stepford factory with the entire program ingrained on their internal database.

You don't really know anything about our business or our company. You've been here for five minutes! Last week you were flogging cheese processors in Wisconsin. The week before, it was a private hospital in Omaha! Quick – who are we again?

Your solutions generally evaporate like fog over a mountain lake. You have your meetings. You "capture" all this stupid stuff on big sheets of paper you tack onto the wall. You rush around with overheads and Magic Markers. We get real creative for a while. Then you go away to put together the results. A few weeks later the report comes. It has all the resonance of old love letters written by another person. The report is filed and forgotten.

You cost a lot. We're dealing with 4% raises, and you cost, what... $25'000 a month? More? How much do you cost, anyhow? We don't know. A few years ago we hired a consulting group on what our newly merged corporation call itself. At the time we were Acme Inc. Several hundred thousand dollars later, we were Acme Corp. Ha!

You alienate us from our bosses. You sort of have to. If staff can solve internal problems, that's the end of you, isn't it, Charley?

You're here to get us fired. We know it. You know it. What you're selling might be Quality or Excellence or Productivity, but you're to figure out ways to kill us, or spend money on something other than us. [...]

And it could be different. This, in the end, is the most tragic and galling part. There are some good practitioners of what you do. They come in with highly specific knowledge and expertise after years of fighting in the trenches. They help the inside guys identify a problem and fight against senior management to get things done. They have ideas but impose none. They come like the blessed rain and are gone like the wind. And they are loved.

No one just out of school can be this person, nor can some ancient mariner ten years into retirement. If you aspire to be a consultant, I wish you well. You've got a life-time of preparation and study before you. An in the meantime? Hey. You're smart and resourceful. Why not get a job?"

Source: Stanley Bing, "Why Consultant Generally Suck", Fortune, May 29, 2000.

There are a lot of consultants in our world. Some funny thoughts from Mr Bing for internal staff and consultants... that have a sense of humor :-]

Managing conflicts

"Despite the self-reliant focus of our formal training as software engineers, our profession entails significant human contact. Most likely, we will experience some conflict with coworkers, bosses, subordinates, customers, or suppliers. Constructively resolving these differences can make or break our ongoing relationships.

[...] The first and most significant strategy point is to take steps to avoid becoming defensive or putting other people on the defensive. If we go on the defensive and attempt to justify ourselves when we perceive a verbal attack, it simply escalates the conflict. A more helpful approach is to do the following:

- stop and let the other person talk;
- actively listen and gather more information about your transgressions;
- echo the problem back to the attacker until the agree that we've characterized the

situation from their perspective, even though we might not agree with them; and

- look at the problem from their perspective, and if you see how they might have a legitimate concern, say so.

[...] A second conflict strategy is to describe the offending behavior rather than evaluating the person. When people feel evaluated, they usually go on the defensive, and this is what we want to prevent. For example, it might be more productive to say, "It feels like you're not listening to me when you work at your computer while we're talking," than to say, "You're really rude." The former describes the troublesome behavior rather than judging the person.

A third strategy is to give "I messages" instead of "you messages." With I messages, the focus is on speaking your reality rather than trying to impugn the value, motivations, or cold bloodedness of your colleague. With you message, it is far too easy to trigger defensive reactions.

[...] A forth conflict strategy is to focus on the future rather than dwelling on past transgressions. Get a clear understanding of the problem and propose a way to avoid the problem in the future.

[...] Here are a few other constructive tips for dealing with conflicts:

- Avoid using the absolute terms "always" and "never" in describing behavior.
- Avoid pointing at someone in an accusatory way. It's guaranteed to inflame defensiveness.
- When someone approaches you with a conflict, and you can't engage at that time, tell the other person exactly when you'll reenter the discussion. Otherwise, the other person will feel dismissed and be further angered.
- If possible, work through conflict in a private setting. People tend to resist changing their positions if there is an audience."

Source: Karen Mackey, "Conscious Conflict", IEEE Software, September/October 1999

Coming next in Methods & Tools

- Risk-Based E-Business Testing – Part 2
- How to Sponsor a Successful Project
- Testing Client/Server Applications

Classified Advertisement

Advertising for a new Web development tool? Looking to recruit software developers? Promoting a conference or a book? Organizing software development related training? This space is waiting for you at the price of US $ 20 each line. Reach more than 6'000 web-savvy software developers and project managers worldwide with a classified advertisement in Methods & Tools. Without counting those that download the issue without being registered! To advertise in the classified section, to place a page ad or to become the distribution sponsor of the next issue, simply send an e-mail to franco@martinig.ch

METHODS & TOOLS is published by **Martinig & Associates,** Rue des Marronniers 25, CH-1800 Vevey, Switzerland Tel. +41 21 922 13 00 Fax +41 21 921 23 53 www.martinig.ch
Editor: Franco Martinig; e-mail franco@martinig.ch ISSN 1023-4918
Free subscription: www.martinig.ch/mt/index.html