Estimating Test Effort – From Billiard Balls to Electron Clouds

Finding the Model That Is Right for You

By Trevor Atkins **August 12, 2008**

More than 2000 years passed between the Greek philosopher Democritus' proposal in 5th century BC, that all matter must be made up of invisible particles called atoms (Greek for uncuttable), and when Dalton proposed his billiard ball model for the atom in 1803. Then in the space of the next 100 years, Dalton's model was progressively evolved from that of a solid homogenous sphere of constant density to the much more sophisticated electron cloud model developed by Schrodinger and Heisenberg, based on their probability functions that describe the regions or clouds where electrons would most likely be found around a nucleus.

However, in every day life we likely think very little of the atomic models developed and refined over the last 200 years. And even if we were professionally required to know about the properties and behaviours of atoms, there would certainly be a significant difference between the models we would select as sufficient, given the likely diversity of our purposes.

This is similarly the situation with the effort estimation of test activities in software projects. The degree of detail required or even possible in an estimate depends on the purpose of that estimate and the data available at the time it is prepared. Likewise the approach or calculations you use to arrive at your estimate will also be necessarily based on this context, leading you to a particular effective form or model for your estimation process – though ideally with significantly faster progression than the evolution of the atomic model.

WHITEPAPER

COPYRIGHT 2008 SILVERPATH TECHNOLOGIES INC.

Each of us has very likely had to do an estimate in the past, whether it was for a set of assigned tasks, for a project, or for the budget of an entire organization. As a tester, the question is commonly presented as, "How long will it take to test this product – and what resources will you need?" and then the person asking stands there, likely somewhat impatiently, waiting for your answer.

Warning – What you say will be used against you at a later time! Estimation is a black box (magic?) process for many organizations and all the more so for their testing activities. The inputs used, the process and end results are therefore open to debate or worse – to un-discussed misinterpretation.

In Steve McConnell's "Software Development's Classic Mistakes 2008" software estimation is mentioned more than once as an area of significant impact. Of the "Top Ten Almost Always", confusing estimates with targets was ranked #5 and shortchanging quality assurance was ranked #3. Overly optimistic schedules was ranked #1 of that list.

> *"…some organizations actually refer to the target as the 'estimate,' which lends it an unwarranted and misleading authenticity as a foundation for creating plans, schedules, and commitments."* – Steve McConnell, "Software Development's Classic Mistakes 2008"

Capers Jones noted in Assessment and Control of Software Risks that most projects overshoot their estimated schedules anywhere from 25% to 100%, but some few organizations have achieved consistent schedule-prediction accuracies within 10% and 5%.

What are these organizations doing different? Are they the quantum physicists of software? Perhaps, but maybe they have simply taken the time to put in place a formal and repeatable estimation process tailored to the types of projects they undertake – something we all can do. Of course, we will not blindly copy these organizations' approach. We simply need to employ similar practices in a framework of continuous improvement while recognizing that a single approach will not fit all of our needs all of the time.

What is an Estimate?

Agreeing to what is an estimate is a necessary first step in defining how to arrive at an estimate. The following definitions provide important cues as to what we need to include in defining our estimation process.

> *"An approximate calculation of quantity or degree or worth."*
> – http://wordnet.princeton.edu/perl/webwn?s=estimate

> *"An assessment of the likely quantitative result. Usually applied to project costs and durations and should always include some indication of accuracy (+/- x percent)."*
> – Project Management Institute, PMBOK 3rd Edition

It is important to realize that estimates are not commitments or promises (targets), but are predictions founded on situational information. Carefully accounting for and describing the situational information allows the estimate to be considered in its proper context and updated as changes occur, with the resulting impacts more easily assessed.

Challenges in Estimation

Developing your estimation process can be quite straightforward, but also there challenges to consider. One challenge is the fact that though the test effort is directly proportional to the size or scope of the implementation in terms of requirements and behaviours that need to be verified, there are also other factors to include like:

- What is the type of project (New Development, Patch Release, Legacy Modernization, COTS Customization, Enhancement Release, etc.) and therefore its quality requirements and goals?
- Do you have the test lab infrastructure and tools you need?
- Do you have your team assembled? Are their skills and experience applicable?
- What is the project methodology and how does your test approach mesh with it?

However, the biggest influence on an estimate's acceptability is how well it is received by other stakeholders. How many times have you worked on an estimate for the test effort and presented it to the project manager or another senior stakeholder only to have them ask you to change it, or they change it themselves. Maybe you have done the same to someone else.

Often stakeholders want to change the estimate to meet a target of theirs:

> *"'You've overestimated this project by a third; go away and cut it by a third.' The division head had 3 or 4 useful ways to cut the project down by a third, but I kept saying I couldn’t do it and the project was killed. I walked out of there wondering what had gone wrong. I did what was probably the best estimate that had ever been done in that company, and the project was killed because of it."* – Daniel Galorath, A CAI State of the Practice Interview

Aside from extraneous things like politics, funding profiles and meeting predefined targets, an estimate may also get changed because of a lack of confidence in the validity of the estimate; the estimation process, the inputs, and/or assumptions upon which the estimate was founded.

In these cases, the stakeholder or approving manager may wish to increase the estimate for the sake of "safety" or may wish to cut it back to remove "unnecessary" padding and put a "healthy" pressure on the team. Any adjustments of this type are likewise made without analytical foundation, but rather with a hunch or "gut feel", collectively resulting in a poor estimate.

Letting poor estimates persist can affect project outcome in two ways:

- **Perfecting:** If the project is over-estimated Parkinson’s Law will apply. The project effort will somehow always expand to use the available schedule and/or budget, even if it could be done for less.
- **Rework:** The more dangerous scenario is to under-estimate the project. In this case, the project will typically only recognize that it is underestimated at the latter stages in the project cycle. At which time, it is usually too late to make real changes. Often people are added to the project team in an attempt to solve the problem, but adding people to a late project without changing anything else will only tend to make it later.

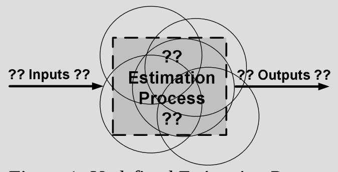

Figure 1: Undefined Estimation Process "Principles and Components of Successful Test Team Management"

Debatable Results (where there are two or more views and each proponent believes they are right)

- With no analytical process, the inputs, process, and output are all open to debate or misinterpretation
- Estimation is reduced to guesstimation

Unreliable Process

- Without a repeatable process, there can be little confidence in the latest estimation results
- It doesn't matter how precise your estimate was for the last project. Can you repeat it with similar success on this project?

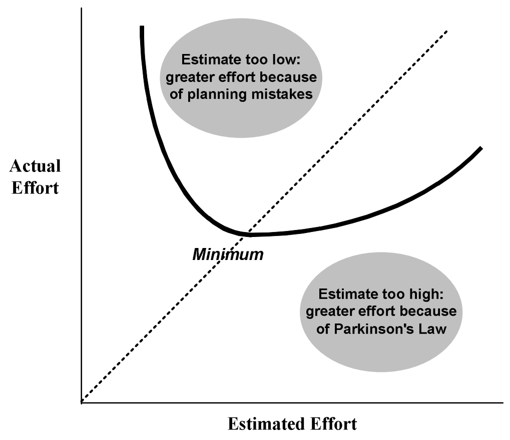

Figure 2: Impact of Poor Estimates
"Principles and Components of Successful Test Team Management"

It is often the case on a project with this type of estimation climate that testing efforts are approached in the two following manners:

- **Available Time:** In this case this test effort is not based on any quantified timeframe. Testing simply starts when the code is determined to be ready and continues until some pre-decided criteria, usually the release date, is reached.
- **Percentage:** In this case the testing effort is estimated by simply taking a percentage of the development time. The development effort is first estimated using some technique that draws upon Lines of Code or Function Points, perhaps even using a commercial tool, and then the test effort is allocated using a pre-determined ratio.

In both of these cases, pressure is put on the test team's strategy to uncover all the significant defects within the given targets or constraint box, or risk being determined to be ineffective – But how were the targets arrived at? The test team did not have a chance to provide input into determining the effort required for sufficient testing versus the quality requirements, or to make visible the risks of the choices before they were made.

See Figure 3 for an illustration of how formalized estimation is able to present certain options to decision-makers and clearly highlight the impact on either cost or schedule of choosing the fastest or cheapest. You can use other similar analysis to make visible such things as the impact of involving testing earlier or later in the project cycle versus achievable test coverage or risk to system quality.

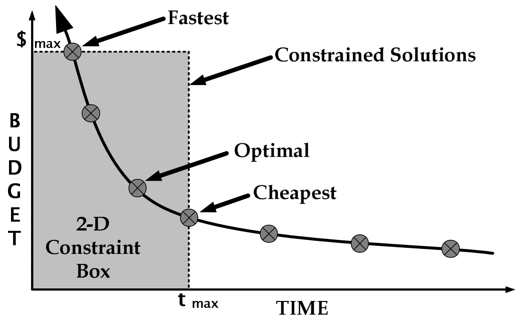

Figure 3: Constrained Solutions
"Principles and Components of Successful Test Team Management"

Good Estimate, Poor Estimate

How accountable are you for the estimates you give? How confident are you in an estimate that someone else gives to you? Would you stake a year's salary on it? A month? A day?

> *"It is very difficult to make a vigorous, plausible, and job-risking defense of an estimate that is derived by no quantitative method, supported by little data, and certified chiefly by the hunches of the managers."* – Fred Brooks, Mythical Man Month

A good plan is a cornerstone to project success. However, as we have seen, that plan can be compromised from the beginning if based on a poor estimate. But what makes a good estimate good?

As in the case of describing quality, the most useful description depends on the audience and their point of view or needs. As we are talking about a "quality" estimate, it is appropriate to use those same aspects of elegance, correctness, and fitness-for-use here. If we liken elegance to sophistication or complexity, and correctness to precision or perfection we can immediately feel the increasing effort required to pursue these aspects as the dominant factors in our estimation process. Fitness-for-use or practical-for-purpose gives us a framework of thought that allows us to be both flexible in the approach and demanding of the outcome, in the sense that the estimation process must be reasonable to undertake and that the result it provides must be useful for our purpose. Adding "repeatable" to our description of a good estimate will result in a process that we can rely on to meet our needs again and again.

Remember that all of these aspects will be in balance with each other; sophistication, precision, practicality, and repeatability. However, the exact mix of proportions will be determined by your own needs, priorities, and the effort you are willing and able to invest. See Figure 4 for an illustration of the non-linear relationships expected when searching for that balance.

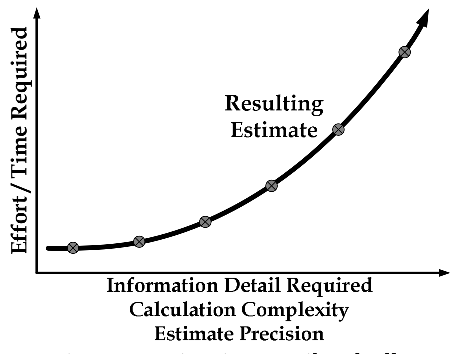

Figure 4: Estimation Detail and Effort

There can be many approaches to the effort estimation of a given project and some will fare better in different circumstances than others; what is important is to have a strategy that allows you to approach the task in a systematic manner with a defined yet flexible technique – where further research and experimentation will improve the methods used to arrive at increasingly valid estimates.

The following are some of the expected benefits or by-products of a good estimate:

- Providing visibility of options to stakeholders
- Providing a reliable foundation for project planning
- Providing an agreed repeatable process for estimation that takes the debate away from the output and to the inputs, e.g. prioritization of features and trade-offs of scope or quality against time or resources.

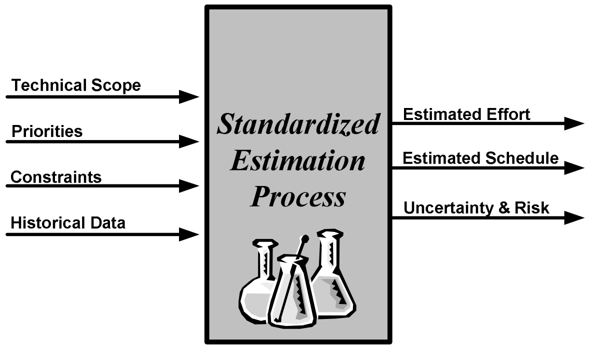

Figure 5: Standardized Estimation Process
"Principles and Components of Successful Test Team Management"

Components of Calculating an Estimate

Whatever your estimation model will look like you will need to get a starting number around which you can base your estimate. The number of tests* needed to fully exercise the functionality included in the scope of work can provide this foundation.

As requirements and tests are directly proportional, a well-defined requirements specification, being a set of structured documents that follow certain standards in authoring and describe the scope of the system to be implemented in a well enumerated fashion, is the best source for producing an estimation of the number of tests. These qualities of the requirements allow the system to be sized using a variety of techniques that can quantify the system's functionality, its complexity, and other non-functional requirements. If you don't have requirements documented per-se, you will need to determine the effective number of tests to be performed in some other manner.

*** Tests:** A test must be defined to be of relatively uniform size. For example if one test case contains three things that will be verified and another contains one, then together they should be considered as four tests for the purposes of estimation.

The following provides a sample outline for just one approach to constructing and refining an effort estimation model for testing activities.

Basic Estimation Calculation Model

The basic elements to consider when performing an estimate for test effort is the:

- Size of the system in terms of tests or verifications to be performed
- Level of productivity the resources that will be performing the work in terms of the amount of work completed in a fully productive hour

For our basic model let us consider the following test activities where each is described in terms of the number of tests:

- Identifying and documenting the tests as test cases/scenarios
- Executing the tests and recording the results (single test cycle)

! Because of their complex nature and similarity to a full software project, automation and performance testing efforts should be considered and treated as their own projects with respect to the planning and estimation of the effort to create the executable scripts (i.e.: a separate effort).

Example calculation: For the purposes of illustration, let us assume the following:

- The scope of work is 350 requirements
- The average conversion rate of requirements to tests is 1:3.5
- The tests can be documented at a rate of 7/hour and manually executed at a rate of 12/hour

This gives us the estimated effort for documenting 1225 tests as 175 person hours and the execution of a single manual test cycle as 102 hours.

A simple refinement to the above would be to prioritize the tests and estimate the effort to document and execute each group of tests. This will allow you to present options regarding how much testing can be completed within given constraints of time, resources, and/or multiple builds while making sure the more important tests are always included first.

With the size of the scope of work and the average productivity level of the team by each basic activity, we are able to quickly produce an estimate for the testing effort that is directly proportional to the work being undertaken. Of course this level of estimation is quite coarse and does not account for many of the "overhead" activities, multipliers or uncertainties related to performing software testing. However, for some project types, like patch releases, you may not need to do more than the above. In that case this basic "billiard ball" estimation model would be "fit-for-use".

! Until you have all your tests documented and prioritized you won't know for sure what the breakdown is, but that is one place where you can use assumptions when estimating.

Extended Estimation Calculation Model
To help you get to the next level of your estimation model, consider the phases of a typical software development lifecycle (or iteration). In each project cycle there are three stages or groups of test activities; test planning, test design, and test execution.

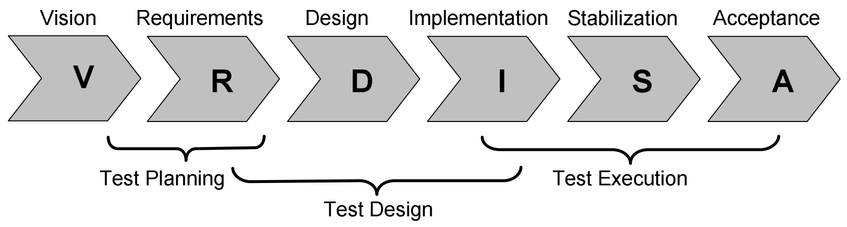

Figure 6: Testing Activity Groups
"Increasing Test Effort Estimation Effectiveness"

To aid you in remembering what to include in your estimation process, think of each group and their related activities. Which will you need on your project? A few of the more common examples are given below:

- Test strategy
- Requirements review
- Design review
- Requirements traceability matrix
- Test data preparation
- Defect logging
- Analyzing/reporting results/metrics
- Team management and coordination
- Meetings

"Projects that are in a hurry often cut corners by eliminating design and code reviews, eliminating test planning, and performing only perfunctory testing…This often results in the project reaching its feature-complete milestone but then still being too buggy to release." – Steve McConnell, "Software Development's Classic Mistakes 2008"

- Away time allowance (vacation/sick)

These "overhead" factors to test execution depend on the extent of upfront planning and early involvement testing has in the software development lifecycle.

Extended Estimation Calculation Model with Multipliers
How many builds are planned to be delivered to testing? How many times are you going to execute a given test in the life of the project? Derive this from the project's build plan.

Other factors that can cause work or similar work to be undertaken more than once are:

- Test artifact reviews, rework, and updates
- Defect fix verification testing
- Compatibility and configuration testing
- Localization and internationalization testing
- Performance testing and optimization
- Other non-functional test types

! Simply by asking for/requiring strategic planning artifacts from the project team, such as the build plan, testing is adding value and will also have more useful information for integrating into its own effort estimate.

You can include these activities in your estimation calculation as a percentage or multiplier of the original effort to which they are most closely related. Keeping each factor on a separate line in your calculation will help make the impact of each item on the overall estimate visible.

Example calculation: For the purposes of illustration, let us assume the following:

- 1225 tests are required to verify the 350 requirements
- The effort to document these tests is 175 person hours
- The effort to peer review these tests is 10% of the original effort, or 17.5 hours
- The effort to incorporate the feedback of the peer review is 20% of the original effort, or 35 hours

This gives us the total estimated effort for documenting 1225 tests as 227 person hours.

! Predict defect counts as a function of the number of tests executed in a given project cycle. This will allow you to estimate effort for both defect logging and defect fix verification testing.

Considering Uncertainty in Estimation
Counting the requirements and applying formulae is certainly the basis of the models above, however there are a number of uncertainty factors and influences to be considered when examining the project for test effort.

- Are requirements, designs, and plans available and are they clear, concise, and accurate?
- Do project stakeholders have realistic expectations in terms of schedules versus scope? (e.g.: a "rush job" or time to "do it right")
- Are there clearly defined milestones during the project for testing? (e.g.: code complete, code freeze, release candidate)
- What is the expected quality of the code at each phase or milestone?
- How well managed are the change control processes for project and test plans, requirements, designs, and code?
- Does the project team have the skills, experience, and tools needed for this project?
- Is the project team established or is there expectation of ramping up or

turnover during the life of the project?
- To what extent can the project re-use test assets from previous projects?
- What is the required investment in the test lab infrastructure and tools set-up and maintenance?
- Do all the supported configurations need to be tested to the same degree? Do all components?
- What percentage of the tests is sufficient to confirm the functionality is working as intended for a localized version of the system?

All of these questions can create uncertainty and a perception of weakness in the estimate if they remain unanswered. Of course these questions and many others are not always answerable at the time of producing the estimate.

To address this you can take two approaches:
- **Fudge Factors:** This approach assesses the impact on the estimate in the case that the question is eventually proven to have a negative effect on the planned approach and associated effort. Assigning a "fudge factor" or contingency adjustment to the base effort increases the overall estimate for that activity such that your confidence of being able to handle the untoward event is sufficiently increased.
- **Assumptions:** This approach states assumptions as truths to aid in producing the estimate. In other words, for the estimate to be valid the conditions stated as assumed must prove to be the case. In this way you can make visible the situational information that you need for your estimate to hold true. Varying these assumptions is also one way to make visible different options or solutions and the testing impacts of choosing one over another.

Example calculation: For the purposes of illustration, let us assume the following:
- The scope of work is 350 requirements
- The average conversion rate of requirements to tests is 1:3.5
- The total estimated effort for documenting 1225 tests is 227 person hours
- The requirements volatility and associated scope creep is 12% during or after the test design phase

This gives us the total estimated effort for documenting 1372 effective tests as 255 person hours.

The potential impact of most of the above uncertainty factors can be mitigated through upfront planning and investment. But if you don't have the time or the ability to do so, make sure to take this into account when considering the certainty of your estimate. When used in combination, the two approaches above will give you the most balanced estimation result upon which to base your next planning steps.

Estimation Tips

Some tips to keep in mind for your next estimate:

- The response to a request for an estimate should be “Let me get back to you on that.” Then, take the time to work through the estimate and get it down on paper, rather than just tossing one out the instant you are asked for it.
- When providing an estimate, present the numbers in a range or as a central point with +/- values based on specific risks and uncertainty. Single-point estimates imply precision where none exists.
- When you de-scope to get your estimated effort to fit within the constraint box, keep the “customer” in the loop even if that is just your manager. You don't want to surprise someone with: “Here is your application on schedule on budget, but we had to reduce the functionality tested by 50%”.
- Estimation of testing effort is an iterative process where the test strategy can only be finalized when the proposed effort, schedule and budget estimates are approved. But of course the test strategy has a significant impact on the estimate, so work on these simultaneously, updating each as the project progresses.

! Include in your test strategy a balance between your formal test cases/scenarios, checklists, and ad-hoc/exploratory testing.

- There isn’t just one time to do an estimate. As the project progresses, the estimate can be refined for the activities that remain. The estimated effort for those activities may or may not change but the confidence level in the estimate should rise. Make the impact of any changes visible for good or bad so the appropriate informed decisions can be made.
- If you can produce one estimate, you can create several, based on different, clearly stated, assumptions or conditions. Review and modify these options to find the optimal solution for the project.
- Have those closest to the work do the estimating to increase accuracy of the estimate and also to create a sense of ownership and commitment in the numbers provided.
- Keep the estimate modular and decoupled so that changes to assumptions, a new input, or project phase can be easily accommodated.
- Leverage historical data in the estimate. Objective reality helps avoid overruns from overly optimistic thinking.
- Brainstorm with the team to avoid overlooking uncommon activities and underestimating the related size of the effort (reviews, rework, SCM).
- Have more than one person do the estimate. Discussion of differences in numbers can make visible and clarify assumptions or advantages of approaches. Compare, contrast and converge to get the best from each.
- Buffer or contingency time helps cope with the unknowns and the unexpected. Base the buffer on quantified risk analysis whenever possible. In the absence of granular detail you can generically buffer against risks, e.g.: by scheduling people for only up to 80% of their availability or simply add a factor of 20% to the estimate. As your estimation technique becomes refined you will be able to break this number down and reduce the relative size of this sort of fudge factor.
- Document the estimation process so you and your colleagues can better review/defend your current estimate as well as leverage the process next time.

Conclusion

Estimation process improvement presents a significant opportunity to get more success out of your software projects. And just as it is critical to offer something more than an off-the-cuff answer for the development activities, it is as important to know how to produce a good estimate for the testing effort portion of a project. We can achieve this by leveraging many of the same principles that are applied as best practices in general estimation, while developing a specific framework for predicting test effort.

As more understanding of what influences your estimates is gained and more iterations of each estimate are completed, your model will increase in sophistication; similar to the increase in understanding gained between the billiard ball and the electron cloud atomic models. The approach outlined above offers the ability to approach the task in a systematic manner with a defined technique and supporting data. This is a significant practical advantage over ad-hoc techniques or guesstimation, and allows the lessons learned from further experimentation to be applied as improvements to the methods used, thereby driving increasingly valid estimates. Keep focused on maintaining the "fitness-for-use" aspect of a good estimate and you will experience the benefits.

Don't wait until your next project to work on your estimation model. Start now, and try it on next week's or next month's tasks.

About the Author

Trevor Atkins is Principal Consultant with Silverpath Technologies (www.silverpath.com). Most recently he was a Regional Director of Quality Services with UST Global and before that he was a founder and the Vice-President Operations of QA Labs, the largest independent software testing company in Canada.

After obtaining a degree of Applied Science in Electrical Engineering from the University of British Columbia, Trevor has been involved in all aspects of hundreds of successful software projects for the last 12+ years, and is dedicated to the design and improvement of quality processes for use across projects and organizations.

About Silverpath Technologies Inc.

Silverpath Technologies provides results-centric consulting and training services targeted at improving the effectiveness and efficiency of quality and testing activities across the software development lifecycle. Visit http://www.silverpath.com for more information.