Book excerpt from: "Realistically Estimating Software Testing Projects"

CALCULATING YOUR TEST OVERHEADS

Common Areas of Omission on Test Projects
Part 1: Directly Related Tasks

Do you remember receiving your very first paycheck? That initial happy feeling which turned to confusion when you saw that your expected $100 payment had shrunk to $28 after oodles of deductions? Welcome to the world of overheads.

When does a two-hour testing task take two days? When a manager, who is not very close to the work and is informal in his or her analysis of situations, casually observes only the visible part of the work. The conspicuously visible part is a tester tapping on the keyboard, running the test cases, and muttering occasionally in dep thought, excitement or bewilderment. (Testers can get passionate about their work.)

The test execution takes two hours, but is the tip of the iceberg. The overhead for all the support activities often is much larger than the visible part. These support activities include everything from setting up test equipment to writing problem reports. (It is better not to label these activities as "overhead" because they are – or should be – a natural, integral part of the test process.) One secret of realistic estimating is to identify all of the contributions needed to complete an activity. This point seems obvious, but it is often overlooked.

Exercise 3, Part 1: Calculating Your Direct Test Overhead (Allow 5 minutes for this exercise.)

Facing reality, and analyzing where our time is going currently, is an important step in realistic estimating. The purpose of this exercise is to calculate how long you and by extension your team are really spending on testing activities, compared to the obviously visible part of testing.

Consider a typical system test project in which you have been involved recently. This project should have contained at least one task where you were assigned the job of manually testing a feature or a set of related features, and you tested them primarily by yourself. Instead, if a team of three people jointly tested the features, is harder to analyze the other people's time than your own. In your time accounting, do not include automated testing tasks or non-feature testing tasks (e.g., load testing or configuration testing).

CALCULATING OVERHEADS
(Continued)

For every two hours that you spent at the keyboard, manually running the feature test cases and doing nothing else, approximately how long did you spend on the other related tasks? These tasks include preparation such as setting up the test environment and follow-up such as writing problem reports,

Activity	Typical amounts of time
Preparation	_____
Hands-on Testing	2 hours
Follow-up	_____

What if You don't have Testing Experience?

So far in this exercise, I have assumed you have considerable hands-on testing experience, in other words, you are personally quite familiar with the work content on a test project. A solid base of real-world experience is a powerful tool for an estimator.

But not everybody in our audience will necessarily have that background. You may be a new test professional but still be expected to produce estimates, or an experienced tester who has moved into a new and different work environment, or an experienced project manager or business professional who's never personally spent much time testing.

If you do not have direct personal testing experience, I would like you to do this exercise from a different perspective. Instead of remembering your own experience and recording how time is being spent on your testing projects, write down the amount of overhead which you think *should be allowed* in each category. Your common sense and general background knowledge are enough to give you a sense of what's reasonable. Later, compare your answers with the testers in your organization. Your points of agreement and disagreement will be illuminating.

What if every Project is Different?

Some people live in rapidly evolving and even chaotic worlds. There is little continuity of how their projects are done – the only constant is that software development, testing and maintenance methodologies are always changing. It is difficult to know how and where the patterns of past behaviors will apply to the future.

If you are in this situation, the exercise is still doable but harder. For each overhead item listed below, provide a range of three numbers: your most likely, most pessimistic (highest) and most optimistic (lowest) guesses. For those overhead factors which you consciously or unconsciously feel are fairly stable, the spread among the three numbers will be low.

When the spread is wide, indicating considerable uncertainty, you can take the average of the three numbers and mark it up by a generous (25% to 50%) factor of safety, to compensate for the risk introduced by the uncertainty. This is not fudging the numbers, this is prudent risk management. (If the boss buys this justification, tell him about the need to restore the testers' bug hunting powers with a rest trip to Hawaii.) Seriously, though, adding a generous factor of safety to an item which has high uncertainty is a good practice.

Exercise 3, Part 2: Calculating Your Direct Test Overhead (Allow 20 to 30 minutes for this exercise.)

Now, please re-estimate using the so-called micro-estimating or bottom-up technique. This approaches utilizes a detailed list of typical work activities rellated to the test execution (see below).

For every two hours that you spent at the keyboard, manually running the feature test cases and doing nothing else, approximately how long did you spend on each of the other tasks listed here?

CALCULATING OVERHEADS
(Continued)

Customization Pre-Requisite

The tasks listed here will not exactly fit how you do things. You may need to modify the list of tasks to match your approach before you collect and enter the data.

Testing Support Task	**Time Consumed** for each two hours of hands-on testing (Approximate hours)
A. Test Preparation	
Participate in a meeting with the team leader to coordinate your activities, or with the test team to coordinate everyone's activities, and to receive your next work assignment: to test the set of features	_____
Review the functional specs. or an equivalent description of the relevant feature(s) – this includes verbally quizzing the developers or the users about the feature	_____
Resolve questions about the specs. and the expected behavior of the feature and the system	_____
Review the pertinent parts of the test plan	_____
Learn about any changes to the system functionality and test plan since you last accessed them, and their impact on how to test the feature	_____
Review problem reports which have been fixed and are ready for testing	_____
Explore the feature(s) on-line, or a prototype of the feature	_____
Speak with the software engineers to resolve questions about the feature implementation	_____

CALCULATING OVERHEADS
(Continued)

Develop the test requirements, if these do not already exist _____
(Or: review and clarify the test requirements, if they already exist)

Search for existing test cases which can be adapted and reused _____

Adapt the existing test cases as needed for this use _____

Develop additional new test cases (where no existing ones can be adapted) _____

Develop the test data needed by the test cases, if these do not already exist _____
(Or: review and if necessary update the test data files, if they already exist)

Gather the equipment and software needed for the testing _____

Set-up and check-out the test facilities _____

Integrate or build the version of the system to be tested _____

Establish version control or configuration management control over the new system version _____

Load the system version we are testing into the test environment _____

Load the data files and databases needed for this test _____

Establish version control or configuration management control over the loaded test environment _____

Review and become familiar with the testware, test cases and procedures _____

Confirm everything is configured correctly, run a smoke test and verify that you can proceed with the testing _____

Total for Test Preparation: _____ hours

CALCULATING OVERHEADS
(Continued)

B. Test Execution and Follow-up

Execute the test cases manually	2.0 hours
Evaluate the test results	_____
Re-run test cases to replicate and confirm the system's behavior	_____
Trouble-shoot, debug and fix the testware	_____
Confer with the software engineers about the symptoms found (i.e., the test case failures) and possible bugs	_____
Document the test results log	_____
Write the problem report(s)	_____
Enter data into the bug tracking system	_____
Monitor the problem resolution and the availability of the fix to be re-tested	_____
Re-test to confirm the fix works	_____
Regression test to find unintended sides effects of the fix	_____
Document and archive the test results	_____
Restore the test environment to a known pristine condition ready for the next test suite	_____
Break down the test environment and return hardware used (cables, computers, etc.)	_____
Confer with the boss on the project status and direction	_____

CALCULATING OVERHEADS
(Continued)

Update the test plan and test case libraries, if necessary _____

Take a well deserved break _____

Total for Test Execution and Follow-up: _____ hours

Unplanned Interruptions during the Work _____

Other miscellaneous related activities (allow at least 10%) _____

Total including All Activities : _____ hours

CALCULATING OVERHEADS
(Continued)

One Test Team's Answer to this Exercise

When they took the time to analyze where their time being spent, a test team with a large multinational bank found that they spent approximately two days of effort in all, for each visible two hours of test execution. The team's time allocations, for each two hours of manual test execution, were as follows:

Testing Task	Time Consumed (Hours)
A. Test Preparation	
Participate in a test team meeting to coordinate the team's activities and to receive your next work assignment	0.75
Review the functional specs. or equivalent description of the feature or system	0.5
Resolve questions about the specs. and the expected behavior	0.5
Review the pertinent parts of the test plan and test procedures	0.50
Learn about the changes to the system functionality and test plan since you last accessed them	0.25
Review problem reports which have been fixed and are ready for testing	0
Explore the feature(s) on-line, or a prototype of the feature	0
Speak with the software engineers to resolve questions about the feature implementation	0
Develop the test requirements, if these do not already exist	0.5
(Or: review and clarify the test requirements, if they already exist)	
Search for existing test cases which can be adapted and reused	0.5
Adapt the test cases for the new use	1.0
Develop, document and review additional new test cases (where no existing ones can be adapted to meet the need)	1.50
Develop the test data needed by the test cases, if these do not already exist	0.5
(Or: review and if necessary update the test data, if they already exist)	
Gather the equipment and software needed for the testing	0.00
Set-up and check-out the test facilities	0.25
Integrate or build the version of the system we are testing	0.00

CALCULATING OVERHEADS
(Continued)

Establish version control or configuration management control over the new system version	0.00	
Load the system version we are testing into the test environment	0.25	
Load the data files and databases needed for this test	0.25	
Establish version control or configuration management control over the loaded test environment	0.25	
Review and become familiar with the testware, test cases and procedures		
Confirm everything is configured correctly, run a smoke test and verify that you can proceed with the testing	0.17	
Total for Test Preparation		7.67 hours

CALCULATING OVERHEADS
(Continued)

B. Test Execution and Follow-up

Execute the test cases manually	2.00	
Evaluate the test results	1.00	
Re-run test cases to replicate and confirm the system's behavior	0.33	
Trouble-shoot, debug and fix the testware	0.20	
Confer with the software engineers about the symptoms found (i.e., the test case failures) and possible bugs	0.33	
Document the test results log	0.07	
Write the problem report(s)	0.50	
Enter data into the bug tracking system	0.10	
Monitor the problem resolution and the availability of the fix we are testing	0.10	
Re-test to confirm the fix works	0.33	
Regression test to find unintended sides effects of the fix	0.20	
Document and archive the test results	0.20	
Restore the test environment to a known pristine condition ready for the next test suite	0.00	
Break down the test environment and return hardware used (cables, computers, etc.)	0.00	
Confer with the boss on the project status and direction	0.33	
Update the test plan and test case libraries, if necessary	0.17	
Take a well deserved break	1.50	
Total for Test Execution and Follow-up		7.36 hours
Unplanned Interruptions (*)	2.0	
Total Time:		***17.03 hours***

(*) Meetings not directly related to the task at hand, phone calls, consultations on other projects, etc.

Limitations of this Analysis

Unless we conduct a prohibitively expensive effort to collect and validate the data, we should not assume that numbers like the above ones are very accurate.

These numbers are rough averages because each task varies, sometimes taking no time and sometimes much more than the numbers shown.

In addition, testers have their own styles and do things in different ways, regardless of what the written test procedure is. People don't even use the same names for the same things, so achieving consistency in the timing data is problematic.

There's also the question of the care that was taken in the measurements. Imagine that the duration of each task was under-reported by 5 minutes (0.08 hour). It doesn't seem significant. However, with about 35 tasks on this list, the aggregate under-reporting is approximately 2.8 hours or 15% of the total estimate.

Conclusions from this Analysis

Since the bank managers have the impression that the testing took two hours (the visible part), that's the amount of time that they are inclined to estimate for future similar test tasks. (This test team does not have much input to the decision makers on the project budget and schedule decisions.)

How effective do you think a test team is which spends 17.03 hours to get two hours' worth of hands-on testing? This does not appear agile: 88.25% of their time is not being spent directly testing. What do you think the ideal ratio should be? Should 100% of their time be spent testing (zero overhead)? That means there is no time to plan, just mindless banging on the keyboard, and no time to evaluate and document the test results.

On the surface, test process improvement seems like a good idea for the test team at the bank, to help them become more efficient.

The impression of inefficiency is misleading, though. The test team could be highly efficient already or quite inefficient; this data does not tell us. Our conclusion of inefficiency is based on the way I presented the data: "two days is needed to complete two hours' work".

Let's say I introduced the same data with the statement: "Look how much the testers can do in only two days". I could bias the suggestible to conclude that the team is industrious.

Until the improvement is realized, though, the bank managers should use the reality-based ratio of 17.03-to-2.0 (8-to-1) if they want to accurately estimate.

Some project planners disagree – they think that allowing 88.25% of the testers' time to be spent on "overhead" is condoning inefficiency, and that a goal of higher productivity should be established by setting aggressive estimates.

CALCULATING OVERHEADS
(Continued)

Perhaps targeting the overhead ratio at one-to-one will motivate the testers to work smarter and harder, but probably also lead to project schedule and budget overruns or lead to poorer-quality delivered systems.

Testing Practices

What ratio is typical of the total test effort to the hands-on test execution? In a survey which I conducted of 111 testers, their overhead ratios, in a situation where the hands-on testing took two hours, were as follows:

Ratio of Total Test Effort to Hands-on Test Execution	**Percentage of Testers**
2-to-1, or less	0%
2-to-1 to 4-to-1	6%
4-to-1 to 6-to-1	21%
6-to-1 to 8-to-1	37%
8-to-1 to 12-to-1	16%
12-to-1 to 16-to-1	10%
16-to-1 to 20-to-1	5%
20-to-1 or more	5%

Exercise 3, Part 3: Calculating Your Direct Test Overhead (Allow 20 to 30 minutes for this exercise.)

Repeat the process of calculating your test overhead for each of these circumstances which happen on your projects:

(1) A test of a new system which is done manually.

(2) A test of a new system which is automated, not manual.

(3) A test of a change to an existing system which is done manually.

(4) A test of a change to an existing system which is automated, with an existing regression test facility and automated test case libraries.

(5) A test of a system's performance characteristics and ability to handle load, which does not include functional testing.

CALCULATING OVERHEADS
(Continued)

Comparing Ratios with the Managers

Perhaps the most important exercise is to compare people's perceptions with reality. Do your managers know what your overhead ratios are? If you are unsure, ask them to give you their sense of the ratios. If they are significantly different from your own numbers from this exercise, the differences may be worth a follow-up discussion.

Identifying Inefficiencies

We can also use the work sheets above to ask: "Where did all the time go?" Often, surprisingly large amounts of time are frittered away of "little" inefficiencies. Analyzing how the test team's time was actually spend is an important step in streamlining the test process.

When I helped a test team in a financial services organization analyze how their time was being spent, we were amazed to find that thousands of hours per year were being spent trying to find existing test cases, figure out what they did, and assess whether they could be used on new projects. Once the inefficiency was apparent, the solution was obvious – to improve the test data management and build an organized, indexed repository of test cases.

The need was not apparent without the time analysis, though. Some testers assumed that the existing inefficient methods were the ways things had to be. Others were so busy getting the work done that they had not noticed where their time was being spent.

Common Areas of Omission – Part 2

Uh oh – what do you mean, Part 2?? I thought we were done with this topic of omissions. The answer is that we have started but not yet finished identifying the overhead associated with testers' work.

I find that is helpful to analyze test overhead (and thus omissions) in two phases:

(1) The directly related tasks necessary to support the hands-on test execution (which we have addressed above).

(2) Everything else.

The term "everything else" covers a lot of territory. (This line about everything else covering territory is an old joke in philosophy, used by professors to wake up their undergraduate classes. If you believe that what we can know is unbounded, "everything else" is a convenient way to ensure we haven't left something out. Of course, nobody agrees on what this term means.) Since it's hard to define everything else, we'll use a checklist of common omissions to help identify which ones apply to us.

Bottom-up estimates are often too low because of inadvertent omissions, as we saw above. By default, any testing activity which is omitted from the detailed task list and thus from the work plan receives an allowance of zero hours and zero dollars in the project budget. So it is worth our time to be reasonably complete in identifying the omissions.

I've listed the types of work activities below which are often omitted or understated in test project plans, and which are beyond those directly related to the hands-on test execution. There is some overlap between these listed items and the tasks listed earlier. Since it is hard to delineate between the direct and the second-order overheads, there is a small amount of deliberate redundancy. Don't doubled-count the same overhead item because it appears on both lists.

The percentages in parentheses are representative of the amounts of overhead that we should anticipate, if we follow typical industry practices. (Don't assume that these percentages are necessarily right for you.) These amounts, or better yet your best guesses at these numbers of your environment, will be added to the "raw" estimate of testing time (e.g., the 17.03 hours in the bank), as a factor of safety.

CALCULATING OVERHEADS
(Continued)

"Second-Order" Test Overheads

- Unanticipated time delays in the availability of test resources, such as testers with specialized skills, test equipment, test databases, tools, etc. (5% to 15%)

- Unanticipated time delays while waiting for bugs to be fixed, and before testing can continue. (5% to 20%)

- Unanticipated delays and added work because of enhancements or other changes to the system functionality during testing. (5% to 20%)

- Glitches and gremlins in the test environment. (5% to 15%)

- For geographically dispersed testing projects, the overhead for travel, meetings and coordination. (0% to 5%)

- Overhead for test-related activities which supposedly are outside a narrow job description for the testers, but which the testers get stuck with, like version control, providing customer service, and psychological counseling with distraught programmers. (5% to 15%)

- Overhead for test-unrelated activities which the testers can get stuck with, like technical writing and user training. (5% to 30%)

- Turnover and fragmentation (through time-juggling across multiple projects and work activities) of the testing team members. This can include people being pulled off the project temporarily for other activities. (5% to 50%, if the turnover and time-juggling are likely to be high.)

- Over-confidence that newly introduced technologies, tools and methodologies will work, and under-appreciation of the learning curve and the debugging and fixing needed to get the testing process working smoothly. (10% to 15%, and much more in the worst cases.)

- Overhead of sharing resources, contention for those resources, and interference between activities using the shared resources. (5% to 15%)

CALCULATING OVERHEADS
(Continued)

- Overhead of bug advocacy, such as testers reproducing bugs for developers and selling them on the importance and urgency of a bug fix. (5% to 15%)

- Hidden overhead of poorly written bug reports (time wasted looking in the wrong places, etc.). (5% to 25%)

- Unanticipated customer or user support issues, such as responding to a high priority user problem in the middle of testing a new version of a system. (5% to 25%)

Note that, taken together, these omitted activities can easily add over 100% to a "bare bones" schedule and budget for the testing project.

CALCULATING OVERHEADS
(Continued)

Exercise 3, Part 4: Calculating Your Indirect Test Overhead (Allow 15 to 20 minutes for this exercise.)

Let's assume you have a test activity in your organization which you expect to take two days, including test preparation, execution, results evaluation and follow-up, and all directly related tasks as discussed earlier.

Gauge how much overhead realistically should be added to the two days for each of the second-order items listed above, such as lengthy delays in bugs being fixed and the fixed versions of the features becoming available for re-testing.

What percentages of your time is being spent in (a) direct hands-on testing, (b) directly related tasks such as test preparation, and (c) second-order overhead?

Be ready to present and discuss your findings with your test team members and your managers, assuming that you have not fainted in the meantime.

For each of these factors, approximately what percentage of schedule delay is introduced into your projects typically? We are talking about elapsed time here. If you can creatively fill in the idle time with other tasks which contribute to making the overall project deadline, you should adjust the amount of delay.

	Overhead Factor	**Allowance for this Overhead** (Expressed as a percentage of the total project schedule)
o	Delays in the availability of test resources	_____
o	Waiting for bugs to be fixed	_____
o	Added work because of enhancements or other changes	_____
o	Glitches in the test environment	_____
o	Travel, meetings and coordination	_____

CALCULATING OVERHEADS
(Continued)

- Overhead for any support activities which were not previously counted _____
- Turnover and fragmentation _____
- The learning curve for new technologies, tools and methodologies _____

Other major causes of delays:

_____ _____

_____ _____

_____ _____

Total overhead: _____

ALLOWING FOR OVERHEADS

The Insidious Nature of Overheads

By now, it will be obvious that overheads which we overlook can wreak havoc with estimates, but I'd like to share another example to reinforce the point.

Imagine that you need to attend a one-hour meeting to review a test plan. It takes you 10 minutes to get to the meeting room. The meeting starts 10 minutes late. Half way through, the meeting is interrupted for 5 minutes as one of the group has to take an outside call. The test plan is not all complete and ready for review, so part of the content is postponed for a future follow-up session. And on your way back to your own office, someone distracts you for a few minutes with another issue.

At the end of the day, you groan: "Where does all the time go?" The meeting, for which you had mentally allocated one hour, will eventually end up taking three hours in all.

The moral of this story is that it is perilous to estimate without factoring in all the overheads.

This section discusses ways we can allow for overheads, especially the unanticipated ones like interruptions. It addresses the concept of loaded vs. unloaded hours, allowances for overheads such as project management and test quality control, and the contingency for changes. This section does not provide an exhaustive coverage of all the sources of overhead.

Loaded Vs. Unloaded Hours

We usually are mistaken when we assume that a test professional will be able to complete 40 hours worth of designated tasks from the test project work plan in one week. To complete the work, that person may need to spend 50, 60 or more hours on the job that week.

In an estimate, an unloaded hour is an hour of actual work on a task. By contrast, a loaded hour includes the overhead that must accompany the actual work, such as unplanned interruptions for other activities which are not directly related to the test task itself.

The loaded hour does not include the overhead for other tasks related to the primary task; these other tasks must be identified, listed and separately, or accounted for in a different category of overhead. The loaded hour does include an adjustment for the unrelated overheads which make a one-hour task take one and a half hours, such as impromptu meetings about other work activities.

ALLOWING FOR OVERHEADS
(Continued)

In an internal study, IBM found that a loaded hour only includes about 0.65 hour of unloaded work time. In other words, we only get about 5.5 hours of work in a standard 8-hour day. The remainder of the day is absorbed by unplanned activities, such as telephone calls, interruptions, and attending meetings which are not directly associated with the task at hand.

If we take the estimate of the unloaded hours for a project and divide by 8 to calculate the number of days required, instead of dividing by 5.5, we will underestimate the elapsed time needed (measured in days) by about 35%.

In addition, the calculation of the days needed for testing must include a realistic allowance for weekends, holidays, and vacations. (Saying "No vacations are allowed until the test is over" may not motivate the staff very well.)

Project Management Overhead

In addition to loading the hours of effort and allowing a factor of safety, additional hours need to be added for the testing project management. Very small (one to three people) testing projects tend to be self-managed.

For testing projects which are staffed with more than three people, an additional 15% should be added to the sum of the test work activities in order to allow time for a part-time or full-time test project leader. See later in this section for more details on estimating the project management overhead.

Activities such as project management are best estimated as a separate line item in the overall project budget, even if it is calculated as a fixed percentage of the testing work load. The project management should not be added into each individual task as parting of the loading of that task.

Test Quality Control Overhead

Remember to plan in overhead for reviews of the test plan, test cases and test results, and possible revisions. This overhead usually adds 10% to 20% to the work activity, depending on the level of risk and complexity. Peer reviews and allowance for revisions should be identified as distinct tasks in the test project work plan, not bundled into other tasks, so that they can be tracked.

ALLOWING FOR OVERHEADS
(Continued)

Contingency for Changes

A project's scope usually increases, by 10% to 15% for small projects, and 30% for large projects, from the time the project is initiated to when it has been completed. Even if the scope does not increase, there almost always are changes to the functionality while it is being tested, changes to the technical support environment, and so on.

The purpose of a change budget, which can be measured in hours or dollars, is to leave some flexibility to accommodate change without having to go back and re-estimate, and if necessary increase the budget or slip the schedule, for every little change which might occur.

Factors of Safety

A "fudge factor", or factor of safety, needs to be included in the test estimates in order to provide a contingency for uncertainties. It is dangerous to assume that all testing tasks were identified up front and that the estimates are sized accurately, and that the testing will go smoothly. There may be unanticipated delays in starting tasks, and tasks could take longer than planned.

A reasonable factor of safety for a moderate-size, moderate risk testing project is 15& to 25%.

For a high-risk, large-size project, the factor of safety should be much higher: perhaps 33% to 100%, assuming that the managers and system clients will allow this generous factor of safety.

The factor of safety either can be a stand-alone item in the estimate, or can incorporate some of the other factors mentioned earlier (e.g., the contingency for change).

The factor of safety should not be hidden padding, distributed in some pattern across the various tasks. While the factor of safety can profitably be either included in each atomic task or added as a lump sum to the overall project budget and schedule, it should be acknowledged and publicized in the documentation of the estimate. If the padding is hidden and scattered at random, nobody -- including the test planners -- will remember where it is or what it is intended to be used for.

ESTIMATING INDIVIDUAL TEST TASKS

How do We Estimate each Individual Task?

The bottom-up or micro-estimating approach is only as good as our ability to estimate each individual task. A person who is over-confident and naive might underestimate tasks, by amounts which vary from 10% to 50%. A person who is intimidated by what they do not know and by the potential difficulty of the work might over-estimate, by 25% to 100%.

Task estimating is a time-consuming and laborious chore but is absolutely necessary, because there is no substitute for careful thinking. We need to ask these questions for each and every task, or at least for each group of similar tasks, on the test work plan:

- Who on our team is the best person to estimate this type of work? Who is the closest to this type of work and the most knowledgeable? (The most knowledgeable and the person assigned to do the task may not be the same person.)

- How well do these people understand the task? How much applicable prior experience do they have with this type of work? How similar is that prior work to the task to be done?

- Is this task sufficiently important, large or poorly understood that we want two people to separately estimate it, then compare and reach consensus?

- What are steps the tester will follow to accomplish this task? How orderly, routine and well understood is the process by which to accomplish the task?

- Who, or what type of person, is likely to actually be assigned to perform the task?

- How experienced, competent and well-trained is the assigned person likely to be for this task? (A common mistake in bottom-up estimating is that the estimator, who is a highly experienced person, thinks about what it would take for him or her to do the work. It does not occur to the estimator that what he can do in one day, could take someone else three days. Another common mistake is to assume that someone who is very good at a different type of work will be equally productive for this task)

- What could go wrong with this task? During the execution of this task, what is risky? What type of "fudge factor" should be allowed in the task estimate for anticipated and unanticipated gremlins?

ESTIMATING INDIVIDUAL TEST TASKS
(Continued)

- What are the expectations about the situation when the task begins? What preparation and pre-conditions are expected to be met before starting this task? What work is likely *not* to be done, so it should be included within the scope of this task or otherwise taken care of?

- What are expectations about the interim deliverables from this task? Exactly what is included in the scope, or not? For example, at the end of a task to build a test case, has that test case been documented, reviewed, debugged, cross-referenced to the features it testes, and entered into the test case library under version control?

- What overhead and support is likely to be needed by the person doing the work? For example, is the tester likely to need access to a test automation specialist to help troubleshoot the test environment?

- What specific prior experience do we have to compare with this new task? How big is our sample of prior similar tasks to draw upon? (Hopefully, the sample size is more than one.)

- How precise is our information about these prior tasks? Are we relying on anecdotal (and perhaps distant) memories which clutter the brain cells of various team members, or data that is more solid?

- How similar in scope, objectives and the nature of the work on those prior tasks? Do we have to tweak or adjust the prior experience so that it is more similar and thus more comparable?

- What types of resources, effort and duration did the prior similar tasks take? Since here typically is a range, what were the times for shortest, the typical and the longest of the prior tasks? For example, one apparently similar task could have taken three days, while another apparently similar one took three weeks.

- What factors account for the differences between apparently similar tasks, causing their durations to vary from three days to three weeks? How do we calibrate or adjust for these factors in estimating this particular task?

Hopefully, most of these questions have already been answered earlier, during the process of developing and reviewing the test work plan. If so, we do not have o re-invent the wheel, though it is always a good idea to give these questions a quick second review.

THE RELATIONSHIP OF ESTIMATING AND WORK PLANNING

I have been asked many times: "How do I estimate the size, duration and resources for my testing project?" I answer this question with another question: "Do you have a detailed work plan for the testing activities?" The answer to this question is often a "no", together with some puzzlement and irritation about why we are talking about planning, when they are interested in estimating, not in project work plans. Isn't there a simple, quick formula that will produce an estimate without having to bother with work plans? Not if we want a trustworthy estimate!

Without good project work plans, there can be no accurate estimates. An effective work plan is modeled on the work breakdown structure for the project. The plan identifies the tasks of the testers, the sequence and flow of these tasks, their dependencies on each other and other events, the schedule of when each task is expected to begin and end, who is responsible for each task, and the resources allocated to each task.

Planning their work usually takes a lot of time, thought and effort, so some testers avoid it if they can. Or they develop a detailed plan, but their plan has a futurity of only a week or so, because it is too difficult to see much further into the future.

Planning requires thinking, and sometimes action-oriented people prefer to "do, not think". These people say: "Let me get on with the testing, instead of being a bureaucrat and paper pusher." They rationalize that they can manage their work "intuitively" and adaptively as the situation evolves. And since the work plan needs to be updated as conditions change, which imposes a certain overhead, they say that planning is just not possible on their project

Some people believe that planning somehow curtails their "test creativity". It is true thatcreativity is important in testing. But it is not very creative to have to stay up all night, running that unplanned, last minute "creative" test.

Good planning, of course, is justified for its own sake, even if it is not needed as a basis for estimating. Work plans give the testers a way to track and assess the status of test projects, and give early warning if mid-course corrections are needed during the project. Testers who have planned their work, and not just "on the fly", tend to work in a more organized fashion, have fewer last- minute surprises, and are less likely to make silly mistakes.

The good news is there are many excellent books and classes on project planning.

PROJECT CHECKLISTS AND ESTIMATING WORKSHEETS

The Use of Templates and Checklists

Because the bottom-up technique tends to underestimate through the omission of tasks, it is a good idea to develop templates in order to minimize these oversights. A template is a detailed outline or checklist of the main work steps needed to test a system, in other words, the checklist provides a generic work breakdown structure and task list to use as a starting point. With this detailed set of reminders of what tasks may need to included in a testing project, omissions are less likely.

A template provides a checklist of all the main activities in a system development or maintenance project which the testers will be involved in or dependent on, not just strictly the testing activities. (For brevity, the template later in this section addresses performance testing only; it does not include other activities which are outside the scope of the performance test project.) Since testing is context-specific and no two testing projects are exactly the same, it is a good idea to develop a generic checklist for each major type of project the testing team is likely to undertake. Then a generic checklist will need to be reviewed further and tailored to fit a particular testing situation.

The Main Steps in Template-Driven Estimation

(1) Review the background of a situation to determine which of the off-the-shelf checklists in the testers' library best fits the situation. If none fit, a new template must be created.

(2) Modify the generic template for this particular project.

(3) Use the modified template to develop a "raw" bottom-up estimate.

(4) Use the estimate adjustment factors (described later) to calibrate this raw estimate and deliver the final adjusted estimate.

Estimating Worksheets

Fill-in-the-numbers worksheets offer convenience and relative consistency, even if the simplification to make a worksheet usable across different project situations and by people with different backgrounds and skills means their accuracy is limited. This section provides an example of a fill-in-the-numbers worksheet for estimating performance and load testing projects. The idea is to use this worksheet as a guideline to "guesstimate" how much time is needed for each task in a particular performance test project.

PROJECT CHECKLISTS AND ESTIMATING WORKSHEETS
(Continued)

Performance Testing Work Task	Estimated Days of Work (*) Low	Mod	High	Our Project
Initial performance impact assessment				
Data gathering	1	3	5	_____
Development of initial conclusions	1	2	3	_____
Development of performance test plan				
Draft of the plan	4	6	8	_____
Review and approval of the plan	1	2	3	_____
Preparation for measurement (**)				
Set-up of test environment	3	4	5	_____
Installation of load generation and measurement tools	1	2	3	_____
Trial run and check-out of the test environment	1	2	4	_____
Development of new automated test cases for this project	7	10	15	_____
Performance measurement				
Test execution and data gathering	10	15	25	_____
Re-runs for additional data gathering, validation and exploration	5	7	10	_____
Data analysis and interpretation				
Data analysis	5	7	11	_____
Coordination with performance tuning specialists	1	2.5	4	_____
Peer reviews and validation of findings and conclusions	2	3.5	5	_____
Development and presentation of conclusions and recommendations	2	3	5	_____
Post-mortem to review the lessons learned	1	1.5	2	_____
Total estimated number of days' effort	45	70.5	108	_____

(*) These ranges are based on prior similar performance testing projects.

(**) Assuming all the test equipment and tools needed are already available; no time has been allowed for acquisition of equipment or learning new tools.

Note that this checklist includes zero time for early preventive tasks, such as the performance testers' participation in system design reviews. To prevent under-estimating by omissions, it is important to examine the scope of the testing project and compare it to the checklist.

The worksheet first needs to be calibrated to your organization, and then can serve as the generic worksheet for performance testing in the organization. Before it can be used in any particular situation, the generic version should be modified again to fit that situation by inserting, deleting or modifying the work tasks on the list.

The blanks on the worksheet are filled in either through informed guesses, prior experience or reviewing the list of estimate adjustment factors which is provided later.

On a particular project named Nimble , for example, let's assume that the initial impact assessment has been done. In this case, the first two items on the checklist (data gathering and development of initial conclusions), would both receive an estimate of zero days. We could insert a substitute task, to review the initial impact assessment results, and allocate this task an estimate of 0.5 days based on prior experience.

We also have a sense that the Nimble project is small to moderate in size, compared to our prior performance testing projects.

On this project, we are reasonably confident that existing automated test case libraries can be re-used but that some re-work will be needed. All the test equipment and tools needed for this project are already available. No extra time is needed for acquisition of equipment or learning new tools. However, for this project the testers need to write a new interface between the load generation tool and the statistical package which will be used to analyze the performance measurements.

The biggest uncertainty on the project is how much support and coordination will be needed by the performer tuners who will be using the performance measurements, so a wider than normal range of times will be given to this activity. The project manager also does not believe in post-mortems (a mistake on his part), and is unwilling to allocate time for them. However, he's willing to allow a 10% contingency to be added to the estimate.

PROJECT CHECKLISTS AND ESTIMATING WORKSHEETS
(Continued)

We would fill in the blanks on the template and produce a first estimate as follows:

Project Name: Nimble Software

	Performance Testing Work Task	Estimated Days of Work
1	Initial performance impact assessment (*)	
	Review of the results	0.5
	Development of performance test plan	
	Draft of the plan	3 - 4
	Review and approval of the plan	0.5
	Preparation for measurement	
	Set-up of test environment	3 - 4
	Installation of load generation and measurement tools	1 - 2
	Writing of new interface to statistical analysis package (**)	3 - 4
	Trial run and check-out of the test environment	1 - 1.5
	Adaptation of existing automated test cases for this project	5 - 8
	Performance measurement	
	Test execution and data gathering	12 - 15
	Re-runs for additional data gathering, validation and exploration	3 - 5
	Data analysis and interpretation	
	Data analysis	5 - 7
	Coordination with performance tuning specialists	2 - 8
	Peer reviews and validation of findings and conclusions	3
	Development and presentation of conclusions and recommendations	3
	Total estimated number of days' effort for this project (not including the contingency)	44 - 65 days

(*) Already done; not included in this work effort.
(**) One-time activity for this project only.

We then would add the contingency:

Project Name: Nimble

	Estimated Days of Work
Total estimated number of days' effort for this project	44 - 65 days
Contingency (10% of total)	5 - 7
Total estimate including the contingency	49 - 72 days