System Testing Without a Specification

By Sanford M. Sorkin

Testing without a specification is usually a major challenge to Quality Assurance organizations. QA is unaware that a project is nearing completion and no time is allocated for the unexpected work. The immediate reactions upon being presented with the problem, run the gamut from:

Not again.

How can I be expected to test without specifications?

What's the delivery date?

Let's get started.

Regardless of the response, it is an application that must be tested. Later on we can deal with the issues of defect prevention, the potential value of early QA involvement, and how this happened, but for now we have a requirement to test, which at this point may be the only requirement we have.

With no preplanning on the part of QA, ordinary testing procedures must be preceded by a few documentation tasks that normally would be prerequisites to passing work over to the QA group. What follows is an approach that starts with the receipt of the work to be tested, moves through documentation steps, and then on to various types of testing.

Step 1: Create a functional specification
Step 2: Size the work effort (writing the specification)
Step 3: Provide management with an estimate of time and test coverage
Step 4: Review requirements and start managing the project
Step 5: Automated testing decision
Step 6: Additional preparation

Variations to approach and procedure will occur with every testing project, but there is very little that is unique to a particular job. Therefore, look for the commonality to determine if prior testing materials and plans can be resurrected to facilitate the unexpected work.

Step 1: Create a functional specification

Resist the temptation to just start testing. This approach is fraught with potential disasters. Organization will be the key to successfully completing this project and the best first step is to size the project and determine just what can be expect from this testing exercise. This will, of course, necessitate that *you* create a specification for the application. Clearly this is work that should have been completed much earlier by someone else, and will require an extraordinary effort to do it now, but it is the first step.

If you are still considering the consequences of starting without writing a specification, consider the following:

You do not have a clear statement of objectives.

You are uncertain how long the testing project will last (though you may have been given your delivery date as well.)

It is not clear who should be assigned to this project.

It will be extremely difficult to provide any sort of accurate risk assessment beyond the known risks of starting to test with incomplete information,

Automating some, or all, of the tests is probably out of the question.

The long term problems of maintenance and enhancement will be compounded by implementing an undocumented system. Costs associated with all future activities will probably increase exponentially.

We may not know when testing is complete.

If the application to be tested is a simple standalone system the list of concerns is lengthy, but if we are preparing to test in a client server environment it gets much more complicated. Deliver to management a brief explanation of the risks associated with inadequately testing this type of application. It will also be necessary to alert the network group and involve them in testing the communication aspects of the system.

The additional complexity of testing C/S applications suggests that the risk and contingency section of the test plan will be lengthy. (Make certain this information is very close to the top of the plan so that it can not be overlooked in the review.)

"No time to plan, I'm late, I'm late."

If it is absolutely necessary that you start immediately, at least attempt to answer the following questions:

What are the business objectives of this project?

Who are the customers and what are their expectations?

What are the customer's experience levels?

Will customer support be readily available to answer questions?

Is the application to be tested new development or maintenance?

Who worked on the project and was the corporate development methodology followed?

When must the system go into production?

What are the risks inherent in system failure? (Financial, confidence, business opportunities, etc.)

What is the intended primary functionality of the system?

The list of potential questions is endless so keep inquiry to a minimum. Once the primary functionality is understood, ancillary functionality will start to appear and you should document and proceed to the next steps.

As indicated, if this is a C/S application, special care must be taken to get all of the other potential testing participants in this project organized--especially the network team. They will be responsible for the communications aspect of the system.

Step 2: Size the work effort (writing the specification)

The specification can be in an outline format simply listing each of the system requirements. If there is a standard format your organization uses for creating specifications, then it should be followed. Requirement development approaches:

Talk to the person who delivered the application to be tested, even though it may be difficult to pin this person down.

Run the program and possibly review the code.

Interview the customers, (be careful, it may not be good politics to tell the customer you have no idea what they were trying to accomplish.)

Review programmer notes and correspondence. Find as much documentation as possible.

Take your cues from the content, peruse the help screens, and look for descriptive messages. If this is a maintenance project, try to identify what was changed or added to the system to localize much of your initial testing. It is still possible that you will not have all of the requirements, but this is definitely a viable place to start.

As you interview customers or talk to programmers, they will have a tendency to relate what the system is intended to do by enumerating all of the inputs followed by the perceived processing steps necessary to produce the desired outputs. A much more effective interviewing technique will be to ask them to start by listing all of the expected outputs of the system. Identifying the outputs provides a sense of the system objectives and begins the process of identifying the system test plan and acceptance test plan contents. These outputs may include:

Reports

Screens / displays

Messages / interfaces

Files

Sounds (audible messages)

Next discuss the inputs to the system followed by the processing steps required to produce the outputs. Remember that while ideally the system requirements are provided, without specification we are reverse engineering to determine what the requirements were. If and when the test plan is reviewed, each of the requirements will be confirmed.

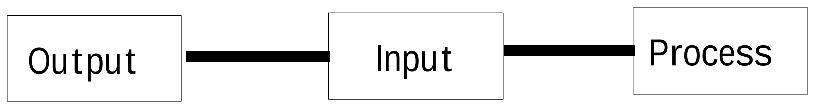

Figure 1: Interview approach

Once a picture of the total system is created it is time to develop estimates. Management must be kept informed to assure that the best decisions can be made.

The product of the sizing effort is a document that lists each of the system requirements with enough detail that other team members can review the work and assist with estimating the project. It is unlikely that the requirements definitions will be complete, but nevertheless they are now the foundation for creating functional specifications and design documents to facilitate any future enhancement efforts.

The sample document that follows contains an assessment of the risks and contingencies associated with testing the application. While it may be incomplete in certain respects, it will provide an opportunity for others to fill in missing elements during the review.

The sections of this document may include:

Functional Requirements

Risks & Contingencies

Reports

Screens

Messages

These are the basic elements of the requirements document and will appear to be sparse when compared to projects that are thoroughly planned.

Functional Requirements

The functional requirements contained in this document are the result of a customer interview, reviewing the reports generated by the system, reading the context sensitive help screens, and using each of the input screens.

Risks & Contingencies

It is possible that not all outputs have been identified. No documents have been identified that list all possible system error messages. Reasonable guesses will be used to test individual fields. Field testing will be performed because there are no unit test plans to be reviewed.

Reports:

Inventory Listing by Product Description
Inventory Listing by SKU
Inventory On-Order by Date

Screens:

Splash Screen, Logo
Main menu
Inventory Entry
Inventory Reporting

Messages:

Inventory Entry:	Invalid SKU
	Item not found
Inventory Reporting:	Print to screen, file, printer?
	Invalid date entered

Step 3: Provide management with an estimate of time and test coverage

Based upon the sizing activity it is now possible to provide an estimate of the testing effort. Using the organization's estimating methodology, it should now be possible to develop a reasonable estimate of test time and coverage. If there is not a standard method to estimate testing in the organization, consider the following guidelines:

10-15	System test cases per screen, medium to low risk situations
20-25	System test cases per screen, high risk
1-2	Test cases per report filtering criteria
1-2	Test cases per message

Sample Test Estimate

Test time and coverage estimate

	Risk	Est. Tests
Screens:		
Splash	Low	2
Main Menu	Low	5
Inv Entry	High	25
Inv Reporting	Medium	10
Messages		15
Reports		20
Integration		13
TOTAL		90

Test Planning-	@ 15/day, 90/15	= 6 days
Test Execution-	@ 25/day, 90/25	= 4 days
Debug & Fix-	Retests @ 25 = 1 day	
TOTAL		= 11 days

Step 4: Review requirements and start managing the project

The approximate size of the project has been determined and a basic estimate of testing effort computed. However, before proceeding we should review results via a walkthrough and then apprize management of the results. If possible, introduce a project management tool at this point to record the planned tests with schedules.

The project management software will also generate reports to demonstrate how much testing can be accomplished in the available time. Another major benefit of the project management tool is that changes are inevitably suggested during the review and the impact of the changes can be more easily assessed through the software.

Step 5: Automated testing decision

Decisions regarding automating the testing will reflect prior automation experience and the current expertise in the department. If the group has no experience with automation, this is probably not the best time to start learning. An exception to this might be the unusual circumstance where we are surprised by a project that is extremely large and our estimated testing is in the 1,000 hour range.

While it is not recommended that people learn to use automation tools while they are dealing with a new project, if you have the expertise, consider automating the testing. One approach that may be the most productive is to ask the testing group to develop their test cases and then turn them over to automation specialists. These specialists might be two or three individuals that have a job responsibility that encompasses automated tests for the rest of the group.

Remember to add time to the estimate if a decision is made to automate. Another factor to consider is that this project will be back and economical regression testing rarely is possible as an afterthought.

Step 6: Additional preparation

Schedules provide organization, but they don't create additional hours in the day to get the work done. Review your testing checklists, assemble new ones if necessary, and try not to leave anything to memory.

Document everything. Even if the you are working in a short time frame, everything must be documented. As questions arise, you will not have adequate time to research issues.

Produce frequent project status reports from your project management software.