QUARDEV
Integrity, Service, Solutions

SBT Lite

Components of Session-Based Test Management

Date: May 2007, Version 1

Author: Sam Kalman, Quardev, Inc.

An Overview of Session-Based Test Management

At this time, many words have been written about Session-Based Test Management (SBTM), also called Session-Based Testing (SBT). It is a method for testing software applications in an purely exploratory manner, while delivering just as much tester accountability as test cases and scripts provide. There are many benefits and considerations for using SBTM in any testing project. Some of the more widely recognized benefits include the following:

- **More flexibility** – The test team can change what they're doing quickly, at any time, and for any reason.
- **Less setup time** – Preparing to run a session does not take as much time as preparing a suite of test cases. Additionally, no time is lost on setup for test cases that might become legacy.
- **More accountability** – The strongest benefit of traditional exploratory testing is perhaps also its weakest – freedom. When exploratory testing is performed in sessions, the dynamic elements of exploratory testing can be maintained, but the freedom is backed by additional documentation. This documentation provides enough information about the activities during testing to track progress and other metrics.

In addition to the benefits, some widely recognized drawbacks of SBTM include the following:

- **More training required** – SBTM adds bureaucracy to exploratory testing. There is ramp-up and learning required of the lead and engineers performing the testing. It takes practice to produce a good session.
- **More mental weight on the tester** – The reporting elements will always sit on the back of the tester's mind, keeping their attention somewhat focused on reporting activities. This is mental energy that is taken away from the testing itself.
- **More time required for analyzing and reporting** – New data is created during each session. For the tester to be held accountable for that data, it must be analyzed and discussed by the leads and engineers involved. Again, this is time and energy focused away from performing testing duties.

These benefits and drawback outline some difficult trade-offs. The most important thing to remember is that employing SBTM on a project requires time. Two of the three described drawbacks consume time that could be spent testing. It is true that SBTM can save time during setup activities. However, when it comes to non-setup related activities, SBTM is clearly more time consuming than free-form exploratory or scripted testing. And since time is money, it can be

hard to make that time commitment without a clear return on investment. Even managers who are familiar with SBTM theory can be reluctant to adopt it as a practice.

What's misleading about SBTM is that it isn't a single entity. Rather, it is a bundle of practices and techniques that work harmoniously together. You don't have to use them all together. Instead of adopting and accepting the "clump" of SBTM with all of its drawbacks and benefits, you use one or two of the techniques to try it out as a "Lite" version. In this way, you can experience the types of benefits and considerations that are required for full blown SBTM. From there, you can ease into the full suite of all techniques, or retain only the best techniques for your project. In SBT Lite, each one of the practices and techniques that makes up full-blown SBTM is called a component.

Components - The SBT Lite Concept

SBT Lite is a streamlined version of SBTM. Where SBTM provides a full five course meal, SBT Lite provides a menu of components which you can order individually to include on your project. Each component has its own unique benefit and time cost, outlined below. This should prepare you to pick and choose the best components to maximize benefit and minimize costs for your project. Please keep in mind that the described costs are only a guideline.

There are two categories of components: Project components and Reporting components. Project components primarily involve processes and practices to follow as a test team at the project level. Reporting components primarily contain information recorded by the test engineer during a session of exploratory testing. This differentiation is made primarily because Reporting components are at the testing activity level, and Project components are at the project process & management level. as a whole are considered a Project component. It is possible to omit the Reporting components entirely. But if they are included, they can be customized at an additional level of detail. Let's dig into the different components now.

Project Components

The heart of Session-Based Testing is a unit of work time called the "session." It is the one component that cannot be omitted from SBT Lite. Without a session, there is no SBT at all – only free-form exploratory testing. Project components are the practices and tasks performed at a higher-level of the project. Your choice of Project components will mostly affect the time during the day that is not spent within a session. There are three Project components – the Session Report, the Debrief, and the Parsing Statistics.

Session Report

Cost: High

The Session Report is a collection of data that is recorded by the tester while they are running a session. The specific data on the report can be customized, which we'll discuss later. Generally speaking, it is this data which provides the accountability associated with SBTM. The data from the Session Report can be fed into the Parsing Statistics.

Debrief

Cost: Medium for Lead, Low for Engineers

The Debrief is a discussion between the lead and the tester after a session has been completed. Think of it as a mini post-mortem. The lead learns what the tester touched during the session, smooth and sticky points, and key focuses. The story that comes out of the Debrief can affect what type of testing will be done for the next session.

Parsing Statistics

Cost: Low w/Session Report, High w/out Session Report

The Parsing Statistics are used to quantify data recorded during Sessions. A tool is used to parse through the data from Session Reports and generate high-level information collected throughout the testing project. This information can then be shared with the client or stakeholder. It can also provide insight for the Lead, so they can steer the project in the right direction. If the Session Report is omitted, any desired data for Parsing Statistics will need to be stored elsewhere.

Reporting Components

The Reporting components are part of the Session Report. When the tester performs a session, each type of data they record is a different Reporting component. These components each correlate to a different data field on the Session Report. As more Reporting components are included, the overall cost of the Session Report increases.

Charter

Cost: High

The charter is the mission of the session. It is the tester's focus, area, or task to accomplish. The charter can be determined from a list of pre-established charters, or it can be created immediately before the session is run. It can also be created as a result of a Debrief.

Tester

Cost: Low

This is simply the name of the engineer who performed the testing. If this is included, personal accountability can be traced.

Configuration

Cost: Low

The configuration is generally the testing environment. It can include hardware or software variables. For example, web compatibility testing might include OS, Browser Type, and Browser version as configuration variables.

Area

Cost: Medium

This is an outline of the features or functionality that were tested during the session. This can be closely tied with the charter. It can also be edited or changed as exploration continues. When Parsing statistics are used, this gives wonderful indication of coverage.

Start Time
Cost: Low

The exact time the testing session began. This is only important for projects that require extreme accountability for time.

Duration
Cost: Low

A loose idea of how long the session lasted. Short, normal, or long. This provides a good rough estimate of how complex the charter or area turned out to be.

Notes
Cost: High

Notes are the tester's breadcrumbs about what happened during the session. They paint a path followed during testing. They are used in the Debrief, and help greatly with providing accountability for coverage.

Bugs
Cost: High

A simple list of the bugs entered. Bug number, severity, and title are recommended. It’s helpful to see how many bugs were uncovered during that particular charter.

Issues
Cost: Medium

Questions or problems that arose that could be indicative of bugs. These are usually something that seems like a bug, but it’s not entirely clear. They can also be items that put testing progress at risk.

TBS Metrics
Cost: Low

A percentage representation of time spent in Testing, Bug Investigation, and Setup tasks. This provides another angle into the types of activities that went into the charter.

Charter vs. Opportunity
Cost: Medium

This component leverages the strength of the exploratory nature of sessions. The common scenario is for a tester to be following the charter, then notice something that looks like a problem but is not inside the charter. They are free to investigate the new issue at their discretion. They simply categorize this time as opportunity time. They report their time under charter and opportunity as percentages.

SBT Lite in Action

This concept of flexible and selective component of SBTM was put to the test on a real-world project in the first quarter of 2007. This section will recount the process and dynamics of the project, and recount how the adaptable aspects of SBT Lite were beneficial in the long-run.

Mission

I was the lead of a three-person project to test an add-in for Microsoft Outlook. Our team was brought onboard to supplement an existing (but very small) internal test team. Our scope was full-experience. Setup, functionality, integration, auto-update, documentation, and auto-repair were all in scope plus more. We were provided some narrow test cases of limited value by an internal tester who was scheduled to transition off the team shortly. Our goal was to provide thorough testing of all areas in six different configurations and provide adequate testing in place of the departing internal tester.

Tactics

Because we had a smaller team and a moderately large scope, we decided that SBT Lite would be the better approach when compared to developing and executing test cases. The product had a history of changing rapidly with an occasional fire alarm. With this in mind, we determined that spending time developing test cases which could (and likely would) quickly become irrelevant was not the best use of our time. Instead, we invested some time in setting up a customized tool in SharePoint to track and maintain all our sessions.

The SharePoint tool was made primarily for tracking purposes. Our client had no knowledge of SBT, and didn't seem interested in gathering any metrics when the project began. By using SharePoint's integration with Excel, we knew that building a reporting tool would not be too difficult.

Project Beginning

Our project began with some ramp-up on the ideas behind SBTM, including some one-on-one time with Jon Bach. Of my team, I was the only one who had previous experience employing SBTM on a testing project. For this and other reasons, we decided against using the full-blown SBTM suite and opted for a simplified SBT Lite approach. We spent plenty of time talking about how to use SBT Lite, including Session Reports and creating Charters. We might have spent too much time in theory mode. There was an overwhelming feeling that SBT and the underlying techniques wouldn't be fully understood until we actually began using it. This was evident in a Charter brainstorming meeting, where it was nearly impossible to break the habit of anticipating individual tests to be run, rather than broader areas to explore in a Charter.

Our next step was entering all our anticipated variables into our SharePoint Tool. With the information we had collected so far, and the items we anticipated tracking, we predicted data sets and created them. In hindsight, we did not create all the variables that would be needed for the project, and we paid a small price for this later in the project. This will be discussed in further detail later on.

Internally, we also had to work to establish expectations regarding the level of detail in the Notes component of the Session Reports. This was the first SBT project for both of the engineers, and

they required a bit of extra practice to learn how to keep thorough notes. This didn't take more than two days to work through. Holding short Debriefs with each engineer post-session was instrumental in clarifying and reaching expectations.

We began running sessions regularly throughout the day, and providing the Session Reports at the end of each day. Our client began replying to each one of our issues with clarifying information, questions, or requests to enter them as bugs. These session feedback emails became a consistent element of our project. It was evident that our client was reading the reports at the end of each day. Sometimes a valuable dialog was opened as a result of the Session Reports, evidence that the time investment on both ends was worthwhile.

Adapting to Shortcomings

As we came closer to the middle of the project, it became clear that we were not recording all the data that would be required for proper coverage tracking. This was the first time our lack of comprehensive planning would bite us. Luckily, we were in a position where not too much data had been gathered yet. Using our SharePoint tool, we added new categories to track variables of the project.

For quite some time after this, we settled into a groove of running sessions. Reports were detailed, feedback was abundant, and progress was being made. The consistency of Notes in the Session Reports had leveled out, and Debriefs were not yielding any critical information or project implications. With that in mind, we stopped conducting Debriefs. Our client had expressed appreciation in the regular Session Reports and our reputation from his perspective was very favorable.

The project continued running, and one engineer had a week's worth of vacation time planned. A different engineer was brought in to keep the team size consistent. The new engineer also required some ramp-up on SBT methodologies. We had plenty of completed Session Reports for the engineer to review, and his ramp-up was quite speedy. The number of Session Reports we submitted by the end of that week were slightly lower than our stride numbers, but still very impressive when the complete SBT ramp-up for one engineer was taken into account.

Our client continued to be satisfied with the information we were providing. This was evident as we were requested to perform new types of testing in different areas that were not previously in scope. After a long while of consistently running sessions, we began to see some lull days where high accountability was not necessary. With the client's permission, we took a break from running Sessions to execute free-form exploratory testing instead. The mental weight that was lifted from lack of Session Reporting was a nice break. One of the engineers actually chose to continue running sessions, much to my surprise.

The next person to have schedule vacation was me. At this later point in the project, there was established credibility between us and our client. Internally, there was established routine amongst the team. Needless to say, I felt no risk would come about from my absence. Indeed, my team was able to ramp-up another new tester who would replace me during my time away. The week of sessions continued without a snag.

Downhill Slope

When I returned, our client began expressing a desire to see some more detailed information about the amount of testing we had performed in any given area. This had positive and negative implications. Thankfully, we had anticipated a request for a Parsing Statistics tool might come along, and we were prepared to add that as a Project component. Our approach was to export our data to Excel from SharePoint, and extract metrics with a macro. Unfortunately we had not actively been tracking the areas we had been testing. Instead, we had been relying on the notes in each Session Report to be sufficient. To gather data of this type, we had to manually review all the completed Session Reports (almost 250!) and add this information. This is a perfect example of how a conscious decision to omit a Reporting component at the start of the project turned out to be the wrong decision. If we had decided to track the areas of testing from the beginning, implementing the new areas would have been virtually painless. Nevertheless, we worked through this shortcoming and came out with more data at the end.

The macros were completed and detailed reports were provided to the client two weeks before the project ended. Our client was extremely happy to see the data, and our numbers provided that extra confidence of our coverage. More importantly, the client engaged us in a discussion about the meaning behind the data. All-in-all, the investment to develop the Parsing Statistics component was well spent and worthwhile. When our project concluded, we received many words of appreciation from different stakeholders about the coverage we attained. It was unanimous that this version of the product was the most stable release ever.

Stakeholder Feedback

What you've just read is a true story about how SBT Lite was initially planned, organized, and adapted as the project continued. This perfectly reflects the core value of SBT Lite components, which is to mix and match the techniques used and data gathered to best fit the project and the stakeholders' needs. Toward the end of the project, I asked for an honest assessment of our execution of SBT Lite for the project. These are the words our client had to say:

> "I think, if I could have spared the time early on, I would have recommended a brainstorming session or two on charter development, and maybe maintain that in a more coordinated fashion than we did [...] along the lines of generating more charters in varied areas, some of the charters didn't get created until we ran into issues, I think a bit more planning on my part might have been able to prevent that."

It is true that for our project, many severe issues were discovered which then prompted us to create investigative charters. If we had plotted out our charters in a more detailed fashion, areas around severe issues would have been thoroughly covered at the planning stages. The lesson learned is: Getting a detailed group of charters at the early stages will lay a better groundwork for finding issues as a result of running sessions, instead of creating sessions to investigate issues that were discovered.

Lessons Learned

Our project suffered from lack of thorough anticipation of reporting requirements, and it cost us time later in the project. Part of our project required adaptation, as our client himself realized that new information would be valuable late in the project. We made the decision to spend the time required to add the new component. Components can be added or removed during the project as needs change. The trade-off is the time required to implement the changes. From this perspective, preparing a Session tracking mechanism which allows for creation and removal of data categories is extremely valuable. With such a system in place, a Test Lead can create brand new components of their own design. Jon and James Bach's Session Report format (http://www.quardev.com/whitepapers/sbtm.pdf) and the Perl scan tool (http://www.quardev.com/content/tools/sessions.exe), are an excellent pilot for this type of flexibility.

Suggested Variations of Component Combinations

Now that we've discussed the meaning of SBT Lite and provided a comprehensive real-world example, let's explore some fictional scenarios in which some more extreme or unexpected flavors of SBT are appropriate. We'll begin by customizing the Reporting components, the data recorded by testers on the Session Report.

No Charter

You wouldn't want to use a charter if you want to bring more of a free-form approach to each session. This allows you to keep full flexibility of work performed while still tracking progress and providing accountability.

Only Notes, Bugs, and Issues

This combination can be useful if the environment or area is not important to track. Testing a database for a website might be a good example for this kind of session report.

Only Charter and Bugs

This is best if the stakeholders don't care how the testing was performed. They just want to know the testing focus and bugs found. They'll have to read the charters themselves.

Configuration, Area, Bugs, and Issues

Most notably, this omits Notes on its own. This is primarily to give the stakeholder a very high level of coverage. It also gives the tester a little less mental weight to keep track of their notes while they are testing.

Now we'll talk about Project component Combinations

Session Report Only

This is best when the testers on the project are all familiar with SBT. It's also critical to ensure that the client does not want any detailed reports. This combination is by far the easiest for the Lead to manage.

Session Report + Debrief

This combo is a good pairing for testers who are learning the ropes of SBT.

Debrief Only

This is fantastic for a paperless version of SBT! Rather than recording data, all the dynamics are openly discussed amongst the team members.

Parsing Statistics only

You'll have to enter the data somewhere so it can be parsed. To reduce the overhead of this task, it's important to slim down the collected data as much as possible.

Your Own SBT Lite Flavor

The next step is to create a customization for your own testing project. How is this done? Start with a suggested combination that makes the most sense for your project's needs. From there, add or remove components. Remember that each component is optional, and there is no minimum requirement.

The challenge in using SBT Lite effectively is to understand what each individual component achieves, and what it "costs" in terms of time and training. The best way to gain this understanding is through experimentation and practice.

Conclusion

We have discussed the high level concept behind SBT Lite; customizing a combination of components to create a unique system of Session-Based Test Management. We introduced the categories of components, both Project and Reporting and analyzed the individual components within those categories. We provided a real-world example of SBT Lite and how it was adapted to fit the project needs. We also suggested some interesting SBT Lite deployment possibilities for specific needs or project dynamics. Finally, we provided some considerations for picking the best SBT Lite options for your own project. To continue your journey of SBT Lite, you must take the next step and try it for yourself. See if SBT Lite can make your testing better, smarter, and more efficient.

Bibliography

Bach, Jonathan. "Session-Based Test Management." Quardev Laboratories. 2000.
http://www.quardev.com/whitepapers/sbtm.pdf