We've only got **One Life to Live**, so we shouldn't waste it on ineffective test design. Follow this **Guiding Light** to **The Bold and the Beautiful** world of

Soap Opera Testing

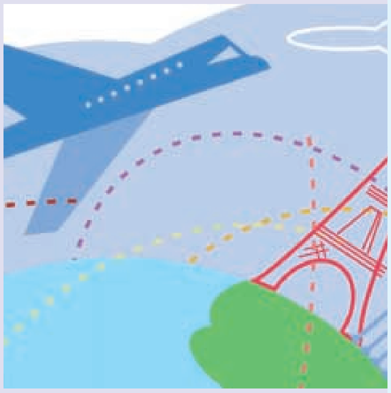
BY HANS BUWALDA

TEST PROCESSES ARE SUCCESSFUL IF THEY are effective, efficient, manageable, and fun. A major factor in how well these four objectives are achieved is the approach to making test cases. One such approach is "soap opera" testing.

As many readers know, soap operas are dramatic daytime television shows that were originally sponsored by soap vendors. They depict life in a way that viewers can relate to, but the situations portrayed are typically condensed and exaggerated. In one episode, more things happen to the characters than most of us will experience in a lifetime. Opinions may differ about whether soap operas are fun to watch, but it *must* be great fun to write them. Soap opera testing is similar to a soap opera in that tests are based on real life, exaggerated, and condensed.

Info to Go

- Tests should be fun and aggressive.
- Add structure to retain, and even improve, manageability.
- Write scenarios that are reality-based, exaggerated, and condensed.



ILLUSTRATIONS BY LISA HENDERLING

The idea for turning test cases into soap operas came to me while testing a new financial system. A group of end users was mobilized to come up with a large number of test cases quickly. It was important that the test cases be very good and very aggressive, even though time was limited due to extra pressure from the potential Y2K problem. Because the system had to do sensitive work such as calculating pensions for retirees, errors were a big no-no.

The end users were the people with the most practical knowledge, but they had no IT or QA background. The testers lacked the proper financial background and the day-to-day experience. To solve this, the testers and end users were asked to sit together in small work groups (four to five people each) and come up with stories based on the most extreme examples that had happened, or that could happen in practice. Imagination was invited and exaggeration was welcome. To help the process, I asked the groups to imagine that they were writing soap operas. This helped create lean, mean test cases fast. It also made the whole experience a bit more fun, which was important in this situation, where end users had to put in a lot of time testing—something not in their job descriptions.

Mechanical Approaches to Testing

To set the stage for soap opera testing, I want to contrast it with the more common ways test cases are usually made, which I call the "mechanical approaches."

By a mechanical approach, I mean developing tests following a simple, prestructured process. For example, consider the following process. Start with a finite number of screens or requirements. For each screen (or requirement) in a system, make a test case that inputs values in that screen. After entering the values for each screen, verify that those values appear in the system. Then, check whether the system has caught illegal input, such as mandatory fields that haven't been filled in. If the above tests are completed successfully, testing is complete.

These tests may be developed "top down"—break down functional specifications and/or requirements until the items are elementary enough to create tests for them. Or they may be developed "bottom up"—make one or more tests for each small unit in a system, then organize those tests into larger suites. These are analytical ways of thinking, common for IT people. For testing, however, a better direction is what I like to call "outside in"—work from the business environment toward the system under test. This is the essence of soap opera testing.

I'm not saying that mechanical approaches aren't any good. On the contrary, mechanical approaches tend to be straightforward and give reasonable confidence that the functionalities in the system under test are in good

Count the Goodies

BY HANS BUWALDA

Our soap, called "Count the Goodies," is about the Goody family.

In this episode, the Goodys have won $750,000 in the lottery. They decide to buy that big house on the corner of the street that they've always dreamed about. Father, Bing Goody, goes to the banking office to make the necessary arrangements. Of course, the bankruptcy that Bing had two years ago complicates the approval, but Jim, the guy at the bank, is very helpful and even sells Bing on the idea of a beautiful vacation property in Mexico. Bing has a brilliant idea. Why not ask neighbor Jones if he is interested in co-owning the vacation property? Like everything else with Mr. Jones, his credit is perfect.

One week later: After a small party with some new friends, only about $720,000 of the prize money is left, so the mortgage arrangements need to be changed. Jim, Mr. Jones, and Bing decide to meet the next morning to make the necessary adjustments. Since it was a great party, they are not at their best when they meet, so many mistakes are made. They correct some immediately, others they miss and will need to fix later. They find that an additional second mortgage on Mr. Jones's home is needed in order to still qualify for the vacation home.

A story like this can potentially test all kinds of objectives, such as

- Entering a customer with a big family
- Mortgage arrangements including a first and a second house
- Property abroad
- Both a primary residence and a vacation property
- Weighing of the income of a second owner
- Co-ownership on the second house only
- Establishing a second mortgage to finance a vacation property
- Bankruptcy two years ago
- Modification of the down payment
- Correcting errors upon entry
- Corrections after the application has been processed

Disorder Depot

BY LISA CRISPIN

There are 20 preorders for Elite Force Aviator George W. Bush Action Figure in Enterprise awaiting the receipt of the items in the warehouse. Finally, the great day arrives, and Jane at the warehouse receives 100 of the action figures as available inventory against the purchase order. She updates the item record in Enterprise to show it is no longer a preorder. Some time passes, during which the Enterprise background workflow to release preorders runs. The 20 orders are pick-released and sent down to the warehouse. Meanwhile, Joe loses control of his forklift and accidentally drives it into the shelf containing the Bush action figures. All appear to be shredded to bits. Jane, horrified, removes all 100 items from available inventory with a miscellaneous issue. Meanwhile, more orders for this very popular item have come in to Enterprise. Sorting through the rubble, Jane and Joe find that 14 of the action figures have survived intact in their boxes. Jane adds them back into available inventory with a miscellaneous receipt.

This scenario tests

- Preorder process
- PO receipt process
- Miscellaneous receipt and issue
- Backorder process
- Pick-release process
- Preorder release process
- Warehouse cancels

enough shape for normal, everyday usage. For many systems, such knowledge relieves a lot of concerns; if anything does go wrong after release, it can probably be fixed with manageable impact.

Also, since mechanical test cases tend to correspond to tangible things like screens or requirements, their development and execution are easy to manage. It is possible to predict how many test cases have to be developed and executed at any point in the project. But this also leads to a disadvantage of mechanical test cases: Since the individual tests are usually small, they come in great numbers, which can be hard to organize and track, particularly when there are multiple versions of a system under test, such as foreign languages. For example, when a system needs to undergo adaptations, it will be hard to know which tests are impacted. Getting the test cases up-to-date again can become a prohibitive task.

The biggest concern I have with mechanical testing, however, is the typically low "ambition level." By that I mean the degree of aggressiveness designed into a test to find hidden problems. Most tests I see in organizations are not overly ambitious; if all requirements are met and every screen works, the tester and the QA manager are happy. That might be fine at a certain point in the process, but later on some tests should be run that aggressively pressure the system to find hidden problems. Even if you don't fix the issues immediately, it is always valuable to know what can break your system.

Last but not least: Always following a predefined process is just not much fun. It doesn't stimulate creativity, which is a key asset for a good software tester.

You may be wondering, "What about exploratory testing? It typically does not follow a predefined process and also leads to testing with a higher ambition level." I agree. Exploratory testing is "ad hoc" in the good sense. Thanks to the work done by James Bach and others, procedures for exploratory testing are now available that easily outperform mechanical tests in efficiency and effectiveness. However, in exploratory testing, test design and text execution intermingle. The soap opera testing technique, on the other hand, has separate test design and test execution phases. Tests are described and reviewed by stakeholders first, before they are (potentially much later) executed, preferably by automation.

Imagine, Weed, and Feed

Soap opera testing gives you all of the confidence of mechanical testing but adds the complexity necessary for highly ambitious testing. It also allows you to maintain separation between test design and test execution. Your testing can be automated but not bogged down in a large number of tests to organize and track. Finally, it allows you to have fun and be creative, key to getting your testing mind flowing.

To create effective soap opera tests, first write the situations without worrying too much about the technical details of the system under test. Try to immerse yourself into the business the system was designed for and take it from there. Don't try to invent completely new stories every time. Instead, do the same thing "real" soap operas do: Invent a fixed set of characters that experience (are subjected to) various events. This way the test cases become like "episodes," which are easier to come up with. For instance, in the case of a financial system, the cast could be a family with parents, children, grandparents, etc. We'll call them the Smith family. Depending on the scope of the tests needed, surround them with neighbors ("the Joneses"), the local bank office, and in case security is an issue, add bank robbers, con men, tax officials, etc. The resulting "world" is rich enough to produce financially relevant situations with ease.

In one episode, one of the Smiths' children is getting engaged to one of the Joneses' children. The couple looks for a house and begins applying for a mortgage and a loan. However, somewhere in the process the engagement is called off. Similarly, the children might drop out of school, start working, or study abroad. Various family members marry and divorce, become ill and die—all in one hour, just like on television.

At this point, the events have not necessarily been translated into financial terms, let alone into transactions for the system under test. The next step is to do these translations, while at the same time weeding and feeding. Leave out what is not relevant, and come up with additional events that create extra trouble for the system under test. For example, father does the finances, but he often lives with his head in the clouds, thus making mistakes that have to be corrected (financial systems usually don't like rollbacks). And the bank subsidiary is in the middle of a re-org, resulting in changes in authorizations, creating even more mistakes. A robbery results in having to recover lost data, etc.

You may have to gather more information in order to feed your tests. I've been surprised by how hard it can be to base tests on "real life." In more than one case I have had a hard time placing myself into a particular environment with enough feel to work with it creatively. That's when cooperation with others, such as end users and subject-matter experts, is so helpful. If challenged, they can come up with all kinds of nice anecdotes that provide excellent inspiration for mean tests.

In other words, you cannot always just stay in your cubicle and think hard. It might be necessary to go out, talk to people, and watch what is going on. This might need some management approval (show this article to your supervisor), since it looks time-consuming, but it is much more likely to save significant time and money, as tests can be produced faster, and the tests are leaner and meaner. From a business perspective, there are even more savings and mitigation of risks and problems because more bugs are found and they are found earlier.

Flights of Fancy

BY HANS SCHAEFER

Hans likes to travel in the cheapest possible way, so he often combines business and holiday trips. He lives in Norway but has to give a speech at a conference in Amsterdam. Afterwards, he wants to go for a holiday in Beijing. He also wants maximum benefits through his airline bonus program, so he prefers to use "Star Alliance" flights.

He asks for a tourist-class flight to Amsterdam, returning through Copenhagen, Denmark. Additionally, he asks for a tourist-class flight from Amsterdam to Beijing two days later, returning in two weeks. He wants the cheapest possible price, so he is flexible about the return date, asking for prices for three different days. His other hidden agenda is cutting corners on the way back from China: If the flight goes through Copenhagen, he would not need to go all the way to Amsterdam but could just fly back home directly.

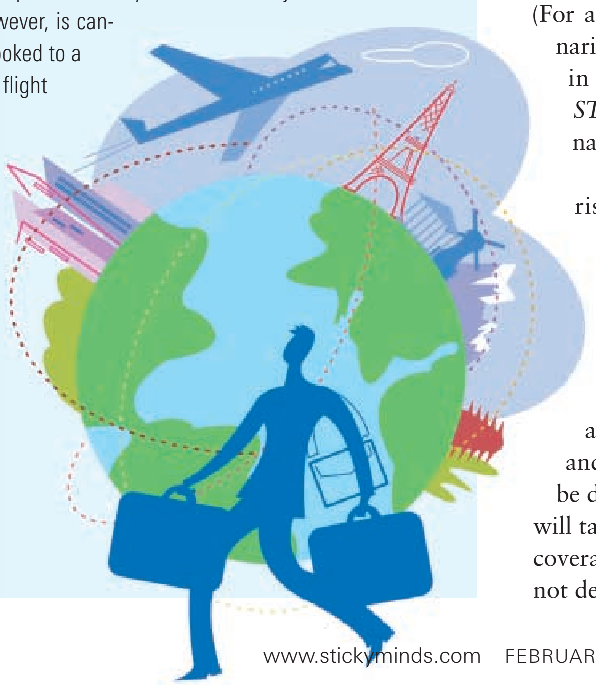
He gets a few flights and prices, but finds it too expensive. He asks for other flights between Amsterdam and China, not only through Copenhagen. There is one, considerably cheaper, through Frankfurt, Germany. He chooses this one.

Then a German customer wants to schedule a meeting. Hans decides to put it into his travel plans and changes the return flight, with a several-hour break in Frankfurt. A week later, the deadline for printing the tourist-class ticket runs out, but Hans does not yet want to get the final ticket. He changes the reservation again, changing the date for the outbound journey Amsterdam–Beijing to one day later. The new deadline for printing the ticket comes up, and finally all the tickets are printed and paid for.

Now comes the flight.

The first leg is overbooked, so Hans chooses to accept the airline's offer and takes the next flight. However, his baggage is lost. It turns up the next day in Amsterdam, just in time for checkin for the Beijing flight. This flight, however, is cancelled due to aircraft failure. His flight is rebooked to a different airline the same day (for free). This flight goes through Paris, but the first leg is delayed, so he does not reach the connecting flight to Beijing. Again, a rebooking results and he finally makes it.

Conditions tested include
- Searching for alternative flights
- Complicated connections
- Rebookings
- Not using booked flights
- Overbooking
- Delays and related service
- Lost baggage
- Rebooking on the way
- Ticketing deadline



Add Structure

Soap operas are a kind of scenario test. (For a more general treatment of scenario testing, see Cem Kaner's article in the September 2003 issue of *STQE*.) As with other kinds of scenario testing, there can be pitfalls.

For instance, manageability is a risk for soap opera testing. Do soap operas help when we have to test a large amount of functionality? Can the process be kept under control from a managerial perspective? What about coverage? It can be hard to predict and control the number of tests to be developed and how much time it will take; plus, it is not clear what the coverage of the tests is since they are not developed systematically.

To address this problem, let me first talk about test objectives. Test objectives are basically a list of statements, usually about the behavior of the system, that describe minute properties that the tests minimally need to address.

Here are some examples of test objectives:

- a negative value is not allowed for the age field

- if a person works part time, pension rights are built up for the total of the hours worked

- when server X goes down, server Y takes over

- the response time stays below 8 seconds when 5000 users visit our home page in a 5-minute period

- the HELP key works for each screen in subsystem B

In the case of a mechanical test technique, this connection will typically be a one-to-one relationship between test objectives and tests (see Figure 1): one objective is met by one test case, and each test case addresses one objective. For a creative technique like soap opera testing, however, this relation is more likely to be many-to-many: one objective can be tested in several scenarios, and a single scenario might capture a multitude of objectives (see Figure 2). By relating tests to test objectives, you can ensure coverage for soap opera tests.

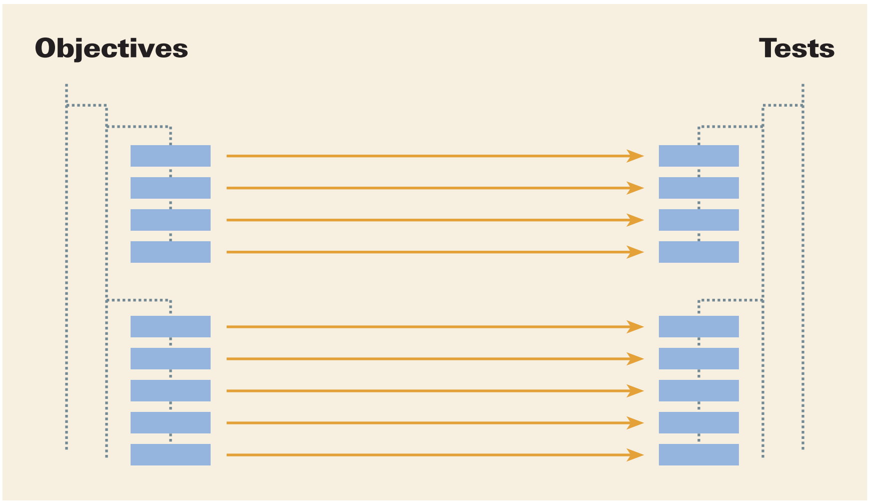


Figure 1: Mechanical model—tests produced one by one from objectives

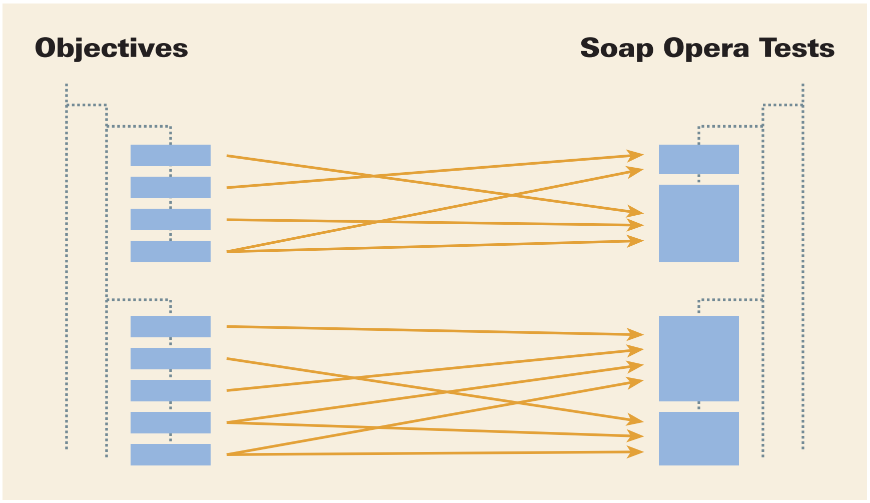


Figure 2: Soap opera tests have a many-to-many relationship

It can make sense to relate objectives and tests in a late stage; first develop the tests, and then determine whether these cover the objectives. When not all objectives are matched, either extend the existing scenarios and/or develop additional ones. This late matching can be serendipitous: things are tested that weren't intended, but it might uncover unexpected problems (in fact, bugs are unexpected by definition; otherwise they wouldn't be there, would they?). Another idea is to use different people for the tasks; for example, a more analytical person to do the test objectives and a more creative one to come up with the stories.

A system of any substantial size will need so many tests and test objectives that you can't tractably treat them as a single lump. Instead, break them into manageable clusters. A cluster is a set of tests with a similar and specific scope. Such a scope could be "functional tests for module XYZ in system ABC" or "performance tests for Web page W." When clustering, it is essential to not follow an exclusively top-down hierarchy but to divide the tests according to a combination of different criteria, such as which functional area of the system is addressed or what the intended ambition level of a test is. Clusters with a low ambition level and with small simple test cases typically would go first in the execution.

If possible, clustering should be done prior to developing individual tests. Therefore, the result of the clustering is a list of *potential* cluster candidates. For each individual cluster on the list, quick decisions can be made as to if and when it would need to be developed and executed, and who would be involved in doing so. Also, a choice can be made about the testing techniques to apply, like soap opera testing.

I call this combination of soap opera testing and traceability to test objectives "structured soap opera testing" (see Figure 3). It is lightweight enough to encourage creativity, and also complements the soap opera technique.

Vary Complexity

Since scenarios combine a lot of features, early in the test cycle when a system still has a lot of simple bugs, such as non-

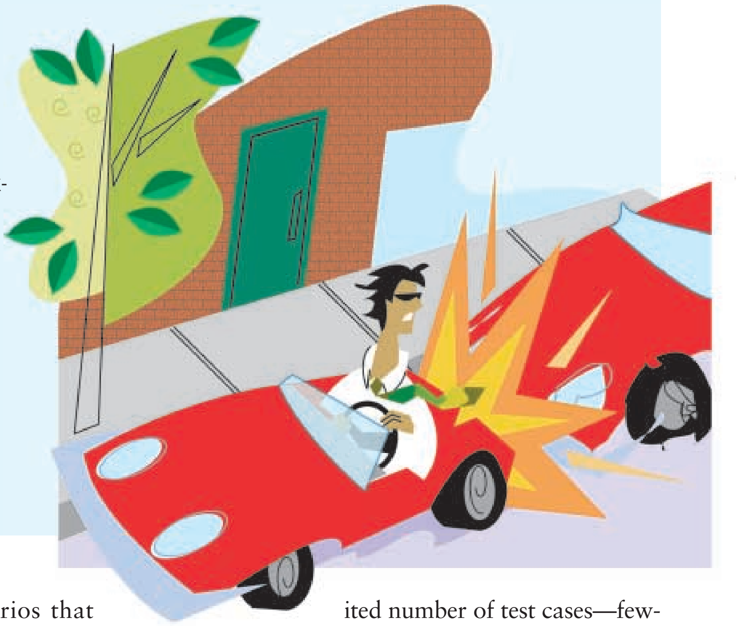
The Rented and the Wrecked

BY BRIAN MARICK

A customer named Marick hires a car for a three-day business trip. Midway through the rental, he extends it for another week. (This, by the way, gives him enough rental points to reach Preferred status.) Several days later, he calls to report that the car has been stolen. He insists that the Preferred benefit of on-site replacement applies, even though he was not Preferred at the start of the rental. A new car is delivered to him. Two days after that, he calls to report that the "stolen" car has been found. It turns out he'd mis-remembered where he'd parked it. He wants one of the cars picked up and the appropriate transaction closed. Oh, and one other thing: the way he discovered the mislaid car was by backing into it with its replacement, so they're both damaged.

This scenario would test the following conditions

- Upgrading to Preferred status (during rental)
- Extending a rental
- Handling a stolen car
- On-site replacement
- Undoing on-site replacement
- Undoing handling of stolen car
- Return of a damaged car



functioning buttons, they can be a challenge to execute. To say it in soap opera terms: It doesn't make sense to write about a Hollywood wedding in a beautiful church if the lock on the church door is broken and you can't get in. The best way I know to limit this is to have both simple clusters to test the basics of a system and more advanced clusters, for example with soap operas, to run after the basic tests have passed.

Give It a Try

Try to write scenarios that are (1) based on real life, (2) exaggerated, and (3) condensed into a limited number of events. Cover functionalities that either are in the system under test or, even better, should have been in the system, but weren't recognized in the design process. Test cases should be efficient in design and able to inflict a lot of stress with minimal actions. Create a limited number of test cases—fewer, more efficient tests are easier to manage, can be executed faster, and produce fewer results to be analyzed. Use structure to overcome coverage pitfalls. Finally, remember to try the lock on the church door before the big day arrives, making sure nothing stands in the way of the great happiness that, as every soap opera writer knows, is so typical for Hollywood marriages. {end}

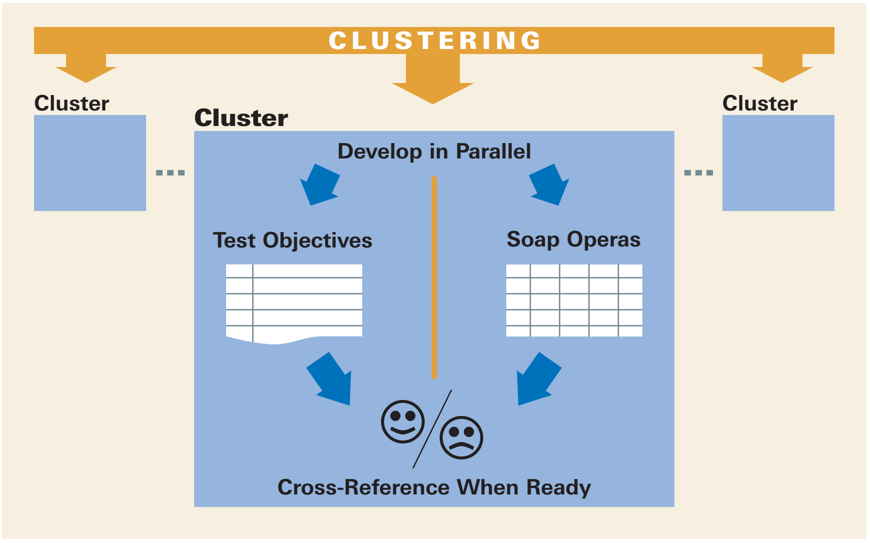


Figure 3: Structured soap opera testing

Hans Buwalda, now at LogiGear in California (and formerly at LogicaCMG in Europe), is the chief architect for Action Based Testing (ABT), a comprehensive method that is based on his ideas for creative and manageable testing and test automation, such as action words and soap opera testing. You can reach him through www.logigear.com.

Sticky Notes

For more on the following topics go to www.stickyminds.com/bettersoftware

- Soap operas and action words