“Those weren’t very accurate Script directions. Why have you brought me to the wrong place to test?”

White Paper by Mark Murray and Robbie McConnell

Abstract
This article describes a new context driven approach to software test automation. The content deals with how testing with context dramatically affects the issues of script reliance, script failure and maintenance compared to existing approaches.
This information is targeted at both experienced software testers and business managers who want to understand the issues involved.

Stopping your car at the side of the road, you ask for directions. You are told to *“Go 500 yards, take the second on the right, follow the road for another 800 yards and take the second on the left and finally take the third on the left and your destination is 100 yards on the right”*.

Unfortunately due to a private lane, you take the first right to start with and although this turning is only 100 yards from the intended one, having followed the rest of the directions faithfully you end up several miles away from your destination.

How can one small mistake be magnified to such an extent? The answer is of course that a serial list of directions relies completely on the ‘end state’ of the previous direction. As each individual step does not have any context, any small mistake will result in you becoming completely lost.

Current Automated Functional Testing tools suffer from this exact problem: the scripts used to navigate are a long serial list of individual instructions without any context.

To run these testing tools, you navigate to where you wish to test and on arrival perform a number of checks to ensure that your application is producing the results as expected.

While it would make sense to concentrate your time and resources on actually deciding how and what to test, in reality navigating to the correct place causes substantial issues and is often the major drain on your testing resources.

“Aha!”, we hear you say*, “What about the Play and Record facility? Surely that would take you accurately to where you want to test”*. While it is true that this facility will faithfully record the serial list of scripts, the testing community has found that except for extremely simple cases this facility is rarely used because it just doesn’t work reliably. It has therefore become universal practice for the scripts to be coded and maintained by hand.

What if we bunched these serial scripts into named usable bundles attached to an icon? Surely this would eliminate the problems? While it will certainly reduce the chance of a mistake within the actual bundle, the bundle itself is still totally reliant on the end state of the previous bundle. Again, as there is no context to cross-reference where you are, any mistake will be undetectable and you will again end up at the wrong destination.

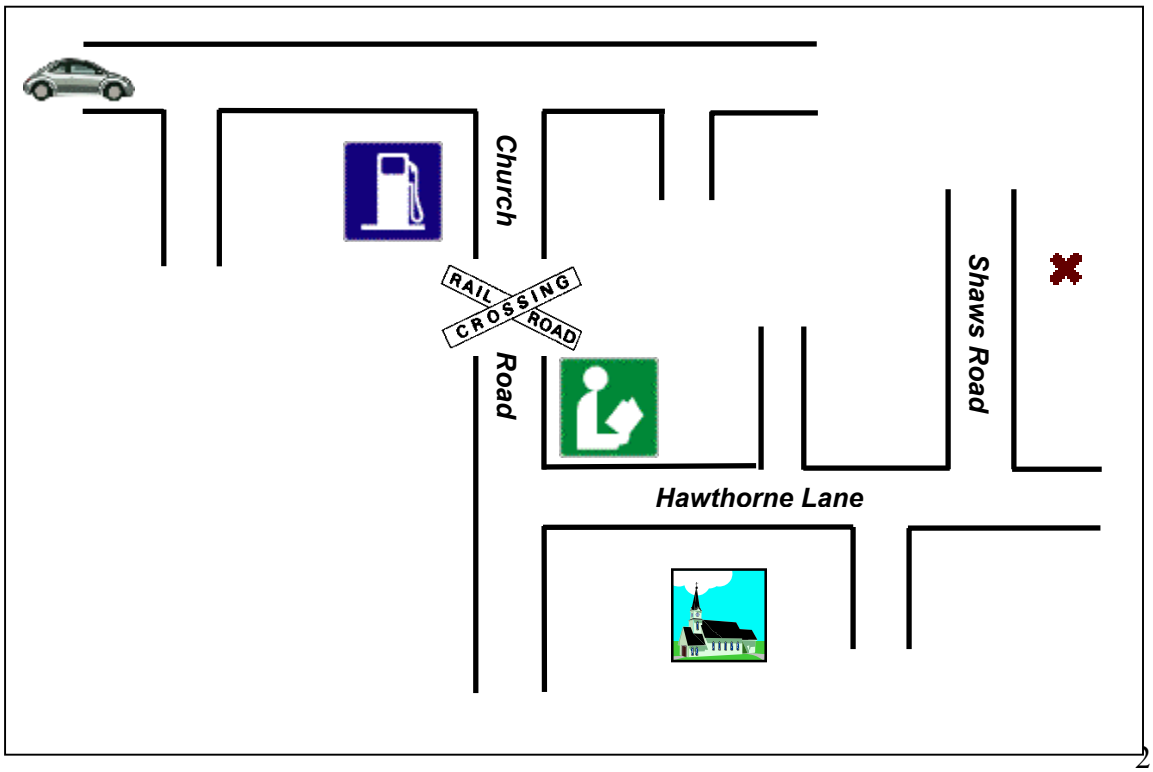
While the whole process of testing still needs to be treated as a development effort and the requirement for developing standards and processes for automation are still critical, is there a solution to help resolve these serial scripts issues and let the Tester spend more time on actual finding defects?

Imagine if you again stopped at the side of the road, and this time you were given the following directions: *"Go to the gas station and after passing take the next right called Church road, follow this going over a railroad crossing until you come to the village library and immediately take the next on the left called Hawthorn Lane. Follow this lane passing the church on your right before coming to the Good Food restaurant. Take the next road on your left called Shaws road and you will see your destination 100 yards on the right"*

Why are these directions considerably easier to follow? It is of course because you have context and are able to continually cross-reference where you are against where you should be. Each step has context in its own right and by being able to recognize places you are not completely reliant on the end state of the previous direction. Should you make a mistake, this is easily detected during the journey and you can retrace that step and find the correct route.

So could this be made even easier? How about a visual map with this context.

Figure 1



As well as giving you a visual overview, you can easily cross-reference and recognize, in context any place while traveling.

What would this 'map-based approach with context' do for functional test tools?

- Have a visual map of your test coverage
- Each application location on the map would have context
- Be able to simply point and click on the map to confirm you have a route to where you want to test
- Navigate on the map to where you want to test and then perform the test
- Due to the reduction in script programming the Business User as well as the programming tester would be kept involved.
- Reduce the time to test or increase the test coverage.

"Sounds great!", you say. *"What about the thorny issue of maintenance? Would this approach solve one of the major issues facing the Tester: If my application or process change, every single script affected by this change requires amending. Not only that, the tester must have an in depth knowledge of the proprietary script language to program these changes."*

To answer this, let's return to our directions.

Some time later you return to visit your destination again and refer to the serial directions i.e. *"Go 500 yards, take the second on the right, follow the road for another 800 yards etc"*. Unbeknown to you, due to a new housing development in the area, a new road between the first on the right and the correct second on the right has been built.

As you have no context and follow the directions serially you are going to end up in the middle of the new housing development at the end of your directions. Even worse, you will not even be aware that as you travel, you are going wrong.

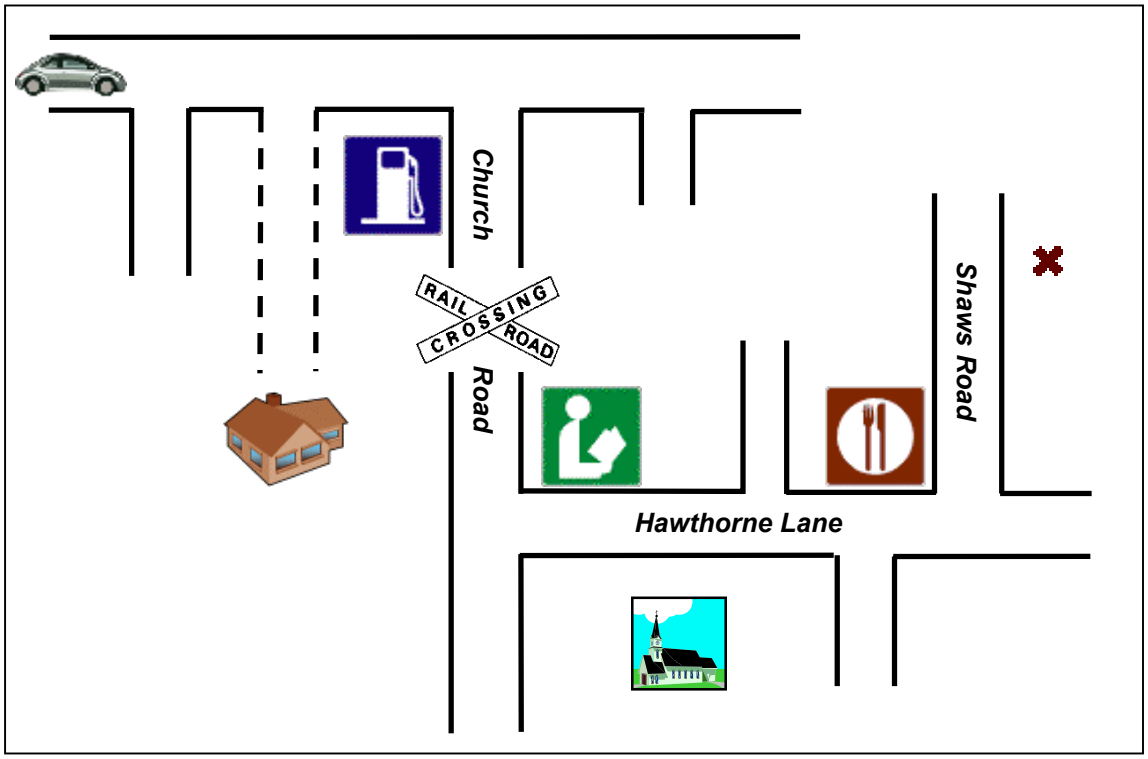
Moreover, any set of directions which go through this section of road will also be redundant and need to be individually changed.

Lets look at your directions with context:

Given that you know to turn right after the gas station into Church road, the fact that there is a new road just before the gas station is pretty much irrelevant, and given you have this context to turn after the gas station, you will follow the route accurately and correctly arrive at your final destination.

This is even more obvious when you add the new road to your map.

Figure 2



Most importantly, in updating the new additional road on the contextual map, any sets of directions that go through or pass this new section of road are also updated.

So what would this ‘map-based approach with context’ do for testing maintenance?

If you add a new place to your map, all other places still retain their context and therefore the map can easily be updated. This means:

- All routes through and passing this new place are updated.
- Point and click will define all the new routes.
- Simple to move to where you wish to test and perform your tests.
- Maintenance becomes simply the updating of one place and not individually having to re-program all the scripts that go through or pass that new place.
- Although you may still have small localized ‘scriptlets’, your overall script maintenance is massively reduced.
- Substantially reduces the time to test and increases test coverage.

Conclusion

Software testing without awareness of context can be a painful experience. Large amounts of time need to be spent ensuring that existing scripts continue to work effectively, barring non-technical users from much involvement at the "coalface" of testing. Testing with context can address this issue directly while still recognizing the need for scripts as a component of testing. In the real world when traveling between places in a car we have moved away from lists of serial directions and use maps (often with landmarks). Maybe it is time that we should consider doing this for automated software testing as well.

Bio

The authors have been creating software testing and automation tools for nearly 20 years and are best known as the authors of the QARun product acquired by Compuware Corporation in the mid 90's. The authors' continual frustrations with how difficult software testing tools are to use and maintain (including the ones they previously designed and implemented) has led them to continuously search for improvements in how to actually improve the process.