



PACIFIC NW
28TH ANNUAL
SOFTWARE
QUALITY
CONFERENCE

OCTOBER 18TH – 19TH, 2010



ACHIEVING
QUALITY
IN A COMPLEX
ENVIRONMENT

*Conference Paper Excerpt
from the*
CONFERENCE
PROCEEDINGS

Permission to copy, without fee, all or part of this material, except copyrighted material as noted, is granted provided that the copies are not made or distributed for commercial use.

Working with Complex Applications

Engin Uzuncaova
Software Design Engineer in Test, Bing Maps
Microsoft Corporation
enginuz@microsoft.com

Abstract

Software testing, in general, is a human-centric process where creativity and technical skills mix and mingle in unique ways to both understand and evaluate software systems. When we are faced with new and original testing challenges, it is usually the human element in the process that makes it possible for us to repackage the existing tools and methods, again, in new and original ways for viable solutions. As the complexity of a system under test increases, this task becomes harder to accomplish. What do we do when the level of understanding required to properly analyze and test such systems is beyond our comfort zone?

Testing geospatial data and route planning algorithms is a good example for this; the inherent complexity of geospatial data, the representation of the data and also novel algorithms that consume this data make testing more of a daunting (or fun?) task at both levels: understanding and analyzing. This paper presents a testing approach that we have developed at **Bing Maps**¹ to attack the complexity of our route planning service focusing on the following areas:

- Heuristic test oracles,
- Efficient test infrastructure,
- Rich visualization solutions for complex geospatial entities and scenarios, and finally
- Integration of statistical methods with the testing process.

This approach has provided for improved coverage in our testing and enabled us to identify more quality issues during development. We also benefited from a more efficient evaluation mechanism for complex issues found during testing. While our improvements in each individual area have proved to be highly valuable, the main benefit we observed with this approach has come from using all these improvements in tandem representing a strong test strategy.

Biography

Engin Uzuncaova is software development engineer in test for Bing Maps Directions team at Microsoft. He joined Bing Maps in October 2008 and he's been working on routing algorithms and data.

Previously, Engin interned at IBM working with the WebSphere Application Server team and at Google working with the Google Updater team. He received a master's degree in Computer Engineering from U.S. Naval Postgraduate School in 2003 and a Ph.D. degree in Software Engineering from the University of Texas at Austin in 2008; his dissertation was titled "Efficient Specification-based Testing Using Incremental Techniques".

¹ **Bing**® is a registered trademark of Microsoft Corporation.

1. Introduction

As software systems grow in size and complexity, testing them becomes more challenging, especially, since testing is often done manually. A key challenge is working with complex applications; i.e. applications that are based on sophisticated algorithms that use large and complex data structures. As we work with such applications, it becomes tougher for testing to achieve higher confidence and greater efficiency.

Testing starts by understanding the problem and then the solution, i.e., the software under test (SUT). As our understanding gets better, we develop more effective testing solutions to help us achieve the desired level of quality. Based on my experience with the route planning service in Bing Maps for almost two years, I believe it is a difficult testing problem. This paper presents a testing approach developed at Bing Maps to tackle the difficult challenges of working with complex geospatial data and sophisticated route planning algorithms.

Route planning service, in the simplest terms, can be considered as a user-friendly solution to the well-known shortest path problem [Cormen09] that is defined for an actual road graph. For today's mapping services, these graphs tend to include millions of edges and the algorithms that are used on this data are bounded by many functional and non-functional requirements.

When you are testing the route planning service, you mostly work with geospatial objects that are represented, basically, with coordinates (i.e., floating point numbers). Putting this into a context, given a user query (such as the driving directions from point A to point B), a routing algorithm finds an optimal path by examining the gigantic road graph. There are plenty of different routing algorithms that can be optimized based on various quality criteria [Delling09]. In the road graph, each edge represents an actual road segment with a large set of primitive properties such as speed and length; and some complex properties such as turn restrictions.

This paper focuses on three main problem areas that we have constantly dealt with in this domain:

1. The route planning domain involves many interesting and complex scenarios that need to be rigorously tested such as dense highway systems and missing/incorrect road data. Defining tests for such cases can be a tedious task. As the routing data gets updated, tests may become obsolete, which we may need to replace with valid ones.
2. Testing the route planning service requires executing the tests on a large set of test inputs, which usually goes through multiple iterations. Any inefficiency in test infrastructure causes unproductive test development cycles and lengthy execution and analysis times.
3. The main hurdle with test results in this domain is that there are usually large test suites used for certain functionalities and, as a result, there is a large body of results that need to be analyzed. This can be a difficult task when results represent complex test criteria, especially when optimizing subjective areas such as route-quality.

2. Example: Evaluating a Behavioral Change in the Routing Algorithm

This section illustrates a specific testing problem for evaluating a behavioral change in the routing algorithm and presents an approach to address the issue.

2.1. The Customer is Always Right..!

In a typical product life-cycle customer feedback is a constant input to quality improvement process. As an example, let's assume that we have received significant amount of customer feedback indicating that our driving directions favor low-quality roads, hence suggesting longer and non-ideal driving directions. After an investigation, the feature team verifies the problem and designs a solution to mitigate the route quality issue. Figure 1, hypothetically, depicts the original behavior on the left preferring inner roads, and the new behavior, on the right, preferring highways. It is now in the hands of the test team to check that the original issue is resolved, and also there is no undesired impact on the rest of the behavior.

2.2. Testing Cannot Show Absence of Bugs

With the new design, we can safely assume that some driving directions will be different than before, hopefully in a positive way. However, keep in mind that this is a subjective change, which cannot be measured with the same precision as a litmus test; making human judgment an important part of the evaluation process. In addition, we might need to go through multiple test iterations to optimize the new behavior and repeat the tests for different configurations, which necessitate a reasonably quick turnaround for test passes. Moreover, we might need to check a number of relevant non-functional requirements depending on the nature of changes introduced by the new design such as performance.

One of the main steps in this process is test generation, i.e. constructing a set of test inputs to be used for assessing correctness and quality. The test inputs we select must provide adequate coverage (e.g. geographical, functional and code). Obviously, given the size and complexity of the routing data, exhaustive testing would not be feasible and analyzing the results of a large test suite would introduce a substantial overhead. If we don't have reusable tools in place, developing an automated solution to do the trick might turn out to be a costly investment for such a specific problem.

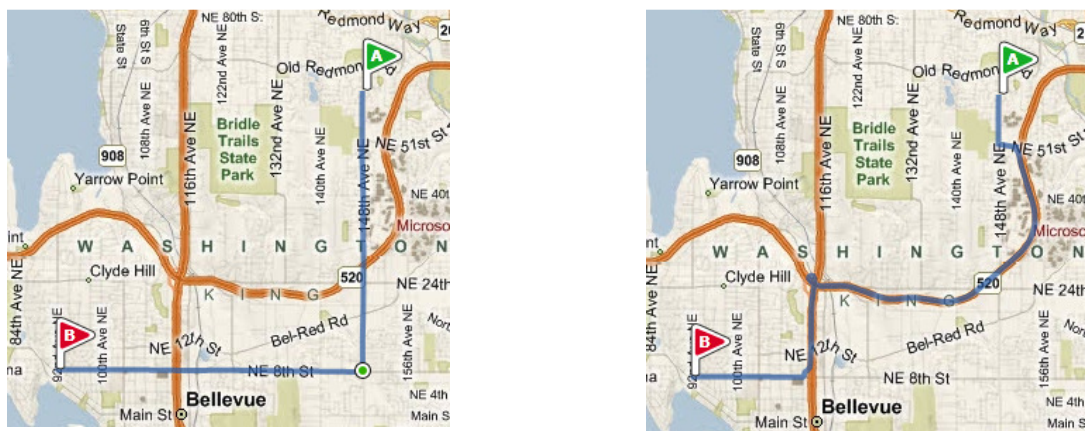


Figure 1 - Two different driving directions showing the original behavior on the left and the new behavior on the right.

As much as we would desire a minimal test suite, it is not unusual to work with thousands of test inputs in the routing domain for behavioral analysis. For instance, this is true for geographical coverage; coverage can only improve with more tests. As we increase the size of the test suite, however, test execution becomes more of a bottleneck considering that we will need to run these tests multiple times for different configurations.

Adding on top of that the ever-increasing pressure to release new features (to the feature-hungry end users), there is reasonable motivation to be traditional and stick to what we have at hand, let it be manual or automated. This pressure also leaves little room for a full-fledged formal analysis of the change.

2.3. An Approach to Tackle the Problem

For this particular problem, we used a number of different approaches and tools that we have invested on for some time. Working on similar problems over a prolonged time has given us the opportunity to think about various key investment areas to increase productivity and efficiency of our test efforts.

2.3.1. Test Execution

With a large enough test suite, it is more likely to achieve a higher confidence in our test results; however, this is generally restricted by the amount of time that we can afford. To mitigate this limitation, we redesigned our test framework to support parallel test execution. With this capability, we achieved an order of magnitude improvement in test execution time. This translates into being able to run tests faster, hence enabling shorter turn-around period for test feedback. It has also allowed for scaling up to larger test suites more efficiently.

2.3.2. Test Generation

One of the main issues with this kind of a behavioral change is measuring the unintended results of the change (i.e. regression). For that, we need to achieve significant coverage for topological entities (such as streets, state-highways, interstate-highways and their intersections) and geographical regions (different parts of the world represent dramatically different road structures).

One of the direct ways to generate test suites is to use guided-random test generation to improve coverage based on certain criteria. For example, a rich set of test inputs can be generated using custom graph traversal algorithms that run on top of the underlying routing data.

2.3.3. Heuristic Test Oracles

Heuristics provide an approximation to the optimal solution, which is particularly useful when what is correct cannot be defined precisely. As we get better heuristics, the chance of identifying issues with heuristics also increases. For our scenario, numerous different heuristics can be used to identify potential issues ranging from simple ones such as differences in driving distance and duration, and the number of turn instructions issued per route to more complex ones such as ranking driving directions on a set of predefined criteria to measure quality of the generated results.

2.3.4. Visualization Tools

Geospatial entities are usually represented complex properties and coordinates. Within the context of geospatial analysis, working with this data in the raw format is neither intuitive nor easily comprehensible. For example, let's assume that we are trying to understand why and how two driving directions are different. Key elements of this analysis are the properties of the geospatial entities (i.e. roads) and the algorithm behavior (i.e. optimum path selection). Using a very primitive approach as described above and looking at the raw data next to each other is, in its simplest terms, painful (imagine that doing this for hundreds of different instances!). Instead, a visualization tool that presents the essential data pertaining to the analysis in an intuitive way can play a vital role in the assessment process.

3. Problem Areas

This section describes the problems we have experienced while testing the route planning service. We categorized the challenges in this domain into three areas as in the following sections.

3.1. Test Generation

One of the weakest areas of testing practice is test input selection/generation. A common pitfall is the *hello-world syndrome*, as I call it. This can be described as the tendency of testers to use the most basic instances as test inputs especially when the test inputs are complex data structures. The main reasons for this include but are not limited to the lack of in-depth understanding of the SUT and the overhead of generating a large set of test inputs. In the long run, however, this pattern of behavior causes some gaps in test coverage, which triggers significant costs later on in the development process.

The desired approach in test generation is to test the SUT for all possible combinations of test inputs; however, this is usually infeasible, if not intractable. Therefore various techniques are used to sample the test input domain to satisfy certain coverage criteria such as equivalence partitioning and boundary value analysis [Hamlet88]. What makes such techniques less useful for complex data structures is because of the mere fact that they are complex and maintaining a consistent test suite can be a challenging task if the underlying data and/or its representation is dynamic in nature (such as changes in the routing). For example, a valid test case for the right turn instruction based on a boundary test case can become obsolete if the routing data updates the topology at the intersection even slightly.

To illustrate, think about the problem laid out in the example in Section 2. The optimization aims at improving the quality of the directions generated by the algorithm. As the algorithm is modified, the results that it generates are likely to be different (in the case that they are not different it is another interesting problem because our intent is to alter the behavior in a certain way). There are different coverage criteria when selecting the test inputs such as structural properties of the routing graph, geographical coverage and code coverage. To satisfy these criteria, manual approach is not an option due to being a tedious and error-prone process, whereas automated methods can prove to be useful in scaling up to generating large test suites.

3.2. Test Execution

As mentioned before, the route planning service works on top of a very large data structure that represents the road graph. The amount of functional test points and the corresponding coverage criteria

for each of these points necessitate use of large test suites. In addition, these tests are usually run multiple times in conjunction with the development efforts to evaluate certain aspects of the underlying data and the algorithms iteratively. Changing just one-line of code in the algorithm can initiate a lengthy test run including thousands of test cases. Automation is the preferred solution for executing these tests due to obvious efficiency concerns compared to manual approach. Even then, test execution can be a bottle neck especially under time constraints, which we are all very familiar with.

Continuing on the same example, executing the selected test suite involves generating a route result and collecting the required information for evaluation. Depending on the selected test suite, the execution time can be significant and causes longer turn-around time for test feedback and scheduling conflicts among other tests that may require the same resources.

3.3. Test Evaluation

Working with complex data introduces challenges not only for test generation and execution but also for evaluation and sometimes in a much more unanticipated ways. The evaluation process is usually full of false-negatives and false-positives. For example, a route-quality optimization may work for longer driving directions but for very short ones it may introduce an undesired behavior. In some other situations, data and functional issues may be hard to distinguish from each other; for example, what is reported as a non-optimal result may in fact be due to a problem in the routing graph such as incorrect road-speed or turn-restriction. Moreover, sometimes test results may fail to detect subtle issues, such as increased use of left turns due an optimization in the algorithm. Issues like this require a deep understanding of the subtleties in the domain, which can be gained through experience. The crucial component of an effective evaluation process is presentation of the relevant test data in intuitive ways.

Think about route results in general; they have a set of edges that represents a shortest path in the routing graph. Each edge has many properties such as connectivity, speed, shape, restrictions and many others. How do we go about verifying that the generated path is in fact the shortest? Besides, a result can fail in many different ways; a turn restriction may be violated, a non-optimal result can be generated or driving instructions may be incorrect. Each combination of these factors, as a failure, requires human involvement to verify the issue. Collecting all required data points on a case by case basis through manual means can bring evaluation process down to highly inefficient and unproductive levels.

4. Our Approach

This section describes our approach to address these issues presented in Section 3. We invested on four different areas: (1) improving the test infrastructure, (2) using heuristic-based testing, (3) using statistical methods, and (4) developing visualizations tools.

4.1. Improving the Test Infrastructure

The main motivation for improving the test infrastructure is to be able to transition into more productive test development process and be more efficient in test execution (Section 3.1).

4.1.1. Parallel Computing

To achieve a certain level of geographical coverage, we usually need to run our tests on a large number of test inputs and, especially for algorithm optimizations, usually through multiple iterations, which can end up taking significant amount of time. One obvious way to address this scalability issue is that we can reduce the number of test inputs so the tests complete faster. However, this approach has a direct negative impact on test coverage. Instead, we can explore opportunities that lie in the parallel computing area. Today, even an average desktop computer is configured with multi-core processors. Most of the time, tests are sequential and don't depend on each other, which makes it possible to run them in parallel. This is very simple to achieve and delivers a dramatic increase in efficiency in test execution time. We have seen an order of magnitude speed-up using this approach.

4.1.2. Developing Common Libraries

It is an unfortunate fact that better software engineering practices are not usually followed in the testing domain. With constant release pressure and our hard-to-change tendencies based on waterfall development processes, it is really rare to see testers improving the quality of their own code. While this faulty behavior is generally justified by the amount of time saved for actual testing, ignoring this area almost always introduces adverse effects in the long run.

By recognizing this, we invested on developing common libraries for test framework covering the following areas:

- Test configuration
- Test generation
- Service proxies
- Database-specific operations

Among many, the main benefit of having common libraries for testing is that we are now able to develop test programs in a much productive way since most of the complex operations are encapsulated in these libraries and we can better focus on developing actual tests for critical scenarios.

4.2. Using Heuristic Test Oracles

Using heuristic test oracles for testing provides an effective approach for testing, especially when the correctness criteria for a test cannot be defined precisely [Douglas99]. In the route planning service we have many such scenarios; the following section lists some of these scenarios and the heuristics that we have used extensively in our testing.

4.2.1. Checking Driving Instructions

The route planning service, in general, generates a shortest path and a sequence of instructions to help navigate along this path such as turn instructions. Road topology and other properties are commonly used to control the level of detail of the instructions shown along the path. A simple test to ensure that the instructions are correct requires a complete test oracle, which is essentially a redundant implementation of the SUT. Instead of using such a complete oracle, we can use the following heuristic to help us identify the potential issues:

- Identify the direction of a given instruction (rather than the instruction itself) based on the angle between road segments involved.
- Evaluate sub-paths to identify cases where too many instructions are presented in very short distances.

4.2.2. Checking Algorithm Behavior

For brevity of the discussion, let's assume that we employ a shortest path algorithm that runs on the road graph and generates a shortest path for any given pair of start and end locations. Similar to the discussion in the previous example, designing an oracle that returns the correct answer for each route query is impractical. With the following heuristics, we can easily achieve significant test coverage:

- The length of a shortest path cannot be smaller than the great-circle distance [Wiki_01] between the start and end locations.
- Any sub query along a parent shortest path must return a shortest path that is a subset of the parent. Using this heuristic, we can test a route query iteratively until the start and end locations are the same.
- A shortest path that is more than the great-circle distance by a certain margin can be considered as a potential issue especially in densely populated areas.

4.3. Using Statistical Methods

Statistics is generally used to draw conclusions from data [Montgomery07]. Given that, it is hard to miss the fact that there is a direct relation between testing and the main premise of statistics, especially when we are testing complex applications. Considering the route planning service, there are two factors that make application of statistical methods more pertinent to our problem areas: Using large test suites and analyzing large bodies test results.

As described before, by using large test suites we intend to achieve better coverage. In order to accomplish this, the test suites that we use must represent correlation to effective coverage criteria. For example, if you are measuring performance, your test suite must correlate to the actual operational profile of you product. Similarly, if you are aiming at geographical coverage with your tests then your test suite must correlate to the actual coverage area of your service. The statistical methods become important when the coverage area is large (such as the entire set of user requests or entire service coverage area). Statistical methods, such as sampling, can be used effectively in these scenarios for test input generation especially in larger scales.

Besides test generation, the task of analyzing test results also represents opportunities for statistical methods. For example, confidence intervals can be used to assess significant deviations in data, which are likely to point to issues. Think about a scenario where you test a new routing data release. Out of 5000 routes that are calculated, 20% shows longer overall trip time. Looking at the results more closely reveals that only 0.1% of these results (actually five routes) have a difference of more than 10%. With careful use of statistics we can support our findings and eliminate noise in the results.

4.4. Developing Visualizations Tools

The previous chapters aimed at highlighting the difficulties involved in working with geospatial entities and presented some ideas for automated testing. Interestingly, visualization tools can be considered on the

other end of the spectrum emphasizing careful evaluation by skilled testers. We believe that these two approaches are not contradicting, but rather complementing each other. During testing, we work with and generate large amounts of data in the forms of test scenarios, test inputs and test results. In its raw format, this data is represented as geospatial entities with complex properties, which makes understanding and analyzing more of a tedious, difficult and error-prone task. For example, when the route planning algorithm generates a route result that does not look correct, it may be due to many different factors:

- Road data may have some problems such as invalid turn restrictions and incorrect speed,
- Algorithm may have some problems such as logical errors and incorrect optimizations,
- Data and algorithm may be correct but the scenario may represent an edge case, or
- Test configuration may be incorrect; hence, the result may be a false-positive.

As a tester, as much as it is challenging to be able to ask these questions while investigating the issue, it can be more challenging to get the answers if we don't have the proper tools available. Visualization tools help us comprehend the complex data and scenarios by presenting a rich set of relevant data in an intuitive way and improve productivity immeasurably.

To illustrate let's look at a simple case where we compare route results that are calculated based on two different routing data releases. As mentioned before, we apply some statistical methods and identify that a handful of results are actually of any significance. Using the available visualization tools, the results would be presented separately as it is shown in Figure 2 (a) and (b). With this approach, it is difficult to pinpoint how the two results differ. However, we can programmatically analyze these results and identify their differences, which then can be used by a visualization tool to provide a more intuitive depiction. Figure 2 (c) illustrates this approach.

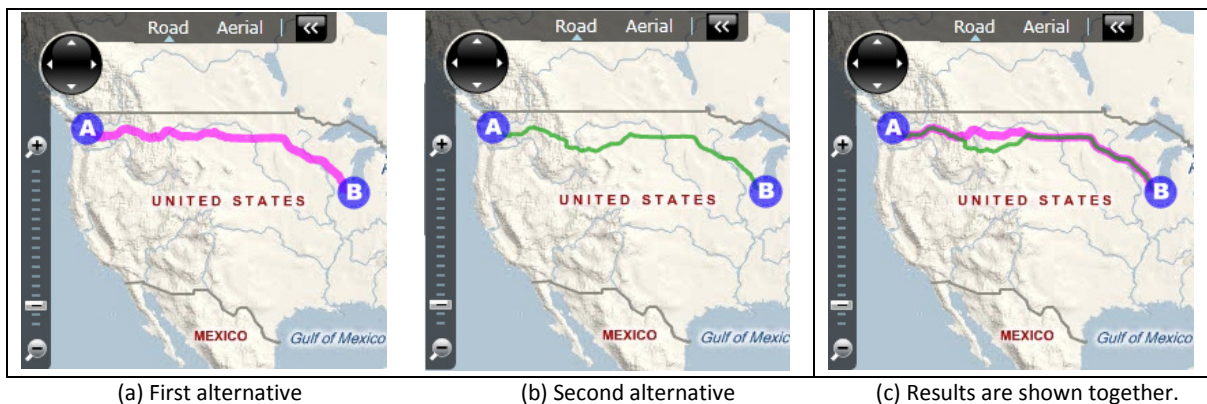


Figure 2 - Intuitive visualization helps depict the issue more clearly.

5. Conclusions

Testing complex applications can be a challenging task for test generation, execution and evaluation. Given the unique challenges imposed by such applications, testing may become more difficult than development at certain times. Understanding the application domain and the common problems can present opportunities to develop effective solutions for testing to boost efficiency and productivity. The benefits of such solutions can be dramatic:

- Given the constant time pressure and the need for delivering novel and compelling features, test feedback plays a vital role in developing high-quality software; any improvement in test feedback quality and time is quite valuable for the entire team.
- Complex data introduces challenges to grasp the subtleties of the application domain; investing in new and original approaches such as heuristic test oracles can extend our capabilities to both understand and address the issues that may be otherwise hard to find.
- Rich visualization tools provide for increased productivity in assessing the issues at hand; smart investment in this area has a great potential for return on investment.

In testing, we are never short of new and original problems; as we gain more experience and maturity in our expertise, we can develop more effective solutions by focusing on the emerging patterns in those new problem domains.

References

[**Cormen09**] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms*. The MIT Press.

[**Delling09**] Delling, D., Sanders, P., Schultes, D., and Wagner, D. (2009). *Engineering route planning Algorithms*. In *Robust and Online Large-Scale Optimization* (pp. 182-206). Springer Berlin / Heidelberg.

[**Hamlet88**] Hamlet, D., and Taylor, R. (1990). *Partition testing does not inspire confidence*. *IEEE Transactions on Software Engineering*, 1402-1411.

[**Douglas99**] Hoffman, D. (1999, March/April). *Heuristic Test Oracles*. *Software Testing and Quality Engineering Magazine*, 1 (2).

[**Montgomery07**] Montgomery, D. C., & Runger, G. C. (2007). *Applied Statistics and Probability for Engineers* (4 ed.). Wiley.

[**Wiki_01**] *Great-circle Distance*. Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Great-circle_distance.