Introduction to the Hexawise test design tool

February, 2011
Introduction
- Why Hexawise?
- Where and when can Hexawise be applied?

Benefits Explained
- Faster test case creation
- More coverage per test
- More thorough test plans find more bugs
- Additional benefits
- Evidence-based ROI calculations

Additional Information
- Contact information for free training and support
- Case studies and links to more information
Hexawise is a software test design tool that helps teams “test smarter, not harder.” Use Hexawise to:

- Create software tests much faster
- Design more efficient and effective tests
- Find more defects (and find them earlier)
- Hexawise complements other popular testing tools like QC and QTP; it does not seek to displace them

“Hexawise will have a major impact on the testing industry.”

- Capers Jones
  (one of the world's leading authorities on software quality and author of 15 books)
Why Hexawise? Ease of Use

Hexawise is an easy to use tool with three straightforward screens.

1. Define Inputs
   Test designers, using information from requirements documents and conversations with stakeholders, identify test inputs and manually enter them into Hexawise. Once test inputs are entered, users simply click on the “Create Tests” button.

2. Create Tests
   Within seconds, Hexawise generates the test conditions that should be executed to achieve the desired coverage in the smallest possible number of tests. Users can adjust thoroughness settings to quickly adjust the coverage strength of plans. Settings range from reasonably thorough 2-way solutions (which typically have a few dozen tests) to extremely thorough 6-way solutions (which could have a few thousand tests).

3. Analyze Coverage
   Once Hexawise creates a solution for a user-identified level of thoroughness, the test designer can examine how well the user-specified thoroughness goal is achieved at each point throughout the course of the test plan.

“Hexawise is the most user friendly test design tool I’ve found.”
- Charelle Lacasse

“A really cool test planning tool.”
- Meghan Sumrell

“Allpairs on steroids.”
- Roland Stens
Hexawise can be used to select and document tests in virtually every type of software testing project in every phase of testing.

- In every type of application (web-based, desktop, mainframe, etc.),
- In every type of industry,
- In every phase of software testing, (from Unit Testing to User Acceptance Testing), and
- In every type of software testing (functional testing, load testing, performance testing, security testing, etc.)*

“We’ve achieved consistently excellent results using Hexawise to design all kinds of tests from Unit Tests to User Acceptance Tests.

Generating tests with this approach takes far less time than it used to take us to select and document them by hand. In addition, the tests generated by Hexawise are consistently better at finding bugs than tests generated by hand.

I was skeptical at first but after seeing the benefits myself in many kinds and sizes of projects, I’m firmly convinced. I’ll never go back to manual test case selection and documentation methods.”

- PH
Large, European-based Global Bank

* Hexawise is most often used to design functional test cases.
Benefits: Faster Test Creation

Testers using Hexawise create tests much faster than it takes them to select and document test cases by hand.

Manual test case selection:
1. Document each test script by hand
2. Repeat step 1 dozens or hundreds of times until all tests are completely documented

Hexawise
1. Manually enter test inputs into Hexawise
2. Click the “Create Tests” button in Hexawise to generate all your tests at once within seconds

Average time required to select and document test cases

Selecting and writing test cases by hand

30-40% less using Hexawise

“We recently did a pilot project using Hexawise. We found that we generated better tests with Hexawise in one quarter of the time that it took us to select and document tests manually.”

-RS
CRM Test Lead

Source: Conservatively interpreted data from several dozen recent pilot projects. Time savings are often significantly larger than 40% and will almost always exceed 30%.
Benefits: Better Coverage

Not only are Hexawise-generated tests faster to create, they’re objectively much better. They achieve higher coverage in fewer tests.

The coverage charts below illustrate Hexawise’s superior coverage. The first 13 tests of a client’s actual test plan (on the left) have large gaps in coverage and a great deal of inefficient repetition of test steps from test to test.

In contrast, every one of the 6 test steps in every one of the 13 Hexawise-generated tests is focused on achieving the maximum amount of additional coverage by seeking new combinations of input values that have not been tested for yet. This kind of combinatorial optimization is impossible for the human brain to perform effectively without the aid of a tool like Hexawise.

13 Actual Client Tests 13 Hexawise Tests

“Hexawise consultants have been extremely responsive and fun to work with. They provided us with extensive customized training and helped us achieve success on our engagement by providing support on pilot projects, and reviewing test plans created by new users.”

- PM
Large Global Bank
The superior coverage achieved by Hexawise-generated tests consistently result in more defects being found during testing.

"My experience with statistical testing is that this test design approach consistently generates 2X to 5X as many defects found per tester hour when compared to manual test case selection."

- RP
Large Global IT Consultancy
Test Lead in CIO Organization

"We conducted a pilot project using Hexawise. We found 44% more defects using our Hexawise-generated tests than we found using our manually-selected tests."

- CB
Large Global IT Consultancy

Using Hexawise has several additional benefits.

<table>
<thead>
<tr>
<th>Benefit</th>
<th>Details</th>
</tr>
</thead>
</table>
| **Using test models vs. a sets of individual test scripts** | • Manual processes result in a long list of individual test scripts  
• Hexawise test design processes automatically create test models  
• It is easier to create tests from models because tests are automatically generated with the simple push of Hexawise’s “Create Tests” button  
• It is easier to review models because inputs appear in an executive summary and coverage strength choices explicitly describe what will be tested  
• It is easier to modify models because changes only need to be made once vs. changing all impacted tests  
• It is easier to reuse models because they can be picked up by someone new to a project months or years later and be easily understood |
| **Adjustable coverage** | • You can recalculate an entirely new set of tests within minutes to adjust coverage in response to:  
  • Risk-based testing considerations  
  • Last-minute changes to testing schedules  
  • Stakeholder discussions about clearly defined tradeoffs between an approach consisting of higher coverage w/ more tests strategy vs. an alternative approach consisting of lower coverage w/ fewer tests |
| **Template test plans and testing checklists** | • Template test plans (e.g., test plans for insurance ratings engines, search features, web-based applications, etc.) are built into Hexawise to facilitate rapid test plan creation; more template plans are coming  
• Testing checklists help testers remember to test for things that have caused defects in production on prior projects |

“Maintaining and re-using test artifacts is a breeze with Hexawise.

When I get last-minute requirements changes, I am able to add them into the tool in minutes and almost instantly determine how many additional test should be executed.

We get faster, better quality feedback from stakeholders now too. The review materials we export from Hexawise clearly highlight the important aspects of the test plan to be reviewed.”

- PH

Large, European-based Global Bank
Sign up for a trial account now:

The first 5 users in each organization are allowed to use Hexawise for free. Sign up at: http://hexawise.com/signup
For more information, see:

A fun introduction to combinatorial testing (online presentation)

An introductory presentation to Hexawise with screen shots and examples of test case generation

An IEEE Computer article, Combinatorial Software Testing with an empirical study showing combinatorial test design methods more than doubled tester productivity as compared to manual test case selection methods

“Hexawise is a great tool. I recommend Hexawise to anyone seeking to improve the efficiency and effectiveness of software testing.”

- Lester Bostic, BlueCross/BlueShield
### Hexawise Demo: Define Inputs

#### Mortgage Application Example

<table>
<thead>
<tr>
<th>Region (7)</th>
<th>NY</th>
<th>NJ</th>
<th>FLA</th>
<th>TX</th>
<th>CA</th>
<th>DC</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tier (6)</td>
<td>L</td>
<td>M</td>
<td>H</td>
<td>H+1</td>
<td>H+2</td>
<td>H+3</td>
<td></td>
</tr>
<tr>
<td>Property (6)</td>
<td>1 Family</td>
<td>2 Family</td>
<td>3 Family</td>
<td>4 Family</td>
<td>Coop</td>
<td>Condo</td>
<td></td>
</tr>
<tr>
<td>Credit (5)</td>
<td>A+</td>
<td>A</td>
<td>A-</td>
<td>B</td>
<td>&lt;B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Res (3)</td>
<td>Pri</td>
<td>Vac</td>
<td>Inv</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LTV (3)</td>
<td>80%</td>
<td>90%</td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NIV (2)</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAV (2)</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refinance (2)</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Closing Cost (2)</td>
<td>Customer</td>
<td>Bank</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intro Rate (2)</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank Emp (2)</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example used by Bernie Berger at STAREast 2003 and later by Paul Jorgensen
12 parameters with varying levels of values results in

$7 \times 6 \times 6 \times 5 \times 3 \times 3 \times 2 \times 2 \times 2 \times 2 = 725,650$ possible tests
Hexawise identifies the most important tests to execute first based on these test inputs. It only takes only 48 tests cover all 2-way interactions. In other words, in this “pair-wise” set, every pair of test inputs will be tested together in at least one of the tests. Variation is maximized. Repetition is minimized.

<table>
<thead>
<tr>
<th>Region</th>
<th>Tier</th>
<th>Property</th>
<th>Credit</th>
<th>Res</th>
<th>LTV</th>
<th>NIV</th>
<th>NAV</th>
<th>Refi</th>
<th>Closing Cost</th>
<th>Intro Rate</th>
<th>Bank Emp</th>
</tr>
</thead>
<tbody>
<tr>
<td>NY</td>
<td>L</td>
<td>1 Family</td>
<td>A+</td>
<td>Pri</td>
<td>80%</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Customer</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NY</td>
<td>M</td>
<td>2 Family</td>
<td>A</td>
<td>Vac</td>
<td>90%</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Bank</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NY</td>
<td>H</td>
<td>3 Family</td>
<td>A-</td>
<td>Inv</td>
<td>100%</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Bank</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>NY</td>
<td>H+3</td>
<td>2 Family</td>
<td>A+</td>
<td>Vac</td>
<td>90%</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Customer</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>NY</td>
<td>H+1</td>
<td>Coop</td>
<td>A</td>
<td>Vac</td>
<td>80%</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Customer</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NY</td>
<td>H</td>
<td>Condo</td>
<td>A</td>
<td>Pri</td>
<td>100%</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Customer</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
80% coverage of all the possible pairs of input values is reached after just 22 tests. This information can be used to answer “how much testing is enough?”
Why focus on testing all pairs of test input values? Because multiple thorough studies show approximately 85% of defects in production could have been detected by simply testing all possible pairs of values.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tiny Percentage</td>
<td>We can identify the “50 best tests” (48 out of 726,000)</td>
</tr>
<tr>
<td>Purportedly Effective</td>
<td>This small set of tests will trigger most of the bugs</td>
</tr>
<tr>
<td>Implication</td>
<td>This would far exceed Pareto’s 80/20 rule-of-thumb; This would be &gt;80% of the benefit in &lt;1 % of the effort!</td>
</tr>
</tbody>
</table>
“Time for a Reality Check”

Sounds nice in theory, but does it work consistently in practice?

“In God we trust...

All others must bring data.”

W. Edwards Deming
These same methods have been successfully applied for more than four decades in other industries.

William G. Hunter (my dad)
- Founder of ASQ Statistics Division
- Design of Experiments* expert
- Loved experiments, “getting the data” and improving processes

George E. P. Box
- “Justin, you’ll see three phases:
- First: It won’t work.
- Second: It won’t work here.
- Third: Of course this stuff works; it’s just common sense.”

Design of Experiments a specialized field of applied statistics. Design of Experiments methods are designed to answer the pragmatic question: “How, with a minimum number of experiments [read ‘test cases’ when you're applying these methods to software testing], can we find out as much actionable information as possible?”
Average results across 10 published pilot projects in software testing when Hexawise is used:

**Test design time** (required to select and document tests)
36% less

**Test execution efficiency** (bugs found / tester hour)
2.4X as many

**Testing thoroughness** (defects found overall)
13% more

by Richard Kuhn, Raghu Kacker, Jeff Lei, and Justin Hunter.
Let’s assume we would like to create test cases to test a health insurance ratings engine with some simple inputs:

**Primary Applicant Info:**
- Age
- Location (by zip)
- Gender

**Spouse?** (If yes → age)
**Children?** (If yes → Ages)

**Include dental?**
**Include maternity?**
Appendix Slides

• **Demo / Health Insurance Example**
  (with more detailed explanation of adjustable coverage strength options)

• **Why Does This Approach Work So Well?**
  (How, specifically, are Design of Experiments-based test cases superior to hand-selected test cases?)
In the first screen of Hexawise, the “Define Inputs” screen, a test designer would input the following test inputs:

<table>
<thead>
<tr>
<th>New Parameter</th>
<th>Values (each value on a new line)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age of Primary Applicant (4)</td>
<td>0.1 - 17.9 18 - 39.9 40 - 64.9 65 - 140</td>
</tr>
<tr>
<td>Zip Code (3)</td>
<td>Urban Rural Out of State</td>
</tr>
<tr>
<td>Gender (3)</td>
<td>Male Female Blank</td>
</tr>
<tr>
<td>County (2)</td>
<td>Valid Invalid, the...</td>
</tr>
<tr>
<td>Age of Spouse (4)</td>
<td>N/A 18 - 39.9 40 - 64.9 65 - 140</td>
</tr>
<tr>
<td>Add Child? (2)</td>
<td>Click on Add... Do Not Click...</td>
</tr>
<tr>
<td>How Many Children (4)</td>
<td>0 1-3 4-6 7-12</td>
</tr>
<tr>
<td>Age of One Half of the Children (4)</td>
<td>N/A 0.1 - 17.9 18 - 39.9 40 - 80</td>
</tr>
<tr>
<td>Age of Other Half of the Children (4)</td>
<td>N/A 0.1 - 17.95 18 - 39.9 40 - 81</td>
</tr>
<tr>
<td>Dental (2)</td>
<td>Do Select Do Not Select</td>
</tr>
<tr>
<td>Maternity? (2)</td>
<td>Do Select Do Not Select</td>
</tr>
</tbody>
</table>
Can’t Test All Possibilities

These simple inputs on one screen will create almost 295,000 possible software tests. This is far more tests than testers can realistically execute.

<table>
<thead>
<tr>
<th></th>
<th>Possible Combinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age of Primary Applicant (4)</td>
<td></td>
</tr>
<tr>
<td>Zip Code (3)</td>
<td></td>
</tr>
<tr>
<td>Gender (3)</td>
<td></td>
</tr>
<tr>
<td>County (2)</td>
<td></td>
</tr>
<tr>
<td>Add Spouse? (2)</td>
<td></td>
</tr>
<tr>
<td>Age of Spouse (4)</td>
<td></td>
</tr>
<tr>
<td>Add Child? (2)</td>
<td></td>
</tr>
<tr>
<td>How Many Children (4)</td>
<td></td>
</tr>
<tr>
<td>Age of One Half of the Children (4)</td>
<td></td>
</tr>
<tr>
<td>Age of Other Half of the Children (4)</td>
<td></td>
</tr>
<tr>
<td>Dental (2)</td>
<td></td>
</tr>
<tr>
<td>Maternity? (2)</td>
<td></td>
</tr>
</tbody>
</table>

\[4 \times 3 \times 3 \times 2 \times 2 \times 2 \times 2 \times 4 \times 2 \times 4 \times 2 \times 2 = 294,912 \text{ possible software tests}\]
Clicking on Hexawise’s “Create Tests” button solves the problem of “what tests should we execute”?

Hexawise calculates the most powerful set of tests that are required to meet the tester’s coverage objectives. For a coverage objective of “create a set of tests such that every conceivable pair of test inputs is tested together in at least one test case,” only 28 tests are required. Hexawise calculates the test conditions for these 28 test cases in less than a second.

If a tester wants a larger, more comprehensive, set of tests with higher coverage strengths, Hexawise can also quickly generate such test set solutions. This capability is shown on the next slide.
Why do we start with testing all possible combinations of every pair of test inputs?

Because it has been proven to be an extremely efficient and effective way to test. This is because even though only a tiny percentage of the total possible tests are required to test for every single 2-way (or pairwise combination), testing all pairs of test conditions together in at least one test case will uncover the vast majority of software defects.

As demonstrated on the chart on the right, while it is common for defects to be caused by the combination of two test inputs, it is quite unusual that a defect can only be triggered as a result of three or more specific test inputs. Studies have shown ~85% of all defects that have made their way past software testing teams and into production could have been detected with simple 2-way tests.

*Combinatorial Software Testing* - IEEE Computer, 2009, Dr. Rick Kuhn, Dr. Raghu Kacker, Dr. Jeff Lei, and Justin Hunter.
Testing for every pair of input values is just a start. The test designer can generate plans with very different levels of testing thoroughness.

The 2-way test cases Hexawise generates have been consistently shown to be more thorough than standard test cases created by most test teams at Fortune 500 firms. Even so, Hexawise allows users to “turn up the coverage dial” dramatically and generate other, extraordinarily thorough, sets of tests. In this case, we see Hexawise can generate test set solutions for this simple insurance ratings engine example ranging in size from 28 test cases (for users who prioritize speed to market) all the way up to 3,925 test cases (for users who desire extremely thorough testing).
The “Analyze Coverage” screen shows you how much coverage is achieved at each point in the set of tests. In other words, what percentage of the targeted combinations have been tested for after each test?

This chart gives teams the ability to **make fact-based decisions** about “how much testing is enough?” Here, for example, 83% of the pairs of test inputs entered into this plan have been tested together after only 12 tests (out of 295,000 possible tests).
Better Than Hand-Selected Tests

If you take a close look at any set of Hexawise-generated test cases you will notice that there is an enormous amount of variation from test case to test case (and the smallest amount of repetition that is mathematically possible to achieve).

In contrast, if you were to translate your existing manually-selected test cases into a similar format and analyze them, you would find that the manually-selected test cases have far more repeated test combinations and far less variation from test case to test case. This is a big part of the reason why Hexawise generates dramatic efficiency improvements.

In addition, if you were to graph the percent of the targeted 2-way combinations achieved by your existing manually-selected test cases, you would find that there are many pairs of test inputs that were never covered by your tests. The fact that Hexawise will ensure every pair of test inputs gets tested in at least one test case is a big part of the reason why Hexawise-generated tests result in superior coverage and more defects found during test execution.
Appendix Slides

• Demo / Health Insurance Example
  (with more detailed explanation of adjustable coverage strength options)

• Why Does This Approach Work So Well?
  (How, specifically, are Design of Experiments-based test cases superior to hand-selected test cases?)
What is DoE-based testing?

<table>
<thead>
<tr>
<th>Topic</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Definition</strong></td>
<td>Design of Experiments-based testing is a test design approach used to identify a small subset of tests (from many possible ones) in order to find as many defects as possible in as few tests as possible.</td>
</tr>
</tbody>
</table>
| **Why it Works** | Test conditions are constructed to ensure:  
  - No combinations of conditions get accidentally omitted  
  - Unproductive repetition is minimized |
| **“AKA”**   |  
  - Pairwise / AllPairs  
  - Orthogonal Array / OA / OATs  
  - 2-way, 3-way, ... t-way |
Software Testing Challenges

• Software applications are very complex; it is impossible to test every possibility

• Extraordinarily smart, pragmatically-oriented applied statisticians created the field of “Design of Experiments” to solve exactly this challenge; for the last 40+ years they have developed highly effective math-based covering array techniques and similar strategies which are now broadly used in many areas including manufacturing, advertising, and agriculture

• These proven Design of Experiments techniques, which are designed to find out as much information as possible in as few test cases as possible, also have direct applicability to the software testing field

• Unfortunately, the vast majority of software testers in the relatively young field of software testing have never heard of any Design of Experiments concepts like MFAT vs. OFAT, Orthogonal Array coverage, pairwise coverage, or even the existence of the “Design of Experiments” field

• Instead of using 40+ years of Design of Experiments-based knowledge to design tests that are as effective as possible, testers almost always manually select the combinations of test conditions they use in their tests, and as a result...
... the results from manual test case selection efforts are consistently far from optimal:

Missed combinations

Wasteful repetition
In contrast, Hexawise algorithms use Design of Experiments-based methods to generate tests. The result is that Hexawise-generated tests consistently find more defects in fewer tests. Hexawise-generated tests pack more coverage into each test.
Please feel free to contact us directly if you have any questions. I would be happy to help you conduct a proof of concept pilot project designed to measure the benefits delivered by this test design approach.

Justin Hunter
Founder and CEO
justin.hunter@hexawise.com