Evolution of Advanced Combinatorial Testing for Software and systems (ACTS) from Design of Experiments (DoE)

Raguh Kacker
National Institute of Standards and Technology
Gaithersburg, MD

Carnegie-Mellon University, 7 June 2011
Outline

• Brief review of Design of Experiments (DoE) methods and early history of Combinatorial Testing (CT)
• Review evolution of tools for generating test suites
• Discuss special aspects of CT for software and systems
• Orthogonal Arrays (OAs) and Covering Arrays (CAs)
  – Limitations of OAs, benefits of CAs for software testing
• Mathematicians behind DoE/OAs/CAs
• Some comments on CT for software and systems
• List some applications areas for combinatorial testing
Combinatorial testing is a variation of Design of Experiments (DoE) adapted for testing software

- DoE began in agricultural in 1920s, then animal science, medicine, chemical industry, manufacturing, electronics, computers & communication hardware-software
- Modern applications of DoE type methods include
  - Biotechnology (genetic analyses)
  - Combinatorial drug discovery methods
  - Combinatorial high throughput materials development
  - Combinatorial testing for software and systems
- Present new challenges, offer new opportunities, require different adaptations of classical DoE approach
Classical DoE methodology and objectives

• Methodology to change values of a number of test factors, measure corresponding change in response to obtain useful information about a cause-effect system
  – DoE useful for study of systems subject to combinatorial effects, measurement error and random variation
  – Information obtained with minimum expense of time and cost
  – Term DoE includes associated data analysis

• Objectives in classical DoE:
  – Compare treatments
  – Identify important factors
  – Identify optimum combinations of test settings
  – Determine parameters at which variability is minimum
Classical DoE factors, plans, analysis

• In addition to test factors, concomitant factors include: uncontrolled factors, background factors (controlled in experiment, not in use conditions)
  – Various techniques such as replication, randomization, blocking (homogeneous grouping) used to deal with such factors
  – Not important in CT for software and systems

• Classical DoE plans
  – Randomized block designs, Balanced incomplete block designs, Factorial and fractional factorial designs, Latin squares, Orthogonal-Latin squares, Orthogonal arrays

• Basic statistical analyses associated with DoE include
  – Main effect: average effect over all values of other factors
  – 2-way interaction effect: how effect changes with value of another
  – ANOVA to determine significant main effects interaction effects

• Estimate parameters of linear models for prediction
Example of DoE experiment plan

- Five test Factors: four with 2 values and one with four
  1. Viscosity \(a\) with 2 values \{0, 1\}
  2. Feed rate \(b\) with 2 values \{0, 1\}
  3. Spin Speed \(c\) with 2 values \{0, 1\}
  4. Pressure \(d\) with 2 values \{0, 1\}
  5. Materials \(e\) with 4 types \{0, 1, 2, 3\}

- Combinatorial test structure \(2^4 \times 4^1\)
  - Total number of test combinations: \(2^4 \times 4^1 = 64\)

- Object: evaluate main effects only (no interaction effects)

- Possible to evaluate main effects using only 8 test cases
  - Use orthogonal array OA(8, \(2^4 \times 4^1\), 2) to set experiment plan
Orthogonal array: OA(8, 2⁴×4¹, 2)

• Strength 2: every two columns contains all possible pairs of combinations an equal number of times

  a  b  c  d  e  data
  1. 0  0  0  0  0  y₁
  2. 1  1  1  1  0  y₂
  3. 0  0  1  1  1  y₃
  4. 1  1  0  0  1  y₄
  5. 0  1  0  1  2  y₅
  6. 1  0  1  0  2  y₆
  7. 0  1  1  0  3  y₇
  8. 1  0  0  1  3  y₈

  • Associate 5 factors with columns, values {0, 1}, {0, 1, 2, 3} with entries
  • Rows of OA specify 8 test cases
  • Every test value paired with each test value of every other factor
  • Main effect of factor a: \( \frac{(y₂+y₄+y₆+y₈)}{4} - \frac{(y₁+y₃+y₅+y₇)}{4} \)
  • Other factor values averaged over
  • Need more than 8 test cases to evaluate 2-way interaction effects
DoE plans are balanced

- DoE plans can be expressed in matrix form
  - Columns: test factors
  - Entries: test values
  - Rows: tests cases

- In DoE “main effects” and “interaction effects” are linear combinations (called contrasts) of response data
  - Average of N/2 data minus average of N/2 other data

- DoE plans must be balanced for main effects and interaction effects to be meaningful
  - Each value of other factors must be included in both averages

- In combinatorial testing “interaction” means “joint combinatorial effect of two or more factors”
Early history of combinatorial testing for software and systems

- Mandl (1985) “Use of orthogonal Latin squares for testing Ada compiler” often cited first publication
- USA/late-1980s descendent orgs of AT&T (Bell Labs, Bellcore-Telcordia) exploring use of OAs for combinatorial testing; developing tools based on OAs: Brownlie et al (1992), Burroughs et al (1994)
- In 1990s use of OAs for testing of computer and communication hardware-software systems expanded
Evolution of tools for generating combinatorial test suites

• Early tools for generating test suites for pairwise testing
  – OATS (Phadke AT&T) 1990s (not public)
  – CATS (Sherwood AT&T) 1990s (not public)
  – AETG (Cohen et al Telcordia) 1997 (commercial)
  – IPO (Yu Lei NCSU) 1998 (not public)
• www.pairwise.org (Czerwonka, Microsoft) lists 34 tools
  - Tconfig
  - TestCover
  - AllPairs[McDowell]
  - IPO-s
  - CTS
  - DDA
  - PICT
  - Jenny
  - AllPairs
  - EXACT
• Primary algorithm in NIST-UTA tool ACTS is IPOG
  – Generalization of 1998 IPO (Yu Lei UTA NIST)
  – Freely distributed
Investigation of actual faults

• Kuhn et al (2001, 2002, 2004) investigated actual faults in a variety of software and systems to determine what kind of testing would have detected them
  – 15 years medical devices recall data from FDA, Browser, Server, NASA distributed database, Network security system
  – 2-way testing could detect 65 % to 97 % faults
  – 3-way testing could detect 89 % to 99 % faults
  – 4-way testing could detect 96 % to 100 % faults
  – 5-way testing could detect 96 % to 100 % faults
  – 6-way testing could detect 100 % faults in all cases investigated

• Kera Bell (2006, NCSU) arrived similar conclusion

• Empirical conclusion: 2-way testing useful, may not be inadequate; however 6-way testing may be adequate
Combinatorial high strength ($t$-way) testing

- Dynamic verification of input-output system
  - against its known expected behavior
  - on test suite of test cases selected such that
  - all $t$-way combinations are exercised with the
  - object of discovering faults in system

- Earlier combinatorial test suites based on orthogonal arrays of strength 2 useful for pairwise (2-way) testing

- Now tools available for high strength $t$-way testing
  - ACTS (NIST/UTA) 2009
  - IPOG (Yu Lei UTA) optimized for $t$ from 2 to 6
    - Built-in constraints support
    - Freely downloaded by over 750 organizations and individuals
Special aspects of CT for software and systems-1

- System Under Test (SUT) must be exercised (dynamic verification)
- CT does not require access to source code
- Expected behavior (oracle) for each test case be known – determined from functionality and/or other information
- Final result for each test case: passing or failing
- Objective of CT to reveal faults; a failure indicates fault
- Depending on fault required strength $t$ can be from 2 to 6
- Each $t$-way combination must be exercised to reveal
- No need to run a $t$-way combination more than once
Special aspects of CT for software and systems-2

- Numbers of test values of factors may be different
- A test case is combination of one value for each factor
- Certain test cases invalid, incorporate constraints
- From pass/fail data identify $t$-way combinations which trigger failure among actual test cases (fault localization)
- No statistical model used in data analysis: test plan need not be balanced like classical DoE
- Choice of factors and test values highly critical for effectiveness of combinatorial testing
  - Information about nature of faults to be detected helpful
Orthogonal arrays

- Fixed-value OA($N, k, v, t$): $N \times k$ matrix such that every $t$-columns contain all $t$-tuples the same number of times
  - Strength: $t$
  - Index: $\lambda = N/v^t$
- Mixed-value orthogonal array OA($N, v_1^{k_1}v_2^{k_2}...v_n^{k_n}, t$)
  - Rows: $N$
  - Columns $k = k_1 + k_2 + ... + k_n$
  - Entries: $k_1$ columns have $v_1$ values...$k_n$ columns have $v_n$ values
  - Every $t$-columns contain all $t$-tuples the same number of times
  - Index different for different columns
  - In this notation OA($N, k, v, t$) $\equiv$ OA($N, v^k, t$)
Combinatorial test structure $2^4 \times 3^1$ Strength $t = 2$

OA for $2^4 \times 3^1$ dose not exist

<table>
<thead>
<tr>
<th>OA($8, 2^43^1, 2$)</th>
<th>CA($8, 2^43^1, 2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a\ b\ c\ d\ e$</td>
<td>$a\ b\ c\ d\ e$</td>
</tr>
<tr>
<td>1. 0 0 0 0 0</td>
<td>1. 0 0 0 0 0</td>
</tr>
<tr>
<td>2. 1 1 1 1 0</td>
<td>2. 1 1 1 1 0</td>
</tr>
<tr>
<td>3. 0 0 1 1 1</td>
<td>3. 0 0 1 1 1</td>
</tr>
<tr>
<td>4. 1 1 0 0 1</td>
<td>4. 1 1 0 0 1</td>
</tr>
<tr>
<td>5. 0 1 0 1 2</td>
<td>5. 0 1 0 1 2</td>
</tr>
<tr>
<td>6. 1 0 1 0 2</td>
<td>6. 1 0 1 0 2</td>
</tr>
<tr>
<td>7. 0 1 1 0 3 2</td>
<td></td>
</tr>
<tr>
<td>8. 1 0 0 1 3 2</td>
<td></td>
</tr>
</tbody>
</table>
Covering arrays

- Fixed-value CA($N$, $k$, $v$, $t$): $N \times k$ matrix such that every $t$-columns contain all $t$-tuples at least once
  - Strength: $t$
  - OA($N$, $k$, $v$, $t$) of index $\lambda = 1$ is covering array with min test cases, however OA of index 1 are rare
  - Most CA are unbalanced

- Mixed-value covering array CA($N$, $v_1^{k_1} v_2^{k_2} \ldots v_n^{k_n}$, $t$)
  - Rows: $N$
  - Columns $k = k_1 + k_2 + \ldots + k_n$
  - Entries: $k_1$ columns have $v_1$ values…$k_n$ columns have $v_n$ values
  - Every $t$-columns contain all $t$-tuples at least once
  - In this notation CA($N$, $k$, $v$, $t$) $\equiv$ CA($N$, $v^k$, $t$)
Combinatorial test structure $2^4 \times 3^1$ Strength $t = 2$
OA for $2^4 \times 3^1$ dose not exist

OA(8, $2^44^1$, 2)  CA(8, $2^43^1$, 2)

\[
\begin{array}{ccccc}
\text{a} & \text{b} & \text{c} & \text{d} & \text{e} \\
1. & 0 & 0 & 0 & 0 \\
2. & 1 & 1 & 1 & 1 & 0 \\
3. & 0 & 0 & 1 & 1 & 1 \\
4. & 1 & 1 & 0 & 0 & 1 \\
5. & 0 & 1 & 0 & 1 & 2 \\
6. & 1 & 0 & 1 & 0 & 2 \\
7. & . & 0 & 1 & 1 & 0 & 3 & 2 \\
8. & . & 1 & 0 & 0 & 1 & 3 & 2 \\
\end{array}
\]
Limitations of test suites based on OAs

- OAs do not exist for many combinatorial test structures
  - Construction requires advanced mathematics
- Catalog of OAs
  http://www2.research.att.com/~njas/oadir/
- Most OAs of strength $t = 2$; Some $t = 3$ recent
- Most fixed-value; Some mixed value OAs recent
- Combinatorial test structure fitted to suitable OA
  - Need $2^4 \times 3^1$ use OA$(8, 2^4 \times 4^1, 2)$ make 4-values out of 3
- Constraints destroy balance property of OA
Benefits of Covering arrays

- CAs available for any combinatorial test structure
  - Constructed by computational algorithms and mathematical methods (e.g. IPOG, IPOG-D in ACTS)
- CAs available for any required strength (t-way) testing
- For a combinatorial test structure if OA exists then CA of same or fewer test runs can be obtained
- For large numbers of factors, CAs of few test runs exist
- Generally CAs not balanced (like OAs), not needed in software testing
- Certain tests invalid, constraints can be incorporated
  - Coverage defined relative to valid test cases
Mathematicians behind DoE/OA/CAs

- 1832 Évariste Galois (French, shot in duel at age 20)
- 1938 R C Bose (father of math underlying DoE)
- 1947 C R Rao (concept of orthogonal arrays)
  - Hadamard (1893), RC Bose (1938), KA Bush, S Addelman, G Taguchi, JN Srivastava, ...
- Catalog of OAs http://www2.research.att.com/~njas/oadir/
- 1993 N J A Sloan (definition of covering arrays)
  - Renyi (1971), Katona (1973), Kleitman and Spencer (1973), ..., Roux (1987, French, disappeared after PhD), ..., Alan Hartman
- Connection between needs in software testing and CAs
- Sizes of CAs (Charlie Colbourn ASU)
Some comments on CT for software and systems

• CT one of many complementary testing methods
• CT can reveal faults, not guarantee their absence (software testing is about risk management)
• CT can reveal many types of faults
• CT can be used in many stages of software development
• CT better than random (fewer test runs); may be better than human generated test suites (better coverage)
• CT does not require access to source code; expected behavior (oracle) for test cases needs to be determined
  – From functionality and/or other information
List some applications areas for combinatorial testing

• **Software testing**
  – Test input space, test configuration space

• **Computer/network security**
  – Network deadlock detection, buffer overflow

• **Testing Access Control Policy Systems**
  – Security, privacy (e.g. health records)

• **Explore search space for study of gene regulations**
  – [http://www.plantphysiol.org/content/127/4/1590.full](http://www.plantphysiol.org/content/127/4/1590.full)

• **Optimization of simulation models of manufacturing**
Summary

• Combinatorial testing is a variation of DoE adapted for testing software and hardware-software systems
• Early use of combinatorial testing was limited to pairwise (2-way) testing
• Investigations of actual faults suggests that up to 6-way testing may be needed to reveal some faults
• Combinatorial $t$-way testing for $t$ up to 6 is now possible
• Combinatorial testing is one of many complementary methods for software and systems testing
• ACTS is useful tool for generating $t$-way test suites, supports constraints
• Combinatorial testing useful when test cases can be expressed in terms of factors with discrete test values