

Black Box Software Testing

(Academic Course - Fall 2001)

Cem Kaner, J.D., Ph.D.

Florida Institute of Technology

Section: 13 :

Domain Testing Paradigm

Contact Information:

kaner@kaner.com

www.kaner.com (testing practitioners)

www.badsoftware.com (software law)

www.testingeducation.org (education research)

Copyright (c) Cem Kaner 2001.

I grant permission to make digital or hard copies of this work for personal or classroom use, without fee, provided that (a) copies are not made or distributed for profit or commercial advantage, (b) copies bear this notice and full citation on the first page, and if you distribute the work in portions, the notice and citation must appear on the first page of each portion. Abstracting with credit is permitted. The proper citation for this work is Cem Kaner, *A Course in Black Box Software Testing (Academic Version)*, Fall 2001, www.testing-education.org. To copy otherwise, to republish or post on servers, or to distribute to lists requires prior specific permission and a fee. Request permission to republish from kaner@kaner.com.

Domain Testing

- AKA partitioning, equivalence analysis, boundary analysis
- Fundamental question or goal:
 - This confronts the problem that there are too many test cases for anyone to run. This is a stratified sampling strategy that provides a rationale for selecting a few test cases from a huge population.
- General approach:
 - Divide the set of possible values of a field into subsets, pick values to represent each subset. Typical values will be at boundaries. More generally, the goal is to find a “best representative” for each subset, and to run tests with these representatives.
 - Advanced approach: combine tests of several “best representatives”. Several approaches to choosing optimal small set of combinations.
- Paradigmatic case(s)
 - Equivalence analysis of a simple numeric field.
 - Printer compatibility testing (*multidimensional variable, doesn't map to a simple numeric field, but stratified sampling is essential.*)

Domain Testing

- **In classical domain testing**
 - Two values (single points or n-tuples) are equivalent if the program would take the same path in response to each.
- **The classical domain strategies all assume**
 - that the predicate interpretations are simple, linear inequalities.
 - the input space is continuous and
 - coincidental correctness is disallowed.
- **It is possible to move away from these assumptions, but the cost can be high, and the emphasis on paths is troublesome because of the high number of possible paths through the program.**
 - » Clarke, Hassell, & Richardson, p. 388

Domain Testing

- **Strengths**
 - Find highest probability errors with a relatively small set of tests.
 - Intuitively clear approach, generalizes well
- **Blind spots**
 - Errors that are not at boundaries or in obvious special cases.
 - Also, the actual domains are often unknowable.

Domain Testing

- **Some of the Key Tasks**
 - Partitioning into equivalence classes
 - Discovering best representatives of the sub-classes
 - Combining tests of several fields
 - Create boundary charts
 - Find fields / variables / environmental conditions
 - Identify constraints (non-independence) in the relationships among variables.

Domain Testing

- **Some Relevant Skills**

- Identify ambiguities in specifications or descriptions of fields
- Find biggest / smallest values of a field
- Discover common and distinguishing characteristics of multi-dimensional fields, that would justify classifying some values as “equivalent” to each other and different from other groups of values.
- Standard variable combination methods, such as all-pairs or the approaches in Jorgensen and Beizer’s books

Domain Testing

- **Practice Exercises**

- Find the biggest / smallest accepted value in a field
- Find the biggest / smallest value that fits in a field
- Partition fields
- Read specifications to determine the actual boundaries
- Create boundary charts for several variables
- Create standard domain testing charts for different types of variables
- For finding variables, see notes on function testing

Additional Basic Exercises

- **Examine common dialog boxes, such as these in MS Word:**
 - Print dialog
 - Page setup dialog
 - Font format dialog
- **For each dialog**
 - Identify each field, and for each field
 - Name the type of the field (integer, rational, string, etc.)
 - List the range of entries that are “valid” for the field
 - Partition the field and identify boundary conditions
 - List the entries that are almost too extreme and too extreme for the field
 - List a small number of test cases for the field and explain why the values you have chosen are best representatives of the sets of values that they were selected from.
 - Identify any constraints imposed on this field by other fields

Domain Testing: Interesting Papers

- Thomas Ostrand & Mark Balcer, *The Category-partition Method For Specifying And Generating Functional Tests*, Communications of the ACM, Vol. 31, No. 6, 1988.
- Debra Richardson, et al., *A Close Look at Domain Testing*, IEEE Transactions On Software Engineering, Vol. SE-8, NO. 4, July 1982
- Michael Deck and James Whittaker, ***Lessons learned from fifteen years of cleanroom testing. STAR '97 Proceedings*** (in this paper, the authors adopt boundary testing as an adjunct to random sampling.)
- Richard Hamlet & Ross Taylor, *Partition Testing Does Not Inspire Confidence*, Proceedings of the Second Workshop on Software Testing, Verification, and Analysis, IEEE Computer Society Press, 206-215, July 1988

Domain Testing: Paper of Interest

- *Partition Testing Does Not Inspire Confidence*, Hamlet, Richard G. and Taylor, Ross, Proceedings of the Second Workshop on Software Testing, Verification, and Analysis, IEEE Computer Society Press, 206-215, July 1988
- *abstract* = { Partition testing, in which a program's input domain is divided according to some rule and test conducted within the subdomains, enjoys a good reputation. However, comparison between testing that observes partition boundaries and random sampling that ignores the partitions gives the counterintuitive result that partitions are of little value. In this paper we improve the negative results published about partition testing, and try to reconcile them with its intuitive value. Partition testing is show to be more valuable than random testing only when the partitions are narrowly based on expected faults and there is a good chance of failure. For gaining confidence from successful tests, partition testing as usually practiced has little value. }
- From the STORM search page:
<http://www.mtsu.edu/~storm/bibsearch.html>