Embedded functions for test design automation

George B. Sherwood[1][0000-0003-0865-1679]

[1] Testcover.com LLC, 41 Clover Hill Road, Colts Neck NJ 07722, USA
sherwood@testcover.com

Abstract. Testcover.com introduced an embedded functions feature into its combinatorial test design service. The feature allows functionally dependent relations among test factors to be defined as functions in a general purpose programming language, PHP. These relations enforce constraints among test factor values and insure that all valid combinations of determinant factors are considered for the test design. Resulting usability improvements enable automated pairwise test designs to meet novel objectives: Cover equivalence classes of expected results; verify univariate and multivariate equivalence class boundaries; verify corners among intersecting boundaries and edges. The demonstration illustrates how embedded functions can improve automation, accuracy, control and flexibility in the test design process.

Keywords: automated test design, boundary testing, combinatorial testing, constraints, embedded function, equivalence class, functional dependence, interaction testing, PHP, software test design, test case generation.

Introduction

Recent concerns about engineering the Internet of Things (IoT) include conflicting requirements and inadequate analysis and verification. Diomidis Spinellis observes: "No doubt, a paradigm shift from balkanized IoT applications to an integrated infrastructure in which individual IoT nodes are first-class citizens raises formidable challenges… when multiple IoT nodes and applications get integrated, diverse requirements will interfere with each other…" [1]

Similarly, Vinton Cerf writes: "Concerns for safety, security, privacy, and control must be assuaged by systematic analysis of increasingly complex use scenarios. It might even be argued that these analyses will need to be carried out automatically just to keep up with the non-linear growth in potential use cases and device interactions as the devices proliferate." [2]

This demonstration of embedded functions (EFs) illustrates the automated design of software tests to conform to requirements and meet test objectives. Embedded functions define and enforce required constraints among test factors (e.g. configurations and inputs). [3, 4] They can insure that required test factor combinations (e.g. for system states or equivalence classes) appear in test cases, and that invalid combinations do not. The EF feature resolves composite relations among test factors so that test cases conform to the chains of functional dependence.

When equivalence class factors or boundary value factors are included in the design, additional test objectives can be met automatically, as needed:

- Cover equivalence classes of expected results [4-6]
- Pair equivalence classes with nondeterminant factor values [4-6]
- Verify univariate and multivariate equivalence class boundaries [4, 6]
- Pair boundary values and edges to verify corner cases [4, 6]

The embedded functions feature represents test design automation progress, leading to improved usability and efficiency for practitioners. It offers significant contributions for testing systems developed in various processes. We anticipate a range of uses from embedded software to agile systems and the IoT. We know of no other automated test design tool that can meet these objectives for practicing software engineers.

Combinatorial testing. Combinatorial testing generates small sets of test cases to cover interactions among test factors in complex systems. These test cases must conform to system constraints to be valid. When a test case has an invalid combination of values, it cannot verify the expected result. If such a test case is omitted, the valid combinations it may have contained might be missing from the remaining test cases. So the test cases must contain all the required combinations and none of the invalid ones.

A second constraint challenge is generating test cases that lead to a particular expected state or class of expected results. Test cases need combinations of factor values that steer the system under test for the behaviors to be verified. Otherwise the test process may be inefficient and time consuming.

Research and development progress has led to tools that conform to constraints, but constraints remain a challenge to the broad adoption of combinatorial testing in practice. [3]

Functional dependence. Constraints in test models can be expressed as functionally dependent relations. For example, test factors for a date input may include Month, Day and Year, with respective values nov, 1 and 2017. The first and last day of each month are boundary values for the Day factor. All months start with Day 1, but the last day of the month depends on the values of Month and Year. The Day value can be determined by a function last_day(Month,Year). With an appropriate definition for the last_day function, the boundary values for Day can be listed as 1 and last_day(Month,Year). In this relation Month and Year are determinant factors, and Day is the dependent factor.

The essential concept for embedded functions is to use simple functions in a well-known language to describe the system and test constraints. Automated evaluation of embedded functions reduces the manual analysis work and yields designs that can meet a variety of test objectives. The analysis is limited to selecting test factors and values, and defining the embedded functions. PHP was selected as the language for this implementation of embedded functions. [7]

Description

Testcover.com provides an online, commercial test design service. [8, 9] The Software as a Service is accessible with a standard browser. It can be used with a variety of development environments, processes and tools.

The embedded functions feature was proposed in 2015. [3] During its development elements of the feature were described and demonstrated at IWCT 2016. [5] Subsequently support for substitution functions (described below) and for the functions editor was included. The EF feature was deployed for controlled introduction June 30, 2017.

Basic Blocks. Reference [6], example 2 illustrates a test design using calendar constraints. There are 5 blocks of factor values (Month, Day and Year) in the test design request. All combinations of factor values in each block are allowed; combinations that do not appear in any blocks are disallowed. Thus the set of allowed combinations is the union of all the blocks' combinations.

Forty test cases are generated, and they cover all allowed pairs of factors. The design covers the Day boundary values also.

Embedded Functions. Reference [6], example 2 also illustrates the calendar design using a single block and the embedded function last_day($month,$year). The $month and $year factors are renamed as PHP variables, so they can be arguments for the last_day function. These factors list all months and years to be included in the design. The Day factor contains the fixed values 1 and 10, and the last_day($month,$year) function.

The last_day function is a wrapper for the PHP internal function cal_days_in_month. [6, 7] When the request is processed, the last_day function is evaluated for all combinations of $month and $year values. These are used to generate the same 40 test cases as with the basic blocks request.

The last_day function is a *combination function* to be evaluated *before* test case generation. [3] Each combination function is called for all combinations of its arguments' values. The function returns a list of one or more allowed values for generating test cases. Combination functions should not return values for invalid combinations of arguments.

Substitution functions are evaluated *after* test case generation. [3] They are used to relax the requirement for pairwise coverage, e.g. when test case values should be random or unique (function fUser in example 6 [6]). In example 5c [4, 6] the $Weight factor value is a test input computed by the substitution function Weight_boundary. Pairing the boundary value with other factors is not an objective for this design, so Weight_boundary is defined as a substitution function.

Substitution functions should return an individual value for each test case (not a list). They also resolve composite relations among test factor values, similarly to those of combination functions.

Equivalence Class Factors and Boundary Value factors. Use of embedded functions, together with equivalence class factors and boundary value factors, enable au-

tomated combinatorial test designs to meet the objectives listed above. Test cases can be constrained to cover expected equivalence classes by the inclusion of equivalence class factors. When their factor values are given by combination functions, every expected class determined by the other factors appears in at least one test case. [4-6]

Similarly, test cases can be constrained to compute and cover boundary values automatically, using factors to specify the required boundaries. [4, 6] When multiple boundaries are covered, the intersections cover their respective corner cases. And whether all corners or selected corners are covered can be controlled by the assignment of combination or substitution evaluation to each EF. [4, 6]

Conclusions

The demonstration shows that specifying constraints (including equivalence class and boundary requirements) as simple functions, in a language familiar to software engineers, can automate much of the analysis for software test design. Moreover, functions can be reused for different designs, and they can enhance consistency and accuracy as test factors and values change.

Reference [4], sections 4-6 illustrate in detail the control and flexibility offered by embedded functions. They show different test design choices, based on various test objectives, leading to different patterns of coverage.

Test designs must accommodate system complexity and size, as well as diverse objectives. The embedded functions feature offers improvements in automation, accuracy, control and flexibility, for advances in test efficiency and quality.

References

1. Spinellis, D.: Software-engineering the Internet of Things. IEEE Software 34(1), 4-6 (2017).
2. Cerf, V. G.: A brittle and fragile future. Communications of the ACM 60(7), 7 (2017).
3. Sherwood, G. B.: Embedded functions in combinatorial test designs. In: IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp. 1-10. IEEE, Graz, Austria (2015).
4. Sherwood, G. B.: Test design automation: equivalence classes, boundaries, edges and corner cases. 2016/7/3. http://testcover.com/pub/background/ecbecc.pdf, last accessed 2017/7/27.
5. Sherwood, G. B.: Embedded functions for constraints and variable strength in combinatorial testing. In: IEEE Ninth International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp. 65-74. IEEE, Chicago, IL, USA (2016).
6. Testcover.com embedded functions examples (2017), http://testcover.com/pub/background/examples2017.php, last accessed 2017/7/27.
7. M. Achour, F. Betz, A. Dovgal, et al.: PHP Manual. http://php.net/manual/en/index.php, last accessed 2017/7/27.
8. About Testcover.com, http://testcover.com/pub/about.php, last accessed 2017/7/27.
9. Testcover.com performance, http://testcover.com/pub/performance.php, last accessed 2017/7/27.