最近出てきたテスト技法

辰巳 敬三
Notes on new software testing techniques / Keizo TATSUMI

1. はじめに

4年程前にソフトウェア・テストPRESSで「xxテストの氾濫と混乱ーテスト技法ってどのくらいあるのだろう」というコラムを書きました[1]。そのとき、Myers、Beizer、Jorgensen、Kaner、Craig、Copelandの各氏が書いたテストの書籍やSWEBOKに出現する『xxテスト(xx testing)』をひろい出して数えたのですが、ざっと70個以上ありました。また当時最新のJSTQB用語集ではさらに多い120個ありました。このように、テストの分野ではいろいろな名称の技法が提案されてきましたが、最近もいくつか新しい名称のテスト技法が出てきています。

本稿では、2000年以降に現れた以下のテスト技法について調べたことを紹介します。

- Combinatorial testing
- Search-based testing
- Concolic testing

2. Combinatorial testing (組み合わせテスト)

直交表などを利用して組み合わせテストケースを生成するテスト技法は、日本では1984年の富士通の論文、米国では1985年のMandlの論文が嚆矢で、1990年代前半にAT&Tグループ企業の論文が発表されて以降、米国で広まりました。このようにテスト技法そのものは30年近い歴史がありますが、"Combinatorial testing"(組み合わせテスト)という名称で議論され研究が深まったのはここ10年です。NISTのWebサイトに掲載されている研究論文数の推移のグラフ[2]を見ると2003年ころから論文件数が増え始めていることが分かります。そして、今年4月にはICST 2012に併設で初めての組み合わせテストの国際ワークショップが開催されるに至っています。

このように比較的新しい名称であるため、ISTQB(JSTQB)用語集にはまだ"Combinatorial testing"(組み合わせテスト)という用語は掲載されておらず、2007年12月の用語集改訂でようやくOrthogonal Array testing(直交表テスト), Pairwise testing(ペアワイズテスト)が追加されたという状況です。なお、Combinatorial Interaction Testing (CIT) という名称を使っている論文もあります。

組み合わせテストと言うと、直交表やペアワイズによるテストケース生成方法のみに目が向いてしまいがちですが、研究分野(技術要素)は以下のような広がりが出てきています[3]ので、現場で組み合わせテスト技法を適用したり改善を考えるときに関連の文献を調べると参考になると思います。

1) モデル化(入力パラメタモデリング) Modeling
2) テストケース生成 Test case generation
3) 制約条件 Constraints
4) 障害解析 Failure characterization and diagnosis
5) 組み合わせテスト手順の改善と適用 Improvement of testing procedures and the application of CT
6) テストケースの実行優先順位付け Prioritization of test cases
7) メトリック Metric
8) 評価 Evaluation

3. Search-based testing

ソフトウェア工学における各種の課題をヒューリスティックサーチの技法を使って解決を図るSearch-based software engineering (SBSE)が2001年に英国のHarmanとJonesにより提案されました[4]。

このサーチ技法をテストの分野に適用するのがSearch-based testingと呼ばれる技法で、2004年に英国Sheffield大学のMcMinnが"Search-Based Software Testing (SBST)"という用語を使い始めたのが最初です[5]。サーチ技法を使った手法そのものは以前から発表されていましたが、HarmanやMcMinnがSBSEやSBSTという名称をつけて技法として体系化する取り組みを開始しました。

サーチ技法としては、「ヒューリスティック(heulistic、発見的解法)」と呼ばれる近似解を求めるアルゴリズムが用いられ、いろいろな問題に適用できるものは「メタヒューリスティック(metaheulistic)」と呼ばれます[6]。サーチ技法として用いられるアルゴリズムには次のようなものがあります。

- Hill climbing (山登り法)
- Simulated annealing (焼きなまし法)
- Genetic algorithms (遺伝的アルゴリズム)
- Tabu search (タブーサーチ)
- Greedy algorithm (欲張り法)

McMinnの論文[4]によると、テスト分野にサーチ技法を適用するためには、解決したい問題をサーチ技法で扱えるようにするための「表現(representaion)」とサーチ技法で求めた解を評価するための「適合関数(fitness function)」の2つを決める必要があります。サーチ技法が使われているテストの例として次のものがあります。

1) 時間に関するテスト(Temporal testing)の分野
   実行時間の最良(best-case)値、最悪(worst-case)値を求めるための入力値を生成する手段としてサーチ技法が用いられています。適合関数としてはソフトウェアの実行時間が用いられています。
2) 機能テスト(Functional testing)の分野

駐車場の空きスペースを管理し適切に車を誘導する制御装置のテストとして、サーチ技法によるシミュレーションが行われている例があります。適合関数は駐車するまでに衝突する最短距離を用い、制御装置の誤りの検出を図っています。

3) 構造テスト(Structural testing)の分野

プログラムの経路(パス)を実行するためのテストデータの生成のためにサーチ技法を用いるものですが、そもそも最初にテストでサーチ技法が用いられたのがこの分野です。適合関数はパスやブランチのカバレッジということになります。

以上から分かるように、この技法はブラックボックス・テスト、ホワイトボックス・テストといったテスト技法と並立するものではなく、それぞれのテスト技法を適用するための実装手段としてサーチ技法を用いるというものです。

なお、前述の Combinatorial testing はテストケース生成アルゴリズムとしてサーチ技法を使っているものがありますので、Search-based testing の範疇に入る場合もありそうです。

4. Concolic testing

"concolic"という単語は聞き慣れないですね。それもその筈で、これは concrete と symbolic という単語を組み合わせた混成語[7]で、2005 年に米国イリノイ大学の Sen (現在はカリフォルニア大学バークレイ校)の論文[8]で初めて使われました。concrete は、数値など具体的な値(concrete values)をプログラムに与えて実行する(concrete execution)通常の「テスト」のことで、symbolic は、具体的な値ではなく記号値をプログラムに模擬的に与えて"実行"することによりソフトウェアの解析を支援する「記号実行(symbolic execution)」のことです。

記号実行は 1970 年代半ばに研究・発表され始めた技法です[9]。プログラムの実行経路の分析において、分岐がある場合、入力が具体的な値のときはどちらに分岐するかは一意に決まりますが、記号値のときは決まるとは限りません。記号実行では、それぞれの分岐方向が成立する条件を解析して、これらの記号を含んだ論理式や数式で出力を表現することになります。

Concolic testing は、通常のテストと記号実行を連携させて実行することにより、プログラムの実行可能経路を全て確認できるテストデータを自動生成することを目標としています。テストの手順は、(1)プログラムに具体的な入力を入れて実行する、(2)記号実行を行い分岐点における分岐条件を求める、(3)(1)で実行されなかった経路を実行するための入力を(2)に基づいて決定する、(4)(3)で求めた入力で実行する、ということを繰り返すという流れになります。

なお、Concolic testing と同様のアプローチの技法として Whitebox fuzz test(あるいは Whitebox fuzzing)[10]、Dynamic symbolic execution という名称で研究を進めているグループもあります。

5. おわりに

最近出てきた 3 つの新しい名称のテスト技法を紹介しました。一般に認知されて定着しているとするには最低 10 年くらいの状況は見るべきだと思いますが、今回紹介した技法は概ね認知されたと考えてよさそうです。これらに共通するのは、以前からある考え方や手法であるが、コンピューティングパワーとアルゴリズムの進化によりツール化が可能となり実用可能なテスト技法になったということではないでしょうか。新たなテストの分野を開拓する研究者や技術者にとって、ツール化する力は必須のものになっています。

【参考文献】

[1] 辰巳敬三, ソフトウェアテスト・ヒストリー, ソフトウェア・テスト PRESS, Vol.8, 技術評論社, 2009

[2] NIST, Combinatorial Testing Papers by Year
http://csrc.nist.gov/groups/SNS/acts/research-program.html

[3] C. Nie and H. Leung, "A Survey of Combinatorial Testing," ACM Computing Surveys, Vol. 43, No. 2, 11:1-11:29, 2011

[4] M. Harman and B. Jones, “Search-Based Software Engineering,” Information and Software Technology, 43(14), 2001
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.9996&rep=rep1&type=pdf

[5] P. McMinn, “Search-Based Software Testing: Past, Present and Future,” SBST 2011, IEEE Computer Society
http://philmcminn.staff.shef.ac.uk/papers/2011-sbst.pdf

[6] Alberto Palacios Pawlovsky／アズウィ, 考えるコンピュータのアルゴリズム Java&C++で学ぶ遺伝的進化的免疫アルゴリズム, ソフトバンククリエイティブ, 2007

[7] Concolic testing - Wikipedia
http://en.wikipedia.org/wiki/Concolic_testing

[8] K. Sen, D. Marinov and G. Agha, “CUTE: A Concolic Unit Testing Engine for C,” ESEC/FSE'05, 2005
http://www.cis.upenn.edu/~alur/CIS670/cute.pdf

[9] 玉井哲雄、福永光一, 記号実行システム, 情報処理, Vol.23 No.1, 1982
http://ci.nii.ac.jp/naid/110002761556

[10] P. Godefroid, M. Levin and D. Molnar, “Automated Whitebox Fuzz Testing,” NDSS'08, 2008
http://research.microsoft.com/en-us/projects/atg/ndss2008.pdf