ソフトウェアのテスト設計手法

Test Architecture Design Method for Large-Scale Software Development

加瀬　直樹　　鷲見　毅　　市田　憲明
■KASE Naoki　■SUMI Takeshi　■ICHIDA Noriaki

ソフトウェア開発の最終工程となるテスト工程では，限られた期間にできるかぎり多くのテストを実施して品質確保に努めている。しかし近年，ソフトウェアの大規模・複雑化に伴ってテストすべき項目は増える一方で，開発期間は短縮傾向にあり，このような手法では品質確保が難しくなる懸念がソフトウェア開発共通の課題として生じている。

東芝は，全体としてどのようなテストがどれだけ存在するかを可視化するテストアーキテクチャを用いて，テスト全体の多寡のバランスを調整し効果的にバグ（コーディングの誤りや欠陥）を発見する新たなテスト設計手法を開発し，デジタル製品や社会インフラシステムの開発におけるテストに活用している。

In the final stage of software development, optimization of the testing process by performing as many tests as possible during a limited period is required to assure quality. While the number of items to be tested has increased with the expansion of large-scale and complex software systems in recent years, shortening of the software development period is a growing trend. It has therefore become difficult to assure the quality of software products by using a large number of test cases as in the past.

Toshiba has developed a software test design method using test architecture diagrams that make it possible to visualize the entire software tests of a product, and to add or delete missing or redundant test cases. This test design method is now being applied to the development of some of our digital products and social infrastructure systems.

1　まえがき

ソフトウェアの大規模・複雑化に伴い，ソフトウェアの動作を確認するためのテストも大規模・複雑化している。テストは従来，限られた期間で考えうるかぎり多くのテストを実施することで品質を確保するのが一般的であったが，開発期間の短縮により，効率化と品質確保の板挟みにあっている。

テストには二つの側面がある。トラブルの原因となるバグを発見するためのものと，テストの対象が正しく動作することを確認するためのものである。限られた期間で妥当な品質を確保するには，動作確認のテストよりはバグを発見するテストを増やす必要がある。バグを発見すれば，次の品質向上のための修正作業につなげることができるからである。

期間短縮のためにテストを効率化するには，バグ発見の効果の高いテストを選別できなければならない。しかし従来は，テストの選別を作業者の経験やノウハウに頼らざるをえなかった。

テストの選別を合理的に行うためには，なぜそのテストが必要かあるいは不要かを明示できなければならない。また，テスト作成の際には，個々の内容を設計するだけでなく，大規模・複雑化したテストでは，特に他のテストとの関係を考慮しバランスをとることが重要である。他と内容が重複していれば不要であるし，他とは違う観点のテストであれば残すと判断できる。

東芝は，ソフトウェアのテスト全体のバランスを調整し，最適で効果的なテストを設計するために，テストが重複しているかどうかを可視化するテスト設計手法を開発した。それぞれのテストが全体の中でどのような位置づけになっているかというテストの構造，すなわちテストアーキテクチャを可視化する手法である[1]。ここでは，開発したテスト設計手法の概要と特長，及び製品開発への適用事例について述べる。

2　テスト設計手法の概要

テスト工程では，通常，どのようなテストをすべきかを記述したテストケースを作成するテスト設計の作業と，そのテストケースに基づきテスト環境を準備し，テスト対象の操作を行い，その結果を判断するテスト実施の作業がある。

従来のテスト工程は，単体テスト，結合テスト，及び総合テストに分割され，各工程でテスト設計とテスト実施を行っていた。今回開発した手法では，工程間の重複を含めて全体を最適化するため，一度全ての工程で実施すべきテストを俯瞰（ふかん）し全体の見通しを立て，テストを選別してから各工程に分割し直す方法を採用した。このためテスト設計を，仕様書や設計書からテスト全体の構造を決めるテストアーキテクチャ設計と，各々のテストの内容を作成するテストケース設計の二つに分けた（**図1**）。テストアーキテクチャ設計において考慮するポイントを次に述べる。

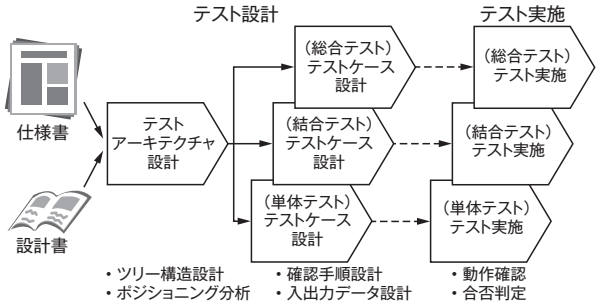

図1．テスト工程の全体像とテスト設計の位置づけ ― テストアーキテクチャ設計で全体の構成を決めてから，各テスト工程のテストケースを設計し，各テスト工程でテストを実施する。

Test architecture design and positioning of test item design

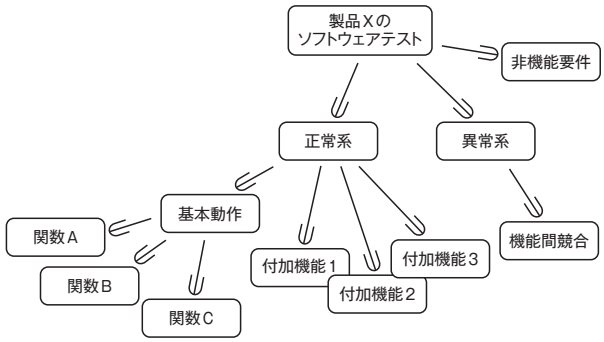

図2．テストのツリー構造 ― テスト全体の構造として，テストを分割していく過程をテストの親子関係としてツリー構造で表すことができる。

Tree structure of test items

2.1　テストケースの狙い

テストアーキテクチャ設計では，抜けや漏れをなくすための網羅性確保が最重要である。テストケースを作成した際に，それぞれのテストケースがどこまでの範囲をカバーできているかというスコープ，すなわち守備範囲を特定する必要がある。この手法では，守備範囲をテストケースの狙いと呼び，ソフトウェアのどの部分を対象にした何のテストであるかを明確化する。テストケースには，大小様々な守備範囲が設定される。守備範囲が明確になれば，それぞれを重ね合わせると穴が見つかるのでテストの抜けや漏れを発見できる。

守備範囲の大きなテストケースは，テストケースの狙いが大ざっぱで曖昧になりがちでバグを見逃しやすく，効果的なテストを構成しにくい。これを比較的小さな守備範囲のいくつかのテストケースに分割し，テストケースの狙いを明確にしていくことが，テストアーキテクチャ設計では重要になる。

2.2　テストケース間の関係性

守備範囲の大きなテストケースでは，何をどう分割したかというテストケースの親子関係を記録しておく必要がある。この親子関係のことをテストの包含関係と呼ぶ。包含関係はツリー構造で表すことができる（**図2**）。

ただし，テストはトップダウンで全てが分割していけるとは限らず，その網羅性を確保するため様々な観点からテストケースを作成する。このため，大きさも守備範囲もまちまちなテストケースが洗い出される。このような雑多なテストケースを整理し，分割や統合をしながらテストケースの大きさや守備範囲の重なり具合を調整して全体の構造を決定していく。

テストケースの包含関係を明確にしておけば，いくつかの子テストで確認しようとしていた狙いをまとめて一つの親テストで一括して実施できたり，一つの親テストを分割して結合テストと総合テストの二つの工程でテストを構成できたりする。

2.3　テストのポジショニング

テストの守備範囲は，自然言語で表現するよりも位置情報として可視化したほうが直感的で扱いやすい。テストケースには，テスト対象の粒度や，テストの種類，想定する実行環境やリスクの大きさなど，様々な属性が付加されている。その中から二つの属性に注目すると，2次元座標上の点として位置づけることができる（**図3**）。

テストケースの位置づけを様々な軸で分析すると，そのテストケースが何を狙ったものなのか，どういうことを確認しようとしているのか，を視覚的に把握することができる。すなわち，何枚かの2軸の座標系を用意し，テストケースがその座標系上でどの位置に配置できるかを確定していけば，テストケースの持つ狙いの本質を明確化していくことができる。

更に全てのテストケースをそれぞれの座標上に配置してみると，全体のバランスや偏りを把握でき，必要なテストケースの漏れや同じようなテストケースの重複した作成を発見できる。あるいは，テストケースの粒度がふぞろいである状況を発見することもある。これらは，テストケースの内容を設計するうえで曖昧さやゆらぎの原因となるため，狙いをより明確にして，テストケースを分割したり統合したりして修正する。

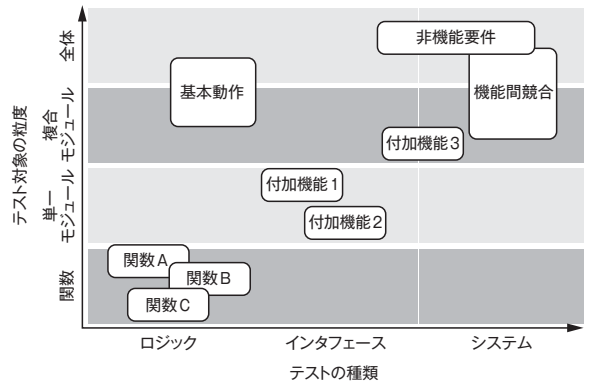

図3．テストのポジショニング ― テストケースを座標上に位置決めして，他との位置関係を可視化し，妥当性を分析する。

Positioning of test items

3　テスト設計手法の特長

前述したテストのツリー構造とポジショニングの図を用いてテストアーキテクチャを可視化し，テストの全体構造を決定することでテスト設計を進める。ここでは，開発したテスト設計手法の特長について述べる。

3.1　テストのフロントローディングを可能にする抽象表現

図1では，テスト工程をテスト設計とテスト実施に分離したが，これはテストをなるべく早期に着手するフロントローディングを目指したものである。早期にテスト設計に着手できれば，テストの設計にあたり必要な開発設計側の問題点，すなわち設計が曖昧であったり矛盾を含んだりしている部分や見落としている部分を発見でき，即座にフィードバックすることができる。

更に，テスト設計の中を，テストで扱うデータや動作手順を決めるテストケース設計と，その前段階であるテストの構造を決め個々のテストケースの狙いを定める部分，すなわちテストアーキテクチャ設計とに分離した。

従来，テストはソフトウェア開発工程の終盤に行われるため，通常は開発設計や実装が完了しており，テスト対象の仕様や動作は明確になっている。これに対して，この手法では，実装や開発設計が完了していない段階でテスト設計に着手することになる。そこで，テストアーキテクチャ設計ではテストケースの狙いを明確化することに重点を置き，確定していない設計や実装を想定しなければならない具体的なテスト手順やテストデータの設計，すなわちテストケース自体の作成は後回しにした。テストアーキテクチャを設計する際のテストケースは抽象的な表現にとどめてテストケースの特性を明らかにし，開発設計が進んだ際に具体的なテスト手順やテストデータの設計に進む二段階方式にした。

3.2　網羅性をまず重視しMECEは最終的に実現

テストアーキテクチャを記述する際に，どうしても抜けなく重複なく（MECE：Mutually Exclusive and Collectively Exhaustive）を気にしがちである。最終的にはテストケースの狙い，すなわち守備範囲はMECEであるが，初めからMECEを求めすぎると，視野が狭まってテスト設計が行き詰まったり，大きな抜けが生じたりすることがある。そこで，テストケースの守備範囲は重なりを許容し，網羅性を確保してから可視化によって重複を省いていく手法とした。

テストのポジショニングの軸自体も，全てをMECEにしようとすると軸の選択が困難になる。表現しやすい軸で概略を描き，テストケースに重複があれば軸も含めて洗練していく，というのがよい。これは，テストでは網羅性が最重要であるという考え方に基づいているからである。

3.3　図の表記は単純さを重視

テストアーキテクチャ設計は，大規模なテスト全体を対象にするため一度に全てを扱うことは困難で，部分的に作成しては他との重複や見落しを修正し洗練していく形をとる。そのため，図の表記は単純なものにし，なるべく制約のないものとした。例えば，従来であればテストケースの名前は中身を表す重要なものであるので正確さを求められたが，この手法では仮決めのまま作業してもよいとした。洗練化作業で分割や統合を繰り返すため，作業が収束してきた段階で妥当な名前を与える。

4　製品開発への適用事例

当社は，今回開発したテスト設計手法を，社会インフラや，デジタル製品，電子デバイスといった様々な分野の製品開発のテストへ適用し評価を行っている。また，製品開発におけるテストケースの新規作成だけでなく，既存製品のテストケースの分析でテストのよしあしを検証する作業も行っている。

4.1　電子デバイス製品のテストケース新規作成事例

電子デバイス製品の事例では，テスト対象の設計が十分に進んでいない段階でこの手法をテストケースの新規作成に利用した。テスト設計のフロントローディングの実現であり，ここでは，テストのツリー構造を作成していくことで，特に注力すべきテストを洗い出した。

従来手法では難しかった部分であるが，テストの全体構造を図として表現することによって，どのようなテストを実施しようとしているのか，どの部分が重要なのかを図の上で指し示して議論できるようになった。この作業を通して，機能的な確認項目以外に，特に設計が複雑でバグの残存リスクが高い資源競合のテストを，モジュール結合の段階，すなわち結合テストで重点的に手厚く実施するテストアーキテクチャを示すことができた（**図4**）。従来，総合テストで確認していたものを前倒しするテストアーキテクチャである。

また，これによってテスト工程全体を俯瞰でき，テストの規模見積りに利用することができた。すなわち，どのようなテス

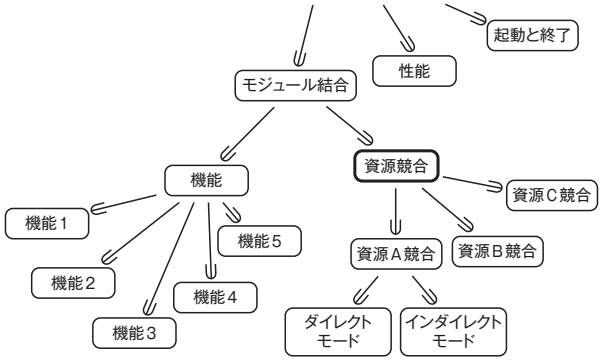

図4．電子デバイス製品への適用事例 ― バグの残存リスクが高い資源競合のテストを総合テストからモジュール結合テストに前倒しした。

Example of application of software test design method to development of digital products

トケースがあり，どれだけの件数となるかを見積もることが，設計の具体化が完了する前に可能となる。

4.2　社会インフラシステムのテストケース分析事例

社会インフラシステムの既存のテストケースを分析した事例では，派生開発によって蓄積してきたテストケースの中で削減できるところはないかを分析した。品質重視から，仕様変更や追加によってテストが新たに加えられ続けたため，似たようなテストケースで重複感はあるがなかなか削減しにくく，テストケース数が多くなりすぎていた。

分析作業では，既存のテストケースを2軸のポジショニング図に配置した。初期分析として，縦軸がテスト対象の大きさ，横軸が要求の大きさである図と，縦軸は同じで，横軸がテスト環境である図の二つの図を描いた。テストケース分析作業では，テストケースの狙いやそれら相互の関係性を把握することが必要になるが，図示することで概略を容易に把握できた。更に，テストケースの内容を詳細に分析し，縦軸がタスク名，横軸が機能名の図などを用いて，各テストケースの特性のポジショニングを確定していった。途中，複数の領域にまたがるテストの狙いが曖昧なテストケースも存在したので，いくつかのテストケースに分割し洗練していった。

このように各テストケースのポジショニングを図示することで，テストが重複しているかを客観的に示すことができた。その結果，数千のテストケースの中で20 %程度に重複があったと分析できた。もちろん，全ての重複を避けることが賢明なわけではなく，ある程度の重複を許すことでリスクを低減することも重要である。

5　テスト設計手法のソフトウェアツール化

ここでは，テストアーキテクチャを表現する2種類の図の編集作業をツール化して利用しやすくした工夫[(2)]について述べる。手描きよりも速く効果的にテストアーキテクチャを記述することができ，作業者の煩わしさを軽減するためのものである。

前述のテストケースの包含関係を記述するツリー構造の図を，このツールではTRMap（Test Relation Map）と呼ぶ。TRMapはテストの構造化を主眼に置くので，テストケースはテストケース名だけの表示にとどめ，簡単に作成できるようにした。作業者はテストケースの洗出しとそれぞれの関係付けに集中し，テストケースを分割，統合して，テスト全体の構造を設計していく。見やすさを確保するための整列はツールが行う。

また，テストケースのポジショニング図をLTMap（Layer Tier Map）と呼ぶ。LTMapは縦軸と横軸を決め，テストケースを配置していく図である。テストケースそれぞれの特性は複数のLTMapを用いる。例えば，各軸がテスト対象とテスト環境の図とテスト対象とリスクの大きさの図を使って，同じ縦軸で横軸を変えた2枚の図で表現する。テストケースはそれぞれの図に現れるが，縦軸の座標は同じ対象を示していなければならない。このような制約条件を加味することはツールの得意とするところで，作業者はツールのサポートを受けながら，矛盾のない図を描いていくことができる。

6　あとがき

当社は，ソフトウェアの効果的なテストを行うために，テストの全体構造であるテストアーキテクチャを記述し，テストを選別するテスト設計手法を開発した。テストアーキテクチャでは，テスト全体のツリー構造と各テストケースのポジショニングを可視化する2種類の図を活用する。様々な開発プロジェクトに適用することで，手法の洗練化と拡張を進めている。特に，テストケースのポジショニング図の2軸をどのような座標系にすべきかのノウハウを蓄積している。今後は，これを汎用化しフレームワークとして手法に取り入れていく予定である。

また，テスト設計の効果は，最終的にはテストを実施しバグが発見できたかどうかで決まる。テストケースとその合否結果を管理するテスト管理システム[(3)]と連携することによって，テストアーキテクチャ洗練化のフィードバックにつなげていく。

文　献

(1)　鷲見　毅 他. “テスト設計手法PROST!”. FIT2011 第10回情報科学技術フォーラム講演論文集，第一分冊. 函館，2011-09，情報処理学会 他. 2011，p.43－50.

(2)　市田憲明 他. “大規模テストへのPROST!の適用”. ソフトウェアテストシンポジウム2012東京. 2012-01，ASTER. ＜http://www.jasst.jp/symposium/jasst12tokyo/pdf/A4-1_paper.pdf ＞，（参照2012-05-11).

(3)　河村　透 他. ソフトウェアのテスト管理システム. 東芝レビュー. 66，1，2011，p.32－36.

加瀬　直樹　KASE Naoki
ソフトウェア技術センター プロセス・品質技術開発担当参事。
ソフトウェアテスト技術の開発に従事。
Corporate Software Engineering Center

鷲見　　毅　SUMI Takeshi
ソフトウェア技術センター プロセス・品質技術開発担当主務。
ソフトウェアテスト技術の開発に従事。情報処理学会会員。
Corporate Software Engineering Center

市田　憲明　ICHIDA Noriaki
ソフトウェア技術センター プロセス・品質技術開発担当。
ソフトウェアテスト技術の開発に従事。
Corporate Software Engineering Center