Vol.2011-SE-171 No.5
2011/3/14

機能木を用いたテストケース管理方法の提案

史寧[†]　八重樫理人[††]　高木智彦[††]　古川善吾[††]

既存のテストケース管理ツールは，テストに必要なドキュメントやテストデータ，不具合情報などの管理や，テスト担当者間のコミュニケーションを支援する事を目的としており，テストケースと機能との間との関連性に関する情報が含まれていない．そのため，機能変更がおこった場合にテストケースを変更する必要があるかどうかを判断したり，実施する必要のあるテストケースをみつけるのに時間がかかり，効率的なテストやテストが妥当であることの確認はほとんど不可能である．

本稿は機能変更が発生した際に実施することが必要なテストケースを効果的に探しだすために，ソフトウェアの機能を機能木で表現してテストケースと関連付けることによってテストケースを管理する方法を提案する．

Test-Case Management by Using a Function Tree

Shi Ning[†], Rihito Yaegashi[††], Tomohiko Takagi[††] and Zengo Furukawa[††]

Almost all management tools of test-case aims at keeping necessary documents and test date or bug information, and supporting the communication between people in charge of tests. It does not have the information about the connection between the test-case and the function.

Thus, it is almost impossible to judge if the change of test-case is demanded or not when the function is changed. Since it takes too much time to find the necessary test-case to carry out, it is also nearly impossible to verify if the test is appropriate. In order to find out the necessary test-case when the function change is occurred, this paper shows the software functions as a function tree, and proposes the approach to manage the test-case.

1. はじめに

現在，ソフトウェアの機能は多種多様/複雑化し，ユーザのソフトウェアに対する要求は年々厳しくなっている．安定稼働中のソフトウェアであっても，機能の変更が繰り返し行われ，ユーザの要求を満足すべく様々な改良がおこなわれる．繰り返し行われる機能変更は，ソフトウェアシステム全体の品質保証を困難にしている．

現在，高品質なソフトウェアを作るための手法として，レビューやテスト[1)]の充実，ソフトウェア設計構造の最適化など様々な手法が提案されている．特にテストは，不具合の発見や修正を行うとともに，ソフトウェアの出荷品質を評価する重要な工程となっており，これは機能変更を伴うソフトウェアの更新においても同様である．しかしながら，軽微な変更であっても生成されるテストケースの数は膨大であり，限られた労力の中でそれらすべてを実施するのは困難な場合がある．たとえば，携帯電話に搭載されるソフトウェアでは，定められた一連の操作ステップを基に動作確認を行うテストケースの数は1機種あたり1万5000件以上もあると言われており，ドキュメントビューアやWebブラウザなどを搭載する高機能モデルに至っては10万件以上ともいわれている．生成されたテストケースをすべて実施することは負担になるだけでなく，たとえすべてを実施しないとしても，その中から最低限実施する必要があるテストケースを選び出すことも困難である．

多くのソフトウェア開発プロジェクトにおいて，表計算ソフトを用いたテストケース管理が従来行われていたが，数万件を超えるテストケースの実施状況や，テストに合格するまでの経緯，関連するドキュメントなどをまとめて管理することは難しく，こうした負担を軽減するためにテストケース管理ツールを利用する企業が増えている．しかしながら，既存のテストケース管理ツールは，テストに必要なドキュメントやテストデータ，不具合情報などの管理や，テスト担当者間のコミュニケーションを支援する事を目的としており，テストケースと機能との間との関連性に関する情報が含まれていない．そのため，機能変更がおこった場合にテストケースを変更する必要があるかどうかを判断したり，実施する必要があるテストケースをみつけたりするのに時間がかかり，効率的なテストやテストが妥当であることの確認はほとんど不可能である．

そこで本研究は，機能変更が発生した際に実施することが必要なテストケースを効果的に探しだすために，ソフトウェアの機能を機能木（Function Tree）と呼ばれる木構造で表現し，テストケースと関連付けることによってテストケースを管理する方法

† 香川大学大学院工学研究科
Graduate School of Engineering, Kagawa University
†† 香川大学工学部
Faculty of Engineering, Kagawa University

を提案する．

2. 既存のツールの解決方法

テストケースの管理方法について述べる前に，まずはテストケースについて述べる．簡単な例として，オンラインショッピングシステムのテストケースを表 1 に示す．なお，本稿では，このオンラインショッピングシステムを例題として議論を行う．テストケースは，一般的に以下の情報から構成される．

- ID
- タイトル
- 操作手順およびテストデータ
- 予想結果
- 作者
- 作成日時

「ID」「タイトル」「作者」「作成日時」はテストケースの管理のための情報で，「操作手順およびテストデータ」と「予想結果」はテスト実行のための情報である．しかしながら，これらの情報だけでテストケースを効果的に管理することは困難である．近年ソフトウェアの複雑化に伴って，ソフトウェアテストも複雑化している．小規模のソフトウェア製品でも，1 万件を越えるテストケースを用意することは珍しくない．開発現場ではテストを行うことが困難となりつつあるのが実情であり，大量かつ複雑なテストケースを効率的に扱う方法が求められている．たとえば，バグを修正したり機能の変更が生じたりした際には，当該ソフトウェアのテストケースをすべて実行するのではなく，変更した機能に関連するテストケースだけを実行することが求められる．しかしながら複雑なテストケースが大量に存在するため，本当に実行が必要なテストケースだけを検索することが困難な場合がある．

このような問題に対する解決方法として，一部のテストケース管理ツールではテストケースに対してキーワードを設定できるようになっている．テスト担当者は，任意のキーワードや条件を入力することによって，ある程度必要なテストケースを検索したり，検索したテストケースを分類したりすることができる．また，テストケース毎の重要度を算出し，テストケースの優先順位付けが可能なツールも存在する．しかしながら，キーワードを用いたテストケース管理方法には，キーワードを設定することに対する難しさがある．キーワードを適切に設定するための基準がなければ，キーワードの内容が偏ったり漏れたりすることが多い．その結果，検索あるいは分類したテストケースの内容も偏ったり漏れたりすることがある．3.では本章でしるした問題を解決するために，ソフトウェアの機能を機能木（Function Tree）と呼ばれる木構造で表現し，テストケースと関連付けることによってテストケースを管理する方法を提案する．

表 1　オンラインショッピングシステムのテストケースの例

ID	タイトル	操作手順およびテストデータ	予想結果	作者	作成日時
1	1 回検索して購入	1．顧客ユーザ A としてログインする． 2．メーカが S 社，値段が 5 万円以下のカメラを検索する． 3．3 番目の商品を選択して，一つを買う． 4．自分の注文状態を確認する． 5．ログアウトする．	異常なく注文が完了する． ※ 詳細は省略	史寧	2010-12-13
2	2 回検索して購入	1．顧客ユーザ A としてログインする． 2．メーカが S 社，値段が 5 万円以下のカメラを検索する． 3．メーカが C 社，値段が 6 万円以下のカメラを検索する． 4．ログアウトする．	正確な検索結果が得られる． ※ 詳細は省略	史寧	2010-12-13
3	商品登録	1．販売ユーザ B としてログインする． 2．１つの商品を自分の店に登録する． 3．登録した商品を検索する． 4．ログアウトする．	異常なく登録が完了する． ※ 詳細は省略	史寧	2010-12-13

3. 機能木を用いたテストケース管理方法

ソフトウェアの機能を機能木で表現することで、ソフトウェア全体をまんべんなく体系的に表すことができる．この機能木とテストケースを関連付けて管理することによって，複雑なテストケースが大量に存在するような場合であっても，必要なテストケースだけを効果的に検索したり，検索したテストケースの優先順位を生成したりすることができるようになる．

3.1 機能木とテストケースの関連付け

機能木は，木構造を用いてソフトウェアの機能を段階的に細分化しながら記述するためのモデルである．ソフトウェアに対する機能要求の全体像を体系的に記述することが可能であり，たとえば，クライアントサーバシステムのための機能木を生成する手法などが検討されている [2]．簡単な例として，オンラインショッピングシステムの機能木を図 1 に示す．システムの全体は，機能木のルートノードである「OL_Shopping」に対応しており，システムは「ユーザ管理」，「商品管理」，「商品販売」という主な機能から構成されている．そして，「ユーザ管理」は「ログイン」，「ユーザ情報管理」，「ログアウト」の子機能から構成されており，「商品管理」は「商品情報管理」と「商品検索」の子機能から構成されており，「商品販売」は「予約」と「注文」の子機能から構成されていることが分かる．

本手法では，特定の機能に関連するテストケースだけを検索できるようにするために，図 2 に示すように，機能木とテストケースの間の関連付けを行う．たとえば，テ

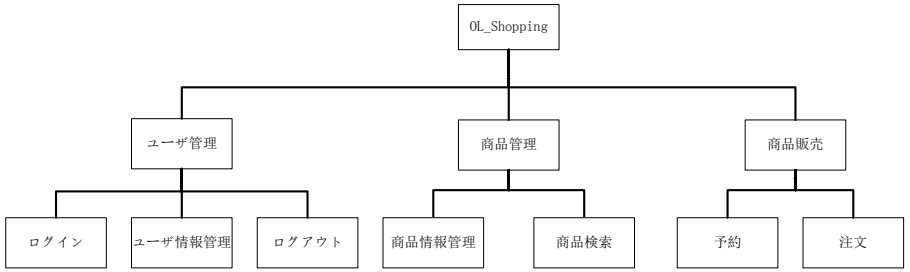


図 1. オンラインショッピングシステムの機能木

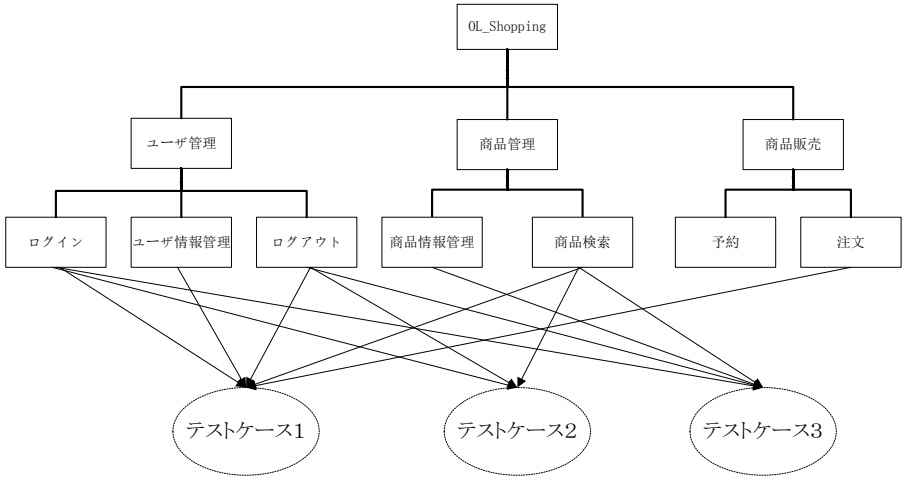


図 2. 機能木（図 1）とテストケース（表 1）の関連付け

ストケース 1 は,「ログイン」,「商品検索」,「注文」,「ユーザ情報管理」,「ログアウト」の 5 つの機能を実行しているので，機能木上の対応する 5 つのノードと関係線で接続されている．ただし，機能木とテストケースの間の関連は，表 1 の「操作手順およびテストデータ」の内容をテスト担当者が読めば分かるものの，明示的に定義されているわけではないので，自動的に関連付けを行うことはできない．そこで本研究で提案する手法では，機能実行系列という新たな情報をテストケース作成時にテスト担当者が付加する．表 1 の各テストケースの機能実行系列を図 3 に示す．機能実行系列は，「操作手順およびテストデータ」にしたがってテスト対象ソフトウェアを実行した場合の，起動される機能の順序を表したものである．操作手順の 1 ステップで複数の機能が実行される場合は，その複数の機能を機能実行系列に記述する．したがって，操作手順のステップ数と機能実行系列のステップ数が一致するとは限らない．

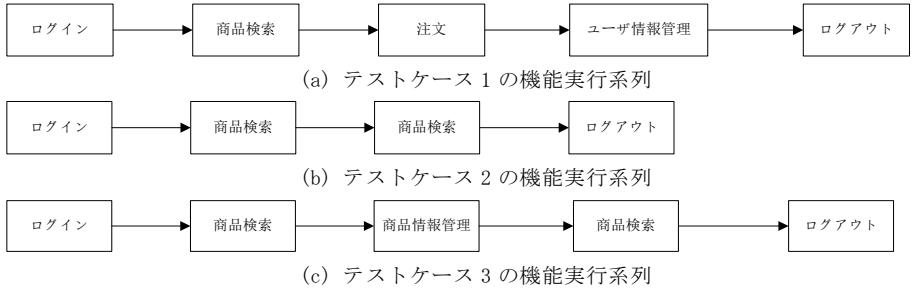


図 3. 表 1 のテストケースの機能実行系列

以上により，特定の機能に関連するテストケースだけを検索できるようになった．たとえば,「注文」という機能をテストしたい場合は,「注文」を検索条件としてテストケースを検索する.「注文」が含まれているテストケースは，テストケース 1 だけであるので，これが検索される．また本提案手法では，機能木のリーフノード以外のノードに対応する機能（子機能をもつ機能であり，親機能と呼ぶ）を検索条件に設定することもできる．たとえば,「商品販売」という機能をテストしたいと仮定する.「商品販売」はリーフノードではないのでテストケースと直接関連付けられていない．しかしながら,「商品販売」の子機能の「注文」は，テストケース 1 と関連付けられているので，テストケース 1 を検索することができる．

3.2 テストケースの優先順位付け

前節で示した方法によって，テストしたい機能に関するテストケースを検索できるようになった．しかしながら，多くのソフトウェア開発プロジェクトにおいてテストケースの量は膨大なものとなるので，検索したテストケースも多くなってしまう場合を考慮しておく必要がある．そこで本手法では，検索したテストケースを効率的に利用するために，テストケースの優先順位付けを行うための仕組みを用意する．テストケースの優先順位付けを行う際の基準として，機能間の影響度，有効ステップ数，お

よび総ステップ数を用いる．

3.2.1 機能間の影響度

機能間の影響度とは，他の機能を動作させた場合の，当該機能に対する影響の程度，あるいは，当該機能を動作させた場合の，他の機能に対する影響の程度のことである．一般的に機能間の影響は，他の機能と同じコンポーネントを使用したり，同じ資源（たとえば変数やファイルなど）にアクセスしたりすることに起因して生じる．影響度を表す尺度は，ソフトウェア開発プロジェクト毎に適切なものを検討する．本研究では特に定めないが，たとえば 1（影響度小）から 5（影響度大）までの 5 段階で表すことが考えられる．開発担当者は，定められた影響度の尺度を用いて，まず機能木中のすべてのリーフノード間の組合せについて影響度を評価する必要がある．例題のオンラインショッピングシステムにおける機能間の影響度を示したのが図 4 である．機能間の影響度は有向の属性である．たとえば図 4 の「注文」と「ユーザ情報管理」の間に着目すると，「注文」から「ユーザ情報管理」までの影響度は「4」で，逆の「ユーザ情報管理」から「注文」までの影響度は「3」となっていることが分かる．

次に各テストケースにおける機能間の影響度を評価する．テストケースにおける影響度は，テストケースの機能実行系列に基づいて算出する．すなわち，機能実行系列中のテストしたい機能に関連するすべての影響度から，それらの合計値または平均値または最大値を当該テストケースの影響度とする．たとえば，例題のオンラインショッピングシステムにおいて，「注文」の機能をテストしたいと仮定すると，テストケース 1 の機能実行系列（図 3 (a)）において着目すべき機能間の影響度は，図 5 (a)のようになる．したがって，テストケース 1 の影響度は，14（合計値），または 3.5（平均値），または 4（最大値）である．同様に，「商品検索」の機能をテストしたい場合の機能間の影響度は図 5 (b)のようになるので，テストケース 1 の影響度は 5（合計値），または 1.7（平均値），または 3（最大値）となる．

3.2.2 有効ステップ数および総ステップ数

有効ステップ数とは，テストケースの機能実行系列において，テストしたい機能に対する影響度をもっている機能の個数のことである．これに対して，テストしたい機能に対する影響度がない機能の個数を無効ステップ数という．有効ステップ数と無効ステップ数の和は，機能実行系列を構成するステップ数（総ステップ数）と一致する．たとえば，オンラインショッピングシステムの例題において「商品検索」をテストしたい場合のテストケース 1 の有効ステップ数を考える．図 5 (b)より，「商品検索」から「ユーザ情報管理」への影響度がないことが明らかであるので，テストケース 1 の機能実行系列におけるステップ 4（すなわち「ユーザ情報管理」）は無効ステップである．したがって，テストケース 1 の有効ステップは「ログイン」「商品検索」「注文」「ログアウト」の 4 つであり，有効ステップ数は 4 となる．

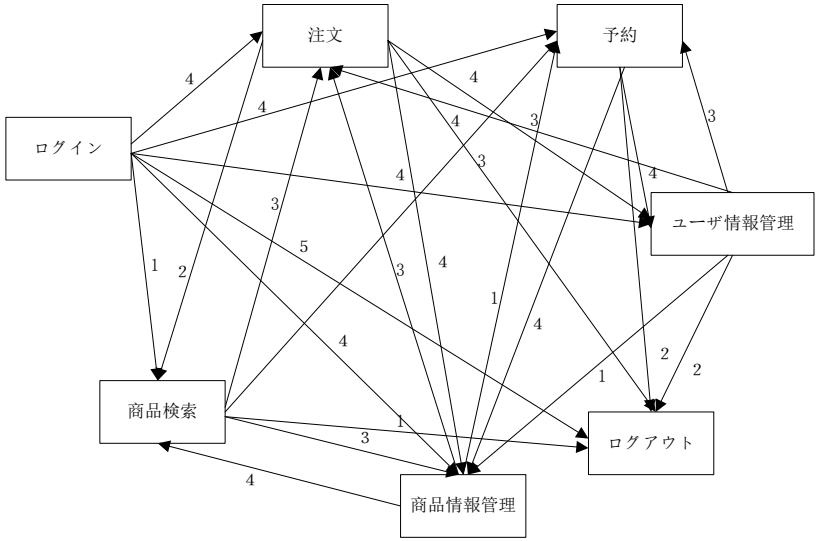


図 4. 機能間の影響度

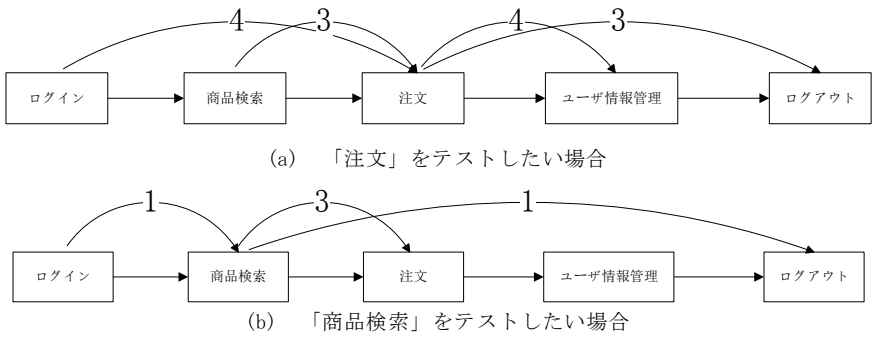


図 5. テストケース 1 における機能間の影響度

3.2.3 優先順位の決定方法

機能間の影響度，有効ステップ数，および総ステップ数を用いてテストケースの優先順位を決定するための手順を以下にまとめる．

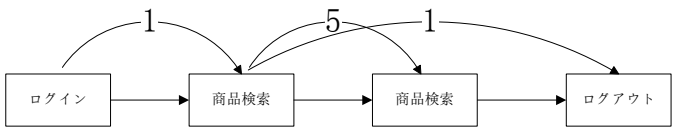
1. 機能間の影響度を合計値，平均値，最大値のいずれで評価するかを決定する．そして，各テストケースにおける機能間の影響度を評価する．
2. 有効ステップ数の多いものを優先するのか少ないものを優先するのかをテスト担当者が決定する．そして，テストケース毎の有効ステップ数を算出する．
3. 総ステップ数の多いものを優先するのか少ないものを優先するのかをテスト担当者が決定する．
4. 機能間の影響度，有効ステップ数，総ステップ数の 3 つの基準に対する優先順位をテスト担当者が決定する．
5. 以上に基づいてテストケースを並び替える．

3.2.4 優先順位付けの例

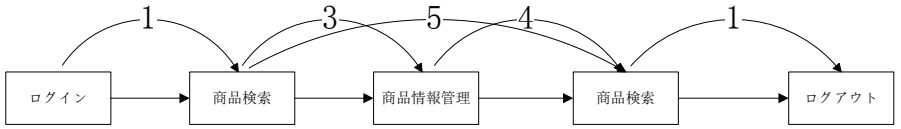
以下では，オンラインショッピングシステムの例題を用いて優先順位付けの実行例を 2 つ示す．現在保持しているテストケースは，表 1 中のものですべてであると仮定する．

まず，テストしたい機能が機能木のリーフノードに対応している場合の例として，「商品検索」をテストしたい場合を示す．優先順位付けの対象となるのは，「商品検索」と関連しているテストケース 1，2，3 のすべてである．各テストケースにおける機能実行系列の機能間の影響度は図 5 (b)および図 6 のようになるので，これらに基づいて影響度を「平均値」で評価することにする．有効ステップ数の少ないもの，総ステップ数の多いものを優先することにする．そして，機能間の影響度，有効ステップ数，総ステップ数の順番で優先順位付けを行った場合，表 2 に示す結果が得られる．

次に，テストしたい機能が機能木のリーフノード以外のノードに対応している場合の例として，「商品管理」をテストしたい場合を示す．すでに述べたとおり，親機能はその子機能を介してテストケースと関連付けされており，先の例とほぼ同様に優先順位付けを行うことができる．「商品管理」は子機能として「商品検索」と「商品情報管理」をもつので，この 2 つに関連するテストケース 1，2，3 が優先順位付けの対象となる．算出したテストケースにおける影響度は，この 2 つの機能に関する影響度に基づいて計算する．たとえばテストケース 3 は，「商品検索」と「商品情報管理」の両方を含んでいるので，図 6 (b)に示した「商品検索」に関する影響度だけではなく，図 7 に示す「商品情報管理」に関する影響度も含めて計算する必要がある．ただし，「商品検索」と「商品情報管理」の間の影響度は図 6 (b)と図 7 の両方に含まれるので，重複計算しないようにする．各テストケースにおける機能実行系列の機能間の影響度は図 5 (b)，図 6，図 7 のようになるので，これらに基づいて影響度を「合計値」で評価することにする．有効ステップが多いもの，総ステップの少ないものを優先することに



(a) テストケース 2 の影響度



(b) テストケース 3 の影響度

図 6. 「商品検索」をテストしたい場合の機能間の影響度

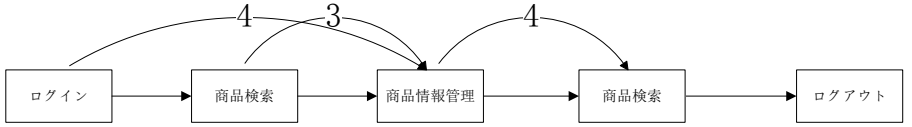


図 7. 「商品情報管理」をテストしたい場合の機能間の影響度

表 2 テストケースの優先順位（例 1）

テストしたい機能	優先順位	テストケース	影響度	有効ステップ数	総ステップ数
商品検索	1	テストケース 3	2.80	5	5
	2	テストケース 2	1.75	4	4
	3	テストケース 1	1.25	4	5

表 3 テストケースの優先順位（例 2）

テストしたい機能	優先順位	テストケース	有効ステップ数	影響度	総ステップ数
商品管理	1	テストケース 3	5	18	5
	2	テストケース 2	4	7	4
	3	テストケース 1	4	5	5

する．そして，有効ステップ数，機能間の影響度，総ステップ数の順番で優先順位付けを行った場合，表 3 に示す結果が得られる．

4. 考　察

近年，ソフトウェアテストを限られた労力の中で効果的に実施するためのひとつの方策として，テストツールを導入してテストの自動化を推進する取り組みが広く行われている．テストツールによって，特にソフトウェアの更新に伴うテスト（回帰テスト）を効率的に実施できるようになった．回帰テストでは，修正を加えた箇所が正しく修正されていることを確認するだけでなく，関連する他の機能への副作用がないことも確認しなければならない．それには，すべてのテストケースを再度実施するのが最も確実な方法であるが，コスト上困難な場合がある．たとえば，もし 1 万件のテストケースがあり，1 つのテストケースを実施する時間が 10 秒であると仮定する．すべてのテストケースを実施すればおよそ 28 時間かかってしまうので，そこまでコストをかける余裕がない場合もある．ただし，修正した部分に関するテストケースのみを実施するようにすれば現実的なコストに抑えることができる可能性がある．ここでテストケースを管理していないと，1 万件のテストケースの中から必要なものを選択するという作業を人間がやらざるを得ない．テストケースを実施するよりも時間がかかったり，作業ミスによるテストの漏れが発生する可能性が生じたりする場合があり，現実的には困難であると考えられる．しかしながら，本提案手法を用いれば自動的にテストケースを絞り込むことができる．どの程度絞り込むことができるかはソフトウェアの修正の内容に依存するが，1 万件の中から除外できるテストケースを x 件，除外できるか否かを判定するためにテストケース 1 件あたりに要する処理時間を t とすると，$x>1000t$ のときに本手法は有効であるといえる．本研究では本提案手法を支援するツールを開発中であり，今後このツールを用いて有効性を調査する予定である．

本研究の特色は，テストケースにソフトウェアの機能に関する情報を関連付けた点や，機能間の影響度やステップ数などを導入した点である．これらによって，テストケースを絞り込み，修正した機能に関係する重要なテストをすぐに実施できるようになった．機能木を利用することによって，ソフトウェア全体をまんべんなく体系的に表すことができるので，単純にキーワードを設定する方法と比較して，テストケースの検索結果に漏れや偏りが生じにくいと考えられる．また，ソフトウェアへの要求が変更された場合には，ソフトウェアの機能変更と共にテストケースも変更しなければならないことが多いが，本手法を用いれば，変更する必要のあるテストケースを検索し，要求の変更を反映することができる．したがって，テストケースを保守する上でも有用であると考えられる．

関連研究としては，既存の機能木を再利用して新たな機能木を効果的に作成するという方法に関する研究 [2] がある．我々の手法の有効性は機能木に依存しているので，このような機能木の効果的な作成方法との連携が不可欠である．もし再利用する機能木にテストケースを関連付けることができれば，テストケースの再利用もできる可能性がある．また，ブラックボックステストのためのテストケースの優先順位付けに関する研究 [3] もある．テストケースを目的や特性に基づいてグループ分けしておき，あるテストケースがソフトウェアの不具合を検出したときにそれが属するグループ全体の優先順位を上げるという方法である．我々の手法とは異なる点に着目して優先順位付けを行っているので，導入することができるかどうか，そしてどのような効果があるか検討する予定である．

5. おわりに

本論文では，機能木に基づいてテストケースを管理することによって，必要なテストケースを効果的に検索し、検索されたテストケースの優先順位付けを行う手法について提案した.従来は,キーワードを用いてテストケースを管理する方法が主であり，キーワードの設定方法によってはテストケースを適切に検索できない場合があるという問題があった．そこで本論文では，テストケースと機能木を関連付ける方法を提案した．機能木によって体系的にテスト対象ソフトウェアの機能を表現できるので，機能木中にあらわれる機能でテストケースを検索することによって必要なテストケースを適切に検索できると考えられる．さらに，機能間の影響度や有効ステップ数などを用いて，テストケースの優先順位を計算する方法も示した．これによって，検索したテストケースの数が多い場合であっても効果的にテストを実行できるようになると考えられる．

今後の課題として，機能木を効果的に作成する方法や，影響度を自動的に計測する方法，テストケースを多様な視点で優先順位付けする方法などについて検討をおこなっている．本提案手法を支援するツールの開発を進めており，実際のソフトウェア開発に試験的に適用することによって有効性を明らかにする予定である．

参考文献

1) B. Beizer, Software Testing Techniques, Van Nostrand Reinhold, 1990.
2) Hiramatsu, A.Naito, A.Ikkai, Y.Ohkawa, T.Komoda, N.Fac,“Case based function tree generator for client-server systems configuration design”, Systems, Man, and Cybernetics, Vol.3, pp.2069-2074, 1997.
3) Bo Qu Changhai Nie Baowen Xu Xiao fang Zhang,“Test Case Prioritization for Black Box Testing”,Computer Software and Applications Conference, Vol.1, pp.465-474, 2007.