第７分科会（Ｃグループ）

変更影響の可視化によるテスト効率の向上

Efficient improvement of software testing by visualizing the impact of software modifications

主査	：	保田 勝通	（つくば国際大学）
副主査	：	高橋 寿一	（ソニー株式会社）
		西　康晴	（電気通信大学）
リーダ	：	柴田 康伸	（キヤノン株式会社）
研究員	：	小渕 一幸	（三菱電機株式会社）
		田中 聡	（矢崎総業株式会社）
		芳賀 裕幸	（アンリツエンジニアリング株式会社）

（敬称略　あいうえお順）

1. 概要

近年、ソフトウェアはハードウェアの性能向上などの影響を受け、多機能化、大規模化を続けている。また、企業間競争の激化から製品のライフサイクルは短くなり、更に多機種の製品展開を余儀なくされている。そこで、短期間での多機種開発を可能にするため、共通のソースコードを用いた派生機種の開発が盛んに行われるようになった。そのため、ソフトウェアのテスト工数は増大の一途を辿っている。

本研究はターゲットを同一ソースコード（もしくは同一設計）からの派生製品開発に絞り、ソフトウェアの変更部分がどのように影響を及ぼすかを可視化することにより、テストを必要とする部分を明らかにすることで､テスト効率の向上を図る手法の研究を行った。

まず、既存の各種手法についての調査を行い、次にオブジェクト指向設計で開発されている製品と構造化手法を用いて設計されている製品の可視化手法を検討し、実施手順書を作成した。

Being affected by the improvement in hardware performance etc., software has been becoming more multifunctional and large-scaled in recent years. Moreover, the life cycle of a product is becoming short and companies are forced to develop many products due to intensified inter-enterprise competition. For this reason, development of derived models using common source codes has come to be actively conducted in order to enable multi-model development in a short period of time. Consequently, man-hours for software testing have been increasing.

In this study, we narrowed down the target to the development of derived products using the same source code (or design). We studied a method for improving test efficiency by visualizing the impact of software modifications and clarifying the part that needs to be tested.

First, we investigated various existing methods. Next, we considered visualization techniques: one for products developed using object-oriented design method; and one for products developed using structured design method.

2. 背景

近年、ソフトウェアはハードウェアの性能向上や企業間競争の激化から、多機能化、多品種化、短納期を求められている。それらの要求を満足するために、共通のソースコードと機種依存のソースコードを分離し、派生機種の開発を短期間で行う取り組みが盛んに行われている。しかし、構成管理方法が未熟であるため、派生機種間のソフトウェアの変更影響管理も十分に行われていないのが現状である。未熟な管理が原因で、テスト範囲を限定することができず、テスト工数は増大する一方である。

このような状況下でテストの現場では経験を元にしたテスト項目の絞込みを行っている。しかし、この経験に頼る方法では、テストの過不足が生じ品質リスクが増大してしまう。

そこで、ソフトウェアの変更が及ぼす影響を可視化するための表現方法とその手順を明らかにすることで、テストすべき範囲を特定し、テスト効率の向上と品質リスクの低減を図ることを目的に「変更影響の可視化によるテスト効率の向上」を研究テーマとした。

3. 活動目標

変更影響を可視化しテスト効率の向上を検証するための活動目標を以下のように定めた。

(1) 顧客要求と設計情報の両面からアプローチし、影響範囲の可視化方針を決める。
(2) QFD(Quality Function Deployment:品質機能展開)の二元表を利用した具体的な可視化の手順を決める。

4. 活動内容

派生製品開発での問題点を議論し可視化するための方策を絞り込み、過去に実施したプロジェクトの実データを基に可視化方法の検証を行った。その経過を表 4-1に示す。

表 4-1 活動経過

作業の経過	活動内容
既存手法の調査	・ TSPL（Testing a Software Product Line）：効果は期待できるが、開発プロセス全般に関わる取り組みが必要となる。 ・ 機能ツリー：機能の関連は可視化できるが、ロジックの関連を表現できない。 ・ QFD：変更箇所を可視化するには二元表が効果的である。 ・ その他の調査：CRUD 分析および市販ツールの調査を実施。
可視化方針の決定	可視化の手法として専門的なスキルが不要であり、依存関係の種類や強さが表現可能である二元表を選択した。 オブジェクト指向設計と構造化設計のプロジェクトそれぞれの設計情報を利用して二元表を用いた可視化方法の検証をする事とした。
二元表を使用した可視化の手順書作成	既存のプロジェクト設計情報をもとに二元表を作成しながら具体的な手順を明らかにして手順書を作成した。

5. 変更影響の可視化方法

5.1. 変更影響の可視化方針

再利用型ソフトウェア開発においては、変更点に着目して、ソフトウェア開発を効率化することが重要となる。テストにおいても、変更点、およびその影響に着目し、適切なテスト範囲決定や変更の種類に応じた適切なテスト方法選択が重要となる。

一般的に変更影響は、DFD（Data Flow Diagram：データフローダイアグラム）やクラス図，関数コールグラフなどの依存関係をグラフとして表現している文書情報から得ることができる。変更影響の伝播はグラフ上の変更点を起点として、辺を辿っていくことで認識できる。一方、グラフの読解には、それぞれの分野の表記方法に関する専門的なスキルが必要となる。また、グラフを直接比較し変更点を検出することは困難である。

一般に、グラフによる依存関係は表にマップすることが可能である。個々の専門的なグラフによる表現から、依存関係のみに着目した統一的な表による表現に変換する方法を定義することにより、グラフの表記方法に専門的なスキルを持たない人でも、依存関係にもとづいた変更影響を認識することが可能となる。

依存関係を表す表の一つに、QFD における二元表がある。この二元表には以下のような特徴がある。

- 表のある軸に着目して別の表に展開することにより、表をまたがる依存関係を保持しつつ様々な視点での分析が可能
- 表の各軸は階層構造を持っており、各軸項目の粒度を変えた分析が可能

ある文書にグラフ表現されている依存関係は､一つの二元表として表現することができ、文書間の依存関係の関連は、二元表を展開することで表現することができる。また、変更点は変更前後の二元表で内容が変化したセルとして認識できる。

したがって、本研究では依存関係を QFD の二元表を用いて表現し、変更前後の二元表を比較することにより、変更影響を可視化する。

5.2. 二元表における依存関係の表現

変更影響の可視化において、顧客要求をシステム内部の構造に割り付ける R-C (Requirement – Component：顧客要求-システム内部の構造)の二元表と、システム内部の依存関係を表現する C-C (Component – Component)の二元表の 2 つを基本構成と考える。これにより、顧客要求からのシステムに対する変更影響、およびシステム内部の影響の伝播を把握する。

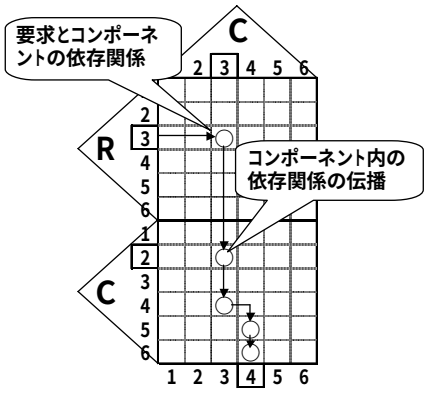

図 5-1 二元表のイメージ

単純に依存関係の有無を表現する場合には、依存関係の方向に応じて横軸から縦軸（もしくはその逆）の交点に○をつけることにより表現する。通常は対象物により依存の種類を依存記号として定義し、その強さを数値化して影響の大きさを表現することになる。図 5-1に R-C, C-C の二元表のイメージを示す。

5.3. 二元表における変更影響の特定

変更は、変更前後の二元表をセルごとに差分表示することにより認識する。空のセルから何らかの依存記号ができた場合には追加、セル中の依存記号の種類が変化した場合には変更、依存記号があるセルが空のセルになった場合は削除として認識する。

追加，変更，削除として認識されたセルから縦軸（もしくは横軸）に依存関係のある要素を辿ることにより、変更影響を認識することができるようになる。

R-C の二元表では、顧客要求の変化にもとづき、再利用システムの中から変更が必要な設計要素を見つけ出し、直接的な影響範囲を特定する。C-C の二元表では、システム内の設計要素の依存関係を再帰的に辿ることにより、間接的な影響範囲を特定する。

これにより､経験に依存しない変更影響範囲の特定手順を定義することができる｡図 5-2 に二元表による変更影響の特定方法を示す。

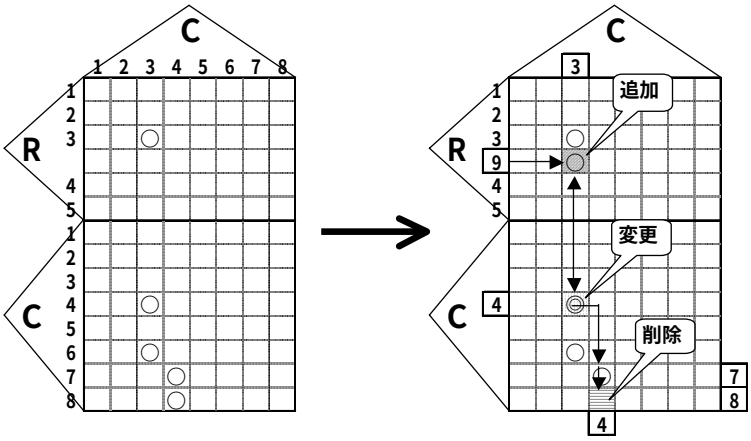

図 5-2 二元表による変更影響の特定

5.4. 二元表における依存関係の高度な表現方法

二元表を用いて影響範囲を特定する場合、もとのグラフの種類に応じた依存記号を個別に定義することで、単なる影響範囲の特定だけではなく、どのような影響を与えるかを可視化することができる。これにより、さらにテストの効率を向上することができる。

5.4.1. オブジェクト指向設計の場合

設計情報同士の関連の 1 つに、UML の「関係」がある。「関係」には表 5-1に示す種類があるため、それぞれに略号を割り当てる。表 5-1に例を示す。

要素間に複数の関係がある場合は、1 つのセルへ複数の略号を入力する。これにより、クラス間がどのような関連で結ばれているかを二元表上に表現することができる。

表 5-1 UML での関係の種類と二元表での表現方法例

関係の種類(日本語)	依存記号
Association（関連）	a
Aggregation（集約）	b
Composition（コンポジション）	c
Dependency（依存）	d
Generalization（汎化）	g
Realization（実現）	r

実プロジェクトおいて部分的に二元表による可視化を行った例を図 5-3に示す。二元表は共通部分とアプリケーション独自部分から構成され､「独自部分が共通部分に依存している範囲」,「流用元の共通部分における変更量や割合」などを全体として俯瞰することができる。

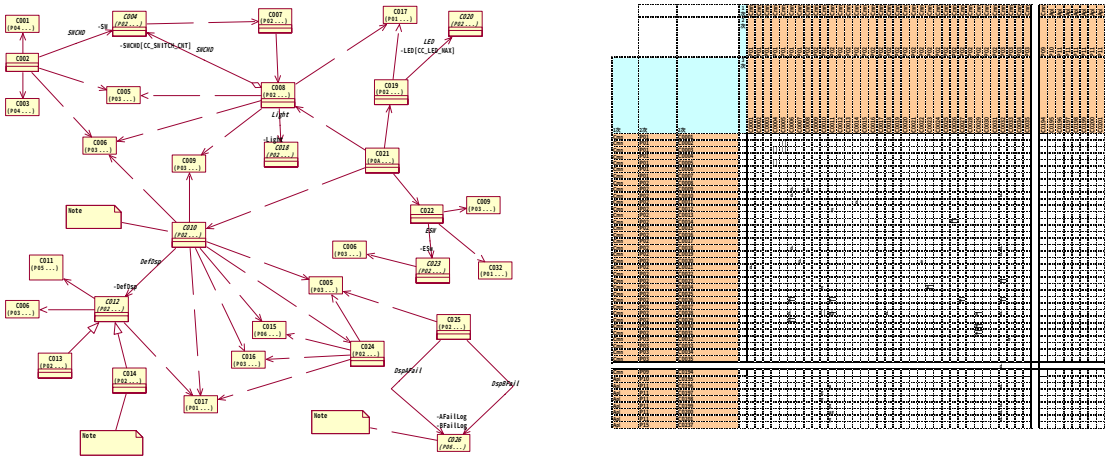

図 5-3 クラス図および C-C の二元表(抜粋)

5.4.2. 構造化設計の場合

設計情報同士の関連の 1 つに、関数と関数のコール関係がある。コール関係には関数の種類により異なる依存関係があるため、それぞれに依存記号を割り当てる。表 5-2に例を示す。これにより、関数間がどのような関連を持つかを二元表上に表現することができる。

表 5-2 関数の種類と二元表での表現方法例

関数の種類	関数名セルの状態	依存記号
グローバル関数（データ渡しなし）	セルが青色で囲まれている	○
データ渡しのあるグローバル関数	セルが青色で塗られている	◎
スタティック関数（データ渡しなし）	セルが色付けされていない	△
データ渡しのあるスタティック関数	セルが水色で塗られている	▲

実プロジェクトにおいて部分的に二元表による可視化を行った例を図 5-4に示す。二元表では関数の呼び出し関係を表し、「呼び出しが集中している関数」,「データ渡しのあるグローバル関数」など呼び出し関係と依存の種類を一覧して表現することができる。

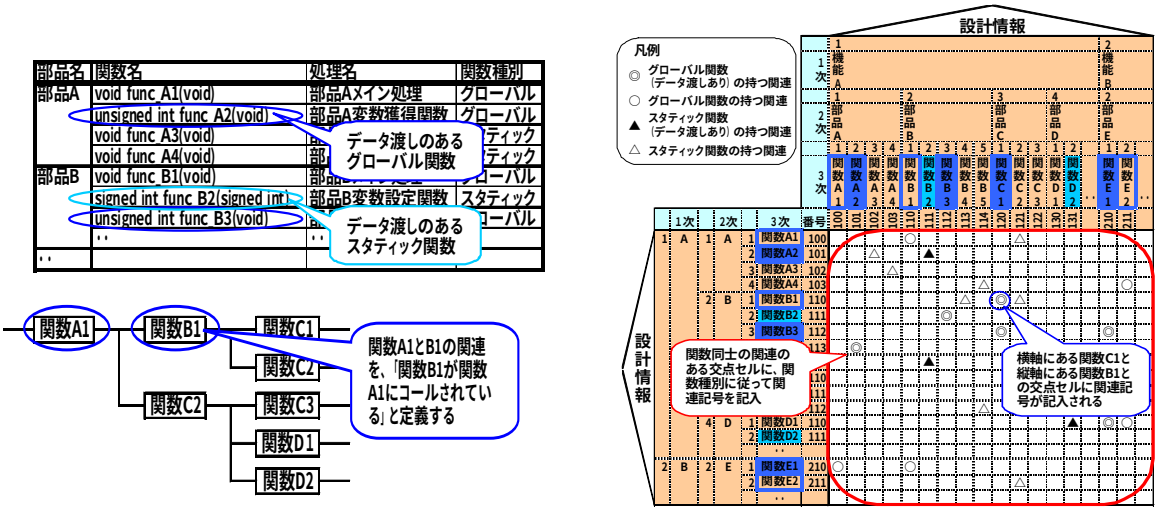

図 5-4 関数と C-C の二元表(抜粋)

6. 可視化結果の活用方法

二元表により可視化された変更影響は、以下の節で述べる考え方にもとづいてテスト効率化に利用することができる。これにより、一律な基準によるテストではなく、変更種別によるテスト方針の決定や変更影響によるテスト範囲の重みづけが可能となる。

6.1. 二元表によるテスト効率化の適用範囲

本手法は顧客要求や設計情報の関連を辿ることで影響範囲を特定しているため、関連情報が得られない場合、または不完全な場合は影響範囲の特定が困難となる。このような場合、何らかの手段による情報の補完が必要となるが、補完により変更影響の特定精度の低下や可視化のための工数の増大が発生する可能性がある。したがって本手法には二元表に含まれる関連情報による適用限界が存在する。本手法の適用範囲を表 6-1に示す。

表 6-1 二元表によるテスト効率化の適用範囲

テスト工程 / 二元表に含まれる情報	システムテスト	結合テスト	単体テスト
要求仕様，外部設計，内部設計の関連	○	○	○
要求仕様，外部設計の関連	○	△	×
外部設計同士の関連	△	○	×
外部設計と内部設計の関連	×	△	△
内部設計同士の関連	×	△	○

○=効率化可能 △=条件付で効率化可能（補足する資料を作成すれば可）×=効率化が非常に困難

6.2. 二元表によるテスト効率化の手法

6.2.1. テスト方針決定時の着眼点

以下の変更分類に着目しテスト方針を決定する。

- 追加では、機能提供側の使われ方が変化すると考えられる。したがって、依存関係を順方向に走査して、対象となる機能提供側が追加された機能利用側の要求する機能を提供できるかを検証するテスト項目を追加する。
  追加の影響範囲にあるテスト項目は、基本的にそれぞれの製品ドメインで実施していた「新規」に相当するテスト方針が適用できる。
- 変更では、機能提供側，機能利用側の双方が変化すると考えられる。したがって、依存関係を順方向、および逆方向に走査して、それぞれの変更点に対するテスト項目を追加する。
  変更の影響範囲にあるテスト項目は、「改造・修正」に相当するテスト方針が適用できる。
- 削除では、全ての機能利用側で利用する必要がなくなったと考えられる。依存関係を逆方向に走査して、削除された機能の呼び出しが不要になったことを検証する、もしくは、代替の手段が提供されているかを検証するテスト項目を追加する。
  不要や代替手段の検証は、実際にはコンポーネントの無効化設定やパッケージ移動の確認といったテスト項目となる。
- 未変更部分では、回帰テストとして用意されたテストセットを実施する。

6.2.2. テスト範囲特定時の着眼点

依存関係を順方向、および一定回数辿ることにより影響範囲を特定する。更に、関係の重みづけを利用することにより影響を数値化し、あるしきい値までを影響範囲とすることで、より定量的な影響範囲の特定が可能となる。

例えば「◎＝0.5，○＝0.3，△＝0.1」といった対応関係でしきい値が 0.1，影響の伝播が「◎→○→△→△…」となっている場合は 0.5×0.3×0.1＝0.015＜0.1 となるため、「◎→○」を影響範囲と特定することができる。図 6-1 に例を示す。

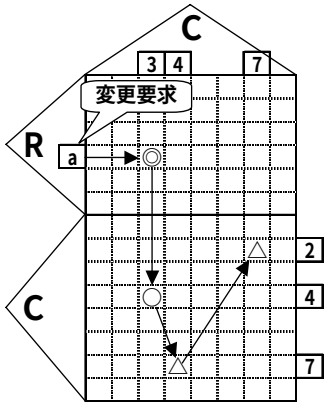

図 6-1 影響範囲の特定方法

7. 他領域への活用

本論文では変更影響の可視化によるテスト効率の向上手法について述べているが、変更影響の可視化結果はテスト工程のみではなく、その他の工程での活用も考えられる。
想定されるその他の活用方法を表 7-1に示す。

表 7-1 想定される活用方法

工程	適用項目	理由
計画	開発及びテストの規模と工数の算出 新規／再利用開発の判断	変更規模と影響範囲が明確
開発	レビュー範囲の判断	影響範囲の特定が可能
	レビュー注力箇所の特定	変更種別、依存関係が明確 階層ごとの変更数が明確
	設計の良否判断	コンポーネント階層間の依存数が明確 依存関係の結合度が明確
	テストパターンの再利用	変更影響の有無が明確
	デグレードの防止	影響範囲の特定が可能
保守	最適な修正方法の選択	影響範囲の特定が可能

8. まとめ

再利用を前提としたソフトウェア開発では顧客要求の変更がシステム全体に及ぼす影響を的確に把握し開発リソースを重点配分することが重要となる。本研究ではソフトウェア開発文書にあらわれる依存関係に着目し、QFD の二元表を用いた統一的な表現方法を定義した上で、再利用時に二元表の差分を取ることにより変更およびこの影響度を把握する手順を示した。

これにより開発文書の読解が難しい経験の浅いテスト担当者であっても変更による影響範囲を直接的または間接的に特定し、テスト項目を効率良く抽出することができる。また、本手法は開発工程における欠陥摘出作業の効率を向上させるだけに留まらず、7章に示す再利用型開発プロセスの全工程で活用することが可能である。この場合、二元表へのマッピングは個々の活用方法に応じて適切な粒度で行うことが重要である。

二元表は開発に特化した個々の表記法ほどの表現力はないがシステム全体を俯瞰できる点が利点である。本研究では、実プロジェクトに対して部分的な適用による検証しかでき

なかったため定量的な効果を示すことはできなかったが、今までの設計情報のままでは認識できなかった設計情報同士の関連によりシステム全体の変更影響を誰でも認識可能になることは確認できた。また 2 種類の開発手法による検証からオブジェクト指向設計は構造化設計に比較して曖昧性が少ないため、より二元表への展開が容易であり将来的には構成管理ツールとの連携も考えられる。

9. 参考文献

(1) 21 世紀へのソフトウェア品質保証技術：
菅野 文友, 吉澤 正(監修)：日科技連出版社：1994/09/21

(2) 実践的 QFD の活用―新しい価値の創造―：
新藤 久和 (編集), 赤尾 洋二, 吉沢 正 (著)：日科技連出版社：1998/06/30

(3) 品質展開入門：赤尾 洋二 (著)：日科技連出版社：1990/11

(4) UML モデリングのエッセンス―標準オブジェクトモデリング言語入門：
Martin Fowler (著), Kendall Scott (著), 羽生田 栄一 (翻訳)：翔泳社：2000/04/15

(5) ユースケース実践ガイド―効果的なユースケースの書き方：
Alisteir Cockburn (著), ウルシステムズ（株） (翻訳)：翔泳社：2001/11/20

(6) はじめて学ぶ UML：竹政 昭利 (著)：ナツメ社：2002/12/27

(7) 図解でわかるソフトウェア開発のすべて：
Mint（経営情報研究会）（著）：日本実業出版社：2000/07

(8) Unified Modeling Language Specification：http://www.omg.org/

(9) SESSAME－組込みソフトウェア管理者・技術者向け用語集－：
http://www.sessame.jp/

(10)品質機能展開による高品質ソフトウェアの開発手法（解説編）：
情報処理進行事業協会技術センター（編）：コンピュータエージ社：1989/03

(11)品質機能展開による高品質ソフトウェアの開発手法（活用事例編）：
情報処理進行事業協会技術センター（編）：コンピュータエージ社：1989/03

(12)2002 年ソフトウェア開発環境展専門セミナー「第三者テスト手法と実際」：
富士通株式会社 有村 雄二氏：2002/06/28

(13)属性つきゴール指向要求分析法:電子情報通信学会技術研究報告 Vol. 101, Mar. 2002:
海谷治彦, 佐伯元司, 海尻賢ニ

(14) ”An Application of QFD to System Integration Test”:
品質, Vol.30,No.1,pp.113-125 received on 2000-10-28：Takami Kihara, Charles E. Hutchinson, Dan Dimancescu, Hisakazu Shindo and Tadashi Yoshizawa

(15) ”A new approach to software design with QFD-like tabulations of UML diagrams”：10th International QFD Symposium 2004：Yoshimichi Watanabe, Yunarso Anang, Masanobu Yoshikawa, Yujiro Kasai and Hisakazu Shindo