第 5 分科会(B グループ)

テスト観点テンプレートを使用したテストケースの充実

Substantial test cases using template of test viewpoint

主査　　：奥村　有紀子（有限会社デバッグ工学研究所）
副主査　：秋山　浩一（富士ゼロックス株式会社）
副主査　：堀田　文明（有限会社デバッグ工学研究所）
リーダー：稲葉　新（日本電気株式会社）
研究員　：松尾　修（株式会社インテック）
　　尾崎　直弘（富士通九州ネットワークテクノロジーズ株式会社）

研究概要

研究員が所属する品質保証グループによる品質見極めテストで摘出されたバグの原因を分析すると，開発グループのテストケースを設計する観点の漏れに行き着くことが度々ある．本研究では，テストケースを設計するために必要な観点を記載したテスト観点テンプレートを作成し，テストケースの設計時やレビュー時に適用することで，テスト観点の漏れを防止し，テストケースの充実を図った．また，具体的な適用事例を通してその効果と今後の課題を考察した．

Abstract

When the cause of the bug extracted by the quality ascertaining test by the quality assurance group to which a researcher belongs is analyzed, the leak of a viewpoint which designs a development group's test case is sometimes reached frequently. In this paper, we indicated to prevent the leak of a test viewpoint and to be more substantial test cases, by creating the test viewpoint template and applying at the time of the design of a test case, and a review. Also, we verified the effect and future tasks through a concrete application experience.

1. はじめに

研究員の 1 人は，品質保証グループに所属し，品質見極めテストを実施している．本テストにおける 2011 年度の実績は，16 プロジェクトで 143 件のバグ検出であり，1KLOC あたりのバグ数は 0.23[件/KLOC]であった．本テストは，開発グループが実施する総合テストと同時期，もしくは後に実施されており，品質見極めテストで摘出されるバグの件数は，0 件を目標としている．理由は，品質見極めテストは，バグ出しを目的としたテストではなく，品質保証を目的としたサンプリングテストだからである．そこで，この開発グループが実施するテストを改善のための研究対象として取り上げた．文献[1]によると総合テストにおける 1KLOC あたりのバグ数の中央値は 0.3[件/KLOC]であり，これより多く出ているプロジェクトもある．さらに，前述の 143 件のバグの内容を精査した結果，約 88%に当たる 126 件が，開発グループによるテストで摘出可能な内容であることが判明した．たとえば，上限の境界値テスト，ユースケースに沿って機能を順次実行するテストによって摘出可能なバグであった．これらのバグは，発生条件が合えば 100%摘出できることや，テストケースの作成やテストの実施も容易にできるものであることから，開発グループが実施するテストで漏れを防止すべきであると判断した．改善として現在行っている設計作業でバグを作りこまない未然防止策の他に，テストの充実も必要であると考えた．テストの充実には，不足していると考えられる観点を抽出し，それにひもづくテストケースを追加することが必要である．このためには，テスト対象に必要なテスト観点を抽出できる仕掛けが必要と考え，その手段を本グループで検討した．その結果，テスト対象の種類やドメインによらず使用可能なテスト観点を列挙したテスト観点テンプレートをあらかじめ作成しておけば，

具体的なテスト対象に適用すべきテスト観点の漏れを防止できると考えた．

本論文の構成は以下の通りである．2 章では課題を定義する．その課題を解決する手段を 3 章と 4 章で述べ，テスト対象への適用事例を 5 章で説明する．6 章では適用結果と考察を述べる．7 章は結論と今後の課題である．

2． 課題

本章では開発グループが実施するテストケースに漏れが発生する原因について述べる．ここでは，研究員が所属する企業のうち，ミドルウェア製品を扱う A 社と新聞広告の情報を管理するシステムを扱う B 社が抱える課題について分析した．

A 社では，開発グループが作成したテストケースを品質保証グループがレビューしている．レビューの対象としているテストケースは，コンポーネントテスト[2]のレベルで開発グループが実施する機能テストである．レビューでは，品質保証グループが保有するチェックリストを使用している．このチェックリストは，過去の品質見極めテストで摘出したバグから,テストケースを設計する際に使用するテスト観点の弱点を抽出したものである．表 1 にサンプルを示す．

表 1 チェックリストのサンプル

大項目	中項目	内容
インストール	正常系	アンインストール後にインストール先フォルダと関連レジストリは削除されているか
		インストール先やログ出力先などのデフォルトディレクトリよりディレクトリ先を変更した評価(その他ドライブ先(C以外のドライブ)の変更、インストールパスのフォルダ名に全角やブランクが入った名前などの評価も考慮する)
		前バージョンからの上書きインストールが問題ないか評価する。(上書きインストール時の問題は多い)
		インストール中にキャンセルボタンを押した場合、正しく終了するか。その後、正しくインストールできるか。
	正常系	設定フィールドに対する半角/全角、特殊記号、空文字、半角ｶﾅを意識した評価。(特に半角、全角の同一名を入力した場合によく不具合が見られる)
		全てのラジオボタン(単一)、チェックボックス(複数)が選択できるか。
		プルダウンで表示される内容に対して、長い文字列(特に半角英数字のみで構成された文字列)を入力してある場合に文字切れ、表示切れが無いか確認する
		10進数の数値入力フィールドに「012」を入力する。⇒8進、16進の入力が出来てしまう場合がる。
		ソート(昇順、降順)の処理に問題ないか。
		検索されるはずの無い項目が検索されたり、逆に検索されるべき項目が検索されない現象が発生しないか確認。
		テキストボックスへの入力時にテンキー(Enterも含む)の入力を受け付けるか。
		グラフなどを複数同時に表示させても、画面が固まらないか。
		「追加」「削除」を繰り返した時、動作および画面の表示に不整合が発生しないか確認。

これまで，品質保証グループが実施してきたレビューでは，追加テストやバグ摘出につなげるような効果が高い指摘ができていない．原因として以下の問題点が挙げられる．

(1) チェックリストを使用しても，実際の指摘は担当者の経験やスキルに依存している．
(2) チェックリストは，過去のバグから得られた弱点のリストであるので，テストケースを設計する際に盛り込むべき観点を網羅していない．

(1)の具体例として，チェックリストの記載内容のまま担当製品に適用できない場合，チェック項目の意図を読み替える行為が必要になることや，チェックリストの記載内容だけが確認対象となってしまい，記載されている以外の観点ではチェックされないことが挙げられる．

B 社の開発グループの結合テストは，テストケースを作成するとき，機能仕様書，過去のテストケース，および過去のフィールドバグをテストベースとしている．しかし，これらのテストベースのみでは，テストケースを設計する観点に漏れが発生するため，結合テストの後になってもバグが多発している．この理由として現状のテストケース設計には，以下の問題点が挙げられる．

(1) 仕様作成者自身がテスト設計を担当することが多く，テストケースを作成するときに「テスト観点」に着眼するということが定着しておらず，機能仕様書からテストケースを作成する際に必要な項目が漏れてしまう．

(2) テストベースとなる機能仕様書に，組合せや順序，非機能要件の記載が不十分である．

A 社と B 社の両方に共通する課題は「テストケースを設計・レビューする観点の漏れ」である．A 社のチェックリスト，B 社の機能仕様書や過去のテストケースを入力文書とするだけでは，テストケースを設計するための観点を抽出しきれていない．そのため，A 社では今までテストケースのレビューで指摘できなかったテスト観点を指摘できるようにすることを目的とし，B 社では今まで挙げることができなかったテスト観点を挙げるようにすることを目的とし，テストケースを設計する観点の発想を促すことで，テスト観点を充実させる手段が必要と考えた．

また，1 章で述べた通り，A 社の品質見極めテストで摘出したバグの約 88%は，開発グループによるテストで摘出可能と判断した一方，残り約 12%はテストの改善だけでは摘出できない内容である．たとえば，再現率が 100%ではない障害がある．この類は設計工程でバグを作り込まないような対策の方向性になるので，本研究の対象外とする．

3. テスト観点抽出方法の検討

テストケース設計のために観点を漏れなく抽出する記法として，NGT(Notation for Generic Testing)[3]がある．NGT は，テストケースを設計する観点をツリー状に階層化しながら観点同士の関連を検討して，テストの構造を明らかにして最適化を図る記法である．また，観点からテストケースを設計する手法としては，ゆもつよメソッド[4]がある．これは，テストタイプ(機能)とカテゴリ(観点)からテストケースを設計するものである．本研究では，テストケースを設計・レビューする観点を充実させるため，まず観点の抽出漏れを防ぐことを目的としている．そのため，これ以上分解できない細かい観点の整理や，観点を階層的に可視化することに優れていることから NGT を採用した．NGT を使った観点の展開方法を以下に示す．

Step1 字句解析
設計書などのテストベースから機能名などの字句を拾い上げ，テストケースを設計する観点として意味が分かるように記述する．

Step2 簡単なまとめ
入力に関する観点，出力に関する観点，保守に関する観点，顧客操作に関する観点，装置に関する観点，その他の処理条件に分けて大まかに整理する．

Step3 品質特性(機能性，信頼性，使用性，効率性，保守性，移植性)のあぶり出し
テスト対象で備えるべき品質特性からテストケースに必要な観点を抽出する．

Step4 対称性を考慮した整理・補完
入力と出力，通常と例外，正常系と異常系などを考えてテストケースを設計する観点に漏れが無いか確認する．

Step5 経験による補完
過去のバグなどから得られた観点が盛り込まれているか確認する．たとえば，装置側の期待と人の行動との差，データ・操作順番のバリエーションがある．

また，観点を展開する際には下記の点に留意する．

(1) ツリーの一番下は，それ以上分解する必要がない項目にする．
(2) シナリオ(5W1H)を考えると想定外のお客様の行動を抽出することができる．
(3) 構造モデルや動的な振る舞いを示したモデルがあると必要な観点を明確にしやすい．

4. テスト観点テンプレートの作成

テストケースを設計・レビューする観点は，抽象化すればテスト対象のドメインや種類

によらず共用できるので，部門内の複数のプロジェクトで使えるように「テスト観点テンプレート」を作成する．テスト観点テンプレートとは，観点を具体的なテスト観点抽出に利用しやすいように階層化したものである．図 1 に，テスト観点テンプレートの作成イメージを示す．

最初にテスト観点テンプレート上の第一層にあたる大観点を設定する．大観点とは，テスト対象をシステム全体の範囲で俯瞰した時に，影響がある因子を漏れ，重複なく挙げたものである．この大観点を起点として，階層的に詳細化していく．テスト対象を単に実行させるだけでは，単一機能・正常系でプログラムを 1 回動作させただけであり，その対象があらゆる場面で想定どおり正しく動くことを確認したことにならない．したがって，ここに条件の組合せやバリエーションを追加していく必要がある．その例として，データ形式，負荷の状態，ハードウェア/ソフトウェア環境がある．

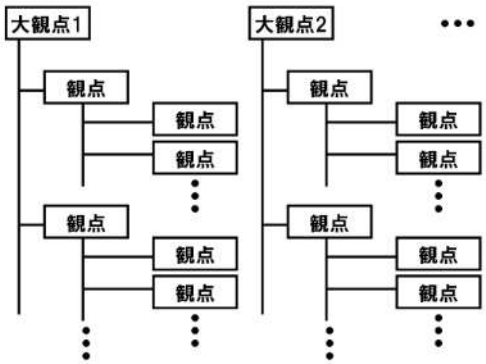


図 1 テスト観点テンプレートの作成イメージ

これらを組み合わせることでテストのバリエーションを増やしていく．そこで，バリエーションを考えるときに，大観点で漏れがないように因子を以下のように考えた．図 2 に，これらの関連を整理した結果を示す．

- 何をテストするのか？…テスト対象 ⇒ 機能
- 何を入力するのか？…入力 ⇒ データ
- どこから入力するのか？…入力媒体 ⇒ GUI
- どんな状態でテストするのか？…読み書きする変数 ⇒ 状態
- どんな環境でテストするのか？…システム構成の変数 ⇒ 環境
- どのような手順や操作でテスト対象を実行するのか？…テスト対象を含めた一連の操作 ⇒ 運用
- 非機能要件を確認するために必要な一定の入力と状態でテスト対象を実行 ⇒ 非機能要件

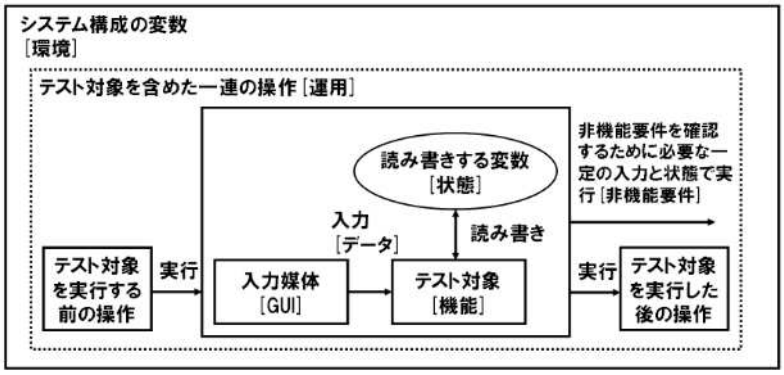


図 2 テスト対象の大観点同士の関連

ここで図中の“ [xx] ”が大観点であり，テスト観点テンプレートの大観点を以下の 6 つに設定した．

1. 機能は，テスト対象機能のことである．たとえば，検索機能なら前方一致/後方一致など検索サブ機能の観点を入れる．メール機能ならプロトコルを意識した観点を入れ

る．インストーラの機能ならアンインストールの観点を入れるなどして，各機能についてテストが必要と思われる観点をまとめた．

2.データ・状態は，テスト対象機能に与える入力と動作中に読み書きする変数である．たとえば，境界値や NULL 値などがある．また，文字入力する箇所については，文字コード，改行文字，2 バイト文字などがある．ここで，正常ケースと異常ケースの対称性を意識すると，より異常系の観点を増やすことができる．

3.GUI は，データの入力媒体である．たとえば，入力チェック，画面遷移，画面制御などがある．また，ボタンやメニューの動作確認だけでなく，他画面との統一性についての観点もある．

4.環境は，お客様が構築するシステム構成の変数である．たとえば，ハードウェア環境では，サーバ台数，スペックなどがある．ソフトウェア環境では，OS，ブラウザの種類，Version などがある．構成については，クラスタ構成，DMZ(DeMelitarized Zone)構成などがある．

5.運用は，テスト対象を有効系のみ実行するのではなく，その前後に実行する操作やエラーとなるケースも考えたものである．たとえば，障害発生時，データ移行時，システム監視をする場面，再起動，ログの確認などがある．実際にお客様が運用する時に起こりうるイベントを意識すると，より観点を増やすことができる．

6.非機能要件は，ソフトウェア製品の品質モデル[2]で定義される要件のうち，機能面以外のことである．たとえば，効率性では性能，信頼性では高負荷などがある．

大観点を設定したら，テストケースを設計するために必要な観点を以下の手順で整理し，テスト観点テンプレートを作成した(作成したテスト観点テンプレートは付録 1－表 3 参照).テスト観点テンプレートの記法は,3 章の NGT を参考にして階層化された構造とした．

(1) 今まで使用していたテストケースを設計する観点を収集する．たとえば，既存のチェックリストや過去のフィールドバグから得られた知見(エラー処理後の動作など)がある．

(2) 大観点だけでは抽象度が高いので，大観点から更に細かく分類できるように階層構造で，関連するテストケースを設計する観点を設定する．観点の設定のしかたは 3 章の観点の展開方法を参考にした．階層化しながらテストケースを設計する観点を具体化していき，観点漏れの検証ができるようにする．ただし，階層を深くしてテスト観点テンプレートに挙げた観点が，特定の製品しか適用できないような細かさになると，従来のチェックリストと変わらなくなるため，階層は 3 から 4 として粒度にも留意する．

(3) 異なる大観点に属する観点間で同じような意味の観点が出てきているのか，関連性を検証する．必要であれば，属する大観点を変更する．

(4) 文献[5][6]から不足していると判断した観点を追記する．今回は，39 件の記載事例を調査して，27 件の事例を参考にして不足観点を追加した．たとえば，料金などの数値計算のロジック誤り，お客様の本番運用環境と運用前のテスト環境との環境差異，担当製品と連携する他システムの障害時の動作があった．

(5) 自分の知識や経験から，不足していると判断した観点を追記する．

(6) 複数のテスト対象で抽出したテスト観点を，整理・統合し抽象化する．

5. テスト対象への適用

4 章で述べたテスト観点テンプレートをテストケースの作成やレビューに適用することを考える．しかし，テスト観点テンプレートをそのまま適用することはできない．理由は以下の通りである．

(1) テスト観点テンプレートは，テスト対象を限定しないで観点を挙げているため，個

別のテスト対象に適用する場合，全ての観点がテストケースに盛り込まれているかどうか確認することは現実的でない．

(2) プロジェクトによって必要な観点が異なることから，観点を選別する作業が必要になる．

(3) 自分の知識や経験からテスト対象に使われる表現に変換する作業や，製品独特の機能をテスト対象にする場合は，観点を追加する必要も出てくる．

そこで，プロジェクトのテスト対象やテストの目的に合ったテスト観点を整理する作業にマインドマップを使用する．マインドマップの作成方法は，一般的な記法(例：右上から時計回りに書いていく)[7]に準ずるものとし，以下の手順で作成した(マインドマップを作成する手段については付録 2－表 4 参照)．

(1) 最初にテスト対象を中央に記載する．

(2) テスト対象の周りにテスト観点テンプレートの大観点を記載する．「機能」をテスト対象の右上に記載し，以降，時計回りに「データ・状態」「GUI」「環境」「運用」「非機能要件」を記載する．

(3) 記載した大観点とテスト対象を線でつなげる．

(4) 「機能」の大観点について，テスト観点テンプレートの「機能」の下の階層にある観点中に，本製品に関係する観点があればマインドマップの「機能」の右上に記載する．関連する観点がなければ，作成する，更新する，など「○○する」と具体的な内容にして「機能」の観点から線をつなげる．このように具体的な処理を記述することで，次の観点を生成する発想を得やすくすることができる．

(5) テスト観点テンプレートの(4)で挙げた観点について，その下の階層にある観点から関連する観点があれば，観点を外側に記載して線をつなげる．関連する観点がなければ，「何を」「どうやって」という視点や，目的，制限，分岐条件などの視点で観点を挙げる．また，単数/複数，最大/最小，一部/全部，正常時/異常時など，対称性を意識すると観点として挙げやすい．

(6) 他の大観点の軸についても，(4)(5)と同様にテスト観点テンプレートから関連する観点を外側に記載して線をつなげる．特に「機能」「データ・状態」「GUI」は，テストケースを設計する際，複合条件として密接に関連することがあるので，関連性に漏れが無いかどうか確認し関連があれば，テストケース設計者が見落とさないようにメモとして残す．また，同じ観点が複数の場所に出てくる場合は，番号を付けて関連があることを示す．図 3 に，マインドマップを作成し，その中にメモを残す例を示す．

(7) マインドマップを作成していくと，ある観点を挙げると「○○の場合はどうなるのだろう？」という発想から別の観点が生まれることもあり，その観点もマインドマップ上に記載する．

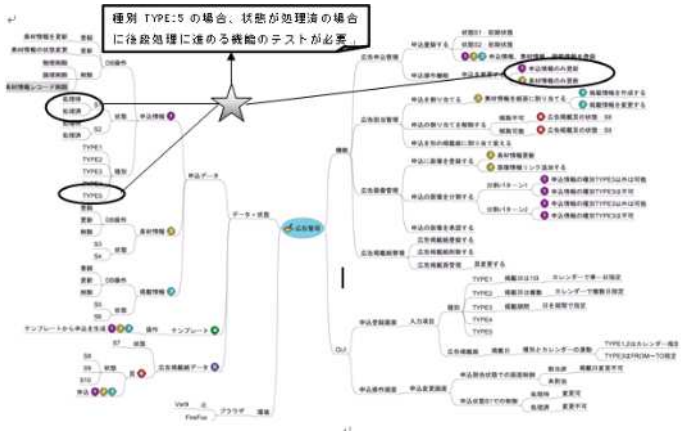


図 3 作成したマインドマップの例

このマインドマップに挙げられた観点で，テスト対象のテストケースを作成する，またはテストケースをレビューする．従来のテストベースに無い観点を探し，その観点でテスト対象のテストケースを作成することやレビューすることで，従来のテストベースには無い観点のテストケースの漏れを防ぐことや，レビューで指摘することができる．

6． 適用結果と考察

本章では開発中に適用した結果の事例を述べる．A 社の事例は，開発グループが作成したテストケースを品質保証グループがレビューしたものであり，機能テスト 1 と 2 は異なる機能である．B 社の結合テスト 1 は，テストケースを作成する際に適用した事例，結合テスト 2 は，開発グループ内で作成したテストケースを筆者がレビューした事例である．表 2 に適用結果を示す．

表 2 適用結果

会社	プロジェクト	テストレベル	テストケース	指摘数(うちテストケースの追加につなげた指摘数)	指摘により追加されたテストケース数	追加テストケースによるバグ摘出数
A 社	A-1	コンポーネントテスト	機能テスト 1	39(12)	114	2
A 社	A-1	コンポーネントテスト	機能テスト 2	16(1)	6	0
B 社	B-1	統合テスト	結合テスト 1	対象外※	60	5
B 社	B-2	統合テスト	結合テスト 2	9(9)	85	4

※テストケース作成時に適用したため，レビューによる指摘は無い．

A 社の機能テスト 1 で摘出された 2 件のバグの内容は以下である．

(1) 契約先組織名を 256 文字以上にするとソート機能が動作しない．
(2) 再契約の際，枝番(XX-Y の Y の部分)をカウントアップして登録を行う処理が正しく動作していない．

(1)は，上限値を確認する対象を抽出できたことにより，テストケースを追加することができた．(2)は，解約後の動作という観点のテストケースが抜けているという指摘をすることで，摘出できたバグである．また，A 社の機能テスト 2 では，バグを摘出することはできなかったが，機能実行時の失敗後，正しい設定で正常終了するかどうかのテストケースを追加することができた．これは，状態遷移で考慮すべき観点をマインドマップ上で抽出できたことによる．しかし，A 社の機能テスト 1 に比べると，指摘数や追加テストケースが少なかった．理由としては，ある条件についてのテストケースを設計する観点が不足という指摘が，テスト対象が実行不可能な内容であり追加テストケースにつながらなかったことと，開発グループも A 社の機能テスト 1 での指摘を受けて，テストケースを設計するスキルが向上し，機能テスト 2 のテストケース設計時に生かしたことが考えられる．前者の理由は，マインドマップから発想を得られた観点を指摘に挙げる時に，テスト対象に必要な観点かどうか選別するスキルが不足していたと考える．A-1 プロジェクトは新規開発であり，製品の理解度が低い開発メンバーが多かったため，マインドマップ上で観点を挙げるだけで，テストケースの漏れに気付くこともあった．

B 社の結合テストで摘出されたバグの例を 2 件示す．

(1) ある種別のデータは，特定の指示を行うことで後処理可能な状態に遷移できる機能について，代替フローが無いまま削除されていた．
(2) 2 台の端末で異なる表示／権限モードで起動した状態で，同じデータにアクセスしたときに同時に更新できる状態になっていた．

(1)は，図 3 に示すマインドマップ上で種別 TYPE:5 の場合，状態が処理済の場合に後段処理に進める機能のテストケースが必要と気づいたことによる．(2)は，機能の組合せによる状態制御のテストケースが抜けている指摘に対するテストケースの追加で発生したバグである．B 社の事例では，上級者がテストケースを作成したので，マインドマップで観点を挙げるだけでなく，観点同士を組み合わせたテストケースを追加することができた．

このように，いずれの事例も本研究の手法を使用することでテストケースの追加につなげることができた．更には，バグ摘出につなげた事例もあった．これより，今まで漏れていたテストケースを設計する観点を充実させることで，これまで摘出できなかったバグの摘出につながり，開発グループのテストで品質向上できることを示した．

7. おわりに

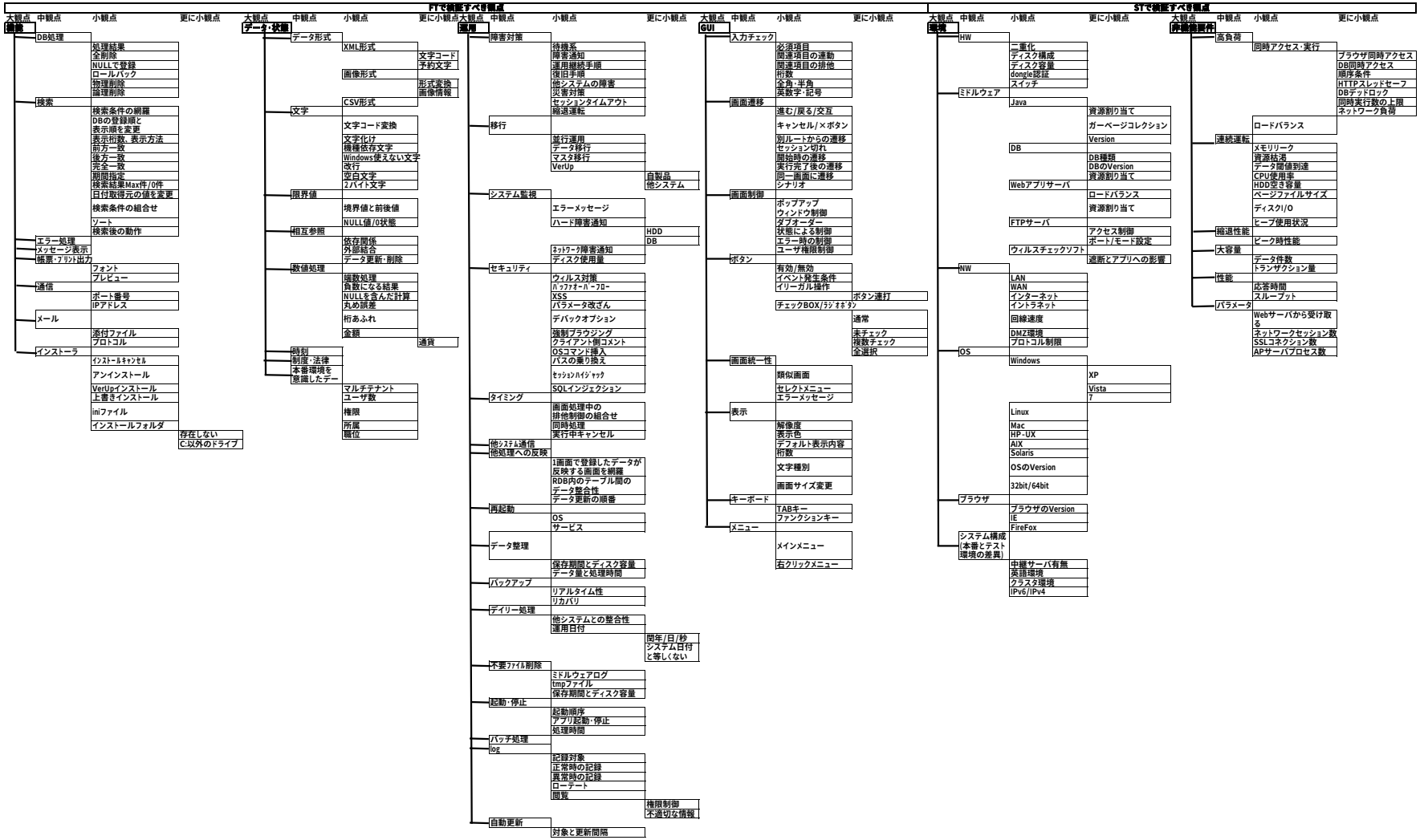
本論文では，テストケースの漏れについて課題認識を持ち，テストケースを設計する観点を充実させるために，テスト観点テンプレートを作成し，テストケースに適用した．テスト観点テンプレートを適用することにより,テストケースを設計する観点の発想を促し，適用前よりテストケースを追加できたことを示した．また，追加したテストケースでバグを摘出し，品質向上に効果があることを示した．

また，今後の課題としては，以下が考えられる．

(1) 今回の適用範囲は，研究員が担当しているプロジェクトであったが，今後は，研究員が所属する部門に展開して適用できるようテスト観点テンプレートを運用するプロセスを検討する．
(2) 今回は，テストケースのレビューにテスト観点テンプレートを適用したが，今後は，仕様書のレビューにも適用し，上流工程でのバグ作り込みを防ぐことができるか検討する．

8. 参考文献

[1] ソフトウェア開発データ白書 2012-2013，独立行政法人情報処理推進機構(IPA)
[2] ソフトウェア品質知識体系ガイド(SQuBOK)，オーム社
[3] 西 康晴，テスト観点に基づくテスト開発方法論(VSTeP)の概要，http://blues.se.uec.ac.jp/~nishi/VSTeP121005.pdf
[4] 清水 剛史，バグの流出防止を考える -どんなテストをすればバグを見つけられたのか？-，2011 年度 SQiP 研究会 第 5 分科会 A グループ
[5] 高信頼化ソフトウェアのための開発手法ガイドブック，独立行政法人情報処理推進機構(IPA)
[6] ソフトウェアテスト PRESS Vol.3，技術評論社
[7] 池田 暁・鈴木 三紀夫，マインドマップから始めるソフトウェアテスト，技術評論社


FITで検証すべき観点
STで検証すべき観点
大観点
中観点
小観点
更に小観点
画像
DB処理
処理結果
全削除
NULLで登録
ロールバック
物理削除
論理削除
検索
検索条件の網羅
DBの登録順と
表示順を変更
表示桁数、表示方法
前方一致
後方一致
完全一致
期間指定
検索結果Max件/0件
日付取得元の値を変更
検索条件の組合せ
ソート
検索後の動作
エラー処理
メッセージ表示
帳票・ﾌﾟﾘﾝﾄ出力
フォント
プレビュー
通信
ポート番号
IPアドレス
メール
添付ファイル
プロトコル
インストーラ
ｲﾝｽﾄｰﾙｷｬﾝｾﾙ
アンインストール
VerUpインストール
上書きインストール
iniファイル
インストールフォルダ
存在しない
C:以外のドライブ
データ・状態
データ形式
XML形式
文字コード
予約文字
画像形式
形式変換
画像情報
CSV形式
文字
文字コード変換
文字化け
機種依存文字
Windows使えない文字
改行
空白文字
2バイト文字
限界値
境界値と前後値
NULL値/0状態
相互参照
依存関係
外部結合
データ更新・削除
数値処理
端数処理
負数になる結果
NULLを含んだ計算
丸め誤差
桁あふれ
金額
通貨
時刻
制度・法律
本番環境を
意識したデー
マルチテナント
ユーザ数
権限
所属
職位
運用
障害対策
待機系
障害通知
運用継続手順
復旧手順
他システムの障害
災害対策
セッションタイムアウト
縮退運転
移行
並行運用
データ移行
マスタ移行
VerUp
自製品
他システム
システム監視
エラーメッセージ
ハード障害通知
HDD
DB
ﾈｯﾄﾜｰｸ障害通知
ディスク使用量
セキュリティ
ウィルス対策
ﾊﾞｯﾌｧｵｰﾊﾞｰﾌﾛｰ
XSS
パラメータ改ざん
デバックオプション
強制ブラウジング
クライアント側コメント
OSコマンド挿入
パスの乗り換え
ｾｯｼｮﾝﾊｲｼﾞｬｯｸ
SQLインジェクション
タイミング
画面処理中の
排他制御の組合せ
同時処理
実行中キャンセル
他ｼｽﾃﾑ通信
他処理への反映
1画面で登録したデータが
反映する画面を網羅
RDB内のテーブル間の
データ整合性
データ更新の順番
再起動
OS
サービス
データ整理
保存期間とディスク容量
データ量と処理時間
バックアップ
リアルタイム性
リカバリ
デイリー処理
他システムとの整合性
運用日付
関年/日/秒
システム日付
と等しくない
不要ﾌｧｲﾙ削除
ミドルウェアログ
tmpファイル
保存期間とディスク容量
起動・停止
起動順序
アプリ起動・停止
処理時間
バッチ処理
log
記録対象
正常時の記録
異常時の記録
ローテート
閲覧
権限制御
不適切な情報
自動更新
対象と更新間隔
GUI
入力チェック
必須項目
関連項目の連動
関連項目の排他
桁数
全角・半角
英数字・記号
画面遷移
進む/戻る/交互
キャンセル/×ボタン
別ルートからの遷移
セッション切れ
開始時の遷移
実行完了後の遷移
同一画面に遷移
シナリオ
画面制御
ポップアップ
ウィンドウ制御
ダブオーダー
状態による制御
エラー時の制御
ユーザ権限制御
ボタン
有効/無効
イベント発生条件
イリーガル操作
ボタン連打
チェックBOX/ﾗｼﾞｵﾎﾞﾀﾝ
通常
未チェック
複数チェック
全選択
画面統一性
類似画面
セレクトメニュー
エラーメッセージ
表示
解像度
表示色
デフォルト表示内容
桁数
文字種別
画面サイズ変更
キーボード
TABキー
ファンクションキー
メニュー
メインメニュー
右クリックメニュー
環境
HW
二重化
ディスク構成
ディスク容量
dongle認証
スイッチ
ミドルウェア
Java
資源割り当て
ガーベージコレクション
Version
DB
DB種類
DBのVersion
資源割り当て
Webアプリサーバ
ロードバランス
資源割り当て
FTPサーバ
アクセス制御
ポート/モード設定
ウィルスチェックソフト
遮断とアプリへの影響
NW
LAN
WAN
インターネット
イントラネット
回線速度
DMZ環境
プロトコル制限
OS
Windows
XP
Vista
7
Linux
Mac
HP-UX
AIX
Solaris
OSのVersion
32bit/64bit
ブラウザ
ブラウザのVersion
IE
FireFox
システム構成
(本番とテスト
環境の差異)
中継サーバ有無
英語環境
クラスタ環境
IPv6/IPv4
非機能要件
高負荷
同時アクセス・実行
ブラウザ同時アクセス
DB同時アクセス
順序条件
HTTPスレッドセーフ
DBデッドロック
同時実行数の上限
ネットワーク負荷
ロードバランス
連続運転
メモリリーク
資源枯渇
データ閾値到達
CPU使用率
HDD空き容量
ページファイルサイズ
ディスクI/O
ヒープ使用状況
縮退性能
ピーク時性能
大容量
データ件数
トランザクション量
性能
応答時間
スループット
パラメータ
Webサーバから受け取る
ネットワークセッション数
SSLコネクション数
APサーバプロセス数

付録 2

マインドマップを作成する手段は，フリーハンド，Excel，マインドマップ作成ツールがある．今回においては，マインドマップ作成ツールとして FreeMind を使用した．表 4 に，それぞれのメリットとデメリットを示す．

表 4 マインドマップ作成手段の比較

手段	発想の効果	準備時間	操作性	再利用性
フリーハンド	○:制約無し	○:すぐに着手可	×:修正困難	×:編集不可
Excel	×:整形が必要	○:すぐに着手可	×:整形が必要	○:編集可能
マインドマップ作成ツール	○:制約無し	×:操作法の学習が必要	○:機能が豊富	○:編集可能

A 社では，発想を得た観点を記載する上で制約が無く，すぐにマインドマップを作成できることから，まずはフリーハンドで模造紙に記載した．その後，マインドマップに記載した観点を開発グループと共有するために，Excel に書き直して開発グループに展開した．このように，表 4 の特徴を理解した上で，場面に応じて適切な手段を適用する．