テストケースの自動生成を見据えた 基本設計フォーマット作成アプローチの提案

Proposal of approach to create format of basic design document focused on automatic generation of test cases

2017 年度ソフトウェア品質管理研究会 研究コース 6 TRY チーム
2017 Software Quality Profession Research Society, Research Corse 6, TRY Team
○中川　穂[1)]　奥村　慎[2)]　酒井　雄太[3)]　松平　智晶[4)]
○Mizuho Nakagawa[1)]　Makoto Okumura[2)]　Yuta Sakai[3)]　Chiaki Matsudaira[4)]

Abstract

In software system development, it is not just necessary to make clear the target of the development in the design document and the specification document by strictly controlled descriptions in level of abstraction, but preferable to make the document well-organized according to the varied necessary information in the post-process. Furthermore, it makes the development process more efficient if automated processes, for example automatic generation of test specifications from the functional specification, were realized.

In this paper, we aim to achieve these problems through the trials, on the one hand by structuring the specification and on the other hand by restricting the method. Especially, we suggest the approach to generate an document format for the basic design which has the minimum restrictions enough to automate, flexibility to describe specifications, and usefulness for the post-process by discussing about the method which has flexibility in the contents and description of the specification. Specifically, we aim the well-balanced approach between the strictness and the flexibility of the description by examining generated automatically test cases.

1. はじめに

システム開発において，仕様は，要求を獲得，分析し，ステークホルダと合意形成した結果，何を作るのかを表す．そして，システム開発では仕様を記述した仕様書や，仕様の実現方法について記述した設計書などの文書が作成される．要件定義工程より後の工程においては，仕様に基づき設計や実装，テスト，運用などを行うため，仕様書や設計書は利用者，設計者，実装者，テスト設計者，運用者などのコミュニケーションのハブとなる重要な情報や文書となる．

仕様は合意形成の結果であり，様々な役割を持つ人々が誤解なく理解できる必要がある．そのため，読み手の違いによって複数の解釈が生まれる可能性のある曖昧な記述をなくし，厳密な記述を行うことが必要となる．また，「何を作るのか」を表す仕様は，「どう作るのか」に踏み込みすぎて設計や実装に制約を

2017 年度ソフトウェア品質管理研究会 研究コース 6 TRY チーム
2017 Software Quality Profession Research Society, Research Corse 6, TRY Team
1) 株式会社インテック
INTEC Inc.
東京都江東区新砂 1-3-3　Tel: 03-5665-5098 e-mail: nakagawa_mizuho@intec.co.jp
Shinsuna, 1-3-3, Koto, Tokyo, Japan
2) アイエックス・ナレッジ株式会社
IX Knowledge Inc.
3) wells system design
4) 日本電気航空宇宙システム株式会社
NEC Aerospace Systems,Ltd.
【キーワード：】基本設計書，テスト自動化技法，要求仕様化，Web システム

与えることをしないように，具体的すぎず，と同時に抽象的すぎないものである必要もある．

前述の通り，仕様書や設計書は様々な役割を持つ人々の作業へのインプット情報となるため，それぞれの工程において，参照しやすく，活用しやすい情報として整理されていることが望ましい．テストの工程においては，仕様の内容に対して過不足のないテスト設計を行う必要があり，仕様やその変更に対するトレーサビリティが重要となる．

また，設計書の一部から，テスト仕様を機械的に自動生成することができれば，開発の効率化を行うことができる．自動生成には，抽象度を制御した厳密な仕様を書く必要があり，仕様を構造化したり，記述言語を制約したりする方法がある．しかし，開発ドメインに依存しない汎用的なものはなく，また，制約しすぎると，これから開発しようとするシステムの仕様の自由度を奪ってしまう可能性がある．

そこで，著者らは，仕様の構造化や記述の制約と，内容の自由度とのバランスを取ることが可能かどうかを深く検討するために，開発ドメインや開発対象ごとに，仕様の構造や記述方法を，反復的に検討しながら，仕様の内容や記述の自由度を維持しつつ，主に設計者とテスト設計者にとって有用な，仕様の一部をテスト仕様に自動変換しやすいアプローチの提案を目指して，検討と検証を行った．

具体的には，システム開発における基本設計書を対象として，設計書フォーマットの書きやすさや読みやすさ，テストケースへの自動変換，仕様記述の自由度について実験，考察した．

以下に本論文の構成を示す．第2章では本研究の背景と提案アプローチを述べる．第3章では本アプローチを用いて作成した基本設計書フォーマットと，自動化により作成されるテストケースを示す．第4章では作成したフォーマットが，人が記述することや基本設計書のフォーマットとして妥当であるかを確認するために行った実験とその結果を示す．第5章では本研究を通して得られた考察を述べる．第6章では今後の展望について述べる．

2. 基本設計書に関する問題点と解決するためのアプローチの提案

2.1 対象システム

本研究が焦点を当てる課題は，基本設計書の抽象度や厳密さが書き手によって変化することが，テストなどの後工程での仕様理解の妨げになるということである．

本研究では，具体的な例題として，システム数が多く，定型化しやすいエンタプライズ系の業務システムの中から，複合機のメンテナンス作業支援ツールを題材とし，検証を行った．本ツールシステムは，Webブラウザ上で動作するシステムとし，主にスマートフォンやタブレット端末で扱うことを想定している．

ここでの基本設計書は，画面機能として，ボタンなどの画面上で目視確認できる動作の仕様および設計を設計書として定めたものとし，データベース定義や，詳細なシステム内部処理の設計は含まない．また，自動導出を試みるテストケース一覧は，ある1動作(例:ボタンクリック)による機能仕様の確認項目(条件，結果など)を記述したものを1テストケースとして，一覧にしたものである．

2.2 基本設計書の問題

基本設計書は設計工程だけでなく，製造やテスト工程などの後工程でも利用される．基本設計書の情報不足や曖昧な記述は，後工程の担当者が設計内容を理解する阻害要因となる．例えば，テストケース作成時に操作後の振る舞いや画面遷移の情報が不足していると，仕様理解が困難である．情報量が十分であっても記述量が多いと，設計書に記述すべき内容と関係ない記述も多くなりやすく，一つの事象の仕様を正確に把握することに時間がかかる．設計書のレビュー時においても，情報不足は検出が困難であり，解釈が異なったまま実装やテスト設計が行われることがある．例えば，エラー処理の際に行う処理で，入力制限をかけるのか，または入力後にチェックを行うのかどうかの記述が不足している場合などが挙げられる．また，実装を意識した設計書では，モジュール名や変数名で記述されることがあるが，開発を行う際には必要である情報でも，テストケースの検討には必要がないことがある．さらに，変数に対する説明文がないことはテスト設計者が仕様を理解できない要因となる．記述の問題としては，一つの枠の中に複数の要件を含んだ記述があると，要件を見逃して実装してしまう可能性が生じる．

本研究では，上記に挙げた問題が解決されるような効果をもった基本設計書フォーマットを作成することを目指す．しかし，現場や対象のシステムによって，基本設計書に求める厳密さは異なるため，一概に固定化されたフォーマットを作る事では解決されない．そのため，本研究では，各々の現場で効果的に活用できる，基本設計書フォーマットの導出アプローチを提案する．

業務システムにおける基本設計書のフォーマットに含めるべき項目などについては，[1]などにおいても典型的なものが示されている．本研究のアプローチはテストケースの自動生成を検討することで基本設計書フォーマットの検討を行う点において独自である．

2.3 提案アプローチ

本研究では，上記の目標を達成するための基本設計書フォーマット作成アプローチを検討した．具体的には，自由記述の問題点が生じづらい記述の基本となるルールと，設計書作成のためのアプローチを検討した．

まず，フォーマットで縛ることと縛らないこと，それぞれの利点を以下に示す．フォーマットで入力項目を縛ることにより，曖昧な表現や，基本的な項目の抜け漏れ，項目の不統一な記載場所，入力制限による複数項目の入力防止などの問題の解決が期待できる．逆にフォーマットで縛らないことにより，自由な記述による表現や，読みやすい，書きやすい表現を選ぶことができるなどの利点がある． 加えて，テストケースの自動生成が可能なフォーマットにすることにより，複数項目の入力防止や，テスト設計の早期着手，テスト工程における重複作業の効率化などの利点が期待できる．

テストケースへの自動変換が可能な基本設計書ができていれば，境界値の網羅，AND/OR 条件による分岐網羅のテストケース出力といった，基本設計工程ですでに行っていた作業をテスト工程で再度整理し直す必要がなくなり，より高度なテスト設計に注力できる．

本研究で提案する基本設計書作成アプローチ（手順）を図１に示す．

本アプローチでは，基本設計書からテスト仕様書を手作業で作成し，基本設計書から参照した部分を書き出し，不足していた記述や曖昧な記述を基本設計書にフィードバックする．また，基本設計書からテストケースを自動的に変換できることも確認する．この作業を繰り返し行い，テストケースの自動変換が可能な基本設計書のフォーマットを作成する．さらに今回は，実験として，自動化を見据え作成したフォーマットを実際に使用してもらうことによる妥当性確認を行った．

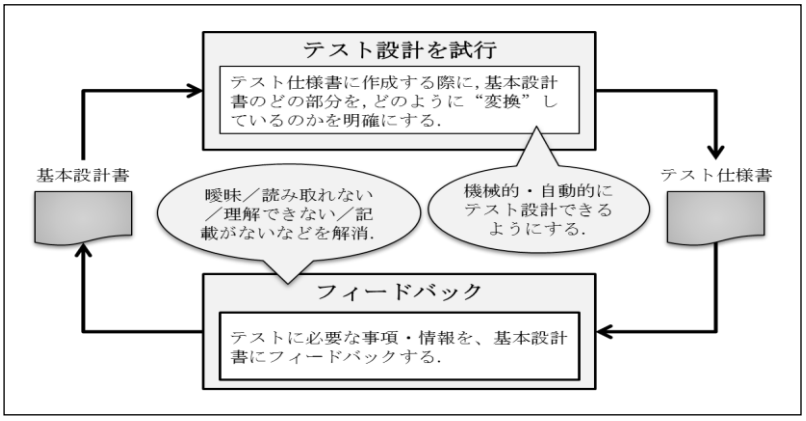

図１　基本設計書作成アプローチ

3. フォーマットの例と生成されたテストケース

本章では，本アプローチによって作成した基本設計書フォーマットの例と，そこから自動生成されるテストケースを示す．

3.1 基本設計書フォーマット

今回，アプローチを用いて作成した基本設計書フォーマットの一部を図２に示す．本フォーマットはスプレッドシート形式（Microsoft Excel）で作成し，プルダウンやマクロプログラムを活用することを想定してフォーマットの設計を行った．また，どこまでがその画面に関する仕様なのかを解りやすくするため，１つの画面に関する記述を１つのシートに行うこととした．

3.2　テストケースの自動生成

以下では，基本設計書フォーマットにより記述された基本設計書から，あるテスト観点に基づくテスト設計により定義されるテストケースを自動生成する仕組みについて述べる．本研究においては，ＵＩ部品の状態，テキストフィールドの入力制限，ＵＩイベント（ボタン・リンクによる状態遷移など）の３つのテスト観点を対象とすることにした．

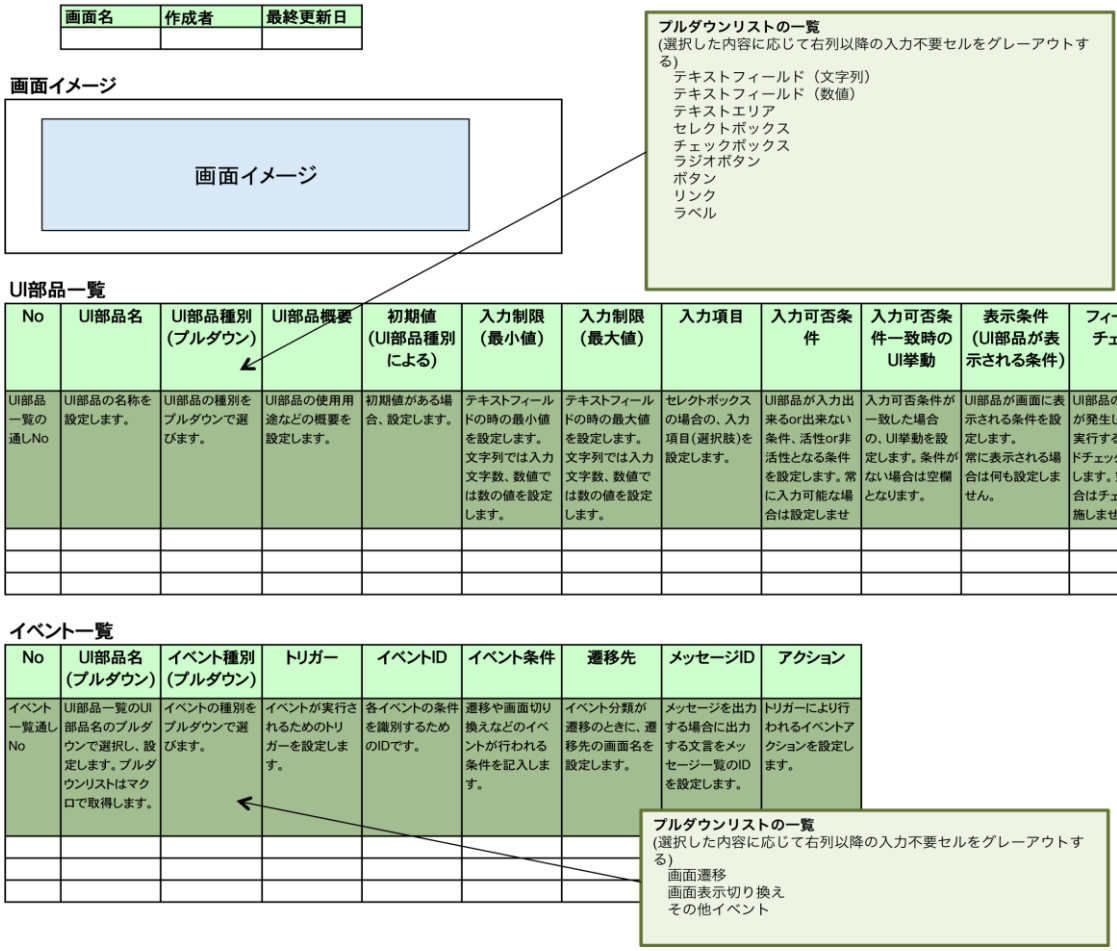

図２　作成した基本設計書フォーマット

本研究で提案するテストケース一覧を自動生成する仕組みにおいては，この 3 つのテスト観点ごとに，基本設計書における個々の記述から該当するテストケースを生成するアルゴリズムを順次実行し，それぞれで得られたテストケースを結合することで，テストケース一覧を得られた．テストケースを生成するアルゴリズムを図３に疑似コードで示す．また，基本設計書の記述例と，それを元に自動生成したテストケースの一例を，表１，２に示す．

表１ 基本設計書例 UI 部品一覧(抜粋)

UI 部品名	入力可否条件	入力可否条件一致時の UI 挙動	表示条件(UI 部品が表示される条件)
ID フィールド	「ログインエラー数」==5	入力不可	
Password フィールド	「ログインエラー数」==5	入力不可	
ロック解除ボタン			「ログインエラー数」==5

表２ 「UI 部品の状態」観点によるテストケース生成結果(表１を入力とした場合)

テストケース(条件)	UI 部品名	結果
「ログインエラー数」==5	ID フィールド	入力不可
not (「ログインエラー数」==5)	ID フィールド	not 入力不可
「ログインエラー数」==5	Password フィールド	入力不可
not (「ログインエラー数」==5)	Password フィールド	not 入力不可
「ログインエラー数」==5	ロック解除ボタン	表示
not (「ログインエラー数」==5)	ロック解除ボタン	非表示

```
テストケース一覧 = []

// 「UI 部品の状態」観点によるテスト生成
for 表示条件,入力可否条件,UI 部品名,入力可否条件一致時の UI 挙動 in UI 部品一覧 & UI 部品一覧.表示条件 OR UI 部品一覧.入力可否条件
  テストケース集合 = if 表示条件 then
    [ 表示条件, not(表示条件) ]
  else if 入力可否条件 then
    [ 入力可否条件, not(入力可否条件) ]
  end
  結果集合 = if 入力可否条件一致時の UI 挙動 then
    [ 入力可否条件一致時の UI 挙動, not(入力可否条件一致時の UI 挙動) ]
  else
    [ “表示”, “非表示” ]
  end
  テストケース一覧 << テストケース一覧生成(テストケース集合, UI 部品名, 結果集合)
end

// 「テキストフィールドの入力制限」観点によるテスト生成
for UI 部品名, 入力制限_最小値, 入力制限_最大値 in UI 部品一覧 & UI 部品一覧.入力制限(最小値) OR UI 部品一覧.入力制限(最大値)
  フィールド名=UI 部品名
  for UI 部品名 in UI 部品一覧 & UI 部品一覧.フィールドチェック = フィールド名
   フィールドチェック UI 部品名=UI 部品名
   テストケース集合 = [ "入力制限(最小値)-1", "入力制限(最小値)", "入力制限(最大値)", "入力制限(最大値)+1"]
   条件集合 = [ フィールド名+" に "+(入力制限_最小値-1)+"入力",
          フィールド名+" に "+(入力制限_最小値 )+"入力",
          フィールド名+" に "+(入力制限_最大値 )+"入力",
          フィールド名+" に "+(入力制限_最大値+1)+"入力"]
   結果集合 = ["フォーム入力エラーが起きること",
          "フォーム入力エラーが起きないこと",
          "フォーム入力エラーが起きないこと",]
          "フォーム入力エラーが起きること"]
   テストケース一覧 << 実行不可なテストケース除去(
    テストケース一覧生成(テストケース集合, 条件集合, フィールドチェック UI 部品名, 結果集合)
   )
  end
end

// 「UI イベント(ボタン・リンクによる状態遷移など)」観点によるテスト生成
for UI 部品名, イベント種別, イベント条件, 遷移先, メッセージ ID in イベント一覧
  テストケース = イベント種別, 条件 = イベント条件
  結果 = if 遷移先 then
   遷移先 + " に遷移すること"
  else if メッセージ ID then
   メッセージ(メッセージ ID)+"が表示されること"
  end
  テストケース一覧 << テストケース生成(テストケース, 条件, UI 部品名, 結果)
End
```

図３　基本設計書の記述からテストケース一覧を生成するアルゴリズム(疑似コード)

4. 基本設計書の妥当性の確認

自動化を見据えて作成した基本設計書フォーマットが，人が記述する基本設計書のフォーマットとして妥当であるかを確認するため，実験を行った.

4.1 実験の目的

本実験では，作成した基本設計書フォーマットを以下の3点の観点で確認した.

(1)自動化を見据えて作成した基本設計書フォーマットが，人が理解でき，読み書きしやすく，システム開発の経験者ならば抵抗なく正確に書くことができるかの確認

(2)作成された基本設計書からテストケースに変換できるかの確認

(3)使用した意見をもらい，改善点からアプローチのサイクルを回すことができるかの確認

4.2 方法

被験者に対して，基本設計書フォーマット，ガイドラインとなる記述サンプル，対象システムの概要画面イメージを提示し，実際に基本設計書フォーマットに記入をしてもらった．その後，記入の難易度，およびフォーマットへの所見をもらい，著者らで，改善の余地のあるフィードバックが得られたかを確認した．また，著者らで，被験者によって記入された基本設計書を，アルゴリズムを用いてテストケースに変換し，その結果から基本設計書フォーマットの有効性を確認した.

4.3 実験対象

被験者：4名

被験者の背景：　チームA：Web画面の基本設計書作成経験あり：1名，なし：1名

　　チームB：Web画面の基本設計書作成経験あり：2名

対象画面：　チームA：ログイン画面，タスク一覧画面

　　チームB：作業内容登録画面，メンテナンス作業画面

4.4 結果

作成された基本設計書においては，項目として記述方法を縛った部分については，被験者による差異はなく，画面遷移時のアクションなど，自然言語で記入する部分に差異がみられた.

以下に被験者により生じた差異の例を示す.

・メッセージIDの記入がある場合と，ない場合がある：記載内容の違い

・画面遷移とその他イベントを分けている場合と，分けていない場合がある：記載単位の違い

作成された基本設計書からテストケースに変換することができた. しかし, 基本設計書での差異に応じて, テストケース数に差異が生じた．また，未記入欄があることで，期待したテストケース数にならなかった部分もあった.

表3 作成された基本設計書の差異（結果1）

No	UI部品名(プルダウン)	イベント種別(プルダウン)	トリガー	イベントID	イベント条件	遷移先	メッセージID	アクション
5	作業結果送信ボタン	画面遷移	クリック	条件5	タスク情報送信処理が完了	タスク一覧画面	msg_001	業務管理システムに現在のタスク情報を送信し，メッセージ表示後画面遷移する

表4 作成された基本設計書の差異（結果2）

No	UI部品名(プルダウン)	イベント種別(プルダウン)	トリガー	イベントID	イベント条件	遷移先	メッセージID	アクション
5	作業結果送信ボタン	画面遷移	クリック	条件5	イベント#6 が正常に完了すること	タスク一覧画面		タスク一覧画面に遷移する
6	作業結果送信ボタン	その他イベント	クリック	条件6	特になし			業務管理システムに現在のタスク情報を送信する.

また，実験とフィードバックの結果，以下の知見を得られた.

・基本設計書として，記入項目数は多くない

・初回はフォーマット理解に時間が多少かかるが，慣れれば多く時間をかけないで作成可能

・基本設計書の作成結果は，ガイドラインに記載されるサンプルの影響を強く受けるため，ガイドラインによって書き方を縛ることができる

さらに，改善点としては以下のような点が挙がった.

・自由記述部分をガイドラインで縛ることで，テストケースを自動化できる部分を増やせるのではないか
・ 項目の意味がわかりづらい部分があったため，文言の検討が必要
・ 空欄が記入漏れなのかが判断つかないため，記述ルールを明確にすることが必要

文言に関する指摘としては，「画面遷移と画面表示切り替えの違い」が得られた．前者は画面の遷移，後者は画面遷移がなく UI 部品のみの表示の切り替えを意図した．

4.5 実験に関する総括

本研究で作成した基本設計書フォーマットは，自動生成できるが人が理解し書くことができるものであると言える．また，自然言語で自由記述を行った部分については，今回のアルゴリズムではそのまま写し取る部分であるため問題は生じない．実験を行うことにより，被験者から意見を得ることができた．意見を取り込み，さらにフォーマットの改善をはかることができ，本提案アプローチのサイクルを回すことができることが確認できた．今後，厳密さと自由さのバランスをとりつつ，設計書の作成ができるアプローチであることの検討をさらに深めていく必要がある．

5. 考察

5.1 自動生成について

本研究は，自動生成を見据えて基本設計書フォーマットを検討するというアプローチを用いた．このアプローチは，基本設計書の記述の厳密さと自由さのバランスを追求する手段として有効であるという感触を得た．自動生成が可能であるということは，記述における一定の厳密さを担保しており，その中で，より自由度の高い記述を模索することが可能となる．

また，そのような模索を通じて，本研究で作成した基本設計書フォーマットは，特定の部分において自由度の高い記述を含んでいても，生成されるテストケース一覧の有効性に影響を与えないということを見出した．つまり，本基本設計書フォーマットは，自然言語による記述に近く，現場で導入しやすいものでありながら，テストケース一覧の自動生成といった後工程での活用につなげることができるものであると言える．

さらに，本基本設計書フォーマットは，自動生成アルゴリズムの導出が容易であるという感触を得た．例えば，自然言語による条件の記述中において複数の条件が「または」等で織り交ぜて記述されると，それらの個々を扱うテストケースを自動的に抽出することが難しい．これに対し，各種条件列におけるセルの記述は，OR 条件を含む内容とならないようにするといった簡単な工夫により，アルゴリズムを簡潔なものにすることができた．これは，例えば，本提案の基本設計書フォーマットを現場にあわせてカスタマイズした場合に，新たな自動生成アルゴリズムを容易に導き出すことができるということを示唆する．つまり，本提案の基本設計書フォーマットは，導入コストが低く様々な現場で導入しやすい，後工程で活用できる文書の自動生成の仕組みであると言える．

5.2 フォーマットとアプローチについて

本アプローチにより基本設計書フォーマットを作成した効果を以下に挙げる．今回，基本設計書のフォーマットとして縛ったことにより次のような利点が得られた．

第一に，設計者にとっては，システム開発経験があれば抵抗なく容易に書ける基本設計書となった．これは，4 章で挙げた実験の結果から，Web 画面の基本設計書作成経験があれば，フォーマットの項目についてガイドラインを参照することで記述できたことから言える．また，この実験結果から，開発者などの後工程担当者にとっても読みやすいことが示唆される．今回は，各画面，被験者が 2 名のみであり，また，読み手側やテスト設計者側への実験は行っていない．そのため，複数人，かつ複数の立場の被験者での実験が今後の課題となる．また，今回はフォーマットの妥当性とテストケース変換を確認する実験を行ったが，自動生成されたテストケース一覧の有効性についても，今後，検証が必要となる．

第二に，書き手の違いによる記述の差異を押さえることができる．これによって，読み手側にとっても，フォーマットが理解できていれば，どこに何の情報が書かれているかが分かり易くなる．また，テスト設計者にとっては，表現を縛ることでテストケースを自動生成することが可能となるため，単純なテストケース導出の手間が省け，プロジェクトにおいてより高度なテスト設計やテスト戦略などに力を入れることができるようになると期待される．

逆に，表現を縛らない部分を設ける利点としては，設計者にとって，条件やアクションに関する特別な

記述言語，例えば，仕様記述言語のような厳密な記述方式を覚えるよりも学習コストが低いことがある．また自然言語で記述できるため，表現の幅が広く，書きやすく，また後工程担当者にとっても読みやすい設計書となる．

本フォーマットを作成するにあたって，仕様記述言語のような厳密な記述方式を設けることも考えられるところ，あえて自由記述ができるような項目，フォーマットにした．結果，予想していた被験者による差異はでたが，書きやすく読みやすい設計書になったと考える．

提案アプローチにおける利点は次のように挙げられる．

・本アプローチでは，フォーマットの厳密さと自由さのバランス調節を検討していくことができ，基本設計書に関する問題点を解決するための一助となる．
・固定化された基本設計書ではなく，作成アプローチのため，厳密さと自由さのバランスは現場，対象システムに応じて変動させ，最適なものを検討していくことができる．

5.3 課題

今回，本アプローチを用いフォーマットを作成し，アプローチに対する課題を得た．

第一に，今回は疑似コードにて実際に自動化できるものとしたが，自動化ツールの作成までに至っていない．実際に自動化ツールを作成することは今後の課題となる．また，今回は自動化により作成したテストケース一覧については必要最低限のものを出力することに注力したが，今後はテスト仕様として十分なテストケースの導出を目指す．

第二に，今回作成した基本設計書フォーマットでは，自由記述欄では予想した通りに記述の差異を確認した．しかし，ガイドラインとなるサンプルをつけることで，部分的に書き方の誘導を図れることがわかった．今後，フォーマットと共にルール付けのためのガイドラインも検討していく必要がある．例えば，テストケースへの変換時に設計書空欄があるものか記入漏れかを判断するために，空欄には必ず明示的にハイフンを入れるなどのルールが考えられる．

第三に今回，フォーマットの妥当性確認のために行った実験で，項目タイトルがわかりづらいなど予想外の意見を得られ，より改善につなげられることがわかった．実験を行うタイミングや被験者経験値などを調整することで，よりよい改善案を得られると考えられる．5.2 節でも触れた通り，読み手を含む複数人による実験を行い，汎用性の確認を行う必要がある．

第四に，今回はテストケースへの自動変換を 1 アクション 1 機能となる基本的なパターンのみを行ったが，1 アクションに複数機能となる複雑なテストケースに対応できるような，有用性のある変換を行えるようにフォーマットと変換方法を検討する必要がある．

6. まとめ・今後の展望

テストケース作成の自動化を見据えた基本設計書フォーマットの作成アプローチにより，今回はライトウェイトだが，テストケース一覧生成の自動化ができる基本設計書フォーマットが作成できた．本アプローチを使用することで，それぞれの現場やシステムにあわせた厳密さと自由さのバランスの取れた基本設計書フォーマットの作成ができる．

今後の展望としては，実験で得られた改善点を踏まえ，基本設計書フォーマットをさらに拡張していき，システム開発における本アプローチの追求を目指す．

謝辞

本論文は，著者らが 2017 年度 SQiP 研究会において行った研究をまとめたものです．

本研究を行うにあたって，終始丁寧なご指導ご鞭撻をいただいた研究コース 6 アドバイザーの九州大学大学院 荒木啓二郎主幹教授，主査の栗田太郎氏，副主査の石川冬樹准教授に心より感謝いたします．研究の機会，環境を与えて下さった日本科学技術連名の事務局の皆様に感謝申し上げます．励まし合いながら共に研究を行い，本研究の実験に協力いただいた GOΦWY チームの皆様に感謝の気持ちと御礼申し上げます．最後に，生活を支えてくれた家族，疲れた時に愚痴を聞いてくれた友人らに心より感謝いたします．

参考文献

[1] 情報処理推進機構，機能要件の合意形成ガイド，2010 年 3 月