第3分科会

設計者自身による設計品質作り込み
ー テストエンジニア視点の活かし方 ー

Approaches to Deliver Quality Design by Designers Themselves
- How to Leverage Test Engineer's Viewpoints -

主査	飯泉 紀子	株式会社日立ハイテクノロジーズ
副主査	田所 孝文	株式会社山武
研究員	大立 薫	ベックマン・コールター・バイオメディカル株式会社
	千葉 美千代	株式会社エス・キュー・シー

研究概要

本研究の目的は、設計者自身が自分の作成した機能仕様書の品質を向上させる方策を見出すことである。

ソフトウェア開発の現場では、多くの不具合をテスト工程で検出している。検出された不具合は設計工程で作り込まれたものも多く、その対策にかかる時間やコストがビジネスチャンス喪失の原因になっている。設計工程で品質を作りこむ施策としてはレビューが代表的である。最近はテスト組織が設計工程のレビューに参画して、テストエンジニア視点を活かした品質作り込みを行う取組みが多々報告されている。しかし、独立したテスト組織がなかったり、組織を超えた取組みに対する壁が潜んでいたりして、実践は容易ではない。そこで我々は、設計者が設計品質を自らの手で向上させることを志し、機能仕様書品質向上の研究を進めた。狙いは、テストエンジニアの視点を活用した機能仕様書の記述漏れ防止である。本報告では、テストエンジニアが設計者になり機能仕様書を作成する実験、および設計者がテストエンジニアになり自分あるいは仲間が作成した機能仕様書を入力としてテスト設計する実験を通して、設計者に不足しがちな観点を抽出したのでこれを報告する。また、機能仕様とテスト仕様を同時に設計するというアクティビティが、テストエンジニアの視点の理解、すなわち設計者に不足しがちな観点を理解する上で有効であることを報告する。

Abstract

In many software develoment projects, most defects are detected in the testing phase. The defects inserted in the design phase can cause additional time and cost for reworks, as well as the threat to potential business opportunities. Although review is well-known to accomodate this issue and various approaches for reviews with the test engineer's viewpoints, it is not yet easy to practice the review process effectively, due to such reasons as no independent testing organization exists or as the barriers between organizations or stakeholders.

With our strong belief, we set our primal goal to leverage the test engineer's viewpoints to prevent the omissions in the specifications so that the designers can improve the quality of the specifications by themselves. We carried out two experiments to observe the missing viewpoints of designers; these are (1) the test engineer designs the functional specification as a designer and (2) the designer designs test specification as a test engineer using the functional specification made by themselves or their own colleagues.

This paper reports the observed results of these experiments and the effectiveness of simultaneous design to understand the mind-set of designers.

1. 研究動機

ソフトウェア開発の現場では、多くの不具合をテスト工程で検出している。検出された不具合は、設計工程で作り込まれたものも多く、その対策にかかる時間やコストがビジネスチャンス喪失の原因になっている。さらに、組込みソフトウェアの場合、製品出荷後に発見された不具合によりリコールとなることもあり、社会へ与える影響は大きい。ソフトウェアの品質を設計段階から作り込む施策として、古くからレビューが知られている。最近では、テストエンジニアが設計工程のレビューに参画して、テストエンジニア視点を活かした品質作り込みを行う取組みが多々報告されている。しかし、組込みソフトウェア開発の場合、少数のエンジニアが設計からテストまでを担当することも未だ多く、専門のテストエンジニアがいるとは限らない。このような環境では、設計者自身が設計品質を向上させるという、ある種本質的な品質作り込みが必要となる。設計品質は機能仕様書に反映され、初めて作り込みとなる。組込みソフトウェア開発では、一からすべての機能を開発することはめずらしく、ほとんどが派生開発という特徴があるため、機能仕様書の品質は、派生させて開発した製品の品質にも大きな影響を与える。

昨年の分科会では、調査の結果、テスト工程で発見された不具合の約40%が要求仕様設計段階で入り込んでいたという事がわかった。そして、要求仕様書の「仕様不明瞭」「仕様記載ミス」「仕様記載漏れ」を防止し要求仕様書の品質を向上させる施策を検討した[1]。今年は、分科会メンバーに設計とテストのスペシャリストが各々存在したこともあり、テストエンジニアの視点を活かし設計者自身が設計品質を機能仕様書に作り込む方策について研究することとした。常に我々が抱えていた問題意識は、「テストエンジニアがテスト仕様を設計するときに気づけることを、なぜ設計者は設計段階で気づけないのか？」ということである。

2. 研究目標

本研究の目標は、設計者が設計品質を自らの手で向上させる方策を考案することである。特に、テストエンジニア視点を活用する方法に着目する。研究は次の手順で進めた。

(1) 設計品質は機能仕様書に反映されることから、機能仕様書の品質について調査する。機能仕様書の品質と後戻りとの関係を考察し、注目する品質の特性を特定する。

(2) テストエンジニアの視点とは何か、設計者の視点とどう違うのかを、先行研究を調査することにより把握する。

(3) 機能仕様書を作成する場合とテスト仕様書を作成する場合とで、思考が異なることに着目し、次に示す2つの実験により、設計者に不足しがちな観点を抽出する。
- テストエンジニアによる機能仕様書の作成
- Wモデルに従った、設計者による機能仕様とテスト仕様の同時設計

(4) 設計者に不足しがちな観点を考察し、設計品質を自らの手で向上させ機能仕様書に反映する方策を導出する。

3. 機能仕様書の品質

まず、本研究で対象とする機能仕様書を定義しておく。本研究では、SLCP-JCF2007 の共通フレーム2007[2]で定義されている"1.6.4 ソフトウェア要件定義"および、"1.6.5 ソフトウェア方式設計"のアクティビティにおいて作成されるドキュメントを対象とし、これを『機能仕様書』と呼ぶことにする。

機能仕様書は、詳細設計・実装・テストなど以降に実施される工程に密接に関わるドキュメントであり、「設計するもののイメージを共有する」、「実装するものを伝達する」、「テストするもののあるべき姿を伝達する」といった目的で使用される。従って、設計品質は機能仕様書に入れ込まなければならない。我々は機能仕様書の品質に着目することとした。

「機能仕様書」の品質については、IEEE 830-1998[3]の「良いソフトウェア要求仕様書の特性」(以下、品質特性と称する)が参考になる。そこで機能仕様書の記述において、IEEE 830-1998[3]の品質特性が損なわれた場合の事象を考えた。そして、埋め込まれた欠陥の検出難易度と修正難易度を後戻りのリスクとし、それぞれ 3 段階評価した。その結果を表1に示す。

表1) 要求仕様書の品質特性(IEEE 830-1998[3])が欠落した場合の事象と後戻りのリスク

品質特性	簡単な説明	品質特性が損なわれた場合の事象	検出難易度	修正難易度
正当性	要求仕様書に記述されている要求が、ソフトウェアが満たすべき事柄と一致していること	間違ったものを作る 間違ったテストをする	**中**	**高**
非曖昧性	要求仕様書に記述されている要求が一意に解釈できること	間違ったものを作る、または間違ったテストをする可能性がある	低	中
完全性	顧客や利用者のニーズが、漏れなく要求仕様書に記述されていること	機能が欠落するため、作られないし、テストもされない	**高**	**高**
無矛盾性	要求仕様書間で矛盾や衝突がないこと	間違ったものを作る、または間違ったテストをする可能性がある	低	中
ランク付け	各要求について、重要度あるいは安定度を示す指標を明確につけておくこと	作業順番に影響を与えることはあるが、重大な問題にはなりにくい	低	低
検証可能性	ソフトウェアが要求を満たしているかどうかを機械的に確認可能であること	テスト不可能なものを作る、テストに時間がかかり十分なテストができなくなる	低	中
変更可能性	要求へのあらゆる変更が容易に、完全に、一貫して行えるような構成になっていること	変更時の作業に影響を与える	高	中
追跡可能性	個々の要求の起源が明確で、将来の開発で改良された文書などとの対応付けが取れること	同上	低	低

検出難易度、修正難易度とも高いものは、後戻りのリスクが高い。表1に示す検討の結果より、「正当性」および「完全性」が損なわれた場合、後戻りのリスクが高くなる。

- 「正当性」の中の「書き間違い」は少なくとも記述されているため、設計者以外の目に触れることが多く、排除できる可能性がある。しかし、本当の要求に合致しているか判断できない場合もあるため、誤りを検出できない可能性も残る。修正量は多くなる。
- 「完全性」の中の「抜け・漏れ」は、想定外の事象も含め、それ自体の存在に気づく可能性が低い。さらに修正量は多く、難しくなる。
- 「非曖昧性」と「無矛盾性」は、ソースコード作成時やテスト実行に曖昧さや矛盾点が指摘されるため、後戻りのリスクはそれほど大きくない。
- 「ランク付け」は、後戻りリスクには影響が少ない。
- 「検証可能性」は、後述するWモデルなどのプロセスにより早期検出が可能なため、後戻り作業を軽減することが可能である
- 「変更可能性」、「追跡可能性」は、「プロセス」や「環境」のフレームワークによって後戻りに対するリスクを軽減することが可能である

一方、我々の開発現場で発生している不具合について原因解析を行った結果からは、機能仕様書の「仕様記載漏れ」による割合が高いことがわかった(図1)。この機能仕様書の「仕様記載漏れ」は、表1の「完全性」の一部である。そこで我々は、現在の機能仕様書の品質に関する重要問題を「完全性」とした。

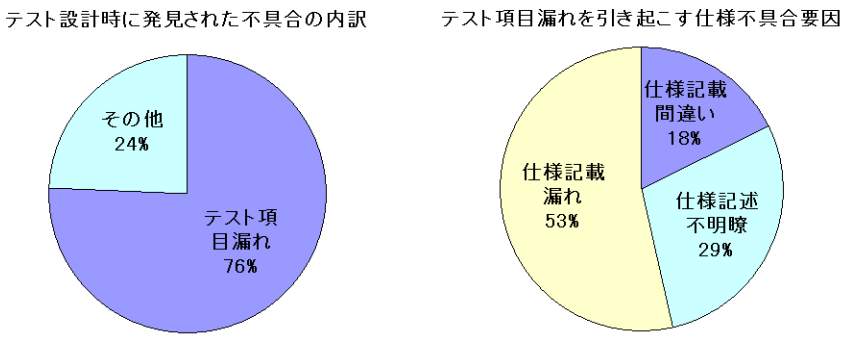

図 1. 不具合原因解析結果

4. テストエンジニアの視点

本章では、テストエンジニアの視点はとは何か、設計者の視点とどう違うのかを、先行研究を調査することにより把握する。

先行研究の一つに、「テストをどのように行うかを説明できるか」というテストエンジニアの視点をレビュー時の欠陥検出に生かすというものがある[4]。これは、一つ一つの機能要件／非機能要件について、要求仕様分析表を用い「目的」「アクタ」「状況」「イベント」「入力」「動作」「結果」「重要度」という項目を埋めていくことで、要求仕様書を検証可能性の観点から分析するものである。従来型レビューと比較した場合、欠陥の総検出数に大きな差は見られなかったが、非曖昧性に関する欠陥検出に効果があるという結果が提示されている。

別の先行研究[1]では、機能仕様書とテスト仕様書を同時に同一人物が設計すれば、設計仕様の誤りや不備を認識できるようになり、設計段階で品質を作りこむことができると述べている。そして、同一人物による機能仕様書とテスト仕様書の同時設計を可能とするプロセスとツールを考案し、それを適用した事例を紹介している。これも、検証可能性の観点を活用した施策である。このような考え方は、Wモデル[5]として認知されてきているが、現場への適用はまだ多くない。

通常、機能仕様書を作成するときは、「間違いや漏れがないように」という立場で「正しいかどうか」に注力し記述する。設計者は頭の中にある思考(設計)を記述することに集中するため、本人が元々持っている誤解や思い込みには気づくことが難しい。一方、テスト仕様書を作成するときは、「本来、間違いや漏れがあるものだ」という立場で「テスト設計によって誤りを見つけ出すこと」に注力する。つまり、テストエンジニアの視点は検証可能性に特化されるものではなく、それ以外の何かがあるはずである。そこで、次の2つの実験を行い、設計者に不足しがちな観点の抽出を試みた。

(1) テストエンジニアによる機能仕様書の作成:テストエンジニアなら、設計者とテストエンジニアの両者の視点を持ち合わせた機能仕様書を作成可能ではないか(ただし、設計スキルの問題を除く)
(2) Wモデルに従った、設計者による機能仕様とテスト仕様の同時設計:同時設計が成功すれば、設計者自ら不足しがちな観点に気づけるのではないか

5. 設計者に不足しがちな観点の抽出

機能仕様書を作成する場合と、テスト仕様書を作成する場合とで、思考が異なることに着目し、設計者に不足しがちな観点の抽出を試みる。本研究では、テストエンジニアが設計者になり機能仕様書を作成するパターンと、設計者がテストエンジニアになり、自分あるいは仲間が作成した機能仕様書を入力としてテスト設計をする 2 種類の実験を実施した。

5.1. テストエンジニアによる機能仕様書の作成

この実験は、テストエンジニアが機能仕様書の欠陥を検出できるのであれば、テストエンジニア自身が機能仕様書を設計した場合にテストの観点が自動的に補完され、結果としてより質の高い仕様書を作成できるのではないか、という仮説に基づく。本実験では、以下を目的とした。

(1) テストエンジニアの視点を盛り込んだ機能仕様書を作成する試み
(2) 設計者の心理をテストエンジニアが理解する
(3) 設計者が記述漏れを起こす原因を探る

テストエンジニアが機能仕様書を作成するにあたり、身近にあって比較的機能をイメージしやすいシュレッダーを例として取り上げた。既存の機能仕様書からの先入観を持たないよう、機能仕様書のフォーマット指定は設けず、意図的に時間的な制限を設け、なるべく実務に近い環境下で実験を行った。機能仕様書作成後は、自ら作成した機能仕様書を用いてテスト設計を行い、実際にテストエンジニアの視点が機能仕様書に盛り込まれているかどうかを検証した。

テスト設計の結果、動作条件外の入力、ユースケースとして稀なケース、複数機能や複数操作が組み合わさった場合、異常系などの記述が漏れていた。このことから、テストエンジニアが機能仕様書を作成しても、設計者と同様の傾向を示すことがわかった。

これは、対象機能が「正しく動作すること」に焦点を当てながら記述をしているためと思われる。記述の仕方についても、多くのテストエンジニアがその範囲に疑問を抱く「など」という言葉の使用に抵抗がなくなり、設計者と

変わらない行動を取ることがわかった。同時に、工数制限が設けられていたこともあり、期限(納期)に間に合うよう必要最小限の事項のみを記述したことも記述漏れの一因である。表 2 に、テストエンジニアが機能仕様書を設計する上で気づいた点を一覧にまとめた。

表 2) 機能仕様書作成で気づいた点

No.	気づいた内容	具体的な理由
1	機能の全体像をイメージしづらい	機能全体の構造や動作に対し、漠然としたイメージしかわかない
2	スキル不足があった	記述フォーマットを決めかねる、描画ツールを使いこなせない、文章での表現が困難
3	仕様決定に確信が持てない	機能に対する一般的な「あるべき姿」についての決定が主観的となり、確信が持てない
4	自己レビューの際に、考えが重複する	同じ箇所を異なるタイミングで自己レビューを重ねると、考えが重複して混乱してしまう
5	機能が複雑になると、考えすぎて混乱する	抜け漏れがないよう、関連する機能も考慮すると、範囲が広くなり考えがまとまらずに混乱する
6	「機能が動作する」視点が狭められる	「機能」に集中し、「操作(シナリオ)」に注意が向かなくなる
7	「正常系」のみを考えてしまう	「機能が動作する条件」の視点に偏り、逆の「機能が動作しなくなる場合」の視点が欠ける
8	曖昧な表現を使用する	すべての対象範囲の洗い出しが煩わしくなり、「など」の曖昧表現を使用することに抵抗がなくなる

以上の実験より、テストエンジニアの視点を無意識に補完することは困難であるという結果に至ったが、自ら作成した機能仕様書からテスト設計を行った場合には欠陥が検出されたということは、テスト設計の時点で設計者思考からテストエンジニア思考に転換されたと思われる。

Beizer が述べているとおり[6]、テストエンジニアの態度は無意識に懐疑的であり、無意識にテスト対象物を破壊する方法を考える。おそらく、自ら作成した機能仕様書においてもこのような姿勢でテスト設計を行ったことにより、機能仕様書の欠陥を検出できたのではないだろうか。そうであれば、設計者とテストエンジニアが同一人物であったとしても、機能仕様書作成後に思考を一度リセットし、記述した内容を「懐疑的な姿勢」で見直すことによって、テストの抜け漏れを防止できるのではないだろうか。

5.2. W モデルに従った機能仕様とテスト仕様の同時設計

Wモデルとは、開発とテストを上流から並行して進めるプロセスモデルである。Wモデル適用による効果は、テスト設計を上流工程で行うことで、テスト実施を行わなくても、仕様の漏れ、曖昧な点、矛盾点などを見つけることができること、そして、すぐに欠陥を修正できることである。先行研究[7]では、機能仕様とテスト仕様を設計工程で同一人物が設計するプロセスを提案し効果を述べている。そして昨年の研究[1]でもこれらと類似した、機能仕様とテスト仕様の同時設計による「仕様不明瞭」の問題解決を提案した。しかし、実際にWモデルに従った機能仕様とテスト仕様の同時設計を現場に適用するには、次のような懸念を払拭する必要があった。

- Wモデルを導入すると、品質が向上するような気はするが、工数・要員も増えるのではないか
- 本当に後戻りが減るのか、導入効果や価値が不安

【現場適用上の工夫】

そこで、以下に示す工夫によって現場適用を推進した。

(1) Wモデルを推進する人が自ら実践する。例えば、自分以外が書いた機能仕様書をテスト設計し、欠陥を指摘する。

(2) 機能仕様書の欠陥により発生する事象を説明し、設計者自らが欠陥検出する価値を共有する。
例えば、
- この機能仕様書で、実装可能か？
- この機能仕様書で、テスト可能か？テスト実施段階で設計のやり直しは起きないか？

上記を説明したあと、メンバー間で自分以外の機能仕様書をテスト設計し、機能仕様書の欠陥が見つかることを体験する。

(3) 振り返りを実施する。どのような機能仕様書の欠陥が見つかったのかメンバー同士で意見を出し合い、習慣づけする。

【現場適用の実際】

今回、同時設計の対象とするドキュメントは、「ソフトウェア方式設計」・「ソフトウェア詳細設計」時に作成される設計書とした。

5.1 節の「テストエンジニアによる機能仕様書の作成」で述べたように、テストエンジニアの思考になると欠陥を検出しやすい。そこでこの実験では、設計者自らが「テスト設計」に取り組むため、次の手順で実施した。

(1) 機能仕様書の作成は、要求(要件)を漏れなく記述することに注力し、容易に考えられるケースの記述に留める。機能仕様書の欠陥を検出するのは、テスト設計時と割り切って早々に切り上げる。
(理由) 設計者の視点のまま時間をかけるのを止めて、テストエンニジアの視点へ早く切り替えさせるため。

(2) 自分以外が作成した機能仕様書をベースに、設計者がテスト設計を行う。
(理由) 自分が作成した機能仕様書だと、設計者の視点に戻ってしまいがちなため。

(3) テスト設計中に検出された欠陥を機能仕様書に確実に反映させるため、自作のツールを利用する。(付録1の図3)
(理由) このツールは、設計途中の仕様変更も反映できるようにするために、テスト仕様書に機能仕様書をリンク貼り付けし、「機能仕様書」と「テスト仕様書」が連動する仕組みとしている。

(4) テスト設計完了と同時に、機能仕様書の完成とする。
(理由) 同時設計におけるテスト設計で検出した欠陥の修正は、工程内修正と見なし後戻りと考えないようにするため。

【現場適用の結果】

このような方法で、Wモデルに従った機能仕様とテスト仕様の同時設計を現場適用した結果を表 3 に示す。同時設計を行った結果、仕様記載漏れによるテスト項目漏れが 15件から 0 件と大幅に改善することができた。さらに、仕様記述不明瞭・仕様記載間違いによるテスト項目抜け・テスト項目ミスも軽減することができた。そして、全体的な不具合発生件数を 30%軽減することができた。

適用効果についてプロジェクトメンバーにヒアリングしたところ、メンバー全員が「仕様記載漏れ」に効果があると答えた。同時設計では、テスト設計の際に記述されている文章を一語一句丁寧に読むので、「仕様記述不明瞭」・「仕様記載間違い」の欠陥も検出できるとの答えもあった。そして、自分以外の機能仕様書をテスト設計することも良かったと答えている。

表 3) W モデル適用前後の下流工程における機能仕様書に起因する不具合件数の比較

	仕様記載漏れによるテスト項目抜け	仕様記述不明瞭によるテスト項目抜け	仕様記載間違いによるテスト項目ミス	合計
適用前	15	8	5	28
適用後	0	3	0	3

今回、設計者自らがテストエンジニアになり機能仕様書を見直す取組み(Wモデルに従った機能仕様とテスト仕様の同時設計)を行ったことを基に、設計者が機能仕様書作成時に不足しがちな観点を以下に述べる。

機能仕様書を書くときは誰でも、要求(要件)を機能に展開することに注力する思考となるため、いわゆる「正常系」の仕様が中心となってしまったり、機能に注力することで視野が狭くなってしまったり、他機能との関連を見落としがちになる。また、設計者は開発対象について知識を持っているため、自分自身の価値基準で「これならわかるだろう」という思いから、「意図的に書かないケース」や、「無意識に漏れてしまうケース」がある。その結果、処理の詳細が不足してしまったり(エラー条件やそのときの対応が不十分) 、機能間にまたがる処理が不足してしまったりする。

また、自身で作成した機能仕様書を自己レビューしても、「疑ってかかる」ことが難しく欠陥を検出できにくいということがわかった。Beizer[6]や Myers[8]が述べているとおりであり、設計者とテストエンジニアでは目標が相反するため、同一人物が仕様を作成し、かつテスト設計を的確に行うのは容易ではないことを実感した。

Wモデルに従った機能仕様とテスト仕様の同時設計によって「仕様記載漏れ」を防止できた要因として、以下が考えられる。

- 本人が書いていない「機能仕様書」をテスト設計することで、誤解や思い込みを排除でき、欠陥(仕様不明瞭、仕様記載ミス、仕様漏れ)に気づけたこと。

・ Wモデル導入以前は、「なぜ、この記述内容で伝わらないのか」という思いを設計者全員が持っていた。しかし、設計者間でテスト設計を試みると、これではテストできないという事態を経験し意識が変わったこと。これは直接的な要因ではないが、取り組む人の強い気持ちは重要である。

6. 本研究の考察

テストエンジニアが設計者になり機能仕様書を作成する実験、および設計者がテストエンジニアになり自分あるいは仲間が作成した機能仕様書を入力としてテスト設計する実験の結果から得た、設計者に不足しがちな観点をまとめる。そして、設計品質を自らの手で向上させ機能仕様書に反映する方策を考察する。

6.1. テストエンジニアの視点の有効活用

設計者がテストエンジニアの視点を活用するというのはどういうことかを考えた。先行研究[4]におけるテストエンジニア視点は、「テストをどのように行うのかを説明できるか」という検証可能性であった。我々はテストエンジニアの視点を活用して、設計者が陥りがちな観点が『正しいという思い込み』であることを突き止めた。『正しいと思い込んでいる』ため、その時点では自身が記述している範囲がすべてとみなされ、それ以外の領域は存在しないことになる。そのため、思考の範囲がそこから広がることはない。しかし、例え設計者とテストエンジニアが同一人物であったとしても、機能仕様書作成後に思考を一度リセットし、記述した内容を「懐疑的な姿勢」で見直すことによって、抜けや漏れを発見できることが実験によりわかった。つまり、設計品質を自らの手で向上させ機能仕様書に反映するには、常に設計者が正しいという思い込みを排除し、懐疑的な姿勢で内容を見直すことである。

今回実施した機能仕様とテスト仕様の同時設計の実験では、設計者とテストエンジニアを別に用意することができたが、リソースの制約などで他者の利用が困難な場合もありうる。そういうときも、一人二役できる環境を用意することで、従来よりも問題を洗い出すことができるようになる。特に一人二役の場合は、次のような方法が効果的である。

(1) 機能仕様書を書いてから、テスト設計するまでに時間を空ける。
(2) 機能仕様書の書き順と異なる順でテスト設計する。

6.2. 機能仕様とテスト仕様の同時設計の効用

機能仕様とテスト仕様の同時設計の実験では、「正常系」の仕様が中心となってしまうことや、設計者は開発対象について知識を持っているため、自分の価値観で「これならわかるだろう」と言う思いがあり、「意図的に書かないケース」などがあることがわかった。つまり、設計者は『無意識にスコープを設定している』のである。ソフトウェアの全体像は、機能仕様書作成時には見えない。ソフトウェアができあがり、ユーザの操作や異常発生をイメージしたときに、初めて全体像がちらつく。このイメージを図2に示す。設計者が設計する段階では、設計者自身が持っている知識(既知の知識)の範囲を仮定しているが、ソフトウェアが徐々にできあがっていく過程で、既知の知識以外の部分(未知の知識)が増大していく。このようなスコープの変化に気づき、設計段階で定義すべきスコープを把握することが、設計品質を自らの手で向上させ機能仕様書に反映させる方策である。

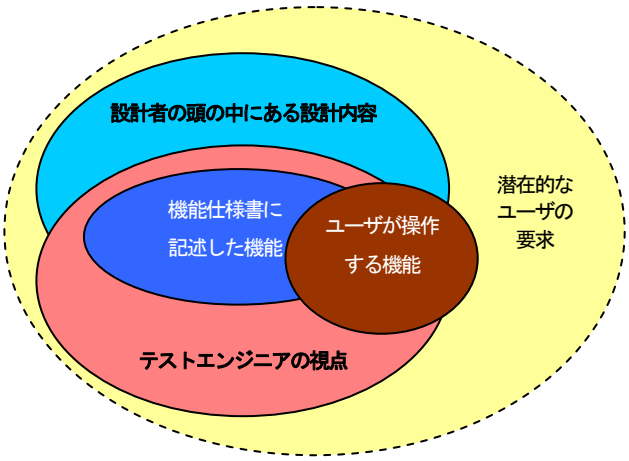

図 2. 設計者のスコープとソフトウェアの全体像

設計者が自ら設定しているスコープに気づくひとつの方法として、機能仕様とテスト仕様の同時設計が有効である。テストエンジニアに成り代わって自分の設計内容を見直すというアクティビティが、無意識に設定したスコープを設計者自身に教えてくれる。

6.3. 機能仕様書の品質の計測

今回実施した、機能仕様とテスト仕様の同時設計では、機能仕様書の記述漏れ防止の効果を出来上がったソフトウェアの不具合原因分布によって間接的に示した。すなわち、組込みソフトウェア開発に機能仕様とテスト仕様の同時設計を適用した結果、仕様記述漏れによるテスト項目漏れを防止できたことから、機能仕様書の記述漏れを防止できたと結論付けた。直接ドキュメントの品質を測定する方法としては、要求工学ワーキンググループによる要求仕様書の品質に関する研究成果報告で提案されている[9]。これは、「要求仕様書自体の品質」について、IEEE 830-1998[3]に示された指針をもとにいくつかのメトリクスを提案したものである。数量化の計算式を付録2にまとめた。この提案によると仕様書をレビューして、関連項目がいくつあったかを検出しなければならない。付録2に示す計算式はすべて、母数が「要求文の総数」となっている。つまり、記述されたものをベースに測定しており、完全性に必須の"記述されていないもの"の観点が抜けてしまっている。やはり完全性の計測は、全体が見えないため非常に難易度が高い問題であることがわかる。

7. 結論

設計者が設計工程で、自分の作成した機能仕様書の品質を向上させるには、「正しいという思い込みを排除すること」と「無意識のうちに設定しているスコープに気づき、これを広げること」である。そして、これらを実感し実践できるようにする方策として、機能仕様とテスト仕様の同時設計が効果的であることを示した。組込みソフトウェア開発では、ハードウェアとの関係から設計・実装・テストまでを分業することが困難であることが多い。このため、本研究の成果である上記2つに代表されるテストエンジニアの視点を設計者が持つことで、設計者自身が機能仕様書に品質を作り込むことが可能となる。

一方、設計者自身が陥りがちな思い込みに気づき、無意識に設定しているスコープに気づいたとしても、それらを機能仕様書にどこまで記載するかは、プロジェクトメンバーや製品によって異なってくる。今後は、このような変動要因を考慮したより効果的な適用ノウハウを研究していきたい。

参考文献

[1] 組込みソフトウェア開発における品質向上への取組み、ソフトウェア品質管理研究会 第 24 年度(2008 年度)分科会成果報告、日本科学技術連盟

[2] 共通フレーム 2007(第 2 版)、独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター編、オーム社

[3] IEEE Recommended Practice for Software Requirement Specifications, IEEE Std 830-1998

[4] 要求仕様書におけるテストエンジニアの視点を活かした欠陥検出方法の提案、 ソフトウェア品質管理研究会 第 24 年度(2008 年度)分科会成果報告、日本科学技術連盟

[5] The W-MODEL - Strengthening the Bond Between Development and Test、 Andreas Spillner、STAREast 2002

[6] ソフトウェアテスト技法ー自動化、品質保証、そしてバグの未然防止のために Boris Beizer (原著)、 小野間 彰 (翻訳)、 山浦 恒央 (翻訳) 日経 BP 出版センター

[7] プロセス改善ナビゲーションガイド～ベストプラクティス編～、独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター(編集)、オーム社

[8] ソフトウェア・テストの技法 第 2 版、Glenford J. Myers(原著)、Todd M Thomas(原著)、Tom Badgett(原著)、Corey Sandler(原著)、長尾 真 (翻訳)、 松尾 正信 (翻訳) 近代科学社

[9] 要求仕様書の品質に関する研究成果報告、情報処理学会 ソフトウェア工学研究会 要求工学ワーキンググループ、2007 年 1 月 24 日

付録１

自作したＷモデル導入ツールを以下に示す。
機能仕様書はMicrosoft Wordで作成する。テスト仕様書はMicrosoft Excelで作成し、テスト仕様書内に機能仕様書が連動して表示される仕組みとなっている。

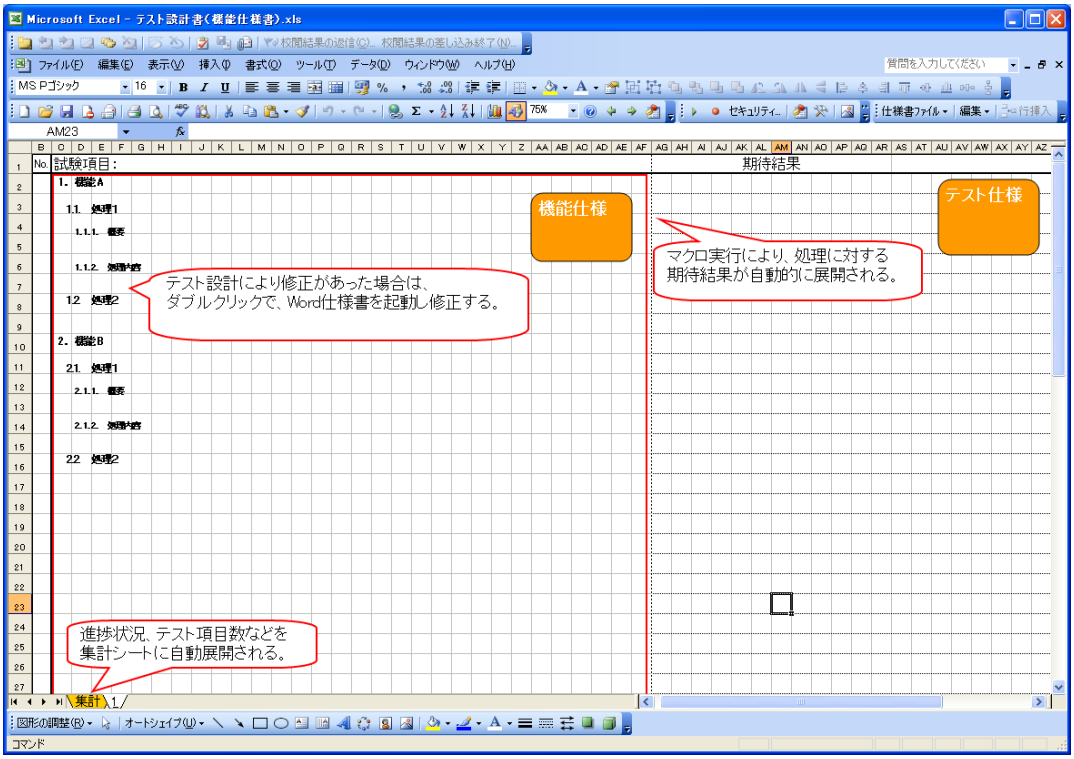

図３. テスト仕様書(機能仕様書＋テスト仕様書)

付録２　品質特性に対する指標(計算式)

表 4) 要求仕様書の品質特性に対する測定計算式[9]

品質特性	計算式
妥当性 (正当性)	要求仕様書中でソフトウェアが真に満たすべき要求文の数 / 要求文の総数
	上位の文書の記載と意味的に対応づく要求文の数 / 要求文の総数
非曖昧性	1－曖昧な要求文の数 / 要求文の総数
完全性	1－抜けがある文の数 / 要求文の総数
無矛盾性	1－矛盾に関わった要求文の数 / 要求文の総数
重要度のランク付け度	重要度のランク付けが正しく表現された要求文数 / 重要度のランク付けが必要な全要求文数
安定度のランク付け度	安定性が正しく表現された要求文数 / 変更される可能性のある全要求文数
検証可能性	1－定性的な表現箇所の数 / 定量的に表現すべき箇所の数
変更可能性	1－冗長な要求文の数 / 全要求文数
	1－依存する要求文の数 / 全要求分数
	1－複数の要求を1つの要求文とした文数 / 全要求文数
前方追跡可能性	名前と参照番号を持った要求文数 / 全要求文数
	要求仕様書を基に作られた文書から参照可能な要求文数 / 全要求分数
後方追跡可能性	要求の起源を参照可能な要求文数 / 全要求文数