派生開発でのユーザビリティの劣化を防ぐ方法

主査　　：飯泉　紀子（株式会社日立ハイテクノロジーズ）
副主査　　：足立　久美（株式会社デンソー）
アドバイザー　：清水　吉男（株式会社システムクリエイツ）
リーダー　　：西澤　賢一（GE ヘルスケア・ジャパン株式会社）
研究員　　：小野寺　聡（株式会社メタテクノ）
　　島崎　稔史（株式会社インテック）
　　吉田　和洋（東京海上日動システムズ株式会社）

研究概要

派生開発では，機能の追加・変更による振る舞いの変化が，思いもよらないユーザビリティの劣化を招くことがある．そのうえ，ユーザビリティに及ぼす影響への配慮不足により発生する不具合の多くが開発終盤やリリース後に発見され，納期遅延やコスト増大といった問題を引き起こしている．そこで我々は,機能の追加・変更による振る舞いの変化が及ぼす現状のユーザビリティへの影響を考えるきっかけと気づきを与える方法を考案した．これは過去の不具合事例から抽出した「派生開発で劣化し易いユーザビリティの観点」を用いて，機能の追加・変更により影響を受ける振る舞いの差分に焦点を当てて設計レビューする方法である．この方法を実際の不具合事例および現在進行中のプロジェクトの機能の追加・変更に対して適用した結果，ユーザビリティの劣化を防止する効果があることがわかった．

1.　はじめに

研究員各社の派生開発で課題となっている不具合の一つが，システムテストやリリース以降に見つかるユーザビリティの劣化である．派生開発におけるユーザビリティの劣化とは，現状のユーザビリティに変化が生じ，その変化が期待されたものではない場合に不具合として指摘されるデグレードを指す．既存システムへの機能追加や部分的な変更を行う派生開発では,要求側も設計側も機能の追加･変更にスコープを限定している場合が多く，機能の追加・変更が他の機能に与える影響への配慮不足に陥り易い．また，派生開発特有のユーザビリティの要求として,「明確に要求していない場合は今までと同じ」という認識が存在することにより，機能の追加・変更に伴う振る舞いの変化がユーザビリティの劣化を引き起こす要因となりえる．

我々はユーザビリティの不具合事例を持ち寄り調査・分析した．その結果，これらの不具合のほとんどが，設計段階で顧客やユーザーに予め確認し，期待と異なるようであれば追加で要求を提示してもらえば回避できたものであることがわかった．さらに設計担当者が顧客やユーザーに予め確認しなかった理由は，そもそもユーザビリティを考えていなかったり要求された機能の実装のみで満足していたりという意識の低さに原因があることもわかった.このため,設計段階でユーザビリティの劣化の可能性に気付くことができれば，手戻りによる納期遅延やコスト増大という問題を防止できると考えた．そこで，設計担当者に機能の追加・変更が及ぼすユーザビリティの劣化について考えるきっかけを与え，意識してもらう作業を設計プロセスに導入する方法を考案した．

以降，第 2 章の現状分析ではユーザビリティの不具合の分析を行い，ユーザビリティの劣化が発生する原因とユーザビリティの劣化に関する先行研究の有無を示す．第 3 章の解決策では機能の追加・変更が及ぼすユーザビリティの劣化への気づきを促すプロセスを示す．第 4 章の検証結果では，提案した解決策を用いることで，ユーザビリティの劣化を見つけることができるか検証した結果を示す．第 5 章にはまとめと今後の課題を示す．

2.　現状分析

2.1 ユーザビリティの不具合分析

派生開発でどのようなユーザビリティの劣化が発生しているかを確認するために，研究員各社の過去の派生開発で発生したユーザビリティの不具合事例を収集した（付録 A)．ここでユーザビリティの劣化とは，既にユーザーに受け入れられている現状のユーザビリティに変化が生じ，その変化が期待されたものではない場合,「使いづらくなった」とか「操作方法がわからなくなった」といった不具合として指摘されるデグレードを指す．例えば,「変更要求に応じてプルダウンメニューの選択項目を増やした．その結果，一部の項目が表示しきれず，スクロールバーが表示されるようになった．機能は満たしているので，気にせずリリースしたが，今までワンクリックで済んでいた操作が，毎回スクロールさせてから選択しなければならなくなり，以前より不便になったとの指摘を受けた」といった不具合である．

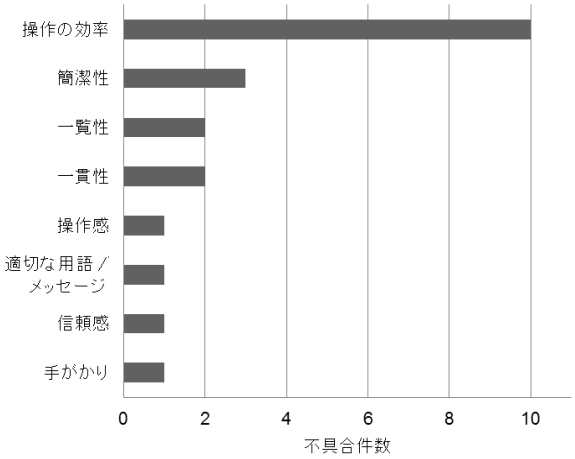

図１　ユーザビリティの不具合分析

収集した不具合がどのようなユーザビリティに影響を与えているか分析した結果を図 1 に示す．分類のカテゴリには山岡氏が提案している「70 デザイン項目」[1]を用いた．

この分析結果により，派生開発で劣化として現れやすいユーザビリティを特定した．これらはユーザーの目線でないと気が付きにくく，かつ定義しにくいものばかりであることがわかった．

2.2 なぜユーザビリティの不具合が発生するのか

これらの不具合事例の内容を詳細に見ていくと，そのほとんどが設計段階で顧客やユーザーに予め確認し，期待と異なるようであれば追加で要求を提示してもらえば回避できたものであった．

そこで，設計担当者が事前に確認しなかった理由をインタビュー調査したところ，以下のような原因があることがわかった（付録 B)．

- 無思慮　（そもそも考えていなかった）
- 自己満足（要求された機能の実装のみで満足していた）
- 思い込み（過信から，ユーザビリティへの影響確認を怠った）

すなわち，派生開発での機能の追加・変更が及ぼすユーザビリティへの影響について設計担当者の意識が低いことが意図しないユーザビリティの劣化に繋がっていた．

我々はこの「派生開発での意図しないユーザビリティの劣化を防ぐ」ということを課題とした．

2.3　先行研究

ユーザビリティの劣化の防止に関する解決策として，ユーザビリティの先行研究が適用できないかを調査した.先行研究の中ではペルソナ手法やシナリオ法といった手法が示されているが,そのほとんどが新規開発を前提としてユーザーの要求事項の明確化やユーザビリティの評価・改善を行うためのものであり，ユーザビリティの劣化の防止を目的として論じているものはなかった．

我々の課題は「派生開発での意図しないユーザビリティの劣化を防ぐ」である．そこで，この視点で派生開発とユーザビリティを扱っている先行研究を調査した．その結果，参考になる四つの論文見つけたので，調査結果を以下に述べる．

まず，複数のプロトタイピング手法から適切な手法を選択する方法を提案する研究[2]がある．この研究ではペーパープロトタイピング手法が紹介されている．この手法は，設計

時に紙を使ってユーザーが利用する画面の構成を作成し,ユーザーの要求を整理,検証するものである.この研究論文において,ペーパープロトタイピング手法は他のプロトタイピング手法に比べて,振る舞いの表現が具体的であるため,既存システムとの差異が把握し易いと論じられている.しかしながら,ユーザビリティの改善の要求そのものを作成することを目的としており,ユーザビリティの劣化を防止する視点では語られていない.

次に，設計段階で不具合の想定をして防止する手法[3]がある．この手法は，変更点に着目し，変更に関わる心配点を導出・レビューすることで未然防止を目指す DRBFM(Design Review Based on Failure Mode)[4]の心配点の導出に XDDP(eXtreme Derivative Development Process)[5] の変更要求仕様書を活用するものである．この手法を使用することにより，変更要求仕様書の変更要求と理由から，お客様視点の心配点を導出し，機能間マトリクスと機能-心配点マトリクスを作成して変更による影響箇所の不具合をなくすことができると論じられている．しかし，心配点はあくまで想定される不具合を挙げるもので，ユーザビリティの劣化を心配点として常に導出できるかという課題がある．

さらに，変更設計書を作成する段階で間接リソース変化点を抽出する手法[6]がある．この手法は，ベースシステムの時間やメモリといったリソースの限界点を抽出し,機能の追加・変更による影響が不具合となるレベルを超えていないかを確認する.しかし，リソースの限界点は，ユーザーが「ユーザビリティが劣化した」と感じる限界点と一致しない場合がある．そのため，ユーザビリティが劣化したかどうかの判断に利用することは難しい．

最後に，チェックリストを作成する手法[7][8]がある．この手法は，過去の不具合事例やレビュー指摘などをベースにチェックリストを作成し,設計段階でデグレードを防止することを目的としている.しかしながら,ユーザビリティに関するチェックリストは提案されていなかった.

上記の結果から，派生開発でユーザビリティを考えるきっかけと気づきを与えることができるような解決策を我々で検討することにした.

3. 解決策

3.1 解決の方針

第 2 章で述べた通り，派生開発において設計担当者が「無思慮」,「自己満足」,「思い込み」という意識の低さにより，ソフトウェアの振る舞いの変化がユーザビリティに影響していることを配慮せず，意図しないユーザビリティの劣化に繋がることがわかった．以下に解決の方針を示す.

まず，ユーザビリティの劣化の事例を過去の不具合から抽出し,「派生開発で劣化し易いユーザビリティの観点」に示す．この観点は，ユーザビリティの劣化を気が付けない設計担当者に，より具体的な劣化のイメージや注意点を提供する.

次に，設計担当者が「派生開発で劣化し易いユーザビリティの観点」を使ってユーザビリティの劣化を意識できるようするために，以下の二つの施策を行う.

- 機能の追加・変更による振る舞いの差分を見つけるために,「振る舞い Before/After シート」を示し，提供する.
- 「振る舞い Before/After シート」を適切なタイミングで使用してもらうためのプロセスを提供する.

以降，派生開発で劣化し易いユーザビリティの観点については 3.2 で，振る舞い Before/After シートについては 3.3 で,適切なタイミングで使用してもらうためのプロセスについては 3.4 でそれぞれ説明する.

3.2 派生開発で劣化し易いユーザビリティの観点の抽出

具体的なユーザビリティの劣化のイメージを持たない設計担当者に，ユーザビリティに影響はないかを考えさせても，ユーザビリティの劣化に気が付けないまま,「考慮はした」と満足してしまう恐れがある．そこで我々は「派生開発で劣化し易いユーザビリティの観

点」を示す必要があると考えた．

当初，不具合事例を「70 デザイン項目」で分類したものを「派生開発で劣化し易いユーザビリティの観点」に採用しようとした．しかし，「70 デザイン項目」は「このようになっていると使いやすい」といった肯定的な表現となっている．例えば「操作の効率」では「操作回数を少なくする」である．この観点をそのまま使用しても，現状のユーザビリティそのものの是非を評価してしまう恐れがあり，設計担当者に機能の追加・変更によるユーザビリティの劣化に気を付けさせることは難しい．

そこで，我々は不具合事例の分類を，ユーザー目線の言葉で否定的な表現（～させない）を用いて再定義し，さらに，表１に示すように，観点名毎に説明と具体例と「70 デザイン項目」とを関係付けて，「派生開発で劣化し易いユーザビリティの観点」として定義した．これを設計担当者自身がユーザー目線でユーザビリティが劣化していないかを考えるために利用する．

表 1 派生開発で劣化し易いユーザビリティの観点

観点名	説明	具体例	「70デザイン項目」での分類
手間をかけない	•余分に操作が増えていないか	•クリック回数が増える •画面入力が増える	•操作の効率
イライラさせない	•システムが何をしているかわからない状況を作っていないか •使いやすい画面か •安全に使えるか	•待ち時間が増える •応答が遅くなる •メッセージが意味不明 •見切れる •文字が小さい	•簡潔性 •操作の効率 •操作感
手が止まらない	•マニュアルに頼らなくてもある程度基本操作ができるか	•表現、文言が紛らわしい •次に何をしていいかわからない	•適切な用語 / メッセージ •信頼感 •一貫性
誤操作させない	•誤った操作をするようなことはないか．もし誤ってもやり直せるか	•ボタン同士の距離が狭すぎる •ボタンが小さい	•手がかり •一貫性
洗練されてない	•表現がバラバラでないか	•同じ意味でも違う用語を使っている •同じ入力項目なのにテキストボックスのサイズが異なる	•一貫性 •一覧性

3.3 ユーザビリティへの影響を確認するための振る舞い Before/After シート

ソフトウェアの振る舞いの変化がユーザビリティに影響していることから，我々は振る舞いの差分を表現する成果物を「振る舞い Before/After シート」として定義した．これを図２に示す．設計担当者がこのシートを用いることで，振る舞いの差分を明確化して，そこに注目することができる．そして，振る舞いの差分を「派生開発で劣化し易いユーザビリティの観点」で見直すことができ，ユーザビリティの劣化に気が付ける．

この「振る舞い Before/After シート」は，「振る舞いの差分」と「ユーザビリティが劣化していないかの確認結果」の二つの要素により構成される．

また，このシートは，振る舞いの差分を表現するために，変更前と変更後の振る舞いを並べて比較することによって，設計担当者が振る舞いの差分に注目できるようにしている．

このシートを作成するにあたり，設計担当者は変更前と変更後の振る舞いを把握する必要がある．我々は変更前の振る舞いを把握する手段として，既存のドキュメントが利用できないかを調べ，テストケースに着目した．ドキュメントには他にも要求仕様書や操作仕様書などがあるが，操作性，使い勝手などのユーザビリティの情報が不足していた．一方，テストケースには一つの操作に対して，いろいろなパターンが丁寧に書かれているため，ユーザーが受け入れている現状のユーザビリティを把握するために有効である．

さらに，機能の追加・変更によって，テストケースが変化する場合があることに気が付いた．つまり，テストケースが変化するということは，ユーザビリティに影響を与える可

能性があるということである.

これらのことにより，変更前と変更後の振る舞いの差分を把握する手段としてテストケースを利用することにした.

なお，近年ソフトウェアの規模が大きくなり，開発時に設計担当者とテスト担当者の分業が進んでいる.このため,設計担当者がテストケースを直接見ることが少なくなってきているが,設計担当者がテストケースを直接見る機会を作ることで,要求からは導き出せなかった振る舞いのパターンを知ることができる.

シナリオ #	テストケース#	[Before]振る舞い		[After]振る舞い	
		テスト手順	期待結果	テスト手順	期待結果
Sen.hoge.001	SWTP.hoge.001	1．初期画面のボタンAをクリックする。 2．事前に登録されている会員番号を入力する。	1．会員検索画面が表示される。 2．会員情報画面が表示される。会員名、前回の担当者の名前が表示される。	1．初期画面のボタンAをクリックする。 2．事前に登録されている会員番号を入力する。	1．会員検索画面が表示される。 2．会員情報画面が表示される。会員名、前回の担当者の名前が表示される。
	SWTP.hoge.002	1．[担当者を変更する]ボタンをクリックする。 2．担当者変更画面にて、担当科目の入力（社会）を行い、検索ボタンをクリックする。 3．担当者Aをクリックする。	1．担当者変更画面が表示される。 2．担当者検索結果一覧が表示される。列には、担当者名、国語、算数、社会というタイトルがある。検索結果では社会の欄がすべて〇である。横スクロールバーは表示されない。 3．担当者Aが選択され、会員情報画面に戻る。	1．[担当者を変更する]ボタンをクリックする。 2．担当者変更画面にて、担当科目の入力（社会）を行い、検索ボタンをクリックする。 3．担当者Aをクリックする。	1．担当者変更画面が表示される。 2．担当者検索結果一覧が表示される。列には、担当者名、国語、算数、社会、理科というタイトルがある。検索結果では社会の欄がすべて〇である。横スクロールバーは表示されない。 3．担当者Aが選択され、会員情報画面に戻る。
	SWTP.hoge.003	1．会員情報画面で空き室情報を探し、、、	1．空き室情報が、、、	1．会員情報画面で空き室情報を探し、、、	1．空き室情報が、、、

劣化しやすいユーザビリティの観点						変更による影響確認	
手間をかけない	イライラさせない	手が止まらない	誤操作させない	洗練されてない	気づき / アクション	確認状況	選択した理由
-	-	-	-	-	-	Done Not yet N/A	UIの見た目、応答時間に関連する変更なし
○	-	-	-	○	列が増えたことで各列の幅が小さくなっている。また、幅に合わせてフォントサイズが小さくなった。これにより、「洗練されていない」とユーザは感じるかもしれない / ユーザに確認する フォントサイズを維持すると、横スクロールバーが出る。スクロールさせれば閲覧可能だが、「手間がかかる」とユーザは感じるかもしれない / ユーザにフォントサイズ縮小の仕様変更を提案する	Done Not yet N/A	○○さんに確認し、表示項目は一覧できることが必須であり、フォントサイズを縮小することで、行の高さも縮んでも、視認性・統一感は維持できるとの回答を得た。これを受けて、列の高さをxx ptとすることとした。要求仕様の関連する項目にも付記する。
-	-	-	-	-	-	Done Not yet N/A	UIの見た目、応答時間に関連する変更なし

図 2「振る舞い Before/After シート」の記載例

3.4　ユーザビリティの劣化を防止するプロセス

「派生開発で劣化し易いユーザビリティの観点」と「振る舞い Before/After シート」を設計担当者に使用してもらうためのプロセスを定義した．これを図 3 に示す.

このプロセスを導入することで「派生開発で劣化し易いユーザビリティの観点」と「振る舞い Before/After シート」を設計担当者が適切なタイミングで利用することができる.

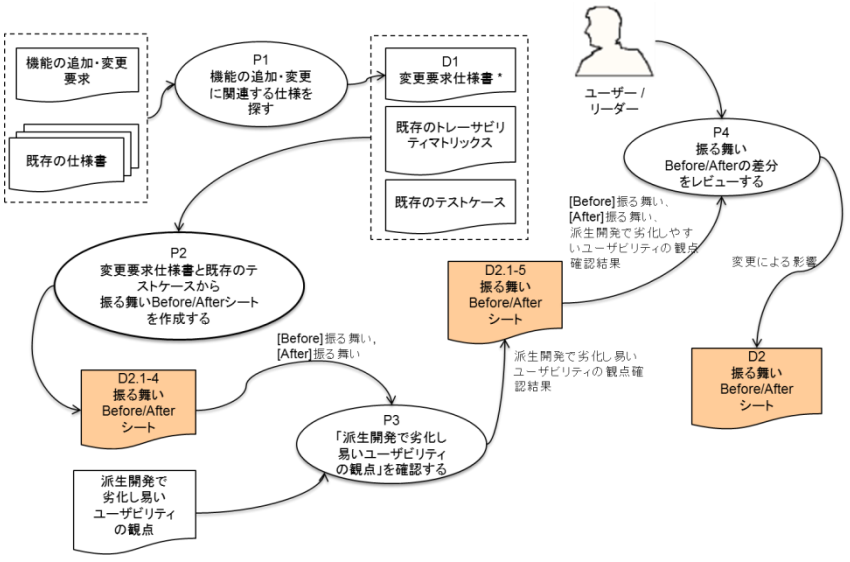

図 3 ユーザビリティの劣化の防止プロセス PFD

以下，プロセスの説明を行う．

P1:機能の追加・変更に関連する仕様を探す

変更前の振る舞いを明らかにするために，機能の追加・変更要求と既存の仕様書から，機能の追加・変更に関連するテストケースを特定する．

P2:変更要求仕様書と既存のテストケースから振る舞い Before/After シートを作成する

まず，変更要求仕様書と既存のテストケースから「振る舞い Before/After シート」を作成する．このシート上の[Before]振る舞いの各項目に以下のように記載する．

項目名	記載内容
シナリオ番号	一連のテストケースに対するユニークな番号
テストケース番号	関連するテストケース番号
テスト手順	テストケースのテスト手順
従来の期待結果	特定したテストケースから抽出した従来の期待結果

次に，作成済みの変更要求仕様書から機能の追加・変更を行った場合に変更になる振る舞いを考える．変更後の振る舞いをシート上の [After]振る舞いの各項目に以下のように記載する．

項目名	記載内容
テスト手順	変更後のテストケースのテスト手順
期待結果	変更後のテストケースのテスト期待結果

[After]振る舞いで[Before]振る舞いから変化している部分は,赤字や太字にするなど強調する．また，ユーザーインターフェースに差が出る部分があればシート上の[Before]振る舞いと[After]振る舞いの期待結果に変更前と変更後の画面イメージを作成し貼り付ける．

この過程で[Before]振る舞いのテストケースが特定できない場合には，変更前の振る舞いを設計書あるいは現行のシステムを動かして振る舞いを確認する．確認後はシート上の[Before]振る舞いに確認した変更前の振る舞いを記載する．

P3:「派生開発で劣化し易いユーザビリティの観点」を確認する

設計担当者は作成したシート上で変更により影響を受ける振る舞いの差分に対して，3.2 で述べた「派生開発で劣化し易いユーザビリティの観点」を用いて確認を実施する．当てはまる観点がある場合は，各観点に「○」を付け，「気づき/アクション」に気付いた点およびアクションを記載する．これによりテストケース毎に設計担当者がユーザビリティへの影響の見直しを行なうことができ，ユーザビリティの劣化に気が付ける．

P4:振る舞い Before/After シートの差分をレビューする

設計担当者が P3 で記載した「気づき/アクション」について問題がないか，関係者でレビューならびに確認を行う．確認した結果は「確認状況」「選択した理由」に記載する．確認ができていないものはさらなる変更が入るリスクが残っており，他にも影響する可能性があるため,すべて確認し OK が出ない限りは修正に進めないものとする．なお，関係者が変更後のユーザビリティを許容できないと判断した場合は，変更要求の見直しを行う．

4. 解決策の検証

4.1 検証内容

「無思慮」,「自己満足」,「思い込み」という設計担当者の意識の低さが，機能の追加・変更が及ぼすユーザビリティへの影響に対する配慮不足を生み出し，ユーザビリティの劣化に繋がっていた．この問題の解決策として「派生開発で劣化し易いユーザビリティの観点」,「振る舞い Before/After シート」，両者を適切なタイミングで使用する「ユーザビリティの劣化を防止するプロセス」を考案した．

そこで，この解決策が有効であるかを判断するためにシミュレーションを行い次の点について検証した．

(1) ユーザビリティの劣化を検出できるか
(2) どの「派生開発で劣化し易いユーザビリティの観点」が用いられるか
(3) 「振る舞い Before/After シート」の作成工数はどれくらいか

検証方法は「ユーザビリティの劣化が発生した過去の事例」と「現在進行中のプロジェクトの事例」に対して「振る舞い Before/After シート」を作成するという方法で行った．

検証を行ったのは各事例に関して設計・コード変更を行ったことがない開発者である．

4.2 検証結果

(1) ユーザビリティの劣化が発生した過去の事例

ユーザビリティの劣化が発生した 1 件の事例を用いて検証をした．本事例では明確なテストケースが存在せず，「[Before]振る舞いのテストケースが特定できない場合」に当てはまるので，変更前の振る舞いを現行のシステムで確認し「振る舞い Before/After シート」に記載した．そして，「派生開発で劣化し易いユーザビリティの観点」を用いて振る舞いの差分の確認を行った．その結果，本事例では「イライラさせない」，「誤操作させない」の二つの観点があてはまることが確認できた．「振る舞い Before/After シート」の作成工数は 2.0 時間であった．

(2) 現在進行中のプロジェクトの事例

現在進行中のプロジェクトでの変更要求の一つに対して「振る舞い Before/After シート」を作成することでユーザビリティの劣化を検出できるかを検証した．本事例では明確なテストケースが存在し，「振る舞い Before/After シート」は問題なく作成することができ，ユーザビリティの劣化を 3 件見つけることができた．また，本事例では「イライラさせない」の観点があてはまることが確認できた．このうち 1 件は有識者により既に指摘されていたものであったが，他 2 件は指摘されていなかった．「振る舞い Before/After シート」の作成工数は 2.5 時間であった．

4.3 考察

検証の結果，いずれの事例においてもユーザビリティの劣化に気が付けた．また，知識が十分にない分野でも既存のドキュメントからユーザビリティの劣化を見つけられることもわかった．

レビューワーには機能の追加・変更要求を出した関係者のみではなく，テスト設計に携わる者やその分野に精通した熟練者も含めるとより効果的である．なぜならば，彼らはテストケースを考える際に要求仕様から振る舞いの変化を考えることが多く，設計担当者が気付くことができなかった変更後の振る舞いを指摘できる可能性があるからである．

「振る舞い Before/After シート」の作成工数は慣れていない人でも 2-3 時間程度であることがわかった．このことから熟練者であれば，より短い時間で実行できると考える．実績データがないため比較はできないが，ユーザビリティの劣化が発生してしまってからの手戻り工数よりも小さいと考える．

本研究において提案したプロセスを導入するにあたり，プロジェクトや組織によっては発生しやすいユーザビリティの劣化は異なる可能性があるため，プロジェクトや組織に合わせてレビューの観点の増減があってもよいと考える．また，本研究では，テストケースはユーザーが受け入れている現状のユーザビリティを含むと捉えて，設計担当者が振る舞いの把握をするために利用した．このことから，今からでも操作系のテストケースを整備していくことは，機能の追加・変更に伴うユーザビリティの劣化を防ぐためにも有効な施策になると考える．

一方，現状分析で使用した過去の不具合事例には Before のテストケースが不十分で，After の振る舞いに差分が出ないもの（付録 A の No.18 参照）があった．この場合，ユーザビリティの劣化に気が付けるかどうかは，設計担当者や関係者の技術や経験，意識の高

さに依存してしまうため，本解決策では対応が難しい．

5. まとめ

5.1 結論

派生開発でのユーザビリティの劣化を防ぐことを目的として，「派生開発で劣化し易いユーザビリティの観点」，「振る舞い Before/After シート」，そして両者を適切なタイミングで使用する「ユーザビリティの劣化を防止するプロセス」を考案した．

上記三つを解決策として，過去の不具合事例や現在進行中のプロジェクトに適用したところ，「振る舞い Before/After シート」を作成することにより，機能の追加・変更による振る舞いの差分を設計担当者に意識させることができた．また，設計担当者が「派生開発で劣化し易いユーザビリティの観点」を用いて振る舞いの差分を見直すことでユーザビリティの劣化への気づきを得られることがわかった．その結果，我々の提案する解決策が，派生開発においてユーザビリティについて考え，ユーザビリティの劣化を気が付ける効果があることを確認した．

5.2 今後の課題

派生開発でのユーザビリティの劣化を防ぐという目的で本研究を進めてきた．今回の分析ではインタビューから不具合の原因を「無思慮」,「自己満足」,「思い込み」という三つで分類したが，今後はより細分化して分類する必要がある．また，検証では二つの事例に対してシミュレーションを行ったが，サンプル数が少ないので，今後より多くの事例に適用し裏付けを強化する必要がある．さらにシミュレーションだけではなく実際のプロジェクトで適用し，有効性を確認する必要がある．

謝辞

本論文の執筆にあたり，UX の過去論文の紹介や派生開発でのユーザビリティについて，多くのコメント及び参考文献を頂きました SQiP 研究会第四分科会の主査の金山豊浩氏,副主査の三井英樹氏，副主査の村上和治氏にこの場を借りて深く感謝申し上げます．

参考文献

[1] 山岡俊樹，安井鯨太，UX の構築と評価方法，SQiP 研究会特別講義，2015
[2] 南齋雄一，高尾俊之，小渕一幸，松井健吾，岡本浩，穂崎尚志，村上和治，田上貴久，中山利宏，プロトタイピング手法の効果的な選択方法の提案，日本科学技術連盟 SQiP 研究会分科会報告書，2008
[3] 安田隆司，古市祐樹，古畑慶次，変更要求仕様書を活用した未然防止方法の提案 -XDDP への DRBFM 導入とその効果-，日本科学技術連盟 SQiP シンポジウム 2011
[4] 吉村達彦，トヨタ式未然防止手法 GD3，日科技連出版社，2002
[5] 清水吉男,『派生開発』を成功させるプロセス改善の技術と極意,技術評論社,2005
[6] 小瀬聡幸，衣斐省伍，井貝智行，杉山幸雄，XDDP の変更設計書から間接リソース変化点を抽出する手法，日本科学技術連盟 SQiP 研究会分科会報告書，2013
[7] 伊藤雅子，酒井由夫，佐藤敏徳，佐藤雅思，松本拓也，組込み製品の品質を高めるための暗黙知の抽出・利用方法,日本科学技術連盟 SQiP 研究会分科会報告書,2007
[8] 関野浩之，大坪智治，外谷地茂，長友優治，XDDP によるデグレード防止効果の検証とその効果を高めるための方法，日本科学技術連盟 SQiP 研究会分科会報告書，2010