第 5 分科会

Web アプリケーション開発における画面モックアップを利用した画面仕様書およびテスト仕様書の自動生成と設計作業の効率化の提案

A Technique for Generating UI and Test Specifications from UI Mock-ups in Web Applications

主査　　秋山　浩一（富士ゼロックス株式会社）
副主査　奥村　有紀子（有限会社デバッグ工学研究所）
リーダ　坂本　一憲（早稲田大学 情報理工学専攻）
　　東海　政治（株式会社 NTT データ CCS）
　　村上　裕子（株式会社セゾン情報システムズ）
　　宮原　里枝

概要

昨今のソフトウェア開発では、品質を確保した上で、コスト削減や短納期化を求められている。短絡的な工数の削減の方法として、画面仕様書とテスト仕様書の作成が省略されることがある。しかし、これらの仕様書の作成を省くことは、テスト品質やソフトウェア品質の低下を招きかねない。最終的に、後工程に進むほど欠陥への対処が困難になることから、ソフトウェアの開発コストを増大させる。

我々は、上流工程で作成される画面モックアップへ UI の付加情報を追記することで、画面モックアップから画面仕様書とテスト仕様書を自動生成する手法を提案する。自動生成可能な情報には限りがあり、テスト仕様書を人手で修正する必要がある。そのため、提案手法は仕様書を再生成する際に、修正による差分を維持する。評価実験にて、提案手法が仕様書の作成コストを低減して、画面モックアップと画面仕様書、テスト仕様書間のトレーサビリティの維持を確認する。この結果から、画面仕様書とテスト仕様書の作成を促し、これらの作成を省く従来の開発プロセスを改善する。

Abstract

Software development requires to reduce development cost and shorten development period without quality loss. The simplistic way to reduce development cost skips the process of creating an UI specification and a test specification. However, the skipping causes the issues of testing quality loss and development quality loss. The issues conclusively increase development cost because it is difficult to deal with defects of software by advancing development.

We proposed a technique which automatically generates an UI specification and a test specification from an UI mock-up (or mock in what follows) which is created in the upper process. Our technique requires to write additional information in mocks for auto-generation. Moreover, our technique requires to edit a generated test specification because of the limitation of the auto-generation. Therefore, our technique keeps the difference of the change by developers in regeneration. Our experiments shows that our technique reduces the cost of creating an UI specification and a test specification and keeps traceability between mock, the UI specification and the test specification. Therefore, our technique helps to create an UI specification and a test specification to improve existing development process.

1. はじめに

近年、Web アプリケーションの大規模化・複雑化に伴って、仕様の理解や共有が困難になっている。一方で、システムの開発期間の短縮化が進み、例えば、開発期間が 3 ヶ月以内の案件は、2006 年度時点では全体の 15%ほどであったのに対し、2009 年には 42%にまで増加した[1)]。

こうした中、 Web アプリケーション開発（以降、Web 開発）では工数削減の為に、画面仕様書とテスト仕様書（テストケースの作成に必要な情報を載せた資料）の作成を省いた開発プロセス（以降、プロセス）が採用されることがあった[2)]。そのようなプロセスでは、実装した UI を元にテストケースを作成するために仕様自体の確認機会を失うこともあり、仕様と実装の差異をソフトウェアテスト（以降、テスト）で検証できないという問題を引き起こす。

また、画面仕様書とテスト仕様書を作成した場合であっても、各仕様書もしくは実装の一部が変更された時に、これらのトレーサビリティが維持されないことがある[3)]。このことは各仕様書と実装間の矛盾をもたらすとともに、ソフトウェアの理解性を低下させる。ひいては、仕様にそぐわないテスト実施の誘因となり、テストの品質低下を引き起こす。

Web 開発では、画面に関する要求獲得において画面モックアップ（以降、モック）を使用する。モックとは設計の早い段階で UI の外観を試作したプロトタイプである。プロセスの上流工程においてモックを作成して、以降の工程においてモックを保守し続けているプロジェクトが存在する[4)]。しかし、モックは仕様書の作成に必要な UI コンポーネントの詳細な情報を含まない。

そこで、我々は画面仕様書に必要な情報をモックに埋め込み、モックから画面仕様書とテスト仕様書を自動生成する手法を提案する。提案手法では自動生成された画面仕様書およびテスト仕様書に人手により詳細情報を追記する事で、より正確な画面仕様書およびテスト仕様書が得られることを示す。また、提案手法の検証実験において、仕様書の作成と保守コストの低減結果を示す。それにより、画面仕様書およびテスト仕様書が作成され、利用されるプロセスの実現を示す。

2. 問題

本節では、既存手法における 2 つの問題点について説明する。

P1）画面仕様書とテスト仕様書が作成されない

開発期間の短縮化を理由に、画面仕様書とテスト仕様書の両方もしくは片方が作成されない場合や、テスト仕様書においては UI が出来上がってからそれを元に作成される場合がある。画面仕様書が作成されない場合、実装した UI が仕様とみなされる。そのため、仕様にそって UI が実装されたなかった際、実際の仕様との間にずれを発生させる。一方、テスト仕様書が作成されない場合、UI におけるテスト対象をテストエンジニアの視点で詳細に分析したドキュメントがないため、テストケースの抜けや漏れを発生させる。これらの理由から、仕様書を作成しないことで開発初期のコストを低減できるが、テストの品質低下、さらに、ソフトウェアの品質をも低下させ、それに伴う欠陥の増加から最終的な開発コストを増大させる[5)]。

したがって、画面仕様書やテスト仕様書の作成を必須とし、しかも手間をかけずに納期という制約を守ることが可能となるようなプロセス手法を立案する必要がある。

P2）画面仕様書とテスト仕様書のトレーサビリティの維持が難しい

Web アプリケーションの開発を進めていく過程で、UI については、実装して初めてわかる操作性の問題があるため、UI の仕様変更が頻繁に発生する。こうした中で、人手によりモックと仕様書の作成と保守を行う場合、人的ミスが発生すると、モック、画面仕様書、テスト仕様書のトレーサビリティの維持が難しくなる。例えば、モックとテスト仕様書の担当者が異なり、モックの修正した際に、テスト仕様書の担当者にその旨を伝え忘れた場合、担当者はモックへ加えた修正をテスト仕様書へ反映できない。このように、モックと画面仕様書、テスト仕様書間でトレーサビリティが維持されないと、テスト仕様書を元に作成したテストと、モックとの間に矛盾を引き起こすという問題につながる。

したがって、トレーサビリティを維持するために、人手による修正ミスの伝播を避けられるような手法が必要である。

3. 提案手法の概要

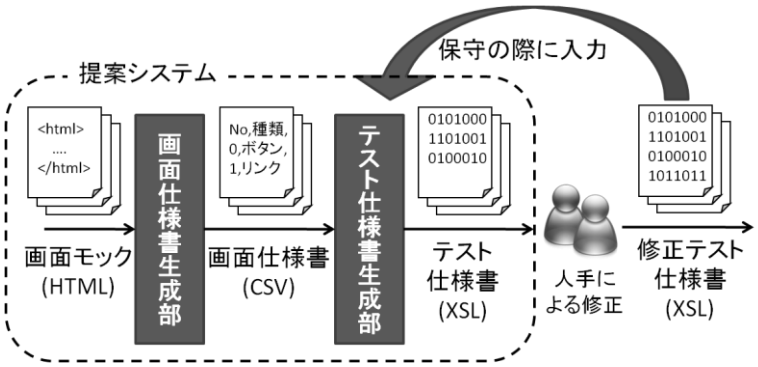


図 1. 提案する手法のプロセス

図 1 にて、提案手法を実現するために考案したプロセスの構成を示す。提案手法は、画面仕様書生成部とテスト仕様書生成部の 2 つの自動化システムによって構成される。従来手法では、モックから UI が実装され、実装された UI を用いてテストケースが作成されていた。なお、付録 1 にて従来手法と提案手法の比較を行う。提案手法では、従来のモックに加えて、特定のタグと我々が定義した構文で書かれたコメントで画面仕様書の生成に必要な情報を表現する HTML ファイルをモックとする。各自動化システムの詳細については第 4 節で説明する。

提案手法の活用場面は、新規にモックから画面仕様書とテスト仕様書を生成する場合、モックの修正を生成済みの画面仕様書とテスト仕様書に反映する場合の 2 通りがある。新規にモックから画面仕様書とテスト仕様書を生成する手順は以下のとおりである。

1. 特定のタグと我々が定義した構文のコメントで画面仕様書の生成に必要な情報を追記したモックの HTML ファイルを作成する
2. 画面仕様書生成部にモックを入力して、画面仕様書の CSV ファイルを自動生成する
3. テスト仕様書生成部に画面仕様書を入力して、テスト仕様書の XLS ファイルを自動生成する
4. 必要であれば、生成したテスト仕様書に仕様を追記する

一方、仕様に変更が生じた際は、モックを修正した上で提案手法を適用することで、各仕様書に修正を反映できる。なお、手順 3 の際に、以前作成したテスト仕様書生成部を合わせて入力することで、以前手順 4 で加えた追記内容が維持されたままテスト仕様書が生成される。

4. 提案手法の詳細

4.1. モックの HTML ファイルの作成

画面仕様書生成部に入力するモックは、特定のタグに対して我々が定義した構文で記載されたコメントが追記された HTML ファイルである。UI コンポーネントは input、a、span によって表現する。上述した 3 種類のタグにおいて、属性からは抽出できない情報をコメントから抽出する。図 4 で ISO/IEC 14977:1996 の EBNF（Extended Backus-Naur form:拡張バッカス・ナウア記法）を用いて情報を抽出可能なコメントの構文を示す[7]。<remark>は注釈、<attribute>は項目名と値の対、<AttributeName>は項目名、<AttributeValue>は項目値、<WS>は空白を示す。例えば、<!-- 名前 = 出発駅 --> は名前の値が出発駅であることを示す。なお、真偽値を取る項目名の場合、<AttributeValue>が no という文字列であれば偽値、それ以外の文字列は真値である。

```
<remark> ::= "<!--" <attribute> (",," <attribute>)* "-->"
<attribute> ::= <WS>* <AttributeName> <WS>* "=" <WS>* <AttributeValue> <WS>*
<AttributeName> ::= 27 項目のうちのいずれかの項目名
<AttributeValue> ::= 項目値を表す文字列
<WS> ::= 半角空白 | 全角空白 | タブ
```

図 4. 情報を抽出可能なコメントの EBNF

4.2. 画面仕様書生成部

画面仕様書生成部は、.NET Framework 4.0 で実装した。画面仕様書生成部は、モックの HTML ファイルを解析して、画面仕様書を CSV ファイルで生成する。具体的には、1 つのコメントから input タグ、span タグ、a タグのいずれかのタグに沿って前に遡っていき、コメント以外が現れるまで記載されているコメントを対象として、<remark>の構文で表記されているコメントを解析する。タグの属性とコメントで同じ項目名について項目値を設定した場合、コメントの項目値が優先される。なお、属性でもコメントでも項目値が設定されない場合は、空文字もしくは偽値を取る。例えば、図 5 の HTML を解析した場合、種別が「ボタン」、名前が「前へ」、遷移先が「index.html」、遷移方法が「新規ウィンドウ」、必須項目が「真値」、入力が「真値」、出力が「真値」、それ以外の項目は空文字もしくは偽値である UI コンポーネントとして抽出する。

```
<!-- 遷移方法=新規ウィンドウ -->
<!-- 名前=前へ, 遷移先=index.html -->
<!-- 必須項目=X -->
<input type="button" name="次へ">
```

図 5. HTML ファイル断片の解析例

以上の方法から、画面仕様書生成部は入力されたモックの HTML ファイルから UI コンポーネントの情報を抽出する。その上で、1 行目に項目名、2 行目以降から各 UI コンポーネントの各項目の項目値を記載した CSV ファイルを出力する。真値は「●」記号でその属性が受け入れられることを示し、偽値は空文字で出力する。このようにして、モックから画面仕様書を自動生成する。

画面仕様書は、各 UI コンポーネントの性質に関する 27 項目の項目値を記載した CSV ファイルで表現される。具体的な項目については付録 1 に示す。

モックの作成は自動化が難しく、人手が必要な作業である。そのため、提案手法ではモック作成時に画面仕様書の生成に必要な情報も記載することで、画面仕様書の作成を完全に自動化した。これによって、画面仕様書の作成に必要なコストはモックへの情報追記のみとなったが、生成された画面仕様書に情報を追加することによって、より品質の高い画面仕様書に仕上げることができる。例えば、画面仕様書の「備考」の部分に、モックに記載しなかった、あるいは書ききれなかった内容を追記したり、画面仕様書を読む側にとって理解しやすい文面に変更するなど、付加価値として加えることができる。

4.3. テスト仕様書生成部

テスト仕様書生成部は、Microsoft Excel の Visual Basic for Applications で実装した。テスト仕様書生成部は、画面仕様書生成部によって生成された画面仕様書の CSV ファイルを入力して、テスト仕様書を XLS ファイルで出力する。テスト仕様書はテストケースを作成する際に利用される。テスト仕様書は、各 UI コンポーネントに対し、8 つの項目を記載したものである。8 項目とは、項番、アイテム名、大項目、中項目、小項目、検証内容、検証結果、備考である。これらの項目は、IEEE 829 を基に網羅性を踏まえて決定した。[6)]

テスト仕様書生成部は、各 UI コンポーネントに対して、項番、アイテム名、大項目、中項目、小項目、検証内容の内容を自動生成する。項番は項目の番号であり 1 から連番で番号が振られる。アイテム名は UI コンポーネントの名前を記載する。各項目の具体的な内容については付録 2 に示す。これらの内容は、画面仕様書にて各 UI コンポーネントの性質に関する各項目の項目値を解析して生成する。なお、検証結果と無効系の検証内容の内容は自動生成しないため、別途入力する必要がある。

テスト仕様書生成部では、無効系の検証内容と、テスト実施の結果である検証結果をユーザが入力する必要があり、テスト仕様書に新たにユーザが書き加える有効、無効の検証内容が存在することを想定している。例えば、モック上のボタンやテキストボックスでは実施されない処理、実際には行わない処理、イレギュラーな動作が動作しないことを確認する必要がある場合は、無効系の検証内容に情報を追加する。

このような手動で加えた情報は、モックや画面仕様書に修正が加わり、テスト仕様書を再生成する必要がでてきた場合に、自動的に再生成したテスト仕様書に反映する必要がある。

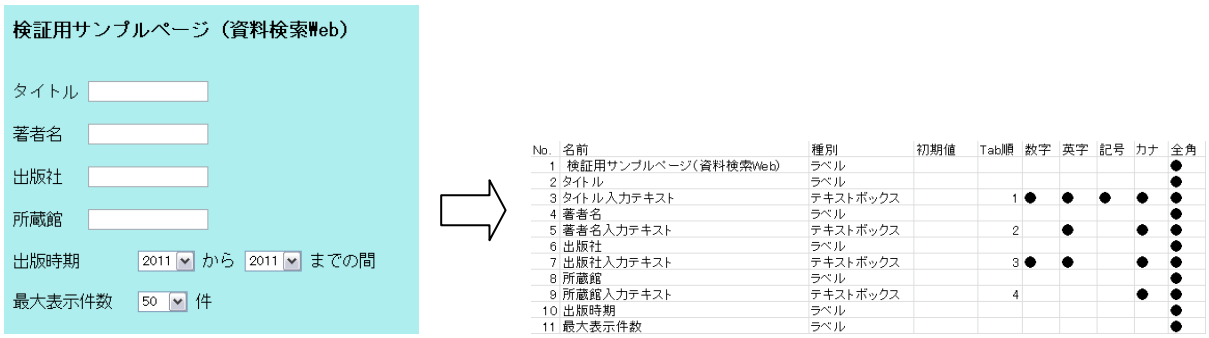
テスト仕様書生成部は、再生成された画面仕様書と手動で修正を加えたテスト仕様書の両方を読み込む。そして、テスト仕様書生成部が自動的に記憶している手動で修正を加える前のテスト仕様書と、手動で修正を加えたテスト仕様書の差分を計算して、テスト仕様書に手動で加えられた修正内容を抽出する。その後、画面仕様書からテスト仕様書を再生成して、その上で、抽出した修正内容を反映する。この方法により、手動で加えた修正を維持しながら、テスト仕様書のトレーサビリティを確保することができる。

5. 検証と結果の取得

今回の検証では、実験用に開発したツールとマクロを利用して、テスト仕様書を一部自動生成した場合と全てを手作業で生成した場合とで、それぞれの項目数とかかった時間を明確にする。

ただし、使用するモックと画面仕様書も実験用に作成したものであり、実際の開発規模にそぐわないことは否めないが、定量的な結果を取得することで、現場に合わせた規模で行った場合の目安と成り得ると考えた。また、同じような悩みを抱えていて、一部でも自動化ツールの導入を検討している現場に対し、今回の実験結果が有用な情報と成ることを目的としている。

実験 1)



No.	名前	種別	初期値	Tab順	数字	英字	記号	カナ	全角
1	検証用サンプルページ(資料検索Web)	ラベル							●
2	タイトル	ラベル							●
3	タイトル入力テキスト	テキストボックス		1	●	●	●	●	●
4	著者名	ラベル							●
5	著者名入力テキスト	テキストボックス		2		●		●	●
6	出版社	ラベル							●
7	出版社入力テキスト	テキストボックス		3	●	●		●	●
8	所蔵館	ラベル							●
9	所蔵館入力テキスト	テキストボックス		4				●	●
10	出版時期	ラベル							●
11	最大表示件数	ラベル							●

図 6. モックから画面仕様書を生成した例

図のとおり、モックから画面仕様書がすでに生成されていると仮定し、この画面仕様書から新規でテスト仕様書を生成するまでについて、項目数とその所要時間を測定する。

結果 1)

画面仕様書から約 1 分程度で 165 項目のテスト仕様書 XLS ファイルを自動生成できた。得られたテスト仕様書は表 1 である。

表 1. 自動生成したテスト仕様書の一部

項番	アイテム名	大項目	中項目	小項目	検証内容
1	検証用サンプルページ(資料検索Web)	種別	無効系		
2	検証用サンプルページ(資料検索Web)	字種	有効系		全角文字形式であるか
3	検証用サンプルページ(資料検索Web)	字種	無効系		
4	検証用サンプルページ(資料検索Web)	必須	有効系		このアイテムを入力しなくても、次の画面へ進むことができるか
5	検証用サンプルページ(資料検索Web)	必須	無効系		
6	検証用サンプルページ(資料検索Web)	文字寄せ	有効系		このアイテムは左寄せで表示されているか
7	検証用サンプルページ(資料検索Web)	文字寄せ	無効系		
8	検証用サンプルページ(資料検索Web)	入出力	有効系		このアイテムは出力専用のアイテムであるか
9	検証用サンプルページ(資料検索Web)	入出力	無効系		
10	タイトル	種別	有効系		このアイテムはラベル形式であるか
11	タイトル	種別	無効系		
12	タイトル	字種	有効系		全角文字形式であるか
13	タイトル	字種	無効系		
14	タイトル	必須	有効系		このアイテムを入力しなくても、次の画面へ進むことができるか
15	タイトル	必須	無効系		
16	タイトル	文字寄せ	有効系		このアイテムは左寄せで表示されているか

自動生成されたテスト仕様書は、UI コンポーネントに関連する有効系の検証内容と入力パラメータに関連する上限値と下限値の検証内容が含まれ、テストケース作成時にそのまま参照可能な項目数は 88 項目であった。なお、無効系の検証内容は空欄ではあるが、必要に応じてユーザが手動で追記可能となっている。

続いて、表 1 で得られたテスト仕様書へ、検証内容が空欄となっている無効系項目を手動で 77 項目追記した。これらの所要時間は、モックと画面仕様書から読み取れる有効系項目自動生成結果の確認時間に約 30 分、無効系項目の手動追記時間に約 30 分であった。したがって、画面仕様書から最終的なテスト仕様書が得られるまで約 1 時間かかったことになる。一方、画面仕様書から全て手作業で同等のテスト仕様書を作成した場合、約 2 時間かかった。これらの結果により新規でテスト仕様書を作成するプロセスにおいては、約 1 時間の作業時間短縮に成功したと言える。

実験 2)

次に、実験 1 のモックと画面仕様書へ修正があったと仮定し、手動で追記したテスト仕様書へ、自動で修正箇所を反映させることで、すでに存在するテスト仕様書の項目へ影響していないことの確認と実験 1 と同様に項目数とその所要時間を測定する。

結果 2)

約 15 分程度モックを修正して、画面仕様書から約 5 分程度で 160 項目が追加されたテスト仕様書 XLS ファイルを自動生成できた。得られたテスト仕様書は表 2 である。

表 3.自動生成したテスト仕様書の一部（修正反映）

項番	アイテム名	大項目	中項目	小項目	検証内容
1	検証用サンプルページ(資料検索Web)	種別	無効系		ラベル形式ではない状態
2	検証用サンプルページ(資料検索Web)	字種	有効系		全角文字形式であるか
3	検証用サンプルページ(資料検索Web)	字種	無効系		文字列の間に余分なスペースなどが入っている状態
4	検証用サンプルページ(資料検索Web)	必須	有効系		このアイテムを入力しなくても、次の画面へ進むことができるか
5	検証用サンプルページ(資料検索Web)	必須	無効系		このアイテムを入力しない場合、次の画面へ進むことができないか
6	検証用サンプルページ(資料検索Web)	文字寄せ	有効系		このアイテムは左寄せで表示されているか
7	検証用サンプルページ(資料検索Web)	文字寄せ	無効系		センタリングされている状態
8	検証用サンプルページ(資料検索Web)	入出力	有効系		このアイテムは出力専用のアイテムであるか
9	検証用サンプルページ(資料検索Web)	入出力	無効系		入力専用アイテムの状態
166	送信ボタン	種別	有効系		このアイテムはボタン形式であるか
167	送信ボタン	種別	無効系		
168	送信ボタン	初期値	有効系		初期値は「検索開始」であるか
169	送信ボタン	初期値	無効系		
170	送信ボタン	タブオーダ	有効系		タブキーを押下すると、8番目にこのアイテムへ遷移するか
171	送信ボタン	タブオーダ	無効系		
172	送信ボタン	字種	有効系		全角文字形式であるか

得られたテスト仕様書へ、検証内容が空欄となっている無効系項目を手動で 69 項目追記した。これらの所要時間は、実験 1 において手動で追記した項目に影響がないことの確認時間とモックと画面仕様書から読み取れる有効系項目自動生成結果の確認時間に約 30 分、無効系項目の手動追記時間に約 30 分であった。したがって、画面仕様書から最終的なテスト仕様書が得られるまで約 1 時間かかったことになる。一方、画面仕様書から全て手作業で同等のテスト仕様書を作成した場合、約 2 時間かかった。これらの結果から、修正反映をしたテスト仕様書の作成プロセスにおいても、約 1 時間の作業時間短縮に成功し、かつ、修正による差分の維持を確認できた。

6. 結果

第 5 節の実験結果を踏まえて、第 2 節の問題点を参照して提案手法の解決結果を述べる。

S1）モックから画面仕様書とテスト仕様書の自動生成

提案手法では、モックにコメントを追記することで、画面仕様書生成部と、テスト仕様書生成部で画面仕様書とテスト仕様書に必要な情報を抽出して、これらの仕様書を自動生成する。画面仕様書は、モックに適切なコメントを記載することで、完全に自動生成できる。テスト仕様書は、有効系を中心とした内容が自動生成されるため、無効系については人手で追記する必要がある。また、無効系の情報を人手で追記した場合であっても、実験結果では作業時間の約 50%の削減に成功している。したがって、提案手法によるコスト削減により、画面仕様書とテスト仕様書を作成する改善プロセスの採用を支援して、P1 の問題を解決する。

S2）人手で加えた修正を維持したテスト仕様書の再生成

提案手法では、テスト仕様書に人手で加えた修正を維持した状態で、再生成する仕組みを持つ。したがって、モックや画面仕様書が変更された際であっても、人手で加えた修正を自動的に反映

させることができる。実験結果では、約 40%の削減に成功している。したがって、提案手法による人手で加えた修正の自動的な反映から、モックや仕様書間のトレーサビリティの維持を実現して、P2 の問題を解決する。

7. まとめと今後の展望

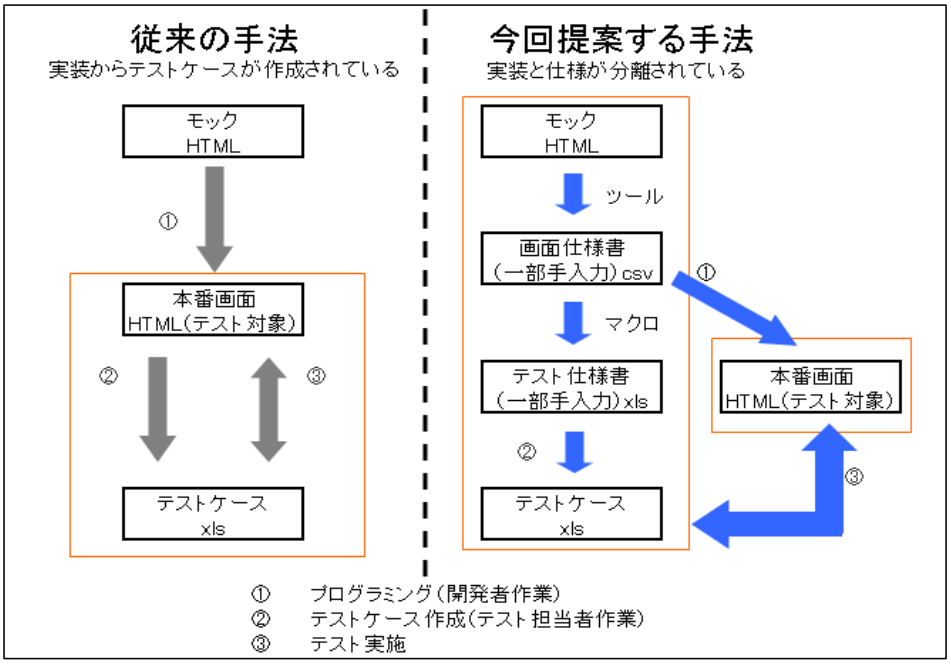
我々は、モックから画面仕様書とテスト仕様書を自動生成する手法を提案した。提案手法では、モックである HTML ファイル中のタグと我々が定義した構文に則ったコメントを解析して、画面仕様書とテスト仕様書を自動生成する。これにより、画面仕様書とテスト仕様書を作成する敷居を下げ、テスト仕様書が作成されないプロセスを改善することが期待できる。また、テスト仕様書を人手で修正した場合、修正内容を維持した状態でモックからテスト仕様書を再生成する。これにより、モックと画面仕様書、テスト仕様書間のトレーサビリティが維持される。従来、テスト仕様書は開発期間の短縮化を理由に作成されていなかったが、本来は成果物として存在しなければならないものである。評価実験では、テスト仕様書を作成することを前提に、提案手法を用いた場合と用いなかった場合で比較して、テスト仕様書の作成時間を約 50%、保守時間を約 40%短縮した。実験結果と既存手法の問題点を照らし合わせ、提案手法の実用性を示した。また、実装した UI を元に作成したテスト仕様書と比較して、提案手法によって得られたテスト仕様書は、モックから抽出された情報を元に人手による修正が行われているため、品質が上がっている。

今後の展望として、モックから画面仕様書のみならず、UI のソースコードを自動生成することが考えられる。例えば、UI コンポーネントの最大文字列長が性質としてコメントから抽出できた場合、最大文字列長を超える文字列を入力できないようにモックの HTML ファイルに JavaScript のコードを埋め込むといった手法が考えられる。これによって、モックからソースコードと仕様の両方を自動生成するような、モック駆動開発手法が実現できる。

8. 参考文献

1) JISA 社団法人 情報サービス産業協会: 情報サービス産業における技術動向調査 2009, JISA 会報, No. 99, pp. 113-124, Oct 2010.
2) 川平 航介、長田 晃、海谷 治彦、北澤 直幸、海尻 賢二: 要求追加によるインパクトの分析に基づく組込みソフトウェア開発の効率化, 第 159 回 ソフトウェア工学研究発表会, pp. 17-24, Mar 2008.
3) 萱嶋 志門、玉木 裕二: ソフトウェア設計方法論の開発と適用, 東芝レビュー, Vol. 61, No. 1, Jan 2001.
4) 水島壮太: モック・オブジェクト・アプローチによる Web アプリケーション単体テストの生産性向上, PROVISION, No. 57, Spring 2008.
5) Barry Boehm and Victor Basil: “Software Defect Reduction Top 10 List”, IEEE Computer, Jan 2001.
6) IEEE Std. 829-2008, IEEE Standard for Software and System Test Documentation.
7) R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, T. Berners-Lee: "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, U.C. Irvine, DEC W3C/MIT, DEC, W3C/MIT, W3C/MIT, Jan 1997.

付録1 従来の手法と今回の手法のプロセスの違い



付録2 CSVの項目と、HTMLのタグ等との対応表

画面仕様書	HTMLファイル	
	HTMLコメント	HTMLタグ
No.※1	―	―
名前	名前=	inputタグのname属性
		spanタグのname属性
		aタグのname属性
種別※2	―	inputタグのtype属性
		spanタグであればラベル
		aタグであればリンク
初期値	―	inputタグのvalue属性
Tab順	Tab順=	―
数字	数字	―
英字	英字	―
記号	記号	―
カナ	カナ	―
全角	全角	―
日付	日付	―
その他	その他	―
必須項目	必須項目	―
最小文字数	最小文字数=	―
最大文字数	最大文字数=	―
最小値	最小値=	―
最大値	最大値=	―
文字寄せ	文字寄せ=	―
フォーマット	フォーマット=	―
遷移先	―	aタグのhref属性
遷移方法	遷移方法=	―
入力	―	inputタグ
出力	―	●
DB入力	DB入力	―
DB出力	DB出力	―
DB項目名	DB項目名	―
備考※3		

※ 1取得した情報から順に展開されるため、Noは自動的に割り当てられる
※ 2種別の項目値は、ラベル、リンク、テキストボックス、パスワード、ファイル選択、
チェックボックス、ラジオボタン、Hidden、送信ボタン、リセットボタン、ボタン、
イメージボタンのいずれかの文字列である
※ 3ユーザが自由に記載する項目

付録3 テスト仕様書生成部の内容

項目	内容	例
アイテム名	UIコンポーネントの名前	テキストボックス1、ラベル、ラジオボタンA
大項目	テスト時に注目するUIコンポーネント	種別、初期値、タブオーダ、字種、最大文字数
中項目	大項目を補う	有効系、無効系、上限値、下限値、範囲外
小項目	中項目でも性質を表現しきれない場合に中項目を補う	大項目が遷移先、中項目が形式、小項目が有効系など
検証内容	大項目と中項目を踏まえた具体的な検証内容と期待結果	画面が遷移する、テキストが入力できる
検証結果	実際の検証結果（自分で記載）	—