# モデル駆動開発におけるユーザビリティ機能を実装したWebプロトタイプの自動生成

紙森 翔平[1]　小形 真平[1]　海尻 賢二[1]

**概要**：近年，業務系 Web アプリケーション開発ではユーザビリティが重視されている．例えば，ユーザーの入力に即時反応し，ユーザビリティを高める機能（以降，ユーザービリティ機能）の要求は，アプリケーションを実際に操作しなければ妥当性を十分に確認することは難しい．そのため，特に実行可能な成果物がほとんどでないであろう上流工程におけるユーザビリティ機能の要求確認の支援は必要である．本論文では，上流工程におけるユーザビリティ機能の要求確認の支援を目的とし，画面遷移やユーザビリティ機能を表す UML モデルからユーザビリティ機能を付加した Web プロトタイプを自動生成する手法を提案する．本論文では，提案手法の有用性を業務系 Web アプリケーションに適用することで評価する．

## 1. はじめに

ユーザビリティとはエンドユーザーが効率よく目的を達成することに関する品質であり，ISO25010[1] や ISO9241-11[2] で標準化されている．近年，業務系 Web アプリケーション開発ではユーザビリティが重視されており，ユーザビリティを高める機能（以下，ユーザビリティ機能）の導入は必須である．ユーザビリティ機能は，例えば，エンドユーザーの入力に即時反応する入力補完機能（例：Google のトップページ）があり，既存である．本機能はビジネスロジックと同様にデータベースやユーザーの権限などのシステムの核に係わるため，開発後期の大きな手戻りを防止できるように要求の妥当性を十分に確認する必要がある．しかし実際の開発では，ユーザビリティ機能に比べてビジネスロジックやデータ構造の分析・設計が優先されやすく，ユーザビリティの専門家とビジネスロジックの開発者が効率的に協業できない現状がある [3]．

故に，ユーザビリティ機能の要求確認を開発初期から効率的に支援できるように以下の課題を解決すべきである．

(1) ユーザビリティ機能はビジネスロジックと異なり，エンドユーザーの嗜好が反映されやすいため，エンドユーザーが早期にシステムを操作して十分に要求を確認できる必要がある．
(2) ビジネスロジックの開発者は，エンドユーザーに理解容易なユーザビリティ機能の要求確認用成果物を効率的に開発できる軽量な方法が必要である．
(3) 上流工程でユーザビリティ機能の仕様を効率的かつ反復的に洗練できるように，当該機能が必要か否かを見極める段階や，必要と判断した当該機能の詳細仕様を決定する段階など，仕様を詳細化するプロセスに応じた効率的な仕様の定義方法が必要である．

そこで，本論文ではユーザビリティ機能の要求確認支援を目的とし，既存の画面遷移モデル [6] を変更したモデル，およびそこに連携できるユーザビリティ機能のモデルを提案する．そして，以下の 2 種類のユーザビリティ機能付きプロトタイプ生成方法を実現する．第 1 に両モデルからユーザビリティ機能が付加されたプロトタイプを自動生成する．第 2 に両モデルと対応づく既存のプロトタイプにユーザビリティ機能を付加する．

前述の第 1 の課題は，ユーザビリティ機能付きのプロトタイプの開発により解決する．第 2 の課題は，既存のプロトタイプが存在しても，ユーザビリティ機能を付加したプロトタイプを生成できる仕組みにより解決する．第 3 の課題は，両モデルを連携させる少量の記述だけでユーザビリティ機能をプロトタイプに付加する方法を実現し解決する．

本論文では提案手法の実現性を，図書館システムの OSS である Next-L Enju[4] を適用事例として確認した．その結果，提案手法により適用事例にユーザビリティ機能が正しく自動付加されることを確認した．

## 2. 用語説明

### 2.1 ユーザビリティ

ユーザビリティとは ISO9241-11 において，“ある製品

[1] 信州大学理工学系研究科
Graduate School of Science and Technology, Shinshu University

© 2014 Information Processing Society of Japan

が，指定された利用者によって，指定された利用の状況下で，指定された目的を達成するために用いられる際の，有効さ，効率および利用者の満足度の度合い” と定義されている．また，度合いを測る 3 要素の定義は以下である．

**有効さ (effectiveness):** 利用者が指定された目標を達成する上での正確さ、及び完全さ

**効率 (efficiency):** 利用者が指定された目標を達成する上での正確さおよび完全さに関連して費やした資源

**満足度 (satisfaction):** 不快さのないこと，および製品仕様に対しての肯定的な態度

本論文では，“有効さ” は入力補完機能などのユーザビリティ機能に着目して向上を支援し，“効率” はエンドユーザーの入力に即時反応する機能に着目して向上を支援する．これらの結果，エンドユーザーの “満足度” が向上できることが期待される．

### 2.2 プロトタイプ

プロトタイプとは，システムの一部を紙や電子媒体で表現した試作版システムである．ユーザーはプロトタイプを実際に操作することにより，要求の妥当性を十分に確認しやすくなり，結果として開発後期の手戻りの防止に繋がる．

本論文では，ユーザビリティ機能を実装し，画面遷移や入出力項目を表す簡潔な画面の HTML，CSS および JavaScript コードをプロトタイプと呼ぶ．

### 2.3 MDA

MDA（Model Driven Architecture，モデル駆動アーキテクチャ）とは，Object Management Group[5] が発表したソフトウェア設計手法である．プログラミング言語や OS などの特定のプラットフォームに依存しない PIM（Platform Independent Model）から，プラットフォームに依存する PSM（Platform Specific Model）へ変換する手法である．通常，変換は自動生成で行い開発コストを削減する．

近年では，スマートフォンの登場により，各種 OS に特化した複数種のブラウザにて利用できる 1 つの Web アプリケーションを開発する機会が増加している．そのため，開発の効率化を目的とし，個々のプラットフォームに依存しない機能の本質を表す PIM の定義は重要性が増している．

本論文では，プロトタイプの生成元となる画面遷移やユーザビリティ機能を表す UML モデルは PIM として扱えるよう抽象化して定義することをねらう．

### 2.4 LIW : Live Interactive Widgets

本研究では，ユーザビリティ機能の要求確認支援を目的とする．特にユーザーが実際に操作しなければ分かり難いユーザビリティ機能として，画面の遷移 [*1] を必要とせず，ユーザーの操作に対して即時に反応する機能がある．この性質を持つ機能を実装した Web ユーザーインタフェース部品を Live Interactive Widgets (以下，LIW) と定義し，焦点を当てる．Google では，ユーザーが検索したい文字列を一部入力するだけで，その入力情報から検索候補が表示される．

### 2.5 画面遷移モデル

画面の入出力項目や画面の遷移を表したモデルのことを画面遷移モデル [6] と呼ぶ．本論文では，プロトタイプの生成元となる画面遷移モデルを先行研究 [6] の変更により実現する．画面遷移モデルは 3 章にて詳説する．

## 3. 提案手法

### 3.1 全体像

全体像の概要を図 1 に示す．本研究では，分析・設計時に画面遷移モデルを記述する開発プロセスを前提とする．以下に提案手法の利用手順を説明する．なお，本手法では LIW 付きプロトタイプを直接生成する方法と，既存のプロトタイプに LIW を付加する方法の 2 通りを実現する．

(1) 開発者が画面遷移モデルを記述する．画面遷移モデルの定義は 3.2 節にて説明する．

(2) 提案する LIW のモデルを利用し，開発者が画面遷移モデルの任意の画面や入力項目に LIW のモデルを対応付ける．LIW のモデルの定義は 3.3 節にて説明し，画面遷移モデルと LIW のモデルの連携方法を 3.4 節にて説明する．

(3) 本研究で実現するプロトタイプ生成ツールを使用し，開発者は LIW モデルと連携した画面遷移モデルから LIW 付きプロトタイプを自動生成する．なお，開発者が手動または既存のプロトタイプを利用したい場合，提案する画面遷移モデルとプロトタイプの HTML コードを対応付けるよう調整し，プロトタイプ生成ツールを使用して，プロトタイプに LIW を自動付加する．プロトタイプ自動生成や LIW の自動付加を行うツールは 4 節にて説明する．

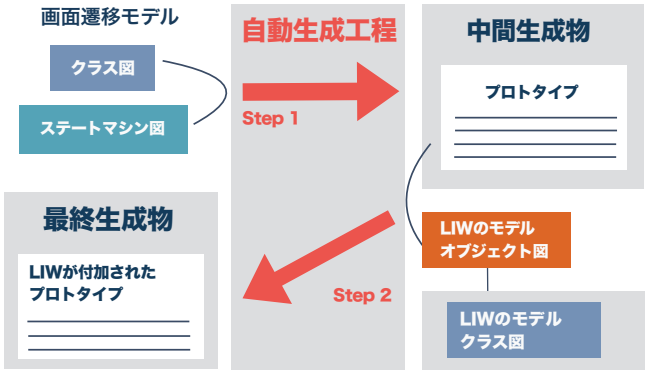


**図 1** 適用の流れ

[*1] 遷移の捉え方は種々あるが，ここでは URL の変化と捉える．

© 2014 Information Processing Society of Japan

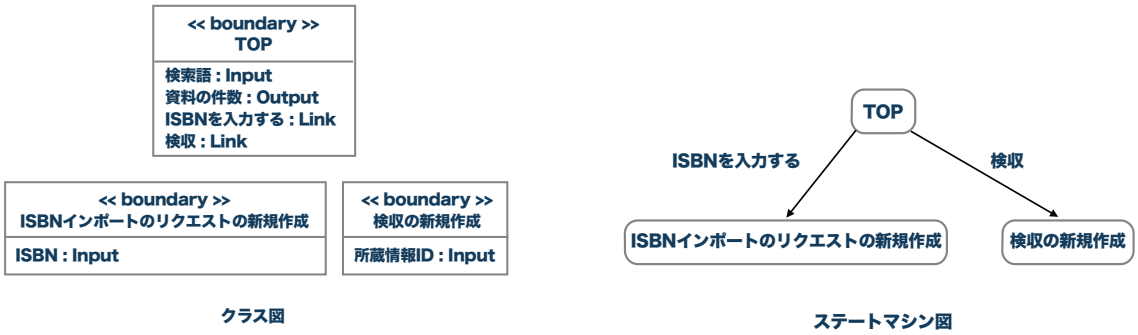


**図 2** 画面遷移モデル

### 3.2 画面遷移モデル

既存研究 [6][7] に見られるように，モデリングによる画面設計法が提案されている．本論文では以下の方針で既存の画面遷移モデル [6] を変更する．

□ モデルの煩雑さを解消できるよう，分割統治を図る．

□ プロトタイプの自動生成に適した簡潔な定義にする．

画面遷移モデルの説明のため，評価の適用事例でもある図書館システムの OSS である Next-L Enju[4] を事例に，モデルの例を図 2 に示す．既存研究のモデルでは画面の構造と遷移がクラス図に表現されるが，本論文では煩雑さの解消のため遷移をステートマシン図にて表現する．

#### 3.2.1 クラス図による画面構造定義

クラス図は UML において，クラスの構造を設計する図にあたる．画面遷移モデルにおいては，開発者は，画面の入出力項目や遷移といった画面の構造を記述する．従来の ICONIX プロセスで行われるロバストネス分析 [9] ではユーザーとシステムの境界を Boundary として表現する．その考え方に基づき，画面遷移モデル中の画面を表すクラスは Boundary として表現する．

画面遷移モデルにおけるクラス図の例が図 2 である．開発の初期段階では，データベースの構造を決定するために重要な情報である画面上の入出力項目の抽出を優先すべきである．また，操作手順などのユーザビリティに大きく影響する画面遷移もまた初期段階で明確化すべきである．以上から，Boundary の属性には入力項目，出力項目，リンクを記述する．これらはそれぞれ Input，Output，Link の型により区別される．各型の説明は以下のとおりである．

**Input:** ユーザーが入力する，画面遷移を生じさせない項目である．例えば，HTML における input タグや textarea タグにあたる．

**Output:** ユーザーが閲覧する文字列などの項目である．

**Link:** ユーザーが入力する，画面遷移を生じさせる項目である．例えば，HTML の a タグや遷移を生じさせる button タグにあたる．

#### 3.2.2 ステートマシン図による画面遷移定義

ステートマシン図は UML において，オブジェクトの状態遷移を設計する図である．画面遷移モデルのステートマシン図では，状態は Boundary と一対一で対応し，状態間の遷移は Boundary 間の遷移として捉え，トリガ名を Boundary の Link 型の属性と一対一で対応づける．なお，遷移間の遷移の矢印の方向は Boundary の遷移先を表す．

例えば，図 2 のクラス図において Boundary である “TOP” は，図 2 のステートマシン図の状態名 “TOP” と対応づく．また，図 2 では状態 “TOP” から状態 “検収の新規作成” への遷移がトリガ “検収” と共に記述されている．これは，“TOP” 画面から “検収の新規作成” 画面への遷移が “検収” と言うリンクによって行われることを表す．

### 3.3 LIW のモデルの提案

本節では LIW のモデルについて提案する．LIW のモデルは次の構成で説明する．まず，入力補完機能などの具体的な LIW のモデル（図 3）を説明する．

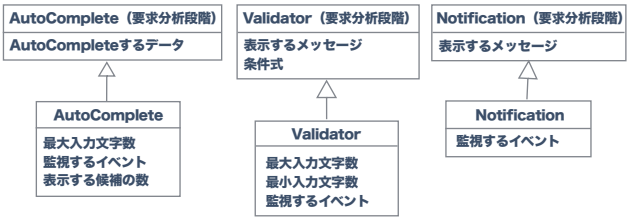


**図 3** LIW のモデル

前述の通り，後の開発の手戻りを防止するために，LIW を付加する必要があるか否かを見極める段階（要求分析段階）と，必要と判断された LIW の基本仕様から詳細仕様を決定する段階（設計段階）に属性を分割した．要求分析段階の属性は，筆者らがその LIW によってエンドユーザーが最も利益を得るために設定すべきものを提案しており，その他の属性は既存の JavaScript ライブラリなどで規定されるもの参考にしている．LIW のモデル化に際しては，AutoComplete，Validator および Notification の 3 種の LIW を取り上げる．実際には既存研究 [11][12] に挙げられるように Undo や Feedback など LIW の候補となる WebUI 部品は多種あるが，現状では Entity 構造やビジネ

© 2014 Information Processing Society of Japan

スロジックに影響がある可能性の高い 3 種のモデルを試作しており，モデルの拡充は今後の課題となる．

次に LIW に共通的に必要な機能実行のトリガとなるユーザーの入力イベントのモデルを定義する．最後に画面遷移モデルと LIW のモデルの連携方法を説明する．

LIW のモデルを説明する次節の表 1,2,3 では，要求分析段階にて決定すべき属性に “*” を付記する．同表では，開発初期段階の開発者のモデル記述量を軽減するために，未定義の属性値が存在した場合に与えられる既定値も掲載する．これらの既定値は，少量なモデル記述によるプロトタイプ自動生成に貢献する．

#### 3.3.1 AutoComplete

AutoComplete とはユーザーの入力した文字列に対して，既存の入力などを基にデータを自動で補完する機能である．この機能により，ユーザーは全てを入力する必要がなく，タイプミスが少なくなる．図 3 左に AutoComplete のモデルを示す．本モデルでは Bootstrap の機能であった Typeahead[13] のソースコードを参考に属性を定義した．各属性の意味を表 1 に説明する．

**表 1** AutoComplete の属性

| 属性 | 説明 {既定値} |
|---|---|
| AutoComplete するデータ* | コンマ区切りで補完の候補となるデータを列挙する．{ex1, ex2} |
| 最大入力文字数 | 入力上限の文字数を記述する．{30} |
| 表示する候補の数 | 一致した候補を表示する最大件数を記述する．{8} |
| 監視するイベント | 本 LIW が動作するために監視する，3.3.4 節のイベントを記述する．{onkeyup} |

#### 3.3.2 Validator

Validator とは，ユーザーの入力した文字列に対して，その入力が正しいか間違っているかを表す機能である．ユーザーは実際に入力しながらその結果を随時確認することができる．図 3 中央に Validator のモデルを示す．設定すべき属性は JavaScript フレームワークである AngularJS[14] を参考にして定義した．各属性の意味を表 2 に説明する．

**表 2** Validator の属性

| 属性 | 説明 {既定値} |
|---|---|
| 表示結果* | 入力が条件式と異なる場合に表示するメッセージを記述する．{It is wrong.} |
| 条件式* | 入力値を x として，入力が正しくなる条件の式を記述する．{x < 10} |
| 最大入力文字数 | 入力上限の文字数を記述する．{10} |
| 最小入力文字数 | 入力下限の文字数を記述する．{1} |
| 監視するイベント | 本 LIW が動作するために監視する，3.3.4 節のイベントを記述する．{onkeyup} |

#### 3.3.3 Notification

Notification はシステムまたはユーザーが発火させたイベントを基に，メッセージを送付する機能である．メッセージを受け取るユーザーは，自らが操作をシステムに与えずとも画面遷移なしでポップアップなどで通知を受け取れる．図 3 右に Notification のモデルを示す．また，図 4 が例である．それぞれの属性の意味を表 3 に説明する．

Notification では本来，エンティティなどのデータの状態変化に応じてポップアップメッセージが表示されることが考えられるが，プロトタイプとしては，画面で何のメッセージを閲覧できるかが重要であり，本論文では，そのための定義となっている．そのため，監視すべきデータの種類やメッセージ表示のきっかけとなるデータの状態変化を定義するなどビジネスロジックとの連携を考慮した属性定義は今後検討したい．

**表 3** Notification の属性

| 属性 | 説明 {既定値} |
|---|---|
| 表示するメッセージ* | 本 LIW が動作したときに表示するメッセージを記述する．{Notification fired.} |
| 監視するイベント | 本 LIW が動作するために監視する，3.3.4 節のイベントを書く．{onload} |


**図 4** Notification の例

#### 3.3.4 イベントのモデル

本節では，イベントを表すモデル定義の説明を行う．イベントには，例えば特定のキーが押下されたことやキーが離されたことなどの種類がある．LIW では，このイベントを表すモデル定義を使用して，これをトリガに発火する．イベントのモデルを図 5 に示す．LIW の作動には，異なる複数のイベントがトリガとなる場合がある．そのため，本論文では異なる複数のイベントにより構成される論理式を EventComponent と呼ぶことにする．

EventComponent は木構造を取っており，EventComponentOperand には W3C で定義されている Event[10]（例 : onclick, onmouseover）を持つ．EventOperator には演算子が定義される．演算子は結合順序を操作する上で，&&, | | の演算子を許す．イベントは例えば $click||keypress$ と表される．

© 2014 Information Processing Society of Japan

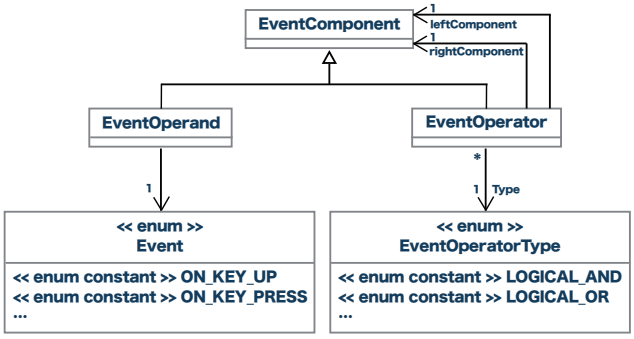


**図 5** EventComponent

### 3.4 画面遷移モデルと LIW のモデルとの連携方法

オブジェクト図は UML において，クラス図にて設計されたオブジェクトのある時点のインスタンスを表す図である．本論文では，画面遷移モデルに LIW のモデルを連携させるためにオブジェクト図を利用する．図 6 に例を示す．“TOP. 検索語： AutoComplete” のように LIW のモデルのインスタンス仕様を記述し，その名前に Boundary 名，または Boundary 名. 入力項目を記述する．

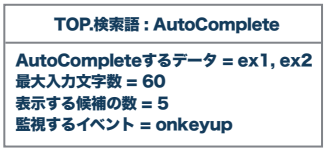


**図 6** オブジェクト図

そして，例えば “AutoComplete するデータ” スロットには，プロトタイプで示す入力補完候補の列挙する．各スロットの記述を省略した場合，LIW のモデルの既定値が設定されたとみなす．これらのスロットの値は，後述で提案するプロトタイプ生成ツールにおけるプロトタイプへの LIW の付加に利用されるが，開発者は既定値を利用することで，スロットを一切設定せずに LIW のモデルと画面または入力項目と対応付ける少量の記述でプロトタイプ生成ツールの恩恵を得られる．

## 4. プロトタイプ生成ツールの構成

プロトタイプ生成ツールは画面遷移モデルと LIW モデル，必要に応じてプロトタイプの HTML コードを入力にとり，LIW 付きのプロトタイプを出力する．本ツールでは，次の二つの機能がある．第 1 に画面遷移モデルと LIW のモデルからプロトタイプを生成する機能である．第 2 に画面遷移モデルと対応づく既存のプロトタイプに LIW のモデルを基に LIW を付加する機能である．

図 7 に全て自動生成されたプロトタイプを示す．本プロトタイプでは，LIW の要求確認を重視し，それ以外は簡潔な表現に留めている．もし，レイアウトや入出力項目を


**図 7** プロトタイプ

より正確に表現したプロトタイプを利用したい場合でも，LIW を自動付加機能により実現できる．画面遷移モデルと既存のプロトタイプの対応づけは HTML コードの input タグの placeholder 属性に，Boundary の属性名を記述するか，title タグに Boundary 名を記述して行う．

画面遷移モデルと LIW のモデルは Astah[8] を用いて記述され，Java で実装されたプロトタイプ生成ツールは Astah API によりモデルの情報を取得する．プロトタイプを直接生成する場合は，モデルの情報を HTML 等のコードに変換する．既存のプロトタイプに LIW を付加する場合は，取得したモデルの情報を JSON に変換し，JSON と既存の HTML を Node.js のプログラムに読み込ませ，HTML および JSON のパースを行い LIW を付加する．

## 5. 評価

### 5.1 概要

提案手法の有用性の一つは，既存のプロトタイプにも柔軟に LIW を付加できることによる適用範囲の広さにあると考えられる．

そこで，提案手法が正しく適用できるかを確認するために，OSS の図書館システム Next-L Enju[4]（以下，Enju）に手法を適用した．Enju には次の 2 つの特徴がある．第 1 に入力項目が多くユーザーが間違いを起こしやすいため，Validator や AutoComplete の恩恵を受けやすい．第 2 にフレームワークを使用しており，HTML はテンプレートエンジンを介して動的に生成される．以上から，適用事例として静的な HTML で構成されないアプリケーションを対象とすることで提案手法の適用可能性を評価した．

### 5.2 考察

Enju の一部に提案手法を適用した結果として，提案手法が HTML の入力タグの placeholder に画面遷移モデルを対応付ける性質上，対応付けの対象となる HTML タグがテンプレートエンジンに直接記載されない箇所があり，適用は困難であった．そこで，テンプレートエンジンにてタグが動的に埋め込まれた後の HTML を抽出した．これにより，LIW を付加できるように画面遷移モデルを作成でき，

© 2014 Information Processing Society of Japan

Enju の画面に LIW を付加できた．結果が図 8 である．


**図 8** Enju への適用結果

提案手法は Enju に適用できたことから，提案手法が正しく適用できることがわかった．しかし，現状の手法ではテンプレートエンジンに対しての適用は困難であったことから改善の余地がある．また，既存の HTML に対して LIW を付加すると言う観点から画面遷移モデルと LIW のモデルの定義が妥当であることや，画面遷移モデルと LIW のモデルの連携が正しく行えることがわかった．

提案手法による効率化に関して，実際のコードをほぼ変更せずに LIW の付加を行えたことから，手動による LIW が HTML，CSS および JavaScript の知識が必要なことを踏まえれば，開発者の負担を軽減し，かつ自動化による効率化が図れたと考えられる．

ただし，本論文は Enju の一部に対して適用した結果であるため，画面遷移モデルや LIW のモデルが Enju の全てに適用するにあたり，十分であったかを改めて評価する必要がある．また，提案手法の LIW は数や機能に限りがあり，包括的にユーザビリティを高める上で十分でなく，手法の限界といえる．

## 6. 関連研究

ユーザビリティ機能の要求獲得を支援する手法として，Juristo ら [11] のユーザビリティ機能の要求抽出を目的としたインタビューのガイドラインの提案や，Roder ら [12] のユーザビリティ機能のパターンをユースケース記述に関連付ける研究がある．しかし，本研究が着目するように開発の初期段階からエンドユーザーが理解容易なユーザビリティ機能の要求確認支援は行われていない．

Kamalrudin[15] らは要求文書からプロトタイプの自動生成する方法を提案しているが，本研究に見られるようにユーザーの入力に即時反応するユーザビリティ機能の要求確認はできない．また，Koch ら [7] は，Web アプリケーションをモデル化する手法を提案しており，本研究よりも Web アプリケーションの画面遷移の表現方法は充実している．そのため，本研究は今後 Koch らの画面遷移モデルに LIW のモデルを適用し，LIW のモデルの汎用性を評価することを検討する．一方，本研究のように Koch らの手法ではユーザビリティ機能の確認支援は対象としていない．

## 7. おわりに

本論文では，業務系 Web アプリケーションを対象として LIW を付加したプロトタイプの自動生成手法を提案した．図書館システムの OSS である Enju に適用した結果，提案手法により既存の HTML コードに LIW を埋め込むことができた．これにより，新規開発のみならず拡張開発でも要求確認支援として提案手法が有用である見込みを得た．

今後は，MDD による開発支援への提案手法の応用は 1 つの展望となるが，テンプレートエンジンなどの既存 Web アプリケーション開発技術に対応できる LIW のモデルへの拡張や，既存研究に見られる LIW 候補のモデル化によるモデルの充実化が課題となる．また，提案手法が要求確認支援としてどのように役立つかを実際の Web アプリケーション開発を通して評価する必要がある．

## 参考文献

[1] ISO/IEC 25010:2011, Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, 2011.
[2] ISO 9241-11:1998, Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11: Guidance on usability, ISO, 1998.
[3] Heiskari, J., et al.: Bridging the Gap between Usability and Requirements Engineering, the proc. of 17th IEEE RE, pp.303-308, 2009.
[4] Project Next-L：Next-L Enju, 入手先 ⟨http://www.next-l.jp/?page=Next-L+Enju⟩ (2014.5.10).
[5] OMG：MDA, 入手先 ⟨http://www.omg.org/mda/⟩(2014.05.10).
[6] 早川弘基, 小形真平, 海谷治彦, 海尻賢二:入力保存機能に着目したモデル駆動ユーザビリティ評価法, 第 20 回ソフトウェア工学の基礎ワークショップ FOSE 2013, pp.65-70.
[7] Koch, N., et al.: Requirements Models as First Class Entities in Model-Driven Web Engineering, the 3rd Workshop on the Web and Requirements Engineering at ICWE 2012, 2012.
[8] Change Vision: Astah, 入手先 ⟨http://astah.change-vision.com/ja/⟩ (2014.05.10)
[9] ROSENBERG, D.: Use case driven object modeling with UML. Massachusets: Addison-Wesley, 1999.
[10] W3C : Scripts in HTML documents, 入手先 ⟨http://www.w3.org/TR/html401/interact/scripts.html⟩ (2014.5.10).
[11] Juristo, N. et al.: Guidelines for eliciting usability functionalities. Software Engineering, IEEE Transactions on, 2007, 33.11: 744-758.
[12] RODER, H.: Specifying usability features with patterns and templates. In: Usability and Accessibility Focused Requirements Engineering (UsARE), 2012 First International Workshop on. IEEE, 2012. pp.6-11.
[13] Bootstrap:JavaScript, 入手先 ⟨http://getbootstrap.com/2.3.2/javascript.html⟩ (2014.5.10).
[14] AngularJS:Developer Guide, 入手先 ⟨https://docs.angularjs.org/guide/forms⟩ (2014.5.10).
[15] KAMALRUDIN, M. et al.: Generating essential user interface prototypes to validate requirements. In: Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering. IEEE Computer Society, 2011. p. 564-567.

© 2014 Information Processing Society of Japan