平成26年度
信州大学工学部情報工学科卒業論文

画面遷移モデルを基にしたシナリオ作成支援

海尻・小形研究室
学籍番号　11T5048H

中村　哲真

目 次

1 序論

1.1 背景

ソフトウェア開発の上流工程では，主に設計が行われる．この時，適切に設計されていることは重要である．なぜなら，設計はソフトウェア開発全体の基となり，設計に含まれる問題は実際のソフトウェアに含まれてしまうためである．また，設計に問題がある場合，早期にそれを発見することも重要である．問題を解決する際には，問題にかかわる実装済みの部分をすべて修正する必要があり，発見が遅れると修正箇所が多くなるためである．

設計段階に問題を検証する方法として，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の提案と評価が提案されている．この手法では UML (Unified Modeling Language) を拡張し，システムの画面遷移モデルと，システムの利用上想定されるシナリオを定義している．これにより，上流工程でユーザビリティに関する問題が発見できる．しかし，この手法のシナリオを手動で入力することは煩雑である．例えば，この手法ではシナリオを作成する時，多くの類似した値を入力しなければならない．また，正しいシナリオを作成するには手法を熟知している必要がある．この手法では UML を拡張しているため，拡張部分を理解しなければ正しいシナリオを作成できない．

このように手法の利用に対する障壁が高い場合，実際の開発では適用されない可能性が高くなる．従って，実用するためには利用に対する障壁を取り除く必要がある．

1.2 目的

本研究は，ソフトウェア開発の上流工程において，特にユーザビリティに関する問題の発見を支援することを目的とする．上流工程でユーザビリティの評価を行う手法は存在する[1]が，利用に対する障壁が大きく，実際の開発では適用されない可能性が高い．そこで，利用に対する障壁を取り除くため，開発者の入力が少なくなるよう，また，初学者でも正しいシナリオを書けるようにツールを設計，実装する．また，実際に手法を適用し，正しいシナリオを記述可能か検証する．

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の提案と評価[1](以下,「適用する手法」) で提案されているシナリオの作成支援を行う．

1.3 論文の構成

第二章では適用する手法に関する語句を説明する．第三章では関連研究について述べる．第四章では手法の説明を行い，第五章では評価について述べ

[1]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の提案と評価

る．第六章ではまとめと今後の課題について述べる．

2　用語説明

2.1　画面遷移モデル

画面遷移モデルとは，システム上の画面間の遷移を定義するモデルである．クラス図上に画面および遷移が記述される．画面遷移モデルの記法はメタモデルで定義されている (図 1).

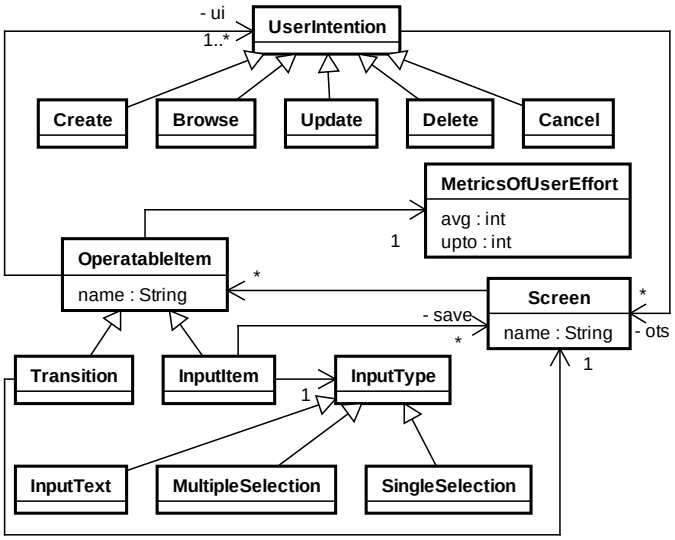

図 1: 画面遷移モデルのメタモデル

記法について説明する．画面 (Screen) はクラスとして表され，名前 (Screen.name) および操作項目 (OperatableItem) を持つ．操作項目は名前およびユーザー意図 (OperatableItem.ui) を持ち，遷移 (Transition) と入力項目 (InputItem) に分かれる．遷移は対象の画面を持つ．入力項目には入力型 (InputType) があり，入力された値の保存範囲を持つ．また，平均入力量 (metricsOfUserEffort.avg) と一画面入力限度数 (metricsOfUserEffort.upto) を持つ．入力項目は文字列入力 (InputText)，単一選択 (SingleSelection)，複数選択 (MultipleSelection) に分かれる．

ユーザー意図はユーザーが画面遷移する動機を表し，作成 (Create)，閲覧 (Browse)，更新 (Update)，削除 (Delete)，取消 (Cancel) の 5 種類に分けられる．また，ユーザー意図は操作対象画面 (UserIntention.ots) を持つ．これはユーザー意図の対象となる画面を表し，操作対象が遷移先でない場合に明示される．

2.2 画面インスタンス

画面インスタンスとは，画面 (Screen) のインスタンスで，各画面で入力された値が格納される．これには以下の 6 個の拡張記法が定義されている．

`<<delete>>` 画面インスタンスが削除されたことを示す.

`<<update>>` 画面インスタンスが更新されたことを示す.

`<<cancel>>` 入力された値が取り消されたことを示す.

`<<destination>>` 意図の中で最後に到達する画面インスタンスであることを示す.

\a 入力値および画面インスタンス名に対する拡張記法．入力値がシステムによる自動入力であることを示す.

\, 入力値に対する拡張記法．一つの項目に対して複数の値を入力する際に用いる.

2.3 シナリオ

適用する手法のシナリオは，想定されるユーザーの入力が記述される．また，シナリオに操作順序は記述されない．操作順序を記述した場合，画面遷移モデルが変更された際にシナリオを修正する必要があるためである.

シナリオは意図と呼ばれる単位で分割される．これはユーザーが行動する指向のことである．例えば，商品を購入する，カートに入れた商品を修正する等.

シナリオは以下の 5 種類のオブジェクト図で記述され，ユーザーの入力はオブジェクト図上に画面インスタンスで記述される.

開始 (Start) シナリオの起点となる画面インスタンスを一つ含む

事前条件 (Precondition) シナリオ開始前に存在する画面インスタンスを任意の数含む

意図 (Intention) 各意図で入力される画面インスタンスを含む

事後条件 (Postcondition) シナリオ完了後に存在している画面インスタンスを含む

終了 (End) シナリオの終点となる画面インスタンスを一つ含む

なお，シナリオに対して意図は複数存在し，他のオブジェクト図は一つ存在する.

3　関連研究

4　提案手法

本研究では，シナリオの作成支援ツール (以下,「本支援ツール」) を実装した．プラグインにより拡張可能であり，適用する手法が利用していることから，本支援ツールは astah*を利用した．また，astah* Plug-in は Java で記述する必要があるため，実装は Java で行い，GUI 部品の表示は swing，awt を利用した．

本支援ツールは画面遷移モデルを解釈して対象のシステムをシミュレートし，開発者に実際に操作させることで，シナリオの作成を支援する．すなわち，web ブラウザーのように画面遷移モデルをブラウジングし，値を入力することでシナリオを作成する．これにより，以下の効果を期待する．第一に，拡張記法をツールが補完し，手法の初学者が文法的な誤りを犯さない効果．第二に，ツールが以前入力した値を保持し，入力量を低減する効果である．

図 2 に本支援ツールの構造を示し，主要なクラスを解説する．

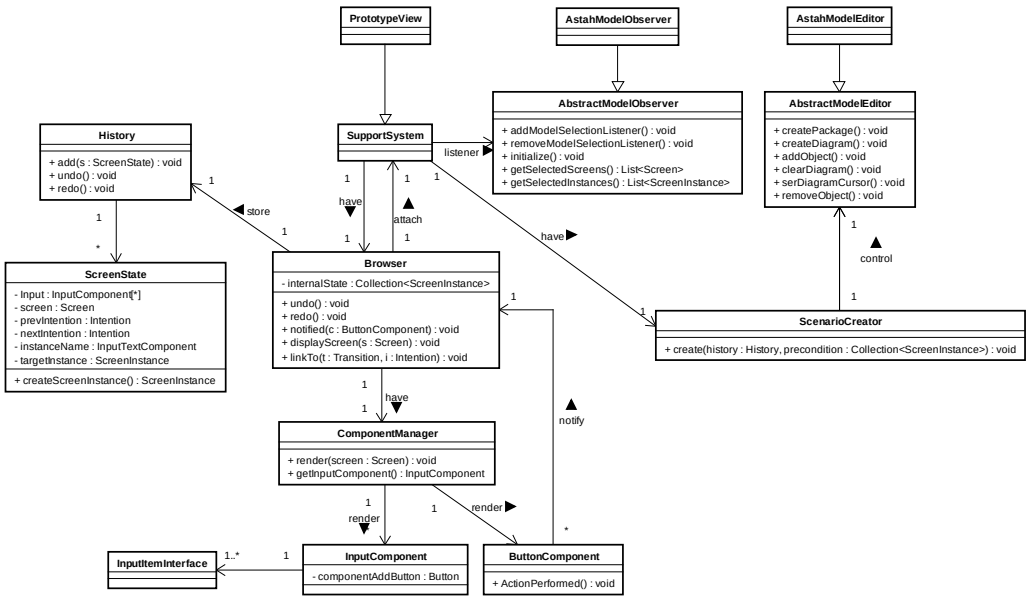

図 2: シナリオの作成支援ツール構造

SupportSystem はエントリーポイントである．このクラスはシナリオ作成全体の状態を管理する．さらに Browser，ScenarioCreator，ModelListener 等の主要クラスの初期化を行い，GUI を実際に描画する部分も担う．

Browser は核となる部分である．このクラスは Screen（画面）の表示と画面遷移を行う．Screen の表示は，GUI 部品の構成を担う ComponentManager

に Screen を渡し，それを表示する．画面遷移は，ButtonComponent から通知を受け，遷移を行う．また，History への値の格納も行う．

ScenarioCreator は，History に格納された値および，設定された Precondition（事前条件）を元にシナリオを生成する．

4.1 画面遷移モデルの解釈

対象のシステムをシミュレートするため，画面遷移モデルを解釈する．

Screen（画面）から GUI を構成する．Screen は OperatableItem を保持しており，これは GUI 部品に相当する．OperatableItem の実体は Transition（遷移）もしくは InputItem（入力項目）である．従って，Transition には操作可能な GUI 部品を，InputItem には入力可能な GUI 部品を割り当てる．本支援ツールでは Transition が ButtonComponent に，InputItem が InputComponent に対応している (図 3)．

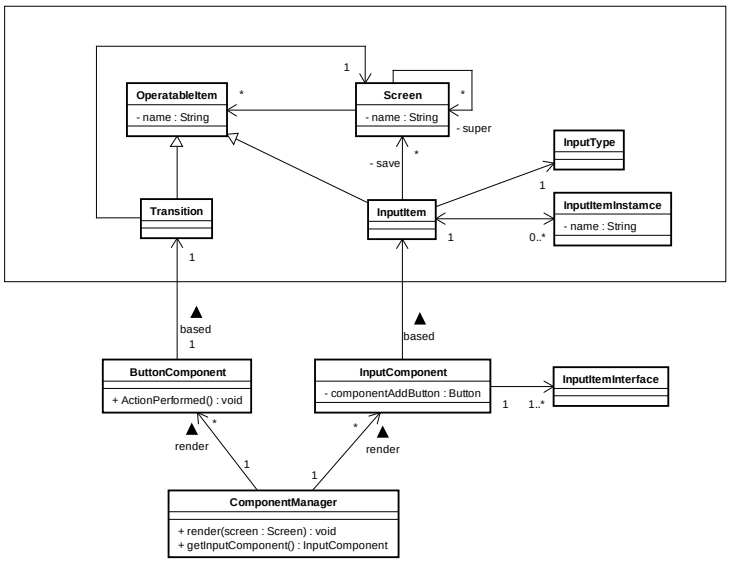

図 3: 適用する手法と本支援ツールの関係 (上：適用する手法)

4.1.1 ButtonComponent

ButtonComponent は操作可能なボタンとして扱われる．また，内部的には Transition の他に UserIntention を持つ．これはどのような意図を持って画面遷移を行ったかを示すもので，シナリオの構成に必要な情報である．

ButtonComponent は押されたとき，Browser へ押されたことを通知する．また，Browser は通知された時，どの意図で遷移を行うか選択肢を提示する．

この時提示される選択肢は ButtonComponent に含まれる UserIntention によって生成される．更新，削除など，対象のインスタンスを指定する遷移の場合，本支援ツールが保持している画面インスタンスを基に選択肢が追加される．また，特別な選択肢として意図のない遷移を追加する．これはシナリオには書き出されない遷移で，明示的に行う必要のない遷移が該当する．

4.1.2 InputComponent

InputComponent は複数の GUI 部品の集合として扱われる．内部に複数の入力項目 (InputItemInterface) を持つためである．これは InputType および InputItemInstance に相当し，入力方法の指定，値の格納を担う．これにより，特定の画面内の一種類の項目に対して複数の値の格納が可能になる (図 4)．また，入力項目を追加するボタン，指定入力項目を削除するボタンも InputComponent に格納される．

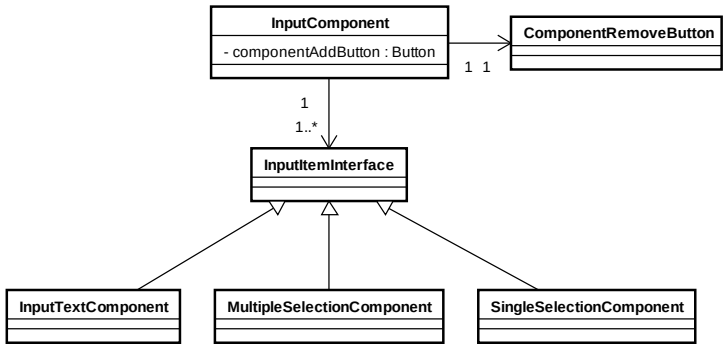

図 4: InputComponent の構造

InputItemInterface は InputType に従い，以下の三種類の実装を持つ．

InputTextComponent（文字列入力） InputText に対応し，文字列入力可能な部品として扱われる．

MultipleSelectionComponent（複数選択） MultipleSelection に対応し，選択および文字列入力可能な部品として扱われる．

SingleSelectionComponent（単一選択） SingleSelection に対応し，選択および文字列入力可能な部品として扱われる．

選択肢は画面遷移モデル上で定義されていないため，選択項目は文字列入力可能な部品になる．

4.1.3 GUI の構成

GUI の構成は ComponentManager が担う．Screen（画面）から ButtonComponent および InputComponent を生成し，レイアウトを行う．レイアウトは上段に入力項目，下段に操作項目を配置する (図 5).

図 5: 構成された GUI 部品

この時，同一の名前の操作項目が定義されていることがある．ボタンに表示すると判別がつかないため，同一の名前の操作項目が配置される場合にはボタンのラベルを「操作項目名 - 遷移先の画面名」に変更し，判別可能にする．

また，配置する際に InputComponent に元の InputItem.name をラベルとして付加する．

4.2 シナリオ作成に必要な要素

シナリオ作成に必要な要素を述べるにあたり，シナリオ作成の順序に従って述べる．シナリオ作成は，主に図 6 に示す四段階に分かれる．

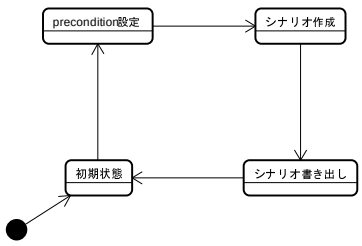

図 6: シナリオの作成順序

4.2.1 初期化

最初に，正常にシナリオを作成するため，初期化を行う．初期状態では，Browser，ScenarioCreator，ModelListener のすべてが初期状態になる．また，画面遷移モデルの読み込みは行わない．

4.2.2 事前条件設定

シナリオを作成する際，初めに事前条件 (Precondition) を設定する．事前条件の設定では任意の画面クラスから画面インスタンスを作成するか，既存の画面インスタンスを利用する．

従って，任意の画面クラスまたは画面インスタンスを選択する必要がある．そこで，ModelListener を使用する．ModelListener はエディタ上で図要素の選択状態が変化した際に呼ばれ，選択している図要素が取得可能になる．

この時，選択した画面クラスあるいは画面インスタンスを ComponentManager を用いて表示する．この時入力された値で画面インスタンスを作成し，Precondition に画面インスタンスを追加する．

また，画面クラス，画面インスタンスを取得する必要があるため，この時に画面遷移モデルを読み込む．

4.2.3 シナリオ作成

シナリオ作成では Browser を使用する．Browser は遷移を決定し，遷移の意図および入力された値を History に格納する．このとき，値は ScreenState として保存される．

ScreenState には入力されたコンポーネント，対象の画面，遷移の意図，画面インスタンス名，対象インスタンスが格納される．これらの要素はシナリオの生成に必要である．また，ScreenState を一つの単位とし，Undo，Redo 操作を実現した．値の入力は間違える可能性があり，その際に最初から作成し直すことは明白にコストが高いからである．

実装は二つのスタックを用いた（図**??**）．通常の入力値はスタック history へ格納していく．Undo 操作を行う際には history から一つ取り出し，一時的なスタック temporary へ格納する．Redo 操作を行う際には temporary から一つ取り出し，history へ格納する．

また，Browser は現在シミュレートしているシステム内の画面インスタンス郡，内部状態 (internalState) を持つ．これにより，作成された画面インスタンスが再利用可能になる．この時，Undo/Redo 操作による内部状態の変更が問題になる．Undo/Redo された操作により値が更新された場合，更新される前の値が保存されていないため，内部状態に不整合が起こる．

この問題に対応するため，二つの解決策を考案した．一つは，ScreenState に内部状態を持たせ，Undo/Redo する度に内部状態を更新する方法である この方法はメモリーの消費が大きいが，計算時間を抑えられる．もう一つは，Undo/Redo する度に内部状態を再度計算しなおす方法である．この方法はメモリーの消費は小さいが，計算時間がかかる．双方実装し使用したところ，顕著な差は見られなかった．従って，今回は後者の方法を採用した．

4.3 シナリオの導出

最後に，シナリオを導出する．これは本支援ツールが自動で行う．また，実際にオブジェクト図を操作するため，ModelEditor を使用する．導出には History と Precondition を用いる．以下に導出の手順を示す.

最初に，Precondition を生成する．これには与えられた Precondition をそのままオブジェクト図に書き出す.

次に，Start を生成する．一番初めの画面インスタンスは History の底に格納されているため，これをオブジェクト図に書き出す.

次にシナリオの核となる Intention を生成する．History を順番に参照し，オブジェクト図に書き出す．この時，画面インスタンス生成時の意図が無い場合，シナリオの構成上必要がないと判断し，書き出されない.

また，意図の変わり目は意図 Create による遷移と意図 Update による遷移の間にあると考えた [要検証]．そこで，意図 Create による遷移の後を状態 Create とし，意図 Update による遷移の後を状態 Update して，状態が変化した場所を意図の変わり目とした．また，初期状態は最初に行われた意図 Create もしくは意図 Update の遷移により決定する.

Intention の生成と並行し，Postcondition を生成する．Postcondition は最終的なシナリオの状態なので，Precondition と各 Intention を合成したものになる．そのため，Intention と同時に生成し，オブジェクト図に書き出す.

最後に End を生成する．これは History の先頭に格納されている画面インスタンスをオブジェクト図に書き出す.

5 評価

5.1 評価方法

[この章は評価計画が確定した後，記述する．]

5.2 考察・結果

[この章は評価実験の後，記述する．]

6　結論

6.1　まとめ

本論文では，ソフトウェア開発の上流工程においてユーザビリティに関する問題を発見するため，特定の手法[1]で提案されるシナリオの作成支援ツールの実装を紹介，説明した．これにより，手法を適用する障壁が取り除かれ，実際の開発において上流工程に手法[1]を取り入れることを可能にした [要検証・追記]．

本支援ツールにより，上流工程においてユーザビリティ評価が行われ，より品質の良い製品が開発されることを期待する．

6.2　今後の課題

今後の課題として，コンポーネントの表現の柔軟化が挙げられる．現在，コンポーネントは決まった種類の部品のみ定義されている．しかし，現実にはコンポーネントがコンポーネントを内包したり，他のコンポーネントに関係付けられたりと，様々なコンポーネントが存在する．これの改善案として，コンポーネントの表現を拡張可能にする案が挙げられる．これにより，現実に即したコンポーネントの構成が行える．また，複数選択の選択肢の指定も併せて定義されると良い．

また，他の手法に対して支援を行うことも今後の課題としたい．本支援ツールは特定の手法[1]にのみ適応できるが，画面遷移モデルが存在し，テストシナリオの生成を目的とする手法であれば支援を行える可能性がある．

[評価実験の後，追記する]

[1]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の提案と評価