M-043

Nagatukiにおけるマッシュアップ再利用の促進

Mashup component recycling method on Nagatuki

井口 圭一† 北野 貴稔† 小山 和也†
Keiichi Iguchi Takatoshi Kitano Kazuya Koyama

1. はじめに

近年注目を集めている SaaS(ASP)によるシステム提供形態では，カスタマイズ機能の重要性が高まっている．SaaS では，システムをサービスとして提供し，複数の利用者間で共有することによって，運用負荷の低減を実現している．しかし共通のシステムでは，利用者毎に異なる要求に合致することは難しい．その差を埋めるためのカスタマイズ性が，SaaS の成功の大きなカギとなる．

我々はサービスのカスタマイズ性を提供するための基盤として，マッシュアップ基盤 Nagatuki の研究を進めている．Nagatuki は各サービスに対するカスタマイズのためのマッシュアップ定義をそのリポジトリ上に蓄積し，その定義に従って，カスタマイズしたサービスをユーザに提供する．

Nagatuki のリポジトリには様々なマッシュアップの定義が蓄積されるため，その効率的な再利用性を確保することが重要となる．サービスに対する要求は利用者毎に少しずつ異なるため，マッシュアップもまた少しずつ異なるものが多数登録されることになる．これらのマッシュアップは同じサービスに対するカスタマイズを提供したり，似た機能を他のサービスに付加したり，互いに関連を持って存在するため，あるマッシュアップに適用された変更，機能追加は，他のマッシュアップにとっても有用である可能性が高い．そのような変更，機能追加を互いに再利用可能にすることで，より高度な改善効果を得ることができる．

2. マッシュアップにおける再利用

マッシュアップの再利用を促進するための部品化のアプローチには二通りが考えられる．一つはマッシュアップそのものを再利用しやすい部品に分解すること．もう一つは，マッシュアップに対する機能の追加や変更などの編集操作を部品化して再利用可能とすることである．本論文では，後者の部品化の手法について述べる．

Nagatuki のリポジトリに蓄積されるマッシュアップは，そのままでは，単なるカスタマイズを実現するための記述であるが，そこに対する編集内容を部品化することで，機能，修正のカタログとして利用することが可能となる．例えば，マッシュアップに対して追加された機能を部品化することにより，他のマッシュアップに同様の機能を追加する際に再利用することが可能となる．また，元サービスが変更になった場合，そのサービスをカスタマイズするすべてのマッシュアップに修正が必要になるが，修正部分が部品化されれば，同じ元サービスを利用しているマッシュアップは部品を適用することで，変更への対応を行うことができる．

しかし，マッシュアップを編集する際に，あらかじめ編集内容を部品化して再利用することを考えて設計することは一般的には難しい．まず，部品化する単位の決定が必要である．小さな部品は汎用性は高いが，機能として不完全であったり，他の要素に対する依存性が残ったりする．逆に大きな単位は，依存関係を部品内に含めることは容易になるが，提供する機能もより具体的な機能となるため，汎用性に乏しくなる．

Nagatuki では，なるべく広いユーザにカスタマイズのためのマッシュアップを作成させたいとの考えから，マッシュアップ編集時には再利用性など考えずとも，基盤が自動的に編集内容を部品化することを実現する．

3. 自動部品化による再利用促進

本章では，マッシュアップに対する編集内容を自動的に部品化し，再利用可能とするためのアーキテクチャを提案する．

提案手法では，まずマッシュアップを構成する要素を定義し，それらに対して変更の種類，依存関係を定義する．これらの要素を組み合わせて作成されたマッシュアップが編集された際は，定義に従って解析することによって，編集内容のうちどの部分が相互に依存した編集内容であるかを判別することが可能となり，編集の一貫性を維持するために必要な部品を作成することができる．

以下に Nagatuki を例に本手法に必要となる要素種類の定義，各要素種類の依存関係の定義，及びそれらを利用した部品化の手順について詳細に述べる．

3.1 要素種類定義

提案手法では，まず対象とする系においてマッシュアップを構成する要素の種類，及びその要素種類に対して，許容される外部に影響のある変更箇所として変更種類を定義する．

以下は Nagatuki を例に定義した要素の例を示す．Nagatuki ではアプリのビュー，データモデル，振舞いに対するカスタマイズを実現するために，以下の要素を定義した．(表 1 参照)

要素種類	変更種類
データ	データ型，値
データ加工	入力データ，加工内容，出力データ型
自動実行	入力データ，出力データ，使用画面変換，実行コマンド
画面変換	対象サービス，編集内容
テンプレート	入力データ，出力 HTML
サービス	画面構造，ページ遷移

表 1

† （株）日本電気株式会社, NEC Corporation.

- データ：テーブル形式で，各列名とその値を持つ．変更種類として，どのような列から構成されるかというデータ型と，実際に保持される値を持つ．
- データ加工：データ要素を入力し，様々な加工処理を行い，別のデータとして出力する要素．他の要素からはデータ加工要素もデータ要素として見える．変更の種類として，加工の対象となる入力データ，入力データに対してどのような加工を行うかの内容である加工方法，出力するデータ型を持つ．
- 自動実行：データ入出力や画面操作などを組み合わせて，元サービスの自動処理を行うための要素．様々な情報を使用するため入力データ，出力データの変更種類を持つ．また，実行するコマンドに関する実行コマンドの変更種類も持つ．
- 画面変換：元サービスの Html に，テンプレート要素で記述された Html 部品を挿入したり，加工したりする要素．
- テンプレート：データ要素を入力とし，画面変換で使用する Html 部品を出力する．
- サービス：サービスの要素種類は，カスタマイズの対象となるサービスの変更を本部品化の対象とするための仮想的な要素であり，実際にリポジトリに保存されるのは，対象サービスへのポインタ及び，変更種類の対象となる画面構造，ページ遷移情報となる．

3.2 依存関係定義

次に，要素種類毎に，関連する要素のどのような変更種類に対して，該要素も変更が必要になるかを示す依存関係を定義する．表 2 に Nagatuki で使用する要素種類に関する依存関係を示す．定義では，各要素種類が依存する対象要素とその要素における依存する変更種類を記述する．独立した要素であり他の要素の変更に依存しないデータ要素と，外部のサービスの変更を示すサービス要素を除く 4 要素について示す．
- データ加工：入力として使用するデータ要素のデータ型に関する変更に対して依存する．
- 自動実行：入出力を伴う自動実行の場合，使用するデータ要素のデータ型に依存する．さらに，自動実行しようとする元サービスの画面構造及び，ページ遷移についても依存する．
- 画面変換：元サービスの画面を認識し，変更を加えるため当然に元サービスの画面構造に依存する．また，テンプレートを使用して画面要素を追加する場合，そのテンプレートに関して存在することのみに依存する．
- テンプレート：データ要素を Html 化するため，入力となる要素のデータ型に依存する．

要素種類	依存対象	依存内容
データ変換	入力データ	データ型
自動実行	使用データ	データ型
	元サービス	構造
		ページ遷移
画面変換	使用テンプレート	存在
	元サービス	画面構造
テンプレート	入力データ	データ型

表 2

3.3 部品化

マッシュアップに編集が加えられた際は，前 2 節で定義した変更種類，依存関係を利用して以下の手順で．編集内容を部品化する．

1. マッシュアップをグラフ化する
   マッシュアップを構成する要素を関連する要素間を結んだグラフで記述する．
2. 変更要素にその変更の種類を示すラベルを張る
   変更のあった要素をそれぞれ要素種類定義に従って解析し，どの種類の変更があったかを解析しラベルを張る．
3. 依存する編集のないリンクを削除する
   依存関係定義に従って，要素間の関連のうち，リンクの両端に変更があり，いずれかの要素の依存内容に記述された変更が他方の要素にあるリンク(図 1 内実線)を残し，それ以外のリンク(破線)を削除する．
4. リンクのつながった要素をグループ化
   3 の操作で残ったリンクでつながった要素に対する編集は，依存関係があるので同時に適用する必要がある編集とみなされる．これらをグループ化し，部品化する．

以上の処理で，一貫性を維持したまま適用可能な最小の部品を自動構成することができる．

本手法では特に，要素種類毎の依存関係を，対象の変更内容にまで踏み込んで定義するため，図 1 における，元サービスと画面変換の例のように，サービスの画面構造には依存するがページ遷移には依存しないような場合に部品を分割することが可能となる．

4. まとめ

本稿では，マッシュアップの編集を一貫性を維持した最小の部品として再構成する手法を提案した．提案手法では，あらかじめ部品種類毎の変更種類，依存関係を定義し，その定義に従って編集を解析し部品化する．

この手法により，最小限の部品を自動的に作成できるため，マッシュアップ編集者は部品化することを意識せずとも，編集内容の再利用を可能とし，ひいては，マッシュアップの有効活用を図ることができる．

今後は，この技術で作成された部品群を効率よく利用するための適用可能な部品の検索技術，部品を適用した際の動作の検証について研究を進めていきたい．

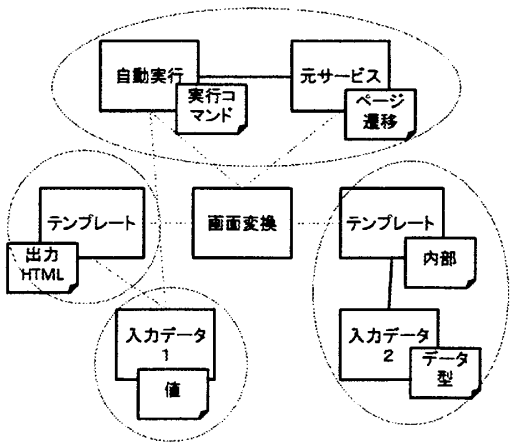

図 1 部品化処理