M-042

Nagatuki: サービスカスタマイズ向けマッシュアップ基盤

Nagatuki: A Mashup Platform for Service Customization

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

1. はじめに

近年、SaaS や SOA など、システムを新規構築するのではなく既存の物を利用、特にサービスとして共用する事で構築と運用コストを低減するアプローチが注目を浴びている。しかし既存サービスの利用では、利用者の要求と既存サービスの提供機能のギャップが問題となる。特に業務利用で、例えば必要なデータ項目が扱えないとか、業務手順と操作手順があわないといった必須要件が満たされない場合、既存サービスを利用する事はできなくなる。また SaaS や SOA に限らず、完全な要求獲得の困難さや要求の時間的変化、利用者間の相矛盾した要求など、様々な理由で提供サービスと利用者要求との間のギャップを完全に解消する事は難しい。何らかのカスタマイズ機能を持つサービスも存在するが、既存のカスタマイズ機能で全ての利用者要求をカバーできるとは限らない。

そこで本稿では、既存サービスに外付けすることで、利用者の要求に適合するようサービスをカスタマイズ可能にするマッシュアップ基盤 Nagatuki について述べる。

2. SOA によるサービスカスタマイズ

SOA は、サービスが I/F を公開してデータや機能を外部利用可能にする事で、利用者がそれらを自由に組合せる事を可能にするものであり、それ自身がサービスのカスタマイズ性向上に対する有効なアプローチである。ここで SOA は、公開 I/F として XML Web サービスなど通信 I/F を公開して主に他のシステムがサービスの機能やデータを利用するものと、公開 I/F として Portlet や Widget などの GUI 部品を提供し、利用者が複数部品を自由に組合せて好みの画面を構成するものの両方を想定している。

しかし従来の SOA では、既存サービスの利用形態が公開 I/F の組合せに限定されるため、呼び出す側でのカスタマイズの余地は、例えば WS-BEPL[1]やポータル等による組み合わせ方の変更に限定されてしまう。例えばサービスがデータや個別機能を細かい粒度で外部利用可能にしていた場合、既存サービスが元々持っていたデータや機能の組み合わせ方の設計・実装を利用することはできず、カスタマイズとして構築しなければならない部分が増えてしまう。組み合わせ方も含めてより多くを利用するためにより粒度の粗い I/F を提供すれば、それより細かい粒度でのカスタマイズ性は失われる。よってサービス設計者による適切な粒度の I/F 設計が重要になるが、I/F の適切さはその利用形態に依存するため、事前に利用形態を予測し適切な I/F を設計する事は容易ではない。

よってサービス設計者が事前に想定できない様々なカスタマイズ要求に低コストで対応可能にするには、サービス全体といった可能な限り大きい粒度で既存サービスを流用しながら、公開 I/F など既存サービスの作りに依存しない形で、サービスの提供する機能や振舞いを様々な形で部分的に変更可能にすることが重要となる。

3. マッシュアップ基盤 Nagatuki

3.1 概要

マッシュアップ基盤 Nagatuki は、サービス全体を共有しつつ様々な部分的変更を可能にすることで、利用者の要求に最適化されたカスタムサービスの構築を可能にするものである。この実現のために Nagatuki は、Web アプリを対象として、Web アプリと利用者の間にプロキシとして配置し、Web アプリの出力する GUI を解析・制御する事で、既存 Web アプリは変更せずに単一アプリの拡張や複数アプリの統合など様々なカスタマイズを実現する。

一般にカスタマイズとして求められる内容は幅広いが、Nagatuki では、既存アプリのビュー、データモデル、振舞いの3の視点でのカスタマイズを考慮し、これらのカスタマイズ実現のために、画面変換、自動実行、データ管理の3つの基本機能を提供する(図 1)。

（1）画面変換: Web アプリの出力する HTML を書換え、GUI 部品の追加、削除、移動などを行う。また HTML 中のデータ抽出や、データ管理上のデータ操作など行い結果を書換え内容に反映させてもよい。画面変換は単純なビューの変更に加え、データ管理とあわせたデータモデル変更や、ボタンを押したときのリンク先変更などによる振舞いの変更、複数アプリの画面合成などにも利用する。

（2）自動実行: ユーザによるブラウザ上での GUI 操作をエミュレートし、Web アプリを自動実行する。主に振舞いの変更として、既存 Web アプリの画面遷移をスキップしたり、一部入力を自動的に行うなどに利用する。また複数 Web アプリ間でのデータの受け渡し等にも使用する。

（3）データ管理: マッシュアップ基盤自身が独自のデータを保持する。データモデル変更で新たに追加された項目や、自動実行で自動入力する値などの保持に利用する。また異アプリ間でデータを共有するのにも用いる。

マッシュアップによるカスタムサービス構築は、これら3機能用の動作定義を与えることで行う。

3.3 Nagatuki システムアーキテクチャ

Nagatuki の実行基盤はプロキシ、ブラウザ制御スクリプト、Wiki の3つの主要モジュールから構成される(図 2)。

プロキシは HTTP プロキシとしてブラウザと Web アプリ間の通信を中継する。既存 Web アプリは必ずこのプロキシを介して利用し、通常は既存 Web アプリの全ての機能がそのまま利用できるが、特定の通信に対しては画面変換の定義が適用される。またプロキシではブラウザ制御スクリプトの挿入も行う。

ブラウザ制御スクリプトは JavaScript で実装された自動実行定義のインタプリタである。自動実行が起動されると、Web アプリの画面に加えてインタプリタがブラウザに読み

† （株）日本電気株式会社, NEC Corporation

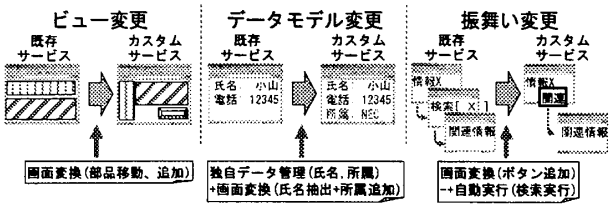

図 1 カスタマイズ視点と実現方法

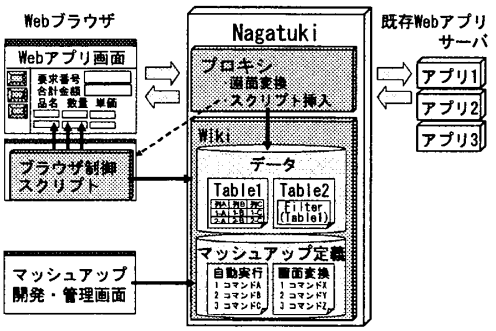

図 2 Nagatuki アーキテクチャ

込まれ、定義に従いブラウザに表示されている Web アプリに入力やクリックなどの操作イベントを送る。

Wiki はデータ管理と各種カスタマイズ定義の管理を行う。データ管理は単純なテーブルの形式でのデータ保持と、テーブルに対するフィルタやソートなどの加工機能を持つ。全てのデータやカスタマイズ定義は Wiki ページとして表現されるため、Web 画面上で定義を開発・修正したり、複数利用者で定義を共有することができる。

3.4 マッシュアップ定義記述

画面変換、自動実行、データ管理の各定義は下記のように記述する。

（1）自動実行:画面操作イベントに相当するコマンドや、for 文などの実行制御、データ管理上のデータ操作などのコマンドを順に列挙する形で記述する。

（2）画面変換: HTML 部品の追加、削除、移動などのコマンドや、HTML からのデータ抽出やデータ管理上のデータ操作などのコマンドを列挙する形で記述する。部品追加では、データ管理のデータを入力とする HTML テンプレートも利用できる。

（3）データ管理:テーブル形式で、列名と、その値としての複数の行という形で定義する。保持データは Wiki のテーブル記法で記述する。加工機能は、新たなテーブルの定義として、他のページを参照する加工コマンドによって記述する。

画面変換や自動実行において HTML 中の特定要素を指定するには XPath を用いる。ただしこの XPath は Web アプリの HTML の作りに強く依存するため、作成には HTML の詳細解析が必要であり、また Web アプリの改版などで画面が変わると XPath も再定義しなくてはならない。そこで Nagatuki では、HTML の要素指定の記述を UI 部品と呼ぶ形で記述を分離し、画面変換や自動実行の記述は UI 部品を参照することで、異カスタマイズ間で UI 部品を共有すると同時に、Web アプリ改版時のマッシュアップ定義の保守を省力化する。

3.5 マッシュアップ例

Nagatuki 機能実証のために、企業内 PC 購入要求のカスタムサービスを構築した。図 3(a)は既存サービスの画面遷

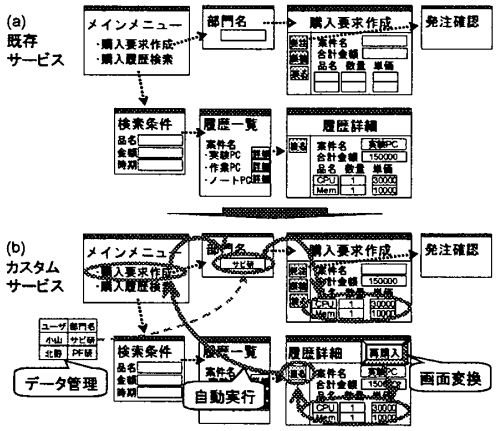

図 3 マッシュアップ例

移で、PC の詳細構成を入力しての購入要求書作成や、過去の購入履歴の参照機能を持ち、それぞれメインメニューから幾つかの画面遷移を経て利用できる。

ここで、類似の構成の PC を頻繁に購入する利用者を想定して、過去の購入履歴の構成をコピーして新たな購入要求を作成する機能を追加するようカスタマイズを行う。(図 3(b))。この場合マッシュアップ定義は、購入履歴の参照画面に「再購入」のボタンを追加する画面変換と、購入履歴画面から構成情報を抽出し、メインメニューから部門名入力を経て購入要求作成画面に遷移して抽出した構成情報を入力欄に入力する自動実行と、ユーザ毎の部門名を保持するデータ管理定義からなる。「再購入」ボタンを押すと自動実行が起動し、同じ構成を入力した購入要求画面を表示して自動実行が停止する。自動実行の前の購入履歴を表示するまでの履歴検索の機能や、自動実行後に購入要求書を発注したり保留したりする機能など、変更した箇所以外は元のサービスの機能をそのまま利用可能である。

5. 考察

関連する従来技術では、プロキシ等での HTML 変換は多数存在するが、ビューしか変更できない。また Web の自動実行も多数存在するが、別システムとの連携用に完全自動化するか、利用者が自分の作業省力化に導入するためのもので、振舞い変更のために自動実行をサービスの一部として提供することは困難である。画面変換、自動実行、データ管理の 3 機能を統合により、多様な変更を施したカスタムサービスを構築可能にしていることが、本研究の特徴であると言える

6. まとめ

本稿では既存の Web のサービスを利用者の要求に合わせてカスタマイズ可能にするマッシュアップ基盤 Nagatuki について述べた。今後は様々なサービスへの適用によるカスタマイズ能力の見極め・強化や、実行オーバヘッドの低減やスケーラビリティの向上を進める予定である。

参考文献

[1] OASIS Open, "Web Services Business Process Execution Language Version 2.0", 2007