2009－SE－163 (12)
2009／3／18

メタデザインに基づいた Webアプリケーション開発プラットフォームの提案

中田 宏昭　　深海 悟
大阪工業大学大学院 情報科学研究科
〒573-0196 大阪府枚方市北山 1-79-1
e-mail : fukami@is.oit.ac.jp

概要

近年、Wikipediaやニコニコ動画といったWebアプリケーションサービス(WebAS)を中心にマス・コラボレーションによるコンテンツ生成が盛んである。マス・コラボレーションは不特定多数のユーザがコンテンツを中心に協業する生産形態で、多様性・品質・生産性の面で注目されている。しかし、WebAS自身は不特定多数のユーザによって拡張することが出来ないためマス・コラボレーションが行われていない。本稿では、この問題を解決する手段として、WebASでメタデザイン構造を実現する手法としてコンテンツ共有型クローンモデルを提案し、それに対応するシステムを実装した。

A metadesign approch to WebApplication Development Plattform

Hiroaki Nakada　　Satoru Fukami
Graduate School of Computer Science, Osaka Institute of Technology
1-79-1. Kitayama, Hirakata-City, Osaka, 573-0196, Japan

Abstract The contents generation with the mass-collaboration is popular on web application service such as NicoNicoVideo, Wikipedia in recent years. Mass collaboration is a form of collective action that occurs when large numbers of people work independently on a single project. Mass collaboration is paid to attention for diversity, quality, and productivity. But, many and unspecified users cannot extend web application service. Therefore, mass collaboration is not done. To solve this problem, we propose the Contents Share Clone Model as a method for the achievement of the meta-design approach to web application service.

1 はじめに

近年、マス・コラボレーションによるコンテンツ作成が盛んである。マス・コラボレーションでは不特定多数のユーザが参加するという性質上、コンテンツの多様性、品質、作成速度の向上が期待される。マス・コラボレーションはWikipediaやニコニコ動画等といったWeb Application Service(WebAS)を中心に行われるが、マス・コラボレーションの場であるWebASそのものはマス・コラボレーションの対象となっていない。なぜなら、マス・コラボレーションが行われるためには対象コンテンツが不特定多数によって共有されていなければならない[1]。しかし、現状WebASは利用者による変更も複製も出来ず共有されているとは言えないためである。

そこで、WebASにおけるマス・コラボレーションを実現するために、メタデザインアプローチに基づいたWebASを開発することを提案する。メ

タデザインとはユーザ自身によってシステムを再設計できるような設計であり、オープンソースソフトウェアをはじめ、マス・コラボレーションが行われているものの中にはこの形態をとるものも多い。本稿では、WebAS にメタデザインを適用する方法について述べた後、メタデザインに基づいた WebAS を構築するために開発した Web アプリケーション開発プラットフォーム「SoS」について述べる。

2 メタデザインによる WebAS の拡張と問題点

メタデザインではシステムを設計段階で完成させずに、拡張する余地を残しておく。これによって利用時点で新しい要求があれば利用者自身の手で継続的に改善をしていく[2]。メタデザインを実現する方法として最も単純なのは、Wikipedia の様に直接対象コンテンツを書き換える方法である。これを直接編集モデルと呼ぶ。また、直接編集を行わずに、対象コンテンツを複製し、複製したコンテンツを書き換える方法がある。これをクローンモデルと呼ぶ。

2.1 直接編集モデル

直接編集モデルではコンテンツを直接書き換える。WebAS に直接編集モデルを適用した場合、 書き換えるコンテンツはシステムのソースコードやユーザインタフェース(UI)となる。既存の Wiki システムなどをベースに開発すればよいためシステムは単純になるが、編集合戦とセキュリティの問題がある。

編集合戦とは、二人またはそれ以上の人数で他の人の編集の一部または全ての差し戻しを繰り返すことである。Wikipedia 等の辞書と違い、WebAS の変更は客観的に良し悪しが判断できない物が多い。例えば「背景色は白と黒どちらが良いか？」といった議論は回答が無く、個人の嗜好に依存してしまう。そのため、編集合戦に陥りやすい。

また、セキュリティの問題もある。稼動しているサービスを直接変更できるようにすると、悪意を持った変更がされてしまう可能性がある。そのため、昨日まで安心して利用できたサイトが次の日には危険なサイトになっているといった、クロスサイトスクリプティング(XSS)と同様の問題が発生し、サービスの信頼性を保つことは出来ない。

このように直接編集モデルでは他者に与える影響が強いため、WebAS の拡張方式としては採用しづらい。そのため、より他者への影響が小さいモデルを利用する必要がある。

2.2 クローンモデル

前述したとおり、直接編集方式では他者に強い影響を与えてしまうため、気軽に編集できない。そこで、コンテンツを複製し、各々で独自に変更を加えるクローンモデルを利用する。クローンモデルでは、複製した個々のコンテンツは独立しているため、他者に影響を及ぼさず、上記の問題が発生しない。

2.3 コンテンツ共有型クローンモデル

WebAS でクローンモデルを実現するには WebAS の複製を行う必要がある。WebAS は図 1 の様に「システム」「コンテンツ」「ユーザ」の三つの要素に分けて考えることが出来る。

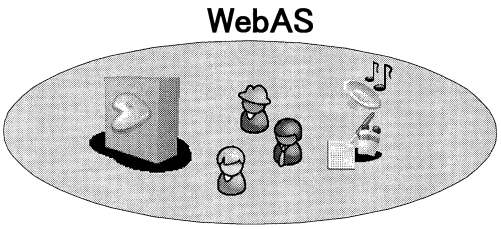

図 1. WebAS の構成要素

システムやコンテンツと異なりユーザの複製は出来ない。また、システムとコンテンツを複製しただけでは、SNS をはじめとして多くの WebAS では意味が無い。すなわち、WebAS の複製は単純には行うことが出来ない。そこで、WebAS において一つのサービスで複数のシステムが存在するという点に注目した。

例えば、図 2 の様に携帯電話と PC の両方から利用できる WebAS は多い。しかし、転送量や画面サイズ、操作性などの関係から携帯電話向けと PC 向けでは UI が異なっている。すなわち、厳密には異なるシステムである。

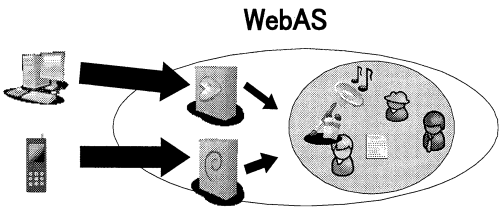

図 2. 複数のシステムを持つ WebAS

このことから、コンテンツとユーザが同じであれば、システム部分が異なっていても同じ WebAS に見えるのではないかと考えた。

この点を踏まえ、本研究では WebAS の拡張方法として、コンテンツ共有型クローンモデルというモデルを提案する。コンテンツ共有型クローンモデルは図 3 の様に、コンテンツとユーザを共有したまま、システムのみを複製し拡張するモデルである。

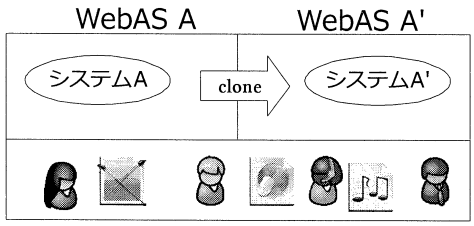

図 3. コンテンツ共有型クローンモデル

WebAS A と、そのクローンである WebAS A'はコンテンツとユーザを共有しているため、どちらかで追加されたコンテンツはもう一方へも反映される。また、WebAS A のシステム A と WebAS A'のシステム A'は異なっているため、システム A やシステム A'に変更を加えても、もう一方へ影響を与えることは無い。これによって他者へ影響を与えずに自分の利用している WebAS を拡張することが可能である。

3 WebAS 開発プラットフォーム「SoS」

本研究では、コンテンツ共有型クローンモデルを用いた WebAS を開発する環境として「SoS」の開発を行った。SoS は WebAS を開発するための WebAS であり、ブラウザのみで WebAS の開発および拡張が可能である。SoS ではコンテンツ共有型クローンモデルのサポート、KGR コンポーネントモデル、UI とモデルの相互変換によりサポートを行う。

本章では、開発したシステムについて述べる。

3.1 Service

本システムでは、システム上で開発されアプリケーションを Service と呼ぶ。Service は複数の Page から構成されており、通常の WebAS と同様に利用できる。Service はコンテンツ共有型クローンモデルのシステムにあたる。Service 間でのユーザとコンテンツは共有されているため、Service を複製するだけで、コンテンツ共有型クローンモデルを実現できる。また、複製された Service 間では親子関係をしめす参照があるだけで、お互いの変更は影響を与えない。そのため、安全に利用している Service の拡張が可能である。

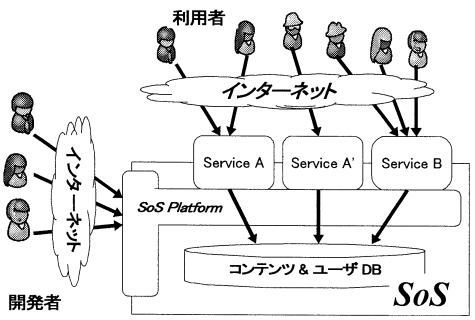

図 4. システム概要

Service とユーザおよびシステムの関係は図 4 の様になる。開発者はインターネットを介して SoS にアクセスし、SoS 上のリソースを使って Service の開発を行う。利用者は同じくインター

ネットを介して各 Service を利用する。図中ではユーザを開発者と利用者の 2 つに分けているが、あくまで役割の話であり実際には開発者と利用者を兼任する場合も多いと想定される。

Service の拡張は、「Service の所有者が変更を行うケース」と「Sevice の所有者以外が変更を行うケース」の2つがある。以下では、2 種類のケースについて述べる。

① Service の所有者の場合

変更対象の Service A を直接編集する。

② Service の所有者ではない場合

変更対象の Service A の複製である Service A'を暗黙的に作成し A'を編集する。

3.2 KGR コンポーネントモデル

本システムでは、非プログラマでも WebAS の開発を行えるようにするために、コンポーネントの組み合わせで Service を構築する。UI の定義は KGR という XML ベースの UI 定義言語を用いる。

Service の振る舞いを UI のイベントにコマンドを関連付けることで定義する。コマンドは goto(ページ遷移)、save(データの保存)等の様に、WebAS で良く行われる処理をまとめたもので、コマンドを用いることによりプログラミングの知識が無くても比較的容易に Service の振る舞いを記述できる。

また、UI の定義およびイベントとコマンドの関連付けは、システム内部では XML データとして管理しているが、より簡単に開発できるように GUI エディタを実装ししてる。GUI エディタは図 5 の見た目を直接弄れる WYSIWYG な UI デザイナ、図 6 のコマンドを選択式で記述できるイベントマネージャ等の機能を提供している。

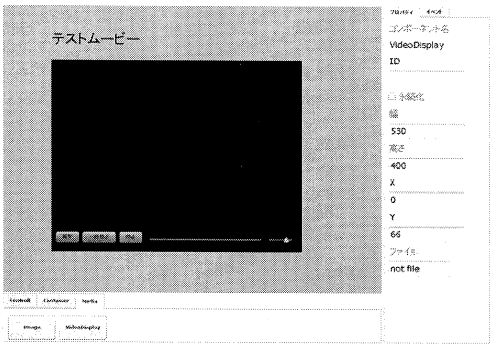

図 5. WYSIWYG な UI デザイナ

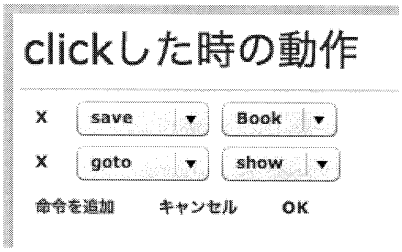

図 6. 選択式のイベントマネージャ

3.3 UI とモデルの相互変換[3]

WebAS はデータベースと連携したシステムが非常に多い。しかし、データベースとの連携はプログラミングの知識に加え、データベースの知識を必要とするためハードルが高い。また、UI を変更すると同じ様な変更を複数箇所にする必要があり、非常に手間である。そこで、本研究では UI とモデルの相互変換を用いることで、これを解決する。

3.3.1 UI からモデルへの変換

多くの WebAS ではモデリングするべきデータは入力、または出力の値として使われている。そこで、UI をデータスキーマとみなし UI からモデルへの変換を行う。

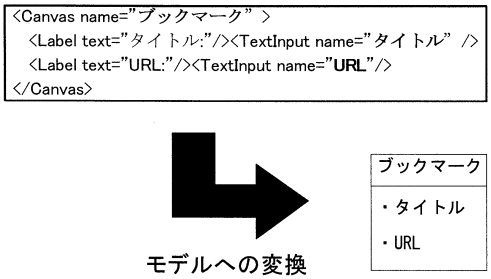

図 7. UI からモデルへの変換

本システムは内部的に UI を XML によるツリー構造で表現している。図 7 の様に UI の階層構造をそのままデータ間の関係へと変換することで UI からモデルへの変換を実現している。

3.3.2 テンプレートによるモデルから CRUD ページの生成

一般的に、一つのデータモデルに対応する CRUD ページはデータモデルの種類に関わらず、似た様な構造になりやすい。そこで、データモデルから CRUD に対応する各ページを自動生成することで、定型作業を省き、開発の初期コストを軽減することが出来る。CRUD ページの生成は以下の様な手順で行う。

1. UI の記述
2. UI をデータモデルに変換
3. テンプレートの読み込み
4. CRUD ページの生成

3.3.3 同じデータモデルを参照する UI の変更の同期

テンプレートによる CRUD ページの生成では、UI 作成の初期コストは軽減されるが、変更のコストは軽減されない。そこで、同じデータモデルを参照している場合、ある UI に加えた変更は他の UI でも変更される可能性が高いという点に注目した。

UI 変更同期の流れは図 8 の様に、まず任意のページに UI を追加し、モデルへと変換する。そして、同じモデルを参照している UI に差分の UI を追加する。

このような手順により、同じデータモデルを参照する UI の同期を行い、一つの UI を変更すると他の UI にも同じような変更をしなければならない問題を解決する。

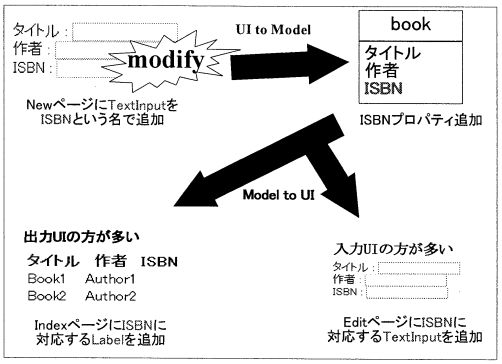

図 8. UI の変更の同期

4 実験

4.1 Service の拡張の検証

Service の複製機能を用いて、元の Service に影響を与えずに、Service の拡張が可能かの検証を行った。検証項目は以下の三つである。

- 元 Service のコンテンツを引き継ぐ
- 複製先への変更は元 Service に影響をあ立てていない

検証には、Book(Title, Author)に対応する単純な CRUD Service A を利用した。A の Service 概要、一覧ページは図 9、図 10 の様になった。

図 9. Service A の概要

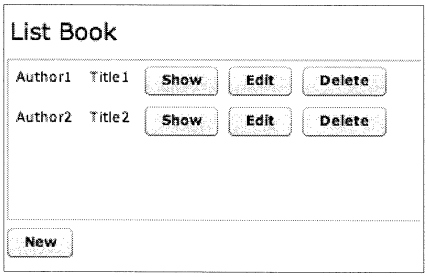

図 10. Service A の一覧

A を複製し、ISBN を追加した A'を作った。A'の Service 概要は図 11 の様になった。図 9 と図 11 より、作者名の変化から Service A'が test ユーザが作成した ServiceA の test2 による複製だと分かる。

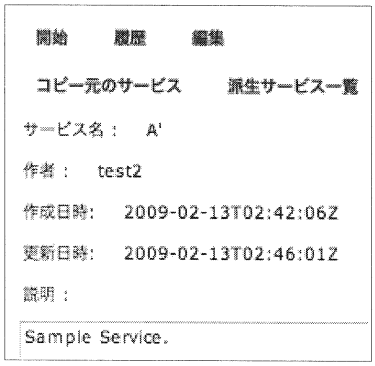

図 11. Service A'の概要

A'でコンテンツを追加した時、A, A' の一覧ページはそれぞれ図 12、図 13 となった。図 10 と図 13 の比較からコピー元である A のコンテンツ(Author1, Title)と(Author2, Title2)が引き継がれているのが分かる。

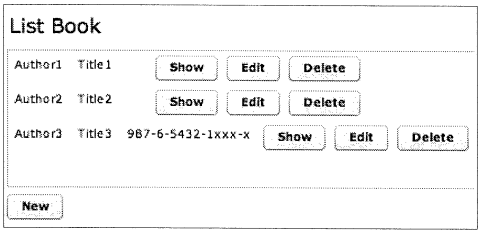

図 12. Service A'の一覧

また、より、A'で追加された(Author3, Title3, 987-6-54321-xxx-x)は、拡張項目である ISBN を省いた(Author3, Title3)が、A にも追加されていることが分かる。これによって A'での拡張は A には影響を与えていないことが分かる。

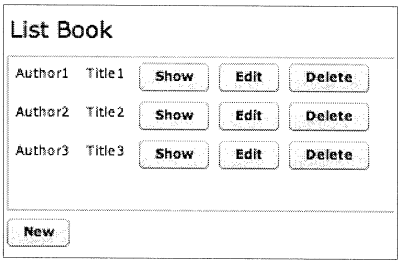

図 13. 追加後の Service A の一覧

以上の結果により、Service の複製機能を用いることで、元の Service に影響を与えずに、Service の拡張が可能であるといえる。

4.2 UI とデータモデルの相互変換の検証

UI とデータモデルの相互変換の有効性を検証するために、Ruby on Rails(RoR)と SoS で同じシステムを開発し、手順数の比較を行った。システム開発に際して、仕様変更を2度行い、SoS の開発モデルで頻繁に行われる拡張時のコストを中心に比較した。

開発したシステムは以下の各仕様のデーモデルに対して「一覧」「投稿」「詳細表示」の三つの機能を提供する単純なCRUDシステムである。開発は、まず仕様1を実装し、それを拡張して仕様2、さらにそれを拡張して仕様3という順番で行った。

- 仕様1 Bookmark(Title, URL)
- 仕様2 Bookmark(Title, URL, Comment)
- 仕様3 Bookmark(Title, URL, Comment, Date)

4.2.1 実験結果

結果は表1のようになった。表内の数字は作成および仕様変更にかかった手順数である。仕様1の作成に関するコストはRoR、SoSともに同じであった。しかし、変更に関するコストはSoSの方が一定して小さくなった。また、RoRは作業内容が各ページに対する変更であった。そのため、ページ数が増加すると手順数も増加すると考えられる。一方、SoSでは作業内容は「CRUDページのUIのいずれかを変更」「UIの変更の同期」であり、ページ数に依存しない。そのため、データモデルを参照するページが多い方がSoSに有利であると考えられる。

表1. RoRとSoSの手順数の比較

	仕様1の実装	仕様2への変更	仕様3への変更
RoR	5	5	5
SoS	5	2	2

5 まとめ

本稿では本システムを用いることで、拡張可能なWebASの作成及びUIとモデルの相互変換による開発の簡略化が行われることを確認した。しかし、研究を進めるうちに新たな問題点及び必要機能があることが明確になった。

①優良Serviceへの効率的なアクセス

現在は、同一SoSプラットフォーム上に存在するServiceへのアクセス方法として、単純なキーワード検索と、更新順のアクセスしか用意していない。将来的にユーザやServiceが増えてきた場合、膨大な量のServiceから自分に最適な物を探すのは困難である。また、簡単にクローン・サイトが作れるという性質上、フィッシング詐欺が行われる可能性もある。そのため、アクセスランキングや協調フィルタリングなどを用いてより効率的な検索方法を提供し、より自分の目的に合ったServiceを利用できるようにすると共に、フィッシングサイトなどの不適切なコンテンツへのアクセスをしづらくする必要がある。

②複雑なUIへのモデル変換の適用

本システムで利用しているUIとモデルの相互変換は、まだ単純なUIに対してしか適用していない。たとえば、Blogの記事の様にコンテンツの中にコメントがあるといった入れ子構造のUIに対応できない。これを実現することで、より多くの種類のWebASの開発が容易になる。

今後は、これら機能を実現するとともに、システムの公開実験を行い「実際にマス・コラボレーションによるWebAS開発が行われるのか」「どのようなWebASが変更されやすいのか」等と行った検証を行う予定である。

参考文献

[1] ドン・タップスコット, アンソニー・D・ウィリアムズ, 井口耕二 訳：ウィキノミクス マス・コラボレーションによる開発・生産性の世紀へ 42-45, 日経BP(2007)

[2] Gerhard Fischer, Elisa Giaccardi, Yunwen Ye, Alistair G. Sutcliffe and Nikolav Mehandjiev. "Meta-Design: A Manifesto for End-User Development."Communications of the ACM , 47(9): 33-37, 2004.

[3] 中田, 深海:データモデルとUIの相互変換を用いたウェブサービス開発環境「SoS」FIT2008 第7回情報科学技術フォーラム, B-017(2008).