DataWikiを活用した社会的アプリケーションの構築

Constructing Socialized Application using DataWiki

江渡 浩一郎　　濱崎 雅弘　　沢田 洋平*

Summary. インターネット上の全てのWebサイトを処理対象とするアプリケーションを設計することは難しい．それぞれのWebサイト毎に異なる処理方法を記述する必要があり，また時間によって対象が変化してしまうからだ．本研究では，膨大な種類を持つ処理対象を，対象ごとに分割してプログラミングできるようにし，ユーザが共同的にプログラミングすることで処理可能とする**社会的アプリケーション**(*Socialized Application*)を提案する．不特定多数のユーザが各々の処理対象毎の処理方法を記述し，データを保持する目的に特化したWikiサイト(DataWiki)に蓄積する．アプリケーションは必要に応じてそのデータを参照しつつ動作することで，大量かつ頻繁に更新される処理対象を扱うことができるようになる．本稿では，提案手法の実例であるAutoPagerizeと，その基盤となるDataWikiの実装であるWedataの運用結果を分析し，本手法の有用性について考察する．Wedataを用いた社会的アプリケーションは種類・利用者ともに増加傾向にあり，2011年7月時点で月間約137万ユニークIPから利用されている．本手法では，網羅性と正確性の維持向上に共同編集が寄与すること，アプリケーションを前提としたデータ構築がコールドスタート問題の解決に寄与することがわかった．

1　はじめに

Webを処理対象とするアプリケーションを構築するには，特有の難しさがある．たとえばスクレイパーはWebサイトにアクセスして特定の情報を抽出するプログラムだが，処理対象毎に異なるプログラムを書く必要があり，対象となるサイトが変化したらプログラミングしなおす必要がある．何千ものWebサイトをこの手法でプログラミングし，維持・管理することは不可能に近い．

共同的なプログラムの設計手法として，オープンソースソフトウェア(OSS)の開発モデルが知られている．Raymondは，著書『伽藍とバザール』で，多数の開発者が同時並行的にプログラムを書く手法を「バザールモデル」と名づけた．多くの人が開発に参加すれば誰かが必要な機能を実装するだろうし，たくさんの目にさらされればバグも発見されるはずと主張した．しかし実際にはOSSにも限度がある．ソフトウェアの機能追加や修正は，中心となる開発者が受け取り，適正かどうかを検査してから組み込むが，あまりに多数のパッチがある場合にはここがボトルネックとなる．何千もの日々変化するWebサイトを処理対象にする場合，中心となる開発者の負荷が高すぎることになる．しかし，安全性の検査が必要ない個所だけなら，検査無しにソフトウェアに組み込むことができるはずだ．問題が起こった場合は後から対処すればよい．

本研究では，インターネット上の集合知を利用して動くアプリケーションを**社会的アプリケーション**(*Socialized Application*)という新しい概念で呼ぶことにし，本稿では社会的アプリケーションを設計する手法について述べる．社会的アプリケーションは，大量かつ頻繁に変化する対象に追従するために，対象の処理方法をソフトウェアパッケージ外に出し，不特定多数のユーザが共同的に処理方法をプログラミングできるようにする．ソフトウェアは，処理方法を記した情報を外部に置かれたデータベースから受け取り，その処理方法に従って処理する．データベースはWikiの利用者参加型設計プロセス[2]を採用し，誰でも編集可能にする．これによって，エンジニアは処理方法をプログラム外にアウトソースすることができる．

本稿の構成は以下のとおりである．まず2章にて提案手法の概要について述べる．3章および4章にてシステムの利用ログ分析結果を示し，本手法の有効性とその特徴を示す．5章にて関連研究について述べ，6章にて本稿をまとめる．

2　社会的アプリケーション

大量かつ頻繁に変化する対象に追従するためには，各対象の処理方法を大量に蓄積し，適切に管理する必要がある．具体的には以下の2点が求められる．

網羅性 多くの処理対象に対する処理方法が記録されていること．頻繁に用いられる対象は当然のこと，めったに用いられない対象もカバーできていることが望ましい．

* Koichiro Eto, Masahiro Hamasaki, 独立行政法人産業技術総合研究所, Yohei Sawada, 産業技術総合研究所 (研究当時)

表 1. AutoPagerize のデータ例．対象ページを示す url 属性，次ページへのリンクを示す nextLink 属性，本文情報を示す pageElement 属性がある．

属性名	属性値
url	`\^https?://[\^./]+\.google(?:\.[\^./]{2,3}){1,2}/(?:c(?:se|ustom)|search|webhp|\#)`
nextLink	`id("pnnext")|id("navbar navcnt nav")//td[span]/following-sibling::td[1]/a|id("nn")/parent::a`
pageElement	`id("res")[not(@role)]/div[ol or div] | id("ofr") | id("rso")`
exampleUrl	`http://www.google.com/search?q=AutoPagerize`

正確性 正しい処理方法が記録されていること．対象が変化した場合にはできるだけ早く更新されることが望ましい．複数の正しい処理方法がある場合には，より適切な処理方法が記録されていることが望ましい．

インターネット上の全ての Web サイトといった膨大かつ日々変化するデータを対象にするとき，網羅性と正確性の維持は少数のアプリケーション開発者で対応できるものではない．

大量かつ頻繁に変化する対象に追従するには，対象の処理方法をソフトウェアパッケージ外に出し，不特定多数のユーザで共同的に処理方法を構築することで対応可能となる．このような手法で実現するアプリケーションを，本研究では社会的アプリケーションと呼ぶことにする．社会的アプリケーションは，著者の 1 人が開発した AutoPagerize を一般化した概念であり，AutoPagerize を構成する手法を *SITEINFO* **方式**と名づけている．以下，SITEINFO 方式とその方式を用いて実装されたアプリケーション AutoPagerize[9] の説明をし，それを踏まえて提案手法について述べる．

AutoPagerize とは，様々な Web サイトでページの自動継ぎ足しを実現するブラウザ拡張である．AutoPagerize が行うページの自動継ぎ足しとは，ページ中に「次へ」などといったリンクで次に読み込むページが指定されている場合，ページを下の方までスクロールすると自動的に次のページを読み込み，現在のページに継ぎ足して表示する．これによりユーザはページをスクロールするだけで次ページを読むことができる．

AutoPagerize はページ中から次ページを示すリンクを見つけるために，Web サイト毎に異なる処理方法が必要となる．表 1 に，Google の検索結果ページの処理方法を示す．AutoPagerize は，表示中のページの URL が url 属性にマッチする場合，nextLink 属性で示される次ページへのリンクを探し，ページが下までスクロールされたらリンク先を読み込み，pageElement 属性で本文情報を切り出し，現在のページに挿入する．

ここで AutoPagerize は処理対象となる Web サイト毎に，(1) どの URL を処理対象とするか，(2) ページ中のどの部品を処理対象とするか，という 2 種類の情報の組を必要とする．AutoPagerize では，この 2 種類の情報の組を SITEINFO と呼んでいる．SITEINFO 方式とは，このような 2 種類の情報の組を不特定多数のユーザに構築してもらい，そのデータの集合 (データセット) を用いてアプリケーションを実現する手法を言う．

SITEINFO 方式では，ページ中の処理対象の指定に，XML 文書中の場所を指定するための記法 XPath を用いている．XPath は高度な記述力を持つが，汎用的なプログラミング言語ではなく，単にページ中の場所を指定するだけで，内容を改変する能力を持たないため，悪意のあるコードを埋め込むことはできない．そのため，第三者が記述した XPath をブラウザで解釈してもセキュリティ上の問題が発生しない．

DataWiki とは，データを保持する目的に特化した Wiki サイトである．我々は DataWiki の実装として Wedata を構築し，運用している [7]．AutoPagerize は，この処理方法を集積したデータセットを Wedata 上に保持している．DataWiki は，spam 防止のために最初にユーザ登録する必要があるが，登録ユーザであれば誰でもデータの追加・編集・削除を行える．

このような，SITEINFO 方式が示した多数の Web サイトを処理可能にする手法を用いて実現されるのが社会的アプリケーションである．社会的アプリケーションは下記の特徴を持つ．

1. 膨大かつ日々変化する物を処理対象とする．
2. 各々の処理対象毎に，処理方法を分割して記述できる．
3. 汎用的なプログラミング言語ではなく，安全性の検査を行える表記方法で記述する．
4. 個々の処理方法を DataWiki に蓄積し，不特定多数が編集できるようにする．

これらの条件を満していれば，具体例である AutoPagerize と同様に，膨大な処理対象を扱うアプリケーションが実現可能であると考えられる．ゆえに，一般的なアプリケーションの新カテゴリーとして，社会的アプリケーションを提唱するものである．

表 2. アクセス数の多いデータセット上位 10 位までの詳細．ユニーク IP 数，参加編集者数 (のべ)，保有アイテム数，複数回編集されたアイテム数，その比率，共同編集されたアイテム数，その比率を示す．

順位	ユニーク IP	データセット名	編集者	アイテム	複数編集	比率	共同編集	比率
1	1035312	AutoPagerize	927	3251	2470	76.0	2104	64.7
2	327431	LDRize	61	515	298	57.9	232	45.0
3	325698	HatenaBookmarkUsersCount	6	8	7	87.5	4	50.0
4	33903	LDRFullFeed	353	3085	1567	50.8	834	27.0
5	23946	Favicons	3	7059	580	8.2	1	0.0
6	4400	Redirector	34	90	75	83.3	30	33.3
7	3734	MobileGatewayRefusers	1	7	1	14.3	0	0.0
8	3693	Replacer	6	2331	95	4.1	1	0.0
9	3344	Lookup	3	42	29	69.0	1	2.4
10	2007	UrlCleaner	8	27	15	55.6	11	40.7

3 DataWiki のログ分析

本節では社会的アプリケーションの基盤となる DataWiki 実装である Wedata の利用状況の分析を行い，提案手法の有用性を示す．Wedata は 2008 年 3 月に公開開始し，2011 年 6 月には月間 4700 万回以上のアクセス，134 万ユニーク IP からのアクセスを記録している．保有データ数は，2011 年 8 月現在で，データセット数 155 個，アイテム総数 52,000 個以上である．

表 2 に Wedata 上のデータセットのうち，アクセス数の多いデータセット上位 10 位までの詳細を示す．編集者数は当該データセットのデータ入力または編集を一度でもしたことがある Wedata 登録ユーザ数である．共同編集アイテム数とは，共同編集（一つのアイテムを複数の編集者が修正する）が少なくとも一度行われたアイテムの数である．共同編集は DataWiki 特有の振る舞いであり，共同編集アイテム数は DataWiki の活用状況を示す指標といえる．

Wiki を用いても，共同編集が起こらないこともありえる．しかし Wedata 全体で編集者数（登録ユーザ数）は 2023 人，共同編集アイテム数は 3000 個を超え，DataWiki として活用されていることがわかる．表 2 を見ると，特にアクセス数上位のデータセットにおいて共同編集の割合が高いことがわかる．共同編集という DataWiki 特有の機能を活用することで，データセットの網羅性・正確性が維持向上されることにより高い価値を持続させることができ，結果として多くのアクセスを集めたのではないかと考えられる．

4 各データセットの分析

提案手法では DataWiki におけるデータセットの構築に，ユーザの参加が生じることが重要である．本節では，ユーザ参加が積極的に行われている指標として共同編集されたアイテム数の比率を用い，共同編集が行われたデータセットとそうではないデータセットを分析し，DataWiki 活用の要因について考察する．共同編集が行われているデータセットとして，AutoPagerize，LDRize を，共同編集が行われていないデータセットとして，Tween，Favicons を用いる．

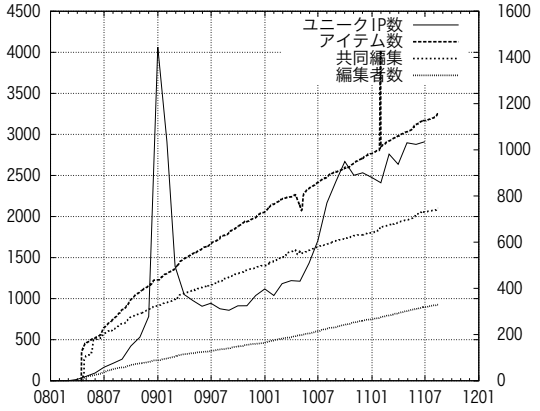

図 1. データセット AutoPagerize の時系列変化．期間：2008 年 3 月〜2011 年 8 月．

4.1 共同編集を活用している例

図 1 は AutoPagerize データセットのアクセス数，アイテム数，共同編集アイテム数，編集者数の時系列変化を示している．左側の縦軸がアイテム数，共同編集アイテム数，編集者数を，右側の縦軸がアクセス数（月間ユニーク IP 数，単位：千回）を示している．なお，以降に示す図 2〜4 のグラフは全て同様のものである．本データセットは AutoPagerize というブラウザ拡張から利用するためのデータセットである．編集者数は 927 人，アイテム数は 3,251 個である．共同編集アイテムは 2,104 個と全体の 64.7%を占め，共同編集が活発に行われていることがわかる．

図 2 は LDRize データセットのアクセス数，アイ

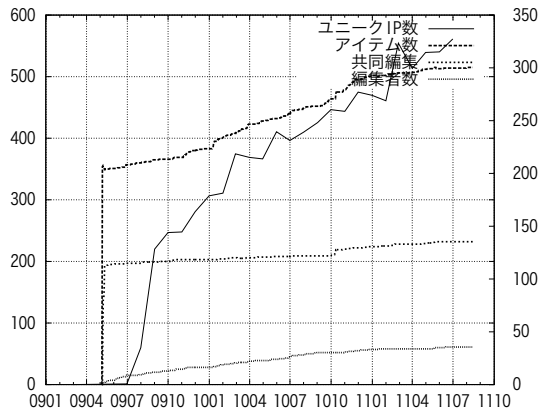

図 2. データセット LDRize の時系列変化．期間：2009 年 5 月〜2011 年 8 月．

テム数，共同編集アイテム数，編集者数の時系列変化を示している．本データセットは，任意の Web サイトで *liverdoor Reader (LDR)*[1] という RSS リーダと同じ操作性で記事を読むためのブラウザ拡張から使われるデータセットである．編集者数は 61 人，アイテム数は 515 個と，AutoPagerize の約 1/10 だが，共同編集アイテム数は 232 個で，45.0%と高い比率を示している

これら 2 つのデータセットは，いずれも SITE-INFO 方式のアプリケーションから利用されることを前提としている．各アイテムは対象ページを示す URL の正規表現と，ページ中の対象個所を示す XPath を持つ．どちらもある程度複雑であり，また，サイト更新等で修正が必要となるが，共同編集により正確性が維持されている．アプリケーションの性質上，全ての Web サイトをカバーすることが望ましいが，多くの編集者により網羅性の維持向上が図られている．

2 つのデータセットどちらにおいても，まず最初に数百個のデータが入力されていることがわかる．これらは Wedata を利用する前から存在していたアプリケーションであり，Wedata に投入された数百個の初期データは事前に作られたものと考えられる．一般に DataWiki 上のデータセットはアプリケーション開発者自身が入力するところから始まる．この制約が，情報共有システムに起こりがちなコールドスタート問題（十分なデータが集まるまでシステムが機能しない）を回避することにつながっていると考えられる．

DataWiki にデータ入力を行うユーザは，そのアプリケーションのユーザであることが一般的だろう．データを入力すれば即座に自分が今使っているアプリケーションに反映され，データ追加の恩恵を得ることができる．ユーザに利用を強制させることができないインターネット上の情報共有システムにおいては，短期的メリットと長期的メリットの両輪が必要だが [4]，本手法ではそれが実現されているために，ユーザはデータ入力を行うのだと考えられる．

これらのデータセットは作成時点ではいずれも特定のアプリケーションからの利用を前提としたものだったが，データが蓄積された結果，他アプリケーションからも利用されるようになっている．例えば AutoPagerize データセットは，AutoPagerize と同等の機能を実現する AutoPatchWork というブラウザ拡張からも利用されている．他に，LDRize データセットは，はてなブックマーク拡張（はてなブックマークのブックマーク数や追加ボタンを表示できるようにするブラウザ拡張）からも利用されている．このアプリケーション拡張は，LDRize データセットでは不具合が生じるデータを回避するために，はてなブックマーク拡張専用の別のデータセットも持っていて，両方を並用している[2]．このようにデータセットを外部に出すことで，当初は共同編集のメリットを持つが，後に他アプリケーションとの共同利用というメリットも持つことになる．単なる共同利用だけではフリーライドとなるが，データセット利用者が増えることにより編集者およびアイテム数の増加につながるため，結果的に元アプリケーション利用者にとってもメリットとなる．

2 つのデータセットどちらにおいても，アイテム数および編集者数は緩やかに線形増加する傾向がみられる．一方で利用者数（月間ユニーク IP 数）は，ある時点で飛躍的に増加する傾向が見られる．利用者数の飛躍的増加にも関わらずアイテム数や編集者数の増加傾向に大幅な変化がないことから，初期ユーザが積み上げてきたデータがある一定ラインを越えたときにアプリケーションの価値が大幅に増加し，一般ユーザの大量獲得につながったのではないかと考えられる．このような利用者数−編集者数−アイテム数の関係をモデル化することで，データセットの現状把握や将来予想が可能になると考えられるが，これは今後の課題である．

4.2 共同編集を活用していない例

図 3 は Tween データセットのアクセス数，アイテム数，共同編集アイテム数，編集者数の時系列変化を示している．Tween とは，Windows 上で動く Twitter クライアントである．Twitter API を使用せずに動作するため，API 制限数以上の情報を扱えるクライアントとして人気を博したが，その Twitter の HTML をパースするための情報を Wedata 上で管理していた．本データセットの対象となるサイト

[1] `http://reader.livedoor.com/`

[2] これが HatenaBookmarkUsersCount がアイテム数が 8 個だけなのにも関わらず多くのアクセス数を持つ理由である．

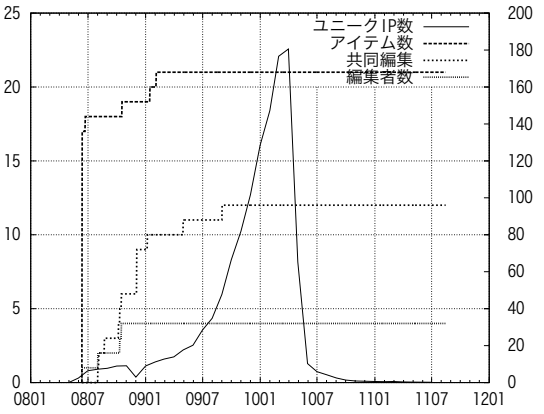

図 3. Tween のデータセット．期間：2008 年 6 月〜2011 年 8 月．

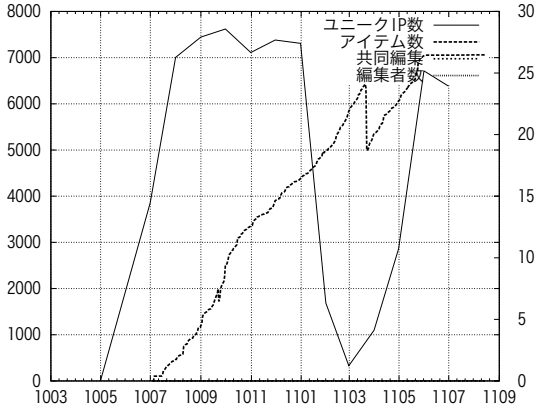

図 4. Favicons のデータセット．期間：2010 年 7 月〜2011 年 8 月．

は Twitter 一つだけであり，アイテム数は多くない．編集者数はおそらく Tween の開発者のみと考えられる．最初に，必要なアイテムのほぼ全て (90%) が登録され，2009 年 7 月ごろより利用者数が大幅に増加し始めるが，アイテム数に変化は無い．利用者数は指数関数的に急上昇したが，2010 年 5 月に Tween が Wedata の利用を停止したため，その後，データセットはほとんど利用されていない．

図 4 はデータセット Favicons のアクセス数，アイテム数，共同編集アイテム数，編集者数の時系列変化を示している．本データセットは，Google Reader という RSS リーダに favicon(Web サイトを示すための小さな画像) を表示させるための Google Reader Plus から使われるデータセットである．アイテム数は多いが編集者数は 3 人，共同編集アイテム数は 1 個と，共同編集はほとんど行われていない．

Tween データセットでは，対象ページが特定サイトに限定されるため，共同編集が期待されるのはサイトの更新への対応のみとなる．しかしサイトの更新に対応しないとアプリケーション側のサービスに大きな影響を与えてしまうため，開発者は自ずと積極的に対応する必要がある．DataWiki にデータを置くことには，通信トラブルによってデータにアクセスできなくなる可能性があるなどのデメリットがあり，そちらの方が大きくなる．大量のデータセットを対象としない限り，本手法は必ずしも有用ではないことを示す例である．

Favicons データセットでは，対象ページが無数にあるため本手法に適しているように見える．しかし favicon の抽出は Web サイトのフィードから機械的に行えるため，人手を必要としない．本データセットのデータは機械的に入力されており，ある種のキャッシュとして使われている．大量のデータセットが対象だとしても，網羅性と正確性の維持向上に人手を必要としない場合は，本手法は効果を発揮しないことを示す例である．

5 関連研究

Wedata のように自由に編集が可能なデータベースは，Google DataWiki[3]，Freebase[1]，Semantic Wikipedia[5] など多数提案されている．Freebase, Semantic Wikipedia はデータを機械可読な形式で記述できることに重点が置かれており，結果として Wikipedia など既存のデータを変換し蓄積することが主な用途となってしまっている．Google DataWiki は本手法の DataWiki に概念的に類似しているが，現在はあまり活発的なデータ登録は行われていない．社会的アプリケーションの基盤として使うことで，活用されるようになるだろうと考える．

PodCastle では，CGM 的に作成されたウェブ上の辞書を利用したりユーザが入力した音声認識結果の訂正情報を利用することで，多様な音声データの認識を可能にしている [8]．森近らのイベント情報構造化システムでは，ユーザが入力した正規表現を用いることで，メールのような色々な書き方がまじったテキストから日付情報や地理情報を抽出している [6]．これらのシステムでは，ユーザが入力したデータを共同編集することはできず，機械学習のための学習データとして扱われる．DataWiki では共同編集によって集合知の集約を図るが，PodCastle や森近らのシステムでは機械学習によって集約を図っているといえる．このような DataWiki を用いないアプローチによる社会的アプリケーションの構築については，今後の検討課題である．

6 まとめ

本研究ではウェブ時代のアプリケーションとして社会的アプリケーションを提案し，その基盤となる

[3] http://datawiki.googlelabs.com/

DataWiki の実装である Wedata のログ分析を通じて，本手法の有用性と特徴を示した．

社会的アプリケーションはユーザの参加によって作りだされるため，ユーザの行動原理を理解することがアプリケーションの設計にとって重要となる．今後はユーザの振る舞いに関する分析を行い，知見を蓄積することで，より多くの社会的アプリケーションが生み出され，利用される環境を作りたいと考えている．

謝辞

本研究は，ngi group 株式会社との共同研究の一環として実施された．本研究の一部は JST, CREST の助成を受けた．本論文をまとめるにあたって国立情報学研究所の武田英明教授には多くの助言をいただいた．感謝の意を表する．データ構築に関わった全ての利用者に心から感謝の意を表する．

参考文献

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proc. of SIGMOD '08*, pp. 1247–1250, 2008.

[2] 江渡 浩一郎. パターン，Wiki，XP：時を超えた創造の原則. 技術評論社, 2009.

[3] B. S. Noveck. *Wiki Government: How Technology Can Make Government Better, Democracy Stronger, and Citizens More Powerful.* Brookings Institution Press, 2009.

[4] H. Takeda and I. Ohmukai. Building semantic web applications as information/knowledge sharing systems. In *Proc. of UserSWeb '05*, 2005.

[5] M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, and R. Studer. Semantic Wikipedia. In *Proc. of WWW '06*, pp. 585–594, 2006.

[6] 森近 憲行, 濱崎 雅弘, 亀田 尭宙, 大向 一輝, 武田 英明. 機械学習とユーザ知識を用いたイベント情報の構造化. 人工知能学会論文誌, 26(2):335–340, 2011.

[7] 江渡 浩一郎. 集合知データベース Wedata の利用動向の分析. 日本ソフトウェア科学会第 28 回大会論文集, 2011.

[8] 後藤 真孝, 緒方 淳, 江渡 浩一郎. PodCastle: ユーザ貢献により性能が向上する音声情報検索システム. 人工知能学会論文誌, 25(1):104–113, 2011.

[9] 沢田 洋平, 江渡 浩一郎. 集合知による Web ページの構造情報の収集. 信学技報, 第 AI2008-21 巻, pp. 27–32, 2008.

アピールチャート

未来ビジョン

筆者はこれまで 8 年に渡って Wiki の研究を継続してきた．その原理を一言で言えば**全てを Wiki 化する**となるだろう．筆者が構築した qwikWeb は，メーリングリストに Wiki を取り入れることで**グループコミュニケーションを Wiki 化する**ことが目的だった．本研究は，**プログラミングを Wiki 化する**ための最初の一歩になると考えている．

このように全てを Wiki 化する研究の最終目標はどこにあるだろうか．筆者は，Wiki の原理を利用した政府を実現すること，つまり**国家を Wiki 化する**ことにあると考えている．Wiki と同じように各々の参加者が持つ知見を一カ所に集結させ，共同的な手法で政策決定を行えるようにすることである．Noveck は，その著書『Wiki Government』[3] において，Wiki の原理を応用した政府を実現する構想を記している．著者は「Peer-to-Patent」という誰でも特許の先行調査を行える仕組みを構築した経験から，閉鎖的なプロセスによる政策決定を批判し，市民に開かれた政策決定を提唱している．

Web の発明者 Tim Berners-Lee は，Linked Data という，データ公開の際の設計指針を提唱している．実際に各国政府はオープンデータの取組みを開始しており，イギリス政府は Data.go.uk，アメリカ政府は Data.gov，日本の経産省はデータボックスを開設している．このように，政府がすでに保有するデータを公開していくことと同時に，参加者が共同でデータを構築していく仕組みも重要だろうと考えている．本研究が提唱する共同での情報構築メカニズムは，そのような共同でのデータ構築の助けになるのではないかと考えている．