なぜそんなにもWikiは重要なのか

江渡浩一郎
Eto Kouichirou

Mobile Society Review 未来心理［7号］掲載
2006年9月25日発行

モバイル社会研究所
Mobile Society Research Institute
http://www.moba-ken.jp/

なぜそんなにもWikiは重要なのか

江渡浩一郎 Eto Kouichirou

私は 2002 年より Wiki を専門として研究をしている。Wiki を専門としているというと、怪訝な顔をされることがよくある。Wiki のような単純な仕組みのどこに研究するようなことがあるのかと。しかし、Wiki は面白い。知れば知るほど、その奥の深さに驚くようになった。何が Wiki をそんなにも特別なものにしているのか、書いてみたい。

Wiki とは何か

Wiki は Web 上で情報を共同編集するためのシステムである。グループで情報をまとめるためのコラボレーション・ツールであるとも言える。ある人は Wiki を掲示板のようなものと言い、ある人はブログみたいなものだと言う。人によって言うことが異なるということは、それだけ幅のあるシステムであるということである。しかし様々な比喩で説明されても、どれもあまりしっくりこない面がある。結局、Wiki は Wiki であるとしか説明のしようがない。

Wiki の代表例として Wikipedia があげられることが多い［1］。Wikipedia はインターネット上で共同で百科事典を作り上げようというプロジェクトであり、不特定多数の集団が集まって作り上げられたにもかかわらず非常に高品質な百科事典が生まれていることから注目を浴びている。普通に Wiki と言って話が通じる場合はこの Wikipedia を指して Wiki であると認識している場合が多い。

Wiki と「Web 2.0」が関連して語られることもある［2］。Web 2.0 の特徴の一つ「ロングテール」は、ごく少数の人しか興味を持たない内容にも対応することで、全体として対応できる対象の幅がとても広くなることを言う。Wikipedia にはこんなマイナーなことまでと驚くようなコンテンツが載せられていることがあり、ロングテールを体現した内容になっている。

また現在「CGM（Consumer Generated Media）」という特徴も注目されているが、これはユーザが生成することによって成立するメディアという意味である。Wikipedia は不特定多数のユーザからの書き込みによって百科事典が作られており、CGM の代表例と言われることもある。

このように Wikipedia という具体例を見て、それがどのように語られているかを見てみても、Wiki が何なのかはまだ明解にはならない。

Wiki は 1995 年に Ward Cunningham 氏によって作られたシステムである。この最初の Wiki はすでに 10 年以上の歴史を持っているが、今でも普通にアクセスできる。

http://c2.com/cgi/wiki

私はこの Wiki の最初の姿がどのようなものだったのかを調べるうちに、実は本当の Wiki のはじまりはもっと前にあることに気付くようになった。本当の Wiki の発祥は、1987 年にある。この本当の意味での Wiki の起源に遡って考えることによって、私は Wiki が何なのかをようやく理解できるようになった。1987 年に一体何が起こったのか、見てみよう。

パタン・ランゲージのプログラミングへの応用

1987 年 の OOPSLA（Object-Oriented Programming, Systems, Languages, and Applications）、オブジェクト指向についての世界最大の学会において、Kent Beck と Ward Cunningham の二人は「オブジェクト指向プログラムのためのパターン言語の使用」という論文を発表した［3］。現在、プログラミングの分野には「デザイン・パターン」と呼ばれる大きな流れがあるが、この論文はその潮流の一番最初に位置する記念碑的論文である。彼らはこの論文で、プログラミングへのパタン・ランゲージ（Pattern Language）[1] の導入を提唱した。

オブジェクト指向とはプログラミングにおける方法論の一つである。それまでのプログラミング言語は、主に手続きに注目して記述するものであり、手続き型プログラミング言語と呼ばれている。オブジェクト指向とは、対象となる物の性質に注目して記述することによってプログラムを理解しやすくするという方法論である。このオブジェクト指向は、当時新しく出てきたものであり、プログラム開発を大幅に改善する手法として熱狂的に迎えられていた。そのようなオブジェクト指向の概念が提唱されて間もない 1986 年に、第一回の OOPSLA が開催された。Ward Cunningham も Kent Beck も、その最初の OOPSLA で論文を発表していた。その第二回目にあたる 1987 年の OOPSLA で、彼らはパタン・ランゲージのオブジェクト

指向プログラミングでの利用を提唱したのだった。

パタン・ランゲージとは、建築家クリストファー・アレグザンダーが提唱した概念であり、建築において一般に良いとされるパターンを集め、それを辞書のようにまとめたものである。建築を設計する際にこのパタン・ランゲージにおけるパターンを使えば、すばらしい建築を設計できるようになる。そのような理論である。

従来の手続き指向だったプログラミングにおける設計から、オブジェクト指向の設計に移るには、設計における方法論そのものを新しくする必要がある。特に、ユーザ・インタフェースにおける設計に重きを置く必要があった。そのために、建築の分野で提唱されてきたパタン・ランゲージを、オブジェクト指向プログラミングに取り入れてみたというのがこの論文の趣旨である。実は論文というより正確にはテクニカル・レポートで、2ページくらいと非常に短い。実践を行ってみての簡単な記録となっている。

彼らはまず画面上のインタフェースにおけるパターンとして、下記の5個のパターンを提案した。

- 1. タスクごとのウィンドウ（Window Per Task）
- 2. ウィンドウに対してペインはできるだけ少なく（Few Panes Per Window）
- 3. 標準ペイン（Standard Panes）
- 4. 短いメニュー（Short Menus）
- 5. 名詞と動詞（Nouns and Verbs）

このパターンをアプリケーション設計のチームに提供したところ、彼らは1日ほどでよくできたインタフェースを記述することができた。

このようなインタフェースにおけるパターンの他に、プログラミングにおけるパターンについても記述している。まずシステムを、ユーザが認識する世界におけるオブジェクトに分割していく。そしてそれをファイルフォーマットなどの低レベルのプログラムへとコーディングしていく。このようにユーザの世界からコードの世界へ順にブレイクダウンさせていく形でパターンを考えていった。

このパターンの具体例を見てみると、現在のデザイン・パターンを知っている人には奇妙に思われるかもしれない。現在のデザイン・パターンでは、このような画面上のインタフェース設計におけるパターンはユーザ・インタフェース・パターンなどといった別の言葉で区別される。単にパターンと言えば、主としてプログラミングのコードのレベルにおいて発生するパターンを指す。

この最初の論文におけるパターンと、現在のデザイン・パターンにおけるパターンは、なぜこのように違うものになってしまったのだろうか。

パタン・ランゲージからデザイン・パターンへの変遷

一番最初のパタン・ランゲージの応用におけるパターンと、現在のデザイン・パターンにおけるパターンとの違いを見てみると、まず対象が違うことに気付く。デザイン・パターンにおけるパターンでは主としてプログラミングにおける問題を扱っている［4］。例えばイテレータ・パターンは、プログラムにおける繰り返しを表現する際のパターンであり、アブストラクト・ファクトリー・パターンは新しいオブジェクトの生成を扱うパターンである。これらはどちらもプログラマがプログラミングを行う際に発生する問題を扱うもので、直接ユーザには関係してこない。しかし、パタン・ランゲージの利用におけるパターンでは、まず直接的なユーザとの接点であるユーザ・インタフェースから議論がはじまっている。ユーザが認識する世界から、実際のコーディングにおける一連の流れにおけるパターンを、パタン・ランゲージの利用の対象として考えている。

このように、パターンは当初はユーザがプログラム開発にかかわる際の一連の流れを扱うものとして認識されていたが、徐々にプログラミングにおけるパターンに対象が変化していったように見える。この違いは些細なものだろうか。とてもそうとは言えない。この違いはその前段である目的設定の違いからやってくる。当初掲げていた目的とは何か。それは明解にこう記述されている。「ユーザは、自分自身のプログラムを書くべきなのである」と。つまり、従来ユーザが仕様を規定しそれを開発者がプログラミングするという一方向的な流れであったものを、システムを利用するユーザ自身が自分自身のシステムを開発するという流れに変えていくべきだと言っているのだ。この目的設定が背景にあって、はじめてパタン・ランゲージの利用という話が出てくる。

この目的設定の違いは大変大きい。この違いは、パターンの捉え方の違いに起因するように思われる。パタン・ラ

ンゲージの利用におけるパターンでは、その生成的な性質が注目されていた。システムを構築していく過程でパターンを発見し、自分でパターンを作り、パタン・ランゲージを組み立てていく。そのように、パタン・ランゲージは自分で発見して組み立てていくものだった。ところが、「デザイン・パターン」は、著者であるエリック・ガンマがそのようにして発見したパターンをカタログとしてまとめたものである。そこに納められたパターンの質が非常に高かったためか、パターンは自分で発見するものとは受け止められなかった可能性がある。たしかにプログラミングにおけるパターンの利用は、現在ではデザイン・パターンとして普及し、プログラマの共通言語として発展し続けているのだが、当初のユーザと開発者の間の共通言語という理念は失われてしまっているように見える。

パタン・ランゲージとは何か

目的設定の重要性を考えるために、元々のパタン・ランゲージに遡って考えてみよう。

元々クリストファー・アレグザンダーがパタン・ランゲージを提唱したときに考えていたのは、建築家、即ち、建築を設計する人と、建築の依頼主、実際に建築を利用する人との間の垣根を無くすことである。

> アレグザンダー氏は、家やオフィスというものは、実際にそこにいる人たちの手によって設計され、作られるべきだと提案している。[3][2]

これは単に建築家と依頼主との垣根を無くせばいいということではない。その依頼主が望む建築をどのようなものにするべきかを一番よく理解しているのは、本来は依頼主自身であるはずだからだ。

> 氏がこう結論付けたのは、ある特定の構造への要求を一番よく知っているのは、彼ら自身だからだ。[3]

このとき、パタン・ランゲージはどのように機能するのか。簡単に言えば、建築家と依頼主との間の共通言語として機能する。お互いが共通にこの言語を使うことによって、同じ土台で設計を行っていくことができる。そのために考え出されたのがパタン・ランゲージだった。

では、なぜパタン・ランゲージを使えば共通の土台を構築できると言えるのか。実はこのパタン・ランゲージという言葉はその意味しているところがなかなか見えてこない。それは実は用語の選択によるところが大きい。パターンもランゲージも非常に一般的な言葉であり、その組み合わせの「パタン・ランゲージ」をある特定の意味で解釈するのは難しい。

例えば「パターン」という言葉は、通常は「繰り返し」という意味で捉えられる。「パターン化している」などと否定的な形で使われることもある。しかしここでの「パターン」という言葉は、単なる繰り返しという意味ではない。自然界を見まわすと、自然にできた形には、あるルールによって生成された形を見付けることができる。例えばヒマワリの花の中に規則的に並ぶ形を発見したり、泥が乾いていく時にできるひびわれに一定の法則を発見したりする。そのような、自然の法則の中に潜む、ある形態を発生させるようなルールを称してパターンと呼んでいる。

また「ランゲージ」という言葉も、通常は文字で表現された言語という意味に捉えられる。この「ランゲージ」も単純に言語と理解してはならない。自然言語はいろいろな部品の集合から成り立っている。文字が集まって単語ができ、主語、述語、助詞、動詞、形容詞などといった様々な区分があり、それらが集まり文ができ、段落になり、節になり、文章になる。文章が集まって本になり、全集になり、百科事典になったりもする。個々の部品間の関係は有機的で、様々な大きさ・種類の部品が集まることによって多様な意味が生まれる。そのような構造全体を指して「ランゲージ」と言っている。

このパタン・ランゲージは、その名も『パタン・ランゲージ』[5]という本があり、この本を読むと理解できることになっている。しかし、実はこの本は5冊からなる著作集の2冊目の本であり、パターンを集めたカタログ、いわば辞書のようなものである。パタン・ランゲージの理論や背景は、1冊目の『時を超えた建設の道』[6]を読む必要があるが、これは理論的な本であり、いきなり理論に取り組むとまたわかりにくいかもしれない。まず最初に具体的な事例から見るのがいいのではないだろうか。5冊からなる著作集の3冊目『オレゴン大学の実験』[7]もあるが、もう少しやわらかい実践の記録として、アレグザンダーの建

築にかかわってきた日本の建築家中埜博氏の記録［8］が参考になる。

この「パタン・ランゲージによる住まいづくり」という本は日本の建築状況において、どのようにパタン・ランゲージを実践していったのか、大変わかりやすい記録になっている。この詳細な記録ではじめて建築家と依頼主をつなぐ共通言語としてパタン・ランゲージを使うとはどういうことなのかが見えてくる。例えば「原寸大の設計」という手法がある。ある場所に家を建てる際に、図面で配置を考えた上で、現場に行ってそこに杭を打ったり線を引くことで原寸大の図面を書いていく。そのような原寸大の図面で見えてきた問題点を、その場で修正する。内装も同じように、家具と同じ大きさの段ボールをその場に配置して見え方を確認する。このようにして、図面で設計したものが現場でどのように見えるのかを実際に確認しながら、その場で設計を直していくのだ。

建築はソフトウェアと違って、一度建ててしまったものをそう簡単には変えられない。そのため、設計と施工を行き来するような手法はそれほど容易に実現できないように見えるが、建築という世界においてなんとかして設計と施工を行き来することができるように工夫したのが、パタン・ランゲージなのだ。

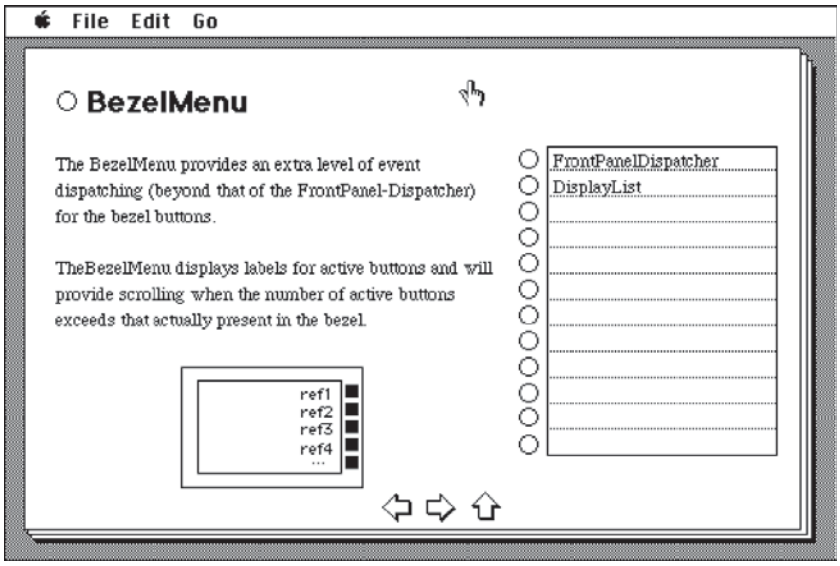
HyperCard における実験

1987 年には、もう一つ大きな事件が起こった。「HyperCard」の発売である。その 3 年前の 1984 年には、Apple 社より GUI を搭載したパソコン「Macintosh」が発売された。その GUI を最大限に生かしたのがこの HyperCard というソフトだった。

この HyperCard は非常に柔軟なソフトだったため、どのようなものかを説明するのは難しい。簡単には「カード型のデータベースソフト」と説明されることがある。例えば図書カードとして、一枚のカードに書名、著者、出版社などを記録して、文献目録を作ったりすることがある。そのようにカードの形に情報をまとめながら、データベースを作っていくことを、コンピュータの画面上で行えるようにしたソフトである。また、「HyperTalk」というスクリプト言語を備えており、動的な仕組みを埋め込ませることができた。カードによるデータと一緒に、カードを操作するインタフェースも含めて扱うことができ、このようにインタフェースとデータを一組にして「スタック」と呼んでいた。

Ward Cunningham は、この HyperCard に衝撃を受けた。彼は、HyperCard [3] を使って、自分が発見したパターンを収集・整理するためのパターン・ブラウザを作り始めた。実はこのパターン・ブラウザが発展してできたのが、今の Wiki なのである［9］。

この HyperCard によるパターン・ブラウザの画面（下図）が残っているので、見てみよう［10］(fig.1)。画面にはカードが表示され、左上には「BezelMenu」というタイトル、その下に説明文章、そして図が挿入されている。右側には関連するカードへのリンク一覧が示されている。カード下部の左右ボタンから一次元的に前後のカードを辿れる。上ボタンでそのカードの概要を示すカードにとぶ。カードを編集したい時は、メニューから編集を選ぶことですぐに WYSIWYG で編集できた。



「fig.1 BezelMenu についてのカード」

この時点で、すでに現在のWikiへつながる特徴が出ている。例えば、CamelCase[4]によるタイトル表記である。関連するカードへのリンクは、文中のリンクではなく右側のリンク一覧で示されている。なお、カード間の一次元的なつながりは今では無くなってしまっている。

カードの中身も見てみよう。これは「BezelMenu」というパターンだが、bezelはモニターの縁のことで、その部分に配置されたメニューという意味になる。モニターの右に縦にボタンが並び、そのすぐ横の画面内にボタンの説明が表示されている。画面内の説明は動的に切り替えられるので、場面毎にボタンの説明表示を変えられる。アメリカの銀行のATMでよく使われているインタフェースと言えばわかるだろうか。

つまり、これはユーザ・インタフェースにおけるパターンの一例を示したものなのである。このようにパターン・ブラウザにおけるパターンとは、ユーザ・インタフェースにおけるパターンも含めて扱うものだった。彼はこのHyperCardのスタックにパターンを収集・蓄積させていった。

このスタックを複数人で共有する実験も行った。何か更新があると、自動的にRecentChangesというカードからリンクがはられるようにした。このようにして、他の人が行なった更新をすぐに確認できるようにした。

この時、このパターン・ブラウザにもっとも欠けていた要素は、もちろんインターネットへの接続である。それを実現するためには、Webの誕生を待つ必要があった。

1995年、WikiWikiWebの誕生

Webが誕生したのは1991年だが、一般に普及がはじまったのは1993年のMosaicの誕生以降と言われている。Webサーバで動的な処理を行うためのCGIという仕組みも同じころに誕生した。そこからさほど時間がたっていない1995年3月25日に、Ward CunninghamはPerlとCGIを使って、最初のWikiWikiWebを作り出した［11］。彼は5月1日に、パターンを議論するためのメーリングリストでこのWikiをアナウンスした［12］。

実はこの記述は若干倒錯している。というのも、Wikiという名前が誕生したのはこの時点だったからだ。彼はこのWebサイトに「Portland Pattern Repository」という名前をつけた。彼が住んでいたポートランドの名前を冠したパターン貯蔵庫という意味である。しかし、編集できる仕組みを備えたWebサイトとしてのPortland Pattern Repositoryを区別するための名前が必要だった。そこでつけたのが「WikiWikiWeb」という名前である。つまりこの時「WikiWikiWeb」は特定のWebサイトを指す固有名詞だったのだ。

ここではじめて、現在Wikiとして知られるWeb上で手軽にページを編集できる仕組みが実現された。ちょうどHyperCardで、データとインタフェースが一組でスタックと呼ばれるように、ページとそれを編集するための仕組みが一緒になって「WikiWikiWeb」という名前で呼ばれることとなった。

この最初のWikiWikiWebを実現するためのソフトウェアは、「WikiBase」と呼ばれている。PerlによるCGIスクリプトで、ソースコードは331行と非常に短い。当時のソースコードは今も入手可能であり、自分のマシンで動かしてみることができる[5]。

本文中へのリンク埋め込みは、この時点で取り入れられた機能である。文中にCamelCaseで単語を記述すると、自動的にその単語をタイトルとするページへのリンクになる。もしその対象が存在しない場合は、その単語には「?」がついて、新しいページを作成するリンクになる。新しいページを作るときは、まず作成したいページへのリンクをどこかに埋め込み、それからそのリンクを辿ってページを作成する。このように、まず存在しないページへのリンクを作ってからそのページを作成するという手順になっており、必ずページからページへのリンクが作成されるような仕組みとなっている。

これまでのHyperCardの時からの大きな違いは、インターネットに接続されているということである。そのため、パターン記述の場であると同時に、コミュニケーションの場としても機能するようになった。不特定多数のユーザがパターンを収集する際に、その収集する方法についてのルールをどのように決めていけばいいのだろうか。解決策はある意味自明である。パターンを収集する方法についてのパターンを考え、それもまた同時にパターンとして記述していくのだ。このように、Wikiはパターン収集の場であると同時に、どのようにパターンを収集していくかのパターン、つまりメタなパターンの実践の場としても機能

することとなった。

このように特定の Web サイトとしての「WikiWikiWeb」が発展していくにつれ、自分のサイトでも同じような仕組みを動かしたいという人もあらわれてきた。最初の Wiki のプログラムである WikiBase のコードは公開されていたので、同じ仕組みを自分のサイトで動かすのは難しくなかった。プログラムを自分なりに書き換え、発展させたバージョンを公開する人もあらわれてきた。そのようなソフトウェアは「Wiki クローン」と呼ばれ、次第に「WikiWikiWeb と同じような機能を持つ Web サイト」を「Wiki サイト」と呼ぶようになっていった。そのような機能を持つ Web サイトやソフトウェアを総称して WikiWikiWeb、略称 Wiki と呼ぶようになっていった。つまり、固有名詞としての WikiWikiWeb から一般名詞としての WikiWikiWeb へ変化していった。

実は現在でも、WikiWikiWeb は固有名詞であり、Ward が立ち上げた最初のサイトだけを指すのに使うべきだという主張もある。そのような主張が背景とするところを理解すると「Wiki とは何か」がようやく見えてくるかもしれない。つまり、Wiki はただのシステムではない。ある特定の機能が実装されているだけで Wiki であると断言することはできない。プログラムで実現されたインタフェースと、それを利用するための方法論とが一体になって、はじめて Wiki なのだ。

エクストリーム・プログラミング

パタン・ランゲージのプログラミングへの応用は、その後どうなったのか。パタン・ランゲージの応用において提唱されたユーザ自身による設計や、開発と設計を頻繁に行き来する手法を聞いて、何かに似ていると思った人もいるかもしれない。そう、エクストリーム・プログラミング（Extreme Programming、略称 XP）という開発手法とそっくりなのである。それもそのはず、エクストリーム・プログラミングを提唱したのもまた、Kent Beck と Ward Cunningham の二人なのである［13］。

エクストリーム・プログラミングを日本語に直訳すると「極端プログラミング」となる。プログラミングにおいて一般に良いとされている手法を、極端なまでに実践してみようというそのような手法である。

例えばコード・レビューという手法がある。自分一人で書いたコードは、どうしてもバグがあるし、洗練されていないかもしれない。そこで、同じような開発者に自分のコードを見せて批評してもらうと、バグが発見されたり、コードが洗練されたりする。これをコード・レビューと言うが、この手法は大変有効だとされている。それならばこのコード・レビューを徹底して、ずっとコード・レビューし続けている状態にしてしまう、そのような考えから生まれたのがペア・プログラミングである。これは二人一組でプログラミングを行う手法であり、一台のコンピュータの前に二人の開発者が座り、一台のキーボードを交互に使いながら開発を進めていく。一人がキーを叩いている間はもう一人はそれをずっと見ている。このように、プログラムを書くこととコード・レビューを常に同時に進めることによって、コードからミスは減り、ずっと洗練されたコードが書けるようになる。

また、プログラムが正しく動作することを確認するためのテスト・プログラムを書くのも良い手法である。しかしプログラムを書いて一旦うまく動いてしまうと、後からそれをテストするプログラムを書くのは億劫になる。そこで、プログラムを書くよりも前に、まずテスト・プログラムから書き始めるという手法が考え出された。これをテスト・ファーストと言う。このようにテストを先に書くことによって、プログラムは書いたけどテスト・プログラムはまだ書いていないということは無くなる。

このような手法は一見極端に見えるが、実は最終的にはプログラマにとって非常に納得がいく開発手法となっているのである。この XP の提唱する手法の中に、ユーザの設計への参加が含まれている。それは、開発しようとするシステムがどのように動くべきなのかを一番よく知っているのはユーザだからである。そう、まさしくパタン・ランゲージの応用における主題がここで繰り返されている。これはなんら不思議なことではない。パタン・ランゲージの応用を提唱した二人がその流れを引き継いで発展させたのが XP だからである。端的に言えば、パタン・ランゲージの応用は、今では XP と名前を変えて発展していると言うことができるだろう。

Kent Beck はこの本の第 23 章「時を超えたプログラミングの道」において、建築設計について語っている。彼の

曾祖父は大工だった。彼の家族が引っ越すたびに、自分の家族のための家を設計して建てていた。アレグザンダーの夢は、このような空間設計の能力を取り戻させることだったと。

そしてまた、彼はアレグザンダーの失敗についても書いている。建築家という職業はあまりに昔から存在しており、社会的な役割が固定化してしまっている。建築家は自分たちの力を放棄せず、また利用者は要求する術を知らなかった。そのためアレグザンダーは最終的には建築家と利用者の間のバランスをとることに失敗した、と。

しかし、ソフトウェア開発における開発者と利用者の関係は、まだ固定した関係には至っていない。誕生して間もないコンピュータの世界であれば、その開発者と利用者という社会的な関係を含めて新しく定義しなおせるかもしれない。Kent Beckは「新たな社会構造を作る機会がある」と語っている。そう、つまりXPの本当の目的は「新たな社会構造を作る」ことにあるのだ。建築の世界では失敗した設計者と利用者の間のバランスをとるという試みが、コンピュータの世界では本当に成功しうるのだろうか。

可能性はある。それがWikiだ。まったく同じパタン・ランゲージの応用という実践から生まれた方法論であるWikiが、今では非常に広範囲な広がりを見せている。恐ろしいことに、利用者が同時に開発者でもあるという極端なコンセプトが、このWikiの世界では本当に受け入れられているのだ。建築の分野では失敗したパタン・ランゲージという試みが、ここではまさしく成功しているように見える。このことが持つ意味を過小評価してはならないだろう。そう、Wikiの本当の目的は「新たな社会構造を作る」ことにあり、その試みは今まさしく本当に成功しつつあるのだ。

Wikiから建築へ

まとめとして、整理しよう。建築の分野において提唱されてきたパタン・ランゲージは、Kent BeckとWard Cunninghamの二人によってプログラミング分野への応用が提唱され、実践されてきた。その過程で生み出された方法論であるWikiは、今では非常に広範囲な広がりを見せ、受け入れられている。また、パタン・ランゲージの利用から発展したXPは、ソフトウェア開発の分野で発展し続けている。パタン・ランゲージは、元々の出発点である建築の分野では、受け入れられたとは言い難い。しかし、そのコンピュータ分野における応用は、今では積極的に受け入れられるようになった。

このような違いはなぜ生まれたのか。建築とコンピュータという対象の違いだけなのか。もしかしたら他に違いがあるのではないだろうか。もしあるとすればどのような違いか。

この次に考えるべきことは、この逆の展開なのではないだろうか。つまり、WikiやXPといったコンピュータ分野におけるパタン・ランゲージの成功を元に、それを建築に逆輸入することである。WikiやXPを基盤とした建築、それはどのように存在しうるのだろうか。私が芸術家として、また研究者としてWikiに取り組み続けている理由はそこにある。

謝辞

本論文は、WalWiki作者の塚本牧生氏、SHIMADAこと島田慶樹氏との共著の論文から生まれた［14］。ここに感謝の意を表する。

[脚注]

1 「Pattern Language」の日本語表記は「パタン・ランゲージ」「パターン言語」などと揺れているが、本文中に記す理由により「Language」は「ランゲージ」のままがよいと判断し、本稿では書籍に合わせ「パタン・ランゲージ」と表記することにした。

2 翻訳に若干修正を加え引用している。

3 Kent Beck は当時 Apple 社勤務で、Apple 社フェローとして在籍していた Alan Kay の元で Vivarium プロジェクトに従事していた。その同じ時期に Bill Atkinson が開発していたのが、この HyperCard である。そのため、Ward Cunningham は、実は発売前から HyperCard を知っていた。
http://c2.com/cgi/wiki?WildCard

4 複数の単語において、それぞれの先頭の文字を大文字にして、繋げて一つの単語としたものである。

5 http://downlode.org/wiki/

[参考文献]

[1]
http://www.wikipedia.org/

[2]
Tim O'Reilly: What Is Web 2.0 — Design Patterns and Business Models for the Next Generation of Software, O'Reilly Network, September 2005.
- http://www.oreillynet.com/go/web2
- 翻訳：Tim O'Reilly,「Web 2.0：次世代ソフトウェアのデザインパターンとビジネスモデル」、CNET Japan, 2005。
- http://japan.cnet.com/column/web20/story/0,2000055933,20090039-5,00.htm

[3]
Kent Beck and Ward Cunningham. "Using Pattern Languages for Object-Oriented Programs" Technical Report No. CR-87-43 (Sept. 1987)。
- http://c2.com/doc/oopsla87.html
- 翻訳：Kent Beck、Ward Cunningham、角征典訳、『オブジェクト指向プログラムのためのパターン言語の使用』、Technical Report No. CR-87-43, September 17,1987。
- http://capsctrl.que.jp/kdmsnr/wiki/transl/?UsingPatternLanguagesForOOP

[4]
E. Gamma, R. Helm, R. Johnson, and J. Vlissides. "Design Patterns: Abstraction and Reuse of Object-Oriented Design" In Proceedings of ECOOP'93, Kaiserslautern, Germany, 1993.
- 翻訳：エリック・ガンマ、ラルフ・ジョンソン、リチャード・ヘルム、ジョン・ブリシディース、『オブジェクト指向における再利用のためのデザインパターン（改訂版）』、ソフトバンクパブリッシング、1999。

[5]
Christopher Alexander, et al., "A Pattern Language: Towns, Buildings, Construction" Oxford University Press, 1977.
- 翻訳：クリストファー・アレグザンダー、『パタン・ランゲージ―環境設計の手引』、鹿島出版会、1984。

[6]
Christopher Alexander, "The Timeless Way of Building" Oxford University Press, 1979.
- 翻訳：クリストファー・アレグザンダー、『時を超えた建設の道』、鹿島出版会、1993。

[7]
Christopher Alexander, et al., "The Oregon Experiment." Oxford University Press, 1975.
- 翻訳：クリストファー・アレグザンダー、『オレゴン大学の実験』、鹿島出版会、2000。

[8]
中埜 博、『パタン・ランゲージによる住まいづくり』、井上書院、1988。

[9]
http://c2.com/cgi/wiki?WikiWikiHyperCard

[10]
http://c2.com/cgi/wiki?BezelMenu

[11]
http://c2.com/cgi/wiki?WikiHistory

[12]
http://c2.com/cgi/wiki?InvitationToThePatternsList

[13]
Kent Beck, "Extreme Programming Explained: Embrace Change" Addison Wesley, 1999.
- 翻訳：ケント・ベック、『XP エクストリーム・プログラミング入門―変化を受け入れる』、ピアソンエデュケーション、2005。

[14]
江渡 浩一郎、塚本 牧生、島田 慶樹、『Wiki 概念の多様性』、LinuxConference 2006, 2006。

江渡浩一郎 [えと　こういちろう]
独立行政法人産業技術総合研究所 情報技術研究部門 情報流デザイングループ 研究員
東京藝術大学美術学部 先端芸術表現科 非常勤講師

1971 年生まれ、慶應義塾大学大学院政策・メディア研究科卒業。コンピュータ・アートを専門とし、ネットワーク上のコミュニケーションをテーマとした作品制作を行う。1996 年、sensorium プロジェクトにて「WebHopper」を制作。sensorium は、1997 年、アルス・エレクトロニカ賞グランプリを受賞。1998 年、Canon ARTLAB との共同制作として「SoundCreatures」を制作。アルス・エレクトロニカ賞 Honorary Mention を受賞。2001 年、日本科学未来館「インターネット物理モデル」の制作に参加。産総研では、メーリングリストと WikiWikiWeb を統合したグループ・コミュニケーション・システム「qwikWeb」を開発・運用している。第二回国際 Wiki シンポジウム（WikiSym2006）運営委員。趣味は深追い及びバッド・ノウハウ研究。
http://eto.com/

プロフィールは発行当時のものです。