Wikiの起源と進化

江渡　浩一郎[†,††]

WikiWikiWeb (Wiki) は，1995 年にウォード・カニンガムによって開発された Web 上の情報共有システムである．Wiki は建築家クリストファー・アレグザンダーによるパターンランゲージの概念から考え出されたシステムであり，利用者が設計に参加できるようにするという目的が存在していた．また，同じ出発点を持つ方法論として XP(エクストリーム・プログラミング) があり，両者は目的設定において非常に近い関係にある．本論では，ウォード・カニンガムが Wiki を作り上げた経緯を出発点とし，どのように Wiki が発展していったのかを概観することによって，Wiki が本来持っていた目的とは何かを明かにすることを試みる．

Origin and Evolution of Wiki

KOUICHIROU ETO[†,††]

WikiWikiWeb (Wiki) is an information sharing system on the Web developed by Ward Cunningham in 1995. Wiki is derived from pattern languages proposed by Christopher Alexander, a building architect. The original goal of Wiki is helping users participate in design processes. The XP (Extreme Programming), a methodology for software development, also has similar backgrounds and goals. In this paper, I describe the history of Wiki and try to confirm the original goal of Wiki.

1. はじめに

Wiki という言葉の定義は曖昧である．Wiki という言葉の理解は人によって少しづつ異なっており，同じ言葉を使って議論していても，少しづつその指し示す領域が異なっているように感じられる．

近年，Wiki をめぐる状況は大きく変化し，Wiki という言葉をしばしばメディアで目にするようになった．大きな理由の一つは *Wikipedia*[1] である．インターネット上の誰でも編集できる百科事典 Wikipedia は，本当に様々な情報が蓄積されている．まさにインターネット上の集合知の集積として機能している．Wikipedia は Wiki の代表例として扱われることが多く，時には Wiki を Wikipedia の略称だと考えている人もいるくらいだ．

また，Wiki を扱う企業も多数現れてきた．日本でも Wiki サービスを提供する会社が登場し，Wiki サービスを提供する会社 *JotSpot*[2] は Google に買収された．*WikiSym* (International Symposium on Wikis)[3] のように Wiki を中心としたシンポジウムも開催されるようになってきた．また，Wikipedia を中心としたシンポジウムである *WikiMania*[4] など，様々な Wiki 関連のイベントが開催されるようになってきた．研究の分野にも Wiki が登場するようになってきた．Wiki に独自の拡張を加えたり，Wiki 的な機能を別のメディアで実現するなど，様々な論文が現れてきている．

しかし，このように Wiki をテーマとした活動が広まり，利用が進んできたにもかかわらず，Wiki という言葉にはいまだに曖昧さがつきまとう．Wiki という言葉が指し示すものは人によって少しづつ異なっており，Wiki にとっての本質は何なのかが共通に理解されていないように思える．

本論では，そのような問題意識から，Wiki の元々の起源を振り返り，Wiki がどのような経緯を経て現在にいたるかの歴史をたどる．それにより，Wiki には本来どのような意味があったのか，どこに Wiki の本質があるのかを示すことを試みる．

2. 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．

† 独立行政法人 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology (AIST)
†† 東京大学大学院 情報理工学系研究科 創造情報学専攻
Department of Creative Informatics, Graduate School of Information Science and Technology, The University of Tokyo

Wiki と近い分野の言葉として，**ブログ**がある．ブログは Web 上で日記を書くためのシステムであり，現在では広く使われるようになっている．ブログと Wiki はよく並べられて語られるが，その違いとして，ブログは一人で書くもの，Wiki はみんなで書くものという点があげられる．しかし，例えば自分一人だけが書き込める Wiki を立ち上げ，情報をまとめるという使い方もできる．逆に，一つのブログを違う人が書いていくというブログもあり，つまり一つのブログをみんなで書いていくこともできる．一人で書くか，みんなで書くかは，システムの機能の違いではなく，システムの使い方の違いになる．

システムの機能の違いに着目する方法もある．ブログは主に日付けによって情報を管理するシステムであり，全ての情報は一義的には日付けによって管理される．それに対して Wiki では，一義的にはページ名で情報を管理する．もしページ名として日付けを使うとすれば，Wiki をブログのように使うこともできることになる．逆にブログではページ名のような柔軟な名前付けはできない．このことから，Wiki という概念はブログという概念を包摂しており，Wiki をブログの上位概念として理解している人もいる．しかし，この場合は一体どこまでを包摂しているのかという疑問が生じる．例えば Web 上の掲示板のようなものも Wiki で実現できるのだとすると，Wiki は掲示板の上位概念でもあるということになってしまう．このように考えた場合，Wiki という言葉の指す対象はあまりに幅広くなってしまい，実質的に意味は無くなってしまう．

このように，Wiki という言葉を巡る現状は非常に混乱している．何が Wiki という言葉の根本にあるのかの共通理解が存在していないことがこの混乱の背景にある．そのため，この Wiki という概念がどのような経緯で誕生したのかを明かにすることによって，Wiki という概念の共通理解を得ることを試みる．

3. パターンランゲージ

パターンランゲージ(Pattern Language)☆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．

元々アレグザンダーは，建築において，建築の利用者と実際に建築を設計する建築家との関係に着目していた．建築家は利用者からどのような建築を使いたいかの意見を聞き，その意見を反映させた建築を建てる．しかし実際には利用者が実際に建築が建てられる前にどのような建築がいいかを言うのは困難であり，結果として建築家が提案する建築を利用者がそのまま利用するという状況になりがちである．建築家は自分の美的センスや，自分が実現したいと考える建築を優先し，利用者はただそれを使うだけ，つまり建築家優先で利用者不在の建築がたくさん生まれていた．

そのようなバランスが崩れた状況を解消するために考え出されたのがパターンランゲージである．パターンランゲージは，利用者と設計者が共通で利用できる事例集のようなものであり，互いが同じパターンランゲージを使って対話することによって，共通の土台を作り上げることができる．そのように共同で設計をすることによって，利用者が設計に参加できるようにしたのがパターンランゲージである．つまり，パターンランゲージの根幹には「利用者の設計への参加」がある．

パターンランゲージという言葉は誤解を招きやすい．パターンランゲージという言葉は,「パターン」と「ランゲージ」というどちらも一般的な概念の組み合わせであるため，一見理解しやすそうに見える．しかし，この「パターンランゲージ」はその字面からすぐに理解できる概念だと考えてはならない．

「パターン」という言葉は，通常単に「繰り返しによる模様」や「模倣」といった意味でとられる.「パターン化している」などと使われる場合は否定的な意味になる．しかしパターンランゲージにおける「パターン」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，ある一定の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，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る言語という意味に捉えられるが，そうではない．自然言語は様々な単位の部品の集合から成り立っている．文字という最小単位から始まり，単語，文，段落，節，文章，本，全集，百科事典と様々な大きさの単位がある．それぞれの単位においても，例えば主語，述語，助詞，動詞，形容詞など様々な種類がある．個々の部品間の関係も有機的であり，様々な部品が相互に接続されることによって多種多様な意味が生まれる．そのような，様々な大きさ・種類の部品が集まる

☆ 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，一語でパターンランゲージと表記することにした．

ことによって多様な意味が生まれるような構造全体を指して「ランゲージ」と呼んでいる．

そのようなことから，建築設計において繰り返し出現するような形態を，言語における辞書のような形に組織化して，再利用できるようにまとめたものがパターンランゲージであるということになる．

このパターンランゲージの背景となる理論は，5 冊からなる著作集の一冊目の本「時を超えた建設の道」[5] にまとめられている．その二冊目の本「パタン・ランゲージ」[6] は，パターンを集めたカタログ，一種の辞書のような本である．これらの本は，パターンランゲージを理論的にまとめた本であり，最初は逆にとっつきにくいかもいれない．三冊目の「オレゴン大学の実験」[7] のような具体的な事例集から見ていく方がわかりやすいだろう．また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「パタン・ランゲージによる住まいづくり」[8] は，日本の建築状況においてパターンランゲージを実践した大変わかりやすい記録となっており，参考になる．

例えば「原寸大の設計」という手法が紹介されている．ある場所に家を立てる際に，図面上で配置を考えるが，それだけでは実際にどのような配置になるのかが一般の人にはわかりにくい．そこでその図面を持って現場に行き，地面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．そのように原寸大に広げることによって見えてきた問題点を，その場で修正する．内装も同様に，家具と同じ大きさの段ボールを配置するなどして，どのように見えるかを確認する．このようにして，図面で設計したものを現場で確認しながらその場で設計を直していく手法である．

このように，パターンランゲージは実際には単なる辞書のようなものではなく，このような建築家と利用者をつなぐための様々な工夫の集積として機能するものである．建築はソフトウェアと違って，一度建ててしまったものはそう簡単には変えられない．そのため，建築では設計と施工を簡単には行き来することはできないが，そのような建築という世界においてもなんとか設計と施工を行き来することができるように工夫したものが，パターンランゲージである．

4. パターンランゲージのプログラミングへの応用

1986 年，第一回目の *OOPSLA* (Object-Oriented Programming, Systems, Languages, and Applications)，オブジェクト指向プログラミングに関する学会が開催された．

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た．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という当時新しく出てきた方法論は，ソフトウェア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．ちょうどそのような時期に第一回目の OOPSLA が開催され，現在オブジェクト指向の大家と呼ばれるような開発者も多数参加していた．その発表者の中に，ケント・ベックとウォード・カニンガムの二人がいた．

第二回目にあたる 1987 年の OOPSLA において，ケント・ベックとウォード・カニンガムの二人は「**オブジェクト指向プログラムのためのパターンランゲージの使用**」という論文を発表した[9]．彼らはこの論文で，プログラミングへのパターンランゲージの導入を提唱した．

従来の手続き指向型だったプログラミングにおける設計の方法論から，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・インタフェースの設計に重きを置く必要がある．そのために，建築の分野で提唱されていたパターンランゲージを，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．実際には論文というよりテクニカルペーパーであり，2 ページだけと非常に短い．実践を行ってみての簡単な記録となっている．

ちょうど同じころに GUI が登場してきたことの影響も大きい．ソフトウェア設計の際に，GUI によるインタフェースの設計と，その背後の動作を切り分けて設計を行うようになっていた．インタフェースの設計は，利用者がどのようにそのシステムを利用したいかという意見を取り入れることが重要となる．もちろん，システム設計には技術的な制約があり，その制限内で使い易さを追及する必要がある．

これはちょうど建築における問題設定と同じである．パターンランゲージが建築の分野で行ったのと同じように，利用者と設計者の間の垣根を取り除く実験を，ソフトウェア開発の現場においても行ったということになる．論文より引用する．「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．」「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．」[9]

このように，利用者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利用者自身であるのと同様に，ソフトウェア開発においても利用者がどのようなシステムを利用したいのかを一番よく知っているのは利用者自身であるはずだからだ．パターンランゲージは，設計者と利用者との共通言語として機能し，お互いがこの言語を使うことによって，共通の土台で設計していくことができるようになる．

彼らはまずインタフェースにおけるパターンとして，下記 5 つ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．

インタフェース設計の後に，その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へと分割していく．次に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と順に細分化していく形でパターンを考えていった．

5. HyperCard によるパターン・ブラウザ

論文が公開された 1987 年，Apple 社から *HyperCard* というソフトウェアが発売された．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．HyperCard は，その GUI を最大限に生かしたソフトだった．

HyperCard はカード型データベースソフトと呼ばれる．例えば図書カードのように，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ることがある．そのようなカードの形に情報をまとめたデータベースを簡単に作れるようにしたソフトである．*HyperTalk* というスクリプト言語を備えており，カードによるデータと共に，カードを操作するインタフェースなどの動的な仕組みを埋め込むことができた．そのようにデータとそれを操作するインタフェースを組で配付することができ，それを**スタック**と呼んでいた．

ケント・ベックは当時 Apple 社に勤務しており，アラン・ケイをリーダーとする Vivarium プロジェクトに従事していた．同時期にビル・アトキンソンによって開発が進められていたのが，コードネーム *WildCard*[10] であり，後に HyperCard という商品名で発売されるソフトウェアだった．

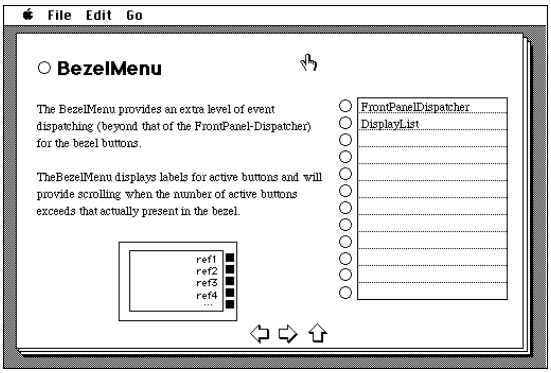

図 1 HyperCard によるパターン・ブラウザの画面

ウォード・カニンガムは，この HyperCard に衝撃を受けた．彼は，HyperCard を使って，自分が発見したプログラミングにおけるパターンを収集・整理するためのパターン・ブラウザを作り始めた[11]．実はこのパターン・ブラウザが発展して出来たのが，今の Wiki なのである[12]．

図 1 がこの HyperCard によるパターン・ブラウザの画面である[13]．カードの左上には「BezelMenu」というタイトルがあり，その下に説明文章，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て，クリックするとそのカードに飛ぶ．メニューから Edit を選ぶことで，WYSIWYG でそのカードを編集できる．

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☆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．

このカードで示される BezelMenu というパターンは，ユーザ・インタフェースにおけるパターンの一種である．bezel とはモニターの縁のことで，つまりモニターの縁に配置されたメニューという意味である．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．このインタフェースは，アメリカの銀行の ATM でよく使われている．

この HyperCard のスタックを複数人で共有する実験も行った．誰かが何かを編集すると，自動的にその編集記録が RecentChanges というカードに残り，リンク

☆ 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．

がはられる．そのため，その RecentChanges を見れば，他人がどのように編集したのかを知ることができる．

このように，HyperCard のスタックとして情報を複数人で共有しつつ，パターンの収集・編集を行っていった．

6. WikiWikiWeb の誕生

1991 年に *WWW* (World Wide Web) が誕生した．1993 年には *NCSA Mosaic* が誕生し，WWW は普及への道を歩みはじめた．それとほぼ同じ時期の 1993 年 12 月に，*CGI* (Common Gateway Interface)[14] という，Web サーバで動的な処理を行うための仕組みを備えた *NCSA httpd* 1.0 が公開された．

1994 年，ウォード・カニンガムは Perl と CGI を使って最初の *WikiWikiWeb* を作り，1995 年 3 月 25 日に彼の Web サイトである *c2.com* で公開した[15]．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[16]．

彼はこの Web サイトに *Portland Pattern Repository* という名前をつけていた．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，Portland Pattern Repository はコンテンツにつけられた名前であり，編集できる仕組みを備えた Web サイトとして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．

このとき始めて，現在 Wiki と言われる手軽にページを編集できる仕組みを備えた Web サイトが出現した．この時取り入れられた機能が，本文中へのリンク埋め込みである．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．

HyperCard によるシステムからの大きな違いは，インターネットに接続されていることである．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，どのようにパターンをまとめていくのか．その収集する方法そのものについてのパターン自体もまたパターンとして記述していくことになった．このようにして，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 スクリプトであり，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[17]．

7. Wiki による文芸的プログラミング

この最初の Wiki の誕生には，興味深いメタな構造が隠されている．

ドナルド・クヌースは，**文芸的プログラミング** (Literate Programming)[18] という概念を提唱した．文芸的プログラミングとは，コードを説明する文章中にプログラムそのものを埋め込むことによって文章とプログラムを一体化できるようにするという概念である．Knuth は実際にこの概念に基いた *WEB* という処理系を作り，その処理系を用いて TEX を作った．

ウォード・カニンガムはこの文芸的プログラミングに影響を受け，Wiki ページ中にプログラムを埋め込み，Wiki とプログラムを一体化できるようにする *HyperPerl* というシステムを作った．まず，ある Wiki サイトの Wiki ページにコードの断片を分散して書いていく．出発点となるページを指定し，そこからページ間接続のリンクを辿るようにして次から次へとページを移っていき，それらのページに埋め込まれているコードを抽出していく．最終的には一つの Perl のスクリプトを生成する．

WikiBase は，実はそのようにして HyperPerl を使って書かれていた．つまり，世界最初の Wiki エンジンである WikiBase は，Wiki による文芸的プログラミング環境 HyperPerl で書かれていた．つまり，Wiki は Wiki を使って書かれていたというメタな関係にあった．

もちろんこれは卵と鶏の関係にある．ちょうど C 言語で C コンパイラを書く手順と同じように，本当に最初の Wiki は手で書いたのだろう．その基盤の上に HyperPerl を構築し，その HyperPerl を用いて WikiBase を再構築したものと考えられる．

この HyperPerl の Wiki サイトには，現在もまだアクセスすることができる[19]．実際に，Wiki ページ一つ一つに WikiBase のコードの断片が分散して書かれており，それらのページを順にたどることによって WikiBase のコードが生成される様子を説明の文章と共に読むことができる．

誰でもコードを編集できるのならセキュリティの面が気になるが，非常にコンパクトなコードの断片が説明文章と共に記述されているため，何かしようとしてもその意図はすぐに見破られてしまい，悪意のあるコードを入れることは難しいだろう．非常に短くシンプルなコードであるために，そのコードのシンプルさ

そのものが，最大の防御になっていたと考えられる．

このように最初の WikiWikiWeb のコードである WikiBase は，説明文章と共に誰でも編集できる形で公開されていたため，同じ仕組みを自分のサイトで動かしたり，自分なりに改造することは難しくなかった．このようにしてウォードの WikiWikiWeb と同じような機能を持つ Web サイトが登場し，それらは Wiki サイトと呼ばれるようになった．また，WikiWikiWeb と同じような機能を持つソフトウェアを Wiki クローンや Wiki エンジンと呼ぶようになっていった．そのようにして，固有名詞としての WikiWikiWeb は徐々に一般名詞としての Wiki へ変化していき，現在では小文字の wiki と表記されるようになっていった．

8. パターンランゲージからデザインパターンへの変遷

プログラミングにおけるパターンランゲージの利用は，その後は**デザインパターン**と呼ばれる大きな流れにつながっていった．デザインパターンとは，プログラミングにおいて繰り返し表われる記述を辞書のように集めたものである．一般には「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」[20] に登場するようなパターンを指している．

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う．例えばイテレータ・パターンは，プログラムに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ある．

初期のパターンランゲージの利用では，パターン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始まっていた．ユーザが認識する世界を出発点とし，実際のコーディングへと細分化していく．そのような一連の流れを含め，パターンランゲージの利用の対象として考えていた．

しかし現在のデザインパターンでは，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において発生するパターンを指す．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，現在ではユーザ・インタフェース・パターンなどといった別の言葉で表されるようになっている．

なぜこのように，最初の論文におけるパターンから現在のデザインパターンにおけるパターンが違うものになってしまったのだろうか．

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ように思われる．当初掲げていた目的は,「ユーザは，自分自身のプログラムを書くべきなのである」ということである．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行的な流れがあったが，それ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．この目的設定が背景にあって，はじめてパターンランゲージの利用という話が出てくる．しかし現在のデザインパターンにはそのような「利用者による設計」といった理念は見えてこない．

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．パター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，パターンランゲージを組み立てていく．そのように，パター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．デザインパターンでは，著者であるエリック・ガンマらが発見したパターンをカタログとしてまとめたものであり，そこに納められたパターンの質が非常に高かったため，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のではないだろうか．

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ていったが，当初のユーザと開発者の間の共通言語という理念は失なわれてしまっているように見える．

9. エクストリーム・プログラミング

パターンランゲージのプログラミングへの応用は，デザインパターンを超え，もう一つの流れとしての *XP* (エクストリーム・プログラミング, Extreme Programming) へとつながる[21]．これは再びケント・ベックとウォード・カニンガムの二人が提唱するソフトウェア開発における方法論である．

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よう，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．ここではまさしくパターンランゲージの主題が繰り返されている．

10. Wiki と XP の理念

ケント・ベックは「エクストリーム・プログラミング」[21] 第二版の第 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，そのような空間設計の能力を取り戻させることだった．

彼はまた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．ケント・ベックは「新たな社会構造を作る機会がある」と語っている．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．

Wiki の本当の目的も，XP と同じようにパターンランゲージの応用から生まれている．Wiki はソフトウェア開発とは直接の関係が無い分野においても使われるようになってきているが，利用者が同時に設計者でもあるというコンセプトはそこでも生き残っている．建築の世界では受け入れられなかった設計者と利用者の間のバランスをとるという試みが，この Wiki の世界では受け入れられているように見える．

このことから，「**パターンランゲージの概念を応用した Web 上のシステム**」というのが Wiki の本質であると考えられる．同じようにパターンランゲージから発展した概念として，XP があり，両者は同じ出発点を持っているという意味で，兄弟のような関係にあると言えるだろう．

11. おわりに

本稿では，ウォード・カニンガムが Wiki を作り上げた経緯を出発点とし，どのように Wiki が発展していったのかを概観した．また，同じ出発点を持つ方法論として XP に着目し，両者は目的設定において非常に近い関係にあることがわかった．Wiki は単なる Web 上のシステムではなく，利用者と設計者のバランスをとることを目的とした概念であり，その意味で単なる機能の集合には還元できないことがわかった．今後はこのような Wiki の特徴を，より明確にし，様々な領域へと広げることを目指していきたい．

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発展し生まれた[22]．また，未来心理編集長の萩原徹太郎氏には，本論文の前身にあたる「なぜそんなにも Wiki は重要なのか」[23] の執筆にあたり，多大なる協力を頂いた．ここに感謝の意を表する．

参 考 文 献

1) Wikipedia: http://www.wikipedia.org/
2) JotSpot: http://www.jot.com/
3) WikiSym: http://www.wikisym.org/
4) Wikimania: http://wikimania2007.wikimedia.org/
5) Alexander, C.: *The Timeless Way of Building*, Harvard University Press (1979).
6) Christopher Alexander, e. a.: *A Pattern Language:*

Towns, Buildings, Construction, Oxford University Press (1977).
7) Christopher Alexander, e. a.: *The Oregon Experiment*, Harvard University Press (1975).
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) Beck, K. and Cunningham, W.: Using Pattern Languages for Object-Oriented Programs, *Technical Report No. CR-87-43* (1987).
10) http://c2.com/cgi/wiki?WildCard
11) http://c2.com/doc/etymology.html
12) http://c2.com/cgi/wiki?WikiWikiHyperCard
13) http://c2.com/cgi/wiki?BezelMenu
14) McCool, R.: The Common Gateway Interface (1993). http://hoohoo.ncsa.uiuc.edu/cgi/
15) http://c2.com/cgi/wiki?WikiHistory
16) http://c2.com/cgi/wiki?InvitationToThePatternsList
17) WikiBase: http://downlode.org/wiki/wiki
18) Knuth, D.E.: *Literate Programming*, Center for the Study of Language and Information (1992).
19) http://c2.com/w4/wikibase/wiki.cgi?HyperPerl.
20) Gamma, E., Helm, R., Johnson, R. and Vlissides, J.: Design Patterns: Abstraction and Reuse of Object-Oriented Design, *Proceedings of ECOOP'93* (1993).
21) Beck, K.: *Extreme Programming Explained: Embrace Change*, Addison Wesley (1999).
22) 江渡浩一郎，塚本牧生，島田慶樹：Wiki 概念の多様性，*Proceedings of LinuxConference 2006* (2006).
23) 江渡浩一郎：なぜそんなにも Wiki は重要なのか，Mobile Society Review 未来心理 Vol.7，モバイル社会研究所，pp.50–57 (2006).