M-015

スマートフォンでの利用に特化した Wiki システムの開発

Development of the Wiki system for smartphones

佐々木佳祐* 丸山 一貴† 寺田 実*
Keisuke Sasaki Kazutaka Maruyama Minoru Terada

1 背景

Wiki は, Web ブラウザから誰でも利用できる, 強力なコンテンツマネジメントシステムである. その特徴から, 個人単位での情報整理や, グループによる情報のまとめなど, 幅広い用途で利用されている [4]. また近年爆発的に普及し始めているスマートフォンは, その携帯性と, アプリケーションの豊富さから, 思い立った時にすぐメモを残すなど, 情報整理デバイスとしての価値も高い. スマートフォンには PC で利用できるものとほぼ同じ機能を備えた Web ブラウザが標準搭載されているため, モバイル環境で Wiki を利用するといったケースも今後増加していくのではないかと考えられる. また, スマートフォンにはカメラや GPS 機能を備えたものが多く, Wiki の利用法として写真や位置情報を付加した情報共有, といったシーンも想定される.

2 目的

スマートフォンの Web ブラウザで Wiki を利用しようとすると, PC 環境との差異から以下のような問題が発生する.

- 画面解像度やアスペクト比の差異による操作感の差
- 入力デバイスの差による記号入力の手間
- インターネットに接続されていない時に編集できない

本研究では, 以上の問題を解決し, スマートフォンから快適に利用できるインタフェースを持つスマートフォン向けの Wiki システムの開発を行い, モバイル環境における Wiki を用いた情報共有を支援することを目的とする.

3 想定される利用シーン

■個人での利用 Wiki は箇条書きや段落分けなどが非常に簡単な記述で表現でき, 情報をまとめておくことに関して非常に便利なツールである. 身近な端末となるスマートフォン上に, インターネット回線を必要としない Wiki があることで, より手軽かつ高速に情報を整理することが可能になる.

■複数人での利用 一般的に, Wiki は複数人によって編集され, 情報を蓄積していく. 本システムも複数人による運用を想定しており, 観光地や飲食店の情報の共有など, 位置情報やカメラを用いた写真といった, 文字以外の情報を付加することで情報が充実する場面において効果を期待できると思われる.

* 電気通信大学 The University of Electro-Communications
† 東京大学 情報基盤センター

4 関連研究

■Wiki システムによる情報共有の事例 Wiki を用いた情報共有の事例として,

- Wiki 型システムによる研究室内情報共有の試み [1]
- 学生の情報共有・交換方法としての Wiki の効果 [2]

などの研究が行われている. それぞれ研究室内の情報を共有するために導入し, 従来の手法 (掲示板やメーリングリストなど) と比較して何が発生したかなどを調査している. 以上の 2 つの研究以外にも Wiki を情報共有に活用した事例は存在するが, 携帯端末を用いた Wiki システムの利用事例を発見することはできなかった.

■携帯アプリによる観光地情報収集システム [3] ユーザに実際に観光地に赴いてもらい, 携帯電話のアプリケーションを用いてゲーム感覚で各地の情報を入力させ, 観光地の情報を収集するシステムである. このシステムは, 観光地のリストが予め与えられ, 実際に観光地に近づくとクイズに回答する権利が取得でき, 更にクイズに正答すれば宝物がもらえる, というスタンプラリー形式のアプリケーションである. また, ユーザが新たに観光地を作成することもできる. この研究によって, 昨今の携帯端末は何らかの”場所”についての情報共有に対して有用であることが示されている.

5 提示システム

5.1 概要

本研究では, 図 1 のように PC の Web ブラウザと, スマートフォンにインストールされたアプリケーションの双方から編集可能な一つの Wiki システムを構築する. なお図中の矢印はデータの入出力関係を表す. スマートフォンおよび PC の Web ブラウザからサーバ上の Wiki を参照する際は, どの端末からも同じ内容が閲覧できる. さらに, 各スマートフォン上のアプリケーションではネットワークに接続されていなくとも Wiki が編集可能であり, ネットワークに接続して同期処理を行うことで, Web サーバ上の Wiki と内容を揃えることができる.

5.2 用語の定義

5.2.1 記事

Wiki が持つ一つ一つのデータを記事と定義する. 記事は,

- タイトル
- 本文
- 画像
- 記事を作成した時点での緯度, 経度
- 記事の作成時刻および更新時刻

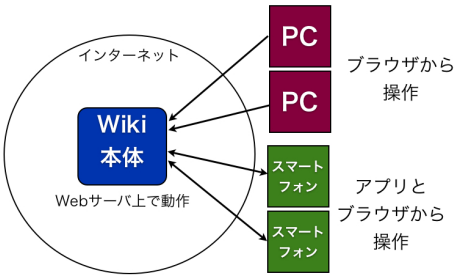


図 1　システム概要図

図 2　一般の Wiki のレイアウト例

などのデータを持つ.

5.2.2　モード

Wiki は一般的に, 記事の本文を表示する画面と, 本文を編集する画面を行き来して利用される. 本論文では, アプリケーションが記事の本文を表示している状態を閲覧モード, 編集を行う画面を表示している状態を編集モードと定義する.

5.2.3　クライアント

本システム専用のスマートフォン用 Wiki アプリケーションがインストールされた端末をクライアントと定義する. 各クライアント内のアプリケーションは同期機能を利用しない限りはネットワーク通信を必要とせず, 単独で動作する. よって, クライアントはユーザにとって, オフラインの携帯用 Wiki となる.

5.2.4　サーバ

本研究では,Web 上で動作している Wiki システムをサーバと定義する. サーバは PC やスマートフォンの Web ブラウザからいつでも参照できる. クライアントはサーバと同期処理を行うことで, 他のクライアントから投稿された記事を参照できる.

5.3　スマートフォン用アプリケーション

第 1 章でも述べたように, スマートフォンは PC と比べ画面の大きさや入力デバイスが異なるため, それに合わせたインタフェースを設計する必要がある.

5.3.1　記事の一覧表示

通常, Web 上の Wiki のレイアウトはそのページの左右どちらかに記事の一覧やメニューが配置され, 余ったスペースに本文が記述されているというパターンが用いられることが多い (例：PukiWiki[*1]図 2) が, このレイアウトでは縦に長いスマートフォンの画面では操作しにくいので, 起動時には記事の一覧が表示されるように設計した (図 4(a)). 内容を参照したい記事のタイトルをタップすると, 本文が表示される (閲覧モードに移行する). ただし, 記事が存在しない場合はその旨を示すメッセージが表示される.

5.3.2　記事の編集

■Wiki 記法　Wiki システムには, Wiki 記法と呼ばれるマークアップ記法が存在する. (例：図 3 のように, 文頭に “*” を付加することで見出しが自動で成形される等) 現在, 本システムが対応している記法を表 1 に示す.

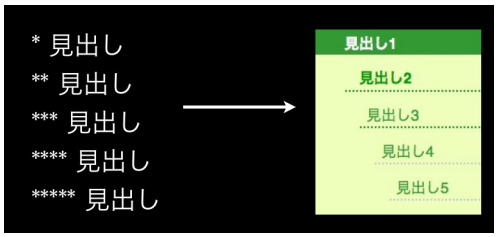


図 3　Wiki 記法での “見出し” の記法

表 1　対応済みの Wiki 記法

記法	意味
! 文字列	見出し (3 段階まで対応)
* 文字列	箇条書き (3 段階まで対応)
[[文字列]]	ページへのリンク
[[URL]]	URL へのリンク

5.3.3　編集画面

図 4(b) が記事を編集する画面である. 画面上部のバーから, ワンタッチで記号を入力することが可能である. また下部のメニューボタンから, 写真の撮影や選択, 位置情報の取得などの機能が利用できる.

5.3.4　記事の閲覧

前節で編集した図 4(b) の記事を, 閲覧用の画面で表示すると, 図 4(c) のように文章が整形される. 画面上部のバーにはタイトル, 編集モードへ移行するためのボタンが配置されている.

5.3.5　画像の添付

カメラまたは端末のストレージから画像を取得, 挿入することができる. 挿入後は左上にプレビューが表示される. 現状では, 挿入できる画像は 1 枚のみで, 画像の拡大表示にも対応していない.

5.3.6　位置情報

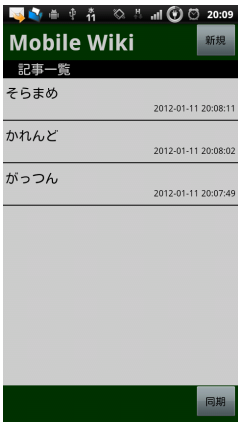
スマートフォンの GPS 機能を用いて, 記事を作成した位置の情報を保存, 閲覧することができる. 位置情報の閲覧には, スマートフォンに標準搭載されている Google Maps[*2]を利用する.

5.4　サーバ用の Wiki システム

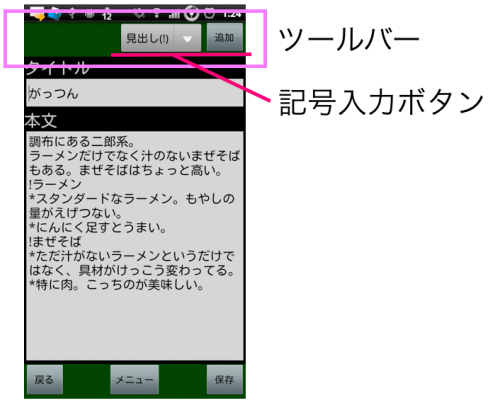
サーバとなる Wiki システムは, 通常の Wiki としてブラウザで利用可能であり, かつクライアントとの通信用 API

[*1] http://pukiwiki.sourceforge.jp/

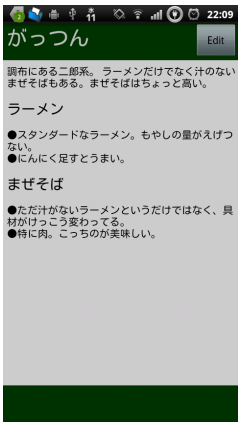
[*2] https://market.android.com/details?id=com.google.android.apps.maps



(a) 記事の一覧



(b) 編集画面



(c) 記事の閲覧

図 4　記事の編集・閲覧

を備える．

5.5　同期機能

Wiki は誰でも同じ記事を編集できることが一つの利点である．よって Wiki システムを名乗る以上, 個人の端末で作成された記事を, 他の端末からも参照, 編集可能にしておく必要がある．複数端末によるデータの共有方式としては, P2P やクライアント・サーバ方式が考えられるが, 本研究では各端末の所持者がどのように Wiki を編集しているかの管理のしやすさから後者を採用し, サーバとして Web 上で動作する Wiki を各クライアントから編集するようなイメージで同期機能を実装する．

■アルゴリズム　同期のアルゴリズムを以下に示す．

1. 端末が, サーバから全記事のデータをダウンロードする
2. 同名の記事が存在しない場合, クライアントに新規に記事を作成, 保存する
3. 同名の記事が存在する場合は, クライアントにある記事よりもサーバにある記事の編集日時が新しい場合に, サーバの内容をクライアントに保存する
4. クライアント上に存在する, ダウンロード時に更新が行われなかった記事データをサーバに送信する
5. データを受け取ったサーバは, 同名の記事が存在する場合上書き保存, しない場合は新規に記事を作成, 保存する

以上の手順によって同期が行われる．ここで, 2 人のユーザが同時に同名の記事を同期した場合, 現状ではリクエストが後に受理されたクライアントの編集だけがサーバに残る．PukiWiki 等の Wiki システムには, 編集中に誰かが更新した記事を保存する際, 互いの変更箇所を変更して自動で補完する機能を備えているものもあるので, 今後, そのような例を参考に衝突の問題を回避する必要がある．

6　実装

6.1　開発環境

クライアントの対象 OS は Android とし, その上で動作するアプリケーションを作成した．なお Android アプリケーションの記述言語は Java である．サーバ用の Wiki アプリケーションは Web アプリケーションフレームワークである Ruby on Rails[*3]を用いて作成した．また記事データベースにはクライアント, サーバともに SQLite[*4]を用いた．Android は標準で SQLite をサポートしており, またサーバに届くリクエストもさほど大きいものでないため, 軽量な DBMS である SQLite で問題ないであろうという判断からの選択である．

6.2　データ構造

サーバおよびクライアントでは, 1 つ 1 つの記事は表 2 のようなデータ構造を持つ記事クラスとして扱う．

表 2　記事のデータ構造

名称	データ型	詳細
title	文字列	記事のタイトル
body	文字列	記事本文
category	文字列	記事のカテゴリ (未使用)
photo	数値	写真の有無を判定する値 (1 以上で写真あり, 0 でなし)
latitude	文字列	緯度 (度)
longitude	文字列	経度 (度)
created	文字列	作成日時 (yyyy-MM-ddTkk:mm:ss) (y:年, M:月, d:日, k:24 時間中の時, m:分, s:秒 T は ISO8601 で定められた, 日付と時刻の組み合わせる際に用いる記号)
updated	文字列	更新日時 (形式は同上)

なお, 記事のタイトルは重複しないものとする．仮に重複したタイトルを入力した場合, 既に保存されている記事を

*3 http://rubyonrails.org/
*4 http://www.sqlite.org/

編集する画面に遷移することで, タイトルの重複を検出することが可能となっている.

6.3 スマートフォン用 Wiki アプリケーション

6.3.1 アプリケーションの構成

■画面の表現　Android のアプリケーションは, 1 つの画面に対して

- レイアウトファイル (XML で記述)
- アクティビティファイル (java で記述)

をそれぞれ 1 つずつ持つ. レイアウトファイルでは画面を構成するためのボタンやテキストなどのパーツの配置を, アクティビティファイルではそれらのパーツに対するユーザからの操作 (タップや長押しなど) にどう対応するかを記述する. アクティビティファイルには基本的に Android が提供する Activity クラスを継承した xxxActivity クラスが記述されている. 以降では, アクティビティファイルとレイアウトファイルからなる画面 1 枚をアクティビティと呼ぶ.

■画面の切り替え　アクティビティの切り替えは, インテントという機能を用いて行う. インテントとは, 遷移先のアクティビティと, 遷移先に渡したい情報をインテントに付加してシステムに送信すると, システムが宛先を判断してそのアクティビティやアプリケーションを呼び出してくれるという仕組みである.

■記事一覧表示　一覧表示のためのアクティビティは, 項目をリスト化して表示するコンポーネントである ListView を手軽に扱える ListActivity というクラスを継承して作成した. アクティビティが呼び出されると, データベースから全ての記事データを取得し, 更新日時の順にソートして表示する. 項目をタップすると, そのタイトル, 本文などのデータをインテントに付加し, 閲覧用のアクティビティに送信する.

■記事閲覧　閲覧モードのためのアクティビティは, タイトルと本文を表示する領域, また位置情報が付加されている記事の場合はそれを表示するためのボタンを持つ. インテントからタイトルと本文, 写真の有無と位置情報を受け取り, それぞれ以下のように処理する.

タイトル　タイトルはそのまま, 上部のバーに表示される.

本文　後述するパーサによって Wiki 記法を Html タグに変換し, 表示する.

写真の有無　添付されている写真がある場合, 記事と紐付けられた画像ファイルを SD カードなどのストレージから読み込み, 表示する. 記事と画像ファイルとの紐付けは, 画像のファイル名を「記事の作成日時.jpg」とすることで実現する.

位置情報　位置情報が空でない (何らかの値が入っている) 場合に, 画面右下にボタンを表示する. ボタンが押下されると, 位置情報をインテントを用いて Google Maps に送信する.

■Wiki 記法のパーサ　パーサは Parser クラスとして作成した. Parser クラスのメソッド parse() に本文を渡すと, Wiki 記法の本文を Html の形式で出力する. 具体的には, 以下の手順で変換を行う.

1. 本文を 1 行ずつ読み込む
2. 行内にリンクを示す記号の左端 ([[) があれば, 中括弧で括られた部分を Html の A タグ () で書き換える
3. 行頭が'!' または'*' である時, その記号の個数を数え, 対応するタグおよび記号に書き換える
4. 書き換えた行の末尾に改行タグ (
) を加え, バッファに格納する
5. 次の行を読み込み, 変換の結果をバッファに追記していく
6. 全ての行の読み込みが終了したら, バッファの内容を文字列データとして出力する.

3. において, 対応する記法とタグの関係を表 3 に示す. ここで, 箇条書きにタグ (
) ではなく記号を利用しているのは, Android のテキスト表示部品である TextView が, 箇条書きのタグに対応していないためである.

表 3　Wiki 記法と Html タグの対応関係

Wiki 記法	Html タグ
! 文字列	文字列
!! 文字列	文字列
!!! 文字列	文字列
* 文字列	●文字列
** 文字列	(半角スペース 1 つ開けて) ○文字列
*** 文字列	(半角スペース 2 つ開けて) ■文字列

■記事の編集　記事の編集用アクティビティは, タイトルや本文を入力するためのテキストボックス, また保存やメニューを呼び出すためのボタンを持つ. 重複タイトルの確認は, テキストボックス内が変更されるたびに行われる. テキストボックスには内容が変更されたイベントを検知し, 処理を行うリスナを設定することができるので, そのリスナに「テキストボックス内の文字列がデータベースのタイトルと一致するか」という処理を加えて実装した. また, 後述する画像挿入のために, 新規作成の画面が呼び出された時点で, その時の日付データを created_at として保持しておく.

■記号入力　追加ボタンがタップされると, 本文のカーソルの位置に選択された記号が挿入される. この処理は, テキストボックス内の本文をカーソル位置で分割し, 前半と後半の間に記号を加えた新しい文字列をテキストボックスに戻すことで実装している. また, リンクを挿入する場合は, リンク先の URL を入力するダイアログが表示される.

■画像の挿入　メニューから「写真を撮る」または「写真を選択」を選択するとと, それぞれの機能を持つアプリケーション (カメラやギャラリー) を呼ぶためのインテントが発行される. 取得した画像は, 記事と画像との紐付けのため, 「(編集中の記事の created_at).jpg」というファイル名で保存する.

■位置情報の取得　メニューから「位置情報の取得」を選択すると, 端末の存在する緯度と経度を取得する. 位置情報を取得する手段は

- Wi-Fi

- GPS
- 3G ネットワーク

の 3 通り存在し, 上にいくほど精度が高くなる. クライアント側で Wi-Fi や GPS 機能をオフにすることも可能なので, アプリケーションではどれを利用するかというのは固定せず, クライアントが利用可能なネットワークのうち最も高精度なものを選択するように実装した.

6.4 サーバ用 Wiki システム

サーバ用システムは, 記事のデータベースとそれの読み書きが最低限可能であればよいので, Rails に標準搭載されている scaffolding 機能を利用して実装した. scaffolding を利用すると, データベースへのレコードの挿入やデータの編集などができる Web アプリケーションが自動で生成される. この自動で生成されたコードを編集し, アプリケーションからの同期に対応する API を作成した.

■API 設計 API は, 主に HTTP メソッド[*5]の GET と POST を用いる.

GET メソッドは, Web サーバからデータを取得する際に用いられるメソッドで, ブラウザが Web ページの URL から html を取得する, などといった場面によく用いられる.

POST メソッドは, GET とは逆に Web 上にデータを送信する際に用いられるメソッドである. HTTP メソッドはこの他にも PUT, DELETE などがあるが, 本システムでは以上の 2 つを用いる.

記事一覧の取得 GET メソッドで, 全ての記事を JSON[*6]形式で取得する. 同タイトルの, タイムスタンプが端末上の記事よりも新しいものがあるか, 端末上に存在しないタイトルの記事が一覧にある場合に, それを端末に保存する.

記事データの送信 GET メソッドによりサーバに存在する記事と端末上に存在する記事の差分ができるので, その中でサーバに存在しないものを JSON 形式に変換し, POST メソッドでアップロードする.

6.5 同期機能

クライアントとサーバのデータのやり取りの際には, 記事データを JSON 形式に変換して行う. XML や他のデータ形式も選択肢にあったが, エンコードやデコードが容易で, 余計なタグなどがなく軽量であることから JSON を選択した. アプリケーションでは同期処理をダウンロードとアップロードの 2 つのタスクに分割して実装した.

■ダウンロード 前述の API にアクセスし, サーバ上の記事データを JSON の配列という形式で受け取る. JSON を変換すると, 記事クラスの配列となる. この配列を 1 つずつ, クライアント内の記事と比較し, クライアントに保存するかどうかを判断する.

■アップロード アップロードは,JSON 形式に書き換えられた記事データを 1 つずつ送信する. 送信時も受信時と同様, 送信する記事一覧を JSON の配列にすることも可能であったが,

- リクエストが簡潔になる
- サーバ側の実装が容易

などの理由から, 1 つずつ送信するという実装にした.

7 評価実験

本学の学生, また Twitter[*7]経由で募集したスマートフォン利用者計 7 名に対し 2 つの実験を行った. 実験は 2 つ行い, それぞれ

- 同じ文章を, 作成したアプリケーションと, 被験者が日常で利用しているメモ用アプリケーションとで作成し, 入力にかかる時間を計測
- 作成したシステムを用いて, 被験者同士で東京都周辺の飲食店情報を共有し, 使用感などについてアンケートに回答

という内容である.

7.1 結果

7.1.1 実験 1

3 人の被験者の, 文章を入力するまでにかかった時間を表 4 に示す.

表 4 実験 1 の計測結果

	本システム	メモ帳	短縮時間
被験者 1	4 分 54 秒	5 分 51 秒	57 秒
被験者 2	4 分 06 秒	4 分 16 秒	10 秒
被験者 3	4 分 26 秒	3 分 50 秒	-36 秒

7.1.2 実験 2

アンケートで得られたアプリケーションおよびシステムに関する主な意見を以下に示す.

- 良い点
  - UI がシンプルで良い
  - 記号入力が簡単で幸せを感じた
- 悪い点・意見
  - 操作中に上下のバーが隠れることがある
  - 記事の検索やカテゴライズ機能が欲しい
  - 記事ごとに同期, または同期する記事を選択したい

また, 現在のサーバ仕様では写真の共有ができないものの, 本実験におけるシナリオにおいては, 写真を添付することによって情報量が増加し, 更に効率的な情報共有が期待できるはず, との意見も賜った.

7.2 考察

実験 1 の結果より, 編集時の画面上部に配置したツールバーは, 実験 2 でも得られたコメントの通り, 文章が長くなると上部のバーが隠れてしまう現象によって, 逆に入力に時間がかかるというケースも見られたので, バーを固定するなどの処置を施せば, 非常に効果的な UI 部品となりうると考えられる. 実験 2 の結果, Wiki を PC で利用した経験の有無に関わらず, 本アプリケーションは Wiki として情報を取りまとめるのに効果を発揮できるであろうという結果が得られた. しかし実験期間が短かったため, 被験者同士に

[*5] http://www.studyinghttp.net/rfc_ja/rfc2616
[*6] http://www.json.org/
[*7] http://twitter.com

よる相互な記事編集が想定したよりも活発に発生しなかったのが残念な結果となった．また, シンプルな UI がわかりやすいという意見が多いため, 機能追加の際にはシンプルさを保ちつつ適切な UI を設計する必要があると考えられる．

8 結論

8.1 結論

評価実験の結果, 設計した UI が, スマートフォンにおける Wiki の編集に有効である可能性は立証できた．また, モバイル環境における情報共有に Wiki が有効かどうかは, 判断材料に欠けるという結果となった．

8.2 今後の課題

■アプリケーションの機能の改善　スマートフォンの画面のサイズでは, PC と比べ表示できる情報の量が少ないため, 記事数が増加した際に所望の記事に辿り着くまでに時間を要する．なので, 記事を検索できるような機能が必要になると考えられる．

■同期機能の改善　現状では, アプリケーションとサーバにある記事の全てを対象に同期が行われる．記事の数が増加すると処理に時間がかかる上, 編集されていない古い記事について処理を行うのはリソースの無駄である．また, 自分が編集した記事だけをアップロードしたいという意見もあったので,

- 選択した記事だけをアップロード
- 前回の同期の日付よりも更新日時が新しいものだけど同期の対象とする

という 2 つの同期モードを追加すべきであると考えられる．また, 同期の際の衝突の問題も解決すべき課題として挙げられる．

■運用実験　今回の実験では, 思いの外被験者同士の記事の編集が発生せず, モバイル環境における Wiki の有効性を判断するには材料不足となる結果となったので, もう少し長期間の運用実験を行い, 改めて有効性を判断すべきであると考えられる．

また, 新たなシナリオ (観光地案内など) を用意した実験も行い, 想定したシーンにおいて本システムが有効であることの立証を確かなものとしたい．

参考文献

[1] 伊藤 久祥:電子情報通信学会技術研究報告. ET, 教育工学 103(226), 13-18, 2003.
[2] 村木 翔, 美馬 義亮:情報処理学会研究報告, コンピュータと教育研究会報告 2008(128), pp. 69-74.
[3] 市村 匠, 鎌田 真, 目良 和也, 新美 礼彦, 第 15 回日本知能情報ファジィ学会中国・四国支部大会講演論文集 (2010) pp. 5-8.
[4] 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.