Vol.2010-GN-76 No.18
2010/5/21

# 開発者の状況を考慮した開発者知識共有環境の構築とアジャイル開発への適用

山中啓祐[†1]　大西雅宏[†1]　高田秀志[†2]

システム開発に導入されている知識共有のためのシステムは，開発作業以外に行わなければならない作業が多いために，継続して利用されないことが問題として指摘されている．本稿では，このようなシステムが継続して利用されるようにするために，開発者にとって負担の少ない知識共有環境について述べ，また，開発者の行動から，開発者の作業内容に関連のある知識を自動的に抽出する手法を提案する．本手法では，開発者の作業状況を考慮した協調フィルタリングを行うことによって，開発者の作業内容に関連のある知識を抽出する．また，提案する環境を用いることによって，仕様や設計の変更に対して迅速な対応が必要となるアジャイル開発においても，効率的な知識共有が可能になる．

# A Developer Knowledge Sharing Environment Considering Developer's Context and Its Application to Agile Development

Keisuke Yamanaka,[†1] Masahiro Ohnishi[†1]
and Hideyuki Takada[†2]

Knowledge sharing systems that have been introduced into system development can not be utilized continuously because such systems require developers to perform other tasks besides development work. This paper describes a knowledge sharing environment that requires a few developer's task and proposes a knowledge extraction method that can extract knowledge related to developer's tasks by utilizing the information from developer's activities. This method enables knowledge extraction related to developer's task by applying collaborative filtering in consideration with the developer's context. Furthermore, this environment enables developers to effectively share knowledge on Agile Development where the developers need to work on specification changes or design changes quickly.

## 1. はじめに

近年の企業活動において，組織内での知識共有は重要な要素の 1 つであり[1)2)3)]，ナレッジマネジメントシステムのような企業内で知識を共有するための仕組みが積極的に導入されている[4)]．

しかし，システム開発を行う現場では，知識共有のための仕組みが継続して利用されず，十分に機能しなくなる場合がある[5)6)]．これは，開発者が知識共有のためのシステムを利用するときに，知識の登録や検索など，開発作業とは別の作業が開発者にとって負担になってしまうことが原因にあげられている．特に，後の開発のために成果物を残すことよりも，製品の生成に重きをおくアジャイル開発において，知識共有のために知識の登録が必要なシステムを利用することは，開発者にとって負担となる．

したがって，知識共有のためのシステムが継続して利用されるためには，知識共有に必要な開発者の負担を減少させることが重要となる．そこで本稿では，開発者にとって負担の少ない知識共有環境の構築と，その環境のアジャイル開発への適用について述べる．知識共有環境の構築では，開発者間での知識の共有を「知識の収集と組織化」と「知識の抽出」という 2 つのプロセスのからなると捉え，それぞれのプロセスの自動化を行う．知識の収集と組織化のプロセスでは，開発者の行動をもとに開発作業で得られる成果物や背景知識の関係付けを行い，開発者知識として情報を蓄積する．また，知識の抽出のプロセスでは，開発者が開発作業を行うときに過去に作成された成果物を参照する行動に着目し，参照行動の履歴から，開発者の作業内容に関連のある開発者知識を自動的に抽出する．これらの手法により，開発者に特別な操作を要求することなく，開発者間での知識共有を実現することが可能となる．アジャイル開発への適用では，アジャイル開発の代表的な開発手法であるエクストリームプログラミングの実践の一つの計画ゲームで用いられるカードを，開発者知識ネットワークで構造化し，知識の提供を行う．カードを構造化することによって，顧客の要求の変更が影響を及ぼすカードを抽出可能にする．

†1 立命館大学大学院理工学研究科
Graduate School of Science and Engineering, Ritsumeikan University
†2 立命館大学情報理工学部
Department of Information Science and Engineering, Ritsumeikan University

## 2. 知識共有の現状と課題

### 2.1 知識共有に関する既存研究

知識共有に関する研究は，どのような情報を共有する対象とするか，どのような意味や価値を共有する知識に付加していくのか，といった「対象となる情報」に関する研究と，どのように知識の収集や抽出を行うのか，といった「知識の共有方法」に関する研究の 2 つに分類することができる．

#### 2.1.1 対象となる情報に関する研究

システム開発において共有する対象となる情報に関する研究として，近年では，開発中に生成される成果物に意味や価値を付加するためのさまざまな手法が提案されている．これらの手法には，プロジェクトの開発プロセスなどの形式化している情報を成果物に付加して利用するものと，設計書やソースコードなどを作成するときの背景知識やノウハウなどの暗黙的な情報を形式化し，成果物に付加して利用するものに分けられる．

開発のプロセスや，成果物が作られたフェーズなどの形式化している情報を成果物の情報に付加して利用する手法のひとつに，成果物間の関係に意味を付加する手法[7] がある．この手法では,「ナレッジリンク」と呼ばれる成果物間の関係を定義している．ナレッジリンクは，成果物間の導出の関係や,「参考にする」,「補足する」などの関係をクラスとして分類し，ある成果物が別の成果物とどのような関係であるのかを表す．

ノウハウや背景知識などの暗黙的な情報を形式化して利用する研究としては，背景知識やノウハウを文書に付加する手法[8] が提案されている．この手法では，電子的なテキスト形式のメモを利用者に提供し，それに文書が作成されたときの背景知識やノウハウを利用者に記述してもらい,「知識メモ」として文書と合わせて管理する．知識メモを共有することによって，文書だけではなく，文書を作成する場合に用いた知識を共有することが可能になる．

#### 2.1.2 知識の共有方法に関する研究

知識の共有方法に関する研究として，システムの利用者にとって負担が少なく，効率のよい知識の収集と抽出ができるような手法が提案されている．

知識の収集方法に関する研究として，利用者が業務プロセスを実行することによって成果物間の関係付けが自動的に行われる手法[7] が提案されている．業務プロセスの内容は，タスクとアクションとしてあらかじめ登録されており，開発者がアクション内で成果物を生成し，成果物をシステムに登録するときに，アクションの情報をもとにして登録した成果物と他の成果物の間の関係が自動的に付加される．

知識の抽出方法に関する研究では，文書体系に基づく横断検索を実現する手法[9] が提案されている．この手法では,「ドキュメントの種類」や「作業内容の種類」などの 5 つの属性を定義している．この属性は，システムの管理者によって文書に付加される．利用者は，検索条件の属性を自由に切り替えることが可能であり，その属性に関連した情報へ柔軟にアクセスすることができるようになる．

### 2.2 知識共有における問題

知識共有のための仕組みには，実際に導入されても継続して利用されず，十分に機能しなくなるという問題がある．

その原因の一つとして，知識共有は開発者にとって負担になることがあげられている[5]．知識共有を行う場合，前節でも述べたように，利用者はシステムに知識の登録や検索をするために，本来の開発作業とは別の作業を行う必要がある．この知識の登録や検索にかかる時間は，開発者にとって負担になってしまう．また，システムを利用しても，利用者が今直面している課題に対して有効な知識が得られない場合がある．このようなことから，開発者が知識共有システムに対して，開発者が否定的な意識を持っていることも原因の一つとしてあげられている[6]．

ドキュメントの管理よりも，顧客の要求の変更に柔軟に対応することに重きを置くアジャイル開発では，知識共有のための特別な作業が必要とされると，迅速に対応することが難しくなる．そのため，知識共有のためのシステムの導入には向かないと考えられる．

しかし，開発者に対して行った調査の結果，多くの開発者から知識共有で扱われる情報は有益である，という回答が得られている[6]．

そのため，システムを利用するときの負担を下げ，利用者の状況に適切な情報を抽出できれば，開発者のシステムの継続した利用やアジャイル開発における知識共有のためのシステムの導入が可能になると考えられる．

## 3. 開発者の状況を考慮した開発者知識共有環境

本章では，開発者にとって負担の少ない知識共有を実現するために，知識と開発者の行動の収集・組織化を自動的に行う知識共有環境と，開発者の行動に合わせて知識を推薦する手法について述べる．本手法で扱う知識は，開発者の作業によって作成された成果物に対して，開発作業から得られる情報を付加したものであり，これを「開発者知識」として定義する．

### 3.1 開発者知識ネットワーク

開発者知識ネットワークは，開発者知識を，要素と，要素間の関係のネットワークによっ

Vol.2010-GN-76 No.18
2010/5/21

て表現する．開発者知識ネットワークで定義されている要素と要素間の関係について，以下で述べる．

### 3.1.1 開発者知識ネットワークの要素

開発者知識ネットワークでは,「成果物」,「構成物」,「開発者」，および「注釈」の 4 つの要素が定義されている．

成果物は，仕様書やソースコードなど，開発作業で作成される文書を表す．また，構成物は，成果物を論理的構造に基づいて分割したものである．開発者は，成果物の参照や編集を行う人を表す．注釈は，成果物に対する背景知識を記述するためのものである．背景知識は，成果物を作成するときのノウハウや，成果物がどのような経緯で設計されたのかといった，開発者が記述することのできるテキストのことである．開発者が記したテキストを，注釈として開発者知識ネットワークに蓄積する．

### 3.1.2 開発者知識ネットワークの要素間の関係

開発者知識ネットワークでは,「添付」,「対象」,「導出」,「包含」,「閲覧」，および「依存」の 6 つの関係が定義されている．

開発者知識ネットワークにおけるそれぞれの関係を図 1 に示す．添付は，開発者と注釈の関係で，どの開発者がどの注釈を添付したかを表す．対象は，注釈と構成物の関係で，どの注釈がどの構成物に添付されているかを表す．導出は，成果物と成果物の関係で，どの成果物からどの成果物が作成されたかを表す．包含は，成果物と構成物，もしくは構成物と構成物の関係で，どの構成物がどの構成物，または成果物に含まれているかを表す．閲覧は，開発者と構成物の関係で，開発者が構成物にどれだけ関わっているかを表す．依存は，構成物と構成物の関係で，ある構成物が別の構成物にどれだけ影響を与えているかを表す．

添付，対象，導出，および包含の 4 つの関係は，いったん要素間に関係がつけられると関係が変化することはないが，閲覧と依存の 2 つの関係は，開発者の行動によって関係の強さが動的に変化する．

### 3.1.3 開発者の行動からの動的な関係の抽出

本手法では，開発者が開発作業中に取る行動のうち，どの構成物を閲覧しているか，という点に着目する．図 2 に開発者の行動の例を示す．図 2 では，開発者は構成物 A, B, C の 3 つの構成物を閲覧しながら開発作業を行っている．図中の実線で表された部分は，その時刻に構成物を閲覧していたことを示す．本手法では，図 2 のように開発者が 1 つの構成物を閲覧している部分を「区間」として表現する．

また，図 2 では，時刻 $t_7$ から $t_8$ の間に，構成物 C を保存する動作を行っている．本手

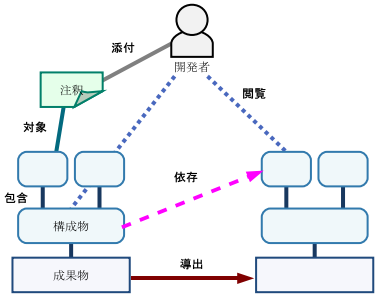



**図 1** 開発者知識ネットワークの要素間の関係

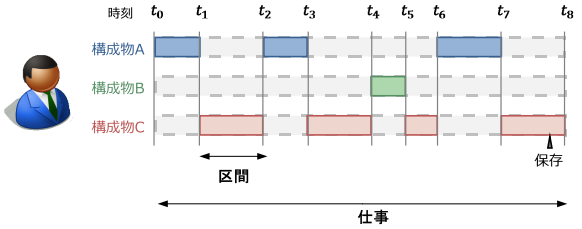



**図 2** 開発者の行動の例

法では，作業が開始されてから保存が行われるまでの区間の集合を「仕事」と定義する．仕事は，ある構成物を編集するための区間の集まりであると捉えることができる．

本手法では，開発者の行動である「仕事」から，開発者知識ネットワークの関係である「閲覧」の関係を抽出する．図 2 の場合，時刻 $t_7$ から $t_8$ の区間で構成物 C を保存しているため，構成物 A と構成物 B を参照するために閲覧しながら，構成物 C を編集，および閲覧をしていた，と考えることができる．そこで，仕事の単位ごとに，開発者がどの構成物をどの程度の時間，参照，または編集していたかを，開発者知識ネットワークに記録する．この

手法により，開発者と構成物との関係を自動的に抽出し，蓄積しておくことができる．

### 3.2 開発者コンテキストを用いた知識推薦

本節では，開発者の作業状況に合わせた自動的な知識推薦の手法について述べる．

#### 3.2.1 開発者の参照行動を利用した自動的な知識推薦

開発者にとって負担の少ない知識共有を実現するためには，開発者が自分で知識の検索を行わなくても，開発者の作業状況に合った有益な知識を推薦できる必要がある．

そこで，本手法では，開発者が成果物を参照する行動に着目し，協調フィルタリングを利用する．成果物に対する開発者の参照履歴から，協調フィルタリングを用いることによって，開発者がまだ参照しておらず，開発者に関連のある知識を抽出し，推薦する．

実際の開発現場では，開発者は様々なプロジェクトに参加し，さまざまな役割を担当する．したがって，開発者の閲覧履歴には，さまざまな状況が混在しており，そのまま協調フィルタリングを用いた知識抽出を行うと，開発者が今いる状況とは関係のない知識が抽出されてしまう．そこで，開発者知識ネットワークで定義されている「仕事」の単位に着目する．1 つの仕事の中で行われている開発作業は，1 つの構成物を編集する作業であり，そのときに参照されていた構成物同士には互いに関係があると考えられる．そのため，開発者の閲覧履歴を，仕事の単位で区切り，知識推薦を受ける開発者の現在の作業の状況と同じ状況の仕事のみから閲覧履歴を抽出することで，互いに関連の強い閲覧履歴のみを用いて協調フィルタリングを行うことができる．

本手法では，作業を行っている開発者は「ある 1 つの状況で作業を行っている」と捉え，この状況を「開発者コンテキスト」と呼ぶ．協調フィルタリングを用いて知識推薦を行う場合には，他の開発者の閲覧履歴のうち，知識推薦を受ける開発者の開発者コンテキストと同じコンテキストで行われていた仕事の閲覧履歴を利用する．開発者の現在のコンテキストと他の開発者のある仕事のコンテキストが同じかどうかは，現在の仕事で開発者が閲覧した構成物がその仕事でも閲覧されていたかどうかで判断する．開発者コンテキストを考慮して抽出された閲覧履歴から協調フィルタリングを行うことによって，開発者の状況に合わせた知識の推薦が可能となる．

#### 3.2.2 知識推薦の手順

本節では，開発者のコンテキストから開発者の状況に合った知識を推薦するための具体的な手順について述べる．本手法では，開発者のコンテキストから，開発者がまだ閲覧していない構成物の重要度のスコアを求める．このスコアは，0 〜 1 の数値で，数値が大きいほど，同じコンテキストで他の開発者が多く閲覧していたことを意味する．得られたスコアが高いものから順に，開発者に知識推薦を行う．知識推薦のためのスコアは，あるコンテキストでの開発者の閲覧履歴の取得，開発者間の類似度の算出，開発者の構成物に対する閲覧の評価値の算出，構成物への閲覧のスコアの算出を順に行うことによって算出する．

以下で，それぞれの手順について説明する．

#### 3.2.3 あるコンテキストでの開発者の閲覧履歴の取得

協調フィルタリングで開発者の閲覧履歴からのスコアの計算を行うために，3.2.1 節で述べた開発者コンテキストに基づいた閲覧履歴の抽出を行う．

開発者の集合を $D$，知識推薦を受ける開発者を $d \in D$ とし，開発者の現在のコンテキスト $c$ での構成物の閲覧履歴を $R_{d,c}$ とする．また，他の開発者 $o \in D$ の仕事の集合を $W_o$ とし，開発者 $o$ の仕事 $w \in W_o$ は，構成物の閲覧履歴 $r$ の集合であるとする．コンテキストが $c$ である仕事の集合 $W_{o,c}$ は，構成物の閲覧履歴 $r \in R_{d,c}$ を含む仕事の集合であり，

$$W_{o,c} = \{w \mid w \in W_o, r \in w, r \in R_{d,c}\}$$

となる．また，他の開発者のコンテキスト $c$ での構成物の閲覧履歴 $R_{o,c}$ は，

$$R_{o,c} = \bigcup_{w \in W_{o,c}} w$$

となる．ここで得られた $R_{o,c}$ を，開発者 $o$ の閲覧履歴として利用する．

#### 3.2.4 開発者間の類似度の算出

3.2.3 節で得られた開発者の閲覧履歴から，知識推薦を受ける開発者 $d$ と他の開発者 $o$ の閲覧行動の類似度を求める．開発者間の閲覧行動の類似度を求めることで，開発者 $d$ と類似した行動をしている開発者の閲覧履歴をより強くスコアに反映させることができる．

開発者間の行動の類似度は，$R_{d,c}$ に含まれている閲覧履歴のうち，同じものが $R_{o,c}$ にいくつ含まれているかで求める．コンテキスト $c$ における開発者 $d$ と開発者 $o$ の行動の類似度を求める式 $Sim(d,o,c)$ は，次のようになる．

$$Sim(d,o,c) = \frac{|R_{d,c} \cap R_{o,c}|}{|R_{d,c}|}$$

この式で，開発者間の類似度を 0 〜 1 の数値で求めることができる．

#### 3.2.5 開発者の構成物に対する閲覧の評価値の算出

コンテキスト $c$ において，開発者 $o$ が 閲覧履歴 $r$ でその構成物をどの程度閲覧していたのかを，評価値として求める．評価値を求める式 $Rate(r,o,c)$ は，次のようになる．

$$Rate(r,o,c) = \begin{cases} 1 & r \in R_{o,c} \\ 0 & otherwise \end{cases}$$

今回は単純に，コンテキスト $c$ で開発者 $o$ が $r$ が指す構成物を閲覧している場合は 1，そ

うでない場合は 0 とした．

#### 3.2.6 構成物への閲覧のスコアの算出

最後に，協調フィルタリングを利用して，知識推薦を受ける開発者がまだ閲覧していない構成物への参照 $\hat{r} \in R_{d,c}$ のスコアを求める．$\hat{r}$ のスコアは，開発者の集合 $D$ に含まれる開発者について，それぞれの開発者の開発者 $d$ との類似度 $Sim(d,o,c)$ とそれぞれの開発者の $\hat{r}$ に対する評価値の積を平均することで求める．スコアを求める式 $Score(\hat{r},d,c)$ は，次のようになる．

$$Score(\hat{r},d,c) = \frac{\sum_{o \in D, o \neq d} Rate(\hat{r},o,c) \times Sim(d,o,c)}{\sum_{o \in D, o \neq d} Sim(d,o,c)}$$

この式を用いることで，類似した閲覧行動をとっている開発者がどの構成物を多く閲覧していたのかを求めることができ，開発者のコンテキストに合った知識推薦を行うことができるようになる．

## 4. シミュレーション

本稿で提案している知識推薦の手法は，開発者の行動を利用して知識抽出を行うため，本手法が開発者知識ネットワークに蓄積されている開発者の行動の精度から受ける影響について検証を行う必要がある．そこで，開発作業のモデル化を行い，シミュレーションによって本手法の検証を行う．また，知識推薦のときに，開発者コンテキストの考慮の有無が及ぼす影響について検証を行う．

### 4.1 シミュレーションのモデル

本節では，提案する開発者コンテキストを考慮した知識推薦手法の検証のための開発作業のシミュレーションのモデルについて述べる．

#### 4.1.1 想 定 環 境

今回のシミュレーションでは，複数人の開発者によってプロジェクトが 1 つずつ実行され，構成物が作成されていくことを想定する．開発者は，プロジェクトと作成する構成物を割り当てられ，構成物を作成する作業を行う．すべての作業が終了すると，開発者は再び新たなプロジェクトを割り当てられ，構成物を作成する．このようなプロジェクトを複数回実行し，開発者が開発作業を行っていくことで，開発者知識ネットワークにプロジェクトや構成物に関する情報が蓄積されていく．

#### 4.1.2 開発作業のモデル

プロジェクトの中で開発者がどのような行動をするかを決定するために，開発者が行う開発作業のモデル化を行う．開発者の作業中の行動を決定するために，プロジェクトで作成される構成物に「内容」を割り当てる．1 つの構成物には 1 つのランダムな内容が含まれる．開発者は，ある内容を含む構成物を作成するときに，過去に作成された同じ内容を含む他の構成物を閲覧しながら作業する．これにより，開発者が過去の成果物を参考にしながら自分の作業を行う状況を再現する．また，ある内容を含む構成物を作成する作業は，開発者の集合のうちの，特定の数名にしか割り当てられないようにする．これにより，ある内容についてよく知っている開発者とほとんど知らない開発者がいる状況を作り，開発者が持つ知識に偏りが出ている状況を再現する．

この行動から，3.4 節で述べた「閲覧」の関係を抽出する．1 つの構成物を作成する作業を 1 つの「仕事」と捉え，その作業中に行った参照を「1 つの仕事の中での閲覧」とし，開発者知識ネットワークに蓄積する．

### 4.2 検 証 方 法

前節で述べた開発作業のモデルをもとにシミュレーションを行い，手法の検証を行う．手法の検証は，シミュレーションによる開発者知識ネットワークの構築と，手法による知識抽出の，2 つのステップからなる．

#### 4.2.1 開発者知識ネットワークの構築

開発者知識ネットワークの構築のステップでは，開発作業のモデルに対してパラメータを与えてシミュレーションを実行し，開発者知識ネットワークの構築を行う．今回のシミュレーションで利用するパラメータの一覧を表 1 に示す．

表中の「開発者が作業内容に関連した構成物を閲覧する確率」は，開発者が作業中に他の構成物を閲覧するときに，作成している構成物の内容と同じものを含む構成物を閲覧する確率を表す．実際の開発作業では，開発者が 1 つの仕事の中で関係のない構成物を閲覧する場合や，開発者の仕事が正しく抽出できない場合が考えられる．そこで，これらの状況を再現するために，ある確率で開発者が作業内容と関連のない構成物を閲覧するようにする．関連した内容を閲覧する確率が下がるほど，開発者知識ネットワークに蓄積される閲覧の履歴が不正確なものになり，抽出される構成物の適合率も減少していくことが予想される．開発者が作業内容に関連した構成物を閲覧する確率が $x$ のときに構築された開発者知識ネットワークを「精度 $x$ の開発者知識ネットワーク」とし，ある精度の開発者知識ネットワークをシミュレーションによって構築する．確率を 0〜1 の間で 0.05 刻みで変化させ，それぞ

Vol.2010-GN-76 No.18
2010/5/21

**表 1** シミュレーションで利用するパラメータの一覧

プロジェクトに関わるもの

| パラメータ | 値 |
|---|---|
| 開発者数 | 100 |
| プロジェクト数 | 100 |
| 1 つのプロジェクトに参加する開発者数 | 5 ～ 10 |
| 1 つのプロジェクトで作成される構成物数 | 90 ～ 110 |

開発者の行動に関わるもの

| パラメータ | 値 |
|---|---|
| 構成物に含まれる内容の種類数 | 500 |
| 1 人の開発者が知っている内容数 | 50 |
| ある内容について知っている開発者数 | 10 |
| 開発者が 1 つの仕事で閲覧する構成物数 | 3 ～ 7 |
| 開発者が作業内容に関連した構成物を閲覧する確率 | 変動 |

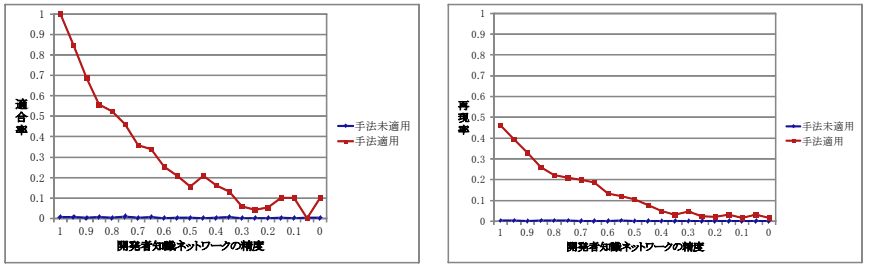
れの確率をパラメータとして与えたシミュレーションを行うことで，開発者知識ネットワークの精度が抽出される構成物の適合率にどのような影響を与えるのかを検証する．

#### 4.2.2 適合率の算出

シミュレーションによって構築された開発者知識ネットワークから，実際に知識抽出を行い，抽出された知識の適合率を求める．適合率の算出を行うため，ある開発者がある内容を含む構成物を作成する状況を想定する．この開発者は，開発者知識ネットワークが構築されるときと同様に，作業内容と同じ内容を含む過去の構成物を閲覧する．この構成物の閲覧履歴を利用し，提案手法と，開発者コンテキストを考慮しない通常の協調フィルタリングで，それぞれ知識抽出を行う．抽出された構成物が含む内容が，知識推薦を受ける開発者が作成している構成物の内容と同一であれば「関連がある」，さもなくば「関連がない」とし，適合率を求める．

また，知識抽出を行うときに利用する，知識推薦を受ける開発者の閲覧履歴の個数を変化させる．協調フィルタリングでは，ユーザ間の類似度や，アイテムのスコアを計算するときに，情報推薦を受けるユーザの行動履歴の数によって精度が変化する．そのため，閲覧履歴の個数を変化させた場合，どちらの知識抽出手法も構成物のスコアを算出する過程で影響を受ける．また，提案手法では，他の開発者の同一コンテキストの仕事における閲覧履歴を抽出する場合にも，知識推薦を受ける開発者の閲覧履歴の個数に影響を受ける．そこで，開発者の閲覧履歴の個数を 1～10 まで 1 つずつ増やし，閲覧履歴の個数が適合率に与える影響を検証する．

今回の検証では，それぞれの精度の開発者知識ネットワークに対して，閲覧履歴の個数が




(a) 閲覧履歴数 2 のときの，スコアの上位 10 個の構成物の適合率

(b) 閲覧履歴数 2 のときの，スコアの上位 10 個の構成物の再現率

**図 3** 開発者コンテキストの有無によるの適合率，再現率の変化

1～10 の場合でそれぞれ 100 回ずつ適合率の算出を行い，平均を求めた．

### 4.3 シミュレーション結果と考察

#### 4.3.1 手法適用の有無が適合率に与える影響

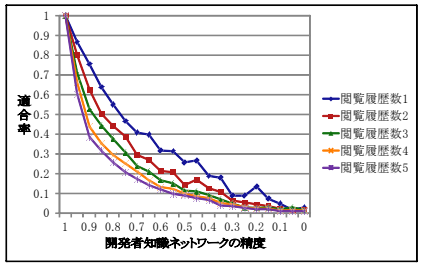
シミュレーションによって構築された開発者知識ネットワークから，提案手法を用いた抽出方法と，通常の協調フィルタリングを用いた抽出方法の，それぞれの方法を利用して知識抽出を行い，抽出された構成物の適合率と再現率を調べた．開発者の閲覧履歴を 2 個利用して知識抽出を行い，得られたスコアのうち上位 10 個の構成物の適合率のグラフを図 3(a) に，再現率のグラフを図 3(b) に，それぞれ示す．

図 3(a) では，手法を適用していない場合の適合率は，非常に低かった．例えば，開発者知識ネットワークの精度が 1 のときでも，スコアの上位 10 個の構成物の適合率は 0.006 となった．一方で，手法を適用した場合は，開発者知識ネットワークの精度に比例して適合率が下がってしまうものの，関連のある構成物が抽出されていた．

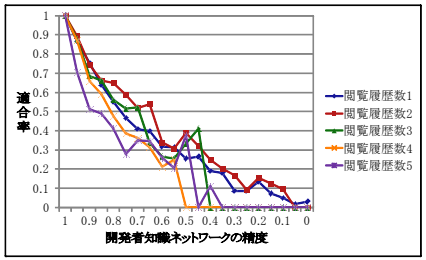
図 3(b) では，抽出された構成物のうちスコアの上位 10 個の構成物の再現率は，開発者知識ネットワークの精度が減少するにつれてほぼ線形に減少していった．しかし，手法を適用しない場合，再現率はほぼ 0 になってしまった．

提案手法では開発者の閲覧履歴の一部のみを利用するのに対し，手法を適用していない場合は閲覧履歴の全体を利用する．そのため，手法を適用していない場合は抽出される構成物の数が多くなり，大多数が関連のない内容のものとなる．したがって，有効な知識抽出を行えているとはいえない．これに対し，提案手法では，開発者コンテキストを考慮した閲覧履

Vol.2010-GN-76 No.18
2010/5/21




(a) 提案手法で抽出されたすべての構成物の適合率




(b) 提案手法で抽出されたスコアが 0.8 以上の構成物の適合率

**図 4** 提案手法による抽出結果

歴を利用することで，関連のある構成物が抽出される可能性が低くなることが考えられるが，閲覧履歴から関連のある知識を抽出することができた．これらのことから，手法を適用することで，抽出する知識の適合率を大幅に向上させることが可能になることが示された．

#### 4.3.2 開発者知識ネットワークの精度が適合率に与える影響

図 4(a) は，提案手法で抽出された構成物のリストについて，スコアを考慮せずにすべての要素の適合率を求めたものを示している．また，図 4(b) は，提案手法で抽出された構成物のリストから，スコアが 0.8 以上の要素だけの適合率を求めたものを示している．

図 4(a) では，開発者知識ネットワークの精度が 1 からわずかに減少するだけで，急激に適合率が減少していることがわかる．これは，ある内容を含む構成物は一定数しか存在しないのに対し，ある内容を含まない構成物は多数存在しているため，複数の開発者の閲覧履歴を重ね合わせることにより，関連のある構成物への閲覧履歴がほとんど増えずに，関連のない構成物への閲覧履歴が増えてしまうためである．また，知識抽出に利用する閲覧履歴の個数を増やしていった場合も，同様の理由により，適合率が減少している．

また，図 4(b) から，提案手法で求めているスコアを利用することで，適合率が向上することが示されている．ただし，閲覧履歴の個数が 3 個以上のときは，開発者知識ネットワークの精度が 0.5 を下回ったあたりから，適合率が 0 になっているが，これは抽出された構成物のうちスコアが 0.8 を上回るものが存在していなかったためである．これらのことから，提案手法で求めているスコアが，関連のある構成物の抽出に有効な働きをしていることがわかる．

しかし，スコアを考慮した場合でも，開発者知識ネットワークの精度が 0.8 のときは最も良い場合でも適合率が 0.65 となり，必ずしも良い結果が得られているとはいえない．そのため，開発者に有益な知識推薦を行うためには，仕事や閲覧履歴の抽出の精度が重要であるといえる．

## 5. アジャイル開発への適用

### 5.1 アジャイル開発における反復時の課題

システム開発において，顧客の要求の変更や開発内容の変更が発生した場合に，開発中の変更が成果物に及ぼす影響を把握することは，開発者にとって困難な場合がある．このような影響を把握することが遅延すると，開発効率の低下につながる．特にアジャイル開発では，要求定義やクラス設計，実装などの工程を反復して開発を行うために，把握の遅延による問題が発生する可能性が高い．そのため，アジャイル開発において変更による影響箇所の情報の共有を支援する環境が必要であると考えられる．

アジャイル開発では設計書や仕様書などの成果物を後のために残すことよりも，顧客の要求の変更に対応できるようにすることに重きをおいている．そのため，成果物に対して書かれる情報量が少ない場合や，成果物によっては開発終了後に破棄されてしまう場合がある．そのような成果物も，意味や価値を付け，管理を行えば有用な知識になると考えられるが，2.1.1 節でも述べた知識共有のための手法では，知識の登録や検索など，開発作業とは別の作業を行うことを前提としているため，アジャイル開発には適用することが難しい．

### 5.2 アジャイル開発における知識共有手法

本稿で提案する環境では，情報の収集と組織化を自動化して行うため，アジャイル開発でも適用できると考えられる．例として，アジャイル開発の代表的な開発手法であるエクストリームプログラミングへの適用を考える．エクストリームプログラミングで実践される計画ゲームで扱われるストーリーカード，タスクカード，および CRC カードの 3 種類のカードを，開発者知識ネットワークで収集し，開発者知識ネットワークの関係である「依存」で構造化する．変更が発生した場合,「依存」の関係を辿ることによって，影響が及ぶカードを開発者知識として抽出可能にする．また，顧客の要求の変更内容が新たな要求の追加であった場合，新規の要求が書かれたカードと既存のカードは関係が付けられていない．そのため，カードに含まれる用語の共起率が高いカード間に関連があるとし，共起率による関連度で開発者知識を抽出する．カードが追加されたときに，関連度と依存の関係を用いた開発者知識の抽出を行い，カードが修正されたときは,「依存」の関係のみで開発者知識の抽出を行う．

### 5.3 シミュレーション

#### 5.3.1 検証内容

実際にエクストリームプログラミングの計画ゲームのイテレーション（反復）で生成されるカードを提案する環境で管理し，アジャイル開発時に用いる開発者知識推薦手法を用いた場合に，どのような情報が開発者に提供されるかを机上で検証する．今回は，既存のソースコードを元に，開発者がどのようなカードを生成するか，机上でシミュレーションを行った．

検証では，計画ゲームを 3 回のイテレーションで開発者に行ってもらい，また，イテレーション中の開発者の行動履歴を元に，開発者知識ネットワークにおける「依存」の関係付けを行う．

本手法が実際の計画ゲームで有効性は，各カードが追加された後で本手法を適用し，提供されるカードの再現率と適合率を用いて考察する．再現率と適合率は，追加されたカードが実際に影響を及ぼすカードの数を $\alpha$，手法によって提供されるカードの数を $\beta$，手法によって提供されるカードのうち，実際に影響が及ぶカードの数を $\gamma$ とすると次の式で表される．

$$\text{再現率} = \frac{\gamma}{\alpha}, \text{適合率} = \frac{\gamma}{\beta}$$

#### 5.3.2 検証結果

シミュレーションによりストーリーカード，タスクカード，および CRC カードが追加，または修正されたときに手法を適用した場合の α，β，および γ の平均値，また，導出されるカードの適合率および，再現率の平均値を表 2 に示す．

**表 2** 1 枚のカードを追加・修正したときに導出されるカード数と適合率・再現率の各平均値

| | α | β | γ | 適合率 (%) | 再現率 (%) |
|---|---|---|---|---|---|
| ストーリーカード | 14.0 | 18.0 | 1.0 | 5.6 | 8.3 |
| タスクカード | 2.5 | 4.3 | 0.3 | 10.9 | 18.3 |
| CRC カード | 1.7 | 6.5 | 0.2 | 5.8 | 12.5 |

#### 5.3.3 考　　察

シュミレーション中に生成されたカードの分析の結果，抽出できる開発者知識には，イテレーションによって，要求や仕様の変更が発生したときに影響が及ぶカードと，開発を行うときに参考になるカードの 2 種類に分類可能であることがわかった．また，関連度を用いて開発者知識を抽出するとき，あるカードに含まれている用語が，多くのカードに含まれる用語であり，実際には関連のないカードが多数抽出され，適合率や再現率が低くなる場合が存在した．これは，開発者の状況を考慮していないことが原因の一つとしてみられる．そのため，イテレーションと開発者コンテキストを考慮した知識抽出を行えば，開発者の状況に適した知識推薦を行うことができると考えられる．特に開発者コンテキストの考慮は，4.3.1 からも知識抽出に利用する閲覧履歴の個数が少ないほど効果があるため，少人数で開発を行うアジャイル開発には有効であると考えられる．

## 6. おわりに

本稿では，開発者の状況に基づいて知識の収集と組織化を行う知識共有環境と，開発者の参照行動の履歴から開発者の作業に関連のある知識を自動的に抽出する手法を提案した．また，提案する環境をアジャイル開発へ適用例を示した．

知識推薦手法のシュミレーションの結果，開発者コンテキストを考慮することで，抽出する知識の適合率を大幅に向上させることが可能になることが示された．また，提案する環境をアジャイル開発への適用し，シュミレーションをした結果，アジャイル開発においても開発者コンテキストを考慮すれば適合率の向上が見られる可能性があることがわかった．

今後は，アジャイル開発において開発者コンテキストを考慮した知識抽出の検証を行う．

## 参考文献

1) Peter Senge. *The Fifth Discipline: The Art & Practice of the Learning Organization.* Currency, 2006.
2) ThomasH. Davenport and Laurence Prusak. *Working Knowledge: How Organizations Manage What They know.* Harvard Business Press, 1998.
3) 野中郁次郎, 竹内弘高, 梅本勝博. 知識創造企業. 東洋経済新報社, 1996.
4) 日本情報システム・ユーザー協会. 経営を変革するナレッジマネジメント〜その研究と提言〜. ナレッジマネジメント研究部会報告書, 2001.
5) 石田厚子他.　ソフトウェア開発における情報共有の課題と効果に関する研究. http://www.juse.or.jp/software/pdf/17_spc/17dep71.pdf.
6) 青山浩二, 鵜飼孝典, 小幡明彦, 原田裕明. 知識共有を動機付する手法. 人工知能学会 知識流通ネットワーク研究会 第三回研究会, 2008.
7) 小林賢治. 知識継承ソフトウェア KnowledgeMeisterSucceed$^{TM}$. 東芝レビュー, Vol.61, No.7, 2006.
8) 梅田恭子, 安田孝美, 横井茂樹. 知識メモを活用した研究情報共有方式の提案. 情報処理学会論文誌, Vol.42, No.11, pp. 2562–2571, 2001.
9) 森本由起子, 間瀬久雄, 平井千秋, 阿部琢哉, 大野治. システムエンジニア向け情報共有システムの開発. プロジェクトマネジメント学会誌, Vol.7, No.2, pp. 40–45, 2005.