解 説

ソフトウェア開発における知識コラボレーション

Knowledge Collaboration in Software Development

大平 雅雄 奈良先端科学技術大学院大学情報科学研究科
Masao Ohira Graduate School of Information Science, Nara Institute of Science and Technology.
masao@is.naist.jp, http://se.naist.jp/masao/

イェ ユンウェン 株式会社SRA先端技術研究所
Yunwen Ye Key Technology Laboratory, Software Research Associates Inc.
ye@sra.co.jp, http://www.sra.co.jp/ktl/yunwen_ye/

中小路 久美代 （同 上）
Kumiyo Nakakoji kumiyo@sra.co.jp, http://www.sra.co.jp/ktl/kumiyo_nakakoji/

山本 恭裕 東京工業大学精密工学研究所
Yasuhiro Yamamoto Precision and Intelligence Laboratory, Tokyo Institute of Technology.
yxy@acm.org, http://www.pi.titech.ac.jp/

Keywords: human-centric software development, knowledge collaboration, collective creativity, metabolic systems, immaterial labor.

1. はじめに

ソフトウェア開発では，規模の大小を問わず，多種多様な知識が求められる．外部世界にある知識の取込みや，ほかの開発者あるいはユーザとの知識交換など，ソフトウェア開発者は知識コラボレーションを通して開発を進めることが不可欠である．ソフトウェアを開発するという営為は，個々の人間の知的な作業であると同時に集団的でコラボラティブな作業である．本稿では，ソフトウェア開発における知識コラボレーションに関わる研究を解説する．

ソフトウェア工学分野の歴史を振り返ってみると，知識やコラボレーションの側面に着目したソフトウェア開発支援研究には，これまでに大きく三つの流れがあったと考えられる [Nakakoji 10b].

Fred Brooks が Mythical Man-Months（邦題は「人月の神話」）[Brooks 75] において，投入する開発者の人数を増やしてもソフトウェア開発作業のスピードが速くなるわけではないと著したのは1975年である．開発者（プログラマ）のスキルに研究の焦点を合わせた Psychology of Programming という研究分野が出てきたこの時期が，第一の流れである．expert と novice の比較実験などを通して，programming expertise の違いがプログラミング作業のスピードや質にどう現れるか，また，if 文や goto 文といったプログラムの要素や，オブジェクト指向などの方法論，変数名の付け方の効果や学習のしやすさなどが研究されていた [Shneiderman 80, Soloway 86]. Curtis ら [Curtis 88] が，大規模開発プロジェクトのフィールドスタディを通して行った報告において，ソフトウェア開発における三つの課題として，アプリケーションドメインの知識がばらばらに散逸していること，要求仕様をなかなか確定できないこと，コミュニケーションとコーディネーションが破綻すること，をあげたのもこの頃であり，当時からコラボレーションの重要性は指摘されていた．この時代には，AAAI のような人工知能分野においても ACM SIGCHI のような HCI（Human-Computer Interaction）分野においても，プログラミングというものが研究対象としてしばしば登場していたように思われる．

第二の流れは，ソフトウェア開発における協調作業の側面に着目したもので，グループや組織における開発作業のプロセス管理が中心となっている．1980年代末から1990年代にかけて盛んとなったこの流れは，複数人による開発のプロセスをプログラムすることができる [Osterweil 87]，といった Osterweil の考え方に代表されるように，ソフトウェア開発を工場における生産ラインとみなすものである．開発者は仕様書やテスト指示書などのドキュメントによって受けた指示どおりの作業を行い，次のアーティファクトを生産する役割を担うとされた．ソフトウェア開発における協調作業を組織として管理し，プロセスの追跡性や反復性といった観点から組織成熟度といった指標が提案された [Humphrey 89]. 興味深いことに，これらのアプローチにおいては，個々の開発者のスキルの違いは全く考慮されなくなっていった．組織論や経営論の観点からソフトウェア開発が語られることが多くなり，ソフトウェア工学分野の研究者と

AAAI や CHI といった研究分野の研究者らとの交流もほとんど途絶えたようになる．

2000 年代に入ると，第三の流れともいうべき，個々の開発者の認知的なプロセスと社会的なプロセスに着目した研究が行われるようになった．XP（eXtreme Programming）[Beck 99] に代表されるアジャイル（機敏）な開発手法が現場から広まり，個々の開発者が自らの判断とモチベーションとで開発作業の順序や手法を決定するような開発スタイルの有効性が認められ始めた．協調作業としてのソフトウェア開発も，それらを管理するという視点ではなく，個々の開発者が相互に知識や情報を交換しながらアジャイルなプロセスで状況や問題に対処するための方法論や表現手法といったものが着目され始めた [Tomayko 04]．また，オープンソースソフトウェア（OSS）プロジェクトが増えるにつれ，その開発データも公開されるようになった [Augustin 02]．開発途中におけるメーリングリストや掲示板などを介した開発者間のコミュニケーションの過程が広く研究者らにも扱えるものとなり，知識共創やコミュニティといった観点からソフトウェア開発が捉えられることが多くなった．

このような第三の流れを受けて，2000 年代後半からは再び，人間に着目する HCI や CSCW（Computer Supported Cooperative Work）といった研究分野の研究者らがソフトウェア開発を題材として扱うようになってきた．2007 年には，マイクロソフトリサーチが米国 University of Washington との共催で Human Side of Software Development という非公開のワークショップを実施している．ソフトウェア工学分野，HCI 分野，CSCW 分野といった米国内外の研究者約 60 名が集まり，五泊六日の合宿形式で行われた [UW MSR Institute 07]．個々人の知的創造活動が集積された知識コラボレーションとしてのソフトウェア開発支援研究は，これと前後して，加速度的に広まりを見せることになる．

ソフトウェア開発は，個々人の知的創造作業と，直接的あるいは間接的に関わるさまざまなサイズのグループとしての作業とを通して，持続的に行われる．それらの作業を通して，論理的に一貫性をもつ，複雑な，一つのアーティファクトが，ソフトウェアとして構築される．本稿での我々の問題意識は，ソフトウェア開発を工場の生産ラインや建造物の構築といったモノを対象とする製造工程になぞらえる限り，ソフトウェアそのものについてもソフトウェア開発についても，本質的な理解や展望は得られないのではないか，というものである．本稿では，ソフトウェア開発の知識コラボレーションに関わる研究を解説するにあたり，ソフトウェア開発を捉える視点を五つ提供したい．次章以降の各章において

- 知識共創活動（2 章）
- 新陳代謝プロセス（3 章）
- コミュニケーション集約型活動（4 章）
- 自己記述型活動（5 章）
- 知性の統合（6 章）

という観点から捉えたソフトウェア開発について，我々の考えを述べるとともに関連する研究を概観する．

2. 知識共創活動としてのソフトウェア開発

ソフトウェア開発は，多様な文化，役割，情報や知識をもった人々が，開発環境や，所属するグループや組織，また開発するソフトウェアそのものとのインタラクションを行いながら従事する，知識共創活動である．本章では，人と人，人と情報，人と環境，そして人と組織という四つのレベルにおける知識共創活動としてのソフトウェア開発について知識コラボレーションの観点から解説する．

2·1 異文化コラボレーション

ソフトウェア開発は，要求分析，設計，実装，テストといった多様な工程から構成されている．それぞれの工程において高い専門性が求められるだけではなく，前工程の作業成果物を利用するための知識や，専門性の異なる前工程の担当者と情報共有を行うためのコミュニケーションスキルも必要となる．特に，要求分析では，開発するソフトウェアシステムに関して，さまざまなステークホルダ（経営者，発注担当者，エンドユーザ，要求分析者，開発者など）が，対話を通じてシステムに対する要求を発見するための工程であり [Sommerville 97]，コミュニケーション集約型の極めて複雑な作業が求められる [Wiegers 99]．ステークホルダ間の背景知識や専門知識の違いにより意思疎通が困難になるため，たとえ経験豊富な要求分析者であってもすべての要求を的確に抽出することは容易ではない．結果的に，要求が曖昧なまま（場合によっては定義されないまま）開発が進んでしまい，後の工程で顕在化した要求を取り入れるために大きな手戻り作業を必要とすることが少なくない [大西 02]．

ソフトウェア開発でしばしば見られるこのような状況は，異なる文化の異なる言語を用いる者同士が相互理解を構築しながら共通の目的を達成しようとする活動 [Nakakoji 96]，すなわち，異文化コラボレーションと捉えることができる．特に要求分析では，顧客のもつ要求を正確に抽出するというよりは，顧客と分析者が協力し合い，システム構築に求められる要件を共に生み出すという姿勢が重要である．例えば達ら [達 07] のアプローチは，ソフトウェア要求分析を異分野コラボレーションとして捉えるものである．従来工業意匠のデザインプロセスにおける顧客とデザイナとの相互理解支援を目的として開発された異分野協調作業支援環境 EVIDII[大平 00] を，要求抽出会議に適用した実験を行い，従来の会議方法に比べ，より多くの要求を抽出できることを確認している．

2·2 情報と知識の創出

ソフトウェアシステムは知識アーティファクトであ

る．物理的な工業製品とは異なり，ソフトウェアシステムを構成する物質的な原材料は存在しない．ソフトウェアシステムを開発するにあたって使われる「原材料」と呼べるものは，開発者のもっている知識である．ソフトウェア開発者がソフトウェアを開発する際には，情報収集と情報創出のサイクルを繰り返す．

一人の開発者が有している知識で，開発に必要となるすべての知識をカバーすることは，現実的にはあり得ないと考えられる．自分のタスク遂行に必要な情報を，自らで集めてくる必要がある．開発活動のうち24～34%の時間がソースコードの検索と調査による情報収集活動であるとされる[Singer 97]．明示的なコミュニケーションを介した知識交換に限らず，作業結果としてのドキュメントやプログラムも，外在化した知識媒体としてほかのソフトウェア開発者と自分の今後の作業の知識の源となる．

情報収集にあたって，必要となる知識はすべて外在化したものであるとは限らず，共同作業している同僚の頭の中にしか存在しない場合もある．de Souzaらの研究[de Souza 04]によると，システムの構成要素を機能的に独立したモジュールごとに集約するモジュール化を行うだけでは，開発タスクの相互依存性を完全に解消することはできない．ほかのモジュールに依存する部分を開発する場合，前もってデザインされたインタフェースだけを知るだけでは不十分であり，そのインタフェースを実装する詳細を知らなければ開発が進められないといったことがしばしば発生する．つまり，開発タスク間の技術的な依存関係が社会的な依存関係をもたらし，アドホックな協働の必要を生み出す．その際に，積極的にコミュニケーションを起こして，他人から協力を得る能力が求められる．また，他人に協力を求められているときに協力してあげようとする姿勢も技術力と同様に重要である．

複雑なソフトウェア開発では，必要な知識が開発者間に分散されていることが多く，プロジェクトの全体的なKnowing capabilityは個々人のもっている知識を足すだけではなく，個々のソフトウェア開発者のインタラクションにより，相乗効果を求めるべきである．開発にあたって，個々人の開発者のもっている知識をいかに円滑にほかの開発者にトランスファできるかが開発全体の品質と効率の大きな決定要因となる．

2・3 知識増幅のための開発環境

ソフトウェア開発におけるライブラリの利用は，生産性の向上と品質の改善に大きく貢献するとされている．オブジェクト指向技術の成熟とOSSの進展により，再利用できるライブラリの数が急速に増え続けている．Javaの標準ライブラリが含むクラスの数は，1996年で配布された1.0版の212個から2006年で配布された1.6版の3 777個まで増加してきた．これらのクラスライブラリを再利用することにより開発に必要な労力が軽減される一方で，開発者がそのようなクラスライブラリをいかにして習得して利用するのかが大きな課題である．

すべてのライブラリを記憶してから開発に臨むことはもはや不可能である．使うときに必要なライブラリを迅速に発見して，熟知していないライブラリでも利用できるようにする支援技術を，開発環境自体に組み込むという研究が行われてきた．Eclipseが代表する現代のIDEが提供しているクラス名やメソッド名の自動補完機能はその一つである．

ライブラリの数が増えすぎた結果，ある機能のライブラリコンポーネントがすでに存在することを知ることすら困難になってきた．CodeBroker[Ye 05]は，この問題を解決するために開発された自律的にライブラリコンポーネントを推薦するシステムである．CodeBrokerは，プログラマがエディタでJavaプログラムを記述している際にコメント文を入力すると，そのコメントに記述されたテキストの内容と機能的に近いライブラリコンポーネントを自動的に推薦するという機能を備えている．さらに，プログラマの既存知識をユーザモデルとして構築し，そのプログラマに特化した推薦を行う．CodeBrokerを利用することで，存在すら知らなかったライブラリも，すでに知っているもののように利用することができる．

さらに，LittleとMiller[Little 07]は，キーワードプログラミングというアプローチを提案している．ライブラリの正確な名前を知らなくても，なんとなくそれに近いキーワードを入力するだけで，システムがエディタのコンテキストを利用して，自動的に適切なメソッドの呼び出しに変換する仕組みである．例えば，ファイルから一行のテキストを読み込もうとする．“add line”を入力すれば，システムはlines.add（in.readLine（））と自動的に変換する．

これらの開発支援技術により，プログラミングという知的作業に必要となるすべての知識をプログラマの頭脳に内在化する必要がなくなる．知識の一部が外部環境から適時提供されるという形を通じて，プログラマとプログラミング環境が一つの分散認知システムとして機能する．結果として，ソフトウェア開発が，開発者と開発環境との共同作業として構成されることになる．

2・4 ソシオテクニカルコングルエンス

ソフトウェア開発チームは一つの知識コミュニティを形成するとみなせる．このコミュニティのメンバ間で，知識の流通と共創に対する難易度と有効性を決める要因として，さまざまな距離と近接性がある[イェ 08]．

ソフトウェア開発タスクは，組織の構造に沿って割り当てられることが多い．これにより開発者の間に〈組織的〉な距離と近接性が生じる．また，開発者が抱えている開発タスク間に依存性があれば，開発者間に〈テクニカル〉な距離と近接性があると考えられる．

図1に，Apache Httpdプロジェクトにおける，組織

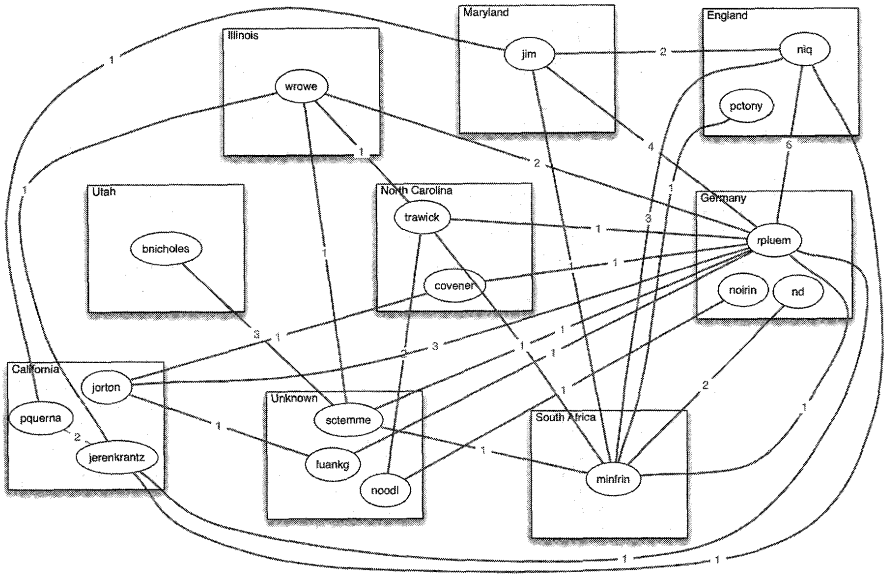


図1 Apache Httpd における組織的近接性とテクニカルな近接性

的近接性とテクニカルな近接性を表す例を示す．各楕円が各開発者を示し，同一の組織に属する開発者を四角で囲っている．開発者間の実線は，それらの開発者間にテクニカルな近接性があることを示し，数字はその強さを示す [Ye 08].

Conway's law[Conway 68] は，システムの構造はそれを生産する組織の構造と一致する，というものである．この系として，開発するソフトウェアシステム構造と，それを担う組織の構造とが合致する（socio-technical congruent となる）べきであるという主張がある [Cataldo 08]．Nagappan らの研究 [Nagappan 08] では，ソースコードを修正する開発者が複数の組織部門にまたがって所属していると，ソースコードの品質が低下する傾向にあると報告している．図1で示す2種類の近接性ができるだけ一致するように開発タスクと開発要員を適切に再配置し調整することで，このような問題を解消することができると考えられる．

3. 新陳代謝プロセスとしてのソフトウェア開発

ソフトウェア開発は，長期にわたる流動的な変化を内包する新陳代謝のプロセスと捉えることができる．本章では，オープンソースソフトウェア（OSS）開発の事例を用いながら，開発されるソフトウェアシステムとそれを開発する人との相互作用による新陳代謝の作用と，そのプロセスを解説する．

3·1 持続可能な知識継承

ソフトウェア開発案件の多くは，新規開発ではなく，機能の追加や保守性を高めるために行う派生開発である．派生開発によってシステムが長期間利用され続けるため，常に固定された開発メンバがシステムの開発・保守に携わるのではなく，ある程度の流動性を伴って開発メンバも変化し続ける．開発プロセス全体に大きな変動はないものの，開発に携わるメンバが定期的に入れ換わるその様は，ソフトウェア開発における新陳代謝のプロセスといえる．

そうした新陳代謝プロセスは，特に，人気の高い OSS の開発においてしばしば観察される．例えば，Apache や PostgreSQL は，10 年以上の長きにわたり，ボランティアの開発者によって機能拡張および保守され続けている．その間，古参の開発メンバが脱退したり，開発プロジェクトに多大な貢献を行った開発者がコア開発者として昇格したりするなど，緩やかではあるが絶えず開発メンバとその役割が入れ換わっている [Bird 07, Fujita 10, Jensen 07] ものの，ソフトウェアの品質は常に一定に保たれている．

一般的なソフトウェア開発組織では，保守担当者が急に変更されると，後任の担当者がドキュメントやコードに関する知識を十分に引き継ぐことができず，システムの保守性を低下させたり不具合を引き起こしてしまうことは珍しくない．OSS 開発における「緩やかな新陳代謝プロセス」は，ソフトウェア開発における知識継承の手段として注目に値するであろう．

3·2 三つのレイヤの共進化

前述の点をさらに深く掘り下げると，ソフトウェア開発は，開発プロジェクトのメンバが個々に開発作業に関わりながら，一つのアーティファクト（ソフトウェアシステム）をつくり上げるプロセスである．個々人の開発者がよく知っているオブジェクトや得意とする分野は，重なる部分はあるもののそれぞれ異なっており，プロジェクト全体で見ると，symmetry of ignorance（無知の均衡）[Fischer 00] 状態となっている．開発が進む

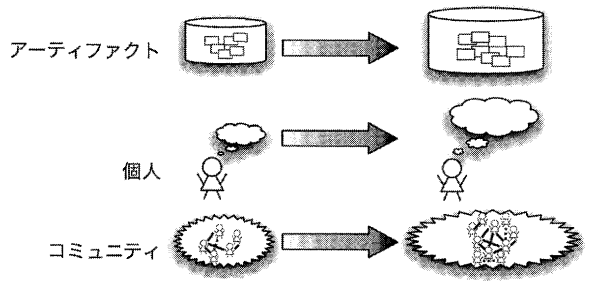


図2 三つの異なるレイヤでのevolution

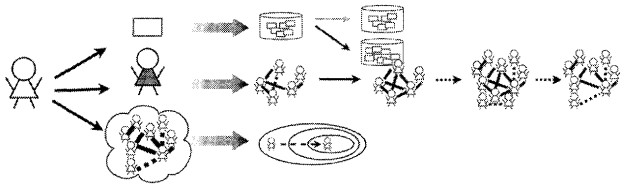


図3 三つの異なるレイヤのco-evolution

につれて，開発者同士の相互理解は進み，また個々の開発者の，プロジェクトそのものに対する能弁さ（project fluency）が進んでいく[Zhou 10]．プロジェクト自体が，作業コミュニティとして成長していっているとみなすことができる．

とりわけOSS開発においては，開発メンバがコミュニティを形成し，自発的に参加者として関わりながら一つのソフトウェアシステムを育て上げていく[Hippel 03]．Pangaro[Pangaro 00]やGiaccardiら[Giaccardi 05]は，協調作業を支援しながらユーザとアーティファクトを結びつけるような社会技術的インフラストラクチャをParticipative Systemと呼んでいる．ここでいうシステムとは，計算機環境としてのシステムではなく，ソシオテクニカルな仕組みとしてのシステムである．OSS開発は，そのようなParticipative Systemの一例であるとみなすことができる．なお，類似した用語としてParticipatory Systemがあるが[Ehn 90]，これはユーザ参加型のシステム開発を指すものであり，Participative Systemとは異なる概念である．

このようなParticipative SystemとしてのOSS開発においては，三つの異なるレイヤでのevolution（成長あるいは進化）が起こっていると考えられる（図2）[中小路 07]．一つ目は，アーティファクト（ソフトウェアシステム）そのものの成長，二つ目は，開発者個々人の知識やスキルの増加による成長，そして三つ目は，コミュニティとしてのプロジェクトグループそのものの成長である．

これら三つのレイヤにおける成長は，相互に依存している．個々のアーティファクトの進化は，個々のメンバが開発作業を進めることによって可能となる．個々のメンバの貢献は，個々のメンバの知識の増加やスキルの向上へとつながる．アーティファクトが進化し，質が向上することによって，新たなメンバが加わったり，コミュニティの結束が高まったりすると考えられる．これら三つのレイヤのおのおので，進化的な成長が進むことで，コミュニティが持続していると考えることができる．そしてこれらの成長は，個々の開発者が，アーティファクト，他の開発者，プロジェクト，それぞれと関わることによってのみ生じている．個々の開発者の開発作業が，知識生態系における代謝のプロセスを促進する要因となっていると考えることができる（図3）．第一に，個人とアーティファクトとの関係の変化による，アーティファクトの発展がある．第二に，個人と他のメンバとの関係の変化による，コミュニティ内の社会的関係の発展がある．第三に，個人とコミュニティとの関係の変化による，コミュニティ構造の発展がある．

個人が“knowledge in the world（世界にある知識）”[Hutchins 96]すなわち，ドキュメントやアーティファクトとして表された情報や，他者の有している知識とインタラクションを行いながら，個々人が知的創造作業に関わるプロセスや，またそれによってつくりだされるモノを，コレクティブクリエイティビティ（Collective Creativity）と呼ぶ[Nakakoji 00]．個々の開発者の視点からソフトウェア開発のプラクティスを捉えると，ソフトウェア開発作業は，コレクティブクリエイティビティに関わるプロセスとみなすことができる[Nakakoji 06]．

4. コミュニケーション集約型活動としてのソフトウェア開発

ソフトウェア開発におけるコミュニケーション支援は古くから取り組まれている研究課題であり，新たなコミュニケーション支援技術を取り入れるアプローチも多くある[山本 10]．本章では，ソフトウェア開発におけるコミュニケーションを巡る様相の変化と，コミュニケーションにおける人と営為の側面に注目したアプローチについて解説する．

4·1 コミュニケーションに対する認識の変化

ソフトウェア開発におけるコミュニケーションを支援するための最近の研究の背景には，次の三つの点があると考えられる．

(1) アジャイル開発が広まって，コミュニケーションが，開発作業のオーバヘッドではなく開発作業の一部とみなされるようになってきたこと
(2) オープンソースソフトウェア（OSS）の普及により，開発時のコミュニケーションデータがアーカイブされ，分析可能となったこと
(3) いわゆるIDE（Integrated Development Environment）と呼ばれる開発環境がOSSとなり，コミュニケーションに関連するメカニズムを開発環境に組み込めるようになってきたこと

ソフトウェア開発において「コミュニケーション」が重要な役割を果たすことは，1980年代から広く認識されていた（例えば，[Curtis 88, Fairley 85]）．しかしながら，コミュニケーションに対する認識は，今は少し変わってきているように思われる．当時はいかにしてコミュニケーションを減らすか，ということが研究の前提としてあった．それに対して2000年以降のコミュニケーションを取り巻く研究は，いかにしてコミュニケーションが円滑に行えるようにするか，ということがゴールとして考えられているように見受けられる．

4·2 コミュニケーションの観察

コミュニケーションを探ろうとするエスノグラフィやフィールドスタディを中心としたアプローチの多くは，研究者が，企業の開発プロジェクトの中に参加し，中長期的に開発環境や開発における開発者間のやり取りを観察するというものである．こうしたアプローチの多くは，日頃から感覚としてソフトウェア開発者らが感じていることを，データとして実証するようなものが多い．そのような感覚を数値データや知見として裏付けすることの意義は，ソフトウェア開発における知識コラボレーションを理解するうえで非常に大きいと考える．

例えば，Koら [Ko 07] は，マイクロソフト社のプロジェクトに参加し，どのような情報を，どのようなソースから最も頻繁に利用しているか，を調査したものである．研究では，17人の開発者について，それぞれ90分の間にどのような情報を利用しているかのデータを収集，分析し，7種類のアクティビティと21種類の情報のタイプを利用していることを同定している．また，それとは別に，42人の開発者に，情報の重要性についてサーベイを行っている．これらの結果から，開発者が最も頻繁に知りたい情報は，「コードに何か間違いがあるのか？」，「(あの) メンバは今何をしているのか？」，「どの部分のせいでこうなっているのか？」であったことを報告している．

また，Damianら [Damian 07] は，IBMにおける，米国，ヨーロッパ，カナダの計9か所をまたがるプロジェクトにおけるコミュニケーションの調査を行っている．研究では，米国とカナダとの4か所のサイトで関わったコンポーネント開発のデータに注目した結果を報告している．それぞれのサイトの間で，不具合管理ツールBugzillaを用いたコミュニケーションの頻度をグラフ表示したところ，ほぼ均等にサイト間でのコミュニケーションが発生しており，そのコンポーネントに関する知識や情報が均等に分散していると思われるといった報告をしている．

また，開発者間で最も頻繁に利用されたコミュニケーションメディアは，対面，Bugzilla，電子メール，メーリングリストの順であるが，どのような情報をやり取りしたかによってその頻度には違いがあり，例えば，作業の進め方の相談をするには対面によるコミュニケーションの頻度が圧倒的に高いが，プログラム上の変更点を伝えるには，メールでのコミュニケーションの頻度が高いといったことを報告している．

ただし，これらエスノグラフィックスタディやフィールドスタディで得られた知見の取扱いについては，次の二つの点で課題もあると考える．

第一に，データから得られた知見が，研究対象となった組織やグループにどのくらい依存しているのかといったことの検討がし難いものも多い．このようなデータ収集に協力する企業側の偏りもあるように見受けられ，その多くが限られた一部の企業のソフトウェア開発プロジェクトを観察したものである．そういった考慮なく得られた知見が一般化されることに若干の危惧も覚える．

第二に，得られた知見をベースとしてそれを支援ツールのデザインへと直接結びつけようとする点がある．例えば，現状の現場で開発者らが対面コミュニケーションを多用していることと，対面コミュニケーションが開発者間のコミュニケーションメディアとして最も優れている，あるいは目指すべきコミュニケーションの形態である，ということとは別のことである．開発者らは日々の限られた時間と開発環境を用いて，仕方なく，あるいは慣習として対面でしゃべっていることもあると考えられる．この問題は，tradition and transcendenceと呼ばれ実務支援のためのツールをデザインする際に常に課題となる問題である [Ehn 90]．研究の結果として得られる現状のプラクティスについての知見において，ツールや環境によらない本質が何であるのか，ということを捉えていくことが重要であろう．

4·3 コミュニケーションの分類

ソフトウェア開発者がほかの開発者とコミュニケーションを行うという際には，その目的から2種類の異なるタイプのコミュニケーションが混在している [中小路 09]．一つは，専門的な知識や情報を得るためのコミュニケーション，もう一つは，作業を調整するためのコミュニケーションである．前者はe-comm (expertise communication)，後者はc-comm (coordination communication) と呼ばれる．

e-commは，専門知識についてのコミュニケーションである．自分の作業について，ドキュメントやコードを見てもわからないことがあり，教えてほしいといった場合に行うコミュニケーションである．c-commは，調整のためのコミュニケーションであり，自分の作業と相手の作業との間でコンフリクトが起きている，あるいは起きる可能性がある場合に相談するために行うコミュニケーションである．

e-commでは，自分の作業の中で必要となる知識をもっていそうな人にその情報を求める．それに関する情報や知識が得られないと次の作業に進めないことが多い．質問して答えてもらう，という関係において，尋ねるほ

うにはメリットがあり，尋ねられたほうには教えてあげることで作業を中断し，回答してあげる，という労力がかかりむしろデメリットとなる．c-comm では，自分の作業と依存関係のある作業に関わっている人と，その作業の調整を行う．その時点で相談しておかないとコンフリクトが生じる恐れがある．作業調整はほとんどの場合双方向的であり，双方に同じようにメリットとデメリットが生じることになる．

知識コラボレーションにおけるコミュニケーションを支援するにあたっては，この2種類のタイプのコミュニケーションを区別することは非常に重要となる[Nakakoji 10a]．まず，相手との関係が対称であるか非対称であるかによって，コミュニケーションの相手を同定する際に用いる技術が異なってくる．e-comm においては，誰が，何についてどんなふうに詳しいかという情報と，誰かが誰かとやり取りした経緯があって，この人の質問であれば誰がすぐに答えてくれそうかという情報とを把握して，相手を同定する．過去に開発したプロジェクトやプログラム，過去にやり取りしたメールや共に参加したプロジェクトといった履歴情報を利用して，expertise profile や social profile を構築するという手法を用いて，コミュニケーションすべき相手が同定される[McDonald 98, McDonald 00, Mockus 02, Ribak 02, Vivacqua 00, Ye 07a]．c-comm においては，今自分の行っている作業が誰の作業に依存しているか，誰の作業から依存されているか，を把握して，依存関係のある開発者を同定し，コミュニケーションすべき相手とする．作業しているコンポーネント間の依存関係が，開発者間の依存関係を包含するとみなし，それを Dependency Network と呼ぶ[de Souza 07]．c-comm においてコミュニケーションを行う相手は，この Dependency Network に関わる開発者となる．

5. 自己記述型活動としてのソフトウェア開発

ソフトウェア開発では多様かつ膨大なデータが日々生み出される．それらのデータを蓄積し自らのタスクのために再利用するという点においてソフトウェア開発は自己記述型活動といえる．本章では，自己記述型活動としてのソフトウェア開発に着目し，記述され蓄積されるデータのアーカイビング，分析，およびその利用に関わる既存研究について解説する．

5・1 知識交換活動のアーカイブ

知識や情報をやり取りしたコミュニケーション内容をアーカイブしておき，後で別の開発者が似たような情報を探している際に，そのアーカイブした情報を見つけることができれば，コミュニケーションの必要がなくなる．技術的な課題として，後に類似した情報や知識を探し求める開発者が，どうやってそのアーカイブされた情報を得ることができるようにするか，という点にある．

やり取りしたコミュニケーション内容をアーカイブするアプローチとしては，古くは Answer Garden[Ackerman 90] がある．Answer Garden では，木構造で順に表示されていく質問をたどっていき，最終的にたどった質問に関する専門性をもった専門家から回答を得られるが，得られた回答をたどった道筋とともに保存しておくことで，後で同じような道筋をきた質問者がその回答に容易に行き着くことができるような仕組みとなっている．同様に，STeP_IN_Java[Ye 07b] でも，Java コンポーネントについて質問を行い，それに関しての回答が得られると，その回答はそのコンポーネントと関連づけられて保管される．後で別の開発者がそのコンポーネントを閲覧した際に，JavaDoc とともにそのやり取りにアクセスすることができる．

これら既存のコミュニケーションをアーカイブするアプローチの中で，回答の鮮度について考慮したものはあまり見られない．ソフトウェア開発に関わる情報は，ほかのコンポーネントや環境情報など外部の状態に依存するものも少なくない．回答した時点では正しい情報であっても，後に質問した際には，外部状況の変化から，すでにその回答が古くなってしまっているといったことも考えられる．現状は，アーカイブされた回答のタイムスタンプを見ながら開発者が判断するということになるが，今後はそのような回答の「賞味期限」のようなことも考慮していく必要があると考えられる．

5・2 データマイニングによる開発支援

構成管理システム，不具合管理システム，メーリングリストなどをはじめとする開発支援ツールは，膨大な開発履歴データを自動的に記録・蓄積している．2000年代半ばから，この存在とその利用価値に多くの研究者が気づき始めた．現在では，ソフトウェアリポジトリマイニングという研究分野として発展を続けている[Mockus 10]．ほぼすべての開発データが公開されている OSS プロジェクトのリポジトリデータを用いることで，研究者が自由な発想で試行錯誤しながら分析できるという点に加え，各種のデータマイニング技法を適用することでこれまでにない新たな知見が導き出せるのではないかという期待感が，当該分野へ多くの研究者をひき付けているようである．したがって，研究者間で研究分野としての究極のゴールが共有されているわけではなく，おのおのの研究者がおのおのの目標に従ってマイニング技法を駆使した結果を共有する場となっており，学術分野としては独特の雰囲気がある．そうした状況を踏まえ，Kagdi ら[Kagdi 07] はソフトウェアの進化に関する研究という文脈に限定して膨大な数の研究の分類を試みている．本稿は紙面の関係上，ソフトウェア開発におけるコミュニケーションデータを対象としたマイニング研究に絞って紹介する．

地理的に分散しているOSS開発者らがどのように，どのようなコミュニケーションを行っているかについて調べた研究は数多く存在するが（例えば，[Lakhani 00, Rigby 07]など），最近ではとりわけ，開発者間の社会的関係やプロジェクトの組織構造が，ソフトウェア開発における知識コラボレーションにどのような影響を与えるのかについて一歩踏み込んだ研究が行われている．

例えば，Ohiraら[Ohira 05]はsourceforge.netの約9万件のOSSプロジェクトを対象にソーシャルネットワーク分析（SNA）を試みている．ごく少数のプロジェクトのみに開発者が集中している（最大272人）一方で，過半数のプロジェクトは1人の開発者によって運営されていること，また，sourceforge.netにはべき法則が成立するいびつな社会構造が存在することを明らかにしている．その社会構造がプロジェクト間あるいは開発者間の知識共有を阻害している原因の一つであると捉え，プロジェクト横断型の知識共有を支援するツールGraphmaniaを提案している．

一方，Birdら[Bird 08]は，商用ソフトウェア開発のように明示的に規定された組織構造をもたないOSSプロジェクトを対象とする新たなSNA手法を提案し，プロジェクトの進化に伴い潜在的な組織構造，特に結びつきの強いサブグループが発現することを複数のOSSプロジェクトを対象としたケーススタディによって確認している．

さらにSarmaら[Sarma 09]は，ソフトウェア開発はアーティファクト，開発者，タスクが複雑にリンクしあいながら行われるため，開発を円滑に進めるためにそれら複雑な関係を過去から現在まで一望できることが重要であるとし，支援ツールTesseractを提案している．Tesseractは，アーティファクト間の関係（コード間の関係など），開発者間の社会的関係，タスク間の関係に加え，3種類の関係の間の関係をもリアルタイムに可視化するツールである．例えば，ある開発者があるタスクに従事している際に，同じタスクに従事し，かつ，自分の担当するモジュールに密接に関係しているモジュールを開発している（連絡や情報交換すべき）ほかの開発者は誰か，などについてインタラクティブに調べることができる．

一方，過熱気味のソフトウェアリポジトリマイニング研究者らに対して，「Let us not mine for fool's gold（見かけ倒しのものを発掘しないようにしよう）」と警鐘を鳴らす研究者もいる[Godfrey 09]．膨大な種類と量のデータをマイニングしていると，解析結果が目新しい発見であるかように思えることも少なくない．ただし，その発見が本質的に何に役に立つのか，ほかのデータセットにおいても同様の結論を得られるのか，などについて十分検証を行わなければ意味のない発見（対象データセットのみにしか適用できない発見）となってしまう危険性がある．

5・3 知識流通支援のためのデータ活用

一般に，ソフトウェア開発者がコミュニケーションすべき相手とは，尋ねたい情報を，確実にかつできるだけ早急に自分に与えてくれるような相手となる[Ye 07a]．コミュニケーションすべき相手を見つける，いわゆるノウフー（Know Who）を支援するツールは，基本的には次の三つのステップで処理が行われる．

step 1. それぞれの開発者がもっている知識や情報をあらかじめ把握しておく．

step 2. 質問者が欲している情報を把握する．

step 3. 質問者の欲する情報を有している開発者を同定する．

典型的なツールとして，例えば，ReachOut[Ribak 02]では，質問すべき相手を見つけてチャットすることができる．あらかじめ自分がどのような分野が得意かを，システムが提供するプロファイリング画面で登録しておく．開発に関して何か聞きたいことが生じたときには，ReachOutの質問画面を表示し，タイトルと質問の詳細を記入する．次に，どのような職種の人に質問したいのか（例えば，顧客側のアーキテクト，サーバのアーキテクト，など）を選択すると，システムが選別したその質問を受け取るべき相手の画面に，その質問が表示され，その質問に答えるか否かを選択するよう促される．答えることにすると，質問者との間でのチャットウィンドウが表示される，といった具合である．

実際の開発現場では，step 1において，開発者が自らの専門性を登録しておくというのは難しい．また，登録してある情報が古くなってしまう可能性も高い．そこで，開発者が過去に関わった作業の情報から，それぞれの開発者の専門性を類推するアプローチが出てきた．例えば，開発者が関わったコンポーネントの版管理システムの記録[McDonald 00]，プロジェクトの開発履歴[Mockus 02]，各プログラマが過去に開発したプログラム[Vivacqua 00, Ye 07a]などを利用したツールが開発された．

また，step 3で，専門性が一致する開発者を同定できたからといって，その開発者に質問を投げると必ずしも即時に回答してくれるとは限らないことが指摘され始めた．そこで

step 4. 質問者にすぐに回答してくれそうな開発者を選別する．

という段階を踏むツールが開発された．Illichが「人間を知識ソースとして使うには，まずはその人間が協力してくれる必要がある」[Illich 71]と述べているように，知識や情報を有しているだけでは知識コミュニケーションは成り立たない[Reder 88]．

Step 4を考慮するために，今この瞬間に答えてあげられるかどうかを開発者が自分で示しておく（willing to answerであることを示す）[Ribak 02]，現在作業が立て込んでいないかどうかをチェックする[McDonald 98]，質問者と似たレベルの専門度であるかどうかを調

べる [Vivacqua 00]，質問者と以前にもメールでやり取りをしたことがあるかどうかを調べる [Ye 07a] などといった手法を組み込んだツールも開発されている．

6. 知性の統合としてのソフトウェア開発

ソフトウェア開発は，個々の創造的な開発者の知性を統合することで一つのアーティファクト（ソフトウェア）を生み出す活動と捉えることができる．その過程においては，個々の開発者が相互に関わり合いながら開発を行うという社会的な側面と，創出するソフトウェアシステムがほかの人々によって利用されるという社会的な側面とがある．個々の知性は，このような社会的過程を経てソフトウェアシステムというアーティファクトへと結晶する．本章では，このような人間同士の関わりに着目し，知性や創造性の統合活動としてのソフトウェア開発を支援する技術と，その示唆するところを解説する．

6·1 オープンネゴシエーション

近年盛んに行われているソフトウェア開発の外部委託，いわゆるアウトソーシングは，主流のインドや中国からさらに安価な労働力を求めて，ベトナムやタイなどへその適用範囲を広げつつある [総務省 07]．ソフトウェアの分散開発は，開発コスト削減のメリットが享受できる半面，分散した拠点間でのアウェアネスの欠如により，しばしば品質の低下や納期の遅延などの問題を引き起こすことが知られている [Carmel 99, Herbsleb 03, Karolak 99]．

開発者が一つの開発拠点に存在する通常のソフトウェア開発では，開発状況や開発者の作業状況などのアウェアネスが確保されており，相手の状況を把握したうえで意識的・無意識的に仕事を依頼したり遠慮したりしている．こうした相手の状況が見える場での明示的·非明示的な交渉，すなわちオープンネゴシエーションは，分散開発においては非常に困難となる．特に，開発者が世界各地に点在しているOSS開発では，開発拠点が多数（無数に）存在するため，タスクの割当ては開発者の自主性に委ねられており，作業の重複が生じることも珍しくない．

そのため，分散開発におけるアウェアネスの向上を支援することを目的として，ソフトウェア開発の可視化に関する研究が盛んに行われている [Storey 05]．例えばAugar[Froehlich 04] は，ソースコードの変更履歴を可視化するツールである．どの開発者がソースコードのどの部分を変更したのかがファイル単位で一目で把握できるため，コードを変更する必要が生じた場合に誰に問い合わせるべきかがすぐにわかる．ACTION（Awareness Communication Tool for Open Negotiation）[伊原 10] は，ソースコードの変更履歴を各開発者の活動履歴および地理情報と合わせて可視化するツールである．不具合修正を行うために問い合わせるべき開発者（直近にソースコードを変更した開発者など）が1日の中でどの時間帯に最も活動しているかを把握することができるため，分散環境下にあってもオープンネゴシエーションが支援される．

6·2 アウェアネス支援ツールとその思想

アウェアネス情報を表示する多くのツールのデザイン思想の根底にあるのは，個々の開発者が行っている作業間でコンフリクトが生じていることを知らずに，双方の作業を統合した時点でバグが発生したり，どちらかの作業がむだになってしまったりすることを事前に防ぐと同時に，できるだけ明示的なコミュニケーションをしなくて済むように，例えば自分と被る作業をしている開発者の存在に気づいたら別の作業を先に済ますといったようにして，コーディネーションを行うといったことを目指すものである．

Palantír [Sarma 07] は，開発環境 Eclipse を用いて開発中のコンポーネントが，ほかのコンポーネントと間接的にコンフリクトをきたす可能性のあることを Eclipse 中にそれとなく表示するツールである．例えば，開発中のコンポーネントが参照していたり，継承関係のあるコンポーネントを間接的なコンフリクトとして検出し，同時にそのコンポーネントのオーナ（開発者）を提示する．間接的なコンフリクトを開発者に提示することで，コンフリクトを解消するべくほかの開発者と作業のコーディネーションを行うことを促す．

FASTDash[Biehl 07] は，小規模開発チーム（5～20名）内での作業アウェアネスを支援するツールである．大画面ディスプレイをチームメンバが閲覧しながら開発を進めるといった状況を想定している．各開発者のファイル編集作業やデバッグ作業の状況をはじめとして，各開発者が現在どのような作業を行っているのかがメンバ全員で共有できるようリアルタイムに情報を更新して表示する．アウェアネスの支援により，直接的なコミュニケーションの削減や作業のコンフリクトの予防を期待することができる．

6·3 非物質労働とソフトウェア開発

社会学者 Maurizio Lazzarato は，製品に内包されている情報的または文化的コンテンツを創出する作業を非物質労働（Immaterial Labor）と定義した．音楽やファッションなどの産業で，製品として成り立つのはその製品の物質的な存在ではなく，その製品が内包している情報または文化的なコンテンツである．これらの製品は消費者の情報的かつ情緒的なニーズを満足させることを目的としている [Lazzarato 96]．ソフトウェアを開発する作業も非物質労働と考えることができる [Ye 10]．

非物質労働としてのソフトウェア開発においてつくられた製品としてのソフトウェアシステムは，消費者の実在的な需要を満たす物質的な製品と違って，抽象物とし

ての需要または想像を実在化させることを可能にするものである．この点において貨幣の「実在的抽象」のような存在に近いと考えられる．生産したものを流通させるフォード型の生産方式から，流通させるものを生産するトヨタ型の生産方式へと，その図式に逆転が生じたように，非物質労働では需要と生産が逆転する．消費者の需要に応じて製作するのではなく，消費者に新しい体験を提供したうえで次なる需要とニーズを開拓するためのイノベーションが求められる．

非物質的な製品としてのソフトウェアシステムが出来上がる時点では，価値は存在しない．ユーザに利用されているという社会化（Socialization）のプロセスを経て初めて，その価値が生まれる．その意味で，ソフトウェア開発は，新たな社会関係を創出するプロセスであると考えるべきといえる．このような特徴は，Wikipediaや Facebookといったシステムでは特に顕著であるが，SaaS（Software as a Searvice）が代表する商用ソフトウェアでも同様な方向に向かいつつある．

イノベーションと社会関係に基礎を置いたソフトウェア開発の主な生産資源は，個々の開発者の知識と知性と主体性（Subjectivity）とである．これに関わる開発タスク，開発プロセスにもまた目に見えない工夫が含まれており，詳細に規定することは困難である．開発プロセスも多様で，反復するものが少なく，開発者にもマネジャーにも常に予測不可能な局面と対峙する能力が求められる．個々の開発者が自ら問題を特定し，特定した問題に対して多数の解が存在するときに，自らの知識と主体性によって解を決定していく．ソフトウェアシステムの全体の品質と価値がこれらの一個一個の自主的かつ主観的な決定により決められる．

知識と知性と主体性がコアになり，生産の目的が問題の特定と解決そしてイノベーションへと社会関係を創出するため，ソフトウェア開発は，（会社の中だけのような）限りのある時間と空間のみで行われるのではない．むしろその範囲は，生活時間の全体にまで拡大する．ソフトウェアシステムは，個々の開発者が決まったロールを果たすことにより構築されるものではなく，社会生活で培ってきた知性と個性を生かした認知的な主体としてプロジェクトに参加し，ほかの認知的な主体と自発的に協働しながらつくられるものである．

7. おわりに

今回，知識コラボレーションの側面からソフトウェア開発に関わる研究を解説するにあたり，ソフトウェア開発を，知識共創活動，新陳代謝プロセス，コミュニケーション集約型活動，自己記述型活動，および知性の統合という，五種類のものとして見立ててみた．本稿を通じて論じたように，ソフトウェア開発における重要な側面の多くは，創造的であると同時に社会的な作業であり，知識集約型であると同時にコミュニケーション集約型の活動であり，需要や要求に基づくと同時に新たな価値や体験を提供する過程である．

1973年，当時のソフトウェア産業振興協会の会長であった服部 正氏が，次のように述べている．「ソフトウェアを，プログラムをつくる仕事と理解している人が多い．工場における生産のごとく，ソフトウェアを生産する．そのような理解がある限り，ソフトウェアの価値は，それに投じられた頭脳労力の総和としてしか評価されない．それをつくる動機，それに対する工夫，そのような無形の価値をどのように主張し，どのような納得を得られるかという点に，我々の将来はかかっている」．40年近くの時を経た今，ソフトウェアを開発するという営為を取り巻く状況に大差が見られないことは，非常に残念なことであると思う．

ソフトウェア開発は，開発対象となるソフトウェアというアーティファクトの規模と複雑さが増すにつれて，開発者の作業をいかにorchestrateするかに主眼が置かれるようになっていった．ソフトウェアを工業製品とみなし，工場のメタファが導入され，開発作業の生産性と品質の向上が大きな目的とされた．開発作業をいかに効果的に開発者らに分配するかといったプロジェクト管理や，ソフトウェア開発過程を工場の製造ラインと見立てそのプロセスを管理するような手法などが，ソフトウェア開発の支援であり開発支援の研究であると捉えられていた時期もあった．

しかしながら，ソフトウェア開発を自動化できていないということは，それが人的要因に大きく深く関わることの現れであろう．我々は，人間中心のソフトウェア開発という，一見したところ当たり前に感じられることを，声高に言うような時機がきているように感じている．このことは，1980年代にCASE（Computer-Aided Software Engineering）という言葉が出てきた状況と似ているのかもしれない．コンピュータを使ってソフトウェア開発を支援しようとわざわざそれをいう必要があったくらい，当時ソフトウェア工学とコンピュータ支援とは無縁であった．ソフトウェア開発支援研究は，現在においても，人的要因や知性が十分に着目されているとは言い難い．

本稿で解説した，ソフトウェア開発における知識コラボレーションに関連する研究に通底する考えやアプローチが，ソフトウェア開発支援のための取組み，ソフトウェア開発の本質的な理解，ひいてはソフトウェアそのものの展望へとつながっていく一助となることを願っている．

◇ 参 考 文 献 ◇

[Ackerman 90] Ackerman, M. S. and Malone, T. W.: Answer Garden: A tool for growing organizational memory, *Proc. ACM SIGOIS and IEEE CS TC-OA Conf. Office Information Systems* (*COCS' 90*), pp. 31-39 (1990)

[Augustin 02] Augustin, L., Bressler, D. and Smith, G.: Accelerating software development through collaboration, *Proc. 24th Int. Conf. Software Engineering* (*ICSE' 02*), pp. 559-563 (2002)

[Beck 99] Beck, K.: *Extreme Programming Explained: Embrace Change*, Addison-Wesley Professional, Boston, MA (1999)

[Biehl 07] Biehl, J. T., Czerwinski, M., Smith, G. and Robertson, G. G.: FASTDash: A visual dashboard for fostering awareness in software teams, *Proc. SIGCHI Conf. Human Factors in Computing Systems* (*CHI' 07*), pp. 1313-1322 (2007)

[Bird 07] Bird, C., Gourley, A., Devanbu, P., Swaminathan, A. and Hsu, G.: Open borders? Immigration in open source projects, *Proc. 4th Int. Workshop on Mining Software Repositories* (*MSR' 07*), p. 6 (2007)

[Bird 08] Bird, C., Pattison, D., D'Souza, R., Filkov, V. and Devanbu, P.: Latent social structure in open source projects, *Proc. 16th ACM SIGSOFT Int. Symp. Foundations of Software Engineering* (*FSE' 08*), pp. 24-35 (2008)

[Brooks 75] Brooks, F. P.: *The Mythical Man-Month*, Addison-Wesley Professional, Upper Saddle River, NJ (1975)

[Carmel 99] Carmel, E.: *Global Software Teams: Colloborating Across Borders and Time Zones*, Prentice Hall, Upper Saddle River, NJ (1999)

[Cataldo 08] Cataldo, M., Herbsleb, J. D. and Carley, K. M.: Socio-technical congruence: A framework for assessing the impact of technical and work dependencies on software development productivity, *Proc. 2nd ACM-IEEE Int. Symp. Empirical Software Engineering and Measurement* (*ESEM' 08*), pp. 2-11 (2008)

[Conway 68] Conway, M.: How do committees invent, *Datamation*, Vol. 14, No. 4, pp. 28-31 (1968)

[Curtis 88] Curtis, B., Krasner, H. and Iscoe, N.: A field study of the software design process for large systems, *Commun. ACM*, Vol. 31, No. 11, pp. 1268-1287 (1988)

[Damian 07] Damian, D., Izquierdo, L., Singer, J. and Kwan, I.: Awareness in the wild: Why communication breakdowns occur, *Proc. Int. Conf. Global Software Engineering* (*ICGSE' 07*), pp. 81-90 (2007)

[de Souza 04] de Souza, C. R. B., Redmiles, D., Cheng, L.-T., Millen, D. and Patterson, J.: How a good software practice thwarts collaboration: The multiple roles of APIs in software development, *Proc. 12th ACM SIGSOFT 12th Int. Symp. Foundations of Software Engineering* (*FSE' 04*), pp. 221-230 (2004)

[de Souza 07] de Souza, C. R., Quirk, S., Trainer, E. and Redmiles, D. F.: Supporting collaborative software development through the visualization of socio-technical dependencies, *Proc. 2007 Int. ACM Conf. Supporting Group Work* (*GROUP' 07*), pp. 147-156 (2007)

[Ehn 90] Ehn, P.: *Work-oriented Design of Computer Artifacts*, L. Erlbaum Associates Inc., Hillsdale, NJ (1990)

[Fairley 85] Fairley, R.: *Software Engineering Concepts*, McGraw-Hill, Inc., New York, NY (1985)

[Fischer 00] Fischer, G.: Symmetry of ignorance, social creativity, and meta-design, *Knowledge-Based Systems J.*, Vol. 13, No. 7-8, pp. 527-537 (2000)

[Froehlich 04] Froehlich, J. and Dourish, P.: Unifying artifacts and activities in a visual tool for distributed software development teams, *Proc. 26th Int. Conf. Software Engineering* (*ICSE' 04*), pp. 387-396 (2004)

[Fujita 10] Fujita, S., Ohira, M., Ihara, A. and Matsumoto, K.: An analysis of committers toward improving the patch review process in OSS development, *Proc. 21st IEEE Int. Symp. Software Reliability Engineering* (*ISSRE' 10*), pp. 369-374 (2010)

[Giaccardi 05] Giaccardi, E. and Fogli, D.: Beyond usability evaluation in meta-design: A socio-technical approach, Technical report, manuscript (2005)

[Godfrey 09] Godfrey, M. W., Hassan, A. E., Herbsleb, J., Murphy, G. C., Robillard, M., Devanbu, P., Mockus, A., Perry, D. E. and Notkin, D.: Future of mining software archives: A roundtable, *IEEE Software*, Vol. 26, pp. 67-70 (2009)

[Herbsleb 03] Herbsleb, J. D. and Mockus, A.: An empirical study of speed and communication in globally distributed software development, *IEEE Trans. Software Engineering* (*TSE*), Vol. 29, No. 6, pp. 481-494 (2003)

[Hippel 03] Hippel, E. and Krogh, G.: Open source software and the private-collective innovation model, *Organization Science*, Vol. 14, No. 2, pp. 209-223 (2003)

[Humphrey 89] Humphrey, W. S.: *Managing the Software Process*, Addison-Wesley Professional, Boston, MA (1989)

[Hutchins 96] Hutchins, E.: *Cognition in the Wild*, The MIT Press, Cambridge, MA (1996)

[伊原 10] 伊原彰紀，山本瑞起，大平雅雄，松本健一：OSS開発における保守対応の効率化のためのアウェアネス支援システム，情処マルチメディア，分散，協調とモバイルシンポジウム論文集 (DICOMO' 10), pp. 1620-1629 (2010)

[Illich 71] Illich, I.: *Deschooling Society*, Harper & Row, New York, NY (1971)

[Jensen 07] Jensen, C. and Scacchi, W.: Role migration and advancement processes in OSSD projects: a comparative case study, *Proc. 29th Int. Conf. Software Engineering* (*ICSE' 07*), pp. 364-374 (2007)

[Kagdi 07] Kagdi, H., Collard, M. L. and Maletic, J. I.: A survey and taxonomy of approaches for mining software repositories in the context of software evolution, *J. Software Maintenance and Evolution: Research and Practice*, Vol. 19, No. 2, pp. 77-131 (2007)

[Karolak 99] Karolak, D. W.: *Global Software Development: Managing Virtual Teams and Environments*, Wiley-IEEE Computer Society Press, Los Alamitos, CA (1999)

[Ko 07] Ko, A. J., DeLine, R. and Venolia, G.: Information needs in collocated software development teams, *Proc. 29th Int. Conf. Software Engineering* (*ICSE' 07*), pp. 344-353 (2007)

[Lakhani 00] Lakhani, K. and Hippel, von E.: How open source software works: "free" user-to-user assistance, *Research Policy*, Vol. 32, pp. 923-943 (2000)

[Lazzarato 96] Lazzarato, M.: Immaterial labour, in Virno, P. and Hardt, M. eds., *Radical thought in Italy: A Potential Politics*, pp. 133-147, University of Minnesota Press (1996)

[Little 07] Little, G. and Miller, R. C.: Keyword programming in java, *Proc. 22nd IEEE/ACM Int. Conf. on Automated Software Engineering* (*ASE' 07*), pp. 84-93 (2007)

[McDonald 98] McDonald, D. W. and Ackerman, M. S.: Just talk to me: A field study of expertise location, *Proc. 1998 ACM Conf. Computer Supported Cooperative Work* (*CSCW' 98*), pp. 315-324 (1998)

[McDonald 00] McDonald, D. W. and Ackerman, M. S.: Expertise recommender: A flexible recommendation system and architecture, *Proc. 2000 ACM Conf. Computer Supported Cooperative Work* (*CSCW' 00*), pp. 231-240 (2000)

[Mockus 02] Mockus, A. and Herbsleb, J. D.: Expertise browser: a quantitative approach to identifying expertise, *Proc. 24th Int. Conf. Software Engineering* (*ICSE' 02*), pp. 503-512 (2002)

[Mockus 10] Mockus, A., ed.: *Proc. 7th IEEE Working Conf. Mining Software Repositories* (*MSR 2010*) (2010)

[Nagappan 08] Nagappan, N., Murphy, B. and Basili, V.: The influence of organizational structure on software quality: An empirical case study, *Proc. 30th Int. Conf. Software Engineering* (*ICSE' 08*), pp. 521-530 (2008)

[Nakakoji 96] Nakakoji, K.: Beyond language translation: Crossing the cultural divide, *IEEE Software*, Vol. 13, No. 6, pp. 42-46 (1996)

[Nakakoji 00] Nakakoji, K., Ohira, M. and Yamamoto, Y.:

Computational support for collective creativity, *Knowledge-Based Systems J.*, Vol. 13, No. 7-8, pp. 451-458 (2000)

[Nakakoji 06] Nakakoji, K.: Supporting software development as collective creative knowledge work, *Proc. 2nd Int. Workshop on Supporting Knowledge Collaboration in Software Development* (*KCSD' 06*), pp. 1-8 (2006)

[中小路 07] 中小路久美代：知識コミュニティによる持続的価値を有するアーティファクトの構築，機械の研究，Vol. 59, pp. 141-148 (2007)

[中小路 09] 中小路久美代，イェ ユンウェン，山本恭裕：ソフトウェア開発における知識コミュニケーションのためのインタラクションデザイン，人工知能学会全国大会 (2009)

[Nakakoji 10a] Nakakoji, K., Ye, Y. and Yamamoto, Y.: Comparison of coordination communication and expertise communication in software development: Motives, characteristics, and needs, in Nakakoji, K., Murakami, Y. and McCready, E., eds., *New Frontiers in Artificial Intelligence*, LNAI6284, pp. 147-155, Springer-Verlag (2010)

[Nakakoji 10b] Nakakoji, K., Ye, Y. and Yamamoto, Y.: Supporting expertise communication in developer-centered collaborative software development environments, in Mistrík, I., Grundy, J., Hoek, A. and Whitehead, J. eds., *Collaborative software engineering*, chapter 11, pp. 152-169, Springer-Verlag, New York, NY (2010)

[Ohira 05] Ohira, M., Ohsugi, N., Ohoka, T. and Matsumoto, K.: Accelerating cross-project knowledge collaboration using collaborative filtering and social networks, *Proc. 2005 Int. Workshop on Mining Software Repositories* (*MSR 2005*), pp. 1-5 (2005)

[大平 00] 大平雅雄，山本恭裕，蔵川 圭，中小路久美代：EVIDII: 差異の可視化による相互理解支援システム，情処学論，Vol. 41, pp. 2814-2826 (2000)

[大西 02] 大西 淳，郷健太郎：要求工学：プロセスと環境トラック（ソフトウェアテクノロジー），共立出版，東京 (2002)

[Osterweil 87] Osterweil, L.: Software processes are software too, *Proc. 9th Int. Conf. Software Engineering* (*ICSE' 87*), pp. 2-13 (1987)

[Pangaro 00] Pangaro, P.: Participative Systems: An Immodest Proposal to Measure the Relative Values of Software Demos, Available at http://www.pangaro.com/ (2000)

[Reder 88] Reder, S. and Schwab, R. G.: The communicative economy of the workgroup: Multi-channel genres of communication, *Proc. 1988 ACM Conf. Computer-Supported Cooperative Work* (*CSCW' 88*), pp. 354-368 (1988)

[Ribak 02] Ribak, A., Jacovi, M. and Soroka, V.: Ask before you search: Peer support and community building with reachout, *Proc. 2002 ACM Conf. Computer Supported Cooperative Work* (*CSCW' 02*), pp. 126-135 (2002)

[Rigby 07] Rigby, P. C. and Hassan, A. E.: What can OSS mailing lists tell us? a preliminary psychometric text analysis of the Apache developer mailing list, *Proc. 4th Int. Workshop on Mining Software Repositories* (*MSR' 07*), p. 23 (2007)

[Sarma 07] Sarma, A., Bortis, G. and Hoek, van der A.: Towards supporting awareness of indirect conflicts across software configuration management workspaces, *Proc. 22nd IEEE/ACM Int. Conf. Automated Software Engineering* (*ASE' 07*), pp. 94-103 (2007)

[Sarma 09] Sarma, A., Maccherone, L., Wagstrom, P. and Herbsleb, J.: Tesseract: Interactive visual exploration of socio-technical relationships in software development, *Proc. 31st Int. Conf. on Software Engineering* (*ICSE' 09*), pp. 23-33 (2009)

[Shneiderman 80] Shneiderman, B.: *Software Psychology: Human Factors in Computer and Information Systems*, Winthrop Publishers, Cambridge, MA (1980)

[Singer 97] Singer, J., Lethbridge, T., Vinson, N. and Anquetil, N.: An examination of software engineering work practices, *Proc. 1997 Conf. Centre for Advanced Studies on Collaborative research* (*CASCON' 97*), p. 21 (1997)

[Soloway 86] Soloway, E. and Ehrlich, K.: Empirical studies of programming knowledge, *Readings in Artificial Intelligence and Software Engineering*, pp. 507-521, Morgan Kaufmann Publishers Inc. (1986)

[Sommerville 97] Sommerville, I. and Sawyer, P.: *Requirements Engineering: A Good Practice Guide*, John Wiley & Sons, New York, NY (1997)

[総務省 07] 総務省白書資料：オフショアリングの進展とその影響に関する調査研究 (2007)

[Storey 05] Storey, M.-A. D., Cubranic, D. and German, D. M.: On the use of visualization to support awareness of human activities in software development: A survey and a framework, *Proc. 2005 ACM Symp. Software Visualization* (*SoftViz' 05*), pp. 193-202 (2005)

[Tomayko 04] Tomayko, J. and Hazzan, O.: *Human Aspects of Software Engineering* (*Electrical and Computer Engineering Series*), Charles River Media, Rockland, MA (2004)

[遠 07] 遠 明憲，大平雅雄，森崎修司，松本健一：異文化コラボレーションとしてのソフトウェア要求抽出の支援，信学論 (D), Vol. J99-D, No. 12, pp. 3151-3160 (2007)

[UW MSR Institute 2007 07] UW MSR Institute 2007, : 2007 Summer Institute on the Human Side of Software Development, http://www.cs.washington.edu/mssi/2007/ (2007)

[Vivacqua 00] Vivacqua, A. and Lieberman, H.: Agents to assist in finding help, *Proc. SIGCHI Conf. Human Factors in Computing Systems* (*CHI' 00*), pp. 65-72 (2000)

[Wiegers 99] Wiegers, K. E.: *Software Requirements*, Microsoft Press, Redmond, WA (1999)

[山本 10] 山本修一郎，神戸雅一：CMC が拓く知識流通ネットワーク，人工知能学会誌，Vol. 25, No. 5, pp. 715-725 (2010)

[Ye 05] Ye, Y. and Fischer, G.: Reuse-conducive development environments, *Automated Software Engineering*, Vol. 12, No. 2, pp. 199-235 (2005)

[Ye 07a] Ye, Y., Yamamoto, Y. and Nakakoji, K.: A sociotechnical framework for supporting programmers, *Proc.6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on Foundations of Software Engineering* (*ESEC-FSE' 07*), pp. 351-360 (2007)

[Ye 07b] Ye, Y., Yamamoto, Y., Nakakoji, K., Nishinaka, Y. and Asada, M.: Searching the library and asking the peers: Learning to use Java APIs on demand, *Proc. 5th Int. Symp. Principles and Practice of Programming in Java* (*PPPJ' 07*), pp. 41-50 (2007)

[Ye 08a] Ye, Y.: Measuring site coupling is distribued software development, *Proc. Software Engineering Approaches for Offshore and Outsourced Development* (*SEAFOOD'08*) (2008)

[イェ 08b] イェ ユンウェン（葉 雲文），中小路久美代，山本恭裕：知識流通における距離と近接性，人工知能学会，第 3 回知識流通ネットワーク研究会 (2008)

[Ye 10] Ye, Y., Nakakoji, K., Yamamoto, Y. and Kishida, K.: Through the looking glass of immaterial labor, *Proc. 2010 FSE/SDP Workshop on the Future of Software Engineering Research*, pp.433-438 (2010)

[Zhou 10] Zhou, M. and Mockus, A.: Developer fluency: Achieving true mastery in software projects, *Proc. 8th ACM SIGSOFT Int. Symp. Foundations of Software Engineering* (*FSE' 10*), pp.137-146 (2010)

2010 年 11 月 1 日　受理

著者紹介

大平 雅雄

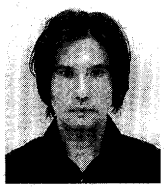
奈良先端科学技術大学院大学情報科学研究科助教．1998年京都工芸繊維大学工芸学部電子情報工学科卒業．2003年奈良先端科学技術大学院大学情報科学研究科博士課程修了．博士（工学）．同年奈良先端科学技術大学院大学産学官連携研究員．2004年同大学情報科学研究科助手（職名変更により2007年から現職）．HCI/CSCW，特にオープンコラボレーション支援環境の研究に従事．電子情報通信学会，情報処理学会，ヒューマンインタフェース学会，情報社会学会，ACM各会員．

イェ ユン ウェン（葉 雲文）（正会員）

（株）SRA先端技術研究所シニア・リサーチャー．1987年中国復旦大学コンピュータサイエンス学部卒業．1990年同大学院修士課程修了後，同大学助手．1993年SRA入社．2001年米国コロラド大学コンピュータサイエンス学部よりPh. D. 取得．2001～08年までコロラド大学客員研究員．2008年より現職．ソフトウェア開発環境，再利用，分散開発，検索エンジン，知識共有に関する研究および開発を行っている．

中小路 久美代（正会員）

（株）SRA先端技術研究所リサーチディレクター．1986年大阪大学基礎工学部情報工学科卒業．同年SRA入社．1993年米国コロラド大学よりPh.D.1994年同大学認知科学研究所客員助教授．1995年奈良先端科学技術大学院大学客員助教授．2002年東京大学先端科学技術研究センター特任教授．2010年より現職．専門は，ナレッジインタラクションデザイン，知的創造活動支援．ACM，日本認知科学会，情報処理学会など各会員．



山本 恭裕（正会員）

東京工業大学精密工学研究所特任准教授．1996年京都大学工学部情報工学科卒業．2001年奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．博士（工学）．2001年日本学術振興会特別研究員（PD）．2002年JSTさきがけ研究21領域グループメンバ．2003年東京大学先端科学技術研究センター特任研究員．2004年同特任助教授．2007年同特任准教授（職名変更）．2010年より現職．専門はソフトウェアのインタラクティビティのデザイン．