2007－HCI－126 (6)
2007／11／12

知的創造作業のためのソシオテクニカル情報環境

－ 共創的知識創造活動としてのソフトウェア開発支援 －

中小路 久美代[1,2]

(1) 東京大学先端科学技術研究センター
(2) 株式会社SRA先端技術研究所

概要: 知的創造作業における情報源としての＜ヒト＞の同定にあたっては，その人が必要な情報を有していることに加えて，適切な情報をタイムリーに提供してくれることが求められる．提案するソシオテクニカル環境は，情報を必要とする側からは＜モノ＞と＜ヒト＞を情報源としてシームレスに利用できることを，情報を提供する側からは負担を感じることなく協働に関われることを目指すものである．本論では，プログラマ個々人の専門性に加えて社会的関係を考慮しながら，関連する＜ヒト＞から成るアドホックなナレッジコミュニティーを動的に形成し，ソフトウェア開発におけるナレッジコラボレーションを支援する我々のアプローチを紹介し，考慮すべき要因について論じる．

A Socio-Technical Environment for Collective Creative Knowledge Work

Kumiyo Nakakoji[1,2]

(1) KID Laboratory, RCAST, University of Tokyo
(2) SRA Key Technology Laboratory Inc., Japan

Abstract: We propose a socio-technical environment where a knowledge practitioner can seamlessly access both artifacts and peers as knowledge resources relevant to his/her task at hand. Different from help-desks, such peers typically have their own tasks to perform and are not there to help information seekers. We thereby need to explore the "right" balance between the needs of information seekers and those of information providers. This paper presents our approach, which uses expertise profiles and social relationships to dynamically formulate an "ad-hoc knowledge community" for a programmer asking peers for help to solve his/her current task.

1 はじめに

知的創造活動の支援を目的として，ユーザが関わる，刻々と変化するタスクの状況に関連すると思われる知識や情報を，システムが能動的に提示するようなツールを開発し，その有効性を確認してきた [Fischer et al. 93]．このようなツールにより，検索クエリを適切に書けなかったり，情報の必要性や関連する情報の存在に気づいていなかったりした場合にも，ユーザは，システムが有する情報や知識を有効に活用することができるようになる．

作られたアーティファクトやドキュメントといった＜モノ＞のみでなく，プロジェクトメンバーやそのタスクのエキスパートといった＜ヒト＞も，情報源として重要であることが指摘されている [Nardi 00]．これを踏まえて，AnswerGarden [Ackerman, Halverson 04]やExpertBrowser [Mockus, Herbsleb 02]といった，エキスパートを探してくる，エキスパートに聞く，ための仕

組みが研究されている.

我々は，知的創造活動に携わる個人から見た場合，使う情報源が＜モノ＞であるのか＜ヒト＞であるのかの違いを意識せずにその情報を利用できることが望ましいと考えている．ドキュメントなどの＜モノ＞を検索し，順序づけし，能動的に提示するのと同様に，支援ツールは，関連する知識や情報を有する＜ヒト＞を，検索し，順序づけし，提示すべきであるという考え方である．

しかしながら，Illich が，教育における教師の役割について言及しているように，情報源としての＜ヒト＞は，＜モノ＞とは異なる．

> *"A thing is available at the bidding of the user—or could be—whereas a person formally becomes a skill resource only when he consents to do so, and he can also restrict time, place, and method as he chooses ."* [Illich 71]

＜ヒト＞を情報源として利用するためには，その人が，情報や知識を提供してくれることが必要となる．さらに，資源としての＜ヒト＞の利用は有限である．同じドキュメントを 10 人のユーザが同時に見ることはできるが，ひとりの人間が同時に 10 人のリクエストに答える，ということは不可能である．

それに加えて，我々が想定しているような知的創造活動において相互に知識交換し助け合うような状況における，情報源としての＜ヒト＞にとっては，情報や知識を提供することが，主たる仕事ではない，という点がある．このことは，ヘルプデスクのエキスパートが，情報を聞かれたら提供する，ということが前提としてあることと対照的である[Ackerman, Halverson 04]．各人が自分のタスクをそれぞれ抱えている状況で，時として，同僚やプロジェクトメンバーに，情報を求めたり，また，求められたりする．忙しいのに情報を求められて困ったり，逆に情報をなかなかもらえずに困ったり，などといった状況は，我々が日常的に経験していることであろう．

われわれは，ソフトウェア開発をナレッジインテンシブな知的創造作業とみなし[Ye et al. 08]，それを支援するツールに必要となるインタラクションデザイン要件[中小路，山本 06]についての研究をおこなっている．ソフトウェア開発の中でも特にプログラミングは，上述の課題に対処することが非常に重要となると考えている [Ye et al. 07a]．プログラミング作業そのものは，極めて個人的な認知負荷の高い作業であり，プログラマの多くは，既存のソースコードやドキュメント，Web ページなどの多くの外部情報を必要としながら，静かにひとりで没頭できる環境を求めているといわれている[De Marco, Lister 99]．しかしながら，プログラマの多くは，多くの時間をコミュニケーションに割いており，他のプロジェクトメンバーに情報を求めることが，プログラミング作業には不可欠であるとされている [LaToza et al. 06]．コミュニケーションを求められて作業が中断されることが問題だと多くのプログラマが考えているものの，タイムリーな情報提供は，プロジェクトを進めるためには必要不可欠であり仕方がないと考えている様子が報告されている [Ko et al. 07]．

このように，ソフトウェア開発プロジェクトといった組織においておこなわれる知的創造作業では，個々人は自分の作業に集中したいものの，他のメンバーから情報を得なければならない場面は日常的にあり，と同時に自分が他のメンバーの助けに応じて答えることも必要となる，といった状況があると考えられる．すなわち，同じ人が，ある情報に関する助け求めつつ，別の情報に関して助けを求められる状況にある．知識が不均衡に存在している場において，助けなければプロジェクトは進まないが，助けてばかりでは自分の仕事が進まない，というジレンマに陥る．と同時に，聞く，答える，といったコミュニケーションは，人間の社会的活動の基本であり，この人に聞いても良いかどうかわからない，聞くのは格好が悪い，答えないと悪い，といった社会心理学的な要因が深く関わってくる．

我々が提案するソシオテクニカル環境は，情報を必要とする側からは＜モノ＞と＜ヒト＞を情報源としてシームレスに利用できることを，情報を提供する側からは負担を感じることなく協働に関われることを目指すものである [Ye et al. 07a]．情報を探す側(asker/helpee) にとっては，タイムリーに適切な情報を得られることが最も重要である．情報を求められる側(answerer/helper)にとっては，他者からの情報提供の要請による仕事の中断をできる限り少なくすることが重要となる．そのために我々がとったアプローチ

には，下記の二点の特徴がある．

(1) 何の情報を誰が求めているかによって，専門性と社会的関係の双方を考慮しながら，エキスパートの同定をおこなう

(2) 情報を求める側，求められる側双方のコミュニケーションコストを減らすことを目的として，同定されたエキスパートからなる，アドホックなナレッジコミュニティーを動的に形成する

続く2章では，知識共創活動としてのソフトウェア開発について簡単に説明する．3章では，我々が構築してきた，個々人の専門性に加えて社会的関係を考慮しながら，関連する＜ヒト＞から成るアドホックなナレッジコミュニティーを動的に形成しJavaプログラマのナレッジコラボレーションを支援するプロトタイプシステムSTeP_IN_Java環境を紹介する．4章では，そのようなアドホックなナレッジコミュニティーを形成するにあたり考慮すべき要因について議論をおこない，今後研究の進むべき方向について論じる．

2 共創的知識創造活動としてのソフトウェア開発

ソフトウェア開発を，クリエイティブな知識共創活動としてみなすことは，ソフトウェア工学分野においては実は長い間あまりおこなわれていなかった．ソフトウェア工学における手法や技術の多くは，ソフトウェア開発プロジェクトにおける多人数の作業をcoordinateし管理することを目的としてきた．開発プロセスを改善することで開発効率をあげるというCMM (Capability Maturity Model)のアプローチを，グループや個人に適用するなどといった流れはあったものの，プロセスをプログラミングする[Osterweil 87]という表現が示すように，開発者は処理をおこなうプロセッサとしてみなされていた．そこでは，個々の開発者(プログラマ)の認知的過程や創造性といった側面は重要ではなく，いかに決められた工程と工数に従って「正しく」作業をおこなうか，ということにのみ関心が注がれていたように思う．Psychology of Programmingという研究分野はあったものの，その関心の中心は，エキスパートとノービスの振る舞いの違いにあった [Soloway et al. 84]．ソフトウェア開発における社会性に着目した研究の多くは，開発者と顧客との間の業務文化の違いやコミュニケーションの破綻といったことを扱うものであり，プログラマ間の共創的な社会的関係をみるものは少なかった[Dittrich et al. 02]．

しかしながら，ここ数年の間に，プログラマ個々人の認知的活動や，プログラマ間の社会的関係に着目し，そのためのツールを開発する研究が多く見られるようになった．すなわち，ソフトウェア開発における人的側面を重視するアプローチが開花してきた[UW-MSR-Institute 07]．その理由としていくつもの要因があると考えられるが，中でも大きな影響を与えた要因として，以下のものがあると考えている．

(1) eXtreme Programming[Beck 99]に代表される，プログラマ個人の自由なアクティビティに駆動されるアジャイルな開発形態が広く産業界に受入れられはじめたこと

(2) eclipseなど多くのIDE (Integrated Design Environment)がオープンソース化され，大学や研究機関で開発されたツールや環境が，プラグインの形態で配布され実務者に使用されるようになったこと

(3) 多数のオープンソースプロジェクトの普及により，開発データや開発プロセスにおけるコミュニケーションの過程が研究者らによって使用可能となってきたこと

このような流れを受けて，プログラマのアクティビティを分析し，それに基づくインタラクションデザインを施したツールを開発する研究や[Ko, Myers 04]，プログラマ間の共創という側面に着目し，ソフトウェア開発における社会的側面を支援するためのツールの研究[de Souza et al. 05]などがはじまりつつある．

我々は，ソフトウェア開発という行為は極めてナレッジインテンシブな認知的活動であるというポジションをとり [Ye et al. 08]，ソフトウェア開発者が，個々人のクリエイティブなフロー[Csikszentmihalyi 90]を確保しながら，他者との共創を気持ちよくおこなえるような環境を構築することを研究の目的としている[Ye et al. 07a]．プログラミングにおいては，各個人がクリエイティブに解を構築していく．その際，他者が作った部分を利用していたり，自分が作った部分を他者が利用していたりする．ソースコードが最も重要なアーティファクトとなるが，ソースコードのみでは包含し

きれない背景知識や前提知識は，コメントやドキュメントとして残されている．しかしながら，関連する情報の多くが，外在化されないまま，開発者自身の知識として残っている [LaToza et al. 06]．その結果，前章でも触れたように，開発者間でのコミュニケーションが頻繁に発生することになる．

このように，ソフトウェア開発における知的共創活動は，扱うアーティファクトが論理的であり(物理的でない)，作った要素間には強い数学的依存関係がありながら，タスクを divide and conquer できないという，かなり特異な特徴を有するものであると我々は考えている．ソフトウェア開発における知的創造活動の支援を考えることで，そこから得られた知見は，広く多様な共創的知的創造活動支援へと応用，展開が可能であると考えている．

3 事例：STeP_IN_Java システム

我々はこれまで，Java プログラマのためのソシオテクニカルな環境のプロトタイピングをおこなってきた．構築した STeP_IN_Java システムおよびメカニズムの詳細は，[葉，山本 06]および[Ye et al. 07b]にある．

本章では，STeP_IN_Java の概説をおこなうことによって，1章で挙げた下記の二点がどのように実現されているかを示す．

(1) 何の情報を誰が求めているかによって，専門性と社会的関係の双方を考慮しながら，エキスパートの同定をおこなう

(2) 情報を求める側，求められる側双方のコミュニケーションコストを減らすことを目的として，同定されたエキスパートからなる，アドホックなナレッジコミュニティーを動的に形成する

STeP_IN システムを用いて，Java プログラマは，まず，現在のタスクに関連する Java オブジェクトを検索する．検索結果からはじめて，ユーザは，関連するオブジェクトのソースコード，ドキュメント (Java doc)，そのオブジェクトの利用例(プログラムソースコード)，およびそのオブジェクトに関してこれまでにやりとりされたディスカッションのアーカイブ情報を順にたどっていくことができる．

それらの情報をみてもまだそのオブジェクトについての情報が充分でない場合には，"Ask experts" ボタンを押すことで，質問のためのインタフェースが表示される．この部分に質問文を入力すると，その質問は，システムが内部的に選定した何人かのエキスパートにメールされる．この時点では，質問者は，誰がその質問文を受け取っているかは知らされない．

エキスパートの選定は，質問者が誰であり，何のオブジェクトを探しているのか，に基づいておこなわれる．各ユーザは，テクニカルプロファイル(何についての経験度，専門性が高いか)とソーシャルプロファイル(誰とこれまで頻繁にコミュニケーションをおこなってきたか)という，二種類のプロファイルを有している．これらのプロファイルは，これまでに開発してきたプログラムのソースコードや，やりとりのメールのヘッダー情報をもとにして，システムが自動的に生成したものを，各ユーザが自ら上書きして書き直したものである．

質問メールを受け取ったエキスパートが返答をすると，そのエキスパートのアイデンティティは明らかとなり，返答文は，質問者および質問文を受け取ったすべてのエキスパートに回覧される．一方，もしもエキスパートが質問に答えたくない場合は，その質問メールを無視していれば，そのエキスパートが質問を受け取った事実は明らかにされないままとなる．

回答を得た質問者は，やりとりを評定し，「有益」と評定されたメールのやりとりは，新たなディスカッションアーカイブとされそのオブジェクトに関連づけられて保存される．

4 考慮すべき要因

我々のアプローチで前提としているのは，プロジェクトに関わるアーティファクトと人とがそれぞれノードとして関連するリンクで結びついた，ナレッジワークスペースと呼ぶ空間である．タスクが同じでも，誰がそのタスクに関する情報を必要としているのかによって，関連する情報は異なる．それと同様に，誰がどのタスクに関する情報を求めているかによって，その情報を提供できるエキスパートも異なるであろう，と考えている(図 1)．逆にいうと，適切に情報提供をしてくれそうにない人には，そもそも情報提供依頼を

出さないでおくことで，プロジェクトメンバー間全体のコミュニケーションの負荷を減らせるのではないかと考えている．

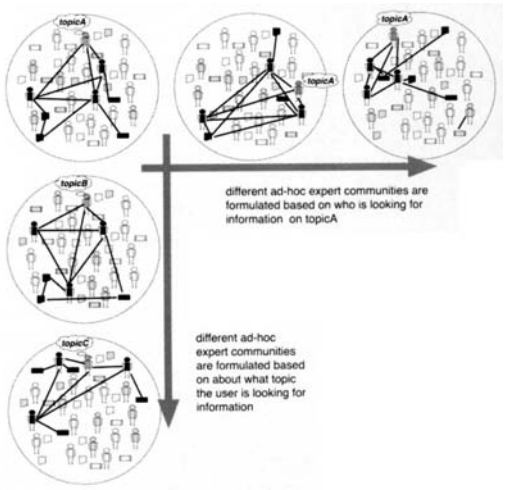

図 1: 誰が何を求めているのかによって，関連する＜モノ＞および＜ヒト＞が異なる

我々は，質問者と「社会的に近い関係」にあるエキスパートを選定することによって，より適切な回答をよりタイムリーに回答してくれるのではないかと考えている．たとえば現在の STeP_IN_Java システムの実装においては，まず，質問しているトピックに専門性の高いユーザを，各ユーザのテクニカルプロファイルを用いて選定する．次に，選定された各エキスパートのソーシャルプロファイルをみながら，

(1) その質問者であれば助けたいと明示的にプロファイルに記述しているエキスパート
(2) 質問者から助けられた回数が多いエキスパート
(3) 質問者と過去のメールのやりとりが，より直近で多いエキスパート
(4) プロジェクト全体で，質問に多く答えてもらっているエキスパート

を順にチェックしていき，該当するエキスパートを5名同定する．そして，これらの同定されたエキスパートからなる動的なメーリングリストを，その質問者の，その質問のためだけに作成する．質問者以外には，誰がそのメーリングリストに入っているのかは，答えない限りはわからない仕掛けとなっている．

しかしながら，この手順によるエキスパート同定が，必ずしも適切な答えをタイムリーに回答してくれるエキスパートである確証はまだない．今後の研究の課題として，これらの，ソーシャルな関係を考慮したエキスパート同定のメカニズムを考える必要がある．たとえば，以下のような要因を考慮することが考えられる．

(1) 義務感と期待感

質問者との関係において，質問者から助けてもらった回数が多ければ，「お返し」として答える義務感が生じ易いと考えられる．質問者との関係ではなく，対プロジェクト全体との関係で，助けてもらった回数の多いユーザは他者を助けてあげがちになるのではないかという予測もできる．現在の STeP_IN_Java のエキスパート同定メカニズムの実装は，これらのソーシャルキャピタルの考え方が基本となっている．

(2) 直近のコミュニケーションヒストリ

より直近にやりとりしたユーザとは，より社会的に近い関係にあるとみなすことができると考えられる．より直近にやりとりをしていれば，質問者の質問のコンテキストをより理解していることも大いにあり得る．それによって，同定されたエキスパートが，より適切な回答を返してくれることも期待できる．

(3) 物理的，部署的な近さ

人は，より物理的に近くに居るひとに質問をし易いと感じがちであるということがいわれている．しゃべりやすい，気心が知れている，など多様な理由があると考えられるが，我々が提案するような，匿名性が保てるようなコミュニケーションメカニズムを提供した場合に，これらの要因の重要性がどう変わるのかは研究の必要がある．

それに加えて，ソフトウェア開発においては，オフショアー開発やアウトソーシングなど，異なるタイムゾーンのメンバーと協働する機会も増加しつつある．時間差のために，回答が遅れてしまう可能性も否めない．

(4) 質問者の組織内での地位，質問の重要性

質問者の組織内での地位が高かったり，質問が非常に重要であったりした場合，質問を受け取った相手もその重要度を同じく認識できていれば，よりタイムリーに要求されている情報を提供するのではないかと考えられる．

(5) 個別要望の記述，嗜好

各個人が相手を指名して，この人が質問してきたら必ず答えたい，この人には答えたくない，といった明示的な記述をすることも考えられる．また，今は忙しいので誰にも答えたくない，聞いてきて欲しくない，といった個人的な状況の説明や嗜好を考慮することも非常に重要であろうと考えられる．

組織やプロジェクトの編成によっては，情報の権利関係やセキュリティなどを考慮し，この人がこのことを聞いてきたことは，それらの人には示してはいけない，といった状況も考えられる．こういった判断を各ユーザにゆだねるのではなくツールがシステマティックにおこなえれば，共創的側面がかなり支援されると考えている．

謝辞

本論に記載した内容の多くの部分は，葉雲文氏および山本恭裕氏との長期間に渡る議論に依るところである．ここに深く感謝する．

参考文献

Ackerman, M.S., Halverson, C., Sharing Expertise: The Next Step for Knowledge Management, in Social Capital and Information Technology, MIT Press: Cambridge. p. 333-354, 2004.

Beck, K., Extreme Programming Explained: Embrace Change, Addison-Wesley Professional, October, 1999.

Csikszentmihalyi, M., Flow: The Psychology of Optimal Experience, HarperCollins Publishers, New York, 1990.

DeMarco, T., Lister, T., Peopleware: Productive Projects and Teams. 2nd ed., New York: Dorset Housing, 1999.

de Souza, C., Froehlich, J., Dourish, P., Seeking the Source: Software Source Code as a Soical and Technical Artifact, Proceedings of GROUP05, New York, 2005.

Dittrich, Y., Floyd, C., Klischewski, R., (Eds.), Social Thinking: Software Practice, MIT Press: Cambridge, MA, 205-222, 2002.

Fischer, G., Nakakoji, K., Ostwald, J., Stahl, G., Sumner, T., Embedding Computer-Based Critics in the Contexts of Design, Human Factors in Computing Systems, INTERCHI'93 Conference Proceedings, ACM Press, pp.157-164, Amsterdam, the Netherlands, 1993.

Illich, I., Deschooling Society, New York: Harper and Row, 1971.

Ko, A.J., Myers, B.A., Designing the Whyline: a debugging interface for asking questions about program failures. ACM Conference on Human Factors in Computing Systems, Vienna, Austria, 151-158, 2004.

Ko, A.J. DeLine, R., & Venolia, G., Information needs in collocated software development teams. International Conference on Software Engineering, Minneapolis, MN, 344-353, 2007.

LaToza, T.D., Venolia, G., DeLine, R., Maintaining Mental Models: A Study of Developer Work Habits, Proceedings of ICSE06, Shanghai, 2006.

Mockus, A.,Herbsleb, J., Expertise Browser: A Quantitative Approach to Identifying Expertise, Proceedings of 2002 International Conference on Software Engineering (ICSE'02), Orlando, FL. p. 503-512, 2002.

中小路久美代，山本恭裕，創発のためのソフトウェア，知性の創発と起源（鈴木宏昭編），「知の科学」シリーズ，5 章，pp.111-131，オーム社，July 2006.

Nardi, B.A., Whittaker, S., Schwarz, H., It's Not What You Know, It's Who You Know: Work in the Information Age. First-Monday: Peer-reviewed Journal on the Internet, 5(5), 2000.

Osterweil, L.J., Software Processes are Software Too, Proceedings of the Ninth International Conference of Software Engineering, pages 2-13, Monterey CA, March 1987.

Soloway, E., Ehrlich, K., Bonar, J., Greenspan, J., What do Novices Know about Programming, in Directions in Human-Comupter Interaction, A. Badre and B. Shneiderman, eds. Ablex: Norwood, NJ, 27-54, 1984.

UW-MSR-Institute, 2007 Summer Institute on the Human Side of Software Development, http://www.cs.washington.edu/mssi/2007/index.html

Ye, Y., Yamamoto, Y., Nakakoji, K., A Socio-Technical Framework for Supporting Programmers, Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the Symposium on the Foundations of Software Engineering (ESEC/FSE 2007), ACM Press, Dubrovnik, Croatia, pp. 351-360, September, 2007a.

Ye, Y., Yamamoto, Y., Nakakoji, K., Nishinaka, Y., Asada, M., Searching the Library and Asking the Peers: Learning to Use Java APIs on Demand, Proceedings of 2007 International Conference on Principles and Practices of Programming in Java (PPPJ2007), ACM Press, Lisbon, Portugal, pp. 41-50, September, 2007b.

Ye, Y., Yamamoto, Y., Nakakoji, K., Expanding the Knowing Capability of Software Developers through Knowledge Collaboration, IJTPM (International Journal of Technology, Policy and Management), Special Issue on Human Aspects of Information Technology Development, Inderscience Publishers, Vol.8, No.1, 2008 (in print).

葉雲文，山本恭裕，Java開発者のオンデマンド・ラーニングを支援するソシオテクニカル環境，SEC Journal，情報処理推進機構，October, 2006.