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

Interaction Design for Supporting Knowledge Communication in Software Development

中小路 久美代[*1,2] イェ ユンウェン[*3] 山本 恭裕[*1]
Kumiyo Nakakoji Yunwen Ye Yasuhiro Yamamoto

[*1] 東京大学先端科学技術研究センター RCAST, University of Tokyo
[*2] (株) SRA先端技術研究所 SRA Key Technology Laboratory Inc.,
[*3] (株) SRA SRA Inc.

Communication in software development has changed from something to be avoided to something to be nurtured. We have identified two distinctive types of communications in software development: coordination communication and expertise communication. This paper discusses different needs for interaction design in support of the two types of communications.

1. はじめに

昨今の工学製品の製造過程の多くの部分は，ソフトウェア開発作業に支えられている．我々は，ソフトウェア開発を，多くの情報を利用，生成しながらプログラムという情報アーティファクトを構築していく，情報インテンシブな知識共創作業であると見なし，ソフトウェア開発の人的要因の支援に着目してきた．ここで着目している人的要因とは，個々のプログラマは固有の専門性を有し，社会的インタラクションを通して知識の生成と交換をおこないながら開発作業を進めていく，という考え方における，認知的，社会的側面である [MSR-WS 2007].

人的要因に着目したソフトウェア開発支援研究の主要なアプローチには，これまでに大きく三つの流れがあったと考えている [Nakakoji 2010]. 1980 年代のアプローチは，プログラミングという作業に必要なエキスパート知識に着目した第一の流れである（たとえば Psychology of Programming [Soloway 1984]）. 1990 年代に多くみられた研究アプローチの第二の流れは，参照するドキュメントを入力，構築するドキュメントを出力とした，開発プロセスを情報処理とみなすものであり，生産ライン制御の観点からマネジメントに着目したものである（たとえば Process Programming [Osterweil 1987]）.

2000 年代に入り，個々の開発者の創造性や開発者間のインタラクションを促す環境の構築を目指す，第三の流れが始まったと考えている．この変化の背景には，下記の二つの点があると考えられる．

(1) オープンソースソフトウェア開発の形態が広まり，開発されるソースコードに加えて，メーリングリストを中心とするコミュニケーションデータや，開発履歴データ，テストデータなどが広く研究者に分析されるようになった．その結果，情報資源の流動性，共有する文化，ピアレビューやピアグローリーといった，個々人の知識創発活動や，社会的なやりとりといった認知的，社会的側面が，実際の開発現場においても重要であることが認識されるようになったこと[Augustin 2002].

(2) XP(eXtreme Programming)を代表とするアジャイルな開発メソッドの普及に伴い，開発するべき対象を細分化し divide-and-conquer 的なアプローチで開発する手法から，開発者が開発する順序やその詳細を自発的に設定したり，チームで連続的なコミュニケーションをおこないながら開発したりするといった手法の有効性が認められてきたこと[Tomayko 2004].

我々は，このような，第三の流れにおける，開発者中心のソフトウェア開発支援研究として，ソフトウェア開発におけるコミュニケーションの支援に着目してきた．ソフトウェア開発におけるコミュニケーションの重要性は古くから認められている．例えばソフトウェア開発作業のうちの 37%の時間がミーティングやメールのやりとりといったコミュニケーションに割かれていることが 1985 年に指摘されており[Fairley 1985]，いかにしてコミュニケーションを減らすか，ということがそれ以後のソフトウェア開発支援研究の大きな目標であった．それに対して 2000 年代以降の，第三の流れの研究においては，コミュニケーションを，＜減らすべきもの＞ではなく，＜支えるべきもの＞として捉えている．ここで，＜コミュニケーションを支える＞ということは＜コミュニケーションを増やす＞ことではなく，非効率なコミュニケーションは減らしつつ，コミュニケーションを開発作業の一部としてみなし，より有効なコミュニケーションのみをおこなえるようにすることである．

我々は，ソフトウェア開発におけるコミュニケーションには，*coordination communication*（協調コミュニケーション）と，*expertise communication*（専門性コミュニケーション）の二種類があると考えた[Nakakoji 2010]. [Nakakoji 2010] では，そのような expertise communication 支援のために重要と考える，九つのデザインガイドラインを挙げた[Nakakoji 2010]．本論では，coordination communication と expertise communication を比較し，必要となると考える支援の違いについて論じながら，そのためのツールに求められるインタラクションデザインのガイドラインの違いについて考察する．

2. 情報ニーズとコミュニケーション

ソフトウェア開発におけるコミュニケーションは，ソフトウェア開発者らが情報ニーズを認識することに起因する．Ko らは，マイクロソフトにおける21人の開発者を観察し，どのような情報ニーズを認識し，どこにその情報を求めるか，をエスノグラフィックスタディの手法を用いて分析をおこなった [Ko 2007]. その結果，(1)「新たに書いたコードは間違っていない？」，(2)「他のプロジェクトメンバーは今何をしているの？」，および(3)「どのコードのせいで今こういう状態になっているの？」という問いかけが，最も多くおこなわれており，それを知るために最も活用される情報源は，(a)ソースコードと，(b)他のプロジェクトメンバーであった，と報告している．

連絡先：中小路久美代, 山本恭裕, 153-8904 東京都目黒区駒場 4-6-1 東京大学先端科学技術研究センター， {kumiyo, yxy}@kid.rcast.u-tokyo.ac.jp; イェ ユンウェン，160-0004 東京都新宿区四谷 3-12 (株) SRA, ye@sra.co.jp

類似のフィールドスタディも同様に，ソースコードとピアメンバー（他のプロジェクトのメンバーや同僚）が，開発者にとって最も重要な情報源となっていることを報告している．ソフトウェア開発におけるコミュニケーション支援研究は，ソースコードを検索し，理解，利用するのと同様に，コミュニケーションすべきピアメンバーを探し出し，情報の交換をおこなうことを目指すものとなる[Ye 2007].

しかしながら，コミュニケーションに至る情報ニーズの目的をより詳細に見ると，二つの異なる目的があることがわかる．「他のプロジェクトメンバーは何をしているの？」という情報ニーズは，自分のしている作業と重複や齟齬がないかを確認することを主な目的として発生しているニーズなのに対し，「新たに書いたコードは間違っていない？」「どのコードのせいで今こういう状態になっているの？」という情報ニーズは，自分のタスクに必要となる情報を得ることを主な目的として発生しているものである．

このような異なる目的に対して発生する情報ニーズに起因するコミュニケーションは，別のタイプのコミュニケーションとして支援するべきであると我々は考え，前者を coordination communication，後者を expertise communication と呼ぶこととした[Nakakoji 2010]. 表 1 に，違いをまとめる．

表 1: 二つのタイプのコミュニケーション

コミュニケーションのタイプ	*coordination communication*	*expertise communication*
目的	作業のコーディネーション	自分の問題解決
ニーズ	重複作業や矛盾の回避 不整合の解決	不足している知識や情報の補填
コストとベネフィット	対称	非対称
利用されるコミュニケーションメディア	対面，個別電子メール，メーリングリスト，チャット，Wiki，コードのコメント	

一方で，ソフトウェア開発者は，いずれのコミュニケーションニーズに対しても，同様に，電子メールやチャット，もしくは対面でコミュニケーションをおこなう．しかしながら，コミュニケーションを支援することを考えると，これら二種類のコミュニケーションは分けて考える必要がある．その理由は，コミュニケーションの送り手と受け手との関係の違いにある．コミュニケーションによってもたらされる利益を得るためには，コミュニケーションのコストがかかるが，利益を得る者と，コストを支払わなければならない者との関係が，二種類のコミュニケーションにおいては異なっている．

coordination communication の目的は，作業のコーディネーションである．作業のコーディネーションがうまくいかないと，自分と，コミュニケーションすべき相手との双方の作業に無駄が出たり，うまく動かなかったりすることになる．それに対して expertise communication の目的は，自分が必要とする情報や知識を質問し，相手から答えを得ることにある．情報が欲しいのは自分（質問者）であり，コミュニケーションの相手は，情報提供を依頼される立場の人（回答者）となる．回答者にとっては，自分の作業を中断し，わざわざ相手が欲している情報を答える必要がある．

この，コストとベネフィットの関係の違いは，コミュニケーションを支援するツールのインタラクションデザインを考える際に，際立った違いとなるべきであると考えている．すなわち，コミュニケーションをする相手の選定の仕方，コミュニケーションのタイミング，コミュニケーションへの招待の仕方，用いるコミュニケーションメディア，コミュニケーション結果の利用の仕方，などが異なる．コストとベネフィットが，コミュニケーションの送り手と受け手で同等であれば，自分がコミュニケーションニーズを感じた際には，同じ重要さでコミュニケーションの相手もそのニーズがあると考えられる．それに対してコストとベネフィットが非対称であると，自分にニーズがあっても相手にはニーズはみられない．相手にコミュニケーションに加わってもらうための仕組みを考慮する必要がある．

[Nakakoji 2010]において我々は，expertise communication を支援するインタラクションデザインの九つのガイドラインを説明している．表 2 左カラムに，ガイドラインを示す．

次節では，coordination communication と expertise communication の比較をおこないながら，共通する要件と異なる要件を考察する．そって，expertise communication のためのインタラクションデザインのガイドラインが，coordination communication を支援する際に適用可能かどうか，可能でないとすれば，どのような異なるガイドラインが必要であるかを論じる．

3. 二種類のコミュニケーションの比較と支援の要件

表 2 の左カラムで示した九つのガイドラインの一つずつについて，coordination communication においても同様に指針とすべきか，そうでなければどのような指針が必要かについて，coordination communication と expertise communication の比較をおこないながら，考察する．

まず，ガイドライン(1)および(2)については，coordination communication においても同様に重要と考える．コミュニケーションは，他の開発作業（たとえばプログラム編集作業やテストデータ編集作業）と切り離されるべきでない．また，誰が何を探しているかによって，誰とコミュニケーションをすべきか，も変わってくると考えられる．

ガイドライン(3)および(4)は，expertise communication はコミュニケーションの相手にとって非常にコストの高いアクティビティであり，可能であれば避けることを目指すものである．しかしながら，coordination communication では，ある開発者にコミュニケーションのニーズが発生する時点で，その相手にとってもニーズは発生している．コミュニケーションを避けるべきか否かは，代替として使える情報源がどれだけあるか，や，コミュニケーションが発生することでグループ／プロジェクト全体にどれくらいコストがかかるか，ではなく，コミュニケーションしないことで，どのくらいのダメージが起こり得るか（すなわち，作業のコーディネーションをおこなわないと，どのようなリスクが生じるか）で考えるべきであると考える．

ガイドライン(5)は，コミュニケーション相手の選定に関わることである．expertise communication をおこなうためには，回答してくれ易い人を選ぶ，ために，所属組織や社会性を考慮すべきであるとした．作業のコーディネーションをおこなうための coordination communication では，誰とコーディネーションすることが最も効果的に矛盾を回避できるか，といった，コーディネーション効果を考慮して，技術的依存性を考慮しながらコミュニケーションの相手を選定するべきである．

ガイドライン(6)は，作業の中断を最小限にするべきであるというガイドラインである．coordination communication を支援する際にも同様に，例えば生じている作業の不整合の重要性に応じて，どのように作業を中断させるかを考える必要がある．その際，作業のコーディネーションのニーズそのものに気付き易い仕組みが必要であると考えられる．

表 2: 二種類のコミュニケーションのタイプとそのためのデザインガイドラインの比較

expertise communication 支援のデザインガイドライン	coordination communication 支援適用の可否
(1). コミュニケーションが他の開発作業から乖離しないこと コミュニケーションは開発作業の一部であり，コミュニケーションのためにシステムスイッチすることは，認知的負荷がかかる	Yes
(2). どの開発者が何についてコミュニケーションをしたいのか，を考慮すること 同じトピックについてでも誰が尋ねているかによって，誰をコミュニケーションの相手として選定するかが異なる	Yes
(3). 同じ情報を得るためにドキュメントやコードがあればそちらを優先し，コミュニケーションがなるべく起こらないようにすること 情報源としての「開発者」はコストが高い．できるだけ＜人＞をわずらわせずに，問題解決することが望ましい	No- リスクに応じて促進
(4). 情報を欲している開発者のメリットと，プロジェクト全体の，グループとしてのメリットを考慮すること 情報が必要だからといって他のプロジェクトのメンバーの邪魔ばかりすることになってはならない	No- リスクに応じて促進
(5). コミュニケーション相手の選定にあたっては，聞き手との組織的，社会的関係を考慮すること 同じ部署にいる人や，社会的関係の近い人（たとえば以前に一緒に仕事をしたことがある人）を選定することによって，答えてくれ易い人を選定する	No- 技術的依存性を考慮
(6). コミュニケーション相手として選定するにあたっては，その人の作業の中断を最小限に抑えること 質問に答えるためには，現時点での作業を中断し，答えをつくり，また作業の続きに戻る，といったコストがかかる	Yes
(7). 質問し易い仕組みをつくること うまい質問の仕方をすると，良い答えをより早く得られることになりがち	No- コミュニケーションを開始し易い仕組み
(8). 回答するかしないかを簡単に選べる仕組みをつくること 答えないオプションが与えられないような仕組みでは，持続的に知識交換が続かない．	No- 緊急性，重要性を判断する仕組み
(9). コミュニケーションチャンネルが，社会性を考慮したもの(socially-aware)であること 人間は，聞くべき人，探すべきところ，ではなく，聞き易い人，探し易いところに情報を求めがちである [Allen 1967]. 回答者は質問者との社会的関係を考慮して回答の有無やタイミングを決める[Cross 2004].	No- インパクト-aware な仕組み

ガイドライン(7)は，質問し易い仕組みを考えること，としている．expertise communication が，質問とそれに対する回答という形態で進むのに対し，coordination communication は，コーディネーションの必要性に気づいた開発者が，コミュニケーションを開始（イニシエート）し，関連する開発者がそれに呼応する形で進む．したがって，coordination communication においては，コミュニケーションを開始し易くする仕組みを考えることが重要となると考える．開発者ではなく，システムが自動的にコミュニケーションを開始するといった場合も考えられる．

ガイドライン(8)は，コミュニケーションに選定された相手が，コミュニケーションに関わり易く（もしくは辞退し易く）することを目指したものである．coordination communication の場合は，作業をコーディネーションすることの緊急性や必要性を判断し易い仕組みが必要である．

最後にガイドライン(9)は，社会性を考慮したコミュニケーションチャンネルの必要性を述べたものである．coordination communication の場合には，作業のコーディネーションをおこなう／おこなわないことで，どのくらいのインパクト／影響があるかという impact-aware なコミュニケーションチャンネルが必要となると考えられる．

既存のコミュニケーション支援のアプローチには，これら二種類のコミュニケーションを明示的に分けて考えているものは見られない．しかしながら，それぞれのコンセプトやツールが，主眼として支援しようとしているアクティビティによって，どちらのタイプのコミュニケーションを対象としているかを大別することができる．表 3 に，大別した結果の，鍵となるコンセプトとツール，およびそれらの主要なフィーチャーをまとめる．

作業のコーディネーションを主眼としたコミュニケーション支援のツールは，アウェアネスや可視化に重点を置いているのに対し，専門知識交換を主眼としたコミュニケーション支援のツールは，エキスパートの同定や選定，社会性を考慮したコミュニケーションチャンネルに重点が置かれているがわかる．

4. 終わりに

本論では，ソフトウェア開発におけるコミュニケーションには，expertise communication と coordination communication の二種類があることを説明し，コミュニケーションにおける送り手と承け手とが担うコストと享受するベネフィットの対称性の違いに起因する，インタラクションデザインのガイドラインの違いについて論じた．

表 3: 二種類のコミュニケーションの支援

コミュニケーションのタイプ	*coordination communication*	*expertise communication*
鍵となるコンセプト	連続的コーディネーション [Redmiles et al. 2007] インパクトマネジメント [de Souza et al. 2008]	知識ソースとしての開発者 [Nakakoji 2006] コミュニケーションチャンネル [Ye 2007]
ツール	Palntir [Sarma 2003] Ariadne [de Souza 2007]	Expert Finder [Vivacqua 2000] STeP_IN_Java [Ye 2007]
主要なフィーチャー	アウェアネス 可視化	エキスパート検索， エキスパートブラウザ socially-aware communication channel（社会的関係を考慮したコミュニケーションチャンネル）

しかしながら我々は，これら二つのコミュニケーションのためのツールを別々のものとして提供することを主張している訳ではない．開発者にとっては，プロジェクトメンバーや同僚といった他の開発者とのコミュニケーションをおこないたい，というニーズがあるのみである．それが，どちらのタイプのコミュニケーションであるのかを自覚し，意識的にスイッチしたいと思うとは考え難い．必要となるのは，支援システムが，開発者毎のプロファイルと，そのタスクのコンテキストから，ニーズとして生じるコミュニケーションがどちらのタイプであるのかを認識し，それに応じたインタラクションを提供することで，シームレスに二つのタイプのコミュニケーションに携わることを可能とすることであると考える．現状では「コミュニケーション」として一括りにされているものに違いのあることをまず認め，それに応じた異なる支援の仕方を同定した上で，それらをユーザとしての開発者の視点からひとつのものとして見せることが，開発者中心の，第三世代のソフトウェア開発支援環境に求められることであろうと考えている．

参考文献

[Augustin 2002] Augustin L, Bressler D, Smith, G: Accelerating software development, 2002.

[Cross 2004] Cross R, Borgatti SP: The ties that share: Relational characteristics that facilitate information seeking. In: Huysman M, Wulf V, (eds) Social capital and information technology. The MIT Press, pp 137–161, 2004.

[DeMarco 1999] DeMarco T, Lister T: Peopleware: productive projects and teams. Dorset Housing Publishing, 1999.

[de Souza 2007] de Souza CRB, Quirk S, Trainer E, Redmiles D: Supporting collaborative software development through the visualization of socio-technical dependencies. In: Proc. of GROUP'07, pp 147–156, 2007.

[de Souza 2008] de Souza CRB, Redmiles D: An empirical study of software developers management of dependencies and changes. In: Proc. of ICSE'08, pp 241–250, 2008.

[Fairley 1985] Fairley R.: Software engineering concepts, McGraw-Hill College, 1985.

[Hassan 2008] Hassan, A.E., Lanza, M., and Godfrey, M., Proceedings of fifth international workshop on mining software repositories. ACM Press, 2008.

[Ko 2007] Ko AJ, DeLine R, Venolia G: Information needs in collocated software development teams. In: Proc. Of ICSE'08, pp 344–353, 2007.

[Mockus 2002] Mockus A, Herbsleb J: Expertise Browser: A quantitative approach to identifying expertise. In: Proc. of ICSE'02, pp 503–512, 2002.

[MSR-WS 2007] 2007 Summer Institute on the Human Side of Software Development, University of Washington and Microsoft Research, August, 2007, Stevenson, WA., http://www.cs.washington.edu/mssi/2007/index.html (visited on 2009/04/20)

[Nakakoji 2006] Nakakoji K: Supporting software development as collective creative knowledge work. In: Proc. of KCSE2006, Tokyo, pp 1–8, 2006.

[Nakakoji 2010] Nakakoji, K., Ye, Y., and Yamamoto, Y.: Supporting expertise communication in developer-centered collaborative software development environments. In: Finkelstein, A., van der Hoek, A., Mistrik, I., Whitehead, J. (eds) Collaborative Software Engineering, Springer-Verlag, 2010 (forthcoming).

[Osterweil 1987] Osterweil L (1987) Software processes are software too. In: Proc. of ICSE'87, pp 2–13, 1987

[Redmiles 2007] Redmiles D, van der Hoek A, Al-Ani B, Hildenbrand T, Quirk S, Sarma A, Filho RSS, de Souza C, Trainer E: Continuous coordination: a new paradigm to support globally distributed software development projects. Wirtschaftsinformatik J, 49: S28–S38, 2007.

[Sarma 2003] Sarma, A, Noroozi Z, van der Hoek, A,: Palantir: raising awareness among configuration management workspaces. In: Proc. of ICSE'03, pp 444–454, 2003

[Soloway 1994] Soloway E, Ehrlich K: Empirical studies of programming knowledge. IEEE Trans Software Eng 10(5): 595–609, 1994

[Vivacqua 2000] Vivacqua A, Lieberman H: Agents to assist in finding help. In: Proc. of CHI'00, pp 65–72, 2000

[Tomayko 2004] Tomayko JE, Hazzan O: Human aspects of software engineering (electrical and computer engineering series), Charles River Media, Inc., 2004.

[Ye 2007] Ye Y, Yamamoto Y, Nakakoji K: A socio-technical framework for supporting programmers. In: Proc. of ESEC/FSE'07, pp 351–360, 2007

[Ye 2008] Ye, Y., Nakakoji, K., and Yamamoto, Y.: Software development as activities creating and utilizing socio-technical information spaces, in Proc. of International workshop on socio-technical congruence, 2008.