組織学習における〈ノイズ〉の役割

――ゲーム・ソフト・プログラマーの事例分析――

阿　部　智　和

1　はじめに

本論文はゲーム・ソフト開発のプロジェクト・チームを観察対象として，組織の問題解決活動と組織学習に関して新たな仮説を創出しようと意図するものである．ここでゲーム・ソフトの開発チームを研究対象とした理由は，ゲーム・ソフトの開発では，①タスク遂行に必要とされる技術が急速に変化し，②最終的な目的となる完成品とそれを完成するための手段が不明確であるという特徴をもつからである．これらの特徴故にゲーム・ソフト開発のプロセスではメンバーたちが次々と新しい知識を生み出し，それを共有して不確実性・曖昧性に対処し，プロジェクトの成果を達成していくというプロセスが顕著に見られる．知識創出と組織学習に関して仮説発見的な研究を行なう上では非常に魅力的な極端事例なのである（見田，1979；Yin, 1994）．ゲーム・ソフト業界の技術進歩は急速である．ゲーム機（ハード）は，おおよそ5年に1度のサイクルで大幅に性能が向上しているため，ゲーム機の表現能力が急速に高性能化し，ソフトウェアの技術進歩が急速に進んでいる．しかもゲーム・ソフトの開発プロジェクトは著しい曖昧性に直面している．ゲーム・ソフト開発は，もちろん，当初に完成品のイメージを構築して進められるのだけれども，それでも製品の最終的な姿の詳細は，事前に完全に規定されることなく，開発プロセスを通じて徐々に創発してくる．しかもその徐々に創発してくる完成品のイメージを達成するための具体的な技術的アプローチそのものも，しばしば開発プロセスの途中でメンバーたちが互いに相互作用を繰り広げながら創造していくのである．

このようなゲーム・ソフト開発においては，常に新しい技術やノウハウが生み出されていくため，新規参加メンバーの個人学習という側面においても，チーム全体の学習と創造性発揮という側面においても興味深い現象が観察されると予想される．技術的に高度な知識を必要とするプロジェクトであるにもかかわらず，事前に用意されたOff-JTは有効に機能せず，チーム・メンバーの開発プロセスにおけるリアル・タイムの相互学習が非常に重要な役割を果たしているはずである．それゆえに，組織の問題解決活動や創造性，組織学習，個人の学習といった問題を考えていく上で，ゲーム・ソフト開発のプロジェクト・チームに注目しているのである．

本論文の結論を先取りするならば，技術変化が激しく，目標と手段に関する曖昧性の高い環境下で開発作業を進めるゲーム・ソフト開発プロジェクトに関しては，仕事場の物理的な設計とそれに関連するチーム・メンバーの行動様式が個人と組織の学習に対して非常に重要な役割を果たしている，ということである．より具体的には，オープンな空間で，他人の作業を「のぞき見る」ことが出来，様々な議論を「聞きかじる」ことが可能である，という物理的な環境設定と，その「のぞき見」と「聞きかじり」を許容し，促進する集団の慣行が，個人だけではなく組織の学習にとってプラスとなっていることを本論文では示していく．

調査は2002年10月から12月にかけて，聞き取り及び観察を中心に行なわれた．インタビュー対象者リストと質問の内容については図表1および図表2に示されている．

本論文においてわれわれが取り扱う事例は，守秘義務のために具体的な名称を公表することができない．それゆえ，以下では，国内大手ゲーム・ソフトメーカーのαプロジェクトと記していくことにする．

2　先行研究のレビュー

ゲーム・ソフト開発プロジェクトを観察対象とし，個人と組織の学習に関する仮説を導出しようと意図する本論文にとって，3つの分野における先行研究が参考になる．すなわち，a)ゲーム・ソフト開発プロジェクトに関する先行研究，

b)状況に埋め込まれた個人の学習，c)組織学習の3つである．

a)　ゲーム・ソフト開発プロジェクト

ゲーム・ソフト開発プロジェクトを直接観察対象とした組織論の先行研究は数多く，また多様である．しかし誤解を恐れずに簡略化するならば，ゲーム・ソフト開発プロジェクトに関しては次の3つの特徴が指摘されてきている．

①開発プロセスにおいて，明確に技術分業を行いながらも，メンバー間の合議や意見のやり取りが重視されている（砂川，1998；小橋，1993）

②それらの開発プロセスを調整するリーダーの存在が重要なこと（生稲，2000；小橋，1997，柳川・水木，2001）

③プログラミング技術などの技能形成はOJTを中心として行われていること（小橋，1998；砂川，1998）

メンバー間の相互作用とリーダーシップ，OJTによる技能形成が重要だという既存研究の指摘は，表現を若干変更するならば，日々の組織的問題解決活動と，その活動を通じた個人学習の重要性の指摘だと言い換えることができるであろう．日々，組織として問題解決活動を実践しつつ，その中で個人が学習していく，というプロセスがとりわけ注目されるのである．このようなプロセスを視野に入れた研究として近年注目を集めているものに「状況に埋め込まれた学習」の研究がある．

b)　状況に埋め込まれた学習

「状況に埋め込まれた学習」(situated learning）の代表的論者であるLave & Wenger（1991）は，新参者を教育するプロセスとして，「正統的周辺参加」(legitimate peripheral participation）という概念を提示した．正統的周辺参加とは，先達者たちが実践を行なっている共同体（「実践の共同体」，communities of practice）に初心者が最初から完全に参加しているのではなく，周辺的な参加状態から徐々に「十全的参加」(full participation）へと移行していくことを指している．Laveらによれば，実践の共同体へ周辺から参加し始め，十全的参

加へと至ることと，熟練を形成していくプロセスは同義である．熟練を形成するための教育プロセスが存在し，その教育プロセスを経て一人前になっていくというのではなく，共同体の一員に成っていくプロセス自体が熟練形成のプロセスなのである（Lave and Wenger, 1991 ; Brown and Duguid, 1991）．

Lave and Wengerの研究に基づいて考えれば，プログラマーたちの人材育成プロセスは，体系的な教育を施すということではなく，むしろ，プログラマーの集団に周辺から参加しはじめ，その集団のメンバーとして一人前に振る舞えるように育っていくプロセスとして位置づけられることになる．集団の周辺から集団の中心へと，あるいは周辺的参加から十全的参加へと変わっていくプロセスとしてプログラマーの世界を認識することが重要だということになるのである．

c）　組織学習

単なる個人の学習ではなく，組織が学習したと表現するためには，①個人の学習した内容が広く組織メンバーに共有される，②必ずしも情報の共有が行なわれている訳ではないけれども，個々人がそれぞれ別々の行動レパートリーを遂行し，組織としての活動全体が改善されている，という2つの可能性のいずれかが成立している必要がある．後者の組織学習は，組織のもつルーティン（組織の行動プログラム）の習熟・変更・創造のことである（Fiol and Lyles, 1985 ; Levitt and March, 1988 ; March, 1991）．

本論文で取り上げるプロジェクトに属するプログラマーは総人員数でも16名であり，フォーマルな組織構造にそれほど縛られることのない自由度の高い集団であるから，情報の共有に関する事実は豊富に存在する．逆に，いわゆる組織内の手続きに体化されるようなルーティンの改変等を，本論文の対象とする集団に関して指摘することはなかなか難しい．しかし，技術進歩のスピードが速い環境というのは，過去に有効だった組織ルーティンがそのまま利用可能ではなくなる環境を意味し，それゆえに次々と新しいルーティンが作られていくことを要求する．それゆえ，文書に体化されることはないとしても，人々の行動に体化された組織ルーティンを次々と廃棄し，創造していくという意味での組織学習は生じている

はずである．すなわち，ゲーム・ソフトの開発プロジェクトでは，文書レベルではなく行動レベルで，学習と学習棄却（unlearning），再学習（relearning）が繰り返される組織学習のプロセスが観察されるであろう（Hedberg, 1981）．

このような組織の学習プロセスと上で見た個人の学習プロセスは，当然密接に関係している．組織学習とはいっても，やはり学習する主体は個人なのであり，組織メンバーが誰も学ぶことなく組織が学ぶということはあり得ない．組織がもはや有効性を失ったルーティンを捨て，その代わりに新たなルーティンを創造する場合には，組織メンバーの誰か，あるいは皆が協力して問題解決を行なったり，新たな知識を生み出したりする必要があるのである．

3　事例紹介

a）作業空間のレイアウト

インタビューの対象となったαプロジェクトのプログラマーは16人である．これら16人のプログラマーは，図表3に示すように30畳くらいのスペースで作業をしている．彼（女）らは3人一組でまとめられ，互いに高さ1.5メートル程度のパーティションで分けられている．パーティションは高さが大人の首の位置程度なので，同僚のディスプレイに映し出されている作業の様子を周囲から窺うことや，同僚が互いに話している内容を聞くことも可能である．このような物理的環境は意図的に作られている．このようにパーティションで区切られたゾーンがαプロジェクトにおいては数カ所存在している．

b）新人育成の方法

ゲーム業界では主要なハードが変わるたびにソフト開発技術が変化していく[1]，その上，プログラム言語も急速に変化していく．それゆえプログラムに関する体系的な研修を行ったとしても，そこで身につく開発技術は，1つないし2つのゲーム開発が終わってしまうころには，大半が陳腐化してしまう．そのため，他のゲーム・ソフト作成会社と同様に，αプロジェクトでも研修を通じたプログラムの学習よりも，実際に仕事をしながら，自ら試行錯誤することや他のプログラ

マーとのコミュニケーションを通じてタスク遂行中に学習することが重要視されている．実際，「わからない事に直面したら即座に周りの人間に問いを発するように」というOJT促進の指示が，当初からきわめて重要視されている．

実際，すぐに周囲に問いかけるという行動慣行を身につけさせるべく，αプロジェクトにおいてはプロジェクトに加入当初から比較的難しい課題が与えられる．ここでは2年目のプログラマーであるC氏が入社1年目を振り返って述べていることと，ベテランのB氏が難しい課題を与える側の意図について語っていることを紹介しておこう．

C氏の発言

一番最初は，確かBさんに，このソフトのプロジェクトの概要とプレイステーションの概要の二本立てを教えられたと思うんですけど．「（最初は資料を）読んで（自分でまずやって）」と．「これとこれを渡すので，読んでわからないことがあったら聞け．」と（言われて）

・・・（中略）・・・

（その作業の中で）行き詰ったときは本当に行き詰ったとわかったら，即座に別の人に「なんかここがわからないんですけど」と聞きに行ってました[2)]．

B氏の発言

最初，わりと嫌がられるんですけど，ハードルの高いことをやらせるようにしているんですね．それはゲームそのものではなくてもツールであったりとか．ゲームそのもので短期間で仕上げさせられた奴もいますし．それぞれのスキルに合わせて，わりとその人にとってハードルを高いことをやらせる．

・・・（中略）・・・

（新人の作業が）後ろから見ていて，明らかに間違っていたらとめますけど．基本的には作業というのは分かれていますから．僕も仕事をしていますし．講義形式ではないですから．これを2週間（以内で）という形で，後は任せちゃう．わからんところがあったら聞いてと．聞いてくれないと困るんです

ね．行き詰られたまま自分でわかると（思い込んで）放っておかれたら（困るので），なんでもいいから聞いてと．わりと高いハードル（を与えること）と「聞け」ということで，わからんことがあったら聞いてディスカッションをするというやり方を学んでもらうと．もちろん（最初から一人では）わからないだろうと．誰にも聞かないで出来るものは与えていないと（思っています）．だから，そこで聞いてやってくれと．聞くことでプログラマーの雰囲気にもなじめるし[3]．

B氏が「わからんことがあったら聞いてディスカッションをするというやり方を学んでもらう」と述べているのは示唆的である．というのも，B氏の指摘はプログラム作業の技法等を学ぶと主張しているのではなく，むしろプロジェクト・チームとしての問題解決の方法を学ぶと主張しているからである．この「わからないことを即座に周囲の人に問う」という作法が，単に新人時代ばかりではなく，その後もαプロジェクトで仕事を進めていく標準的なやり方になっているのである．

c）　実際の作業の様子

αプロジェクトにおいて，疑問が生じた場合や問題発生時に即座に周囲の人に問いかけ，議論を通じて作業を進めるという仕事のやり方が奨励されているのは，先に挙げた技術の変化以外にも次のような要因がある．すなわち，①日々の作業中に創発してくるチーム固有の技術の存在と，②作業におけるバグを可能な限り回避するということである．ここでは，バグ回避の事例を紹介しておこう．

基本的にプログラマーの作業は，個々に分割された作業からなっている．その分割して行った作業をある時点で統合し，次の作業に移っていく．その統合時に他のプログラムを破壊するようなバグが発見された場合，きわめて深刻な問題をもたらすことになる．単に1箇所のみのバグであれば，そこを処理するだけで十分である．しかし他のプログラムを破壊するような重大なバグの発生は，破壊されたプログラムを作り直す作業をプログラマーたちに強いることになる．その結

果，完成するまでの期間が延び，販売時期を逃してしまったり，余計な経費がかかってしまったりすることになる．それゆえ，重大なバグの発生を各作業の統合時よりも前の時点で解決するためにも，各自の作業途中において質問を自由に出来る雰囲気が重要視されているのである．

このようにして発見されたバグは，その場の当事者だけに留まらず，その場にいる全員に即座に共有されることになる．バグはプログラマーたち全員にとって常に意識される重大なテーマであり，そうであるがゆえにバグに関する情報はきわめて多くの人々の注目を集めるのである．この点についてC氏とB氏はそれぞれ次のように述べている．

C氏の発言

> バグが出たというときに，全員が集まって，何人かがバグの原因を探っていって，原因がわかったとき，すごく大きな声で公表されるんですよ．こういう理由でバグが出たと．プログラマーはバグに興味があるので，全員が集まってきて「なるほどこういう理由でバグが出るんだ」という話になる[4]．

B氏の発言

> その話を聞いて，周りで青ざめるやつがいるんですよ．自分も（同じ事を）やっていたと．こうやって，バグの理由がみんなの記憶に残っていくんです[5]．

バグのようにプログラマーにとって重大な情報は，発見した当事者たちのみが認識して終わるわけではない．比較的早いタイミングで全員が作業をいったん停止し，その内容に耳を傾けるからである．そこでバグの発生理由が全員に説明されるのである．その後，同様のロジックを使いプログラムを組んでいた可能性がある人は，自分のプログラムを再検討することになる．このようにして，特定個人の作業途中に発見されたバグはテスト・フェーズに移行する以前に発見され，チーム全員のプログラム中で回避されることになる．しかも，バグの発生する理

由を全員が学習し，共有するのである．

このようにして，αプロジェクト内では，頻繁に質問が周囲の人間に投げかけられている．質問された側は，当座の作業を止めてでもできる限り答えるように奨励されているため，その場で即座に議論になることが多い．また，先にも述べたように非常に密集した作業空間であるため，周囲から「のぞき見」が可能であり，他人の作業の様子をうかがい，その作業内容について指摘することが出来る．このようにしてαプロジェクトの作業空間には，頻繁にタスク遂行上の疑問や問題を解決するための議論が行きかっている．作業空間内を行き交うこの音声を〈ノイズ〉と呼び，これ以降の議論を進めていくことにしたい．自分の目の前にある仕事を処理する上では周囲の雑音のように聞こえながら，その実，自分のプログラム技能を高める上でも，チーム全体のプログラム完成度を高める上でも，またさらに新しい知識を生み出す上でも潜在的に重要な情報を含んでいる可能性がある，という意味で括弧付きの〈ノイズ〉という言葉をここでは用いている．

4　仮説の展開

本論文が注目している〈ノイズ〉の本質的な特徴は，大まかに言うと，2つに集約できると思われる．1つは〈ノイズ〉がそれを傍らで聞いている者たちの背景情報を豊富にするという特徴であり，もう1つは〈ノイズ〉がオープンな議論の場（自由に参入可能な議論の場）の存在を意味しており，そのオープンな議論の場が個人と組織の学習に影響を及ぼすという経路である．つまり背景に流れている〈ノイズ〉を「つまみ聞き」することで個人の保有する情報量が増え，その〈ノイズ〉が琴線に触れた途端に，その会話に自由に参加することができる，という点に〈ノイズ〉の特徴があるのである．以下では，この2つの本質的な特徴に注目しながら，〈ノイズ〉が個人や組織の活動の効率化にどのように貢献し，また個人や組織の学習にどのように貢献するのかを仮説として展開し，整理していくことにしよう．

a）背景情報の蓄積

まず第1に，〈ノイズ〉が存在する場合，周りの人間たちは，(イ)誰が，(ロ)どのようなテーマで議論していたのかを不完全ながらも記憶している可能性が高まる．起こりうる問題や主要なポイント等々，漠然とした形ではあっても，かなりの量の背景情報をチーム・メンバーに共有させる作用を〈ノイズ〉が担っているのである．〈ノイズ〉が背景情報の共有を促進するために，個人の問題解決活動も，チームの問題解決活動（何人かで共同で行なう問題解決活動）も効率が高まる．なぜなら，問題解決活動の重要な一部分，すなわち，どこ（誰のところ）に解を探索にいくかという問題に答えを出すフェーズが，「(イ)誰が，(ロ)どのようなテーマで議論していたのか」を共有していることで簡略化可能だからである．

背景情報の共有は，問題解決活動の効率化に貢献するばかりではない．背景情報の共有は個々のプログラマーたちの学習を促進する効果を持ちうる．αプロジェクトのようなゲーム・ソフト開発の世界では，技術進歩が急速であり，既に確立した技術体系をベテランが新人や中堅たちに教えていくだけでは不十分である．ゲーム・ソフト開発のように技術変化が急速な世界では，次々に生み出されていく技術・解法・ノウハウといった知識は，教科書に取り入れられるのを待つべきものではなく，むしろ口頭で伝えられて急速にチーム内に伝播していくものである．

このような状況下では，体系的な教育ができない代わりに，常に誰に何を訊ねれば良いかが分かる状況を作っておくことで，個人の学習を促進することができる．フォーマルなOff-JTがうまく機能しない場合に，OJTが重視されることは多くの論者が指摘しているが，ここで強調しておきたいことは，同じOJTでも，より効果的かつ効率的に行なうためには，常に創発し続けている技術・解法・ノウハウに関して，リアル・タイムに誰に何を聞けば良さそうかを示し続けることが大切だ，ということである．〈ノイズ〉はその機能を担っていると考えることができるのである．

〈ノイズ〉が個人の学習を促進する経路を定式化して言えば，次のようにまとめることができるであろう．

① 日常的にプログラミングに関する議論を「周辺で聞きかじる」ことによって，背景としての情報が蓄積される．背景としての情報は，十分に確立された知識として残るのではないけれども，(イ)誰が，(ロ)何をテーマとして話していたか程度の情報として記憶に残ることが多いと考えられる．

② それゆえ，漠然とした状態で蓄積されている背景情報が大量に存在するならば，自分が何らかの問題に直面したときに，〈ノイズ〉として背景に蓄積されている情報の中から類似のテーマで話されていた場面が想起される確率が高まる．

③ 〈ノイズ〉は「周辺で聞きかじった」ものであるから，話の全容を想起することは困難であろう．それゆえ，その想起された情報内容について，かつて議論していた当事者たちに，「あの時，こんなことを議論していましたよね」というタイプの問いを新たに発することになる．これだけでも個人の情報処理効率を高める効果があるはずである．なぜなら，誰に問えば，もっとも効率良く自分の求めている情報が手に入りそうかが予想しやすいからである．このような情報が存在しなければ，どこに情報を探索にいくべきかという基本的なところから情報処理活動がスタートすることになる．

④ この問いに対する答えが，回答者にとって既知の簡単なものであった場合，回答者側には学習が生じるとは言い難いが，もちろん質問者側には大いに学習が生じる．〈ノイズ〉が発生する作業環境が特徴的なのは，単に問いを発した人間と回答した人間のやりとりを当人たちだけが聞いているのではなく，その他のメンバーも漠然と聞いているという点である．簡単な回答であるからベテランがその回答から学ぶことはないかもしれないが，まだ若いプログラマーたちにとっては，自分で質問したわけではないけれども，十分に学ぶ余地のある対話が進行することになる．周辺の若手が「何となく」学習するところに〈ノイズ〉の特徴がある

b) 自由参加可能な議論の場

〈ノイズ〉が個人と組織の学習にとって非常に重要な役割を果たす2つめの側

面は，それがオープンな参入を可能とする対話の場になっているという点である．対話に参加していない人にとってはノイズ（文字通りの雑音）であるが，ある時点までノイズであった他者の対話が，自分の関心事に触れていることが分かった場合には，その人はその対話そのものに参加していくことになる．この場合，〈ノイズ〉が流れていることで，自分にとって重要な対話の場に即座に参加することが可能になるのである．このような多様なメンバーが自由に参加してくることで，新しい技術・解法・ノウハウ等が生み出される可能性が高まるのである．

このプロセスも簡単に定式化を試みておこう．a)の議論の最後の④の部分だけ次のように代えることで，新たな知識が生み出されるプロセスを仮説的に定式化することが可能になる．

④質問者の問が，以前にやりとりされた問いと回答とは若干異なるものの場合，学習するのは質問者ばかりではなくなる．回答者は，少し考えて，以前に回答したものに若干の修正を施さなければならなくなる．この場面ですでに回答者も学習することになる．

更に，このような思考と変更・修正というプロセスが皆に聞こえる場でやりとりされることで，周辺の人々がこの対話に参加し始める可能性が高まる．たとえば，この問いがきっかけとなって，前回議論された「解決策」がほんの少し異なる新しい領域でも利用価値があることが判明し，応用範囲の広い技術として皆の記憶に定着していく場合がある．この場合は，問いを発した人ばかりでなく，答えた人も学び，またその場面が印象深いものであれば，集団としても知識が共有され，また新たな対話の促進が行なわれ，チームとしての学習が進んでいくことになる．

⑤さらに質問者の問いがきっかけとなって，新たな視点が加わり，次々と新たな知識が生成される場合もある．〈ノイズ〉はまさにオープンにやりとりされている議論であるがゆえに，その議論の中で問題になっている事柄に興味のある人間や，その解決に貢献できそうな人間が自由に参入・退出可能である．このようなオープンな議論のやりとりが，新しい知識を生み出すことにつながりやすいのである．誰でも参加できる議論の場であるがゆえに，〈ノイズ〉が〈ノ

イズ〉を次々に生み，その内容を高度化し，場合によってはまったく新規のアイディアや組織ルーティンを生み出すということがあり得るのである．

実際，このようなプロセスが実際に生じていることを示す発言は多数聞かれた．ここではその典型例として，プロジェクトの中堅格のＥ氏の意見を示しておこう．

Ｅ氏の発言

> 席をはずそうとしたときに，水の画像を表現する実験をやっているのが聞こえて，「これどういう風にやっているの．」と言って，見に行きました．（しばらく）後ろでそれを見ていて，「じゃあ，テクスチャを二枚重ねて，半透明にしてずらしたらどうか」と口出しをして．それっぽく見えるので（使う）ということになったこともありましたね[6]．

5　おわりに：要約と今後の課題

本論文の主要な論点をまとめると以下のようになる．すなわち，ゲーム・ソフト開発のように技術変化が激しく，目的と手段に関する曖昧性の高い環境下で開発作業を進める際には，仕事場の物理的設計をオープンにし，作業の様子を「のぞき見」し，周囲で行われている議論を「聞きかじる」ことが可能となる環境を設定し，それらの行動を許容・促進することで個人と組織の学習・知識創出が活発化する可能性が高い，ということである[7]．

•

本論文のたどり着いた地点は，Lave & Wenger の状況に埋め込まれた学習の議論と類似した部分と若干異なる部分の両方を含んでいる．われわれの研究対象であるプログラマー集団でも，若いＣ氏やベテランのＢ氏の引用された発言から，新人が周辺的な参加から始まって，徐々に十全的参加へと移行すること，そうして集団の中心的なベテランに成長していることが示唆されていた．

しかし，同時に Lave & Wenger の議論とは異なる点も同時に明らかになった．たとえば周辺的な参加者に指導されるのは「問う」という実践・慣行（prac-

tice）を身につけることであり，集団内で十全的な参加に向かうにつれて身に付いていくものは，「問い」・「答え」・「議論する」という〈ノイズ〉を発生させる実践・慣行と，その〈ノイズ〉の中で仕事を進めていくという身の処し方であって，プログラマーとして実際に遂行していく問題解決活動そのもののスキルではなかった．具体的なスキルを実践する部分でなく，そのスキルを生み出すプロセスに関する実践・慣行に関して正当的周辺参加の議論が当てはまるのである．

この点はおそらくLave & Wengerが取り上げた事例と本研究の取り上げた事例の技術変化速度の相違に起因しているのではないだろうか．Laveらが主たる研究対象として取り上げた事例は，ユカタン半島の産婆や肉の加工職人である．すなわち，それらのタスクを遂行する上で必要とされる技術の変化があまり無いものを対象となっている．それゆえ，その集団における振る舞い方を憶えるプロセスがそのままスキルを憶えるプロセスになっていたのでは無かろうか．しかし技術変化の激しい分野を対象とするならば，スキルは常に組み替えられていくために，スキルそのものに関連する行動を自然に行なえるようになることではなく，むしろスキルを生み出す集団のプロセスで自然に有意義な行動を行なえるようになることが重要になる．Lave & Wengerの議論を技術変化の激しい領域に応用する場合には，具体的なスキルの部分とそれを生み出す行動の部分という少なくとも2つのレベルに注目していかなければならないのではないか，という示唆が得られるのである．

しかも，同じ理由から，Lave & Wengerの議論では初学者と熟練者の関係が非常に固定的であり，徒弟制の議論に見られるような初学者から熟練者へと成長するプロセスと同じ現象なのだという解釈が成立する（上野，1999）．しかし技術変化が激しい領域では，集団が遂行している作業に必要な知識が固定的ではなく，集団が次々に知識を生み出している．その知識創出プロセスにおいて，十全的参加を果たしているベテランもまた知識を生み出し，新参者も自ら知識を生み出さないまでも，ベテランが新しい知識を生み出す材料提供程度の役割を果たしている可能性がある．Lave & Wengerの議論では必ずしも重視されていないベテランが新参者に教えることでベテランも，その周辺にいる人間も学び，新しい

知識を生み出す可能性が高まるという議論が，技術変化の激しい領域を観察対象に選ぶことで見えてきたのである．

本論文における研究にはもちろん多数の問題点が含まれている．とりわけ大きいポイントは，プロセス性に注目しているにもかかわらず，本格的な参与観察を行ったのではなく，現場でのインタビュー調査や観察にとどまっている点である．最先端のゲーム・ソフト開発であり，他社に秘密が漏洩する恐れもあるために部外者が参与観察をする可能性は非常に低いとはいえ，今後，より深く集団に入り込んだ本格的な参与観察を目指した努力が必要であろう．

第2に，本論文では技術変化のスピードが速い産業における学習をテーマにしていたが，その考察対象として選んだものは1産業内の1社，さらにその中の1プロジェクトにすぎない．本論文が主張するような〈ノイズ〉の学習に及ぼす効果が本当に技術変化の速い業界で一般的に見られるのか否かという点に関して，今後，他のプロジェクトや会社，業界など観察対象の範囲を広げていく必要があるかと思われる．

第3に，本論文では，自由闊達な議論を行える環境として，問いを奨励する組織風土や，議論が生まれやすい空間レイアウトの重要性について指摘した．本論文で対象となったプログラマーの総人数でも16人とそれほど大きな集団ではない．それだからからこそ〈ノイズ〉が生成されて，想起可能な背景情報として有用であった可能性もある．たとえば，規模が本論文の対象とした集団よりも大きくなる場合，〈ノイズ〉として生成される情報過多となり，問題解決プロセスには，〈ノイズ〉が存在しないほうが望ましいとも考えることが出来る．また，本論文よりも小さな組織の場合，成員が全員議論に参加する可能性が高いので〈ノイズ〉を発見すること自体が不可能であろう．それゆえ，組織の規模と〈ノイズ〉の関係について検討するために，同じゲーム・プログラマーのプロジェクトでも，規模の違う組織との比較対照を行う必要がありそうである．今後の課題としたい．

図表1　インタビューリスト

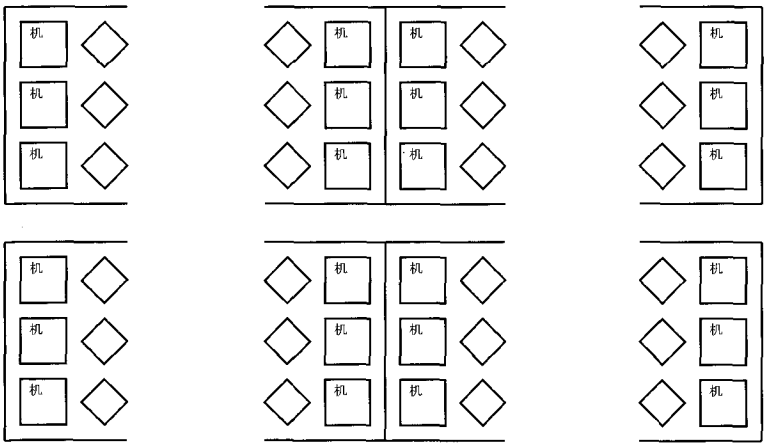
日時	名前	勤続年数	職種	インタビュー時間	備考
10月7日	A氏	17年目	プロデューサー	2時間	
10月16日	B氏（1回目）	9年目	プログラマー（システム）	1.5時間	
	C氏	2年目	プログラマー（ゲーム）	1.5時間	
11月19日	B氏（2回目）	9年目	プログラマー（システム）	1.5時間	D氏のインタビューに同席.
	D氏	8年目	プログラマー（ゲーム）主に敵キャラクターの演出	1.5時間	
12月9日	B氏（3回目）	9年目	プログラマー（システム）	合計2時間	E, F, G氏のインタビューに同席.
	E氏	6年目	プログラマー（システム）	1時間	システム・プログラム部門でB氏に次ぐ地位.
	F氏	5年目	プログラマー（ゲーム）主に敵キャラクターの演出	0.5時間	D氏の直属の部下の一人.
	G氏	1年目	プログラマー（システム）	0.5時間	

注）2002年12月9日のインタビュー終了時のデータである.

図表2　プログラマーを対象としたインタビューの主な質問事項

主な質問事項
プログラマーに必要な技術，プログラム言語など
新人時代の経験（仕事の内容，期間）
新人を指導した経験の有無
作業で行き詰ったときの対処法（自分個人，チームではどのように対処しているか）
周囲の会話の内容が自分の仕事に役に立ったと思える経験
周囲で議論が行われていることに関して，普段どのように思っているか
議論の発生するきっかけ
議論の中から新たな解決策を生み出した経験
バグが発生したときに，どのように共有されるか
定期的な会議での議論の内容
議論や会議以外の情報源

図表3　パーティション図



[出所]　B氏へのインタビュー（2002年12月9日）を基に筆者が作成．
ひし形は人を表す．机の長さが約2メートルくらいである．コの字型に置かれているのがパーティションである．これらがおよそ30畳くらいのスペースに配置されている．

本論文は，一橋大学大学院商学研究科を中核拠点とした21世紀COEプログラム（「知識・企業・イノベーションのダイナミクス」）から，若手研究者・研究活動支援経費の支給を受けて進められた研究成果の一部である．同プログラムからの経済的な支援にこの場を借りて感謝したい．なお本論文作成に当たり，匿名レフェリーの方から大変貴重なコメントをいただいた．ここに記して感謝したい．

参考文献

Brown, John Seely, and Paul Duguid, "Organizational Learning and Communities-of-Practice : Toward a Unified View of Working, Learning, and Innovation," *Organization Science*, Vol.2, No.1, 1991, pp.40–57.

Fiol, C. Marlene, and Marjorie A. Lyles, "Organizational Learning," *Academy of Management Review*, Vol.10, No.3, 1985, pp.803–813.

藤川佳則「ソフト開発を推進するダイナミズムの源泉：任天堂とソニーのビジネスモデル間競争」嶋口充輝・片平秀貴・竹内弘高・石井淳蔵『マーケティング革新の時代（2）製品開発革新』有斐閣，1999，pp.363–387.

Hedberg, Bo L.T., "How Organizations Learn and Unlearn," in Nystrom, Paul C., and William H Starbuck (Eds.), *Handbook of Organizational Design*, Vol.1. New York : Oxford University Press, 1981, pp.3–27.

生稲史彦「家庭用ゲームソフトの製品開発」藤本隆宏・安本雅典『成功する製品開発：産業間比較の視点』有斐閣，2000，pp.187–207.

小橋麗香「家庭用テレビゲームソフト産業の戦略と組織」『ビジネス・インサイト』第1巻，1993，pp.74–90.

小橋麗香「日本における家庭用テレビゲームソフトウェアの開発」『国際研究論叢』第10巻3・4号，1997，pp.81–107.

小橋麗香「日本のゲームソフト会社の人材マネジメント」『国際研究論叢』第12巻第4号，1998，pp.1–22.

Lave, Jean, and Etienne Wenger, *Situated Learning : Legitimate Peripheral Participation* (Learning in Doing : Social, Cognitive, and Computational Perspectives). Cambridge : Cambridge University Press, 1991.（佐伯胖訳『状況に埋め込まれた学習：正統的周辺参加』産業図書，1993.）

Levitt, Barbara, and James G. March, "Organizational Learning," *Annual Review of Sociology*, Vol.14, 1988, pp.319–340.

March, James G., "Exploration and Exploitation in Organizational Learning," *Or-*

ganization Science. Vol.2, No.1. 1991, pp.71-87.
見田宗介『現代社会の社会意識』弘文堂，1979.
砂川和範「日本ゲーム産業に見る起業者活動の継起と技術戦略：セガとナムコにおけるソフトウェア開発組織の形成」『経営史学』第32巻第4号，1998，pp.1-27.
上野直樹『仕事の中での学習：状況論的アプローチ』東京大学出版会，1999.
柳川範之・水木和幸「ゲーム産業に見るイノベーションの構造：リーダーシップの重要性」伊藤秀史編『日本企業　変革期の選択』東洋経済新報社，2002，pp.337-363.
Yin, R.K., *Case Study Research : Design and Method*, (2nd Edition). Sage Publication, Thousand Oaks ; CA, 1994

1）　ここで説明している開発技術とはそれぞれのハード（ゲーム機）ごとに異なる開発環境を扱う技術を指し，プログラム言語などのスキルとは異なるものを想定している．一般的にハードは5年に1度のサイクルでイノベーションが起るとされている．
2）　インタビューC氏　2002年10月16日
3）　インタビューB氏　2002年10月16日
4）　インタビューC氏　2002年10月16日
5）　インタビューB氏　2002年10月16日
6）　インタビューE氏　2002年12月9日
7）　本論文から得られた知見は組織学習論に対しても示唆を提供できる．本論文は，オープンな作業空間が，他人の作業の様子を「のぞき見」や「聞きかじり」を可能にし，個人が情報を蓄積するだけではなく，問題解決の議論を促進する要因であることを示してきた．従来，組織論の研究においては議論を促進する要因として組織の風土や文化などに着目した研究を行ってきたと思われる．本論文では，先行研究から導き出された知見を否定するわけではないけれども，組織風土や文化のみではなく，作業空間の設計も加えて考慮することが重要であると思われる．問題解決に関する議論が，会議室など作業から離れた場で行われるのではなく，作業をしているその場で行われることが，個々人の背景情報の蓄積と集団の問題解決能力の向上に寄与しているのである．

［2004年12月14日受稿
2005年1月21日レフェリーの審査をへて掲載決定］

（一橋大学大学院博士課程）