博士論文

知識モジュール共有への動機とためらいの要因解明

－IT 企業 A 社のソフトウェア開発部門の事例研究－

堀田 耕一郎

主指導教員 國藤 進

北陸先端科学技術大学院大学
知識科学研究科

平成 27 年 3 月

概要

本論文はICT企業A社におけるソフトウェア開発部門内での知識共有行動の動機およびためらいについて明らかにするものである。実際にこの組織においてはソフトウェア知識の共有は成功していない。ソフトウェア技術者たちはソフトウェア知識の共有に対して積極的ではなく，知識提供行動のみならず知識を受領しようということにも消極的である。この研究では，知識共有の行動と組織運営との関係を知識の提供者と受領者の両者の観点から質問票及びインタビューにより調査した。

ソフトウェアの知識はソースプログラムによって過不足なく記述されている。つまり形式知と考えられる。形式知であれば知識共有は容易なはずにもかかわらず，実際には共有されていない理由を明らかにすることで知識共有を進めることができるものと考える。知識移転を促進できる組織運営の方法を解明することができれば，ソフトウェアの重複開発を防ぐことを期待できる。この目的のためにリサーチクエスチョンとして「どういう組織環境が社内のソフトウェア資産の活用を促すか？」を立てた。

A社の組織運営は1990年頃を境に大きく変わった。それまではあいまいな人事評価を行い，各組織ユニットの損益意識も弱いものであった。これを厳密なマネジメントに切り替え，成果主義を取り入れ，組織ユニットに対する損益計算も厳密に行うようになった。この変革前にあったチームをまたいだ高次の場は，変革と時を同じくして小さく，弱くなったことが観察された。この結果，知識共有活動が停滞するようになったと考えられる。

知識提供者についての調査では，知識を抱え込もうという意思は見られなかった。しかし，技術者は知識をチーム外に開示することが許されないのではないかと懸念し，開示をためらっていることがわかった。組織の規範や上司の意思が明確になっていなかったためである。厳密なマネジメントへの変革に知識共有の活動が取り残されたものと考えられる。

一方、知識受領者についての調査では，実際に受け取って利用しているのは開発効率に関心を持ち社内外に人脈を持つ熟練した技術者であった。対象となるソフトウェアは理解しやすいようにモジュール化されているものであったが，提供者に直接質問できるものに集中していることもわかった。実際にソフトウェアを受領して使うにあたっては，ソースプログラムを記述する際の論理の選択理由や留意点などの暗黙知も獲得する必要がある。さらに，不完全性，動作環境への依存性もあるため，ソフトウェア知識の共有のためには提供者との会話が可能なことは重要な要件であることがわかった。

上記よりリサーチクエスチョンへの解答として「組織の方針として知識移転を明確に推奨しながら，技術者に自発的な判断を促しそれを過度に管理しないことが知識移転を促し，社内での知的資産活用につながる」を得た。ここから，本論文では以下の含意を得る。(1) あいまい性を持った組織運営の方が知識共有を推進する。(2) ソフトウェアは形式知要素と暗黙知要素をともに持っていて，この性質から知識移転に影響を与えている。(3) 知識共有に対する明確な意思表示がされると知識共有が推進される。

Abstract

This paper clarifies the motivation and hesitation of knowledge sharing among software development teams in an ICT company, company A. Software knowledge is not successfully shared in the company. The purpose of this research is to investigate the relationship between knowledge sharing activity and management from both sides of knowledge provider and recipient. The clarification of these matters promotes the knowledge sharing activity and reduces redundant software development activity.

Software engineers write program (source code) in human-readable style and the source code is automatically compiled into the software product. So, all of the knowledge in the software is explicitly described in the source code completely. Because of such consequence, it seems that software can be treated as explicit knowledge. If the knowledge is explicit, sharing should be easy. By investigating the reasons why it is not successful, we can encourage knowledge sharing activity.

Company A has tried to adopt the strict management style and lost ambiguity of the management. On the way of this change, BA among the development teams became small and weak. It is one of the reasons that knowledge sharing activity reduced.

From the view point of the knowledge providers, it was clarified that they don't intend to hoard the knowledge. But engineers hesitate to provide the knowledge because they assume that it is not allowed. Company rule or managers' thought is not clarified.

When the knowledge is transferred, the recipients apparently get benefit. But they don't get the software developed by another team so often. If the software is modularized and easy to understand, engineers tend to receive and use the module. Experienced engineers are likely to import software modules if they pay attention to the efficiency of the development. The engineers who have successfully received the knowledge don't necessarily understand the imported modules well.

And through the investigation, it was found that software is not pure explicit knowledge. It needs tacit knowledge portion for developers to use, so it is imperfect or defected by itself and it has context dependency. Because of these characteristics software knowledge sharing is not simple,

As a result, knowledge sharing is promoted on the following two conditions. The first one is that the company policy to share knowledge is clearly announced. The second one is that the management with ambiguity encourages autonomous decision of engineers and knowledge sharing.

From these findings, this paper concludes three kinds of implication. (1) Ambiguous management style rather than strict management style supports successful knowledge sharing. (2) The characteristics of software, which is a mixture of both explicit and tacit knowledge, accelerate or suppress the knowledge transfer. (3) Clear message to share knowledge encourages the activity.

Key words : knowledge sharing, ambiguous management, software module, recipient, motivation

目次

図目次

表目次

第1章 序論

1.1. 研究の動機

大手ソフトウェア企業では多くの種類のソフトウェア製品を開発している。そのため，各製品に対して，それぞれ開発チームが存在している。各チームはそれぞれに独立の製品を開発してはいるが，共通の課題を持っており，類似の技術（知識）を必要としている。そのため，知識が効果的に移転されれば，企業としての開発効率は上がることが期待できる。しかしながら，筆者の所属する ICT 企業である A 社のソフトウェア開発部門において，実際に各チーム間で知識交流を行い，他のチームから知識を導入しているケースは限られている。この結果，類似技術を複数のチームが個別に開発することになり，企業としては重複開発となっている。

ソフトウェア製品は技術者がプログラミング言語を用いて書いたソースプログラムをソフトウェア（コンパイラ）によって自動的にマシンコードに翻訳することで製品となる。そのため，ソースプログラムとソフトウェア製品（マシンコード）とは完全に対応している。つまり，実質的にソースプログラムが知識及び製品の最終形態と等価である。また，ソースプログラムはソフトウェアの動作を完全に形式的に記述したドキュメントであるため，そのまま形式知[1]として移転することが一見容易と考えられそうである。このような好条件にもかかわらず知識共有がなされない理由を解析し，知識共有できる条件を探ることは，ソフトウェア開発の効率向上につながるものと考えられる。

1.2. 研究の背景

ソフトウェア知識の移転は，開発したチーム（知識提供側）が自ら開発したソフトウェア（ソースプログラム）を知識を必要とするチーム（知識受領側）に開示するだけでも成立する。この場合，知識提供側は自らのために開発したものを知識受領側に開示するだけで作業完了となるので，提供のコストはほぼゼロとすることができる。それにもか

[1] 明白で「形式的・体系的な」知識（野中 & 竹内, 1996, p.8）

かわらず，たとえ同一社内であってもソフトウェア知識の提供を拒絶，あるいはためらう事象が発生している。

一方，知識受領側は開示されたソースプログラム（知識）が自らの用途に合うことを確認し，場合によっては（多くの場合）改造してから適用することになる。ソースプログラムにはソフトウェアが動作するために必要不可欠な情報は全て含まれているはずである。しかし，「なぜこのような処理をしているのか」のような理解を助ける背景知識は含まれていない。そのため，自然言語で書かれたドキュメントを理解することと比べて，読解には技術力と時間を多く要する。さらに，提供された知識にはバグが含まれていることが多いため，不用意に利用するとトラブルが発生することもある。そのため，活用には注意が必要になる。しかし，このソースプログラムを読解し理解するためのコストは自ら開発する場合と比べれば少なくなることが期待できる。一方，バグによるトラブルのリスクは自ら開発する場合でも同様に存在するので，受領時特有のものとは言えない。そのため，知識を受領すると開発コストが下がることを期待できる。このように利益が期待できるにもかかわらず，ソフトウェア知識を受領しようとする行動には，現実にはためらいが発生しているように見受けられる。

本研究では，このように完全な形式知と言えるソースプログラムが存在するものの，そこにはバグが含まれている可能性が高く，また理解には困難さが存在するというソフトウェア独特の性質を前提として，知識移転の促進要因および阻害要因を分析し，知識移転を活性化させる方法を検討する。

1.3. 研究の意義

この研究では，ソフトウェア知識の移転・知識共有に着目している。一般の知識共有に比べて，本節で述べるようなソフトウェア開発の特性に着目することで，ソフトウェア開発組織に対して有効が知見を得られるものと考える。ここではまず，ソフトウェアという知識の特性と，それを開発する組織における知識共有の特性について記述する。

1.3.1. ソフトウェア知識の特性

本研究においては，ソフトウェアという知識の特性を理解したうえで事例を分析する。ここまでにも述べてきたようにソフトウェアはソースプログラムで記述されている形

式知である。しかし，それを利用するためには記述の背景としてのプログラムの動作環境を理解したり，ソースプログラムを記述するにあたっての留意点を理解したりすることが必要になる。この背景情報はドキュメントになっていないことも多く，その場合，オリジナルの記述者の暗黙知[2]になっている。受領者は暗黙知を提供者から引き出すか，あるいはソースプログラムを読んで自ら理解することが必要である。

また，バグの存在はソフトウェアにつきものと言える。知識の受領者は利用にあたってバグによるトラブルに見舞われることもある。また，利用しようとしたモジュールの動作がプログラムの実行環境に依存している場合，提供者から見るとバグではないが受領者にとってだけはバグと同様になることもある。

ソフトウェア知識の特徴について以下の 4 点を確認する。

(a) 形式知の要素

ソフトウェアはプログラミング言語によって記述されたソースプログラムによって記述されている。一般にソフトウェアの開発者はソースプログラムを記述する前に自然言語で記述された設計ドキュメントを作り，それに基づいてソースプログラムを記述する。ソースプログラムはそのまま自動的に実行可能なプログラムに変換される。そのためソースプログラムは製品と 1 対 1 に対応する，完全な形式知と言える。

(b) 暗黙知の要素

前述したようにソースプログラムは「完全な形式知」ではあるが，技術者にとって受領したソースプログラムを理解するのは容易ではない。システムソフトウェアの場合，100 万行を超えることも珍しくない[3]。ソフトウェアが大規模になればなるほど理解が難しいことは容易に想像できる。

ソースコードだけから理解することが困難な場合，設計ドキュメントによって補うことを期待するが，多くの場合それだけでは不足する。

不足のもっとも大きく重要な要因は，ソースプログラムに書かれた論理の背景や考え方が記載されていないことにある。ソースプログラムにこのような知識を

[2] 「目に見えにくく表現しがたい暗黙的な」知識（野中 竹内, 1996, p.8）

[3] オープンソースプログラムとして重要で広く普及しているオペレーティングシステムである Linux の場合, 2003 年時点で 200 万行を超え（Hertel, Niedner, & Herrmann, 2003), 2012 年では 1500 万行を超えている（Leemhuis, 2012)。

埋め込む余地として注釈があるが，一般に多くを期待することはできない。ソフトウェア知識の開発者（提供者）の頭の中にある考え方，論理の背景は暗黙知としてそのまま提供者が保持している。受領者がソースプログラムを設計ドキュメントの助けを借りながら理解しようとするとき，過去の経験などの受領者自身の持つ暗黙知を利用して論理の背景を自ら構築し，理解を進めようとする。つまり，提供者と受領者はそれぞれに暗黙知を生み出しながら，一方はそれに基づいてソースプログラムを記述し，他方はそれをソースプログラムから理解しようとする行動を取る。

(c) 不完全性（包含されているバグ）

ドキュメントからソースプログラムを記述するにあたって，技術者は自己の技術力を駆使して変換作業を行う。しかし，入力（ドキュメント）と出力（ソースプログラム）は完全に対応するわけではない。多くの場合ドキュメントは量的に不十分であり，その上誤りも含まれている。それでも技術者がソースプログラムを作り上げることができるのは，設計ドキュメントを開発者自らの技術力と他者とのコミュニケーションによって補うことができるからであると言われており (Jones, 1999, pp.74-76)，ドキュメントの不完全性を示している。

また，殆どのソフトウェア製品には何らかのバグが含まれている。バグは，設計ドキュメントに含まれている場合とドキュメントからソースプログラムを作成するときに組み込まれる場合の両方がある。例えば機械製品の場合には「あそび」と呼ばれるようなゆとりを持って設計し，振動に耐えられるような作りをしている。これに対して，ソフトウェアは極めて論理的な製品であるためこのようなゆとりは存在せず，部分的なバグは必ず製品のバグとなる。しかし，バグが含まれているソフトウェア製品であっても，利用に際してトラブルになるとは限らない。バグによるトラブルの発生条件が限定されるので，長い製品寿命を通じて発覚することのないバグもある。製品開発のプロセスの中のテスト工程において利用シーンを想定したテストが十分にされていれば，その範囲においてはバグによるトラブルが発生しないからである。

しかし，ソフトウェアモジュールを他の用途に流用した場合には，元の製品とは違う使い方をすることになる。そのため，元の製品の開発チームでのテストだけでは不十分であり，そのまま利用すれば内在しているバグがトラブルを引き起こすことも多くある。

(d) 動作環境への依存性

ソフトウェア工学では，ソフトウェアをモジュール化し動作環境への依存性をなくすことが推奨されている。適切にモジュール化されているソフトウェアは再利用する際に部品として容易に切り出すことができ，他の環境（製品）に移植する際には部品だけを確認すればよい。そのため知識移転が容易である。一方，モジュール化が不十分な場合には，部品が依存する環境についても理解しないと再利用（知識移転）できないため，知識移転が困難である。実際には，ソフトウェア部品が必ずモジュール化されて動作環境から独立できているとは限らない。開発者の技術力に依存するところではあるが，部品として使おうとしたモジュールが動作環境に依存していて，容易には活用できないことはよくある。

活用しようとした時点で動作環境に依存しているために動作しないのであれば問題は小さい。しかし，活用できたと思っていたにもかかわらず，出荷後になって発生したトラブルが動作環境に依存するバグによるものだと判明することもある。つまり，動作環境への依存性が高いソフトウェアモジュールは，バグによるトラブルを誘発しやすいと言える。

各モジュールの動作環境への依存性はないことが望ましいが，まったくないということは現実には考えられない。どこからが「環境」でどこまでがモジュールのインタフェースなのかということに明確な線が引けないからである。特に，開発者の技術力が低いときは野放図にインタフェースを広げて環境依存性が高いモジュールを作ることが多い。

知識利用者は, モジュールのインタフェースを確認して環境依存性に対応する。しかし，ソフトウェアが大規模で全体を理解することができないときには環境依存性を見抜くことができず問題になることもある。

1.3.2. ソフトウェア開発組織の傾向

ソフトウェア開発組織内の技術者間の関係について, 以下の 2 点の特徴が考えられる。

(a) 知識移転の相互性

ソフトウェア開発組織における知識の提供者と受領者は，ともにソフトウェア技術者である。そのため知識移転に当たって，あるときには技術者 A から B に知識を提供（移転）するが，別のタイミングでは B から A に提供する可能性

がある。組織内のソフトウェア技術者は技術力の差はあってもそれぞれに専門性を有しており，一方的な技術移転になるとは限らない。つまり，A,B 両者は give and take の関係になる可能性が高い。一旦知識提供をしておくと，次の機会に知識を提供してもらえるという関係の成立を期待することもできるであろう。

このように，提供者，受領者はともに同様な環境におかれていることがソフトウェア知識移転の特性であり，相互依存関係に着目することが本研究の意義の一つである。

(b) 知識の共有と抱え込み

企業内で知識を共有することは企業全体から見れば損になる話ではない。実際に知識共有がうまく進んでいる事例は多く報告されている[4]。しかし現実には知識を抱え込み，同僚や他のチームから秘匿しようとするケースも多く見られる。また，単純に秘匿するわけではなく提供するか秘匿するかを悩み，行動をためらう場合も見受けられる。

これまで，知識共有(knowledge sharing)の対立概念として，知識抱え込み(knowledge hoarding)が考えられてきたことに対し，積極的に抱え込むまでには至らないためらい(hesitation)という中間状況を考え，ためらいから共有に動くための要因を考えることも本研究の意義の一つである。

1.4. 研究の目的とリサーチクエスチョン

本論文は，組織マネジメントと従業員のモチベーションを考慮したうえで，知識受領側の要件，知識提供側の要件それぞれの面から知識移転の動機とためらいの状況を明らかにするものである。

本研究の第一の目的は，ソフトウェア知識移転のうち提供側が持つ動機とためらいを実証的に明らかにするものである。ソフトウェア知識の移転（提供）に関して，移転の素材となるソフトウェア開発・管理の手法については，ソフトウェア工学によって研究が進められてきた。ソースプログラムをソフトウェア知識として理解しやすくするため

[4] たとえば，（黒瀬, 2005）は富士通のアプリケーションプログラム開発部門における知識創造環境を報告したものである。

の技術的な検討もソフトウェア工学分野で古くからなされてきた。特に，あらかじめ部品とする目的でソフトウェアを開発することについては実践も進んでいる（Jacobson, Griss & Jonsson, 1999)。しかし，あらかじめ他者に利用させることを目的とせずに開発したソフトウェアを開発後にあらためて他者に提供することについてはソフトウェア工学においてもこれまで十分に研究されていなかった。また，知識提供者を動かすモチベーションについての研究も十分になされていない。

これに対して，本論文では，提供者のソフトウェア知識提供にまつわる技術的な配慮やコストとそれに関わる作業のモチベーションがどのような業務環境の下で高められ，あるいは低下させられるのかの知見を得ることを目的としている。ソフトウェア開発は人による知識創造活動であり，ソフトウェア技術によって知識共有の道具（ツール）を作っても最終的に知識提供者が知識をその仕組みに乗せなければ何の意味もない。そのため，本論文では知識共有の道具以外の業務環境にも考察の範囲を広げる。

第二の目的は，ソフトウェア知識移転のうち受領側が持つ動機，あるいはためらいの要因を実証的に明らかにするものである。知識受領に関しては，知識の性質とその移転の難易度に関する研究（Szulanski, 1996)や受領能力に関する研究（Cohen & Levinthal, 1990)がある。しかし，受領者は知識を受け取って利益を得る立場であるにもかかわらず，その受領（活用）をためらうケースがあることについてはまだ充分な研究がなされていない。

本論文で対象とするソフトウェアの知識移転においては，大部分の知識はソースコードの形で流通できる。そのため，これまでに述べてように，ソフトウェアの知識は形式知として移転，複製が容易でり，第一の目的の項で触れた知識共有の仕組みが用意されていれば，受領側もそれを使って容易に知識を得ることができるはずである。しかし，たとえ仕組みがあっても知識受領者もこのツールを活用せず，自ら開発することが多い(Tracz, 2001, p.124)。受領側の活動が停滞している原因に，ツール以外の業務の環境が影響しているものと予測できる。業務環境に着目し，受領者のモチベーションを高めるための要因を分析することが本論文の目的である。

研究の目的をまとめると以下のようになる。

a. 知識提供側における促進，阻害要因と業務環境との関係を見出す。
b. 知識受領側における促進，阻害要因と業務環境との関係を見出す。

また, この目的を達成するために, 次のリサーチクエスチョンへの解答を求めていく。

MRQ：どういう組織環境が社内のソフトウェア資産（ソースコード，以下『知的資産』）の活用（ソースコードとして改造，取り込み）を促すか？

SRQ1：自ら開発した『知的資産』をチーム外に開示する行為をためらわせる要因は何か？
SRQ2：他者の『知的資産』の活用を推進する要因，ためらわせる要因は何か？
SRQ3：活用されやすい『知的資産』の性質とそれを運用する業務環境とはどのようなものか？

MRQ は知的資産の共有・活用に向けた環境を探り出すことで，知識共有を進める要件を見出すというこの研究の目的に合致させている。

そのために，SRQ1 で知識の提供者の行動に対しての問いを立てた。自ら開発した知的資産を他者に提供したくないという心理も理解できる。一方，技術者はオープンソースソフトウェア（OSS）の活動に見られるように，自ら開発したソフトウェアを他者に開示し使ってほしいという気持ちや仲間（他者）と共有したいという気持ちも持っている。特に，提供する気持ちはあるものの何らかの理由で提供行動に出られないというためらいの気持ちの要因を見つけ出す。

SRQ2 では知識の受領者に対する推進，阻害の要因を探る。特に，無償での知識受領は，受け取った方が一方的に得をする行動である。それにも関わらず受領をためらう理由を解明する。

SRQ3 では SRQ1, 2 で解明しようとする人間の行動ではなく，知的資産そのものの性質に着目して，共有，活用に向く知的資産を取り巻く環境について考察する。

1.5. 研究の方法

本研究のために，まず，コンピュータシステムを開発する国内 A 社のシステムソフトウェア開発部門を内部から観察した。さらに同部門に所属する上級エンジニアに対して，ソフトウェア知識共有に関心がある者を募り，「知識共有促進コミュニティ」を結成した。ここで知識共有の要件について初期の議論を行い，ソフトウェア開発部門全体に対して質問票調査を行うことを決定した。引き続き，このコミュニティにおいて質問票の作成も行った。

次に，このコミュニティでの議論の結果をもって，部門全体に対して質問票調査を実施した。質問票調査の結果は, 因子分析とその結果に基づくパス解析によって分析した。さらに知識受領の成功体験を持つ技術者に対しインタビューを行い, 知識受領の際の行動や周囲との関係について調査した。

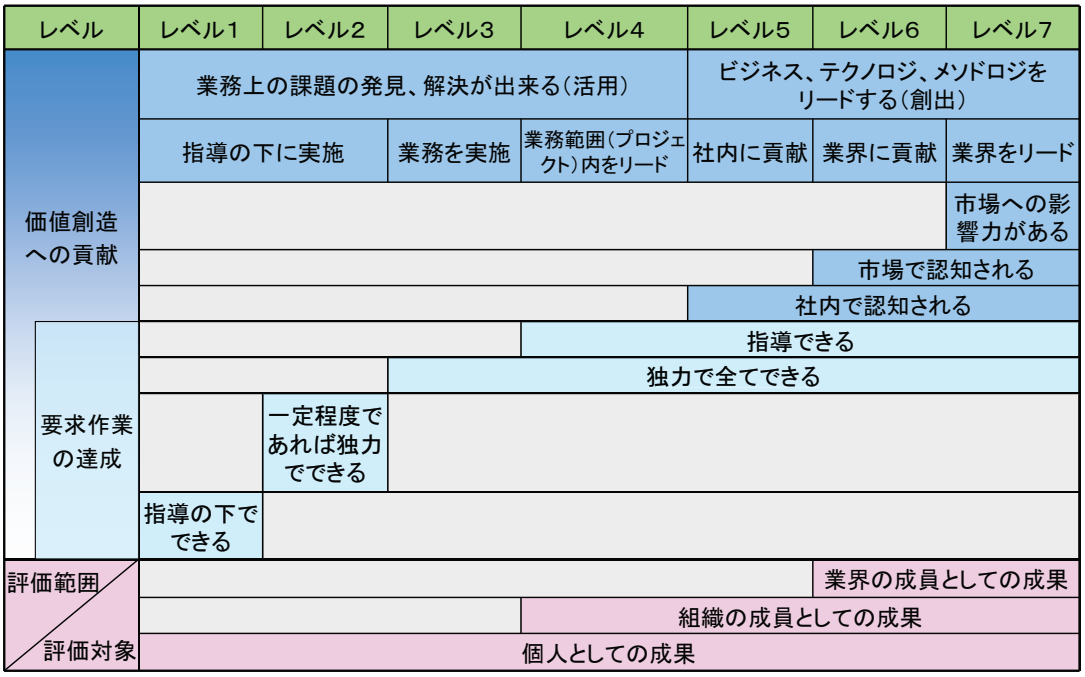
同開発部門内のチームおよび個人相互間で行われる知識共有を対象としたので，1.4 で述べた研究目的の a, b ともに調査対象，構成は同一である。それぞれの技術者はあるときは提供者になり，別のときは受領者にもなりうる。

同開発部門は，独自の基準で従業員の技術力認定を開発力，マーケティング力，技術サポート力の 3 分野で行っている。この認定は（情報処理推進機構, 2012）によって定められた「IT スキル標準」を参考にしている。表 1-1 は情報処理推進機構による技術レベルと評価の概念を示したものである。本論文では調査対象の部門の運用に基づき，上級をプロジェクトをリードするものとしてレベル 4 以上, 中級を自らの判断で業務を実施できるものとしてレベル 3 , 初級は上級者の指導が必要なものとしてレベル 2 以下としている。

本研究にあたっては，技術力認定により，3 分野のどれかで中級以上と認められている 2054 名を対象に調査を行った。なお，開発力の分野で中級以上とならなくても，他の分野で中級以上の者も調査対象とした。これは，どれかの分野で中級以上と認定されていれば, 知識創造にどのように取り組むかを考えていることが期待できるからである。また，（開発力以外の）どれかの分野で中級以上の認定を受けると，開発力の分野での認定を取らなくなっているという実態も考慮した。

2054 名中，実際に回答を寄せた者は計 441 名，開発力のレベルから見ると上級 94 名，中級 246 名，初級 101 名であった。

表 1-1 レベルと評価の概念



（情報処理推進機構, 2012, p.31）より 図 15

開発プロジェクトの中ではそれぞれ数人から十数人程度のグループを構成しているため，個人の行動様式は所属するグループの行動様式に影響される。しかし，一方で，ソフトウェア開発は個人への依存性が強い業務であるため，同一グループと言っても必ずしもすべての行動が統一されるわけではない。たとえば，知識の提供は個人よりもグループの資産を提供することになるため，グループで同一の行動を取ることが予想できる。しかし，知識の受領については，受領した知識を自らの開発に生かす行動は個人の活動に依存する。受領を良しとしないメンバー（個人）は知識を受領する環境におかれてもそれを使わずに自ら開発を進めることになる。つまり，知識受領についてはグループで行動が統一されないことも多い。

なお，調査対象の開発部門では，これまでに述べてきた「グループ」は公式な組織単位（一般企業における部や課のような）として作られておらず，「グループ」のトップは管理職でない場合もある。そのため，調査においてもグループの違いを明確にすることはできず，個人ベースでの調査とした。

1.6. 論文の構成

本論文は以下の構成となっている。

第 2 章では，本論文の対象とする分野であるソフトウェア知識の移転について，ソフトウェア工学及び知識科学の両面から先行研究をレビューする。まず，ソフトウェア工学においてソフトウェアの再利用およびソフトウェア開発プロセスに関する研究についてレビューする。次に，ソフトウェアに限定せずに，知識移転の方法および組織のマネジメントとモチベーションに関する研究についてレビューする。これにより，本研究の背景とソフトウェア知識の移転の難易度，特徴を明らかにして，本研究の位置づけを明らかにする。

第 3 章，第 4 章，第 5 章ではそれぞれ対象組織に関して視点を変えながら調査・分析を進め，報告，検証する。各章の関係を図 1-1 で示す。第 4 章で知識提供行動，第 5 章で受領行動それぞれの分析を行い，その背景となる組織マネジメントについての観察結果を第 3 章で示す。

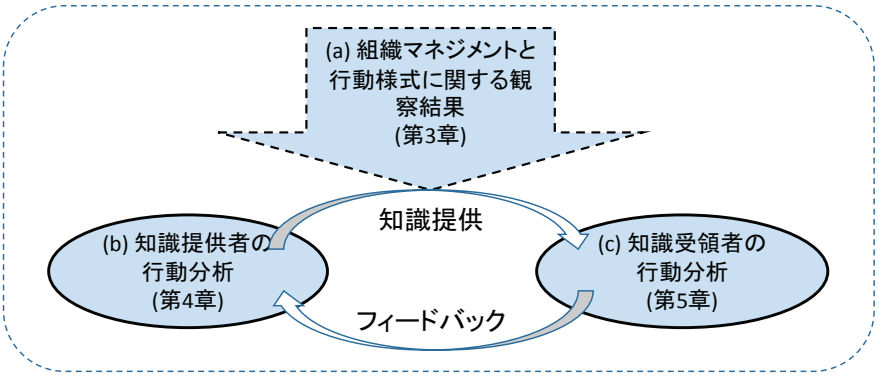


図 1-1 組織分析の視点

まず，第 3 章では，対象としたソフトウェア開発部門に対して，組織マネジメントの方針とイノベーション組織の行動様式について，筆者が内部から観察した結果に基づき考察を加える。この章の考察結果は知識提供と受領の背景としての環境に対する考察で

あり，後の章の分析でも活用される。本論文で対象としたA社は過去に組織運営の方針や人事評価の制度を大きく変更した経験を持つ。過去にはあいまいとも言える制度で運営していたが，変更後は厳密なルールを用意し，損益管理やジョブディスクリプションを明確にした組織運営に切り替えた。明確な運営ルールで管理した場合と，あいまいなルールの下でエンジニアリングチームを自由に行動させた場合とで，知識共有及びイノベーションへの取り組みにどのような違いが生じたのか，事例をもとに検証する。この検証を第4章以降の知識提供，受領の行動との関係の分析につなげる。

次に，第4章では，知識提供側に焦点を当てる。知識を提供しても，提供者や提供側のチームが直接的に利益を得ることは少ない。一方で，自己のチームのために開発した知識（設計ドキュメントやソースプログラム）を他者に提供することは，紙やデータを渡すだけで済むのであれば，提供自体には手間（コスト）が掛かるものではない。このような条件における知識提供・受領のプロセスにおいて，提供側にどのような負荷がかかり，知識提供をためらうことになるのか。あるいはどのような条件が加わると知識提供を進んで行うことになるのかをデータを元に分析する。

最後に，第5章では，知識受領側に焦点を当てて，ソフトウェア知識の受領をためらう理由，受領した知識の活用形態について分析する。知識のやりとりにおいて，知識受領者は特に負担なく知識を得て自らの開発に活用することができる受益者のはずである。しかし，単純に受領して活用できるわけではなく，受領した知識を活用するには技術力（吸収能力）と作業時間が必要となる。そのため，受領者の技術レベルによって行動が異なることが予想できる。その上で，受領条件やマネジメント形態などとの関係を分析することで，受領を促進するための要件を見出す。

さらに，質問票調査に基づく定量的考察に知識受領の成功経験者を対象としたインタビュー結果を加えて，知識受領行動がどのような動機によって推進されたのかを議論する。この議論の中で，組織マネジメントが知識の提供及び受領の行動に与える影響について解析する。組織の規則として与えられる知識提供，受領に対して外発的なモチベーションおよび，組織環境がメンバーに与える内発的なモチベーションの存在が行動に与える効果を考察する。

第6章では，本章でたてたリサーチクエスチョンへの解答を整理し，そこから得られる理論的含意，実務的含意を纏める。最後に，本研究で残した今後の研究課題について言及し，論文を締めくくる。

第2章 先行研究のレビュー

2.1. はじめに

本章では，本論文の位置づけを明らかにするために，関連する先行研究についてソフトウェア工学の観点と知識科学の観点から整理する。

2.2. ソフトウェア工学分野の先行研究

2.2.1. ソフトウェア再利用に関する研究

ソフトウェア知識の移転は 1968 年にソフトウェア工学(Software Engineering)という言葉が使われて間もなく「ソフトウェアの再利用」として検討され始め（Buxton & Randell, 1970)，現在に至るまで同分野で継続的に検討されている課題である。この分野では，(1)モジュール化，オブジェクト指向等プログラムの記述方法の研究の中でプログラムの部品化を進めることによってソフトウェアの読解性，信頼性を高めること，(2)プログラム資産の提供形式を標準化することで再利用を促すこと，の 2 点に着目したソフトウェア製品開発技術に関する課題を追求している。しかし，ソフトウェア開発の組織の在り方や開発者（提供者，受領者）の行動については特に論じてはいない。また, ソフトウェア開発におけるソフトウェアの記述方式等の知識の提供側の技術に集中することになっており, 受領側の要件への追究はほとんどなされていないのが実情である。

ソフトウェアモジュールの再利用は Dijkstra (1970)により構造化プログラミングが提唱されて以来，ソフトウェア工学の分野において長く研究されてきた。ソフトウェア工学はソフトウェアの生産性を高めることを目標に発展してきた。当初はバグが少なく（高信頼性），改変や保守をしやすくすること（保守性）で生産性を高めるため，ソフトウェアの記述方法として構造化プログラミングが提唱された。その後，生産性を高めるために再利用が有効という観点から，可読性および周辺独立性が求められることでモジュール化が推奨され，さらにオブジェクト指向プログラミングへと発展してきた（米

澤, 1984)。これを踏まえながら，ソフトウェア記述方法の発達やソフトウェア資産共有のためのインフラ整備が推進されてきた。

Basili, Briand & Melo (1996) らは学生に対して利用すべきクラスライブラリ[5]を提供し，ソフトウェアを再利用することで生産性，品質（信頼性）の変化を検証し，再利用の効果を確認した。その結果，部品をあるがままに使う場合だけでなく，改造を加えて利用する場合においても，自力で開発した場合と比べて生産性，品質両面で効果が上がっていることが確認できた。Jacobson, Griss & Jonsson (1999) はソフトウェア部品供給についての体系を纏めている。

しかし，Tracz (2001) は技術や環境が整備されたとしても，現実のプログラミングにおいては一筋縄では共有・再利用は難しく，現実の世界ではなかなか共有が進んでいないことも指摘している。共有が進まない理由としては，プログラマの視点からは「自分でやる能力がないと見られるかもしれないから」，「最初から作り直す方が，既存の成果物を活用するように設計をあわせるよりも安く上がるという，プログラマの思い込みがあるから」「コンポーネントを探したり，再利用可能部品でシステムに組み上げたりするツールがないから」のような技術的というよりも心理的な要因を挙げている。また，認知心理学者の視点からは「理解できないものをプログラマは再利用できない」とも指摘している。「習熟したプログラマは，プログラムを読みやすく理解しやすくする名前付けやプログラミングスタイルに関して，ある種明示的な「発話規則」に従っている」として，再利用の要件を自ら満たそうとすることを述べている Tracz (2001, pp.123-127)。

2.2.2. ソフトウェア開発，生産性と品質に関する研究

ソフトウェア開発の品質に関しては，実例からドキュメントやレビューの質，量に関する定量化されたデータが報告されている。Jones (1998) はまずソフトウェア規模を定量化する指標として，機能的尺度 FP（Function Point）を導入した。これはソフトウェアの機能面から入力，出力，論理ファイル，照会，インタフェースの 5 要素の重み付けの和によってソフトウェアの規模とする尺度である。行数で測る方法と比べてプログラミング言語の違いやプログラマの技術力の違いに関わらず一定の値をとる指標となっていると主張されている。その上で，米国におけるソフトウェア開発プロジェクト

[5] オブジェクト指向プログラミングにおけるソフトウェア部品群。ここではオブジェクト指向プログラミング言語である C++によって記述されたソフトウェア部品群である。

を調査することにより，ソフトウェアの規模あたりのコスト，品質，品質低下の要因などを明らかにした（Jones, 1999)。

ソフトウェア開発組織は生産性を高め品質を確保するために様々な試みを実施している。一時期，日本においても「ソフトウェア工場」「ソフトウェアファクトリー」と呼ばれる取り組みが業界に広がった。しかし柔軟性が低いことから成功したとはいえない。ただし，Cusumano (2004, p.202) は，類似システムを複数バージョン作る必要があるようなビジネス形態では機能すると述べている[6]。また，Cusumano (1989) は多くの場面で効率的に使えるモジュール（部品）を開発することの困難さについても述べており，知識共有の難しさを確認している。

2.3. 組織と知識移転に関する先行研究

2.3.1. イノベーションを起こす組織についての研究

ソフトウェア開発は組織によるイノベーション活動である。本論文ではソフトウェアの知識共有を行ってイノベーション活動を行う手段を解明することを目的としている。そのために，イノベーションを起こす組織についての先行研究をレビューする。

野中らは組織における知識創造のモデルとして SECI モデルを提唱した（野中 & 竹内, 1996), (Nonaka, Toyama & Konno, 2000)。特に暗黙知に注目し，SECI の中での共同化（Socialization）で暗黙知を高めるプロセスを明らかにしている。また，企業を場の動的な構成と見て, 企業が相互作用により知を生み出す場所であるという主張がなされている（Nonaka, Toyama, & Nagata, 2000)。ここで場とは物理的な空間を示すのではなく「そこで行われる多元的な相互作用」（野中, 遠山 & 平田, 2010, p.60）であるので, 共有空間であってもバーチャルな空間であっても知識が共有, 創造されればよい。また，場は「自己組織化」，「価値観の共有」，「異質な知を持つ参加者」，「浸透性のある境界」，「参加者のコミットメント」の５つの要素によって活性化される（野中, 遠山 & 平田, 2010, pp.67-73）ことを示した。

加護野（1985）の研究では，伝統的な組織理論においては必須の管理手段であると考えらえている計画の策定とその遵守を組織の創造性には阻害的な影響を及ぼすものとして捉えた。この研究においては創造性のために強いインパクトを与える 4 つの次元と

[6] 本研究ではシステムソフトウェア開発のチームを対象としており，類似システムを複数バージョン作るものには当てはまらない。

して，実験主義，内発的動機付け，ゆさぶり，異質情報の交配を見出した。この 4 項目は伝統的な組織理論では組織を混沌状態に陥れる乱暴なやり方として考えられる。しかし，ここでは，混沌状態に対して，自然な秩序形成能力があることを仮定し，混沌状態の効果を説明している。

Granovetter (1973) は個人間のつながり（紐帯）について，時間の共有度合や背景情報を共有している度合によって紐帯の強さを表現した。そのうえで，弱い紐帯においては異なる背景を持っていることから，新たな知識を導入する機会をもたらしやすく，強い紐帯では知識を育てるのに適していることを示している。

一方，Hansen は紐帯の強さと知識移転について，弱い紐帯は有用な知識を発見することを加速するが，弱い紐帯を通じて複雑な知識を移転することはできない (Hansen, 1999) と指摘した。さらに知識発見と移転のコストを分けて測定することにより，知識共有を遂行するには強い紐帯を活用することが有効 (Hansen, Mors & Lovas, 2005) であることを示した。

Ghoshal, Korine & Szulanski (1994) は松下および Phillips の海外子会社間のコミュニケーションの事例を挙げて研究し，それぞれの子会社の自律性よりも子会社のメンバー間の個人的な関係の方がコミュニケーションに強く影響することを示した。

Davenport & Prusak (2000, pp.180-213) は「仕組まない自然発生的な知識移転が，企業の成功にとって重要だ」と述べている。これは形式知を移転するための手段や規約の整備よりも，自然発生的に知識交換を起こすための戦略を練ることの重要性を述べている。さらに，私的な会話を通じた知識移転の有効性を指摘し，冷水器のまわりや談話室での非公式な会話を最も重要な仕事の形としている。特に，日本の例としてロータス・ノーツのようなリポジトリの運用がうまく行かず，直接顔を合わせる会議[7]によって情報を共有している事例を挙げながら，組織や国の文化にあった知識移転の方法を使うことの有効性を指摘している。

Lipman-Blumen & Leavitt (1999) らはイノベーションを起こす組織の特徴として，組織の規範よりもプロジェクトの遂行に集中する（組織内の）チームを「ホットグループ」と名付けた。ここではプロジェクトの遂行に関心を集中させる一方で，成功のためには組織の規範を軽視し，違反も厭わない行動を取る。成功のためにはチーム内の人間関係を悪化させることも厭わない。企業のマネジメントからは支援されない行動であり

[7] 会話の場は，公式の会議だけでなく仕事後の酒席等を含む

時には潰されることもあるが, 実績を上げることで規範の変更も含めてホットグループが生き残っていく。

Dyer & Nobeoka （2000）はトヨタおよびグループ企業による組織学習の事例に基づき，グループ企業のネットワークを使った知識共有の実現の状況を示した。トヨタのグループ企業ではネットワークメンバーの調整原理を用意してメンバーのアイデンティティを保持することを調整原理としたうえで，メンバー間の知識共有，フリーライダーの防止, 形式知と暗黙知両方の知識移転を実現している。さらに, 強く結合したネットワークは知識探索よりも普及，活用に適していることを示した。しかし，トヨタのアプローチにはネットワークが内向きになり, 多様性がなくなるリスクが存在することも指摘している。

Czepiel （1975）はイノベーションを起こす業界においては競合する組織間での非公式なコミュニケーションが活発に行われていることを示した。

原田（2000）はネットワークポジションによってコミュニケーションの頻度が変わることに着目して研究した結果，職位，年齢，組織在籍期間，技術的能力が類似していても技術的コミュニケーション頻度は上がらないが, 同一グループにいるメンバー間でのみコミュニケーション頻度が高くなることを示した。その上で，コミュニケーションをできるタスク分業のグループを構成することを訴えている。

2.3.2. 知識移転のコスト，方法及び評価に関する研究

知識移転のコストには，知識の性質，受領者のスキル，知識の提供元と受領者との関係性などの要因が関与する。

知識の性質については，Szulanski は，知識の粘着性(Stickyness)の概念を用いて分析（Szulanski, 1996) (Szulanski, 2000)し，環境に依存し，粘着性の高い暗黙値を移転するには大きなコストが掛かるとしている。Hansen, Nohria & Tierney (1999) は知識を形式知に変換したうえで再利用を進めるコード化戦略(Codification)と一人ひとりの専門分野を結びつけて暗黙知を共有できるようにする個人化戦略(Personalization)の概念を用いて，戦略を選択しているコンサルティングファームの実態を明らかにした。

受領者のスキルに着目した Cohen & Levinthal （1990）の研究では，知識の吸収能力(Absorptive Capacity)は多様化したスキルに依存すると指摘し, 多種類の経験がある受領者はそうでない者と比べて受領が容易であることを示している。

Reagans & McEvily (2003) は，提供者と受領者の関係性については，特に知識の暗黙性が高いときには紐帯が強ければ知識移転は容易になりやすいことを指摘している。この論文では同時に，形式知は弱い紐帯の下で効率的に移転されることも述べている。

Hinds, Patterson & Pfeffer (2001) は初心者を教育するにあたり，熟練者よりも初級者の方がうまく行く事例を上げている。この結果は，熟練者は彼らの知識を抽象化して教えようとするのに対し，初級者は基本的な知識のみを提示するので初心者が受け入れやすいためであると結論付けている。

Dixon (2003) は知識移転を「連続移転」（チーム内での繰り返し），「近接移転」（異なる場所にいる同じような業務を行うチーム間の移転），「遠隔移転」（非定型業務に関する暗黙知を他のチームに移転），「戦略的移転」（非常に複雑な知識を２チームの間で移転される），「専門知移転」（頻繁に行わない業務の形式知を移転）の 5 種類に分類して，それぞれの性質を明らかにした。

Borgatti & Cross (2003) は知識や情報を探すに当たって，人が何を知っているかを知っていること，人が知っていることの価値を見極めること，その人の思考へのアクセスができることの 3 点の価値を指摘している。

Leonard & Swap (2005) は「組織とマネージャ個人に大きな優位をもたらす知識」を「ディープスマート」と呼んだ。この知識はノウハウに近いものであり，専門的でシステム全体を把握する力のあるものであり，これを移転することによって組織の中の知識ギャップを埋めることを目指した。大規模なソフトウェアにディープスマートを当てはめると，ソフトウェア全体を把握する力であり，プログラムに直接記述された知識の読解のみならずその背景となる知識を理解して活用する能力に相当する。

Brown らは概念的な知識については，知識に文脈依存性があることを指摘している。ここでは，知識は行動や文化，文脈の中に置かれているのであり，文脈無依存を前提とした教育方法には成功例が少ないことを指摘している（Brown, Collins & Duguid, 1989)。さらに，組織としての学習は非公式なコミュニティによって進められていることを明らかにした（Brown & Duguid, 1996)。

また，Argote & Ingram (2000) は知識移転を客観的に評価することの困難を指摘している。これは，特に暗黙知において，どんな知識を得たのかの調査が直接的にはできないことからの指摘である。知識受領者本人さえも受領した知識を認識できず，ただ業務のパフォーマンスが向上したことのみが見えてくる。そのため，業務のパフォーマンスの向上を測定することで間接的に知識受領を測定している。実際に知識は組織内の（1)個人，（2)業務，（3)ツールのどれかあるいはこの３者の組み合わせに蓄積される。

特に，組織学習の結果について病院の事例を研究し，who knows what という形で経験を通して組織に蓄積されることを示した（Reagans, Argote, & Brooks, 2005)。

2.3.3. 知識移転の動機に関する研究

IT を使った仕組みを用意することで，知識共有・移転の促進を狙うことは多く行われている。一方で，仕組みがあっても人が使おうとしなければ知識共有は進まない。ここでは仕組みを有効に使うことも含め，知識移転の動機に関する研究をレビューする。

Smith & McKeen (2003) は，知識共有は決して自然な行動ではないので動機付けが必要になることを指摘し，主として知識提供側に視点をおいて知識共有の動機について論じている。動機付けには組織としての認知が有効で，褒賞を与えるほどのことはないが，知識移転していることを認知していることが示されればよいとしている。一方，知識移転ツールのような技術的な支援策は，組織としての認知行動ほどには効果がないことも指摘している。

Hall & Sapsed (2005) は知識を共有するか自分のところに抱え込むかについては，組織の持つインセンティブとモチベーションの構造に依存して決まると述べている。また Cabrerra & Cabrerra (2005) は個人に対するよりもチームに対するモチベーション向上が有効であるという研究を発表しており，個人に対する外発的[8]なモチベーションだけでは知識共有は促進されないことが示唆されている。堀江らも，日本の電機系製造業の研究所における知識提供行動について調査した結果として，内発的な動機付け[9]が作用すること，その内発的なモチベーションを高めるのは自律性であることを指摘している（堀江, 犬塚 & 井川, 2007)。

Thompson らは心理的契約，コミットメント及び知識の共有の関連を解明し，特に感情的コミットメントが心理的契約，知識の共有，イノベーションを促していることを指摘した（Thompson & Heron, 2006)。

Bock ら（2005）は，組織の制度と知識共有の動機付けの関係を指摘した。また，Osterloh & Frey (2000) は組織構造の変更によって暗黙知移転の内発的動機を促進することを述べている。

---

[8] 外発的（な動機付け）とは「『望ましい行動に報酬を与えよ。そうすればその行動が繰り返される可能性が増す。』という単純な原理」（Deci Flaste, 1999, p.22）に基づく直接的な報酬を与えることによる動機付けを指す。

[9] 内発的な動機付けとは「まるでその活動をすること自体が『報酬』であるかのような」(Deci & Flaste, 1999, p.23) 現象による直接的な報酬の提供をしない形での動機付けを指す。

オープンソースプログラム(OSS)の開発におけるモチベーションについての検証は Bitzer, Scherettl & Shoroeder (2007) によって，外発的および内発的な動機付けの視点からなされている。ここでは，特に経済学で無視されがちな，遊びの楽しさや贈り物の文化のような内発的な動機付けによって知識提供が推進されていることを指摘している。

また, Lakhani & von Hippel (2003) はOSSのサポートがボランティアによるコミュニティによって行われていることに着目した。ここで，サポートの提供側（回答者）にとっては容易な質問に答えるだけで済むので負担が少ないうえに, 他者の質問を読むことが回答者のスキルアップにもなるというインセンティブがあるから成り立っていると報告している。このインセンティブは提供側にとって自己実現につながるものと考えられる。

一方，受領側については Davy (2006) は，受領者本人は対象とする知識の評判やわかりやすさを見て受領行動に出ることを決めることを指摘している。受領者は知識を自己の手元に入手しても，活用するには理解するためのコストが必要になる。そのため, あらかじめ価値があるかどうかの判断をする基準として, 評判やわかりやすさが利用されているということである。

知識の提供は提供者にとって単純かつ短期的な損得勘定では損[10]と考えられるので, 企業全体の収益や人間関係といった目に見えにくい利点を考えるようにしないと積極的な行動にはなりにくい。それでも実際に知識提供が行われているケースが存在している。大規模な例としては, 企業内の事象ではないが OSS のコミュニティが挙げられる。OSS の開発は開発者による世間一般への知識提供になるが，Lakhani らはその動機として, 一般に考えられそうな外発的な動機 － 知識提供行動がキャリアとして認められることにより職が得られたり昇進できたりするなど － よりも, 単純に創造的な活動が楽しいという内発的な動機によるという研究結果を報告している（Lakhani & von Hippel, 2003)。また，OSS のコミュニティに対して使用方法やトラブルに対する質問をすると，多くの場合，見ず知らずのボランティアからの解答が得られる。これについては, 質問の内容は回答者にとって簡単であるため負担にならないことを大きな理由としているが，そのほかに，他者の質問を読むことが回答者にとって勉強になることが多いこと（Lakhani & Wolf, 2005）も理由としてあげている。いずれの事例でも知識提供者は報酬を求めず, 自己実現の欲求を満たす行動として知識提供がなされているとされ

10 自分は短期的には得をすることがなく，他社を利するだけと考え勝ちであること

ている。つまり，OSS の場合には，報酬等の外発的な利益を期待できなくても，自己実現の欲求を満たす環境があると，知識提供者のモチベーションが上がることが示されている。

2.4. まとめ

ソフトウェア工学分野においては，ソフトウェアの開発技術に着目して，再利用しやすいソフトウェアの開発方法の研究が進んできた。これは構造化プログラムからオブジェクト指向プログラムに至る発展であるが，あらかじめ再利用することを目的としたプログラム開発技法ともいえるが，ソフトウェアの再利用が必ずしも成功していないという現実も指摘されている。ソフトウェアの規模と付随するドキュメント量，レビューコストは知識移転のコストを評価するのに有用な指標である。

知識科学分野での研究に対しては，組織，移転コスト，動機についてレビューを行った。イノベーションを行う組織は組織体制や規範に縛られずにダイナミックな動きをしており，SECI プロセスが有効に働いていたり，ホットグループが活発に活動している。

知識の性質とともに，提供者，受領者双方のスキルや関係性が知識移転のコストに影響する。環境に依存する暗黙知は粘着性が高く移転が困難である。

知識移転の動機についての研究では，内発的なモチベーションを保つことの重要性を多くの先行研究が述べている。外発的なモチベーションの有効性は観察例が少なく，組織としては知識移転の活動を認知していることを示すことが有効である。

本研究で対象にしているのは，主に他のチームが開発したソフトウェアの一部をモジュールとして切り出して部品として使うケースである。ソフトウェア工学の議論は部品を作る側に立っており，部品の切り出しを含む知識移転は本研究の新しい切り口と言える。また，知識移転に関する研究例は知識提供側の観点からの研究が多く，受領者の視点からの動機やためらいに関する研究例は少ない。ソフトウェア知識の特徴と関連させた研究として，本研究の独自性が明らかになった。

第3章 A 社マネジメントと知識創造との関係

3.1. はじめに

本論文では，知識移転を促進あるいは阻害する要件として，本論文では図 3-1 に示す３点に注目している。特に，本章では同図内 (a)で示す組織環境について理解することにより，知識共有に対するためらいとの関係性への指針を得ることを目的とする。そのために A 社ソフトウェア開発部門の組織マネジメントの特徴について，企業の成長期とそれ以降の違いに着目して調査・報告し議論する。

3.2. 知識移転の着目点

初めに着目するのは図 3-1 中(a)の組織運営のしくみが知識移転に向けた行動に与える影響である。知識移転に際して，組織として知識移転の制度や，環境がメンバーの行動に与える影響についての考察を行う。組織内の指示系統や情報流通が制度化されていて，そのルールに従った行動が強く意識されている場合と，行動規範が文書化されておらず自然発生的な人間関係に基づくコミュニケーションに頼って知識移転がなされている場合とでは，知識移転の実態は異なるはずである。

次に，図 3-1 中(b)に示す知識提供側の動機を考察する。知識提供者は提供の方法によっては何のコストも払わずに知識（ソースプログラム等）を開示することができる。しかし，同一企業内にいる人間関係の中で受領者から説明を求められれば，拒絶することは難しい。そのための提供者は説明のためのコストをあらかじめ予測しておくことになる。説明するために多くのコストがかかると判断すれば，提供そのものをやめ，知識を秘匿することもある。また，同一の企業内とはいえ自己の知識を無償で提供することに多少の抵抗を感じることは想像に難くない。この（小さいかも知れないが）知識提供をためらう気持ちの本質は何か，あるいは，提供者がどのような考えを持って知識提供を行うのかを解明することは，知識共有を推進するうえで避けて通れないものと考える。

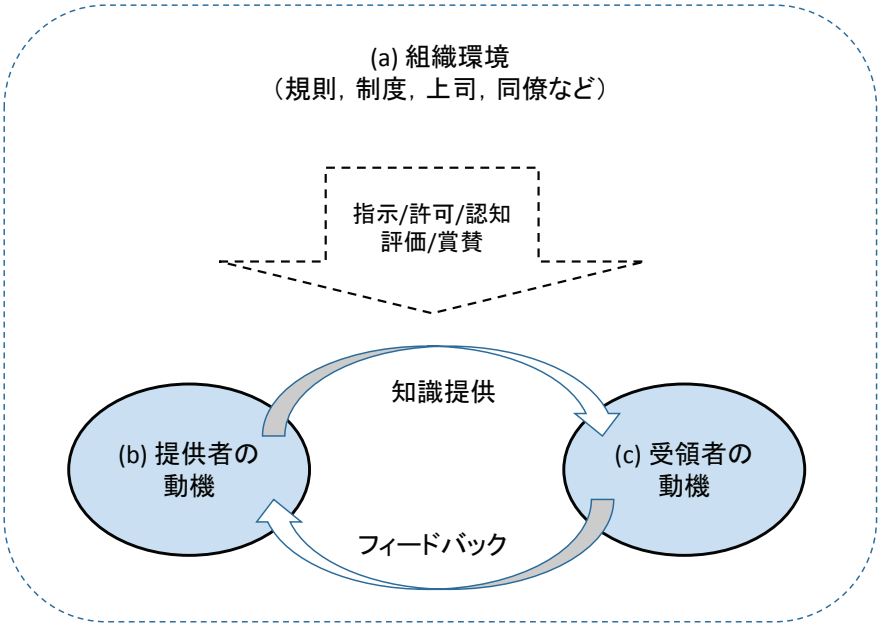


図 3-1 知識移転の注目点

最後に，図 3-1 中(c)に示す知識受領側の動機に注目する。知識受領側は，知識移転によって代償なく利益のみを享受するはずである。しかし，実際には受領側が積極的に知識資産を受領しようとしない場合が多い。そこで，知識受領をためらう理由を調査し考察を加える。

受領側が開示された知識を活用するには，受領側の責任で受け渡される知識を理解しなくてはならない。そこには知識の吸収能力（Cohen & Levinthal, 1990)を必要とする。対価を払う必要がないとしても受領に伴って発生する（内容を理解するという）労働コストがかかる。そのため，自分で開発した方が早いという考えを持つこともある。それが受領をためらう理由となる。本研究においては，他者の知識をうまく吸収，活用して，すべてを自ら開発するよりも低コストで高品質のものを作り出すことができることに期待している。実際の知識共有，移転の成否は，吸収能力に依存する受領のための労働コストと開発コストの削減のバランスによって決まることになる。

以上のような議論に基づき，本研究ではソフトウェア開発業務に取り組むチームにおいて，業務環境や組織のマネジメントとの関係を踏まえながら知識共有はどのような動機によって促進されるのか，あるいは，どのような状況がその阻害要因となるのかの知見を得ることを目的とする。特に，知識の提供側と受領側に分けて観察，考察を進める。

3.3. 分析対象部門の構造

分析対象は同一の企業（A 社）のソフトウェア開発部門に属する複数の「チーム」である。チームは一般企業における「部」レベルの組織単位である。各チームはそれぞれ独自に製品開発を行いそれぞれ収益責任を負っている。つまり，各チームがかけたコストはそこで開発した製品を販売することによって回収することが求められている。

各チーム内には課レベル以下の複数の「グループ」が存在する。そこではチームの活動を分担して推進している。個々の技術者はどこかのグループに所属して活動している。技術的な行動はこのグループのリーダー（多くの場合，課長クラスの管理職）の指揮のもと，個別のグループ単位で行われている。グループとしての収益責任はないが，チームに属するためチームの収益に貢献することは意識している [11]。

知識のやり取りは提供者と受領者とのあいだで行われる。提供者と受領者はそれぞれ異なる製品 A および B を開発するチームに属している。両者の開発する製品相互には外面上の類似点は殆ど見当たらない。しかし，その内部で使っている部品（モジュール）レベルの技術では多くの共通点が存在している。知識の受領者は高度な技術知識を部品として自らの開発する製品に取り入れることで，効率的に高レベルの製品を開発することに知識受領の動機がある。その際に，上司の指示や上司からの評価への期待（または不安）が作用する。

一方，提供者は自らが開発した知識を提供するだけでは直接的な利益を得ることはできない。しかし，知識を提供し受領者がそれを活用する結果，受領者からフィードバックを得て自らの知識（ソフトウェア製品）を向上させられる期待が動機となりうる。このとき，自らの属するチームの知的資産を外に出すので，上司が許可を出さなければ行動が頓挫するであろう。一方，上司は提供の許可を出すだけでなく，提供した部下に対して組織への貢献を理由として高い評価を与える可能性もある。その場合には，提供者にとっての外発的な動機付けとなる。

[11] 組織構造についての詳細は「付録 3　A 社の組織構成と役割」に記した。

また，図 3-1 の活動に対して，外部での評判が高まったり賞賛が得られたりするとしたらそれは提供者，受領者双方の活動のモチベーションになるであろう。なお，この図に描いた「提供者」は別の場面では他の製品開発チームから部品を受け取る受領者になることもあるし，受領者が提供者の役割を果たすこともある。

3.4. 組織マネジメントの方針と知識移転行動の関係

この節では， A 社の基本ソフトウェアを開発する部門についての観察結果をまとめる。この組織は 1980 年代までの高度成長期から日本経済におけるバブル崩壊に始まる低成長期に転換するのと時を同じくして，マネジメント方針を大きく転換させた。それと同時に技術者たちの行動も変わっていった。本節では対象企業の組織マネジメントの変遷と知識移転行動の変化が及ぼした知識受領・提供の行動への影響について筆者の観察に基づいて考察する（Hotta, 2009)。

はじめに，A 社が順調に成長を続けてきた時期と転換後の組織マネジメントについて考察する。次に，この転換期における米国シリコンバレーの IT 企業について先行研究に基づいて検討する。シリコンバレーの IT 企業は A 社と比べると若い企業が多いが，成長し大企業と呼ばれるようになった後でも研究開発チームの活性を保っている。最後に，両者の違いを解析し，A 社において知識共有を促進するための方策について検討する。

3.4.1. 高度成長期末期における日本型組織

まず, 1990 年頃までの A 社のマネジメントの特徴としてあいまいな組織運営を上げ，そこで自然発生的に形成された階層的な場について述べる。

(1) あいまいな組織運営

1990 年代初頭までの A 社においては，組織運営の規範や理念および損益目標は各組織の末端まで浸透させていなかった。従業員は業務に取り組むにあたって，企業や自部門の収益の追求を強く意識することはなく，同僚や上司を含むコミュニティの持つモチベーションを追求する傾向が強かった。そのために，組織内の人間関係を円滑にした組織運営を行うことによって，企業と従業員の関係を家族的なものにすることが見られていた。つまり，あいまいな規範の下で人間関係を円滑にしながら業務を遂行していたの

である。これはA社のみならず，日本国内の企業においては珍しくはない企業ガバナンスの形態だったと考える。

A社の組織のあいまいさは特に以下の２点に発現していた。一つ目の事象は人事評価である。年功序列の比重が大きいとはいえ，能力や業務の実績を踏まえた＋α部分の人事査定がなされていた。しかし，＋α部分についてどのような能力や実績が評価に結び付いたかは密室で決まり従業員に明らかにされることはなかった。従業員は何をすれば高い評価が得られるのかについて明確に知らされることがないまま，業務を遂行し評価を受けていた。つまり，期待される業務が何かは上司あるいは組織と本人とのあいまいな関係性のもとで各従業員が理解し，実行していたと言える。

あいまいさのもう一方の発現は業務の定義にある。ジョブディスクリプションを明確に定義することなく，従業員同士がお互いに業務の重なりを持った状態が自然発生的に出来上がっていた。この結果，特に能力の高い従業員が自分の仕事の枠を広く持ち，さらに困っている同僚に自発的に手を差し伸べることは珍しいことではなかったし，むしろそれを期待されてもいた。また，組織としても弱点が発見されると迅速に弱点の補強に乗り出すことができた。この結果，組織内の協力体制が取りやすく「できる者が処理する」文化が形成され，定型的な作業の生産性は高まっていた。あいまいな人事評価がなされていることも手伝って，助けた側がその行動をアピールすることもなく，結果的に効率的な組織運営に寄与する結果を生じていた。

一方で，企業としての意思決定スピードが遅いという問題も生じていた。ジョブディスクリプションがあいまいなため，責任の所在が不明確になっていたのである。責任の所在が明確でないため組織としての判断は常に必要以上に上位層まで上げられることになり，意思決定に時間が掛っていたものと考えられる。

企業が目覚ましい成長を遂げていた時代には，あいまい性を強く持ったマネジメントシステムは，組織内の軋轢を生まずに業務を遂行する運用形態として有効に機能してきた。つまり，組織あるいは上司から具体的に何を期待されていて，自らの「本業」は何かを明らかにしないまま，組織や上司の意向を慮った上で，各従業員が自発的に業務を進めていたことが効果を上げていたことになる。「明確にしないマネジメント」が企業活動を円滑に進める結果に繋がっていたと考えられる。このマネジメント環境に適合するように自然発生的に存在したのが以降で述べる「階層的な場」である。

(2) 階層的な場

あいまいな環境の下で，A 社内のソフトウェア開発組織には階層的な場が形成されていた。筆者の経験では，競合他社の打ち出した新機能に対抗できるソフトウェアの開発に当たって，複数のチームのメンバーが集まり対策を考えた。各チームのソフトウェアには直接的な関連はなかったが，対面の議論を通じて共通の問題を認識し，それぞれの知識を高め，各チームはそれぞれ競合相手に対抗する製品を開発することができた。

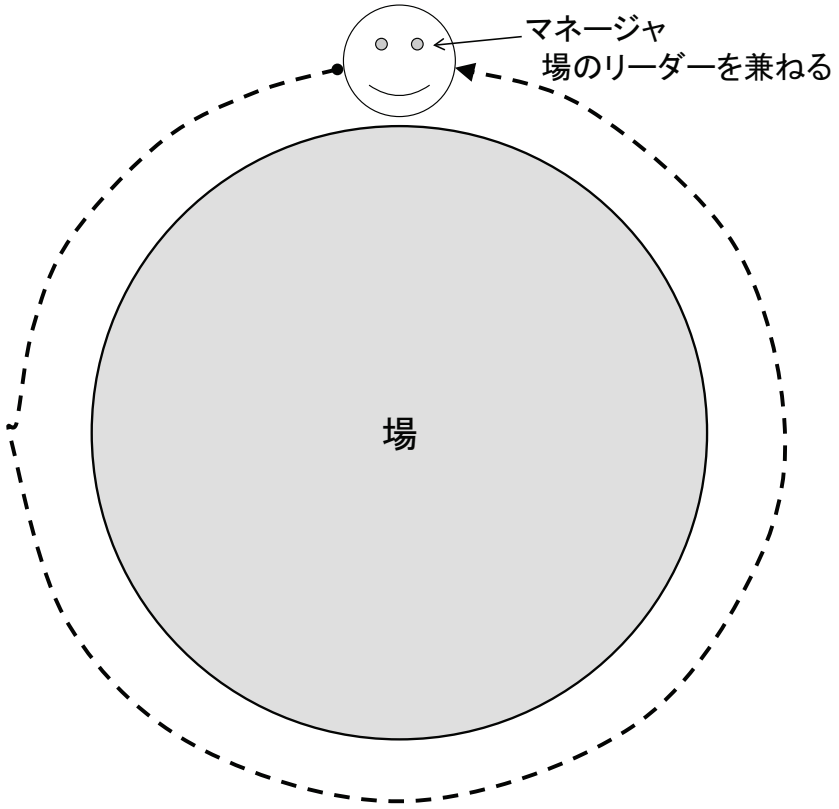


図 3-2 低次の場とマネージャ

まず，組織構成の最下層の開発グループの中に(低次の)場が存在し，活発なコミュニケーションを取りながら開発作業を遂行する。この行動は必ずしも効率的ではなかったが，世界的な巨人とも言うべきライバル社に追い付くという明確な全社目標への共感，長時間残業を厭わず（あるいは暗黙的に強制される），結果だけでなく努力に対しても敬意を払うマインドによって後押しされ，「不夜城」とさえも言われるハードワーク文化

が成立し，結果を出すことができていた。この場においては，メンバーの自発的な行動が促されていた。一方，マネージャはチームに対するマネジメントの役割と，外部に対するチャネルの役割を持っていた。チャネルの役割として，他チームとの交渉の切っ掛け作りなど経験と人脈を生かしたゲートキーパーの役割を果たすことが期待されていた(図 3-2)。

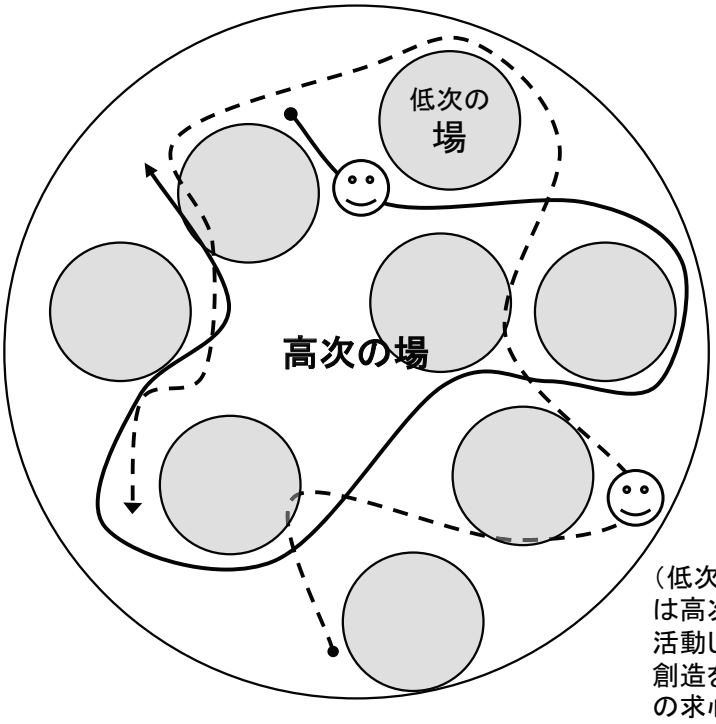


図 3-3 高次の場（階層的な場）

さらに，図 3-3 に示すように低次の場をメンバーとするような「高次の場」が存在することで，階層的な場が形成されていた。高次の場の活動主体はメンバーである低次の場のリーダーであった。リーダーたちは，まず，メンバーである低次の場をつなぎ合わせる役目を担っていた。高次の場は各リーダー同士の交流によって成立する。リーダーの部下である低次の場のメンバーたちは，高次の場の活動に直接関与しないこともあるが，リーダーを通じて高次の場に貢献していた。さらに，リーダーは自らの率いる低次の場を統率しながら，他の場のリーダーと交流しながら新たな知識を創造し，自らの場に取り込む活動を推進していた。

高次の場においては，そのメンバーである低次の場が目標を共有し，ゆるく結束して活動していた。メンバーは個別の利害を主張することなく，for the team, for the company の意識を持ち，共有された目標に向かって自発的な活動が行われていた。その結果として，知識移転，知識共有および，作業協力がスムーズに行われていた。高次の場は最前線の低次の場をメンバーとして弱いつながりを持ち，公式な組織の枠組みを超えて技術情報が流通し，イノベーションに繋がる結果となっていた。グループ・チームをまたいだ高次の場においては，知識の多様性があり，高次の場のメンバー組織として知識の結合が得られた。この活動が各メンバーのイノベーションの源泉となっていた。ここでは，野中，遠山 及び 平田（2010, pp.67-73）による場の活性化要因 5 要素：「自己組織化」，「価値観の共有」，「異質な知を持つ参加者」，「浸透性のある境界」，「参加者のコミットメント」のうち，「浸透性のある境界」以外の 4 要素が成立していたと言える。

利害の主張をせずに活発に活動できたのは，組織運営のあいまい性の結果と考えられる。つまり，社内の組織間でのコスト負担者意識や手柄を主張する意識が薄いことにより，自然な感情として場の仲間を助け，助けられる関係ができていたのであろう。もし，各組織（多くの場合低次の場の母体である組織）のコスト負担や提供物の権利の帰属を強く意識するようになると，それぞれの協力を期待する場の活動が停滞することは容易に予想できる。

低次の場では，一般に公式組織のマネージャ（課長級の管理職）あるいはマネージャに指名された者がリーダーとして活動していた。一方で，高次の場はあいまい性を強く持った環境であり，各メンバーの役割やリーダーは公式には存在しない（そもそも高次の場そのものが公式の存在ではない）。そのため，場合によっては（実質的な）リーダーが何らかの事情で場を離れた後，継承者を立てることができずに瓦解する危険性も持っていた。ただし，活発な活動を続けられる高次の場では，メンバーとなっている（低次の）場のリーダーたちが活発に活動することによって，高次の場の活動の方向性を統一するための接着剤の役割を担っていた。そのため，高次の場のリーダーが不在になったとしても，場のアクティブなメンバーである低次の場のリーダーたちの活動が求心力を生み，活動が存続することも可能である。まさに，「明確にしないマネジメント」の産物と言える。

場，特に高次の場が存在し活動するためにはあいまい性の強い組織運営は有効な環境であったと言える。必ずしも自分の属するチームの収益に直結しない活動，他のチームのみが有利になる活動であっても，明確なジョブディスクリプションが規定されない環

境であり，提供した知識の対価を請求することも成績評価を気にすることもないからこそ，この活動はスムーズに進んだものと考えられる。もし規範が明確な組織運営をしていれば，協力する側も協力を受ける側も打算が働き判断が遅れ，簡単には場の活動を進めることはできなくなったものと推察できる。

3.4.2. 現在の対象組織のマネジメント

「バブル好況」が終了し日本経済は危機に陥り自信を失った。これとほぼ同時にA社の位置するITマーケットは，これまでのメインフレーム中心から「ネットワーク化」「オープン化」「ダウンサイジング」の波によりビジネス構造が大きく変化した。経済の停滞とマーケットの変革という二種類の圧力にさらされA社の組織運営も変革を迫られた。代表的な変化として，人事評価に於ける目標管理制度とビジネス意識の強調の２つがあった。

(1) 人事評価に於ける目標管理制度

年功序列は日本の人事制度の特徴のように言われるが，その時代にあっても人事評価が行われていなかったわけではないことは誰もが認めるであろう。ただし，これまでに述べてきたように評価の基準はあいまいであった。そのため，評価される側からは各自の行動が評価とどのように結びついていたかは明確になっていなかった。

マネジメントの変革によって取り入れられた目標管理制度は，期初に目標を定めそれにしたがって期末に評価を行う制度である。つまりジョブディスクリプションの明確化と達成度の通知が組になった運用を前提とした人事評価制度である。

この制度によって，人事評価が同僚との競争であるという面が（実際には従来からそうであったが）強調されることになった。また，評価される側には自らの仕事とあらかじめ定義されていないことは評価対象にならないという懸念が生じ，同僚を助けたり自らの知識を提供したりすることに対して消極的になったり協調性がなくなった（江波戸, 2002, pp.81-82）と言う声が出ている。

(2) ビジネス意識の強調

ジョブディスクリプションのあいまいさとともに，旧組織管理の下では各組織単位のビジネス意識もあいまいであった。特に，ソフトウェア製品はハードウェアの添え物的に扱われ，大きな収益対象とは考えられていなかったこともあり，ソフトウェア開発部門ではビジネス意識は希薄であったと言われている。

マネジメントの変革により，ソフトウェア開発部門でも収益意識を強く持たされるようになった。その結果，ソフトウェア部門全体としてのビジネス意識だけでなく，末端組織にまでその考えが強調され浸透することになった。つまり，各部門の損益を厳しく管理し，ビジネスユニットの意識を明確したことで，組織の長は部下の行動を損益の観点から管理するようになった。自己の業務のビジネス性を強く意識し，それに基づいた行動を取ることは営利企業の活動として当然のことではあろう。しかし，その一方で社内の他部門の活動への貢献は部門の収益に繋がらない活動として見られるようになり，活動が停滞するようになった。明確にしないマネジメントの環境で活き活きと活動していた 3.4.1(2)の高次の場が動きを弱め，活動が停滞し始めたのである。

3.4.3. 日本企業の改革

A 社に限らず，他の日本企業も経済危機に瀕し，組織マネジメントを変革させた。この変革において多くの日本企業は「欧米風」の運営を取り入れた。この変革行動は，日本風企業運営は古くて間違っているものであり，近代的な「欧米風」が唯一の解であるかのようにも捉えられた。しかし，この変革はフロントラインの活動に新しい問題を引き起こした。変革が形式的，表面的なものにとどまり，組織運営の根底を変えるものになっていないことが，その原因と考えられる。

企業・組織文化や従業員の持つマインドは欧米と日本で大きく異なる。欧米風のマネジメントを日本に移植するには，日本向けのカスタマイズが必要である。例えば，日本の品質管理の成功はデミングによって紹介されたシューハート理論を日本文化に適合するように慎重にインプリメントした結果であった（藤野, 2010)。これは海外で実施していたものを単純に輸入したのではなく，米国生まれの方法を日本の環境の中で実現し，熟成させたものということである。

ビジネスユニットを明確にしたことで，自部門の損益に関する責任が明確になり，各部門（事業部）及びその所属従業員のビジネスマインドが高まった。しかし，一方で部門の損益への関心が高まり過ぎて，全社の利益を意識することが減り，他部門との協力体制は弱いものになるなど，サイロ化を進める副作用も生じた。A 社が先駆けとなりその後各社が続いた目標管理制度も同様で，個人レベルの目標達成が最大関心事となり，チームに対する協力的な活動が停滞したり，リスクの高い業務にチャレンジする精神が薄れてしまったりという弊害が指摘されている（江波戸, 2002, p.82)。

上記の結果による最も大きな変化は高次の場の衰退であった。個人や個々の部門単位の利益を追求する結果，他の部門との協調関係の前提には自部門の収益が据えられて

いった。その結果, 他部門に対する無償の協力を進める高次の場は機能できなくなった。高次の場においては各メンバーである低次の場の接着剤として機能していたマネージャは, 一転して他部門からの割り込みを撥ね付けるガードとして機能するようになった。さらに，高次の場が衰退したことで，低次の場を超えた知識共有は困難になった。複数のチームで知識共有ができなくなると, 各チームは同種の知識創造を重複して実施することになり，業務効率が低下する結果となっている。

このような弊害は，日本企業がすでに持つ文化の尊重をせずに，新しい制度を適用させようとしたことに原因があるものと考えられる。あいまいな組織運営と階層的な場の組み合わせは典型的である。他にも，例えば，実態として終身雇用制になっている社会において, 年功序列は適合したシステムであり, これを覆して若年者を昇進させる場合, 過去の地位を逆転させることへの配慮が必要であり, 成果評価の制度への転換は様々な弊害を生み出す可能性がある。

3.4.4. シリコンバレー企業におけるイノベーション組織

米国において活発な企業活動を続けているシリコンバレーの ICT 企業の状況も当該組織の状況を考察するうえで参考になる。

シリコンバレーの企業におけるイノベーションを起こす組織に見られるホットグループ（Lipman-Blumen & Leavitt, 1999）の特徴として挙げられているのは，ホットグループではミッション遂行のためには手段を選ばないことである。つまり組織の規範がミッション遂行に取って邪魔になるようなときには規範を破ることを厭わない。ミッション遂行のために，議論によってグループの和を乱すことを厭わない。そのため，上位の組織から見ると許容しがたい行動になることも多い。特に，企業の規範が既に定まっている大企業において，たとえミッション遂行のためとはいえ，ホットグループの存続は組織として難しい。しかし組織の規範と戦ってでもミッション遂行に取り組む行動を最終的には黙認する（あるいは制御をあきらめる）ことで迅速なイノベーション活動が進められる。この中でグループの和, 組織の和という感情は薄い。一方で, ミッション遂行をよりどころとしているホットグループはミッションが終了すると解散することになり，寿命は短く知識の蓄積は難しい。

ベンチャー企業のスタート時には企業そのものがホットグループになっていることが多い。古い例では Sun Microsystems の例（Southwick, 1999），最近の例では Facebook の創業の顛末などは典型的な事例（Kirkpatrick, 2011）として紹介されている。しかし，安定した大企業の中においてはホットグループは許容しがたい行動を取る

ものと捉えられ，上位組織によって潰される危険性も指摘されている。日本においても，富士通がコンピュータ事業を開始する頃の池田敏雄の行動は組織の規範に外れ周囲との軋轢も省みないものであったと伝わっている。池田に巻き込まれたその部下，池田の行動を支援したその上司など周囲も含めてホットグループであったと言えるだろう(富士通飛翔編集室, 1999)。

チームのミッション最優先でイノベーションを追求するシリコンバレー流のホットグループは，A 社における高次の場の行動と似ている点がある。A 社の高次の場では企業内の組織のミッションを超越した行動，つまり自部門が遂行すべき業務の範囲を超えて他部門との協力関係を作る行動を取ることでイノベーションを推進していた。

一方，Lipman-Blumen らは日本企業はホットグループを抑圧してきたとも述べている。短期的なミッションのために活動するホットグループと何年もの時間を掛けてゆっくりと慎重にコンセンサスを得ていく日本企業における場とは「寿司とホットドッグ」ほど違うと述べている（Lipman-Blumen & Leavitt, 1999, p. 37)。

両者の大きな相違点は，その寿命と考えるべきであろう。確かに，メンバーの和を重視するかどうかは両者の間の大きな相違点であるが，それは階層的な場，ホットグループの寿命の長さに関与している。メンバーの和を重視した A 社の低次の場では企業内の組織の枠をはみ出さず，長期的なイノベーション活動を遂行していた。その結果，プロジェクトが終了しメンバーが新しいプロジェクトに移っても階層的な場は形を変えながらも存続することができる。一方，ホットグループのような（プロジェクトの）ミッション遂行に重点を置いた行動形態の場合には，メンバーの和は軽視され，プロジェクト終了後はホットグループも解散する。

ともに非公式な活動を進めることでチームの知識創造を推進しているという共通点をもち，本来の組織活動の規範とは異なる規範に基づいている点も日米共通と考えてよいであろう。一方で明確なミッションを持ちチームの和を乱すことも厭わず，プロジェクトが終了すると解消されるホットグループと，社内の知識共有の場として必ずしも自チームの利益に結びつかなくても活動し，和を重んじる A 社の階層的な場の間には本質的な違いが認められる。和を重んじる環境になるとホットグループは育ちにくいと言えるであろう。

3.5. まとめ

本章において，階層的な場に代表される組織環境と知識共有の関係について検討した。そのために，まずA社における「場」の構造を分析した。成長期には階層的な場が存在し，特に上位の場のメンバーたちは公式の職制として所属する各チームのミッションに関わりなく他チームから参加しているメンバーを助ける行動が見られていた。この行動はあいまいな組織運営や明確にしないマネジメントに後押しされ，活性化されていた。上位の場を通じて社内の複数のチーム間で知識共有を進め，イノベーションの原動力になっていたと考えられる。しかし，マネジメントの改革により，目標管理制度やビジネス意識の強調が進むと同時に，掲げられた目標の達成や部門毎のビジネス推進に直接役立たない行動がなされにくくなって行った。その結果として階層的な場のうち上位の場が衰退し，知識共有行動が硬直化するようになった。

A社の知識共有行動の硬直化は高次の場の衰退に原因があると考えるのが妥当であろう。その背景には「欧米風」のマネジメントの単純な導入にあると考えられないだろうか。階層的な場はミッションのあいまい性を許容する文化において成功していた行動様式であった。あいまい性を排除し，組織単位でのビジネスマインドや個人に対する厳格な評価制度を導入したときに，高次の場が衰退するのは納得が行く。突然あいまい性を排除してトップダウンで組織を動かそうとしたことで，むしろ官僚主義に陥って知識は各チームに閉じて持たれるようになり，イノベーションがし難い組織になってしまったものと思われる。

階層的な場とあいまいな組織運営は深い関係がある。あいまいと言っても野放しに経営できるわけではなく，適切な「さじ加減」が経営のノウハウになっている。企業の再生・発展のために，一度あいまい性の排除を試みた組織が，再び適切な「さじ加減」を要するあいまいな組織運営に戻すことは難しいし，従来のような階層的な場を復活させることは短期的には困難であろう。あいまいな運営を適切なレベルに調整するのは短期間には困難と考えるからである。あいまいな運営を断念するならば，部門毎の収益を強く追及しながら，あいまい性を前提とした高次の場は使えない。そのため，「A社版ホットグループ」あるいは「『新』階層的な場」を構築し，チーム間協調を促すことが必要になってきた。

あいまい性を排除し，階層的な場が成立しにくくなってしまった環境において，部門間の協調，知識共有を進めるためには何らかの外発的な動機付けを行い，インセンティブを生み出すことが必要であろう。たとえば，部門間の契約や協調活動を部門の収益に

換算するシステムを立ち上げ，運用が容易なようにすることなどである。ここでは，簡単な手続きによって，契約や収益換算ができることが重要で，換算が正確である必要はない。単に協調作業にかけた工数（人月）を計算するだけでもよい。単純で，むしろ不正確でもあいまいな方式が協調関係を作りやすいのではないか。従業員にとって大きな負担がなく，受け入れられる程度の変革をし，その上で各部門のマネージャがビジョンを語り部門の目標を共有することで，「A 社版ホットグループ」を作り出すことができるのではないだろうか。将来的にあいまいな方式が拡大すると元の階層的な場が復活することも期待できる。

本章で挙げた階層的な場，あるいはホットグループは，本論文でこの後，第 4 章，第 5 章で解析する知識提供行動，知識受領行動の動作環境として大きく関与している。それぞれに対する関与については，各章にて述べることとする。

第4章 知識提供行動の分析

4.1. はじめに

本章では，図 1-1 (b)で「知識提供者の行動分析」とした，自らが開発したソフトウェア製品に含まれる知識を同一企業内で別のソフトウェア製品を開発している同僚に対して提供する行動が，どのような状況によって活性化し，あるいは提供をためらうことになるのか，A 社ソフトウェア開発部門のデータを分析する。

言うまでもなく，知識の移転は提供側と受領側がともに活動することで成立する(van den Hooff & de Leeuw van Weenen, 2004)。特に同一企業内での知識移転に当たっては，非公開の知識を秘密裏に探り出す産業スパイのような行動は考える必要はない。そのためまずは知識の提供があってから移転が始まると考えればよい。知識受領者は提供された知識を自らの業務の中で活用することができる。一方で，知識提供者は知識移転が成立しても短期的な利益を得ることはないという非対称性に留意する必要がある。このような状況において，短期利益を得られない提供者の動機を分析することが，知識移転の促進要因を解明するために有効であると考える。

4.2. 知識提供行動の分析の目的と意義

ソフトウェア知識はソースコードという形で形式化されている。しかし，ソースコードを単純に提供しただけで，受領した側がその内容を十分に理解することは容易ではない。その理由は，(a)ソースコードにはソフトウェア実行の論理のみが記述されていて[12]，なぜその論理が正しいかなどの解説にあたる事柄が記述されていない。(b)ソフトウェアを設計するにあたっての背景知識が記述されていない。の 2 点が挙げられる。ソフトウェア知識の移転においては，多くの場合受領者はソースコードを自己のソフトウェア製品に組み込むために改造する。また，改造しない場合でも，動作確認や製品品質に関

[12] 問題の回答として数式だけが記載されていても，問題の解法を理解し他の問題に応用することは困難であること，あるいは，問題集の回答だけ書き写しても問題を理解することができないのと似ている。

する責任の一端を受領者が負うことが普通である。そのため，内容を理解できていない状態では受領者は知識を活用することができない。つまり，受領者には受領したソースコードを理解する必要がある。そのため，受領者がソースコードだけからでは理解が不十分と感じた場合には，提供者はソースコードの説明を文書や口頭で受領者に対して行うことが要求され，さらに，受領者からの質問が来ることも多い。

上記の理由から，知識提供者は受領者に対してソースコードの説明に手間を費やすことが要求される可能性が高い。一方，知識を提供することで提供者は直接的な利益は得られない。それどころか，知識を提供することで個人の優位性を他人に渡すことになり，自己の相対的な価値を減らし人事評価を落とすことになりかねない。もちろん，企業全体としての利益を考えれば，同僚に知識を提供することによって，企業の競争力や業績が向上し，めぐりめぐって自分の利益にもつながる。また，自分の持つ知識を同僚に提供することで，自己の存在感を高めたり人間関係が豊かになったりするなどのモチベーションはあるであろう。

組織に属するエンジニアにとって，組織での活動は収入を得る正式な業務である。オープンソースの活動のモチベーションとの類似性は期待できるとしても，直接の収益を得ないオープンソースの活動とは必ずしも一致しないであろう。特に，知識提供が新たなコストの発生（説明の手間など）や自己の損失（人事評価の相対的な低下）につながる可能性がある点を考慮しないと，業務活動におけるソフトウェア知識提供のポジティブ，ネガティブ両面の要因を説明できない。知識提供活動は組織全体としては利得が多いと考えられるので，それを促進するためには知識提供行動を分析することに意義が認められる。

4.3. 知識提供者の負担

知識を提供することに対する明示的，短期的な利益は少ないとしても，提供者に取って同一企業内で他のチームに対して知識を提供することによる損失（負担）はどのようなものがあるだろうか。

実際にソフトウェア知識を提供するにあたり，もっとも単純な方法はソースプログラムをそのまま相手に渡すことである。ソースプログラムはあらかじめ存在しているので，この場合，コンピュータに対して数個のコマンドを打ち込むだけで知識提供が完了する。つまり，このレベルでの知識提供者の負担はほぼゼロとみなすことができる。しかし，知識受領者がソースコードを理解することを考えると，ソースコードを引き渡しただけ

では知識の移転を完了したとは言い難い。多くの場合，提供者は受領者から説明を要求されることになる。そのため，提供者は説明資料を整備したうえで説明の時間を負担することになる。この説明資料作成と説明の時間は直接的に提供者が負担するコストとして計算できる。[13]

独立行政法人情報処理推進機構 技術本部 ソフトウェアエンジニアリングセンターでは組み込みソフトウェア[14]の品質基準としてソースコードの量に対するドキュメントの量の指標を示している（情報処理推進機構 技術本部, 2012)。ここでは，ソフトウェアの障害が発生した場合の損害額を見積もった上で信頼性への要求を定義し，それに応じたドキュメントレビュー工数および量の参考値を示している(表 4-1)。この値から，知識提供の際にかかる負担を試算できる。まず，設計書は開発前に作るのが原則であり，知識提供のためにはこのすべてのレビュー工数をかけることはないし，すべてを執筆するものではない。また，レビュー工数は参加者全部（一般に 2～4 人程度）を合わせた人×時で示しており，知識提供時は一人で再確認することが想定できるので，この値の 1/4 程度の工数をかけるものと考えられる。また，知識提供時にはこのドキュメントの見直しや整備が必要になり，それが知識提供者のコストとなる。標準的なソフトウェアとして Type2 または 3 を想定すると，1000 行のソースコードを渡すために 10～40 枚程度の設計書を見直し，知識提供者は 2～3 時間程度（開発時の標準レビュー時間の 1/4 程度として計算）の労働コストを知識受領者のために負担することになる。

[13] 表 4-1 で示したように，ソフトウェアの規模に応じたドキュメントレビュー工数および量の参考値から，知識提供の際にかかる負担を試算できる（情報処理推進機構 技術本部, 2012）。

[14] 組み込みソフトウェアとは，家電製品や自動車，携帯電話等の機器に組み込まれたマイクロプロセッサ（組み込みプロセッサ）向けのソフトウェアを指す。各機器は大量に出荷されるので，１品１ユーザのアプリケーションプログラムよりもシステムソフトウェアとの開発における類似性が強い。

表 4-1 設計レビュー作業実施率と設計書ボリューム率

障害発生時の損害額（人的損失が予想される場合は Type3 または 4）	Type1: 1 億円未満	Type2: 10 億円未満	Type3: 10 億円超	Type4: Type3 のうち特に巨額の損失が見込まれるもの
ソースコード 1000 行あたりの設計書レビュー時間（人時）	4.8～9.6 人時	7.2～12.0 人時	9.6～14.4 人時	12.0～16.8 人月
ソースコード 1000 行あたりの設計書枚数	0～19 枚	9～29 枚	19～39 枚	29～49 枚

(情報処理推進機構 技術本部, 2012, p.74,p.87),に基づき筆者作成

ソフトウェアの開発, 使用にはバグの存在とそれに起因するトラブルの発生は無視できない。ソフトウェア提供にともなうトラブルには以下の 3 パターンが考えられる。

(a) 提供したソフトウェアが元々持っていたが，提供者は気づいていなかった。
(b) 提供したソフトウェアが元々持っていたが，提供者側の使用において発生条件が成立しなかった。しかし，利用環境が変化したことによって顕在化した。
(c) 受領者による適用ミスによってトラブルが発生した。

上記のうち，(a)はソフトウェアを提供したことで判明したトラブルではあるが，本来は提供者が発見，修正するものであり，提供したことによって「仕事が増えた」には当たらない。むしろ提供したことによって顧客先でトラブルになる前にバグが発見されたことになり，幸いしたケースといえる。しかし， (b)は提供者の製品としては発現の可能性がなかったものであれば, 提供したためにトラブルに巻き込まれるケースと言える。(c)はもちろん提供者としては自分の製品には全く関係のないトラブルである。(b), (c)に関しては本来は受領者が解決するべきものである。しかし，現実には当該ソフトウェアをよく知っている知識提供者に質問したり, 特に元のソフトウェアにとっては問題にならない (b)のケースであっても提供側の瑕疵と考えて修正を要求したりすることもある。このような場合，提供者にとっては，ソフトウェアを提供したことにより余計な業務が増えることになる。

こうしたことから，知識提供者は提供行動（引渡しと説明）を完了させた後，受領者がその知識（ソースコード）を利用した後にも，負担が発生することを警戒することが考えられる。

4.4. 調査の対象

本章における分析のフレームを図 4-1 に示す。知識移転のフレームワークの中で知識提供者に集中することで，知識提供者に対する受領者の反応･要求および提供者を取り巻く上司，周囲の環境からの影響を把握する。

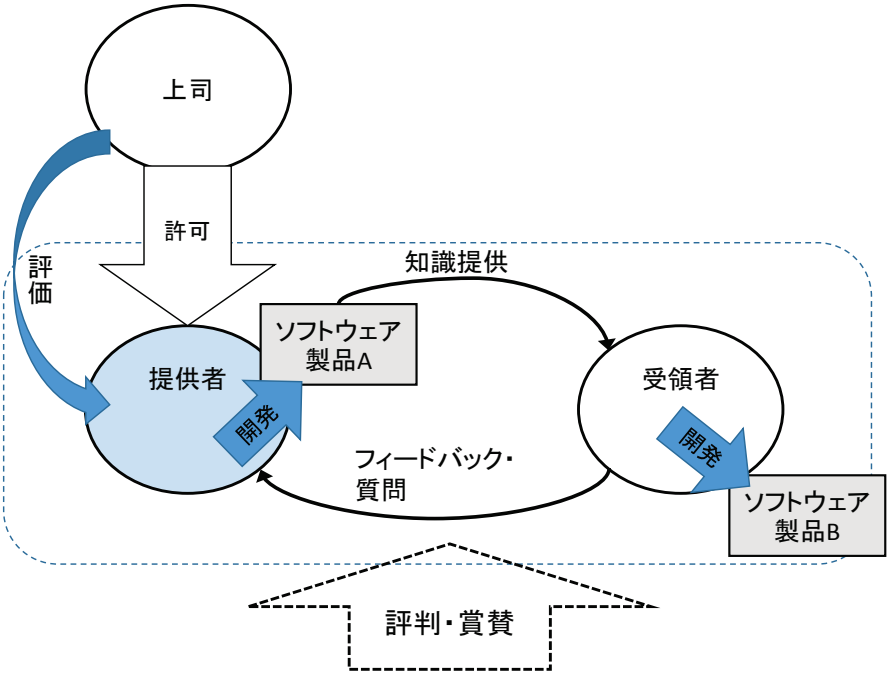


図 4-1 知識提供者に着目したフレームワーク

図 4-1 で，提供者は自らの属するチームで開発したソフトウェア知識をチーム外に提供するに当たって，上司の許可を得る。受領者から提供者へは提供されたソフトウェア知識に対して受領者が施した改善等を「フィードバック」という形で通知されるが，そのほかに「質問」という形で更なる知識提供を求められることもある。提供者の上司

は知識提供によって他チームに貢献したことを成績評価等で高く扱うこともあるが，殆ど評価には関係ないこともある。また，この活動に直接関係のない外部の同僚たちが，知識移転行動を参考にするなどの事象を通して提供者が高い評判を得ることも考えられる。

質問は対象者2054名[15]に対してイントラネットを利用して行い，440件(回答率21%)の回答を得た。なお，記名については任意とした。

4.5. 仮説の設定

調査に先立ち，この部門の中で開発者の分野で上級と認められた技術者に対して，ソフトウェア知識共有に関して関心があるものを募集し，A社の中で「プロコミュニティ」と呼ばれるグループ（以下，「プロコミュニティ」と呼ぶ）を作った。そのプロコミュニティのメンバーで知識共有の事例と問題点について議論し，知識提供のための作業量，負荷を聴取した。その議論の中では，知識提供に対して技術的なメリットが得られることへの期待がある一方で，作業負荷を懸念する意見が多かった。また，組織として期待するのであれば貢献したいという意欲も見られた。そこで，質問票調査を行うために，以下の仮説を設定した。

仮説１．知識提供することによって，知識の更新等のフィードバックが得られるなら提供する。

仮説２．知識提供のために大きな作業が発生しないなら提供する。

仮説３．知識提供を組織として期待するなら提供する。

プロコミュニティの議論では，知識の提供によって提供者が得られる利得としては提供者からのフィードバックが挙がった。これは受領者からソフトウェアのバグの通知や改善方法などの提案であり，提供者への知識移転に相当する。提供者にはバグの対処や受領者への説明などの作業が発生する可能性があるので，それを懸念すると提供にはためらうであろうと予測した。最後の仮説３は，周囲の期待にこたえたいという欲求を満たす行動を技術者も取るであろうという予測による。

[15] 1.5で述べた技術認定により中級以上と認められている従業員を対象とした。

4.6. 質問票の構成 [16]

対象者に対して，まず，他者への知識提供経験の有無を，以下の 2 種類の経験に分類して尋ねた。

a) 無償で他部門に提供する
b) 有償で他部門に提供する

それぞれに対して，提供行動の頻度を 5 段階に分けて回答を求めた。二つの質問に分けたのは，有償の場合と無償の場合で各技術者の動機が大きく異なることが考えられるからである。有償提供の場合は知識提供先の製品からロイヤリティを受け取ったり一括で開発費相当分の負担を求めたりしている。この場合，社内相手であっても知識提供は正式な業務ということになる。そのため，行動は業務命令に基づくものになり，個々の技術者の意思の反映は少ないであろうという推測をした。一方で，無償の場合にはたとえ上司の承認を得ているとしても，その行動には担当者の意思が強く反映しているものと考えられる。

さらに，7 段階リッカート手法を用いて以下の 4 分野にわたって知識提供の動機やためらいの要因に関する 20 件の質問を行った。

a) 提供後のトラブル経験： 4 問
b) 提供にまつわる作業　： 5
c) 社内ルールと行動規範： 5
d) 提供後の期待　：　　6

(a), (b) は仮説 2 の提供のための作業との関連で用意した。(c), (d)は仮説 1 および 3 の提供によって得られるフィードバック，評価や損得に関する質問として用意した。なお，(a) は提供の経験があるものに対してのみ質問している。[17]

[16] 質問票の内容は付録 1 を参照されたい。
[17] (a)のカテゴリで提供後の経験について聞いているのは，提供行動がさらに繰り返されるかどうかの要因になるものと意識したことによる。

4.7. 分析方法

まず，この組織の実態を把握するため，知識提供経験について記述統計データを確認する。表 4-2，表 4-3 に示すように，無償提供は約半数が行っているのに対し，有償提供では 1 割程度しか経験していない。組織としてソフトウェア知識を他部門に提供する行動を取っていないものと理解できる。前述したように有償提供は「業務として」遂行しており個々の技術者の意思に関係なく進んでいるので，個人の知識提供のモチベーションを分析するに当たってはノイズになっている可能性があることに注意が必要である。

獲得したデータに対して因子分析および仮説に基づく変数の導出，パス解析を行って分析した。

表 4-2 無償提供の経験頻度

	度数	%
なし	229	51.9
ごく稀に	100	22.7
ときどき	63	14.3
多くの場合	21	4.8
通常行う	22	5
合計	435	98.6
欠損値	6	1.4
合計	441	100

表 4-3 有償提供の経験頻度

	度数	%
なし	394	89.3
ごく稀に	22	5
ときどき	12	2.7
多くの場合	3	0.7
通常行う	4	0.9
合計	435	98.6
欠損値	6	1.4
合計	441	100

4.8. 分析結果と考察

4.8.1. 有償提供と無償提供の関係

まず，組織としての公的な活動と考えられる有償提供と個人または小グループによる非公式な動機に基づく活動と考えられる無償提供との関係性について確認する。有償での提供活動が無償提供行動に大きく影響を与えているとしたら，有償提供行動の有無によって分析を変える必要があると考えたからである。

有償提供を行っている組織のメンバーは知識提供の手順については業務を通じて習熟していると考えられるので，提供にノウハウを持っている筈である。また，業務を通じて提供していることで，自己，あるいは自チームの知識を他のチームに対して秘匿しなくてはいけないというマインドも持っていないであろうことも想像できる。

一方，有償提供というミッションに基づいて行動したチームは，社内といえども無償提供することに抵抗があることも考えられる。Deci らが挙げた例のように，自発的な行動に対して一度報酬を与えたのちに報酬を停止すると行動も停止する（Deci & Flaste, 1999）ようなことが起こる可能性もある。

表 4-4 有償提供と無償提供のクロス分析

		他チームに「有償で」提供する				
		全くない	ごく稀に	ときどき	多く	通常
他に「無償」で提供する	全くない	220	4	2	1	2
	ごく稀に	90	8	2	0	0
	ときどき	47	7	8	1	0
	多く	17	3	0	1	0
	通常	20	0	0	0	2

質問では，「社内の他チームに無償で提供する」「社内の他チームに有償で提供する」の 2 問に対して，それぞれ「通常行っている」「多くの場合行う」「ときどき行う」「ごく稀に行う」「全く経験がない」の 5 段階での回答を求めた。この調査結果を表 4-4 に示す。

この結果，有償-無償の相関は 1%水準で優位であるものの相関係数は 0.180 と小さかった。一方，外発的動機付け（有償提供）を経験している者が，その動機のない無償

提供をためらうという逆効果も観察されなかった。業務としてソフトウェア技術の有償提供を行うための技術や体制が整うことにより，無償提供を大きく推進したとは言えないが，ハードルを下げたものと理解できる。

4.8.2. 知識提供の動機の分析

知識提供の動機を分析するにあたり，有償提供経験の有無によって大きな変化がないことが 4.8.1 での検討によりわかった。そのため，有償提供経験の有無によりデータを分けることはおこなわず，有償，無償を問わない経験の有無を分析対象とした。

表 4-5　知識提供動機に関する因子

	因子		
質問内容	技術・名誉	コスト・手間	制度・上司の無理解
利用者から感謝される。	.922	−.096	.047
会社の技術力向上に貢献できる。	.855	−.070	.040
他者に助けてもらいたい時に，助けてもらいやすくなる。	.598	−.054	−.049
社内で尊敬を集められる。	.573	−.043	.114
資産の誤りや改善項目について指摘が受けられる。	.534	.218	−.029
社内には自チームの資産を提供すべきである。	.428	.138	−.172
開示できるようにするためには，多くの工数がかかる。	.050	.831	−.052
他チームに資産を開示すると，あとの対応に忙殺されると思う。	−.053	.768	.090
開示するためには，しない場合に比べて，資産を理解しやすい物にしておくことが必要だと思う。	.146	.595	−.048
制度上，資産をチーム外に開示するのは許されないと思う。	−.045	.023	.822
他チームに資産を開示することは，（あなたの意志とは関わりなく）上司に反対されると思う。	.090	−.027	.574

提供の動機の分析のために，まず，20 件の質問項目の回答にたいして因子分析をおこなった。その結果，質問項目から，F1.技術・名誉の獲得，F2.提供のコスト・手間，F3.制度・上司の意向による制約の 3 因子が抽出された（表 4-5）。F1 は知識を提供する

ことによって，自己にとって技術的，精神的な利益を期待することに加え，全社的な貢献をすることを良しとする意識を意味する因子である。F2 は開発技術の観点から，知識提供を行うには受領者との対応のために追加工数が必要になることを懸念する因子である。F3 は自チームで開発したソフトウェア資産の帰属を意識して，ルールや上司の意向として他チームに提供することは認められないだろうという考えを反映する因子である。

また，各因子間の相関関係は，F1:技術・名誉の獲得と F2:提供のコストの間および F1:技術・名誉の獲得と F3:制度・上司の意向による制約の間にそれぞれ相関があるが，F1 と F3 の間の相関は小さい (表 4-6)。

表 4-6　知識提供動機因子の相関関係

因子	F1 技術・名誉	F2 コスト・手間	F3 制度・上司の 無理解
F1	1.000	.218	-.161
F2	.218	1.000	.200
F3	-.161	.200	1.000

因子 F1 は提供することで，技術的なフィードバックを得られたり，名誉が得られたりすることを期待する因子である。知識提供を啓蒙するためにその効果を技術的に分析して訴えることによって高めることができると考えられる。因子 F2 は制度を整備して知識提供者に負荷がかからないように，例えば知識提供者に品質責任を問わないことによってコストを下げることができるであろう。因子 F3 は組織としての意向を明確にすることで変化させることができる。

この結果に基づいて抽出された因子に対してパス解析を行った結果を図 4-2 に示す。ここで，F1, F2 から知識提供へのパス計数は極めて小さく，知識提供行動にはほとんど影響がないことがわかる。一方，制度・上司の無理解については負のパス係数を示している。

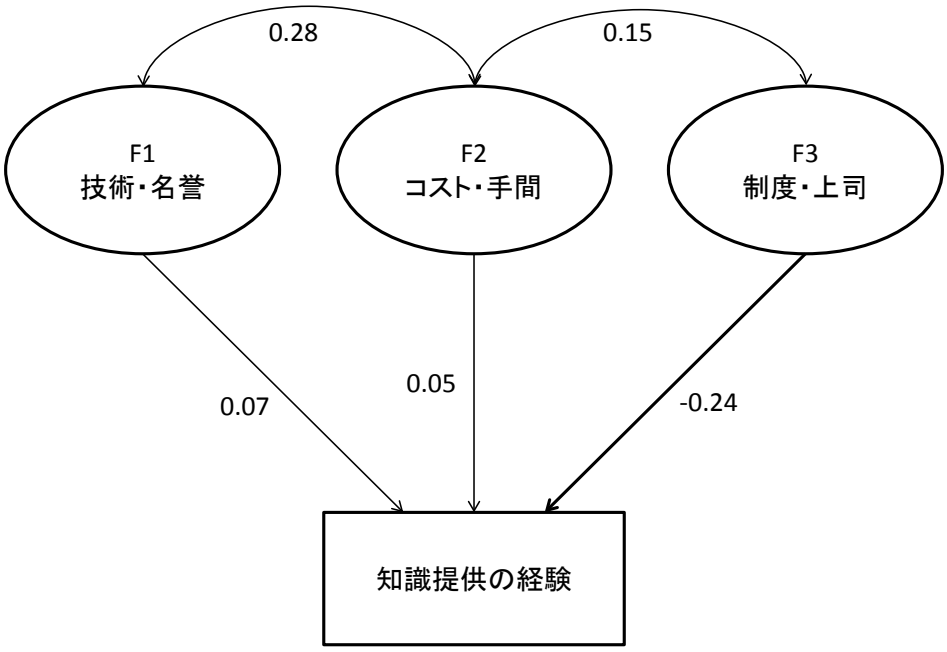


図 4-2 知識提供因子と知識提供行動

F1 の知識提供行動への関与が小さいことから，仮説 1 で挙げたフィードバックへの期待は小さいことがわかり，仮説 1 は棄却された。これは，知識を提供することによって有形・無形を問わず何らかの利益を得ることには技術者たちは関心度が小さく外発的な動機付けの効果は小さいことを示している。

また，F2 の関与も小さいことで，提供の手数に関する懸念は殆どないこともわかった。仮説 2 ではこのパス係数は負の値になると予想していたが，これも棄却された。

提供物の整理に工数がかかったり，質問に忙殺されたりするコストへの懸念が知識提供を阻害するわけではないという結果に対して以下の二つの解釈が考えられる。第一の解釈は他のチームの役に立つことを理解しコストを喜んで負担するということである。

第二の解釈は自らの開発作業において既に作成済のドキュメントが他チームに（ソースコードとともに）渡すのに十分と自負しているということである。

この二つの解釈に対して，F1 の関与が小さく仮説 1 が棄却されて，外発的な動機付けの効果が小さいことと合わせて考えると，F2 の関与が小さいことについては，会社のために喜んでコストを負担するという考え方に基づいて知識提供行動が進むという内発的な動機によるものとすることに整合性がある。しかし，あらかじめドキュメントが整備されているので提供コストが低いという考えを棄却する材料にはならない。

二つの仮説が棄却されたことから，技術的な制度を整備して知識提供の価値を啓蒙したり作業コストを下げたりすることの効果は小さいことが推測できる。

一方，F3 からのパス係数が負になっていることから，他のチームに対する知識提供は好ましくないと思われるという理解に基づいて提供をためらっているものが多いということが理解できる。つまり，仮説 3 は支持されたと言える。第 3 章で触れた各部門のビジネス規範意識を高くしたことが，知識提供を好ましいものと思わない一つの要因になったものと考えられる。

表 4-7 に質問票から知識提供に関する制度や上司の意向を推測した結果を示す。ここから，「制度上，資産をチーム外に開示することは許されないと思う」の平均値が 3.01（7 段階リッカートスケール，7 が大いにあてはまる，1 が全くあてはまらない），「他チームに資産を開示することは，（あなたの意思とは関わりなく）上司に反対されると思う」の平均値は 3.18 であった。ともに，「どちらかと言えばあてはまらない」に該当する解答であり，提供を否定する解答は得られなかった。しかし，「どちらとも言えない」が多数を占め，特に，半数以上の技術者は上司が賛成することに対しての確信までは持てていない。

つまり，各部門，チームのビジネス意識を高めた結果，自部門の持つ知的資産（ソフトウェア知識）を自部門の外に出すことは自部門に対する背信的な行為ととらえられ，たとえ社内といえども簡単には許されないという意識が徐々に浸透していると考えられる。ただし，表 4-5 の質問内容に示すように，回答者は上司の意向を先回りして慮って行動しているのであって，実際に組織として提供を禁止しているとは言いきれないところに問題がある。

表 4-7 制度，上司の意向の推測

2-4：他チームに自チームの資産を提供することについて，思うところを伺います。		
	d：制度上，資産をチーム外に開示することは許されないと思う。	e：他チームに資産を開示することは（あなたの意思とは関わりなく）上司に反対されると思う。
全く当てはまらない	91	82
当てはまらないことが多い	67	49
どちらかと言えば当てはまらない	84	68
どちらとも言えない	147	198
どちらかと言えば当てはまる	16	19
当てはまることが多い	13	5
大いに当てはまる	9	7
有効回答数	427	428
平均値	3.01	3.18

4.9. まとめ

この章では，自部門で開発した知識を部門外に無償提供する行動を推進または抑止する要因を調査し，知識提供行動を推進するための方策を検討した。調査結果から（仮説3が支持されたことで），行動が抑止される原因は制度や上司の意向に関する懸念にあることが明らかになった。

一方，仮説1で予想していた，知識提供に関する外発的な動機による影響は観測できなかった。また，仮説2の提供に伴うコストや責任を恐れて行動を抑止するという事象も観測されなかった。この結果より，A社においては外部からの動機付けやコスト（手間）の軽減は知識提供行動に影響を与えることはないということになる。つまり，提供すべき知識を持っている者に対して外発的な動機付けを行っても，知識提供は促進されない。

それよりも，仮説 3 が支持されたことから，知識提供を推進するには，知識提供に対する組織としての方針を明確にすることの効果が最も大きいことが予想される。つまり，ビジネス意識が強調された環境において，自らのチームの知識を他のチームに提供することが望まれないことではないかという懸念を持ったままの判断のため，知識提供をためらっているという予想である。

ただし，この調査では提供経験者が多くはなかったこともあり，内発的な動機の有無や業務負担の大きさが顕在化しなかった可能性は考えておく必要があるだろう。たとえば，技術力が高くて理想的な開発プロセスを推進している者だけが知識提供をしているとすると，ソースコードに対するドキュメントの準備はすでに整っており，他部門に提供するからといって追加コストをかけて準備を整える必要は少ないので，提供経験者は提供のためのコストを気にしない。しかし，提供行動が当たり前の行動になるように推進していくと，提供のために追加コストを払わなくてはならない者，つまりドキュメントが不十分だったり，質が低い知識を提供してフォローのためのコストが大きくなってしまったりするような技術者も提供者になり，F2（コスト，手間）の寄与が大きく変化していくことも考えられる。

また，この調査では技術者たちは上司の意向を「慮って」回答しており，実際に提供が好ましくない行動とされているのか否かは明確にされていない。所属するチームの損益意識を植え付けられるようになった回答者は以下のように考えるであろう。

- ソフトウェア資産は自部門のコストを払って開発したものである。
- 自らのコストで開発したソフトウェア資産はチームの収益に役立てなくてはいけない
- チーム外に提供すると，自チームにとっては機会損失になる。
- 組織としてはチーム外に提供することは許されないだろう。

この結果として，多くの者が他チームへの知識提供を好ましくないものと理解し，知識提供の事例が少なくなっていたものと推測できる。第 3 章で述べたように各チームの収益意識を高めるように変容してきた A 社はあいまい性を排除してきた。その一方で，知識移転に関する見解はあいまい性を残したままであり，明確になっていない上司の意図を慮る行動が見られていることになる。つまり，A 社のソフトウェア開発組織のように，収益意識を高めるような変化を進めてきた環境においては，それにあわせて上司は知識提供を推奨するような言動を一段と明確にしないと，技術者たちは上司の顔色をう

かがって「知識の提供は悪」と考えてしまう危険性が高い。知識提供を推進するためには，知識提供に対する見解を組織として明確にすることが有効であると考える。

第5章 知識受領行動の分析

5.1. はじめに

本章では，図 1-1 (c) で「知識受領者の行動分析」としたソフトウェアに関する知識を受領する行動が，どのような状況によって活性化し，あるいは受領をためらうことになるのか, 前章に引き続き A 社ソフトウェア開発組織のデータを用いて分析を行う[18]。

さらに，受領による成功体験者に対してインタビューを行い，実体験を聞き取ることで知識移転の要件を分析する。

5.2. 知識受領行動の分析の目的と意義

ソフトウェア製品は技術者が記述したソースプログラムをコンパイラによって機械的に翻訳し，コンピュータ上で実行可能なプログラムにすることで完成する。ソースプログラムは人が読むことが可能なプログラミング言語によって記述されている。そのため，ソースプログラムには完成したソフトウェア製品の論理のすべてが過不足なく，かつ技術者が読み取ることができる形で記載されている。すなわち，ソースプログラムは形式知として完全なものであり，ソースプログラムを授受することでソフトウェアに含まれる知識を完全に移転することができる筈である。

すべての知識を容易に完全に受け取ることができるのであれば，提供側の組織や個人が努力して作り上げた知識を含む知的資産を受領側は労せずして手に入れることができる筈である。それにも関わらず，実際には他者のソースプログラムを受け取る事例が少ない (Tracz, 2001)。この理由を解明し，受領者の視点から知識移転行動を活性化する方策を発見することが，本章での分析の目的である。

これまで，情報工学，ソフトウェア工学の分野においてソフトウェア資産の再利用性について研究が進められてきた。ここではソースプログラムが「モジュール化」されていることや知識共有のインフラ整備という技術的な面での成果が出ている。しかし，実

[18] 本章の内容は筆者らによる研究報告（堀田, 杉原 & 遠山, 2012）を基本として加筆修正を加えたものである。

際に技術が整備されても知識共有が行われないという現実がある。特に上述のようにソフトウェア開発においては知識を移転することが容易と考えられるにも関わらず，実際には知識共有がなされないという状況を鑑みると，技術面だけでなく，知識受領者の意識や組織運営も含めて分析することに意義があると考えられる。

一方，知識科学の分野においては，知識移転の難易度に関して，知識の粘着性(Stickyness) (Szulanski, 1996)や知識吸収能力(Absorptive capacity) (Cohen & Levinthal, 1990)に着目した研究がある。ここでは，移転対象となる知識の性質と受領者との関係について論じている。粘着性の高い知識は知識を保持している個人や組織環境に特有な性質を持っており，その環境を離れて知識のみで使うことが困難である。また，多様化した知識を保持している受領者の知識吸収能力が高くなる。しかし，両者とも暗黙知と形式知を合わせて移転する場合の要件について述べているものであり，ソフトウェアにおけるソースプログラムのような完全なる形式知の移転が困難であることには注目していない。

本章では，ソフトウェア知識の性質を踏まえたうえで，知識科学の研究に則った分析を進め，知識利用行動を促す要因を提示する。

5.3. 再利用におけるソフトウェアの特性

前述のように，ソフトウェアの知識はソースプログラムに完全に記述されていて，かつエンジニア自身が読むことができる。しかし，ソースプログラムに記述されているのはプログラムの論理のみであり，ソフトウェアの実行に直接かかわらない論理の背景の説明などは含まれていない。利用する側の技術者が他者が開発したソフトウェアを利用する場合には，利用の形態に合わせて内容を理解することが必要である。たとえば，他者のソフトウェアをそのまま部品として組み込むのであれば外部仕様のみ理解して利用することも可能であるが，機能の変更が必要になる場合には内容の理解が必須である。また，ソフトウェアに付き物であるバグを自分で修正する必要に迫られた場合には，内容の理解のみならず，論理適用にいたった背景や，なぜそのようなバグが入り込んだのかを理解することが要求される。

ソフトウェアを再利用するにあたって，再利用されるソフトウェアに対して以下の 3 項目の要求項目が挙げられる（米澤, 1984)。

(1) 信頼性

(2) 可読性および保守性

(3) 周辺独立性と再利用性

開発現場では(1)の信頼性は，通常ソフトウェアのテストと設計ドキュメントやソースプログラムのレビューによって向上させている。ソフトウェアは極めて論理的な製品であるため，機械製品におけるいわゆる「あそび」のような余地は殆どなく，プログラム論理のバグは直ちに製品の不良につながる。しかし，発生頻度の少ない条件についての考慮漏れは，たとえ致命的であっても気づかれず潜在的なバグとして製品にそのまま残ることがある。この種のバグは特定の条件下で（開発・検証時に漏らした考慮点に当たったとき）発現すると，利用者に迷惑をかけることになる。そのため，たとえ他者のソフトウェアを再利用した場合であっても，責任をソフトウェア知識の提供者に負わせるわけにはいかない。知識の再利用者は潜在バグがないようにするとともに，バグが顕在化した場合には修正を提供することが求められる。そのため利用する知識を理解することは顧客に対する責任として求められる。

(2)は受領者の知識吸収能力（Cohen & Levinthal, 1990)を補う要件である。受領者は受け取ったソフトウェア知識を理解した上で自らのソフトウェアの中で活用するが，これまでに述べてきたように，それは簡単な作業ではない。受領者に無限の知識吸収能力があればよいが，一般の技術者にとって可読性の高いソースプログラムは救いになる。レビューしたり修正したりするにあたり可読性が高いことは受領者の作業を軽減し，作業を確実なものにするからである。

(3)は再利用しようとする部品がモジュール化されていて他の環境との関係が少ない，つまり知識の粘着性（Szulanski, 1996)が低いことを要求している。ソフトウェアを再利用する際に，外部環境とのインタフェースが多いとその設定に苦労するし，インタフェースが明確でないと正しく設定することができなくなる。そして(2)および(3)は(1)の信頼性を担保するための要件となっている。

ハードウェア部品の場合，たとえば標準規格のネジを利用する際には，ネジのピッチ，太さ，材質や強度の情報を得てどのネジを利用するかを決めることができる。これはほぼ外部仕様だけで選択できる例と言える。一方，ソフトウェアの場合は上記のような要件から，外部仕様として記述されているものが完全でないことが多いため部品の性質を正しく予測することが難しい。そのため，たとえドキュメントがあっても内部まで確認することが要求されるのである。

可読性・読解性の向上のために，ソフトウェア工学ではソースプログラムをモジュール化などの技術を用いてより理解しやすく開発することを教え，技術者はそれを実践すべく努力している。しかし，いくら理解しやすいように記述しても，ソースプログラムは普通の文書に比べると理解が困難である。ここで，理解を助ける資料が存在する。つまり，業務としてのソフトウェア製品の開発にあたって，いきなりソースプログラムを書くことは一般的ではなく，ソースプログラムの執筆前に中間生産物である設計書を記述することが普通である。設計書は自然言語や図を用いて記載された書類であり，プログラムの論理以外に設計の背景やアルゴリズムの解説，留意点なども書かれている。そのためソースプログラムよりも読解性は高く，理解を助けるためには有効な資料である。市販のソフトウェアを部品として利用する場合にはソースプログラムや設計書を期待することは難しいが，同一社内で開発したソフトウェア知識を移転する場合には，この設計書も含めて受け渡すことも可能である。

しかし，1.3.1(c)で述べたように設計書にはもともと全てが書かれているわけではない（Jones, 1999, pp.74-76)。また自然言語で記述されているため（理解はしやすくても）厳密性に欠けたり，執筆者によって自明と思われたことが書かれていなかったりと，知識としての完全性はソースプログラムに劣る。また，設計書は最終生産物ではないため，過不足や誤りが含まれていることがわかっていても修正がなされていないこともあるという実務上の問題も存在している。

周辺独立性の点では，構造化プログラミング（Dijkstra, 1970），オブジェクト指向プログラミング（米澤, 1984）などソフトウェア工学で論じている。しかし，再利用が考慮されずに周辺独立性が低いプログラムモジュールは実際に存在する。そのような場合，うかつに再利用すると想定外の動作をして，受領側が開発するソフトウェアにバグを作りこむ危険性がある。周辺独立性が低い場合は読解が困難であり，その結果として依存性が高いこと自体を見抜きにくいことがある。そのため周辺独立性が低いソフトウェア部品は再利用には向かない。

社内において他者が開発したプログラム資産であっても，その活用の基本的な責任は使用者側にあるという考えは浸透している。そのため，活用する前に知的資産を理解した上でプログラム資産を導入し製品に組み込むことが第一条件になる。上記の３要件はこのための要件である。

また，受領者は他者知識資産の持つバグに遭遇し，その修正に忙殺されることを恐れている。そのため，プログラム知的資産の信頼性は重要な要因である。信頼性，あるい

は信頼「感」は開発者による担保では不十分で，第三者による保証や他の利用者の評判が有効である。

5.4. 調査の対象

知識利用行動に関する分析のフレームを図 5-1 に示す。知識移転のフレームワークの中で知識受領者に集中することで，知識受領時の作業や提供者への期待および受領者を取り巻く上司，周囲の環境からの影響を把握する。

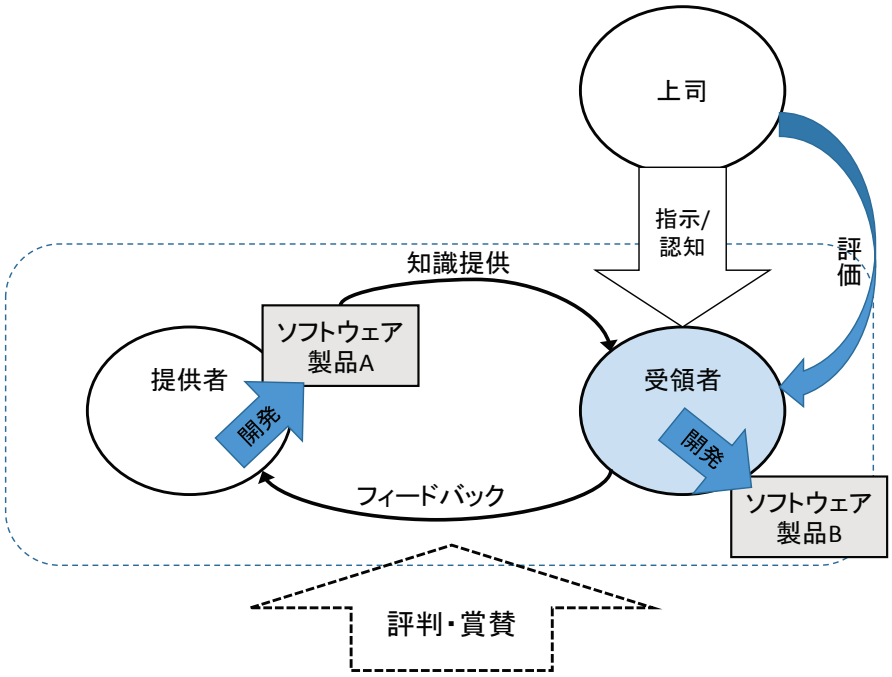


図 5-1 知識利用行動分析のフレームワーク

部門内の上位の技術者を対象としたプロコミュニティでのソフトウェア知識共有についての議論，及びグループインタビューを行ったことは知識提供行動に関する調査と同様である。

質問も知識提供行動に関する調査と同じく，開発業務に関わっている 2054 名の技術者に対し，イントラネットを使用して各自の机上の PC を経由して行い，回答数 440

件(回答率 21%：無記名)を得た。調査対象者は上級(回答数 94 名)，中級(246 名)，初級(101 名)であった。このうち，中・上級者は開発プロセスの設定や後輩の指導を行うなどチームのマネージャやリーダーとして活動しているのに対し，初級者は先輩，上司の方針に従って開発業務に関わっている。

対象とした組織は複数のシステムソフトウェア製品(ミドルウェア)を開発しており，各技術者はそれぞれの製品開発毎のチームに所属している。チーム毎の製品機能の類似性は少ない。そのため，知識移転がなされる場合でも，それぞれの製品が使用する「部品」レベルの移転となり，移転された部品の組み合わせは各技術者が工夫することが要求される。アプリケーションプログラムの場合には，顧客に合わせたカスタマイズ作業だけで出荷するようにソフトウェアの大部分が同一というようなことがあるが，システムソフトウェアの開発部門では複数の顧客に対して同じパッケージを提供するので，開発部門がカスタマイズ作業を担当することは少ない。

5.5. 仮説の設定

組織全体に対する質問表調査の前に，上記のプロコミュニティを通じてソフトウェア開発者から，活用しやすいソフトウェアの性質を聴取した。さらに質問票調査を行うために，プロコミュニティでの議論に基づき，以下の仮説を設定した。

仮説１．他者資産を利用していることを評価する環境では，活用されやすくなる。
仮説２．開発効率が上がると思えば，他者資産を活用する。
仮説３．他者の仕事内容を知っていれば，他者資産を活用する。

仮説１はソフトウェア開発量の指標として新規開発量を重視しない環境においては他者資産の活用が促進されるであろうというものである。この指標を使うと技術者あるいはグループが新しく開発したソースプログラムの行数で実績を測定されるが，他者資産を受領・活用したときには測定の対象外になる。そのため新規開発量の指標が重視される環境においては他者資産を利用すると「成果が小さい」と評価されかねない。逆に，新規開発量の指標の重みが少ない環境であれば技術者は安心して他者資産を利用するものと考えられることからこの仮説を設定した。

仮説２は技術者が他者資産を活用することに価値を認めているかどうかを意識したものである。

仮説３は他者の仕事内容にも関心を持っている技術者は他者資産を活用するであろうという仮説である。関心を持っていれば，どこに自分の必要とするソフトウェア知識が存在するかわかりやすいため，他者資産の活用が促進されるという推論に基づくものである。

5.6. 質問票の構成 [19]

対象者に対して，まず，他者知識受領の経験の有無を，以下の３種類の経験に分類して尋ねた。

a) 他者資産を部品としてそのまま取り込む
b) 他者資産を部品として改造しながら取り込む
c) 他者資産を参考事例として参照する

他者が開発したソフトウェア資産の活用方法として，(a) は一切直さずにそのまま取り込むケースである。これはもっとも簡単にできる方法ではあるが，提供者の用途と受領者の用途が完全に一致していない場合には適用できないので，あまり事例が多いとは考えられない。これに対して (b) では受領したソースプログラムを受領者の製品に合わせるように修正して適用するので，受領したプログラムを理解することが必要になる。技術的には難しくなるが適用範囲が広がり，事例としても多くなることが予想できる。(c) は (b) と似ているが，受領した知識は「見るだけ」で自己の製品には直接は組み込まないケースである。それぞれの参照方式によって受領者の行動が変わる可能性が強いので，あらかじめこの分類を行った。

さらに，７段階リッカート手法を用いて以下の５分野にわたる21件の質問を行った。

a) 参照時の周囲の反応： 5 問
b) 組織としての行動規範： 4
c) 参照することへの周囲の関心： 3
d) 参照のコストと技術力： 7
e) 社内状況に関する知識： 2

[19] 質問票の内容については付録１を参照されたい。

(a), (b), (c) の質問は仮説 1 に関する質問として用意した。(d) 及び (e) はそれぞれ仮説 2，3 に関する質問として用意した。

5.7. 分析結果と考察

調査結果より表 5-1 に示したように，総じて，他者知識の吸収による効果を期待するものの，吸収・活用は困難であり高い技術力が必要という認識の傾向が強く，リスク回避型の意見が強いと言える。

表 5-1 活用への期待とコスト予測

	質問番号	質問内容（7 段階リッカートスケール）	平均値	標準偏差
効果への期待	Q15a	他者資産に触れることは，自分の開発者としてのスキル向上につながると思う。	5.26	1.20
	Q15f	他者資産を活用すると，自らの開発作業が楽になると思う。	4.60	1.24
活用のコスト予測	Q15b	他者資産を活用するには，それを十分に理解することが必要だと思う。	6.15	1.12
	Q15c	他者資産を理解することは難しい。	5.39	1.24

提供者から受領者にソースプログラムが移転されると，受領者はそのソースプログラムを自チームの開発に活用する。ソースプログラムを活用するためには，受領したソースプログラムから該当部分をモジュールとして切り出し，それを自チームのソースプログラム内に取り込む作業を行う。外部仕様が明確に定義されているモジュールは切り出しが容易なのでソースプログラムは知識共有されやすくなる（表 5-2）。たとえば，数

学関数[20]などは入力値と結果の関係が数学的に一意に定義されているため，その内部論理が難解であってもそれを理解する必要はなく，モジュールとしての活用は容易である。

表 5-2 使いやすいソフトウェア知識

	外部仕様が明確に定義されている
1. 全く当てはまらない	6
2. 当てはまらないことが多い	6
3. どちらかと言えば当てはまらない	3
4. どちらとも言えない	15
5. どちらかと言えば当てはまる	75
6. 当てはまることが多い	129
7. 大いに当てはまる	196
有効回答数	430
平均値	6.06

ソースプログラム受領の成功者は，活用に苦労したと言いながらも開発期間の短縮には役に立ったと述べている。一方，同一人物の声として，開発期間の制約がなければ自力開発もありえるとも述べている。プロジェクトの目標を重視する意思が知識移転へのためらいを克服し促進する要因となっているものと考えられる。

回答 441 件のうち他者知識の参照経験があるという回答は 298 件あった(表 5-3)。分析は，この 298 件のうち，独自の判断で開発方針を決定できる技術者である上級，中級の参照経験者である 239 件に対して行う。

[20] 三角関数や対数，指数などのように数学的に定義されていて入力パラメタに対して結果が一意に決まる関数

表 5-3 他者資産参照経験者数

	初級	中級	上級	計
総数	101	246	94	441
参照経験あり	59 (58%)	169 (69%)	70 (74%)	298 (68%)

5.7.1. 因子の抽出

21 件の質問に対する解答から，他者資産参照行動に対して，表 5-4 に示すように F1.他者の目，F2.周囲への関心，F3.開発技術の意識の 3 因子を抽出した。

F1 は，他者資産を参照して（効率よく）開発すると上司や同僚からどう思われるか。賞賛されるのか，あるいは自力で開発できない技術者として低く見られるのかを意識を意味する因子である。F2 は他者の仕事について関心を持ち，知っているかどうかを示す因子である。F3 は自己の開発技術に取って，他者資産を活用することがプラスに働くかどうかの関心を持つことを示す因子である。

表 5-4 抽出された因子

		因子		
		F1 他者の目	F2 周囲への関心	F3 開発技術の意識
Q12c	他者資産を活用したことで， 同僚から敬意を示されている。	.872	-.093	-.129
Q12a	他者資産を活用したことで，成績評価で 上司から高く評価されたことがある。	.784	.083	-.039
Q12b	他者資産を活用したことで，成績評価で 不利に扱われたことがある。	.631	-.034	.094
Q14a	上司から高い評価を得られると思う。	.469	.030	.026
Q14b	上司からの評価は下がると思う。	.298	-.171	.245
Q13b	ソフトウェアの開発量は新規開発量を 重視して測定される。	.162	-.028	.111
Q16b	社内で他社資産を活用して**失敗した**事例	-.142	.826	.179
Q16a	社内で他社資産を活用して**成功した**事例	-.036	.742	-.182
Q13a	上司が薦めている。	.263	.357	-.040
Q12d	他者資産を活用して，トラブルにあった 経験がある。	-.213	-.347	-.235
Q14c	上司は他者資産の利用の有無について 関心がないと思う。	-.016	-.341	.059
Q13d	チーム内の同僚が，他者資産を活用してい る。	.230	.293	-.100
Q13c	他者資産を活用することは簡単なことだと 上司が考えているように思う。	.092	.213	.169
Q15b	他者資産を活用するには，それを十分に 理解することが必要だと思う。	-.080	.174	-.056
Q15f	他者資産を活用すると，自らの開発作業が 楽になると思う。	-.101	.051	-.714
Q15g	他者資産を活用すると， 却って多くの工数がかかると思う。	.144	.062	.666
Q12e	他者資産を活用して， 開発が楽になった経験がある。	-.206	-.115	.471
Q15a	他者資産に触れることは，自分の開発者と してのスキル向上につながると思う。	.002	.079	-.461
Q15c	他者資産を理解することは難しい。	-.092	.151	.252
Q15e	他者資産に頼らず，自ら開発することで 技術者としてのスキルが向上すると思う。	.117	.043	.174
Q15d	自分は他者資産を理解する読解力が 高いと思う。	.081	.074	-.166

この３因子のうち，F1 と F2 の間には相関があることも明らかになった(表 5-5)。これは他者の目を意識するものは同時に周囲への関心を持っているということで，納得できる結果である。

表 5-5 抽出された３因子の相関

因子	F1 他者の目	F2 周囲への関心	F3 開発技術の意識
F1	1.000	0.320	-0.082
F2	0.320	1.000	-0.012
F3	-0.082	-0.012	1.000

5.7.2. 因子と行動の関係

次に，抽出された 3 因子と他者資産受領行動との関係を解析した。受領の形式として，完全流用，改造，参考資料としての使用の 3 種があるので，それぞれについてパス解析を行った結果，図 5-2，図 5-3 および図 5-4 で示すように，以下の現象が認められた。

- F1（他者の目）と参照経験の間には，どの参照方式，開発技術レベルにおいても有意な相関は認められない。
- 上級者は F3（開発技術の意識）が高いほど「改造レベルの参照」行動が活発になる。しかし，中級者においては，この相関が低い。
- 上級者は F3 と「完全流用」とは小さな負の相関になっている。
- F1 と F2（周囲への関心）の相関は，中級者よりも上級者において大きい。

(1) 他者資産利用と行動に対する評価

仮説１．では因子 F1 と他者資産受領との正の相関関係を期待していた。つまり，組織の制度を用いて他者知識受領を推奨することにより，知識の受領行動が推進されるという仮説である。しかし，本調査結果では相関関係は認められなかった。

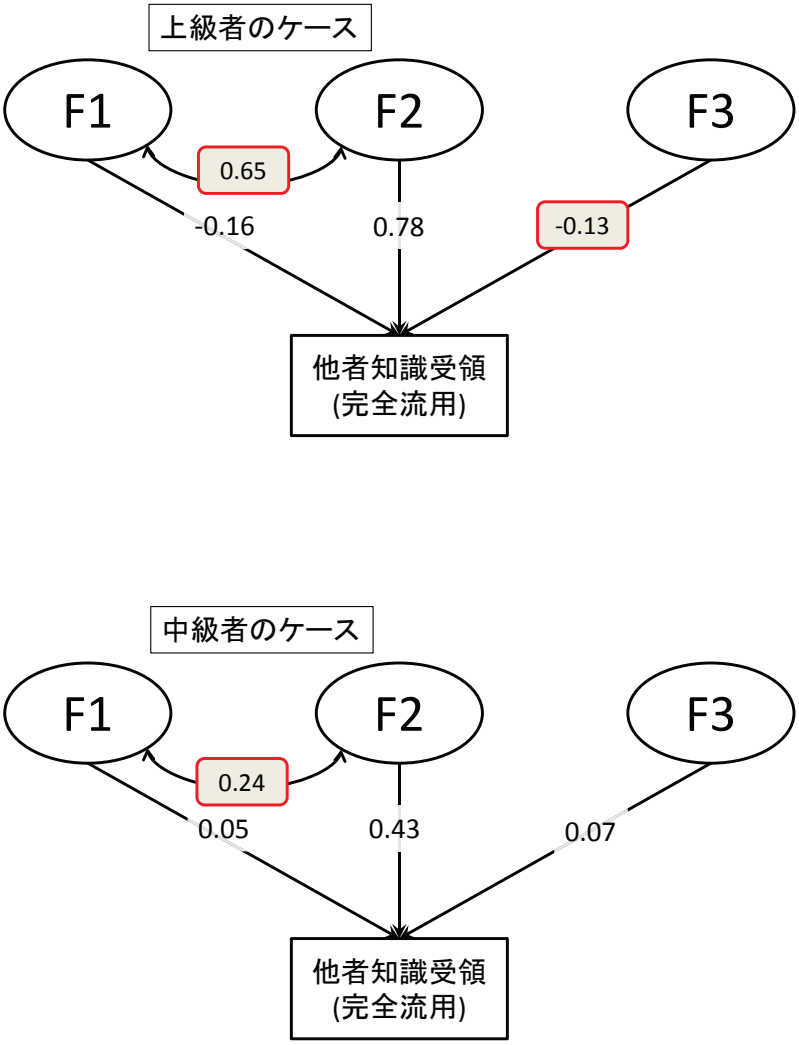


図 5-2 3 つの因子と受領行動(1)-完全流用の場合

この組織の技術者は報酬や他人の賞賛のような外発的な動機では受領行動を進めることがなく，ここでは仮説は棄却された。

受領者は上司や周囲の意図に反しないように留意しつつ知識を受領し，自チームのために活用している。受領に対する上司の意図や成績評価への予測を訪ねた結果を表 5-6 に示す。上司の薦めに対しての平均値は 3.85， 成績評価に関しても 3.75 と中央よりは受領に対して積極的でない上司の意向を予想している。また，ともに「どちらとも言えない」が大半を占めていることから，部下は上司が外部知識を導入するか自己開発するかには関心がないと思っていること示しているものと言える。

表 5-6 受領行動への反応予測

	1-3:社内の「他者資産」を部品または参考資料として活用することに関して，あなたの組織はどのように行動しているでしょうか。 a：上司が薦めている。	1-4：今後，他者資産を活用するとしたら，どのようなことが予想されますか？ a：上司から高い評価を得られると思う。
1. 全く当てはまらない	84	82
2. 当てはまらないことが多い	23	26
3. どちらかと言えば当てはまらない	29	22
4. どちらとも言えない	162	205
5. どちらかと言えば当てはまる	69	64
6. 当てはまることが多い	44	27
7. 大いに当てはまる	20	10
有効回答数	431	436
平均値	3.85	3.75

(2) 開発効率向上の期待と他者資産利用行動

仮説２．は因子 F3 と他者資産受領との相関関係を期待していた。調査の結果によると，開発効率を期待する上級技術者においては強い相関があるが，中級技術者においては有意な相関関係は認められないという結果になった。また，上級技術者では「完全流用」とは小さいながらも負の相関を示している(図 5-2)。

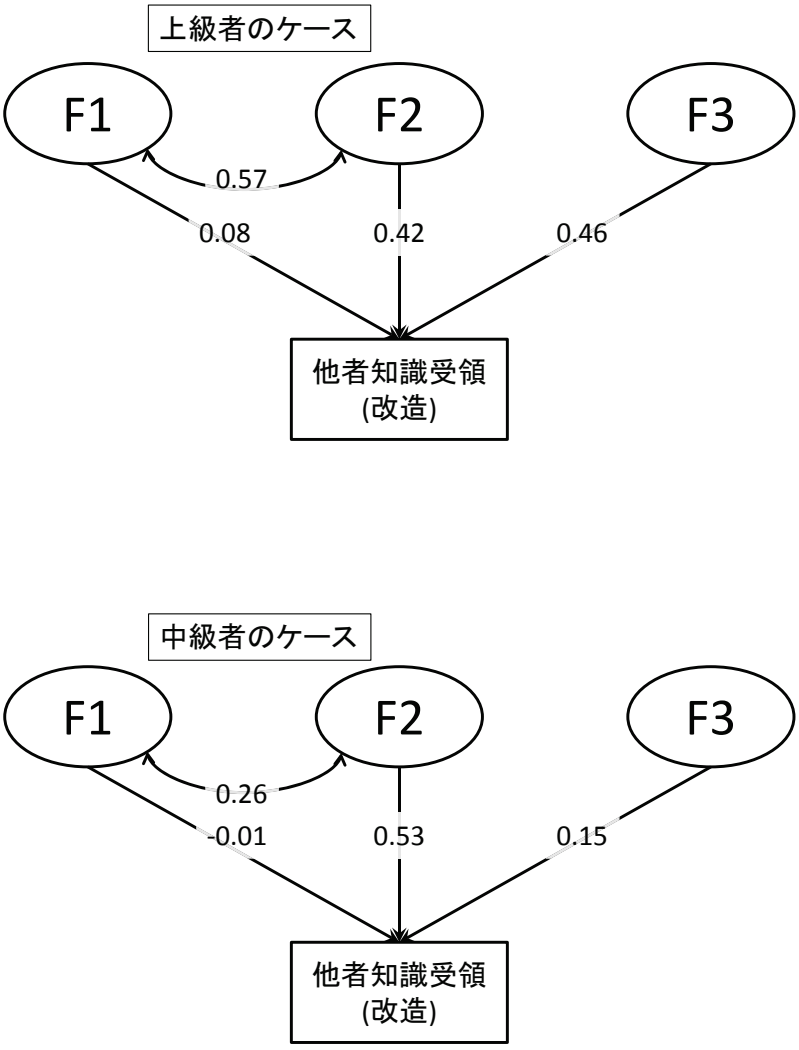


図 5-3 3 つの因子と受領行動(2)-改造の場合

上級技術者は吸収能力が高く，開発効率を追求する際には他者資産を活用に踏み切るだけの力がある。しかし，中級技術者はまだ吸収能力に自信が持てず，他者資産を参照し理解するためには却って時間がかかり，効率低下に陥ることを懸念するものと考えられる。

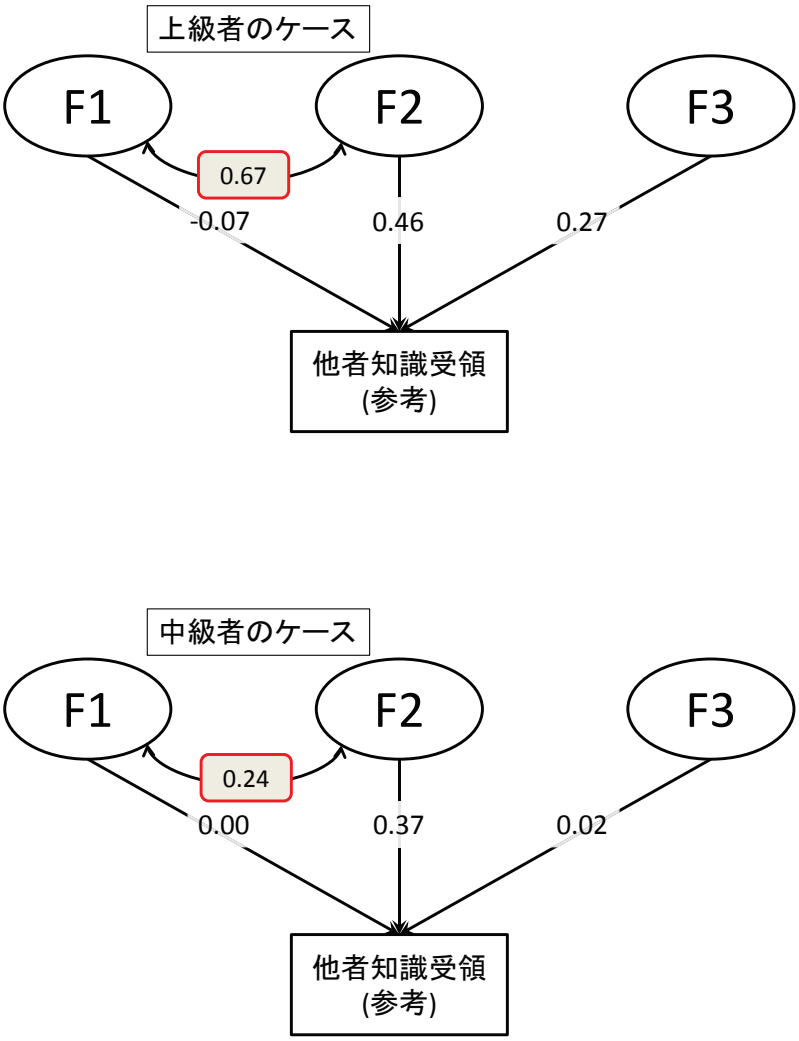


図 5-4 3 つの因子と受領行動(3)-参考資料として

一方，経験豊富な上級技術者は，技術受領の困難さも熟知し，単純に「完全流用」をすることの危険性もその経験から理解している。そのため，一見簡単に思える完全流用には飛びつかないものと考えられる。

仮説２. に対する考察から，技術レベルによる行動の違いが明らかになった。つまり，上級者に対しては仮説は支持されるが，中級者に対しては支持されない。この結果から，技術レベルに応じた知識受領への動機付けが有効であることが示唆されたものと考えられる。

(3) 他者の活動に対する情報と知的資産利用行動

因子 F2 はソフトウェア開発部門内の情報を持っているか，つまり開発チーム内にとどまらずチームの外にも情報網を持っているかどうかを示すものである。仮説 3 は因子 F2 と知的資産受領との関係に関するものであり，これはどの受領状況においても有意な相関関係を示しており，仮説は支持された。

ただし，因子 F2 と知識受領についての因果関係までは確認できていない。知識受領を繰り返しているうちに部門内の知識の所在を含む情報通になったとも考えられる。一方，部門内の情報をよく知っていれば，どこから知識を入手すればよいかが正確にわかるので受領が促進されるのも容易に予測できる。受領行動が受領者の情報網を発達させ，さらに受領が進むと考えることできそうである。

さらに，チームの観点からすればチーム外に広がった情報網をチームのメンバーが活用することで，他のチームの持っている知識が自らのチームに導入されることが期待できる。その結果として，個人のレベルを超えてチーム同士の知識流通網への発展の可能性がある。つまり高次の場(higher BA) (Hotta, 2009) への成長が期待できる。活発な高次の場はチーム間の知識流通の基幹となることを傍証するデータであると言える。

5.8. 知識提供及び利用行動の総合的検討

これまでに，A 社の組織マネジメント方式と知識移転の実態について調査，分析を進めてきた。本節では分析結果の理解を深めるために，インタビュー調査で得られた情報を加えてさらに検討を行う。

5.8.1. 分析のフレームワーク

この節では，知識受領の成功体験についてインタビューによって質的な情報を得て分析する。分析のフームワークを図 5-5 に示す。ここでは，知識移転のフレームワークの中で受領者の行動に焦点を当てた。知識移転における提供者と受領者の間で知識の品質確保や両者の仲介を行う「支援体制」，受領者が知識を取り込む「理解」行動，受領者が実際に知識を自己の開発に生かす「適用」行動を分析する。上司の行動や周囲の反応については本章では触れていない。

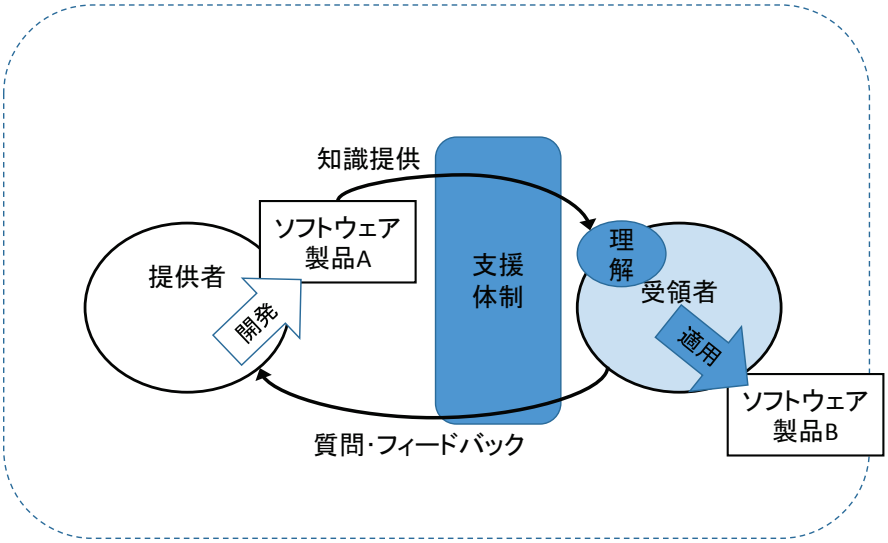


図 5-5 分析のフレームワーク

5.8.2. インタビューの方法

インタビューの対象者は前章までで使った質問票調査（原則として無記名解答）の際に，インタビューに応じる前提で氏名を回答した者である。インタビューは半構造化形式で，以下に着目して行った。

- 受領したソフトウェアについての理解
  - ➢受け取ったソフトウェア資産を利用者としてどのように理解したか。また，どの程度理解できていたか。
- トラブルへの対応
  - ➢適用においてどのようなトラブルがあったか。
  - ➢トラブルはどうやって解決したか。
- 他者知識を活用するにあたっての希望
  - ➢どのような環境が用意されると知識活用が進むと思うか。

第 5 章の調査において「他者資産を活用して，開発が楽になった経験がある」に対して「大いにあてはまる」と回答した者は 33 名(有効回答中 10.5%)いた。このうち遠方事務所に勤務しているなどの理由で条件が合わない者を除き 25 名に対して対面インタビューを申し入れ， 8 名が受諾した。それぞれのインタビュー条件については，付録２に示す。

対象者が受領，活用したプログラム資産には「社内で開発されたソースプログラム」のほかに，「社外で開発した OSS」や「社内で開発したバイナリプログラム [21]」を部品として使ったケースもあった。対象者の属性を表 5-7 に示す。なお，対象者 A, B はそれぞれ 2 種類の知識資産を利用したと回答した。

表 5-7 インタビュー対象者

対象者	開発製品種別	受領プログラム種別	入手手段
A	製品1	ソースプログラム	社内から提供
	製品2	ソースプログラム	関係会社から購入
B	製品3	ソースプログラム	オープンソース
	製品4	バイナリプログラム	社内から提供
C	製品5	ソースプログラム	オープンソース
D	製品6	ソースプログラム	社内から提供
E	社内ツール	ソースプログラム	オープンソース
F	製品7	ソースプログラム	社内から提供
G	製品8	ソースプログラム	社内から提供
H	製品9	ソースプログラム	社内から提供

インタビューはそれぞれの対象者について 1 対 1 の面接形式にてそれぞれ約 30 分間行った。ただし，対象者 B, C については事前に質問票の検討に加わったプロコミュニティのメンバーが加わり，2 対 1 とした。

インタビューの内容は受領したソフトウェア活用するに当たって実施あるいは心がけたこと，およびその際の苦労・トラブルの内容調査である。インタビューの冒頭で当方の調査の目的を説明した後，表 5-7 の開発製品種別，受領プログラム種別，入手手段について確認した。その後，受領したソフトウェアの理解の状況や理解するための方法，トラブルの内容とその解決手順について確認した。最後の時間で，ソフトウェアの社内利用を進めるための仕組みについて見解を尋ねた。

21 コンパイラによる翻訳が終わっており人の目でプログラムの内部を理解したり改変したりできない

5.8.3. 調査結果

(1) 受領ソフトウェアの理解

8 人 10 項目のソフトウェア知識の活用事例に対し，1 項目はバイナリプログラムの活用であったためソースプログラムから内容を理解することはできない。これを除いた残りの 8 人 9 項目について確認する。

まず, 受領したソフトウェアのソースプログラムにドキュメントが付随しているならば，それに頼っている。しかし，ドキュメントはソースプログラムをわかりやすくする効果があるものの, それによってプログラムのすべてを理解することはできない。また, 提供にあたって提供者からプログラムの説明を受ける機会については回答されなかった。受領者はソースプログラムを読み込むことで受領した知識を理解するのであるが, すべてを理解したケースは 1 件のみであった。多くは，「概ね理解している」レベルであった。また，トラブルがない限りはソースプログラムを読まないというケースが多かった。

また，理解のために有効な手段として，開発者に直接質問することも挙げられる。たとえば，インタビュー対象者 C は社外の OSS の開発者に直接コンタクトして知識を得ている。社内の場合でも，対象者 D の場合には，身近な質問しやすいところを提供元として歓迎している。5.7.2(3)で述べた周囲への関心があることが知識受領につながるという結果と関連がある。

この結果は以下の発言から確認できる。

A（製品 1 向けの受領）： 社内から提供を受けたケース

「情報がない。なんでこういう操作にしなくてはならないのか，よくわからないものがザーッと並んでいて」「同じような感じで作ってみた」[22]

A（製品 2 向けの受領）：ソースコードだけ買ってきたケース（ドキュメントはドイツ語だったので役に立たず）

「誰も 100%は理解していない。担当に分かれて理解したところはドキュメント化したけれど」[23]

---

[22] 2009/5/28 A 社横浜事務所でのインタビューより
[23] 2009/5/28 A 社横浜事務所でのインタビューより

B（製品３向けの受領）：OSS を適用

聞き手：ソースは全部手の内にあるはずだが，完璧には理解していないということか？

「そうですね。」

聞き手：全部理解するなら人から貰う価値がないという声もあるが

「そうですね。」

聞き手：全部理解することは最初から考えなかったのか

「ある程度までの理解。ソースの１行１行まで理解するのは骨の折れる作業で，だいたい何をやっているかを斜め読みのレベルで理解する。」[24]

C：OSS を適用

「全部自分が作ったものと同じ様に理解している。研究所の人が我が物にした。オープンソースをもらって，オリジナルの筆者に問い合わせながら自分のものとした。」

聞き手：規模の大きなオープンソースは完璧には理解できていないケースが多いが，ここでは自分の物になっているのか。20Ksteps もあるとなかなか完璧にならないものではないか？

「研究所を離れて他の関係会社に移したら理解が甘くなった」

聞き手：研究所でわかっていても引き継げなかったのか？オープンソースを社内（研究所）に持ってくるよりも，研究所から社内の別組織に動かす方が大変なようだが

「できる人，できない人個人差がある」

「XSLT[25]に関しては，自分で作る選択肢もあったと思う」「Java 版の XSLT は OSS を持ってきたのですけれど，XSLT には C++系のヤツもあって，そっちの方は自分で作ったんです。」「もうちょっとアーキテクチャを考えて，より高速なものを作ろうというモチベーションがあった。けれども，ドラスティックには変わらなかった。」

聞き手：プログラムには開発者の個性が入るという考えをどう思うか？

---

[24] 2009/5/28 A 社横浜事務所でのインタビューより

[25] XSLT とは XML で記述された文書を他の XML 文書に変換するための言語である。ここでは XSLT で記述された変換を実行するソフトウェアのことを指している。

「大規模開発において，プログラムに個性を入れるのはわがままだ。」[26]

D：

「OS に近いところの機能のソースコードをまるまるそのまま取り込んだ。」

「ちょっと離れた同じ製品を作っている人たち」「コンタクトを取りやすい人。会計が一緒[27]。」「ソースコードが二つになった。」

「見る方が大変。設計書がないと１行１行見ることになって大変」「作るときは自分で流れを考えている」[28]

E：SE ツール向けに OSS 部品を適用

「自分で作るのもつらいし，何かないかと思って探した」[29]

F：

「『古いものだから，保守はやりたくない』『ソース上げるからやるなら持って行って』と言われた」

聞き手：組織として既存製品を持ってくる意思があるのか

「あります」

聞き手：自由が利かなくなることがあるのか

「はい」[30]

H：

「短期間で開発するためには，品質が確保されているものを取り込みたい」「規模も 6 Ksteps ほどあった」「マクロのインタフェースを合わせなくてはならなかった。」「精度が違っていた」[31]

---

[26] 2009/5/29 A 社横浜事務所でのインタビューより
[27] 損益計算を合算する組織にいる人の意味。本論文ではチームを共にする人。
[28] 2009/6/1 A 社沼津事務所でのインタビューより
[29] 2009/6/1 A 社沼津事務所でのインタビューより
[30] 2009/6/2 A 社横浜事務所でのインタビューより
[31] 2009/6/5 A 社横浜事務所でのインタビューより

(2) トラブルへの対応

トラブル対応の実体については，インタビューから以下のことが明らかになった。

トラブル対応を提供側に頼れるか，あるいは受領側が対応しなくてはならないかについては，両者の関係性に依存する。両者が会話しやすい関係を持っていれば受領者からの質問を含めて会話を進め，両者で問題解決をしている。しかし，提供者の顔が見えていない場合や関係性が薄い場合には，提供者を探し出して頼ることはせず自力解決している。自力解決には受領者の技術力と時間（コスト）を必要とする。しかし，受領の成功者は解決に時間を掛けたとしてもすべて自力で開発するよりは工程を短縮できたと述べている。つまり成功者の視点ではあるが，トラブルを恐れて知識受領を拒むことは得策ではないという結論を出している。

また，提供されたプログラムを初期適用する際のカスタマイズにおいては，インタフェースを適合させることなどで苦労しながらも自力解決している。

一方，オリジナルプログラムのバグであることが判明した場合には，提供側に修正を依頼しているケースと，自力で解決して提供側に伝えていないケースが見られた。提供者に修正させる場合においても，バグの指摘を受けることで提供者はフィードバックを受けて自らの製品を改良することができるので，提供者は知識提供のメリットを得ることができる。提供者にフィードバックしない場合には，修正のコストは不要であるものの，知識提供のメリットも得られない。今後の提供者側のモチベーションを上げるチャンスを失うことになる。

A-1：

聞き手：その中で理解していないの？

「理解していないわけではないが，細かいところがよくわからない。どうしても」

「自力で調べて問題解決した。」「元のソースにフィードバックしなかった」[32]

A-2：

「開発全体としては楽になったけれど，テストはそれなりにヘビーだった」

聞き手：自力で作った方がよかったのでは？

「それはない。2 年間もらえればゼロから作った方がよかったと思うけれど，1 年間では無理でした。わからないながらでも使うしかなかった」[33]

---

[32] 2009/5/28 A 社横浜事務所でのインタビューより

B-1：

「全体のレビューはしない。トラブルがあったら，そこだけはほぼ完璧に理解する。」[34]

B-2：

「無償提供だが，提供元が保守してくれる。たくさんのバグを指摘した。」「我々のおかげで元の品質がよくなった。」[35]

D：

「なんで動いているかわからないというのが大変」[36]

(3) 他者資産を使うための環境

最後に他者資産を使いやすくするための環境について質問した。この質問には二つの観点がある。ひとつは使える知識資産がどこにあるか，探す手段があることである。これに対して現実には人脈に頼っていることが多いが，共通部品を置くリポジトリが用意されていることを期待する声もあった。

もうひとつの観点は，他者資産の信頼性の確保である。調査前のプロコミュニティでの議論においては，他者資産を導入することによりトラブルを招くことを恐れる声が多かった。他者のトラブルに巻き込まれるくらいならば自ら開発した方が効率が良いという考えは主流であった。そのため，第三者による品質管理体制を期待されるものと予測していた。

インタビューではトラブル発生時に修正作業等で実際に作業をしてくれる開発元が責任を持つことの方が信頼できるという声もあった。さらに，自らが提供者の立場になることを考慮して，提供者の負担にならないように，共通部門の仲介を期待する声もあった。しかし，第三者による品質管理は否定しないものの，多くのインタビューの対象者は，共通の品質管理部門による品質保証に対しては大きな期待をしていないと回答

---

[33] 2009/5/28 A 社横浜事務所でのインタビューより
[34] 2009/5/28 A 社横浜事務所でのインタビューより
[35] 2009/5/28 A 社横浜事務所でのインタビューより
[36] 2009/6/1 A 社沼津事務所でのインタビューより

した。なお，信頼性については受領者としては過度の期待をしていない声があがった。自分で作ったとしてもバグがあるので，バグを気にして他者から貰うことを警戒するのは意味がないという考えである。

A：

「利用者としては作成者が管理していて話せる状態がよい。しかし，提供者の側に立てば，提供後にいろいろ聞かれるのはイヤなのでまとめてくれる部門がほしい。」[37]

B：

「共通の管理部門がニュートラルでよい。個別だと聞きにくい。」[38]

D：

「専門部署はやることが発散する。仕事の密度が薄いから期待できない。検証部も同じこと。」「ドキュメントは裏切らない」[39]

E：

「第三者テストは魅力ではない。どうせ自分でやるから。とりまとめも不要だ。」「とりまとめはニーズに対してあっせんすることを期待する。」[40]

F：

「検証部はテストして終わりだから期待しない。開発部門は直してくれるからよい。」「テストは自分たちできちんとやる。」[41]

G：

（作るのも簡単だったのではないか？）「すでに部品として確立しているこの規模のものを自分で作るのはかなりリスクが高い」「それなりに専門知識が必要な

37 2009/5/28 A 社横浜事務所でのインタビューより
38 2009/5/28 A 社横浜事務所でのインタビューより
39 2009/6/1 A 社沼津事務所でのインタビューより
40 2009/6/1 A 社沼津事務所でのインタビューより
41 2009/6/2 A 社横浜事務所でのインタビューより

ので，それを知っているならいいけれど知らないので，そこまでさすがにやれない」[42]

H：

「個性は封じるべき。芸術ではない。」

「ソースプログラムを流通させるためには標準化しておいた方がよい。趣味じゃないのだから。」[43]

5.9. まとめ

この章においては，A 社に対する質問票調査による実証分析結果を行った後に，知識受領の成功経験者に対してインタビューを行い，知識受領者の動機とためらいについて質的な補完を行った。

質問票調査を踏まえた議論では，活用しやすい他者のソフトウェア資産としては，

a. 「モジュール化」されていて理解しやすい
b. 第三者による保証

という環境の整ったものという解答が得られた。知識活用者（受領者）は使用者としての責任を全うするために，他者知識を利用したことを原因とするトラブルに巻き込まれることを恐れている。この懸念を払拭するために，責任ある第三者検証体制の整備[44]やよい評判を広める仕組みが有効という結論である。

また，他者資産の活用に踏み切るには，制度や成績評価のような外発的な要因では他者知識活用の動機にはならず，図 5-2，図 5-3，図 5-4 に示すように「F2：周囲への関心」が高い者が他者知識活用を推進することがわかった。 つまり，組織内にチームを超えたコミュニケーションの場ができていると知識利用が促進される。この点につい

42 2009/6/5 A 社沼津事務所でのインタビューより
43 2009/6/5 A 社横浜事務所でのインタビューより
44 第三者検証は製品化したソフトウェアを受領する際には必ず済んでおり，特に新しい体制を整備する必要はないはずである。

てはインタビューの結果でも知識の提供者を探す行動[45]に現れている。基本的には探すというよりも話しやすい技術者のところに相談に行くという行動である。知識の提供者を探す行動は，3.4.1(2)で述べた高次の場において効果的に実践されていた筈である。また，知識のマッチングを取るための組織が存在するとよいという考えも示されている。但し，現実には研究対象の組織ではマッチングのための組織は存在していないため，知識受領（利用）は個々の技術者の力に依存している。

また，知識提供について調査・分析した第 4 章で挙がった上司の意向を慮って提供をためらう行動に対応して，知識受領の可否についての慮り行動については，本章の分析結果からは現れなかった。上司は外部知識を導入するか，それとも自力開発するかには関心がないと考えている結果である（5.7.2(1)）。実際に知識受領に関しては自らのチームの知識資産を流出させるわけではないので，上司から直接的に禁止される懸念はほとんどないと考えるのであろう。ただし，データから導かれてはいないが，前述した高次の場の衰退は，高次の場の活動に参加することに対する上司の意向への慮りによって阻害されているとも考えられる。

また，上級技術者の開発効率向上意識が他者知識活用につながるとも言える。上級技術者はチームの方向性を決めるポジションにいる。そのため，上級技術者を中心に啓蒙活動を進めることが有効であると考えられる。

上記より，他者知識（ソフトウェア資産）の活用を促進する条件としては，

- 信頼できる品質検証体制
- 高次の場の活性化
- 上級技術者に対する開発効率向上意識

を進めることが必要であると言える。

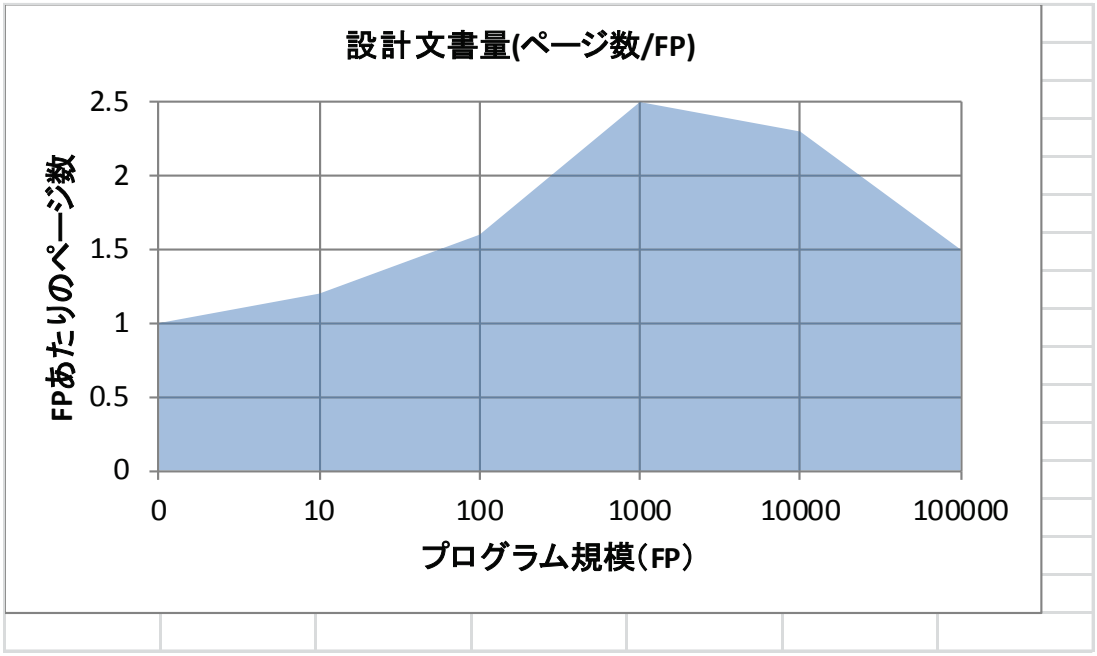
インタビューからは他チームの製品の一部や社外のコミュニティが開発した OSS の活用にあたって興味深い知見として，活用にあたってすべてを理解しようとしていないケースが多いことが得られた。すべてを理解することは確かに困難であり，実際に開発者であってもすべてを理解していないし，開発向けのドキュメントであってもすべてが記載されているわけではないことが報告されている。一方で，理解することを前提条件

[45] インタビューにおいて，知識を探す先として「ちょっと離れた同じ製品を作っている人たち」「コンタクトを取りやすい人。会計が一緒」という声が出ている。

とした結果，他者資産を使うことに踏み切れないケースもある。しかし，他者資産活用の成功者であるインタビューの対象者たちは，理解することを前提条件としないで「使えるように使う」という行動を取っていた。理解することにストイックになって過剰な労力を費やしたり，理解できないことを理由に知識の活用をあきらめたりせずに，楽観的ともいえる態度で行動しているのである。その代わりに，テストを充実させるという代替手段を導入することで，ソフトウェア知識受領を成功に結びつけたといえるのではないだろうか。

確かに，完全な理解をしていない状態でトラブル対応ができるのだろうかという懸念は当然のことであろう。インタビューの結果は楽観的に過ぎることはないだろうか。実際にインタビューで聞けたトラブル事例の多くは，システム構築作業，つまり初期作業の範疇に入り，プログラムの内部を細かく理解するよりも外部とのインタフェース部分の修正が必要な事例が多かった。そのため，インタビュー対象者の経験が容易なものに偏っていて，楽観的な見解が多く得られた可能性はある。一方，内部の論理を理解する必要があったという例も少数ながら得られた。特に，受領者が自力解決せざるを得ない場合にはこの懸念が現実になる。実際に解決に苦労したケースも聞かれたが，その場合でも，最終的には問題を解決させ「自分でゼロから作るよりは早く完成した」ということで，他者の知識を導入することに否定的な見解を述べる例は見られなかった。

ドキュメントに頼ってソフトウェアを理解することに関して，Jones (1999) はソフトウェアプロジェクトの調査の結果として，OS の開発のような大規模なプロジェクトにおいてはソフトウェアの規模を示す FP（Function Point）値に対してドキュメント量が少なく（図 5-6），ドキュメントの完全性が満たされなくなること（図 5-7）を指摘している。つまり，オリジナルの開発者向けにさえも大規模なソフトウェア向けのドキュメントは不完全である。それでもソフトウェアが開発できることについては，「答えは単純に人がシステムを作るからである。われわれはミーティング，会議，電話，メモ，その他のコミュニケーションにより部分的設計を補って仕事を進めていく。もし記述された仕様書のみに頼るのであれば，大型システムの開発はおそらく不可能であろう。」（Jones, 1999, p.75）と述べている。

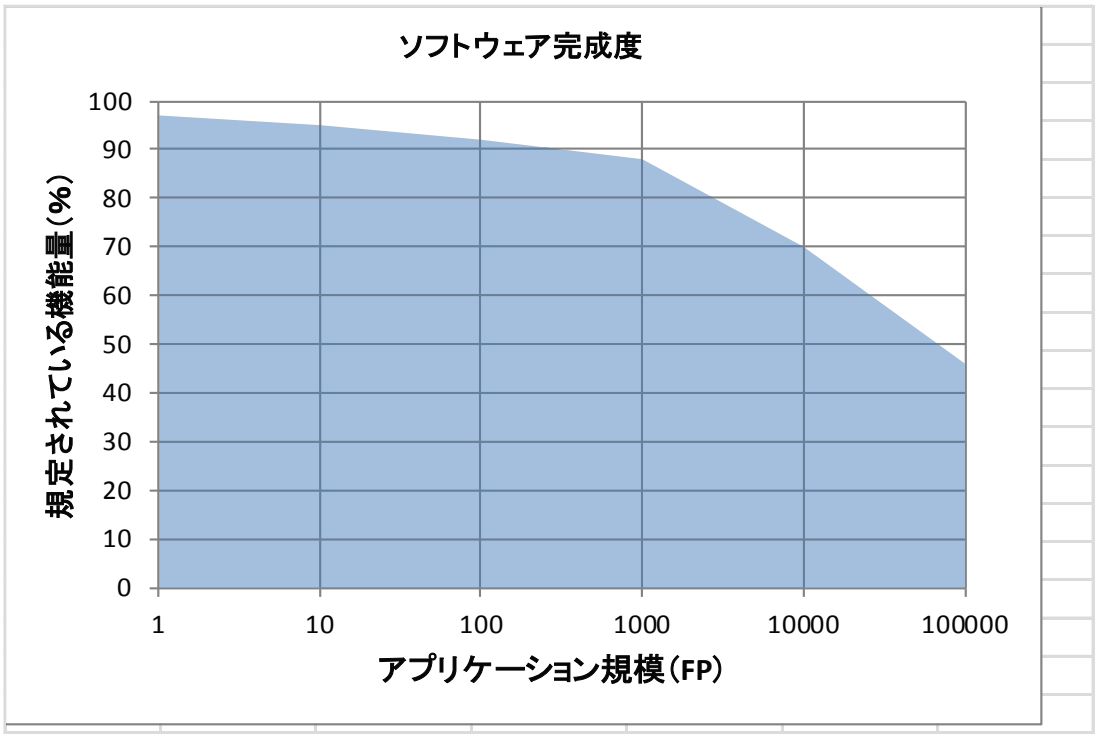


(Jones, 1999, p.74)より

図 5-6 ソフトウェア設計文書量とソフトウェア規模

つまり，不完全な仕様書に基づいても大規模システムの構築がなされているのであるが，その一方で，他者からの知識導入に当たって完全な理解を期待しても仕方ないし，その必要もないと言える。知識利用の成功者たちはこの現実を捉えて行動しているものと考えられる。

また，他者資産が含んでいるバグについては，自分で開発してもバグはあるということで，利用の妨げになるとは思っていないという実態が明らかになった。この点から，受領者の行動において図 5-5 に示す「理解」，「適用」の観点では，受領するために特別なコストを掛けずに進めることが成功のための要件であると考えられる。



（Jones, 1999, p.75）より

図 5-7 仕様書に規定されている機能量の率

知識資産の管理，特に品質検証については，受領の成功者たちは制度に多くを期待していない実態が見えてきた。オリジナルソースプログラムの開発元（提供元）が自分のために検証していることは当然と誰もが考えている。その上で共通部品として公開するために，検証部門がさらに何らかの品質確保をすることに対する期待は大きくないということである。いずれにしろ受領者は再度テストを行うので，部品の流通のための検証部門の活動には期待しないということであった。

この結果は質問票から得た結論と相反しているようにも見える。実際に各回答者の質問票への回答結果では，検証部門による品質確保が好ましいかという質問に対しては「大いに当てはまる」あるいは「当てはまることが多い」という回答をしている。これに対して筆者は，質問票での調査では机上（PC 上）での質問であり検証部門の活動に対して否定するものではなく，むしろベースラインの活動として当然のものと捉えたものと解釈した。上でも述べたように，提供元の製品のために検証は行われているのであり，検証部門による品質確保がゼロになることは誰も想定していない。一方，本章でのインタビューへの回答は，ベースラインに対しては当然のものとした上で，それ以上に

検証部門に期待することはできないという考えから「多くは」期待しないという回答になったと考える。

また，成功者たちは受領したプログラムに含まれるバグについても深刻には考えていない。自力で作った場合でもバグが入る可能性は十分にあると考えているからである。一方で，使える知識（ソースプログラム）を効率的に探せる環境は求められている。人脈に頼らなくても社内のソースプログラムが一括管理されているデータベースが有効という意見である。

知識受領が大いに役立った経験を持つ技術者たちは，ソースプログラム資産を使うにあたって，大きなサポートがあれば助かると考えているものの，それに対して特に期待することもなく楽観的に振舞って活用していると考えられる。技術者たちは他者が作ったソースプログラムを自分が開発するプログラムに取り込むために，特に多大な準備をしたり内容を深く理解したりせずに利用したことで，自らの開発コストを下げることができたと実感している。成功者たちに他者のプログラムを活用するために余計なコストを掛けたという意識はない。

受領行動の成功者の経験から考えると，提供者が知識提供するために大きなコストを払うことは期待されていない。提供者自身が開発時に作成したソースプログラムや付随する文書をそのまま公開すれば受領者にとって十分と考えられる。また，組織としての「支援体制」（図 5-5）にコストを掛ける必要もない。知的資産の存在を示し検索するためのデータベースを準備することで十分であり，品質保証等には新たな体制を整えることは期待されていない。

A 社のソフトウェア開発部門のケースでは，知識移転のプロセスは受領者が外部知識を取り入れたいと思い，提供者を探すところから始まる。受領者が探す対象は既知の間柄（コンタクトを取りやすい人，「会計が一緒」）に限られる傾向にある。そのため，知識共有を推進するマネジメントとしてはニーズへのマッチングをすることが期待される。しかし，受領者は知識移転をサポートするための厳格さや手取り足取りのサービスには期待していない。たとえば，知識共有のために品質管理を強化することは期待を超えている。マネジメントを厳格にすることは，知識導入を却って難しくしてしまい，知識共有を阻害することになる。それよりも，あいまいな管理体制，あるいはホットグループ（Lipman-Blumen & Leavitt, 1999）のような活動様式が期待されている。このことは，第 3 章で述べたあいまいな組織運営による適切な「さじ加減」が有効であることと合致している。マネジメントの立場から自由な行動を容認することこそが知識移転を活発にさせる要因になるものと考えられる。

第6章 結論

ここまでソフトウェア開発を行う企業内での知識移転に関して，a. 知識提供側における促進，阻害要因と業務環境との関係を見出すこと，b. 知識受領側における促進，阻害要因と業務環境との関係を見出すことの二つの目的で，リサーチクエスチョンへの回答を求めて研究を進めてきた。本章では，これまでに述べてきた事項からリサーチクエスチョンへの回答をまとめ，さらに本論文の学術的，実務的な含意について述べて，本論文を締めくくる。

6.1. リサーチクエスチョンへの解答

本節では 1.4 節で立てたリサーチクエスチョンに対して順に解答を述べていく。

6.1.1. SRQ1 への解答

「自ら開発した『知的資産』をチーム外に開示する行為をためらわせる要因は何か？」

第 4 章において知識提供行動の分析を行った結果, 自らの知識が詰まった知識資産をチーム外に開示するに当たって，開示作業に伴う労力への懸念や，他者に得をさせた結果として自己の評価を相対的に低下させることが障害になることはないことがわかった。一方, チームに属するものとしては, チーム外に開示することはチームの規範によってゆるされないのではないかという懸念があることが明らかになった。このことから，SRQ1 に対しては以下の解答が得られる。

組織の規範や上司の判断に対する懸念から，自らの知的資産をチーム外に開示することをためらっている。

知識提供をすることに対してインセンティブをあたえることは上記解答で述べている懸念を払拭する材料にはなるだろう。しかし，もともとインセンティブの不足や手間

を惜しんで開示しないわけではないので，新たに外発的な動機付けをすることの効果はない。インセンティブ本来の目的とは異なるが，インセンティブのルールを掲げることによって，組織の方針が明示されためらいを払拭することができるため，知識開示を促進することが期待できる。

6.1.2. SRQ2 への解答

「他者の『知的資産』の活用を推進する要因，ためらわせる要因は何か？」

第 5 章での分析結果によって，まず，外発的な動機付けによっては他者資産の活用を推進することはできないことが示された。一方，チーム外に情報網が広がっているときには知的資産の活用が進んでいることが示されている。さらに，技術者の技術レベルによって異なるプロセスを適用する必要性も示された。特に上級技術者が開発効率向上への意識を高める活動が有効である。

5.8 のインタビュー調査の結果では，活用の成功者たちは知識受領の困難さは理解したうえで，他者資産を取り入れ活用していた。彼らは知識を他者から取り入れるリスクと自ら開発するときにも発生するリスクを同等と考えたうえで，他者からの受領の方が開発コストが少なく済みそうと判断して，知識を受け入れていた。組織のルールや支援には関わりなく，目標に向かってあらゆる手段を取る姿勢がうかがえる。このことから，SRQ2 に対しては以下の解答が導かれる。

開発の目標に向けてチームで知識共有を進める推進要因は，技術者がチーム外に情報網を広げていることと，目標達成のためには手段を択ばないマインドである。（高次の場あるいはホットグループはこのマインドを持つ。）

組織運営のルールを厳格にしたり組織的な動機付けを進めて知識移転でのトラブルを回避しようとしても，他者資産の活用は推進できない。チーム外に情報網が広がっているときにこの活動が進むので，「あいまいな組織運営」によって技術者を自由に活動させることが有効と考える。

6.1.3. SRQ3 への解答

「活用されやすい『知的資産』とはどのようなものか？」

第 5 章の議論では活用しやすいソフトウェア資産の要件として a. モジュール化されていること，b. 第三者による保証がなされていること，の 2 点が挙げられた。a.のモ

ジュール化についてはソフトウェア工学の観点からも古くから述べられていることである。b.の第三者の保証についてはA社においては開発部門とは独立の検証部門が存在するため特筆すべきことではない。

5.8のインタビューではこれに反するように，第三者による品質保証には期待しないという声も聞かれた。この相違については，前述のように第三者検証を当然のものとして捉えた上で，それ以上の（知識移転のための特別な）検証には期待しないという見解と理解した。決して，第三者検証は不要ということではないだろうが，責任を持ってくれるのは（自分の他には）オリジナルの開発元以外にはないという理解である。

これによりSRQ3に対しては以下の結論を導いた。

モジュール化がなされており，一般の製品として通用しているものであれば，他者資産を活用できる。

この解答はソフトウェア工学において古くから言われてきた理想的なソフトウェア製品と同等である。つまり，ソフトウェア資産（知識）のチーム外への移転に際して，それ以上の特別な要件は発生しないことを示している。

実際にソースプログラムを開発するにあたって，開発者がモジュール化するのは，品質を高め，保守しやすくするためである。つまり，自己の利益のために開発技術を駆使しているのであり，他者に提供するためにモジュール化するわけではない。モジュール化の巧拙は開発者のスキルレベルに依存する。スキルのある技術者が開発したわかりやすいモジュール構造は，受領者に対しても扱いやすいものになることは多い。そのため，上記の「モジュール化がなされており」ということは開発者の技術レベルを要求しているといえる。

6.1.4. MRQへの解答

どういう組織環境が社内の『知的資産』の活用（ソースコードとして改造，取り込み）を促すか？

本論文で立てたメジャーリサーチクエスチョンに対して，SRQ1～3への解答から以下の解答を導いた。

組織の方針として知識移転を明確に推奨しながら，技術者に自発的な判断を促しそれを過度に管理しないことが知識移転を促し，社内での知識資産活用につながる。

知識移転を円滑かつ活発に進めることで，ソフトウェア開発力を強化することができるものと考えてこの研究を始めた。知識提供の阻害要因としては，SRQ1 の解答に挙げたように規範や上司の指示に対する懸念によるものと結論付けた。一方，知識受領の促進要因としては，SRQ2 の解答としてチーム間の交流が活発であることとした。逆に考えると，チーム間の交流が停滞すると知識受領が阻害されるものと考えられる。A 社におけるチーム間の交流の停滞は第 3 章で述べたようにビジネス意識の強化から始まった現象である。しかし，ビジネス意識を強化するために，チーム間の交流の停止を指示されたわけではない。知識提供の阻害要因と同様に，上司の意向を慮った結果として交流が停滞しているだけである。このことから，知識移転が滞る原因は，提供，受領ともに開発現場の従業員に取って知識移転は自らの属するチームに取って悪いことなのではないかという懸念にあることが明らかになった。そうした懸念を払拭するために組織の大方針を明確にすることが社内資産の活用につながるものと言える。

当初考えていた移転に対する困難さや開発コストの上昇については懸念には及ばない。上位の意向を慮っている一方で，知識移転をすることで高い評価などの報奨を得たいという意識は見えてきていない。

このことから，組織の方針・規範を明らかにした上で自由に考えて行動させる環境を作れば，技術者たちは自然に知識流通を活発にさせるものと予想できる。つまり，管理体制をゆるめ，3.4.1 で述べた階層的な場の再生や 3.4.4 で述べたホットグループの育成が目的を達するための第一歩になるものと考えられる。

6.2. 理論的含意

ソフトウェア開発の現場において，特に知識受領側の動機，行動に対して検証し，知識創造の理論に結びつけた例はこれまでなかった。ソフトウェアは厳密な正確さを要求する製品であり，ソースプログラムはソフトウェアの動作のためには過不足なく文書化されている形式知である。そのため知識移転は単純な形式知の引き渡しに過ぎないように見える。しかし，現実にはソースプログラムやドキュメントの引き渡しだけでは知識

の移転が完了しない。引き渡された形式知を活用することの困難さを理解している技術者たちは，他者からの知識を簡単には受け入れようとはしない。

ソースプログラムは形式知であるが，共有対象のモジュールだけではプログラムの動作を完璧に記述できていない。明示されていない動作環境の条件によってプログラムの動作が変わるからである。また，ソースプログラムの開発者が特に留意して作った部分については，複雑だったり，一見不要な記述に思えたりする。開発者がなぜそのようなことをしたのかを理解できないと受領者は移転された形式知を正しく使うことができない。このとき，開発者が自らの思考を完全にドキュメントにしていれば受領者は容易に知識を共有することができる。しかし，実際にはドキュメントが完璧であることは期待できない。そのため，受領者はモジュールの背景知識を理解することが要求される。背景知識は受領者が読み解くこともできるし，開発者や利用者のコミュニティでの議論から獲得することもできる。多くの場合，利用者のコミュニティで獲得する方が効率的と言える。このことから，ソフトウェア知識は形式知と暗黙知が一体になったものと言える。

本論文では，知識移転の背景にあいまいな管理体制が存在することと，SECI プロセス（野中 & 竹内, 1996）における特に共同化（Socialization）フェーズでの知識創造を結びつけた。ソフトウェアモジュールの背景とも言える環境条件や留意点などはソースプログラムには直接書かれていない。これがドキュメント化されていない場合には開発者の暗黙知としてしか存在しない。この暗黙知は形式知であるソースプログラムやドキュメントと一体になって共有される必要があり，共同化によって共有されることが求められるのである。そのため，知識創造理論における共同化フェーズにおいて，非公式な小組織といえる場（高次の場）の中での知識共有，知識創造が重要な位置を占める。知識移転のルールを決めたり，移転のための品質保障活動を開始したりするような管理を厳格にすると，知識移転の当事者（提供者，受領者）が属する場の活力が阻害される恐れがある。異なる知識を持ち寄るためには管理をゆるくすることが有効である。

図 6-1 に示したように，知識移転を行う提供側と受領側の両チームの技術者はそれぞれの上司の管理の元で開発活動を行い，その活動の一環として知識移転を行っている。提供者はチームの資産をチーム外に提供するにあたり，上司の意向を確認する必要がある。この時，部下である提供者が自分の上司について知識共有に否定的だと推測すると確認行動を取るまでもなく知識提供はできないものとして行動を停止する。部下（提供者）が否定的な予測をする可能性として，上司の日常の言動から収益性を強く意識していることが読み取れる場合が考えられる。

部下は上司に対して「他チームに資産を開示することは上司に反対される」とまでは思っていない。しかし，上司が賛成するとの確信までは持てていない。一方，受領者の視点でも上司が他者資産の活用に対して積極的とは見ていない。このことは知識共有の阻害要因として働く。

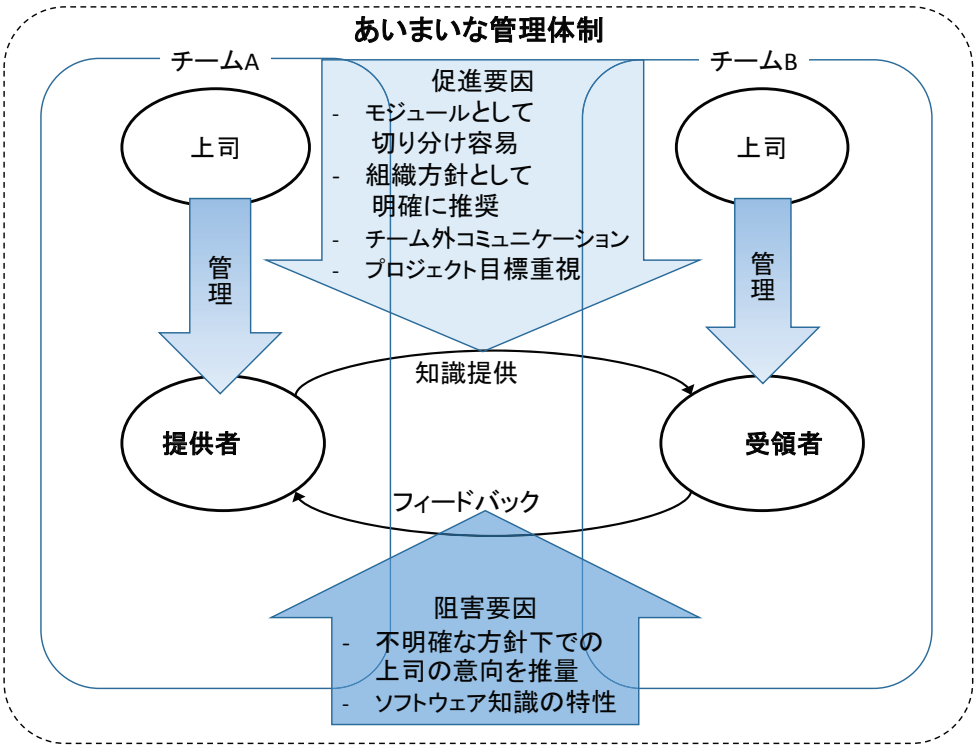


図 6-1 知識共有 促進・阻害要因

知識を流通させるためには，チーム間のコミュニケーションが必須である。これを活性化させるに当たってはチームの損益意識のようなチーム間に壁を作る要因は阻害要因になる。逆に，あいまいな管理は壁を低くしてチーム間のコミュニケーションを容易にし，知識共有の促進要因となる。

明確なプロジェクト目標を持つことは知識共有の促進要因として働く。プロジェクトの目標を達成することを最大の目標として手段を選ばない状況になると，他者の資産にわからないところが残っていても適用している例がインタビューから見出された。シリコンバレーのホットグループ（Lipman-Blumen & Leavitt, 1999)に近い状況になっていると考えられる。

本研究では，ソフトウェア知識の特性が知識共有の促進要因にも阻害要因にもなることも明らかとなった。もし，ソフトウェアがモジュールとして綺麗に切り分けられていて，環境（文脈）依存性が排除されていれば知識移転は容易である。しかし，多くのソフトウェアはソースコードに書かれた形式知の側面のほかに暗黙知の性質を持ち，さらにバグを内在するような不完全性や実行環境に依存する振る舞いをする文脈依存性を持つ。この性質は知識共有の阻害要因になる。

6.3. 実務的含意

本研究を通じて，ソフトウェア開発における知識移転の動機と阻害要因を分析し，技術者たちが知識移転にためらう状況を理解した。そのことから，ソフトウェア開発組織の運営について，特に組織のマネージャ層に対して以下のような実務的含意を提示することができる。

まず，技術者の知識移転に対するモチベーションを高めるには，特別な動機付けの行動を取るよりは方針の明確化により疑心暗鬼の回避が有効であることを説明できる。ソフトウェア開発は技術者個人の活動の集合であることから，個人の自律性を向上していくことが必要である。理論的含意で述べたように，あいまいな管理体制が自律性を喚起し，内発的モチベーションを高め，知識移転行動を促進する。逆に，管理を厳格にして，例えば品質管理体制を強化すると，技術者たちを固定的な活動形式に閉じ込める結果につながる。また，チームレベルでの損益管理を厳格にすると他チームとの知識資産のやりとりは自チームに対する背信行為とみなされかねない。そのためイノベーティブな活動が阻害される要因となる。

一方，知識移転をためらわせる要因には，組織の規範や上司の意向に対する過剰な懸念による要素が強い。マネージャ層はこの懸念を払拭することを中心的な改革方針とするべきである。管理をあいまいにしながらも，知識移転の方針を明確に示すことが実務的に有効である。

つまり，知識移転を促進するためにはあいまい性と明確性の使い分けが必要である。特に，知識提供側に対しては上司の意向については明確なメッセージを発信することで知識提供のためらいを除去する。一方で，知識受領側に対しては，管理をゆるくあいまいに抑え，チームをまたぐ場の行動がしやすくなるように受領する技術者たちを支援する。

厳格な管理体制は外発的なモチベーションを提供したり，禁止事項を設定したりすることにつながることがある。イノベーション組織であるソフトウェア開発組織においては技術者たちの独自判断による行動を阻害する。管理体制をあいまいなものにして個人に任せる状況をつくると，自律性が高まることが，知識移転の推進につながることを示した。

あいまいな管理と放任とは異なる。高次の場でメンバーたちの間を飛び回るマネージャはいわゆる管理職のイメージではない。経験に裏付けられた技術力や人脈を駆使して知識を集めてくる立場である。また，本研究では技術者のレベルにより受領行動が異なる事も明らかとなった。したがって，知識受領についての効果的な指導，指示の方法は，部下の技術レベルに応じて異なる。上級者に対しては仕事を任せることで自律性を強く促すが，中級者に対しては明確な指導の中で知識獲得を促していくべきである。

さらに，理論的含意でも述べたが，プロジェクト目標が明確になっていると受領側は様々な手段を用いて目標を達成しようという動機を持つようになる。その中で，他者の知識資産を導入する行動も活発になる可能性が高い。マネジメント層がプロジェクト目標を明確化することで，知識共有は促進される。

6.4. 今後の研究課題

最後に本研究の限界と将来への課題を述べる。

6.4.1. 理論的な観点での課題

本論文ではA社の基本ソフトウェアを開発する組織に着目し，知識移転の実態を調査し，促進要因と阻害要因を明らかにした。しかし，調査は一社のみにとどまっている。この手の調査を他社に広げることは困難であるが，一般性を担保できていないことは認めざるを得ない。

ただし，学会等で交流する他の企業のソフトウェア開発プロセスにもA社と著しい違いは認められない。おそらくこの評価結果が国内企業には通用するものと推察する。日本のソフトウェア産業の傾向としてこの結論を参照するのは大きな間違いにはならないであろう。しかし，Lipman-Blumen & Leavitt (1999) が述べているように，日本企業とシリコンバレーの米国企業とではマネジメント体制も文化も大きく異なる。そのため，海外組織や外資系企業にまでこの結論が適用できるかどうかについては更なる調査が必要と考える。

本論文での調査は知識移転を行う実務層に集中し，マネジメント層に対しては意識の調査は行わなかった。調査の結果からあいまいな管理体制に意味があることが見えてきたので，今後，マネジメント層に対して，自分の管理するチームの知識を他チームに提供することに対する見解を確認することで，実践的な方策を立案することができるであろう。

知識共有の動機について本論文では特に受領側に重点を置いて調査し論じてきた。しかし調査の結果から，トラブルを自力で解決した後，提供側にフィードバックしていないケースが検出された。このケースでは提供者側のモチベーションを上げるチャンスを失っているものと考えられる。再度提供側の動機についても踏み込んだ調査も必要になってきた。

上記限界を考えた上で，今後の課題として理論的な観点では以下の項目が残る。

- 調査対象を拡大し，一般性の確認。
  - A 社以外のシステムソフトウェア開発部門に広げてソフトウェア産業としての一般性の確認
  - A 社内の他のソフトウェア開発部門（アプリケーションプログラムなど）に広げて，ソフトウェアの性質による差異の確認

この課題を解決することで本論文の普遍性，有用性がさらに拡大できるものと期待する。

6.4.2. 実務的な観点での課題

実務的な観点では，以下の課題が存在する。

まず，本論文では知識移転とソフトウェア開発効率との関係は定量的な調査はできなかった。両者の関係については，質問票調査やインタビューで他者資産を利用したことで効率が上がったという声を得たにとどまっている。本来，組織としての関心事は知識移転の有無に関わらず開発効率の向上である。知識移転を行って開発効率が下がるようでは本末転倒である。ソフトウェアの開発量の評価尺度としてファンクションポイント(Jones, 1998) が提案され普及している分野はあるが，システムソフトウェアに関しては適用例が少ない。移転した知識の量の測定に関して適切な尺度は存在しない。そのため，効率の測定も困難ではあるが，新たな尺度を提案することも含めて評価を進めたうえで開発効率を導き出し，知識移転の効果を数値化することが必要であろう。

また，実際にソフトウェア開発の現場で知識共有を促進するためには，短期的，長期的の両方の視点から利害得失を分析していくことが必要であろう。本研究では短期，長期の区別をせずに質問票を作成したが，調査時点での評価，コストや作業効率に注目した質問になっており，結果は短期的な視点に偏っていると考えられる。ソフトウェア知識の暗黙知的要素の存在を認めたが，これを形式化することのコストとその効果についての分析は行わなかった。短期的な視点ではおそらくコストが大きくなるものと予想するが，長期的な視点では形式知化する効果がコストを上回ることも考えられる。

最後に，管理体制のあいまいさと厳格さのバランスについて実践的な検証が求められる。管理体制をあいまいにすると，技術者たちが自由に他のチームのメンバーと会話すり，そこから他のチームの知識を吸収することが考えられる。一方で，本来行うべきことを行わず，たとえば品質不良をおこすことも考えられる。どの程度の技術レベルの者に対して，どこまで管理を緩めてよいのかは実務上の検証を減ることが必要になるであろう。

まとめると，実際の現場で知識共有を促進してソフトウェア開発効率を高めるために，以下の研究を進めることが有効と考える。

- ソフトウェア開発効率の定量的測定
  - 実務的含意を確認するために，知識共有とソフト開発効率との関係を明らかにする
  - 知識共有の利害得失の長期的，短期的両面での評価
- 技術レベルと管理体制のバランスの実証的検証

この課題に取り組むことで，実際のソフトウェア開発現場において本研究の成果を生かすことが可能になるものと考える。

付録１　質問項目

本研究の調査に当たり，以下の 4 部構成からなる質問票を使用した。

社内の他チームで開発した資産活用の動機について

1-1 あなたがソフトウェア製品を開発するにあたり，社内の他チームで開発された資産（以下，「他者資産」と呼びます）を部品または参考資料として活用した経験についてうかがいます。

解答種別：5: 通常行っている
4: 多くの場合行う
3: ときどき行う
2: ごく稀に行う
1: 全く経験がない

Q11a 他者資産を部品としてそのまま取り込む
Q11b 他者資産を部品として改造しながら取り込む
Q11c 他者資産を参考事例として参照する

1-2 社内の「他者資産」を部品または参考資料として活用した経験のある方に，その時の周囲の反応についてうかがいます。経験が全くない方は，1-3 の質問に進んでください。

解答種別：7: 大いに当てはまる

6: 当てはまることが多い
5: どちらかといえば当てはまる
4: どちらとも言えない
3: どちらかといえば当てはまらない
2: 当てはまらないことが多い
1: 全く当てはまらない

Q12a　他者資産を活用したことで,成績評価で上司から高く評価されたことがある。
Q12b　他者資産を活用したことで，成績評価で不利に扱われたことがある。
Q12c　他者資産を活用したことで，同僚から敬意を示されている。
Q12d　他者資産を活用して，トラブルにあった経験がある。
Q12e　他者資産を活用して，開発が楽になった経験がある。

1-3 社内の「他者資産」を部品または参考資料として活用することに関して，あなたの組織はどのように行動しているでしょうか。

解答種別：Q1-2 と同様の 7 段階リッカートスケール

Q13a　上司が薦めている。
Q13b　ソフトウェアの開発量は新規開発量を重視して測定される。
Q13c　他者資産を活用することは簡単なことだと上司が考えているように思う。
Q13d　チーム内の同僚が，他者資産を活用している。

1-4 今後，他者資産を活用するとしたら，どのようなことが予想されますか？

解答種別：Q1-2 と同様の 7 段階リッカートスケール

Q14a　上司から高い評価を得られると思う。
Q14b　上司からの評価は下がると思う。
Q14c　上司は他者資産の利用の有無について関心がないと思う。

1-5 他者資産を活用する行為と開発コストや技術者の能力との関係について思うところをうかがいます。

解答種別：Q1-2 と同様の 7 段階リッカートスケール

Q15a　他者資産に触れることは, 自分の開発者としてのスキル向上につながると思う。

Q15b　他者資産を活用するには，それを十分に理解することが必要だと思う。

Q15c　他者資産を理解することは難しい。

Q15d 自分は他者資産を理解する読解力が高いと思う。

Q15e 他者資産に頼らず，自ら開発することで技術者としてのスキルが向上すると思う。

Q15f 他者資産を活用すると，自らの開発作業が楽になると思う。

Q15g 他者資産を活用すると，却って多くの工数がかかると思う。

1-6 社内での他者資産の活用状況について，どのような認識があるでしょうか？

解答種別：5: 多くの事例を知っている
4: 少し知っている
3: わずかな事例を知っている
2: なんとなく聞いたことがある
1: 全く知らない

Q16a　社内で他社資産を活用して成功した事例

Q16b　社内で他社資産を活用して失敗した事例

自チームの「資産」の社内の他チームへの開示について

2-1 あなたのチームが開発したソフトウェア資産の社内の他チームへの開示, 提供の経験と状況についてうかがいます。

解答種別：5: 通常行っている
4: 多くの場合行う
3: ときどき行う
2: ごく稀に行う
1: 全く経験がない

a 社内の他チームに無償で提供する
b 社内の他チームに有償で提供する

2-2 あなたのチームが開発したソフトウェア資産の管理状況について，もっとも多いケースを以下からひとつ選んでください。

- **パスワードロックあるいはネットワークから切り離した場所に保持し, チーム外には参照させるつもりはない。**
- **パスワードロックあるいはネットワークから切り離した場所に保持しているが，内部で定めた条件に合致すれば，申し出に対してチーム外にも開示する。**
- **パスワードロックあるいはネットワークから切り離した場所に保持しているが，申し出があれば無条件にチーム外にも開示する。**
- **ネットワーク上にロックせずに置いてあるが, そのアドレスは公表していない。**
- **ネットワーク上にロックせずに置き, そのアドレスをホームページ等で公開している。**

2-3 他チームに自チームの資産を提供した経験のある方にうかがいます。経験が全くないかたは，2-4 に進んでください。

解答種別：Q1-2 と同様の 7 段階リッカートスケール

a　有償で提供した資産が原因で，先方でトラブルが発生したことがある。
b　有償で提供した資産についての質問やトラブルで苦労した経験がある。
c　無償で提供した資産が原因で，先方でトラブルが発生したことがある。
d　無償で提供した資産についての質問やトラブルで苦労した経験がある。

2-4　他チームに自チームの資産を提供することについて，思うところをうかがいます。

解答種別：Q1-2 と同様の 7 段階リッカートスケール

a　提供された部門が一方的に利益を得る。
b　開発部門としての損益を確保することが重要である。
c　社内には自チームの資産を提供すべきである。
d　制度上，資産をチーム外に開示するのは許されないと思う。
e　他チームに資産を開示することは，（あなたの意志とは関わりなく）上司に反対されると思う。

2-5　チーム外に資産を開示するにあたり，必要と思う作業についてうかがいます。

解答種別：Q1-2 と同様の 7 段階リッカートスケール

a　自分で開発した資産を社内で広く使ってもらいたいと思う。
b　開示するためには，しない場合に比べて，資産を理解しやすい物にしておくことが必要だと思う。
c　開示できるようにするためには，多くの工数がかかる。
d　自分が開発した資産を他人に見せるだけの自信がない。
e　他チームに資産を開示すると，あとの対応に忙殺されると思う。

2-6　チーム外に資産を開示した場合，どのようなことが起こると思いますか？

解答種別：Q1-2 と同様の 7 段階リッカートスケール

a　成績評価で高く評価される。
b　社内で尊敬を集められる。
c　資産の誤りや改善項目について指摘が受けられる。
d　利用者から感謝される。
e　会社の技術力向上に貢献できる。
f　他者に助けてもらいたい時に，助けてもらいやすくなる。

活用できる他者資産の性質について

3-1　あなたが他者資産を活用するとしたら，どのような物が好ましいでしょうか？

解答種別：Q1-2 と同様の 7 段階リッカートスケール

a　開発チーム以外の第三者(検証部等)によって品質が保証されている。
b　開発者あるいは開発チームが部品としての品質に責任を負っている。
c　ドキュメントが十分に整備されている。
d　信頼できる技術者が開発した。
e　多くのプロジェクトで活用されている。
f　規模が小さく（100 行程度以下）単純で，内容が理解しやすい。
g　規模が大きく（1000 行程度以上）複雑で，自分で作るのが大変である。
h　中規模のもの（100 行超，1000 行未満）
i　ソフトウェア資産を取りまとめて管理する部署によって管理されている。
j　外部仕様が明確に定義されている。

チームの開発プロセスと開発者としての行動・自信について

4-1　あなたのチームの生産物のわかりやすさへの取り組み状況についてうかがいます。

解答種別：Q1-2 と同様の 7 段階リッカートスケール

a　資産の読解力を高めるトレーニングを行っている。
b　ドキュメント整備について力を入れている。
c　ソースコードやドキュメントの正しさだけでなく，わかりやすさにも重点を置いてレビューを行っている。
d　自チームの世代交代を意識して資産の整備をしている。

4-2　あなた自身のソフトウェア開発に対する思いについてうかがいます。

解答種別：Q1-2 と同様の 7 段階リッカートスケール

a　ソフトウェアの開発力には自信がある。
b　ソフトウェア資産を理解する力には自信がある。
c　ソフトウェアは芸術に近い。
d　よいソフトウェアには開発者の個性が反映されている。

4-3　最後に，当コミュニティの活動や本アンケートについて思うところがあれば，ご自由にお書きください。

付録 2　インタビュー記録

それぞれのインタビューの条件は以下のとおりである。すべてのインタビューにおいて，インタビュー場所は対象者の勤務地，所要時間はそれぞれ約 30 分であった。

対象者	インタビューアの人数	インタビュー日	インタビュー場所
A	1	2009/5/28	A 社 横浜事務所
B	2	2009/5/28	A 社 横浜事務所
C	2	2009/5/29	A 社 横浜事務所
D	1	2009/6/1	A 社 沼津事務所
E	1	2009/6/1	A 社 沼津事務所
F	1	2009/6/2	A 社 横浜事務所
G	1	2009/6/5	A 社 沼津事務所
H	1	2009/6/5	A 社 横浜事務所

付録３　Ａ社の組織構成と役割

Ａ社にはICTビジネスに関する複数の部門が存在する。その中で，本研究の対象としたソフトウェア開発部門は調査時点において以下のような構造を持っていた。

ソフトウェア開発本部
- Ｘ事業部
  - Ｘ１開発部
    - ａプロジェクト
    - ｂプロジェクト
    - ｃプロジェクト
    - ｄプロジェクト
  - Ｘ２開発部
    - ：
  - Ｘ３開発部
- Ｙ事業部
  - ：
- Ｚ事業部

ここで，「事業部」は適用分野に類似性がある複数の部を束ねる組織単位である。たとえば，ＯＳの開発であったり，ソフトウェア開発環境であったりが対象になる。

各部は事業部の下で独自の製品を開発する。あるいは事業部が開発する大きな製品の一コンポーネントの開発を請け負うこともある。ビジネスとして部の単位で完結する場合もあるが，事業部の単位に上げないと完結しない場合もある。本論文では，部をチームという言葉で扱った。

部の中に複数（３～５程度）の「プロジェクト」が存在する[46]。ここでは上位組織単位である部の役割を分担して開発業務を行う。部の一部を担っており，独自の製品がないので，プロジェクトとしての損益は考えられない。また，同一部内のプロジェクトの役割分担や人員異動は部長の裁量で行えるので組織としての流動性は高い。プロジェクトの長は課長級の管理職である。本論文では，このプロジェクトを「グループ」と呼んだ。プロジェクトに属する技術者の人数は 10 人程度である。

知識資産の共有，移転にあたっては，同一部内であれば意識せずに行われている。しかし，部をまたいだり，事業部をまたぐ場合には技術面だけでなく，損益も含めた調整が必要になる。

[46] 一般にはプロジェクトは期限が決まっており業務が終わると解散するものである。しかし，A社の場合のプロジェクトという名前を付けているが，は実質的に永続的な組織になっている。

参照文献

Argote, L., & Ingram, P. (2000). Knowledge transfer: A basis for competitive advantage in firms. *Organizational behavior and human decision processes, 82*(1), 150-169.

Basili, V. R., Briand, L. C., & Melo, W. (1996). How reuse influences productivity in object-oriented systems. *Communications of ACM, 39(10)*, 191-116.

Bitzer, J., Scherettl, W., & Shoroeder, P. J. (2007). Intrinsic motivation in open source software development. *Journal of Comparative Economics, 35*, 160-169.

Bock, W.-G., Zmud, W.-R., Kim, G.-Y., & Lee, N.-J. (2005). Behavioral intention formation in knowledge sharing: Examining the roles of extrinsic motivators, social-psychological forces, and organizational climate. *MIS quarterly*, 87-111.

Borgatti, S. P., & Cross, R. (2003). A Relational View of Information Seeking and Learning in Social Networks. *Management Science, 49*(4), 432-445.

Brown, J. S., & Duguid, P. (1996). Organizational learning and communities of practice. *Organizational learning*, 58-82.

Brown, J. S., Collins, A., & Duguid, P. (1989). Situated cognition and the culture of learning. *Educational researcher, 18*(1), 32-42.

Buxton, J. N., & Randell, B. (1970). *Software Engineering Techniques: Report on a Conference Sponsored by the NATO Science Committee.* NATO Science Committee; available from Scientific Affairs Division, NATO,.

Cabrerra, E., & Cabrerra, A. (2005). Fostering knowledge sharing through people management practices. *The International Journal of Human Resource Management, 16-5*, 720-735.

Cohen, W. M., & Levinthal, D. A. (1990). Absorptive Capacity:A New Perspective on Learning and Innovation. *Administrative Science Quarterly, 35*(1), 128-152.

Cusumano, M. A. (1989). The Software Factory: A Historical Intarpretation. *IEEE Software*, 23-30.

CusumanoA.M. (2004). ソフトウェア企業の競争戦略. (サイコム・インターナショナル, 訳) 東京都, 日本: ダイヤモンド社.

Czepiel, J. A. (1975). Patterns of Interorganizational Communications and the Diffusion of a Major Technological Innovation in a Competitive Industrial Community. *The Academy of Management Journal, 18*(1), 6-24.

DavenportT., PrusakL. (2000). ワーキング・ナレッジ―「知」を活かす経営 (Working Knowledge: How Organizations Manage What They Know). (梅本勝博, 訳) 生産性出版.

Davy, C. (2006). Recipients: the key to information transfer. *Knowledge Management Research & Practice, 1,*, 17-25.

DeciE., FlasteR. (1999). 人を伸ばす力 内発と自律のすすめ (Why we do what we do: The dynamics of personal autonomy.). (桜井茂男, 訳) 新曜社.

Dijkstra, E. W. (1970, April). 7.4 Structured programming. In B. Randel, & J. N. Buxton, *Software Engineering Techniques* (pp. 84-88). Birmingham, England: NATO Scientific Committee.

DixonM.N. (2003). ナレッジ・マネジメント 5 つの方法―課題解決のための「知」の共有 (Common Knowledge: How Companies Thrive by Sharing What They Know). (梅本勝博, 末永聡, 遠藤温, 訳) 生産性出版.

Dyer, J. H., & Nobeoka, K. (2000). Creating and Managing a High-Performance Knowledge-Sharing Network : The TOYOTA Case. *Strategic Management Journal, 21*, 345-367.

江波戸哲夫. (2002). 成果主義を超える. 文芸春秋.

藤野真. (2010). W・E・デミングの管理哲学について. 福岡大学商学論叢, 55(2), 189-206.

富士通飛翔編集室. (1999). 池田敏雄 国産コンピュータに賭けた天才の軌跡. 著: 富士通株式会社 (編), オリジナリティを訪ねて III (ページ: 17-28). 富士通経営研修所.

Ghoshal, G., Korine, H., & Szulanski, G. (1994). Interunit Communication in Multinational Corporations. *Management Science, 40*(1), 96-110.

Granovetter, M. S. (1973). The Strength of Weak Ties. *American Journal of Sociology, 78*(6), 1360-1380.

Hall, J., & Sapsed, J. (2005). Influences of knowledge sharing and hoarding in project-based firms. In P. Love, P. Fong, & Z. Irani, *Management of knowledge in project environments* (pp. 57-80). Oxford, UK: Elsevier Butterworth-Heinemann.

Hansen, M. T. (1999). The Search-Transfer Problem: The Role of Weak Ties in Sharing Knowledge across Organization Subunits. *Administrative Science Quarterly, 44*, 92-111.

Hansen, M. T., Mors, M. L., & Lovas, B. (2005). Knowledge Sharing in Organizations: Multiple Networks, Multiple Phases. *Academy of Management Journal, 48*(5), 776-793.

Hansen, M. T., Nohria, N., & Tierney, T. (1999). What's Your Strategy for Managing Knowledgy. *Harvard Business Review*, 1-10.

原田勉. (2000). インタラクション類似性とコミュニケーション頻度. 国民経済雑誌, 181(3), 93-106.

Hatch, N. W., & Dyer, J. H. (2004). Human Capital and Learning as a Source of Sustainable Competitive Advantage. *Strategic Management Journal, 25*, 1155-1178.

Hertel, G., Niedner, S., & Herrmann, S. (2003). Motivation of sofware developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. *Research Policy, 32*, 1159-1177.

Hinds, P. J., Patterson, M., & Pfeffer, J. (2001). Bothered by abstraction: the effect of expertise on knowledge transfer and subsequent novice performance. *Journal of Applied Psychology, 86*(6), 1232-1243.

堀江常稔, 犬塚篤, 井川康夫. (2007). 研究開発組織における知識提供と内発的モチベーション. 経営行動科学, 20(1), 1-12.

Hotta, K. (2009). Effect of Changing Governance System: Result of Western Style Management Adoption to Japanese Culture of Ambiguity. *Journal of Systemics, Cybernetics and Informatics, 7*(1), 66-71.

堀田耕一郎, 杉原太郎, 遠山亮子. (2012). ソフトウェア知識受領における障害と要件の検討. 研究・技術計画学会第 27 回年次学術大会.

JacobsonI., GrissM., JonssonP. (1999). ソフトウェア再利用ガイドブック (Software Reuse - Architecture, Process and Organization for Business Success). (杉本宣男, 吉田幸彦, 落合修, 田中正仁, 訳) 東京: トッパン.

情報処理推進機構. (2012 年 3 月 26 日). ＩＴスキル標準Ｖ３ 2011. 参照日: 2014 年 10 月 19 日, 参照先: ＩＴスキル標準Ｖ３ 2011: http://www.ipa.go.jp/files/000024839.doc

情報処理推進機構 技術本部ソフトウェア・エンジニアリング・センター. (2012). 【改訂版】組込みソフトウェア開発向け品質作り込みガイド. (ソフトウェア・エンジニアリング・センター, 編) 独立行政法人情報処理推進機構 技術本部.

JonesC. (1998). ソフトウェア開発の定量化手法 第２版 (Applied Software Measurement - Assuring Productivity and Quality Second Edition). (鶴保征城, 富野壽, 訳) 構造計画研究所.

JonesC. (1999). ソフトウェア品質のガイドライン (Software Quality - Analysis and Guidelines for Success). (富野壽, 訳) 構造計画研究所.

加護野忠男. (1985). 創造的組織の条件. 組織科学, 19(1), 11-19.

Karlsen, J. T., & Gottschalk, P. (2004). Factors Affecting Knowledge Transfer in IT Projects. *Engineering Management Journal, 16*(1), 3-10.

KirkpatrickD. (2011). フェイスブック 若き天才の野望 (5 億人をつなぐソーシャルネットワークはこう生まれた) (The Facebook Effect: The Inside Story of the Company That Is Connecting the World). (滑川海彦, 高橋信夫, 訳) 日経 BP 社.

黒瀬邦夫. (2005). 富士通の知的「現場」改革. ダイヤモンド社.

Lakhani, K. R., & Wolf, R. G. (2005). Why hackers do what they do: Understanding motivation and effort in free/open source software projects. In J. Feller, B. Fitzgerald, S. Hissam, & K. R. Lakhani, *Perspectives on Free and Open Source Software* (pp. 3-22). MIT Press.

Lakhani, K., & von Hippel, E. (2003). How open source software works: “free” user-to-user assistance. *Research Policy*, 923-943.

Leemhuis, T. (2012, 1 12). *Kernel Log: 15,000,000 lines, 3.0 promoted to long-term kerne*. (Heise Media UK Ltd.) Retrieved 11 6, 2014, from The H Open: http://www.h-online.com/open/features/Kernel-Log-15-000-000-lines-of-code-3-0-promoted-to-long-term-kernel-1408062.html

LeonardD., SwapW. (2005). 「経験知」を伝える技術 (Deep Smarts). (池村千秋, 訳) ランダムハウス講談社.

Lipman-Blumen, J., & Leavitt, H. J. (1999). *Hot Groups：Seeding Them, Feeding Them, and Using Them to Ignite Your Organization.* Oxford University Press.

野中郁次郎, 竹内弘高. (1996). 知識創造企業 (The Knowledge-Creating Company). (梅本勝博, 訳) . 東洋経済新報社.

野中郁次郎, 遠山亮子, 平田透. (2010). 流れを経営する：持続的イノベーション企業の動態理論. 東洋経済新報社.

Nonaka, I., Toyama, R., & Konno, N. (2000). SECI, Ba and Leadership: a Unified Model of Dynamic Knowledge Creation. *Long Range Planning, 33*, 5-34.

Nonaka, I., Toyama, R., & Nagata, A. (2000). A Firm as a Knowledge-creating Entity: A New Perspective on the Theory of the Firm. *Industrial and Corporate Change, 9*(1), 1-20.

Osterloh, M., & Frey, B. S. (2000). Motivation, Knowledge Transfer, and Organizational Forms. *Organization Science, 11*(5), 538-550.

Reagans, R., & McEvily, B. (2003). Network structure and knowledge transfer: The effects of cohesion and range",. *Administrative Science Quarterly, 48*(2), 240-267.

Reagans, R., Argote, L., & Brooks, D. (2005). Individual Experience and Experience Working Together: Predicting Learning Rates from Knowing Who Knows What and Knowing How to Work Together. *Management Science, 51*(6), 869-881.

Ryan, R. M., & Deci, E. L. (2000, January). Self-Determination Theory and the Facilitation of Intrinsic Motivation, Social Development, and Well-Being. *American Psychologist, 55*(1), 68-78.

Smith, A. H., & McKeen, D. J. (2003). *Instilling a knowledge-sharing culture.* Queens University, School of Business. Kingston, Ontario: Queen's Centre for Knowledge-Based Enterprises 20.1.

Southwick, K. (1999). *High Noon - The Inside Story of Scott McNealy and the Rise of Sun Microsystems.* New York, N.Y.: John Wiley & Sons, Inc.

Szulanski, G. (1996). Exploring Internal Stickiness: Impediments to the Transfer of Best Practice within the Firm. *Strategic Management Journal*, 27-43.

Szulanski, G. (2000). The Process of Knowledge Transfer: A Diachronic Analysis of Stickiness. *Organizational Behavior and Human Decisiton Processes, 82*(1), 9-27.

Thompson, M., & Heron, P. (2006). Relational quality and innovative performance in R&D based science and technology firms". *Human Resource Management Journal, 16*(1), 28-47.

TraczW. (2001). ソフトウェア再利用の神話―ソフトウェア再利用の制度化に向けて (Confessions of a Used Program Salesman: Institutionalizing Software Reuse). (畑崎隆雄, 鈴木博之, 林雅弘, 訳) ピアソンエデュケーション.

van den Hooff, B., & de Leeuw van Weenen, F. (2004). Committed to Share: Commitment and CMC Use as Antecedents of Knowledge Sharing. *Knowledge and Process Management*, 13-24.

Wenger, E., McDermott, R., & Snyder, W. (2002). *Cultivating Communities of Practice: A Guide to Managing Knowledge.* Harvard Business Review Press.

Wright, P. M., Dunford, B. B., & Snell, S. A. (2001). Human resources and the resource based view of the firm. *Journal of Management, 27,pp701-721, 27*, 701-721.

Ye, Y., & Kishida, K. (2003). Toward an Understanding of the Motivation of Open Source Software Developers. *The 25th International Conference on Software Engineering*, (pp. 419-429).

米澤明憲. (1984). オブジェクト指向型プログラミングについて. コンピュータソフトウェア, 29-41.

謝辞

本論文は多くの方々のご支援，ご指導を頂いた結果として完成させることができた。

まずは，北陸先端科学技術大学院大学において，主テーマ指導教員として遠山亮子先生には多くのご指導を頂いた。先生が中央大学に転じられたあともゼミ等において引き続きご指導いただけたこと，本論文をまとめる最後の数カ月間での励まし，ご指導は大変な力になった。遠山先生の下で博士の学位を目指す社会人学生が集まって議論を深めることにより，研究への取り組み方も学ぶことができた。博士後期課程入学から 9 年もの期間がたってしまい，ご心配とお手数をおかけしたことで，感謝とともにお詫びを申し上げなくてはならない。

ＪＡＩＳＴの多くの先生方に，個別ゼミ，全体ゼミという機会を通じてご指導いただいた。特に，杉原太郎助教（現岡山大学）には質問票調査およびデータ分析の手法を通じて厳しい指導を頂いた。ご指導がなければ，質問票は完成させることができなかったものと思う。梅本勝博先生には入学当初多くの参考文献を推薦して頂き，大変に参考になった。また，論文をまとめる最後の段階になって急遽主査を務めて頂くことになり，最後の御指導によって形を整えることができた。井川康夫先生，國藤進先生には，論文提出期限を直前に控え，あきらめかけた時に叱咤激励を頂き，最後の力とすることができた。先生方には心より感謝を申し上げる。

また，遠山先生の下に集まった社会人学生の自主研究会の仲間には励ましと助言を頂いた。仲間の研究姿勢を見ることで，低調になった研究もあきらめずになんとか踏みとどまることができた。

自らが属する組織であるがA社の人事部門および上司には大学院への入学にあたって便宜を図って頂き，大いに感謝する。また，ともに議論した社内コミュニティの仲間，質問票に答えてくれた仲間のおかげでこの研究が成立した。

最後に，家族のバックアップがあったからこそ諦めずに論文を纏められたものと思う。妻　千鶴に深く感謝する。入学後 9 年もかかり，ご迷惑とご心配をおかけした皆様には深くお詫びを申し上げる。この成果を社会に還元していければ幸いである。

堀田　耕一郎

本研究に関する発表論文

査読付き原著論文

Hotta, Kohichiro. (2009). Effect of Changing Governance System: Result of Western Style Management Adoption to Japanese Culture of Ambiguity. Journal of Systemics, Cybernetics and Informatics, vol.7-1, 66-71.

査読付き国際会議報告

Hotta, Koh. (2008). Consideration on Adoption of Western Style Management in Japanese Culture of Ambiguity. The 4th International Symposium on Management, Engineering and Informatics: MEI2008, Orlando, FL, USA. (Session's Best Paper 受賞)

口頭発表

堀田耕一郎，杉原太郎，遠山亮子 (2012). 「ソフトウェア知識受領における障害と要件の検討」，研究・技術計画学会第 27 回年次学術大会