ソフトウェア開発プロジェクトにおける
コミュニケーション支援に関する研究

平成７年８月

仲　谷　美　江

ソフトウェア開発プロジェクトにおける
コミュニケーション支援に関する研究

平成７年８月

仲　谷　美　江

目　次

第1章　序論

初めに、本研究の背景となるソフトウェア開発現場における問題点を述べたうえで、本研究の目的と構成を述べる。

1．1　研究の背景

高度情報化社会の到来が言われて久しい。今や社会生活の至る所に計算機やマイコンが普及し、それをコントロールするためのソフトウェア需要は年々増加している。しかし、ソフトウェアの生産効率は需要の伸びほどには向上していないのが現状であり、生産性向上研究は急務となっている。

さらに近年のソフトウェアは大規模かつ複雑になり、生産に従事する開発プロジェクトの規模も大きくなっている。おのずとコミュニケーションや組織管理の問題など、人間的要素が生産性に大きく影響するようになってきた。

一方、コンピュータの普及に伴い、職場でのコミュニケーションをコンピュータで支援する協同作業支援研究も盛んになり始め、ソフトウェア開発プロジェクトにおけるコミュニケーション支援も注目を集めている。

1．2　開発プロジェクトにおけるコミュニケーションの問題

ソフトウェアは生産物が目に見えないため、他の工業生産物と比較して生産管理、品質管理が難しいと言われている。ソフトウエア工学では、生産技術は大きく二つに分けられる。一つはソフトウェアやドキュメントを作るために直接必要な技術、狭義の生産技術である。もう一つは、それらを実行するために必要な資源（人、資金、環境など）を計画し管理する、いわゆる管理技術である。どちらもソフトウェアの表現の難しさを克服し、他の工業製品と同様に扱うことを目指している。しかし、ソフトウェアは思考の産物であり、時間とコストによって生産性を操れるものではない。その上、複数人による協同開発においては、個人の問題とコミュニケーションの問題が相乗効果をなして生産効率を下げてしまう。ここでは、本研究に先立って実際のプロジェクトにインタビューを行ない、プロジェクトの問題点を整理してみた。その結果、ソフトウェア開発では、思想・文化の違い、人間関係などがコミュニケーションを妨

げていることがわかった。そこでは、ソフトウェア協同開発においては、従来のアプローチの他に人間的・社会的な側面からも支援が必要と考えるに至った。

1．3　本研究の目的

前節で述べたようなソフトウェア開発の現状を踏まえて、これまで開発プロジェクトにおけるコミュニケーション支援に関する基礎的な研究を行なってきた。本論文では、コミュニケーションに関する多くの問題点の中から3つのテーマを取り上げている。第一に、ソフトウェアの協同開発でもっとも難しい問題は意図を正確に伝達することである。そこで意図伝達の支援を試みる。第二に、近年ソフトウェアシステムの大規模化が著しく、構成員が数百人にのぼるプロジェクトも珍しくない。そこで大規模プロジェクトに特有のコミュニケーションの問題を取り上げる。最後に、コミュニケーション問題を調査するにあたって問題点を明らかにする適切な評価手法がなかった。そこでコミュニケーションを評価し、問題点を明らかにする手法の開発を試みる。

以上のテーマに対し、本研究では以下の3つのアプローチをとった。

（1）認知科学的アプローチにより、メンバー間の意図伝達を支援する。

ソフトウェアに関するコミュニケーションが難しいのは、ソフトウェア設計者の意図を表現する適切な方法がないためである。人間がどのようにソフトウェアを理解しているかについての考察を基に、ソフトウェアの構造と機能を時系列的に表示して意図を表現する劇場モデルを考案し、このモデルを実現するための枠組みを提供する。

（2）組織論的アプローチにより、大規模プロジェクトにおけるコミュニケーションのコスト軽減を図る。

構成メンバーが百名を越えるような大規模なプロジェクトでは、小規模なプロジェクトに比べコミュニケーションにかかるコストが膨大なものになる。ここではプロジェクト内で問題が発生し解決するまでのコミュニケーションに着目し、そのプロセスをトラブルコミュニケーションモデルとして表現した。このモデルに基づいて障害発生時の情報伝達先を算出し、コミュニケーションを支援する枠組みを提供する。

（3）社会心理学的アプローチにより、質問紙評価手法を用いてプロジェクトにおけるコミュニケーションの問題を診断する方法論を提案する。

コミュニケーション支援研究には、現場の問題点の分析、支援機能の開発、支援機能の適用と評価、というフェーズがあるが、現場の分析、特にコミュニケーションを評価する研究はほとんど行なわれていない。ここでは質問紙評価手法を用いて新たな評価方法を構築する。まずプロジェクトに欠かせないコミュニケーションを分類して各々の役割を考察し、質問紙により各コミュニケーションの達成度を評価してプロジェクトの潜在的な問題点を明らかにする手法を提案する。

１．４　本論文の構成

本論文は、本章を含め７章から構成されている。２章では、ソフトウェア開発作業に関する従来の研究をまとめる。３章では、プロジェクト調査の結果からコミュニケーションの問題を整理する。２章、３章で本研究の基礎となる問題意識とアプローチを提示する。４章、５章、６章では、プロジェクトにおけるコミュニケーションの問題点を一つずつ取り上げ、それぞれに対して前節で述べたアプローチから支援方法を提案している。７章は結論である。以下、各章ごとにその概要を述べる。

第２章では、本研究の位置づけを明らかにするために、ソフトウェア工学と協同作業支援分野での研究をまとめる。ソフトウェア工学では、個人の視点、組織の視点、コミュニケーションの視点から生産性向上研究を整理する。協同作業支援分野では、ソフトウェア開発プロジェクトに関するコミュニケーション支援研究をレビューし、整理する。

第３章では、実働プロジェクトに対する調査の結果をまとめる。プロジェクトにおけるコミュニケーションの問題を協同作業全般という視点とソフトウェア開発に限った視点から整理する。その後、これらの問題に対する従来の支援研究を踏まえながら本研究の位置づけと特徴を明らかにし、本研究のアプローチを述べる。

第４章では、まずプロジェクトにおけるコミュニケーションの問題の一つであるソフトウェアの意図伝達の重要性と難しさについて述べる。次に理解に関する人間の認知メカニズムの知見からソフトウェアの意図伝達に必要な要件を整理し、その要件を満たす表現方法として劇場モデルを提案する。最後に劇場モデルを実現しソフトウェアの意図伝達を支援するシステムの構築とその評価について述べる。

第５章では、まず大規模なプロジェクトにおける障害発生時のコミュニケーション過程について分析し、その問題点を明らかにして支援の方向を探る。次に障害発生時のコミュニケーションプロセスをシミュレートするモデルを作成する。最後にモデル

を実現し障害発生時の情報伝達先をアドバイスするシステムの構築とその評価について述べる。

第６章では、協同作業におけるインフォーマルコミュニケーションに焦点を当て、インフォーマルコミュニケーションの重要性と従来の研究について述べる。インフォーマルコミュニケーション研究の難しさはインフォーマルコミュニケーションが定量的に把握しにくく、評価方法が確立していないため具体的な問題点が見えないことにある。そこでインフォーマルコミュニケーションを評価する枠組みを提案し、質問紙法による評価方法を開発した。測定結果はコミュニケーションの特徴がわかりやすいように顔の表情にして表している。最後に評価手法の適用例と妥当性について述べる。

第７章は、本論文の結論である。これまでの章で得られた結果をまとめるとともに、今後に残された課題について述べる。

第２章　ソフトウェア生産性向上に関する基礎的考察

２．１　緒言

本章では、ソフトウェア開発プロジェクトにおける生産性向上に関する従来の研究を整理する。まずソフトウェア工学の研究を、次に協同作業支援分野におけるプロジェクト内コミュニケーション支援研究を紹介する。

２．２　ソフトウェア工学における従来の研究

ソフトウェア工学の研究はソフトウェアの生産性向上を目的とするものである。しかし、その対象は多岐にわたり、単位時間当たりのプログラミング効率を追及するもの、品質向上を目的とするもの、テストや保守に関するもの、等がある。ここでは「ソフトウェア開発プロジェクトの成果はメンバー個人の成果とメンバー間の相互作用の成果から成り立っている」という視点から、個人効率を上げる研究と協同作業全体としての効率を上げる研究に分けて考える。後者は、さらに個人間のコミュニケーションを支援する研究とプロジェクト管理の研究とに分けて考える。

個人の効率をあげる研究とは、ソフトウェア製作技術すなわち生産技術の研究と言える。個人間のコミュニケーションの研究とは、協同作業に必要な情報の伝達を支援する研究である。プロジェクト管理の研究とは、プロジェクトを継続し、運営するために必要な、見積りや工程管理、人事などに関する研究である。以下、この３つの視点からこれまでの研究を整理する。

２．２．１　生産技術の研究

生産技術の研究は一般にソフトウェアのライフサイクルにそって考えられる。ライフサイクルとは、ユーザのニーズの分析から始まって、設計、コーディング、テスト、保守、廃棄、までのプロセスを言う。ライフサイクル自身に関する研究も多く、ここであげたものとは異なるプロセスを提案しているものもあるが、大規模ソフトウェアの開発は上記のプロセスが一般的であり、これに従って生産技術研究の動向を紹介する。

（１）要求定義

システム開発の第一段階はユーザのニーズを正確に把握することである。漠然としたユーザの要求を具体的に整理し、真の問題点を探る方法としては、ＫＪ法やブレインストーミングが用いられる。複雑な問題を構造化する方法としては、Ｎ２チャート、構造化分析法（ＳＡＤＴ（Structured Analysis and Design Technique）法）、特性要因図、ビューポイント分析などがある[1)]。これらはプログラミング開始以前にすべての要求を明らかにしておくという考え方である。一方、ユーザの要求をもとにまず簡単なプロトタイプシステムを作成し、その機能やインタフェースを改善しながら満足のいくシステムへ仕上げていくという方法もあるが、大規模システムの開発ではあまり用いられない。

（2）システム設計・ソフトウェア設計

システム設計は要求仕様を実現するためのハードウェア、ファームウェア、ソフトウェアの構成を決める段階である。ソフトウェア設計はコーディングへ向けてソフトウェアのモジュール構造とデータ構造を決める段階である。設計方法論はシステムの分析からモジュール化までのプロセスを支援し、その記述様式を提供する。システムには機能的側面、動作的側面、構造的側面、情報的側面があり、どの側面から捉えるかによって設計手法は異なる。それぞれ構造化設計法、実時間システム用構造化設計法、オブジェクト指向設計法、データ指向的手法、などがある。記述様式としては、機能的側面からはデータフロー図、動作的側面からは状態遷移図、メッセージシーケンスチャート、イベント系列図、ペトリネット、構造的側面からは実体関連図、オブジェクト通信図、情報的側面からは実体関連図、データ構造図、がある[2)]。

ソフトウェア設計は、システム設計をうけて要求される機能をいかに実現するかを決定する。具体的には制御方式や処理方式の設計、データ設計、モジュール化設計を行う。プロジェクトで開発する場合にはドキュメンテーションの体系やインタフェース規約を決定するための標準化設計も行う。モジュール化設計には、関連の深い機能をひとまとまりとして抽象化し独立性の高いモジュールに分割する抽象化技法[3)]やデータや制御のフローに合わせてモジュールに分割する方法などがある。

最近では要求分析からソフトウェア設計までのフェーズを同一の考え方で支援する傾向にあり、構造化分析と構造化設計を統合した構造化手法（ＳＡＳＤ（Structured Analysis and Structured Design）法）、オブジェクト指向分析法とオブジェクト指向設計法を統合したオブジェクト指向法などがある。またジャクソンシステ

ム開発法のように、問題の分析から実現段階までを一貫して支援する考え方もある。形式的手法は代数などの数学を基礎にしてソフトウェアを記述する技法で、仕様からシステムの性質やプログラムが体系的に導出できる。いくつかの形式的記述言語が開発されていて、規格化されているものもある。

（3）プログラミング環境とツール

ソフトウェア設計によりシステムはモジュール単位に分割されているので、プログラミング段階ではモジュール内設計とコーディング、デバッグを行なう。プログラミング環境はプログラミング言語と深くかかわっていて、言語の開発と共にエディタやデバッガが開発されてきた。主なものには、関数型言語の Lisp 、論理型言語の Prolog、オブジェクト指向言語の Smalltalk などがある。言語とは関係なく開発されているプログラミング技法としては構造化プログラミング、視覚的プログラミング、データ指向プログラミング、ルール指向プログラミング、などがある[4]。

（4）テスト

ソフトウェアのテストには、モジュール単位で行なう単体テスト、モジュールを組み合わせて一つの機能的なまとまりで行なう組み合わせテスト、ハードウェアも含めてシステム全体として行なう総合テスト、実機の上でユーザの立会で行なう運用テスト、がある。

単体テストでは、プログラムが仕様どおりに書かれているかどうかを確認し、コーディングミスを徹底的に排除する。ここで十分にテストしておくことがあとのテストの効率化につながる。組み合わせテストでは単体モジュール間のインタフェースを確認し、組み合わせた結果がシステム仕様書どおりに機能しているかどうかを確認する。統合テストではソフトウェアの動作のほかに、ハードウェアも含めた性能、安全性、操作性、機密性、などの検査を行なう。運用テストは実際の運用と同じ環境でユーザの立会のもとで行ない、高度なリアルタイム性や安全性を要求されるシステムでは実用化までの試用期間が１年以上に及ぶこともある。

十分にテストできるかどうかはテスト項目の設定とテストデータの作成にかかっており、系統だった項目作成技法が開発されている[5]。大規模なソフトウェアではテスト専門チームが担当する。またソフトウェアの品質管理はプロジェクト管理の一貫としても行なわれる。

（5）再利用

ソフトウェア部品の再利用が可能になれば生産性は大きく向上する。再利用には概念・知識レベルの再利用と部品としての再利用がある。ここでは後者について考える。再利用には部品の検索とその利用という2段階がある[6]。部品の検索では必要な機能仕様を定義し、それをもとにライブラリを検索する。全く同じ仕様の部品がないときには類似した部品を見つける。同じ仕様の部品があればそのまま、なければ類似部品の一部を変更して再利用する。変更のためには部品のプログラムを読み、理解して作りなおさなければならない。現在のところでは再利用が進んでいるとはいえないが、その理由は、部品ライブラリが十分でない、ライブラリの分類法が適切でなく検索が困難、他人の作った部品を理解して書き直すよりも自分で始めから作ったほうが早い、という問題があるからである。今後オブジェクト指向のように部品化に適した設計手法やリエンジニアリング[7]によって古いソフトウェアの再活用支援技法が開発されてくるにつれ、再利用の可能性が広がると考えられる。

以上、開発プロセス毎に個々の主要技術を紹介した。しかし、最近ではソフトウェアのライフサイクルを一貫した方法論で支援しようという考え方が主流になっている。それはシステムの開発が各段階できっちり完結するものではなく、長いライフサイクルでとらえればコーディングから設計へ保守から設計へというフィードバックがかかるものであり、段階毎に個別に考えるべきではないという考え方や、上流工程の結果を正しく下流工程に反映させるためには設計からコーディングまで一貫した方法論で支援する必要があるという認識が出てきたからである。他方ではワークステーションの普及もあいまって、CASE（Computer-Aided Software Engineering）ツールの開発が盛んになってきた[8]。狭義にとらえればCASEは要求分析や設計の段階を支援するツールであるが、広義にCASE環境といえば設計からコーディング、デバッグ、テスト、管理、通信機能までも含めた統合開発環境を意味し、生産性向上のほか、信頼性、保守容易性、再利用性、などの向上を目指す。CASE環境の基盤となる方法論には構造化手法やオブジェクト指向などの他に、AIを応用したものなどもある。

2．2．2　コミュニケーション技術の研究

ソフトウェア工学では、メンバー間のコミュニケーション管理はマネージメントの一分野として取り扱われる場合が多い。研究の対象となるのは、主に情報伝達の

正確さと効率である。特にソフトウェアは抽象的で表現が難しく、客観的に記述し、正確に伝達する技術の開発が中心となっている。この傾向は標準化という視点でまとめられる。その一方で、技術レビューというコミュニケーション方法が確立している。技術レビューとは、プロジェクトの進行途中にメンバーが各自の作業内容を報告し合うことで、ソフトウェアの標準化しきれない部分を補うコミュニケーションと言える。以下に、この二つのコミュニケーション研究の動向について述べる。

（1）標準化

標準化の目的を一言で言えば、記述枠組みを一定にして表現の曖昧さや解釈の個人差を排除することである。標準化の効果には大きく2側面があって、一つは情報が客観的定量的に記述でき、理解容易性、テスト容易性、保守容易性が確保できることである。プロジェクトが大規模化し、メンバー同士の密接なコミュニケーションが困難になっている現在では、情報を正確に伝達する方法が重要になっている。もう一つの効果は、標準化された記述により、ソフトウェアの品質に一定の水準が保たれることである。これは、ソフトウェアの品質向上と同時にソフトウェアの再利用を可能にする。標準化設計により作られたソフトウェアモジュールは、部品として汎用性を持ち、生産効率を向上させる。

標準化には製品の標準化と作業手順の標準化の2側面がある[9)]。製品の標準化とはインタフェースのように製品の規格となるものである。作業手順の標準化はドキュメントに関するものが多く、流れ図やプログラム構成図、ハードウェア構成図における記号の書き方、プログラム内容の文書化など、記述形式についての標準が中心である。他に製品評価の標準などもある。

また標準化には様々なレベルがあって、その影響範囲によって国際標準、国内標準、企業内標準、プロジェクト内標準がある。国際標準機関には設立の経緯が異なるISO（国際標準化機構）、IEC（国際電気標準会議）、ITU（国際通信連合）の3つがあるが、この3者の協議によって情報技術の国際標準設立が進められている。内容はメディア、プロトコル、プログラミング言語などの規格設定から、品質管理、評価手法、操作方法などの基準の設定まで多岐にわたり、それぞれ専門委員会を設けて検討されている。我が国では、JISC（日本工業標準調査会）のもとで情報処理学会やその他の団体により国際標準に準じた標準が制定されている。企業ではこれらの標準に従ってソフトウェアが開発される。また、その他にも企業内、プロジェクト内で作業を効率的に進めるための独自の標準が設定されている。

このように標準化はソフトウェア開発に欠くべからざる技術ではあるが、すべての情報が標準化できるわけではなく、標準化しきれない部分を伝達する技術も必要となっている。

（2）技術レビュー

技術レビューの目的は、プロジェクトのメンバーがお互いの作業方針がずれていないかを確認し、内容を議論して品質の向上を図ることである。技術レビューは単なる進捗の報告ではなく、内容について検討する場である。経験の浅いメンバーに対する教育的効果も大きく、メンバーの相互理解を深めるという効果もある。レビューには公式なものと非公式なものがあって、非公式なものとはバグ取りを相談するとかプログラミングを教えてもらうなど、必要に応じて適宜行なわれる日常のコミュニケーションである。公式レビューは決められたメンバーによって定期的に開催され、議事録も残される。

レビューの方法にはウォークスルー、インスペクション、ラウンドロビン・レビューなどがある[10)]。ウォークスルーはレビューとして一般的な方法で、担当者が作業内容について手順を追って説明し、その後他のメンバーが自由にコメントを述べる。インスペクションは議論のテーマを予め決めておき、その局面だけを全員で検討するものである。例えばプログラムの移植可能性を調べるレビューであれば、その点についてのみ徹底的に議論する。ラウンドロビン・レビューはウォークスルーの変形で、一つの作業内容についてレビューの参加者全員が１ステップずつ交代で説明する。この方法では全員が十分な理解を得られるほか、保守のための理解容易性などを確かめることができ、ある種のバグを見つけるのにも効果がある。この他にも、レビュー専門のレビューチームをつくる場合もある。レビューは開発プロセスのあらゆる作業が対象となりえるが、通常は設計、プログラミング、ドキュメントなどについて行なわれている。日本で盛んなＱＣ活動は作業内容や手順の見直しを小集団で話し合ってすすめていくもので、品質改善、生産効率の向上に効果を上げている。ソフトウェア開発の現場でも導入されており、その形式は一種のレビューと言えよう。

２．２．３　組織管理技術の研究

プロジェクト管理の最終目標は、制限のある資源を用いて期限内に必要十分な品質のシステムを開発することである。この中には技術管理と人間管理の側面がある。

技術管理の側面では、見積・開発計画、工程管理、原価管理、品質管理、保守管理などがあるが[11]、ここでは人間管理について述べる。

初期の生産管理はテーラーリズムに代表されるような作業の合理化効率化のみに注目されていたが、Weinberg[12]やShneiderman[13]の研究により、ソフトウェア開発における人的要素の重要性が認められるようになってきた。人間の研究には個人に関するものと組織に関するものがある。

個人に関する研究には教育や適性診断がある。教育には大学までの教育[14) 15) 16)]と企業内教育がある。大学までは計算機科学の基礎理論やリテラシを教え、企業内では実務型トレーニングが実施される。特にシステム設計者には教科書的な知識だけでなく現場での経験やコミュニケーション能力が必要となり、その育成方法の決め手がないため現状では慢性的なSE不足となっている。

組織に関する研究としては、Baker[17]、Scott[18]、Mantei[19]らがソフトウェアの性質に適したメンバー構成とその生産性を研究している。開発現場でも対象システムや組織の性質に合った管理形態の工夫が行なわれている[20]。またソフトウェアを他の工場生産品と同様に扱い、部品を再利用可能にして生産性と品質を向上させようという生産管理方式は日本式ソフトウェア工場と呼ばれている[21]。

この他にも外注ソフトウェアハウスの管理の問題がある。大規模なシステム開発では一部を外部のソフトウェアハウスに依頼することが多いが、その開発プロセスはブラックボックスになるため品質管理やコミュニケーション管理が困難になっている。対策としてはドキュメントのチェックや受け入れ検査が行なわれるほか、ソフトウェアハウスの自発的な意識向上をねらった品質管理技法が開発されている[22]。しかし、今のところこの問題に関して体系だった研究は少ない。

以上、ソフトウェア工学の概要を紹介した。次に協同作業支援研究について紹介する。

2．3　協同作業支援研究

近年、ＣＳＣＷ（Computer-Supported Cooperative Work）と呼ばれる協同作業支援研究が盛んになっている[23]。これは協同作業におけるコミュニケーションをコンピュータなどの機器を用いて支援しようというもので、支援の形態は一般に表2.1のように分類される。ＣＳＣＷが対象とする協同作業はグループ意思決定、ス

ケジュール管理、協同執筆、教育、アイデア生成など広範にわたるが、ここではソフトウェア協同開発支援におけるコミュニケーション支援研究を整理する。支援研究には協同作業現場のコミュニケーションを調査・分析する研究、協同作業をモデル化し支援方法を提案する概念レベルの研究、支援機能を実現するための技術的な研究、などがある[24]。

（1）調査、概念レベルの研究

ソフトウェア開発プロジェクトにおけるコミュニケーションの調査研究には、分散開発環境におけるコミュニケーションの様子を調査したもの[25)~28)]や、協同作業のコミュニケーションプロセスをモデル化したもの[29)~34)]がある。いずれも分散開発におけるコミュニケーションの重要性を指摘し、コミュニケーションの不良がソフトウェア品質の低下の原因となると述べている。また、垂水[25)]や古宮[33)]は調査に基づいてソフトウェア協同開発支援へ向けての課題をまとめている。

（2）支援技術の開発

ソフトウェア協同開発作業をモデル化し支援システムを構築した研究では、pot理論[35)]とgIBIS[36)]が著名である。pot理論は、ソフトウァア協同開発プロセスをミーティングの連続として捉え、ミーティング間の情報の流れを構造化してミーティングの効率を支援するもので、パーソナルコンピュータ上の共有スクリーンで実現している。potとは情報を蓄積し整理するための容器を意味する。

表2．1　　コミュニケーション支援形態の分類

空間／時間	対面型	分散型
同期型	電子会議システムなど	遠隔電子会議システムなど
非同期型		電子メイルなど

gIBIS は graphical Issue Based Information System の略で、IBIS モデルに基づいてソフトウェア設計の上流工程の討論を支援するハイパーテキストシステムである。IBIS モデルとは、設計に関する討論が論点（Issue）、立場（Position）、論拠（Argument）とこれらの関係（一般化／特殊化、置換／疑問、疑問定期、対応、賛成、反対）から成るとして、これらの要素を用いて討論の内容を構造化するものである。また垂水[37]は gIBIS を拡張したシステムを開発し、佐藤[38]はソフトウェアの説明をするときに部分と全体の関係を「絵」で提示するツールを開発している。

他に、渡部[39]らはマルチメディアを用いた会議システム MERMAID（Multimedia Environment for Remote Multiple Attendee Interactive Decision-Making）をソフトウェアの分散開発環境支援に応用している。同様に分散開発のコミュニケーションを支援する研究に貫井[40]、小泉[41]などがある。落水らが開発したソフトウェア開発環境 Vela（Versatile Environment based on Logic programming and AI techniques）[42]は、メンバー間のコミュニケーションを支援するだけでなく、プログラミング熟練者のノウハウを蓄積し再利用することを目指した統合 CASE ツールになっている。

２．４　従来の研究の傾向と問題点

以上、ソフトウェア工学の概要とＣＳＣＷにおけるソフトウェア開発プロジェクト支援研究を整理した。

これまでのソフトウェア工学では、コミュニケーション研究は客観性を重視した標準化が主流であり、そこで伝えきれないものは技術レビューや管理者のコミュニケーション能力に任されていた。しかし、いぜんコミュニケーション不良に起因するシステムのバグがなくならないこと、コミュニケーション能力のある管理者が不足していること、プロジェクトが大規模になり分散開発がすすみ、これまでの方法だけではコミュニケーション管理が困難になってきたこと、などのため計算機を用いてもっと積極的にコミュニケーションを支援しようという動きが出始めている。が、今のところソフトウェア工学の分野で体系だったコミュニケーション支援研究は行なわれておらず、CASE による統合的支援環境が開発されつつある他にはコミュニケーションの実態調査やＣＳＣＷツールの試験的導入などがあるだけである。

一方、プロジェクト内のコミュニケーション支援を目的とするＣＳＣＷでは、ソフトウェア開発プロジェクトもその応用範囲の一つである。これまでにもいくつかの研究が見られるが、コミュニケーションの調査から支援システムの開発まで一貫

して行なっている研究は少なく、その対象も少人数のプロジェクトに限られている。現在では分散開発環境に対してマルチメディア通信による支援を行なう技術的な開発研究が主流である。また、ＣＳＣＷにはまだ社会・人文科学系の研究者が少なく、現場でのコミュニケーションの問題点を分析・評価する手法も研究されていない。そのため、コミュニケーションと生産性との関連を具体的に考察したり、支援ツールの効果を評価した研究も少ない。

このようにソフトウェア開発プロジェクトにおけるコミュニケーション支援研究はまだ日が浅く、以下のような課題が残されている。

（1）現場のコミュニケーション調査に基づく問題点の整理

まず問題点を把握しなければ適切な支援機能は提案できない。現段階では、まだソフトウェア開発作業に固有なコミュニケーションの問題が掘り下げられていない。また実験的なプロジェクトや研究所など限られたケースだけではなく、大規模なプロジェクトなど実際の開発現場についての調査も必要である。

（2）人間の相互作用やコミュニケーションメカニズムについての基礎的な考察

協同作業はメンバー同士の社会的行為であり、社会心理学や文化人類学などの観点から相互作用プロセスの分析が必要である。またコミュニケーションについての人間の認知メカニズムを考察し、人間にとってわかりやすくソフトウェアを伝達するためにはどのような支援機能が適切かを検討しなければならない。

（3）コミュニケーションの評価手法の確立

プロジェクトのコミュニケーション支援は最終的には生産性の向上を目指したものであり、そのためには現状の評価と支援システム導入後の評価は不可欠である。しかし、コミュニケーションと生産性の関係が明確になっていないこと、コミュニケーションの定量的な測定が困難なこと、などのためコミュニケーション評価は非常に難しい。評価方法を確立するためにはこれらの問題を解決する必要がある。

本論文では以上の点を踏まえ、コミュニケーションの問題点の分析から支援方法の提案・開発まで一貫したコミュニケーション支援を目指す。

２．５　結言

本章では、ソフトウェア生産性向上に関する従来の研究を整理した後、今後に残された課題について述べた。

ソフトウェア工学は生産技術と管理技術の研究に大別できる。生産技術の研究は主に手法やツールの開発である。近年では上流工程から下流工程までを一貫してコンピュータで支援する傾向がある。管理技術の研究では特に人間管理の側面を紹介した。この方面では教育やメンバー構成に関する研究がある。ここではさらにメンバー間のコミュニケーション支援についてにも焦点を当てた。ソフトウェア開発における情報伝達では正確さを第一義とするため、標準化記述の推進が主流である。標準化できない部分の伝達や内容の検討は技術レビューで補われている。

一方、協同作業支援技術をソフトウェア開発プロジェクトに応用した研究もある。これらの研究には、ソフトウェア開発プロセスをモデル化して支援機能を提案しているものと、マルチメディアのコミュニケーションツールをソフトウェア開発に応用したものとがある。しかし研究の歴史は浅く、支援されるべき問題は多く残されている。

今後に残された研究領域としては、開発現場における問題点の分析、認知科学的観点からの人間のコミュニケーションメカニズムの考察、コミュニケーション評価手法の確立、などがある。

《参考文献》

1) 藤野喜一・花田収悦：ソフトウェア生産技術、電子通信学会（1985）
2) 佐伯元司・郭文音：サーベイ　ソフトウェア仕様化・設計方の最近の動向、情報処理学会研究報告93 - SE - 93、pp.25-34（1993）
3) Parnas, D. L. : On the criteria to be used in decomposing systems into modules, CACM, Vol.15, No.2, pp.1053-1058（1972）
4) 特集：新しいプログラミング環境、情報処理、Vol.30、No.4（1989）
5) 保田勝通：テスト・品質保証技術の現状と課題、情報処理、Vol.28、No.9、pp.873-886（1987）
6) Prieto-Diaz, R. and Freeman, P. : Classifying Software for Reusability , IEEE Software, Vol.4, No.1, pp. 6-16 (1988)
7) 津田道夫：ＣＡＰＳＤＦにおけるリエンジニアリング、情報処理学会研究報告、93 - SE - 93、pp.165-172（1993）
8) 特集：ＣＡＳＥ環境、情報処理、Vol.31、No.8（1990）
9) 特集：ソフトウェア工学における標準化動向、情報処理、Vol.28、No.9（1987）

10) Freedman, D. P. and Weinberg, G. M. : Handbook of Walkthroughs, Inspections, and Technical Reviews, Little,Brown and Company (1982) (岡田正志監訳：ソフトウェア技術レビューハンドブック、ＴＢＳ出版（1987））
11) 特集：ソストウェアマネージメント、情報処理、Vol.33、 No.8（1992）
12) Weinberg, G. M. : The Psychology of Computer Programming, Van Nostrand Reinhold (1991)
13) Shneiderman, B. : Software Psychology, Winthrop (1980)
14) 宇都宮孝一・吉田和幸・藤田米春・遠藤勉・二村祥一・越智義道：専門情報処理教育のための新しい教育環境、情報処理学会第39回全国大会、pp.22-23（1989）
15) 浮貝雅裕・菅原研次・三井田惇郎：情報系学科新入生に対する導入教育とそのための演習教育環境、情報処理学会論文誌、Vol.33、No.1、pp.1-11（1992）
16) 加藤木和夫・白石久敬：プログラミング教育とプログラム理解のモデル化、情報処理学会第47回全国大会、 pp.1-31（1993）
17) Baker, F. T. : Chief Programmer Team Management of Production Programming, IBM Syst. J. 1 (1972)
18) Scott, R.F. : Predicting Programming Group Productivity --- A Communications Model, IEEE Trans. Softw. Eng., Vol.1, No.4, pp.411-414 (1975)
19) Mantei, M. : The Effect of Programming Team Structures on Programming Tasks, Comm.ACM, Vol.24, No.3, pp.106-113 (1981)
20) 特集：システム大規模化で重要性高まるプロジェクト管理、日経コンピュータ、1.15号、pp.72-107（1990）
21) Cusumano. M.A. : Japan's Software Factories: A Challenge to U.S. Management, New York, Oxford University Press (1991)
22) 菅原護・中村陽生：外注ソフトの品質向上活動『あゆみ』、ＥＮＧＩＮＥＥＲＳ、No.496、pp.6-12（1990）
23) 阪田史郎：グループウェアの実現技術、ソフトリサーチセンター（1992）
24) CSCW '86 : Proceedings of the Conference on Computer-Supported Cooperative Work, ACM SIGSOFT, SIGCHI and SIGOIS, Austin, TX, Dec.3-5（1986）
CSCW '88 : Proceedings of the Conference on Computer-Supported Cooperative Work, ACM SIGCHI and SIGOIS, Portland, OR,

Sept.26-28 （1988）

CSCW '90 : Proceedings of the Conference on Computer-Supported Cooperative Work, ACM SIGCHI and SIGOIS, Los Angeles, CA, Oct.7-10 （1990）

CSCW '92: Proceedings of the Conference on Computer-Supported Cooperative Work, ACM SIGCHI and SIGOIS, Tront, Canada, Oct.31-Nov.4 （1992）

25）垂水浩幸：グループウェアのソフトウェア開発への応用、情報処理Vol.33、No.1、pp.22-31（1992）

26）Bendifallah, S. and Scacchi, W. : Work Structures and Shifts: An Empirical Analysis of Software Specification Teamwaork, 11th International Conference on Software Engineering, Pittsburgh, pp.260-270（1989）

27）井上敦子・福田由紀雄：ソフトウェア開発におけるコミュニケーション不良と品質の関係、情報処理学会第38回全国大会、pp.1126-1127（1989）

28）福田由紀雄・井上敦子・津田淳一郎：遠隔地ソフトウェア開発の実験、情報処理学会研究報告、90 - SE - 71、pp.49-56（1990）

29）貫井春美・栗原美佐・三原幸博：グループウェア支援機能の実験的考察、情報処理学会研究報告、90 - SE - 71、pp.57-64（1990）

30）海谷治彦・佐伯元司：ソフトウェアの仕様化過程における協調作業のモデル化、情報処理学会研究報告 91 - HI - 37 (2)（1991）

31）垂水浩幸：ソフトウェア設計用グループウェアに関する一考察、情報処理学会研究報告、91 - SE - 80、pp.159-166（1991）

32）岡田世志彦・飯田元・井上克郎・鳥居宏次・永岡渡・梅本肇・酒井睦雄：ソフトウェアの分散開発のモデル化の試み、情報処理学会研究報告、91 - SE - 82、pp.1-8（1991）

33）古宮誠一：ソフトウェア協調型設計過程のモデル化と知的支援方法について、情報処理学会研究報告、92 - SE - 83、pp.113-120（1992）

34）西正博・海谷治彦・佐伯元司：ソフトウェアの発注者－開発者会議におけるインタラクションの分析、情報処理学会研究報告、92 - HI - 41、pp.17-24（1992）

35）Begeman, M., Cook, P., Ellis, C., Graf, M., Rein, G., and Smith, T. : Project NICK: Meetings Augmentation and Analysis, Proceedings of CSCW'86, pp. 1-6 (1986)

36) Conklin, J. and Begeman, M. L. : gIBIS : A Hypertext Tool for Exploratory Policy Discussion , CSCW'88, pp.102-114 (1988)

37) 垂水浩幸・吉府研治：めろん：仕様に関する議論のフェーズを意識したソフトウェア仕様議論支援システム、日本ソフトウェア科学会第９回大会、pp.57-60 (1992)

38) 佐藤洋一・小池英樹・広瀬通孝・石井威望：Multidimensional Visualization Tool の開発、情報処理学会研究報告、90 - HI - 30 (2) (1990)

39) 渡部和雄・阪田史郎・前野和俊・福岡秀幸・大森豊子：マルチメディア分散在席会議システムＭＥＲＭＡＩＤ、情報処理、Vol.32、No.9、pp.1200-1209 (1991)

40) 貫井春美・松尾朗・太田哲夫：ソフトウェア分散開発支援システムＤ^2、機械学会　設計工学・システム部門講演会論文集、No.920-103、pp.366-371 (1992)

41) 小泉寿男・鈴木昌則・白鳥則郎：リモート協調作業によるソフトウェア開発知的環境、情報処理学会研究報告、 93 - DPS - 63、pp.63-70 (1993)

42) 落水浩一郎・山口高平：ソフトウェア開発における協調支援環境 Vela 、情報処理学会第41回全国大会、 pp.149-150 (1990)

第３章　ソフトウェア生産現場におけるコミュニケーションの問題

３．１　緒言

本章では、前半で２つのソフトウェア開発プロジェクトに実施した調査方法と結果の概要について述べる。調査の目的は協同開発現場でのコミュニケーションの現状と問題点を把握し、支援の方向を決めることである。後半では、コミュニケーション支援に対する本研究のアプローチと、従来研究の中での本研究の位置づけについて述べる。

３．２　第１回目の調査の概要

第１回目の調査の目的は、プロジェクトの中で意図の伝達はどのように行なわれているのかという実状を把握することである。開発するシステムの構造や動作イメージ、各モジュールの関係などがドキュメントやレビューのような伝達方法で十分に伝達されているのか。また、システムエンジニアからプログラマへシステムの説明を行なうとき、プログラマ同士で情報を交換するときなど、どのような経路、方法でコミュニケーションしているのか、について調べる。

３．２．１　調査方法

（１）プロジェクトの構成と内容

メンバー構成を図3.1に示す。開発チーム７人、テストチーム４人から構成される。開発チームはマネージャ、システムエンジニア（以下ＳＥと略）２人、プログラマ４人の階層構造をとる。開発するシステムは中規模のアプリケーションで、マネージャと開発チームのＳＥの３人でシステム設計をし、ＳＥの１人がソフトウェア設計を行なった。当初の開発期間の予定は１年で、２ヵ月延長して完成した。

（２）実施方法

開発チームのＳＥ２人、プログラマ２人に対して各２時間、開発チームのプログラマ１人とテストチームのＳＥ１人に対して３０分、計６名に個人面接でインタビューを行なった。実施時期はコーディングを開始した段階からプロジェクトの終了まで

の期間である。インタビューの他に週１回の進捗報告会に計５回出席し、コミュニケーションの様子を観察した。

（３）インタビューの内容

以下のテーマを中心に、日頃問題と感じていることなどを自由に発話してもらった。

①プロジェクトについて（構成、工程、開発するシステムの内容）

②自分の担当部分について（担当モジュールの内容、作業のプロセス）

③他のメンバーとのコミュニケーションの様子

①と②は、各メンバーが自分の担当する内容と全体との関係をどのように理解しているかを知るために質問した。③は打ち合わせなどの公的なコミュニケーションだけでなく、日常の作業中に誰とどのような会話をしているかを詳しく話してもらった。このテーマの他にも、メンバーがソフトウェア開発作業について感じていることを自由に発話してもらった。

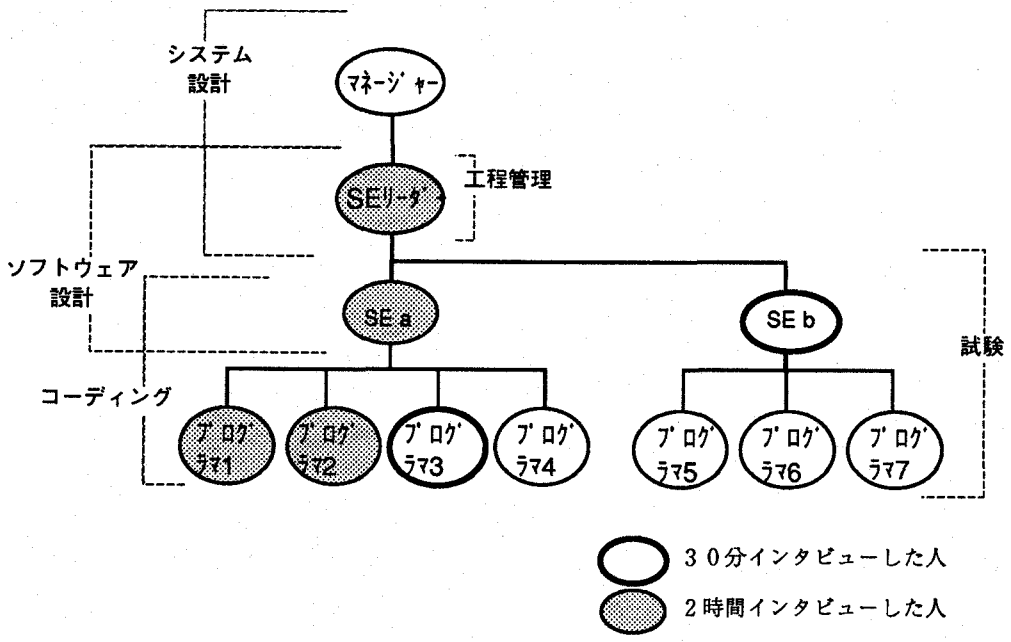


図３．１　　第１回調査におけるプロジェクトのメンバー構成

３．２．２　第１回目の調査結果

このプロジェクトは規模が小さく、定期的ミーティングも全員の出席で行なわれていた。つまり、全員がプロジェクトの進捗やシステム仕様について同じように把握でき、メンバーの意図が統一しやすいと予測される規模である。実際にメンバー間のコミュニケーションの量は少なくなかった。しかし、インタビューの結果からは各メンバーの意図のズレが浮き彫りになった。

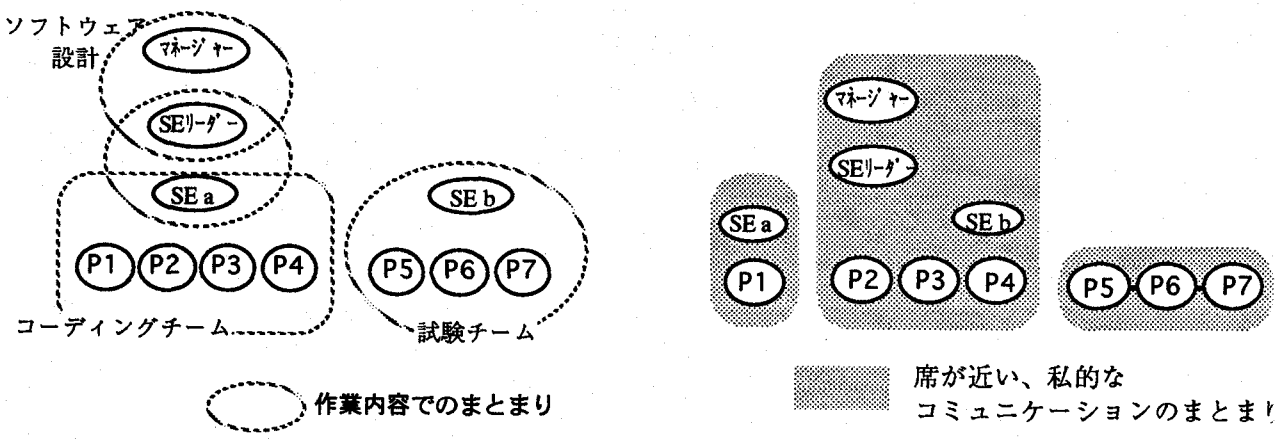
まず、各メンバーによって抱いているシステムのイメージが異なっていた。マネージャは、システムの最終イメージや工程の進み具合を他のプロジェクトとの関係で捉えている。システム設計に関わったＳＥは、システムの目的や意義について既存のアプリケーションと比較しながら具体的に説明してくれた。一方プログラマはシステム開発の経験が浅く、自分の担当部分の機能が説明できるだけでシステム全体の動作についてのイメージは漠然としており、理解のレベルに個人差が大きい。全体と自分との関連を把握していないことも多い（付録・プロトコル１、８、９　参照）。あるプログラマは、始めは全体がよくわからないまま小さい部分から与えられた仕様どおりに作っていたが次第に全体との関わりが見えてくるようになったと、手探りでプログラミングしている様子を述べている。

さらに、メンバーの所属によってソフトウェア開発の方針が異なっていた。開発チームのうちＳＥ１人とプログラマ１人は外部のソフトウェアハウスから派遣されていたが、開発方針について他のメンバーと考え方が異なり、何を重視し何を問題と見るかの視点が異なるために会議などでも意見が合わないことが多かった。このＳＥは、ソフトウェア技術者としての持論を持っていて、ソフトウェアの品質や開発方法にこだわりを持っている。特にソフトウェア開発管理（進捗管理、教育、評価の問題）に関してマネージャと考え方が食い違っていた（付録・プロトコル２、６、７、１０　参照）。

インタビューに応えてくれたメンバーらは、異口同音にシステムのイメージを伝えることの難しさを語った。システムのイメージを伝達しにくい原因の一つにソフトウェアの表現の難しさがあげられる。システムを設計したＳＥでさえイメージをうまく言い表わせないと述べた。未熟練者にとって仕様書の簡単な説明を聞いただけでシステムの具体的なイメージを描くことは難しく、ましてやＳＥの意図を汲んでプログラミングできるほど深くシステムを理解することはできない。インタビュー

でも「イメージがわかない」「どうやって言えばいいのかわからないが．．．」など表現の難しさに関する発話が多くみられた。

表現の難しさは進捗管理や評価などソフトウェア管理の問題にも関わってくる。現状では進捗と成果をドキュメントやプログラミングの量で判断しているため、管理される側からは正しく評価されていないという不満が残り、管理する側には正確な進捗が見えないという問題が残っている（付録・プロトコル２　参照）。また、仕様の説明や進捗報告は定期ミーティングや公式レビューで行なわれていたが、ほとんどの議題は一方的な報告に終わりがちで、十分な相互理解が得られる場とは言えない（付録１、プロトコル１、９　参照）。公式のコミュニケーションで伝達されなかった部分は日常会話で補なわれていた。作業を進めるにあたっての具体的な疑問、確認、相談はミーティングの議題に上ることはなく、身近にいる人に相談して処理される（付録１、プロトコル３、４、５　参照）。そのため、ソフトウェア開発の基本的な考え方やノウハウは頻繁に日常会話をするメンバーの影響を強く受け、作業場所が離れているメンバーとは考え方にズレが生じていた。図 3.2 (a) はメンバーを公式のコミュニケーションをするまとまりでくくったもので、図 3.2 (b) はメンバーを日常会話をするまとまりでくくったものである。図 3.2 (a) と (b) を比べると明らかなように、日常の私的なコミュニケーションは公的なまとまりとは関



（a）公式のコミュニケーションをするまとまり

（b）私的なコミュニケーションをするまとまり

図３．２　プロジェクト内のコミュニケーションのまとまり

係なく行なわれていた。日常会話の内容は様々で、システムについての質問・ソフトウェア全般に対する考え方・プログラミング技術の指導、作業方法の模倣など、意図の伝達のほかに教育的な要素もあった。経験の浅いプログラマは私的なコミュニケーションをする相手の考え方に影響される傾向が見られた。プログラマ１はＳＥａと同じ考え方をし、プログラマ２はＳＥリーダーと同じ考え方をしていた。ＳＥａとＳＥリーダーの考え方は異なっているため、プログラマ１とプログラマ２も異なる考え方を持っていた。

３．３　第２回目の調査の概要

第２回目の調査の目的は大規模なプロジェクトに特有な問題を探ることである。第１回目の調査では、ソフトウェアの意図伝達の実態と問題点が明らかになった。２回目は近年の開発プロジェクトが大規模化している傾向を反映して、メンバーが１００人を越える規模のプロジェクトにおけるコミュニケーションの問題を調べる。インタビューできる人数と時間が限られていたため効率的に問題を聞き出さなければならない。そこでインタビュー内容の焦点を絞るための事前調査としてアンケートを実施し、その結果を踏まえてインタビューを実施した。

３．３．１　調査方法

（１）プロジェクトの構成と内容

メンバー構成はマネージャ、システムエンジニア（以下ＳＥ）、パッケージ製作チームの３階層をとる（図3.3）。ＳＥはいくつかのチームを受け持ち、パッケージ開発を管理する。パッケージ製作チームは外部のソフトウェアハウスから成り、各チームともＳＥと数人のプログラマで構成されている。人数は全体で約２００人でプロジェクトの工程によって増減した。開発するシステムは開発期間が約３年で、客先とメーカとの共同研究という形でシステム設計とデータ設計を行ない、この段階で２年近くかけている。以下の２つの調査はコーディング段階で実施した。

（２）アンケートの実施と結果

インタビューに先だって、プロジェクト内の５０人にアンケート調査を実施した（質問紙は附録１参照）。配付するメンバーはマネージャに一任し、ベテランから

新人まで幅広く回答してもらえるように依頼した。アンケートでは、プロジェクト内での自分の役割、大規模ソフトウェア開発における問題、コミュニケーションについて質問した。

回答者は５０名で、そのうちソフトウェア設計を行なっていると回答したものが３３名（経験年数２０年のマネージャから、１年目のプログラマもいた）、与えられた仕様をプログラムするだけと回答したもの１７名（経験年数５年から５ヶ月まで）である。

アンケートでは、主に以下のような問題が挙げられた。

・仕様の決定遅れや変更が多い
・パッケージ間のインタフェースのずれ
・環境の不備
・進捗が把握できない
・設計、見積の不備
・一人当たりの負荷が大きい
・プログラムの質の低下（教育が不足）
・問題意識の欠如
・メンバーの変更がある
・上下、左右のコミュニケーションの不良

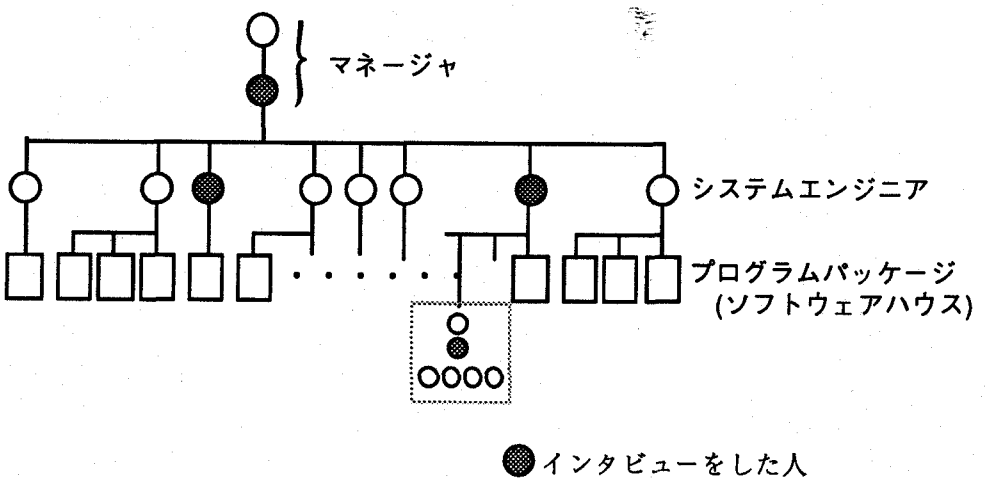


図３．３　　第２回調査におけるプロジェクトのメンバー構成

・感情的な問題（人間関係の問題）
・ドキュメント化が困難

プロジェクトにおける問題は様々な要因が絡んでいるので単純に分類してしまうことは難しいが、ここでは敢て次の３つの視点から上の問題を捉えてみた。

①物、資源の不足に関する問題

・環境の不備：これはコンパイラやデバッガなどが充実していなかったり、試験のための実機が不足しているなどの問題がある。
・人員の不足：他のプロジェクトも並行して担当していたり、経験や能力のあるメンバーに仕事がかたよったりして、一人当たりの負荷が重すぎる。
・工程の不備：設計の不備や見積の失敗、客先との関係などによって、もともと無理のある工程になる場合がある。
・仕様の不備：仕様の細部が未決定のままで作業を進めざるを得なかったり、仕様の変更が多いなどの問題がある。

②情報の流れに関する問題

・進捗が見えない：ソフトウェアは思考の産物であり、進捗を表現しにくい。管理する側もされる側も問題を感じる。
・情報が伝わらない：大規模プロジェクトになると、情報が全体に正確に速やかに伝わりにくい。仕様の変更が伝わっていなかったり、パッケージ間の横の連絡がとりにくかったりする。また全体の情報が製作担当者まで行き渡らないため、担当者からは全体が見えないとの不満が出て、反対に管理者からは担当者が全体を見通した開発をしないなどの問題が指摘されている。

③人的要因に関する問題

・問題意識や意図が統一できない：問題意識の欠如、技術レベルの差、価値観の違い、経験の違い、などによりソフトウェア開発に対する意図や方針が異なってくる。
・感情的な問題：人を管理することの難しさ、命令する人と実作業者とのギャップなどの問題がある。また、モチベーションやモラルの低下も問題である。

これらの問題は相互に影響し合って問題をさらに難しくする。例えば、情報が伝わりにくいという問題は、メンバーのモチベーションや問題意識にも大きく関わってくると考えられる。また、情報が伝わりにくい所へ仕様の変更が多いという問題が重なれば、仕様を間違えたまま開発を進めるという致命的なトラブルも起こりうる。プロジェクトの人数が多くなると、調整・管理・連絡等にかかるコストが増大し、作業のほとんどがコミュニケーションに費やされるといっても過言ではない。

このようなアンケート結果を踏まえてインタビューを行なった。

（3）インタビューの実施と内容

マネージャ1人、ＳＥ2人、パッケージ製作を担当するソフトウェアハウス要員1人の計4人に対して各1～2時間、個人面接でインタビューを行なった。インタビューの他に毎日開催されるシステムエンジニアの連絡会議に1回出席し、コミュニケーションの様子を観察した。

インタビューの内容は、次のテーマを中心に日頃問題と感じていることなどを自由に発話してもらった。

①現在のプロジェクトにおける自分の役割。仕事の内容と進め方について。
②現在のプロジェクトの問題点について。
③現在のプロジェクトではどのようなコミュニケーションが行なわれているかについて。

これらの質問は、大規模なプロジェクトの中で各自の立場によって仕事や問題点をどのように捉えているか、コミュニケーションは階層によってどのように異なるか、を比較することを目的としている。このテーマの他にも、メンバーがソフトウェア開発作業について感じていることを自由に発話してもらった。

3．3．2　第2回目の調査結果

このプロジェクトは人数が多いため定期的ミーティングはＳＥとマネージャで行なわれていた。パッケージを製作するソフトウェアハウスとの連絡は担当のＳＥを介して行なわれる。マネージャは資源管理と全体の進捗管理、ＳＥはパッケージ間のコーディネイトと進捗管理が仕事である。

この規模のシステムになると全体を把握しているメンバーは数人であった（インタビューした中では1人）。ＳＥでも全体を理解できる者は少なく、自分との関連

部分だけを押さえていた。パッケージ担当者は、熟練者でも自分の担当範囲だけしか見通せない状態であった。第１回目に調査したプロジェクトでは全メンバーがシステム全体を見渡す機会があったが、今回のプロジェクトでは各自がシステムの一部だけを見ていると言える。

第一回目の調査ではコミュニケーションしても意図の伝達が難しいことが問題になっていたのに対し、今回のプロジェクトではコミュニケーション自体が不足していることが問題になっていた。各自が各々の立場からコミュニケーションの重要性と現状の問題点について述べた。マネージャは、コミュニケーションにはプロジェクトの統一と問題の早期発見という役割があると考えており、頻繁に作業現場を訪ねてはプロジェクトの雰囲気作りと現場での問題を探ることに努めていた。連絡会議や進捗報告では本当に知りたい情報は得られないと話している。ＳＥは、担当者間の調整コミュニケーションの重要性と難しさについて述べた。パッケージに分割して開発する場合、パッケージ単体の製作には問題がなくてもパッケージの組み合わせ段階で問題が生じることが多く、パッケージ担当者間のインタフェース管理が重要である。そのためには十分な打ち合わせが必要だが、ＳＥ同士の話は抽象論になりやすく、お互いに納得したつもりでも具体的なレベルではズレがあったりするので、頻繁なコミュニケーションをもち、じっくり話し合うことが一番だと述べている。また全体を把握しているＳＥが少ないため、特定のＳＥに問題の処理が集中して負担が大きくなることも重要な問題になっている。パッケージ担当者は全体の情報が下りてこないことの不満を述べた。またＳＥの話と同様に、パッケージ間のコミュニケーションが不足しているため互いが了解していたつもりでも結果がズレていることがある、連絡も徹底していないなどの問題点を指摘した。

３．４　プロジェクトにおけるコミュニケーションの問題――――まとめ

２つの調査結果から、プロジェクト内のコミュニケーションについての問題を整理する。

現場でのプロジェクトの問題には多くの要素が絡んでいて、一つ一つの要素を単独で考えることは難しい。プロジェクトのメンバー構成、開発するシステムの内容、作業環境、予算や納期、客先との関係などの他にメンバー間の人間関係も関わってくる。インタビューから、ソフトウェア自身の曖昧さが立場や思想の違いからくる

コミュニケーションギャップを助長していることが浮き彫りになってきた。そこで、ここでは問題を協同作業の難しさとソフトウェア開発特有の難しさの２側面に分けて考える。協同作業の難しさとは、ソフトウェアに限らず協同作業全般にともなう人間関係の問題である。ソフトウェア開発特有の難しさとは、ソフトウェアが抽象的で表現しにくいことと、完全に独立した部品に分割しにくく部品間の調整が多いことから生じる問題である。この２大要因が影響し合ってプロジェクトのコミュニケーションが難しくなっていると言える。以下ではこの２つの側面から問題をまとめる。

（１）協同作業の側面から見た問題

複数の人間が一つの課題を遂行するとき、様々な問題が生じる。以下であげる問題は調査結果から得られたもので、協同作業における問題のすべてというわけではない。

①メンバーの立場の違いによる難しさ

一般にプロジェクトは管理者層と開発担当者層から成る階層構造をとる。大規模なものになると中間管理者が存在し、３層から４層の構造になる。管理者はプロジェクト外部との折衝や資源の管理を行ない、担当者は実際の開発作業を行なう。中間管理者は管理者と担当者との媒介として、進捗管理や技術的な問題の解決、担当者間の調整を行なう。それぞれの立場によって利害関係や目標が異なるため、全員の意図を統一することは難しい。例えば管理者がスケジュールと予算を念頭において作業工程を決め、一方では担当者が納期よりも品質を優先させようとすれば作業方針に擦れ違いができる。特に大規模なプロジェクトは異なる部署や別会社から派遣されたメンバーで構成される場合が多く、このような価値観の違いによるギャップが生じることが多い。出身の組織によって考え方や利害関係が異なりコミュニケーションが難しくなる。

②メンバーの個人差による難しさ

メンバーの個人差は立場の差よりも大きい。能力差やどのような教育を受けてきたかという差の他に、適性という問題もある。また、これまでにどのような経験を積んできたか、その対象分野や所属プロジェクトの性質がメンバーの思想に大きく

影響する。異なる経験を積んで来たメンバーとの協同作業は、視野を広げるという意味では効果的だがメンバー間の相互理解や親近感の形成には時間がかかる。

③公的コミュニケーションと私的コミュニケーションの存在

プロジェクトには公的なコミュニケーションと私的なコミュニケーションがある。公的なコミュニケーションとはフォーマットが決められているものや記録に残るもので、業務手続き上の書類やドキュメント類、議事録が残される会議などである。私的なコミュニケーションの多くはフォーマットがなく記録に残らないもので、電話による打ち合わせや口頭での質問、確認などである。公的なコミュニケーションは客観的な記述ができ、後で確認が容易であるという特徴がある反面、詳細な部分までは明記されないし、結論しか残らないためそこに至る経過がわからない、という難点もある。その結論に至るまでの交渉、その結論をどう解釈するか、その結論を実行する際の詳細な手続き、などの具体的な事項は私的なコミュニケーションで処理される。そのため、実際の作業内容は私的なコミュニケーションで決められているといっても過言ではない。

問題になるのは、公的なコミュニケーションと私的なコミュニケーションは必ずしも同じメンバーで行なわれていない点にある。公的なコミュニケーションは組織的なまとまり、すなわち作業単位のチームや制度上の関係で行なわれる。一方私的なコミュニケーションは座席の距離や心理的な人間関係のまとまりで行なわれている。仕事のまとまりと人間関係のまとまりが一致している場合は、公的な情報伝達の不備な点を日常会話などで補えるため、業務がスムースに遂行されると考えられる。が、一致していない場合、曖昧な情報の確認に手間取ったり、お互いの誤解に気づかなかったりというような作業効率の低下が予想される。

④多人数による問題

プロジェクトが大きくなると、それだけでコミュニケーションの問題が増大する。

第一に、人数が多くなるとすべてのメンバーと親しくコミュニケーションすることは困難になり、ほとんどのメンバーとは書類上の情報伝達だけになる。前項でも述べたように私的コミュニケーションが不足すると作業がスムースに運ばず、意図の統一が難しくなる。

第二に、大規模になりパッケージ数が増えるとその間の調整という問題が増える。

管理者はパッケージ担当者間の利害を調整しつつ全体を一つのプロジェクトとして統合していかねばならないが、パッケージ数が増えるほどその調整は難しくなる。

第三に、大規模になると伝達ルートが複雑になり、情報の遅延や歪曲が発生しやすい。特に階層型組織では関係者同士直接に話ができず、間に誰かが介在する場合が多い。介在する人数が増えるほど伝達速度が遅れるうえに、途中で第三者の解釈が混ざるなどして意図が正しく伝わらない危険もある。

第四に、知識の分散という問題がある。大規模なシステムになると一人の人間が全体を詳細まで把握することは難しく、知識や情報は複数の人間に分散する。一つの問題を解決するのにも複数の人間から関連情報を収集しなければならなくなり、コミュニケーションの数が増え、コストがかかる。また知識の分散が進むと誰がどの情報を持っているかという情報もわからなくなり、必要な情報が得られなかったり、必要な人間へ情報が届かなかったりという問題も発生する。

⑤進捗管理の難しさ

進捗管理の目的は、メンバーの作業の進み具合を定期的にチェックすることによって、遅れているところがあれば納期に間に合うように調整することと、問題があれば早期発見し解決することである。しかし、進捗報告書には遅れた結果だけが書かれていて真の原因がわからない。そのため管理者はメンバーの本音を探るためにじっくりと話し合える雰囲気作りに気を使っているが、大規模なプロジェクトになるとすべてのメンバーに気を配ることが難しく、問題がわかりにくくなっている。

⑥メンバーの教育に伴う問題

職業訓練としてＯＪＴ（On the Job Training）はもっとも一般的な方法である。講義や本から得た知識とは別に、現場での経験を積み、熟練者を模倣することは意義がある。普通、未熟練者はプロジェクトの単純な作業から参加して作業を覚えていく。しかし、未熟練者が参加すると熟練者が指導に時間をとられ、未熟練者の作業も遅いため工程に遅れが出る、という難点がある。またプロジェクトの大規模化にともない作業が分業化し、全体を見通した教育が難しくなっている。

以上ではソフトウェアに限らず協同作業一般に関わると考えられるコミュニケーションの問題をあげた。特に大規模プロジェクトはプロジェクト内に複数の小プロ

ジェクトを抱えていることになり、問題が複雑になっていると言える。次に、ソフトウェア開発に固有の問題をまとめる。

（2）ソフトウェア開発作業の側面から見た問題

①ソフトウェアの曖昧さから来る問題

ソフトウェアは思考の産物であり、思考のプロセスも結果も抽象的である。ソフトウェアにとって結果よりもプロセスが重要になることが多い。ここでいうプロセスとは、要求分析からソフトウェア設計に至るまでの設計者の思考プロセスと、ソフトウェアシステムがある機能を実現するための処理プロセスの2つの意味がある。ソフトウェアは同じ機能を実現するために複数の処理方式（プロセス）が可能である。例えば計算の方法や分岐の順序、データ転送の方法などである。どの処理方式を用いたかによってソフトウェアシステムの読解容易性や変更容易性ばかりでなく速度までも異なってくる。どの処理方式を用いるかは設計の過程で様々な条件を考慮したうえで設計者の意図で決まる。なぜ設計者がその処理方式を選んだか、他の方式ではなぜいけないのかを理解するためには設計者の思考プロセスを知らねばならない。また、近年のソフトウェアシステムはユーザインタフェースの占める割合が多くなっている。ユーザインタフェースは「わかりやすい」「見やすい」などの感覚的な言葉で表現され、システムの内容や評価をますます曖昧にしている。このようなソフトウェアの曖昧さ、プロセスの多様性が協同開発を難しくしている。

②適切な表現方法がない

ソフトウェアはドキュメントやダイアグラム、プログラムの形式で記述されるが、それは設計者の思考の結果の記述である。前項で述べたようにソフトウェアにとってプロセスが重要であるが、従来の標準化された表記方法では意図や思考の過程を記述することが難しい。また誤解が生じないように記述を詳細なものにしようとすれば膨大な量の仕様書になり、書くことも読むことも困難になる。ソフトウェアのイメージや意図を具体的に表現する適切な方法がないので、意図を伝達するためには何度も議論したり確認したり、丁寧なコミュニケーションが必要になる。

適切な表現方法がないことは、評価や管理の難しさにもつながる。思考の過程が表現できないため、進捗を管理することが難しい。また評価はソフトウェアの成果（ドキュメントやプログラムの量）でしか測定できず、設計プロセスの難易度やそ

の達成度を評価する方法は未だ確立していない。

③部分と全体の問題

システムが完全に独立した部品の集合で構成できれば協同開発が容易になると言える。しかし、ソフトウェアの要素は複雑に相互関連しており、完全に独立した部品に分割することは難しい。各部品に独立性をもたせるためにはコストアップやスピードの低下も避けられず、どうしてもある程度は部品間の相互依存関係が残ってしまう。このような複雑なシステムの分割作業においては、全体の意図を正しく把握していないと一つの部品の役割を正しく理解することができない。ある部品と全体の関係を理解できなければ、その部品を設計者の意図どおりに開発することも困難である。各部品がばらばらの意図で開発されれば、単体としては機能していても組み合わせると全体にちぐはぐなシステムになり、信頼性が低い上に保守や変更も難しくなる。

一方、大規模なシステムになると一人の人間が全体を把握することが難しくなる。管理者は概要だけを把握し、パッケージ担当者は詳細部分を理解する、というようにメンバーは各々のレベルでソフトウェアを理解している。パッケージ担当者には全体が見通せないため、管理者はパッケージの開発方針がズレていないかを常にモニターしておく必要がある。ちょっとした意図の違いでも積み重なると重大な問題に発展する危険もある。

以上ではソフトウェアの協同開発における問題について述べた。ソフトウェア開発では設計者の意図を理解することが非常に重要にもかかわらず、その意図を表現する適切な方法がないためコミュニケーションが困難になっている。先に述べたようなメンバーの個人差や立場の差からくるコミュニケーションギャップにくわえて、ソフトウェア自身の抽象性がコミュニケーションをさらに難しくしている。次に、本論文で扱うコミュニケーションの問題とそのアプローチについて述べる。

3．5　本研究の位置づけとアプローチ

第２章でソフトウェア開発の生産性向上研究について整理し、本章では現場の開発プロジェクトにおける問題点を述べた。従来のソフトウェア工学は、生産技術で

はプログラミング言語やソフトウェア設計手法の開発、管理技術ではメンバーの個人差を排除した標準化が主流で、「ソフトウェアの理論の確立→現場への適用」という図式であった。これに対し、本研究では「現場の調査→問題点の分析、理論づけ→支援機能の開発」という方向をとる。また、従来は工学者による技術的なコミュニケーション支援が主流だったのに対し、認知科学や社会心理学を用いた人間的な側面からの支援を試みる。

一方、ＣＳＣＷ研究では協同作業の調査やモデル化など基礎的な支援研究も行なわれているが数は少なく、対象となるプロジェクトや課題はまだ限られたものであった。また分散環境の支援に対してはマルチメディアツールの提供にとどまり、具体的な問題へ適用している研究は少ない。

このような従来の研究を踏まえ、本論文では以下の方針に基づいて支援技術の開発を行なった。

・現場の調査をもとに支援すべき問題を決定する
・様々な側面からコミュニケーションを捉える
・問題に対し、理論的な考察を加える
・具体的な支援機能を提案する
・プロトタイプシステムを開発し支援機能の実現可能性を示すとともに、その効果を評価する

本章では様々なコミュニケーションの問題が明らかになったが、そのすべてをここで扱うことはできない。そこで特に重要と考えられる以下の問題を取り上げる。

（1）ソフトウェアの意図を伝達するための適切な表現方法がないという問題。
（2）大規模なプロジェクトにおいてコミュニケーションコストが過大なものになっているという問題。
（3）プロジェクト内のコミュニケーションを評価し、問題点を明らかにする手法がないという問題。

これらの問題に対し、それぞれ以下のようなアプローチで支援方法を考察する。

①認知科学的アプローチ

ソフトウェアの意図を理解するための人間の認知的メカニズムを考察し、それに基づいた記述方法を提案する。

②組織論的アプローチ

組織におけるコミュニケーションプロセスを分析し、支援方法を提案する。

③社会心理学的アプローチ

社会調査の技法を応用し、コミュニケーションを定性的に測定して問題点を明らかにできる評価手法を提案する。

3．6　結言

本章では、実働プロジェクトに対する調査結果からコミュニケーションの問題を整理し、その問題に対する本論文のアプローチと従来の研究の中での位置づけについて述べた。

調査結果から、コミュニケーションの不良がソフトウェア生産性を阻害する重要な要因であることが確かめられた。コミュニケーションが失敗する原因としては、メンバー間の立場や経験・思想の違い、ソフトウェアに適切な表現がないことがあげられる。他にも大規模プロジェクトでは小規模プロジェクトにはなかった問題が発生することがわかった。調査では全てのメンバーが各々の視点からコミュニケーションの重要性を指摘し、コミュニケーション支援の必要性を再確認できた。しかし、これまではコミュニケーションの問題は管理者個人のコミュニケーション能力に任されがちで、積極的な支援へつながっていなかった。本論文では、重要と思われるいくつかの問題に対して従来とは異なるアプローチから支援を試みる。

《付録1》　プロトコル抜粋

1「あまり説明はしてもらってないですね。あんなの徐々にわかってくるんですね。だからこの全体がわかったことないです。まだ全体を把握していないんですけども。」（新人プログラマがシステム全体の説明について）

2「遅れていることしかわからない。外に見えているのは時間的な遅れですね。～その原因までわからないから～そんなんじゃハッパかけるしかないんですね。」（システム設計者が進捗管理について）

3「やっぱり隣に座っているって利点がありますね。～いろいろ付随したお話しとかしてくださるから。」（新人プログラマが日常のコミュニケーションについて）

4「本当は（プログラマが）机を並べてやっていくのが一番いいんですけども～横の連絡がすごく難しいです。」（システム設計者が日常のコミュニケーションについて）

5「仕事の話は、やはり隣の人に聞くことが多い。コーディングのことならば（離れているプログラマに）電話で聞き、わからなければ出かけていきます。」（新人プログラマが日常のコミュニケーションについて）（新人プログラマが日常のコミュニケーションについて）

6「もう全部自分で作ってしまいたいという感じで、」（システム設計者がプロジェクトの運営について）

7「管理者はあまりいらないんじゃないかという持論をもっているんですね。」（システム設計者がプロジェクトの運営について）

8「始め、（このシステムが）何やってるかわからなかったです。それをわからないまま何か作ってやってた感じ。」（新人プログラマがシステムイメージについて）

9「だから、つい最近までどういう風になって動くのか（プログラマは）みんなわからなかったですね。」（新人プログラマがシステムイメージについて）

10「設計って言っても、そんな100%書ききれないですよ。～これくらいのことは当たり前に解釈して、ここまでやってくれるはずだと、それを前提としないとどうしようもないんですね。」（システム設計者が仕様書について）

《付録2》 質問紙

中研では、ソフトウェア開発における協同作業支援の研究に取り組んでいます。前回は、制電シ生技部においてインタビュー調査を行ない、その結果をもとにソフトウェア意図伝達支援ツールのプロトタイプを試作しました。

今後も大規模ソフトウェア開発における協同作業の問題点とその支援方法についてさらに研究を進める予定です。その資料とするために、皆様の仕事内容と、大規模ソフトウェア開発における問題点についての御意見を伺いたいと思っています。御自身の体験・疑問・不満など、忌憚のない御意見をお聞かせください。

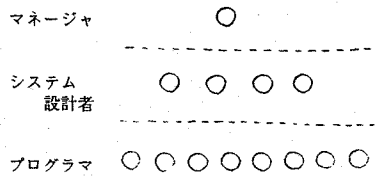
[1．あなたはどのような仕事に従事しておられますか。さしつかえなければお答えください。]

1-1 この仕事を始めて何年くらいになりますか？

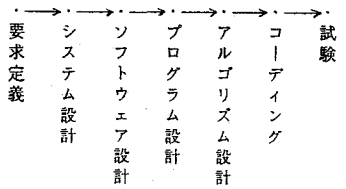
1-2 下の図に当てはめて言えば、あなたの仕事内容はどのあたりですか？
できれば、なるべく詳しく教えてください。

1-3 あなたが担当しているプロジェクトの規模・内容はどのようなものですか？
（プロジェクトの人数、工数、開発期間、など）

プロジェクトにおける役割範囲



開発プロセスにおける担当範囲



[2．次のようなテーマについて、自由にお書きください。テーマにこだわらなくても結構です。]

2-1 大規模ソフトウェア開発における問題
（工程の遅れ、仕様のズレ、メンバーの変更、バグ探し等々、体験したこと、聞いたこと）

2-2 その問題に、どのように対処しましたか

2-3 プロジェクトにおけるコミュニケーションの問題
（人間関係の摩擦、管理の難しさ、モチベーションの低下、など）

2-4 受注先との問題

2-5 その他、困ったことやそれをどう解決したかなど、何でも

第４章　劇場モデルに基づいたソフトウェア意図伝達支援システム ＣＯＭＩＣＳ

４．１　緒言

第３章で述べたように、プロジェクトにおけるコミュニケーションの問題の一つはソフトウェアの意図を表現する適切な方法がないことである。本章では、ソフトウェアの意図やイメージを表現する方法として劇場モデルを提案する。劇場とは、俳優が舞台の上でドラマを演じ、観客に見せる空間である。観客はドラマの進行を見て脚本家の意図を読み取ることができる。これは人間がまとまりの中から意味を把握する認識メカニズムを持っているからである。ここではソフトウェアの構造がドラマと類似していることに着目した。ソフトウェアの進行を適切に表現すれば、その中から設計者の意図を読み取ることが容易になると予測できる。そこで劇場モデルに基づいた表現方法を実現する意図伝達支援ツールＣＯＭＩＣＳ（COmputer-based Media for Intention Communication in Software）を試作し、その有効性について検証する。

以下では、まずソフトウェア開発プロジェクトに対するインタビュー結果からメンバー間のコミュニケーションの実状と問題をまとめる[1)]。その結果を踏まえてソフトウェアの意図伝達方法として劇場モデルに基づいた方法を提案し、それをワークステーション上でコミュニケーション支援ツールを開発した[2)]。また、本ツールの試用を通じて、その有効性についても検証した。

４．２　プロジェクトにおけるコミュニケーション

４．２．１　コミュニケーションの重要性

コミュニケーションの問題を考える前に、プロジェクトにおけるコミュニケーションの重要性を考える。

第一に、ソフトウェアは思考の産物である。複数で開発すれば、複数の思考で作ることになる。全体として統一の取れたシステムにするためには、各人の意図を統一し、矛盾のないものにしなければならない。

第二に、ソフトウェアは、構成要素が密接に関連しているため、単純に分割して

開発するのは困難である。しかし、現実には大規模システムは、協同で開発せざるを得ない。構成要素が相互に関連しているものを協同開発する場合、各要素の担当者も密接に連絡を取りながら作らねばならない。

第三に、一部を担当するだけのプログラマでも、全体を知らなければ良いプログラムを書けない。単独のモジュールとしては最良のプログラムでも、全体にとって良いとは限らないからである。無駄に見えるような変数を用意しておいたり、バグが見つけやすいような構造にしておくと、モジュールとしては冗長でもシステム全体としては効率がよくなる。全体にとって良い部分を作るためには、部分と全体との連絡をスムースにしておく必要がある。

このように、ソフトウェアのプロジェクト開発にとって、意図伝達のコミュニケーションは重要な役割を果たすものと考えられる。

4．2．2　コミュニケーションの問題点

第3章で述べたインタビュー結果の中から、本章で扱う問題に関係の深いものを再度まとめる。

プロジェクト内でのコミュニケーションは、記録に残る公的なもの（ミーティング、レビュー、等）と記録に残らない私的なもの（電話、口頭での雑談、等）に大別される。公的なコミュニケーションは話し手から聞き手への一方的な伝達になりがちで、意図が伝わらないケースがある。私的なコミュニケーションは公的なコミュニケーションで伝わらない部分を補っているが、プロジェクトの中でいつくかの私的コミュニケーショングループができてしまい、プロジェクト全体の意図が統一されにくいという危険がある。

以上の結果から、調査したプロジェクトには次のような問題点が考えられる。

・プロジェクト内のコミュニケーション量が多いにもかかわらず、意図の統一がとれていない。
・そのため、プロジェクトメンバーが抱いているシステムイメージは、人によりかなり異なっている。
・また、極端な場合には、システムを理解できていないままコーディングを始めることもある。

またインタビュー結果から、コミュニケーションを妨げる主な要因として次の2点があげられる。

①メンバーの知識・経験・視点の差が大きいこと

経験や能力の個人差ばかりでなく、立場や環境による考え方の差も大きい。

②ソフトウェアが抽象的で表現が難しいこと

ソフトウェアを表現する適切な方法がない。そのため、自分のイメージを言い表せなかったり、人の意図をうまく理解できなかったりする。また、自分の中でもイメージが曖昧な場合が多い。

これらは、ほとんどのプロジェクトに共通の問題であると思われる。ソフトウェアの表現方法としては従来から仕様書やフローチャートが研究されてきた。これらの方法は個人差や誤解を防ぐために標準化され、詳細な内容を正確に伝達するには効果的である。しかし、一方では以下のような問題点があり、具体的なイメージや意図が伝わりにくかった。

・量が多く、作成するのにも読解するのにも時間がかかり過ぎる。

・細部は詳細に記述されるが、全体のイメージがわかない。

・要素間の関連がわかりにくい。

仕様書にソフトウェアの意図を盛り込むことはできないというのは、多くのシステム設計者の意見である（第３章付録・プロトコル１０　参照）。経験の豊富なプログラマであれば、仕様書からシステム全体のイメージをふくらませることもできる。しかし、経験のないメンバーは、システム全体がどんな動きをするのかわからず、自分に直接関係ないところはわからないという状態のままプログラミングしてしまう。その結果、プロジェクトの生産性低下、ソフトウェアの品質低下、などの危険が生じる。

プロジェクトにおけるコミュニケーションを支援するためには、従来の方法では表現できなかったシステムの意図やイメージを表現する方法が必要である。そこで、次章ではソフトウェアの意図の表現方法を検討する。

４．３　劇場モデルに基づいた意図の表現と伝達

４．３．１　人間の理解のメカニズム

適切な表現方法を考えるためには、まず人間の理解のメカニズムを知り、それに適した方法を採らなければならない。

仕様書や説明書は、機能別・構造別に記述されている。人間が機械などを理解するとき、構造と機能を見る。それは機械がどうやって動くかという理屈を知りたいからである。例えば、自動車のエンジンが回転し、このギアが力を伝えて車輪が回るという仕組み（理由）と車が走るという機能（結果）が頭の中で一致すれば、車が納得できる。故障しても、どこが壊れているのかが推測できる。しかしソフトウェアの仕様書は、仕組みと機能を別々に述べている。プログラマは、自分の作ったモジュールとシステムの動きとの関係をうまく結び付けられない。構造と機能の関係を結び付けるものとして、次の点が必要と考えられる。

①システムの動作が、具体的にわかる。

②部分の動きと全体の動きの関連がわかる。

③部分同志の関連がわかる。

これらの要因がイメージや意図の伝達に必要な理由を、人間の理解という視点から考え直してみる。人間の認識のメカニズムには、

・文脈の中で現象を捉えようとする

・まとまりを見ようとする

という特徴がある[3)]。人間は日常何かの現象を認識するとき、前後の流れや周囲の環境の中で理解する。個々の要素の意味を認識するより先に全体の意味を認識する傾向がある。例えば、文章にサッと目を通せば大体の意味を把握することができる。もし単語が一つ間違っていて、文法的に見れば意味が通らない場合でも、それに気がつかずに大要を理解できる。逆に一つ一つの意味にとらわれ過ぎると、全体が見えなくなる。例えば、外国語を丁寧に逐語訳していると、結局概要をつかめなかったりすることがある。また、「走る」という行為の説明として「目的地になるべく早く到着するために急ぐ様子」と言われるより、「遅刻しそうなので走る」と言われたほうが頭の中でイメージがわいてくる。その行為自体について説明されるより、その周囲の状況・理由・目的を聞くほうがよくわかる場合もある。

ソフトウェアの理解も同様のことが言える。個々のモジュールにばかり注目していると全体が捉えられなくなる。ソフトウェアの仕様書からイメージが描きにくいのは、個々の要素を詳細に正確に述べることに重点を置き、人間の認知メカニズムに適さないからだと考えられる。

4．3．2　劇場モデルの提案

ここで、意図伝達方法として劇場という表現方法を考えてみる。劇場は、抽象的

なテーマを観客に伝える手段である。脚本家は自分の意見や感動などのテーマを表現するためにシナリオを書き、俳優や背景・大道具を設定する。俳優や道具達は、舞台の上でストーリィを演じてドラマを構成する。脚本家の意図は俳優の一人一人、大道具の一つ一つにあるのではなく、それらの相互作用プロセスの中にある。観客は舞台の上のプロセスを見て脚本家の感動や意図を読み取っていく。登場人物はその劇の中で位置づけられる。劇場は脚本家と劇と観客から構成され、脚本家の意図が劇として表現されて観客に伝達される「場」である。

著者はソフトウェアの構造と機能は劇と類似していると考えられる。そして、脚本家が自分の演出した劇を観客に伝達するコミュニケーションと、システム設計者が自分の設計したソフトウェアをプログラマに説明するコミュニケーションの機能と構造も類似している（表4.1）。システム設計者は、ある機能を実現するためにソフトウェアを設計する。システムが効果的に機能するようにフローチャートを考え、モジュールやデータ、ハードウェアなどの要素を設定する。ソフトウェアはこれらの要素が相互作用してシステムを実現していく。設計者の意図は全体の流れの中にあり、個々の要素だけを見てもわからない。モジュールは、単体でのプログラムとしての良さよりも全体の中でどのような役割を果たしているかが重要になる。

表4．1　劇場とソフトウェアの意図伝達

類似点	劇場の意図伝達	ソフトウェアの意図伝達
構　造	脚本家、登場人物、観客で構成される	システム設計者、設計内容、プロジェクトメンバーで構成される
機　能	脚本家の意図は、登場人物１人１人が持っているのではなく、動作やセリフの連続の中に表現されている。 観客はその流れの中からテーマを読み取る。	設計者の意図は一つ一つのモジュールを見てもわからない。それらの関係性の中に意図がある。 意図がわからなければモジュールの本当の役割はわからない。
	同じテーマでも脚本家によって演出の仕方が異なり、観客の印象も異なる。	同じ機能のシステムでも設計者によって全然違うソフトウェアになることもある。
	観客は劇を一つのまとまりとして見ることができる。が、一人の俳優を中心に見ることもできる。	メンバーはシステムの全体像を知ることが必要である。が、自分の担当モジュールに関連したところを重点的に知ることも必要である。

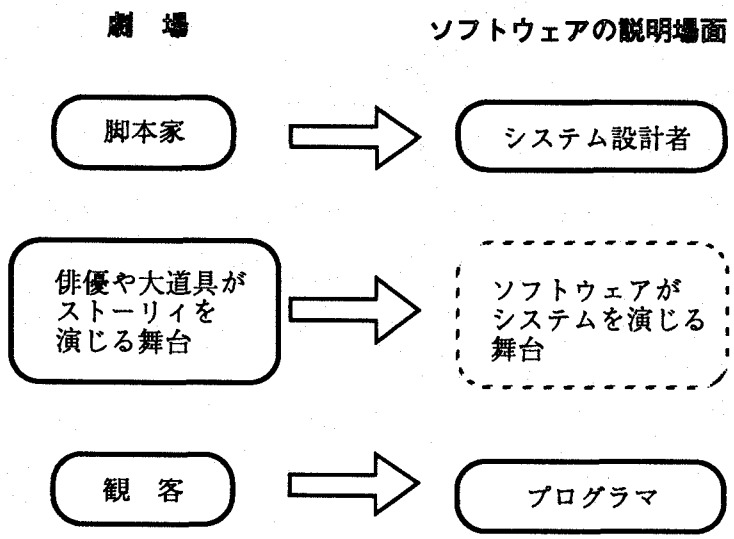


図４．１　劇場モデル　(劇場からソフトウェアの説明場面へのメタファー)

観客が劇を見て脚本家の意図を読み取ることができるのは、人間が「まとまりが全体として持つ包括的な意味」を理解するような認識メカニズムを持っているためである[4)]。従って、プログラマがソフトウェアをまとまりとして見れば、その中からシステム設計者の意図を読み取ることができるはずである。しかし、従来の方法ではソフトウェアをまとまりとして表現できない。

そこで、システム設計者の意図をドラマにしてプログラマに伝える表現方法「劇場モデル」を提案する（図 4.1）。劇場には、以下の点が表現できる舞台空間を設定する。これらは人間が構造と機能の関係を理解するときの３つの要件を考慮したものである。

①システム全体のイメージと具体的な動き

②全体の中での、個々の構成要素の位置づけ

③構成要素同士の関係性

このような舞台を実現することにより、ソフトウェアの具体的な動きを伝達し、プログラマが自分が担当するモジュールの位置づけと役割を理解することを支援する。

なお、劇場というメタファーを用いた試みはこれまでにもいくつかなされている。一つには、Programming-by-Rehearsal システム[5)]や対話的グラフィックプログラミン

グ環境 Pict/D[6] などの視覚的なプログラミング環境がある。また、Perfoming Art Paradigm[7] は、ソフトウェア開発プロセスそのものをパフォーマンスと見なしたプロセスモデルである。これは、従来の Water-Fallモデルに対するメタプロセスモデルで、ソフトウェア開発の設計からテストまでを Practice : Performance : Postmortem の３つの段階、すなわち練習・実演・反省の繰り返しとして考えようというものである。

これに対し、本劇場モデルは、ソフトウェアの表現方法を提案し、コミュニケーションを支援するためのモデルであり、劇場メタファーの用い方が異なっている。

４．４　劇場モデルの実現ツール　ＣＯＭＩＣＳ

４．４．１　ＣＯＭＩＣＳの概要

ソフトウェアにおける舞台を提供し、ソフトウェアの意図伝達空間を創りだすツールとしてＣＯＭＩＣＳ（COMputer-based Intention Communication System）を開発した。ＣＯＭＩＣＳの目的は、正確で詳細な仕様を伝えることではなく、システムの具体的なイメージを描き、ソフトウェアの意図を伝えることである。ＣＯＭＩＣＳは、劇場モデルを基にした表現方法を実現している。

ＣＯＭＩＣＳ上での劇場モデル表現について説明する。劇場モデルは、システムの構造を劇の構造と対比させ、システムをソフトウェアモジュールやハードウェアが演じる一連のドラマと見なし、そのドラマをワークステーション上で表現しようというものである。その他に、劇場モデルには視点という概念を導入する。システムを説明するとき、説明者の意図や説明内容によって視点の取り方が異なってくる。本論文では、視点の取り方も説明者の意図を表現する重要な情報であると考え、説明者の視点を明示する方法を取った。以下、劇場モデルを構成する要素についての定義と、ＣＯＭＩＣＳでの実現方法について述べる。

（１）視点

脚本家は自分なりの捉え方でテーマを表現する。同じテーマでも脚本家によって異なる表現になる。この「ものの見方・捉え方」を視点と呼ぶ。

ソフトウェアの場合も同様に、様々な視点から眺めることができる。システム設計者の意図や説明の仕方によって表現方法が異なる。例えば、同じシステムでも、プログラマの視点をとればモジュールに分割した表現になり、ユーザの視点をとれ

ばインタフェースを中心とした表現になるであろう。

ＣＯＭＩＣＳでは一つの視点を一枚のウィンドウで表す。視点は複数設定できる。システムのドラマが進行しているとき、視点のウィンドウは常に画面上に表示されている。

（2）ドラマの構成要素

劇場は、ドラマを作る脚本家、ドラマを構成する俳優・舞台装置・小道具・照明等の要素、ドラマを見る観客、から成る。

ソフトウェアの場合は、システムを説明する者、システムを構成するモジュール・データ・ハードウェア・ネットワーク等の要素、システムの説明を聞く者、から成る。

ＣＯＭＩＣＳは、システムの構成要素を表現するツールである。ＣＯＭＩＣＳでは、一つの構成要素を一つのボックスで表す。ボックスは視点ウィンドウ上に記述される。例えば、モジュール構成という視点のウィンドウ上には、ソフトウェアモジュールの構成図がボックスで記述される。各ボックスは一枚のピクチャーを持っている。ピクチャーは、ボックスに関する説明など任意の内容をテキストや図で書くメモ用ウィンドウである。自由に開いて見ることができる。

（3）リンク

ドラマの構成要素間には、人間関係や人間とモノの関係など、何らかの関係がある。その関係はドラマのストーリィ展開の上で重要である。

ソフトウェアの場合にも、要素間の関係は非常に重要である。モジュール間、モジュールとデータ、ネットワーク構成など、要素はすべて何らかの関係でつながっている。

ＣＯＭＩＣＳでは、関係をリンクで表現する。任意のボックス同志をリンク付けでき、リンク付けするとボックスの間に線が引かれる。異なるウィンドウ上のボックスをリンク付けしたときは、リンク情報は存在するが、線は引かれない。

（4）シナリオ

劇場では、脚本家がシナリオを書く。シナリオは、ドラマの構成要素の動きやストーリィ展開を書いたものである。

ソフトウェアの場合、システムを説明する者がシナリオを書く。シナリオとは、

システムの動き、プログラムのアルゴリズム、開発の進捗などを示したものである。一つのシステムでも、説明の内容によって複数のシナリオができる。

ＣＯＭＩＣＳでは、シナリオを場面の連続として表現する。一つの場面は一つのボックスで表し、ボックスを並べたものがシナリオになる。一本のシナリオは一枚のシナリオウィンドウ上に記述する。このボックスにもピクチャーがついており、場面の状態をテキストと図で書くことができる。

（５）舞台

劇場では、ドラマの構成要素がストーリィを展開していく場所が舞台である。この舞台の時間的展開を追うことにより、観客は脚本家の意図を把握することができる。

ソフトウェアの場合も同様に、システムの構成要素がシナリオにしたがって時間的に展開していく様子を見せることにより、ソフトウェアの展開、即ち、システム

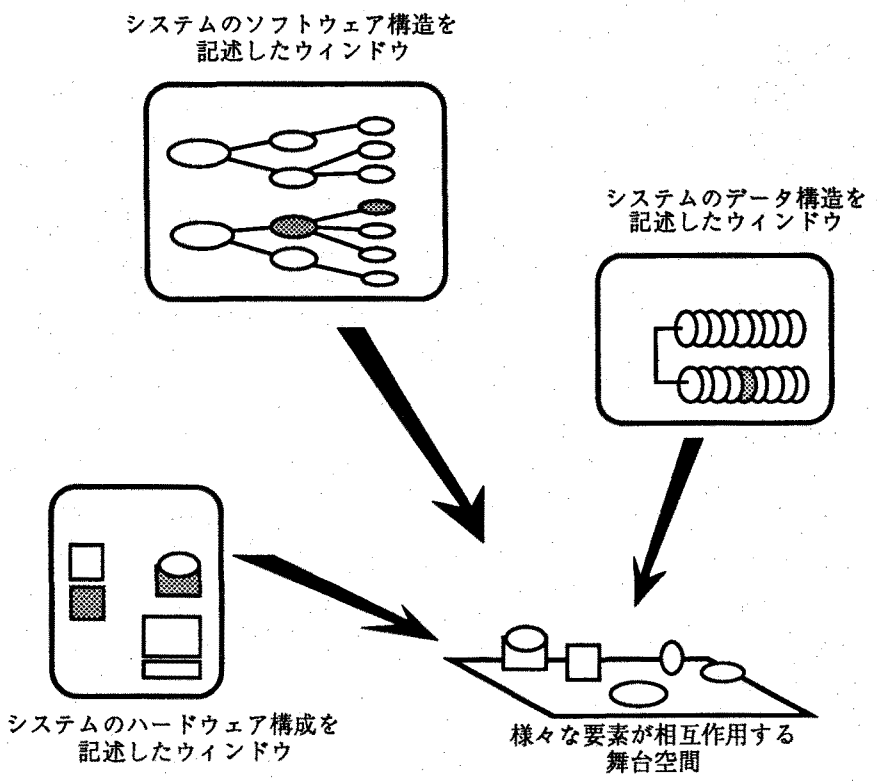


図４．２　劇場モデルによるソフトウェア表現の例

の動き、プログラムのアルゴリズム、開発の背景等を伝えられるものと思われる。

ＣＯＭＩＣＳでは、舞台をステージとして表現する。画面にステージ用の空間を設け、場面の様子を表示する。場面の様子は、その場面に関連するボックスを点滅させると同時に、ピクチャーをステージ上に表示することにより表現する。シナリオの展開に従って場面が進み、ボックスの点滅状態とステージに表示されるピクチャーが紙芝居のように変化する。点滅するボックスや表示するピクチャーは任意に設定できる。

図 4.2 はソフトウェアの劇場表現イメージである。劇場には、俳優の集合や小道具の集合、効果や背景の集合など、いくつかの要素集合がある。舞台では、それらの要素が交互に登場し、相互作用によってストーリィを表現していく。ソフトウェアも図のようにいくつかの要素集合から成る。ＣＯＭＩＣＳでは要素集合間の関係と、それらが舞台の上で相互作用してシステムというドラマが進行していく様子を表現する。図 4.2 のイメージをそのまま画面にしたのが図 4.3 である。

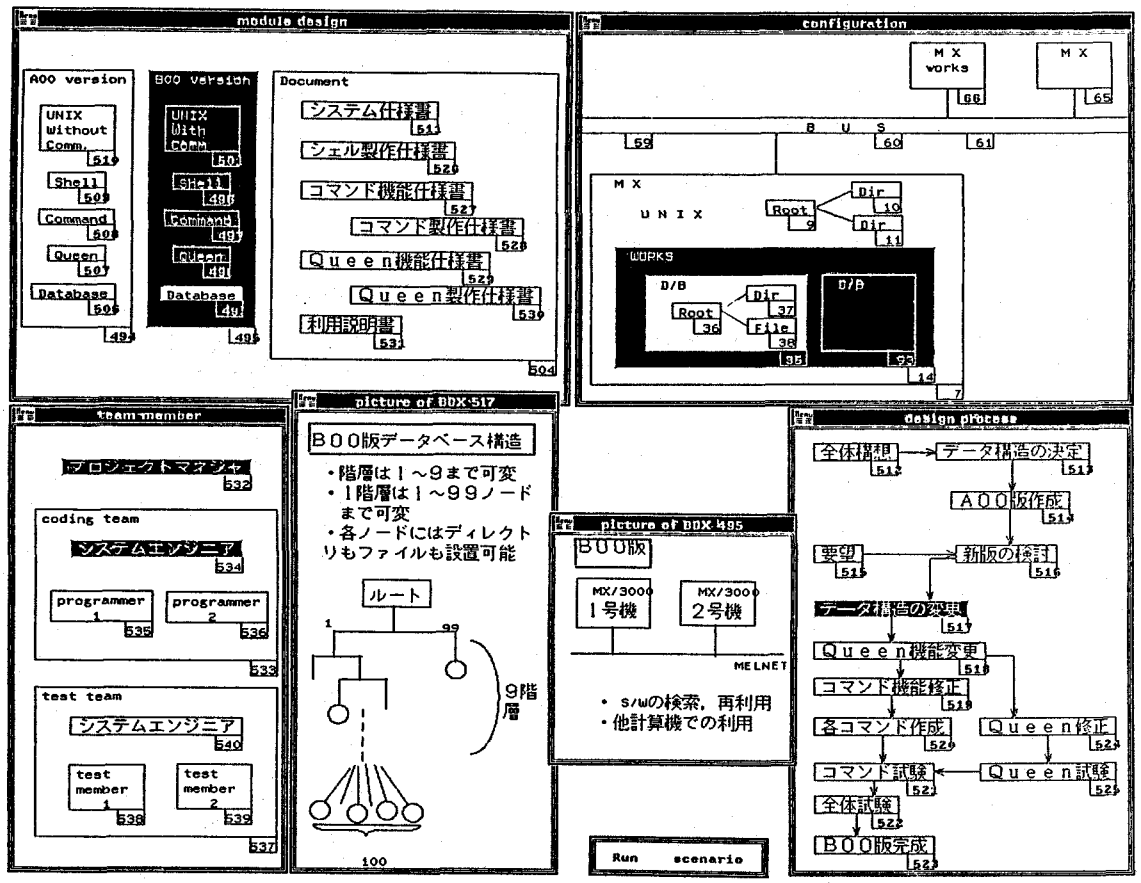


図４．３　ＣＯＭＩＣＳ　の画面例

図 4.3 は実際のシステムのデータを入力したＣＯＭＩＣＳの画面例である。これは、システムが第一版から第二版へ更新されたときのプロセスを説明するためのデータである。画面は、プロジェクトメンバーの構成図ウィンドウ（左下）、ソフトウェアモジュール構成とそれに対応するドキュメント構成のウィンドウ（左上）、ネットワークとディレクトリ構造のウィンドウ（右上）から成る。右下のウィンドウはシナリオウィンドウで、システムが第一版から第二版に更新されたときのプロセスをボックスで表現している。ここでは、シナリオの第６場面での状態が表示されている。この場面は、システムの更新に伴いデータ構造がどう変更になったかを説明する場面である。この場面に関連のあるボックスが点滅し、画面中央のステージ部分には場面の状態を説明するピクチャーが表示されている。

図 4.4 はこのシナリオが第５・６・７場面と進行している様子である。第５場面は第二版の開発方針を説明する場面、第６場面は図 4.3 と同じもので、更新に伴うデータ構造の変更を説明、第７場面はそ
のデータに直接関わるモジュールの機能変更を説明する場面である。場面が変わると点滅するボックスが変化し、ステージ部分のピクチャーも変化している。

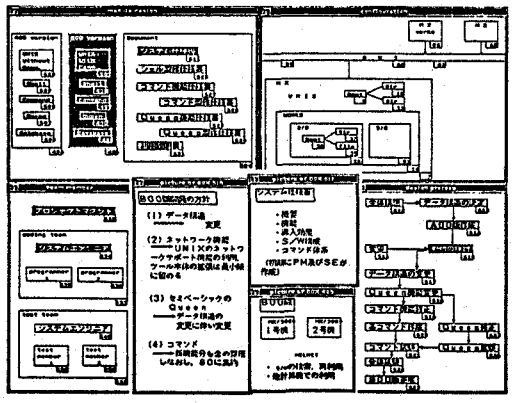
なお、ＣＯＭＩＣＳは、ソフトウェアの説明ツールとして開発したが、ソフトウェアの説明には様々なレベルがある。プロジェクトの初期にシステム設計者がプロジェクトメンバーに行なうシステムの概要説明、開発途中にプロジェクトのメンバー間で行なわれるシステムに関するディスカッション、開発終了後、開発担当者から保守担当者に対して行なわれる説明等、プロジェクトの工程に応じた説明レベルが考えられる。説明の内容・目的によってソフトウェアの記述レベルも異なってくる。ＣＯＭＩＣＳでは、視点や要素の設定の仕方によって概要からプログラムレベルまでの記述が可能であり、様々なレベルの説明に利用できるものと考えられる。

４．４．２　ＣＯＭＩＣＳの機能

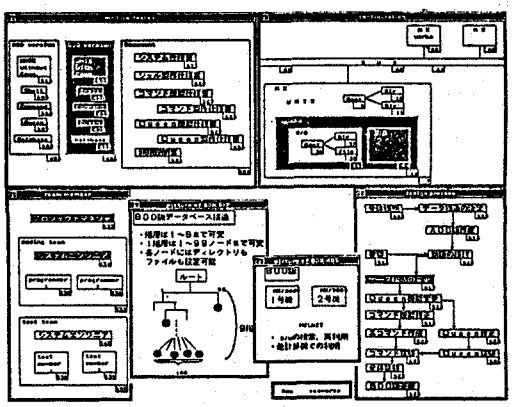
ＣＯＭＩＣＳには、以下のような機能がある。

○設計者の自由な視点からシステムの構造を記述できる。

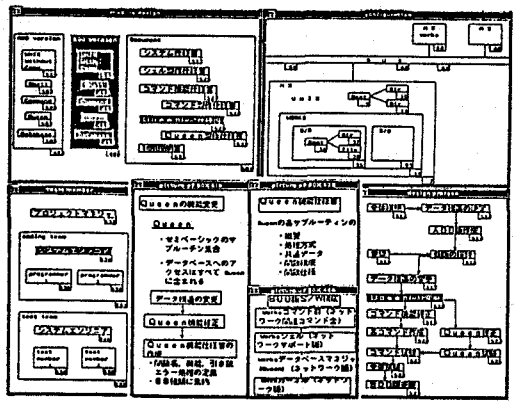
視点は任意に設定できる。誰にどのようにシステムを説明するか、という意図によって設定の仕方が異なってくる。例えば、「各モジュールの担当メンバーとドキュメント」という視点でシステムを説明する場合には、プロジェクトチームのメンバー群、ソフトウェアモジュール群、ドキュメント群、という要素群を設定すればよい。また、大モジュール群、サブルーチン群、基本関数群、など同じソフトウェアを記



第５場面
〃システム第二版の開発方針について〃



第６場面
〃システムの更新に伴う
データ構造の変更について〃



第７場面
〃データ構造の変更に関わる
モジュールの機能変更について〃

図４．４　シナリオの動作例

述レベルによって分類することも可能である。

○ソフトウェアの動作をシナリオとしてダイナミックに表現できる。

システムの動きはシナリオで表現する。視点とシナリオの設定の仕方によって、システムの動きやアルゴリズムだけでなく、ソフトウェアの設計課程や更新のプロセスをドラマとして表現することができる。

○プログラマは、自分の理解に合わせたペースでソフトウェアの内容を見ることができる。

シナリオはフローチャートのようにウィンドに表示されるので、プログラマはシステムの動きを目で読むことができる。また、シナリオを自動で走らせたり、わからない箇所だけを繰り返し見るなど、任意のスピードでシナリオが走る。さらに、個々の要素についての詳細な記述や、要素間のリンクも自由に表示させられる。このような機能により、プログラマは説明を受動的に聞くのではなく、自分で様々な角度からシステムを眺め、その動きを確かめられる。自ら主体的に認識しようと働きかけることは、理解を支援する[8)]。

○入力・編集が容易なので、設計者はＣＯＭＩＣＳ上で設計できる。

ＣＯＭＩＣＳで説明するソフトウェアの内容は、システム設計者が入力しなければならない。そのため、設計者に対する負荷が増加する。そこで、ＣＯＭＩＣＳはテキスト入力以外の入力・編集をマウス操作のみで行なえるようにした。操作が容易なので、設計者はＣＯＭＩＣＳ上で設計し、その結果をそのまま説明に使うことができる。

また、ＣＯＭＩＣＳ上で設計すれば、設計者の設計プロセスが記録・再現でき、意図の伝達にとって効果的であると考えられる。

○ソフトウェアの設計履歴を再現することにより、設計者の思考プロセスを辿ることができる。

入力履歴の再現は、まず画面がクリアされ、白紙の状態から入力された順にボックスが登場する。同時にボックスの詳細な説明を表示することもできる。複数の視点で並行してボックスが生成されていく様子から、設計者がそのボックスを設計した経過が表現できる。設計プロセスを再現することにより、「なぜ、どうやって」という視点からソフトウェアを説明でき、設計者の意図の理解を支援するものである。

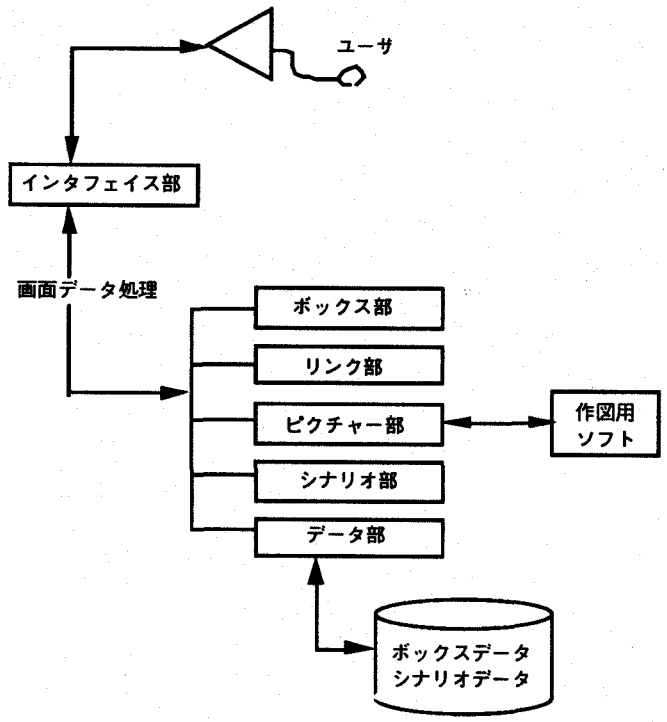
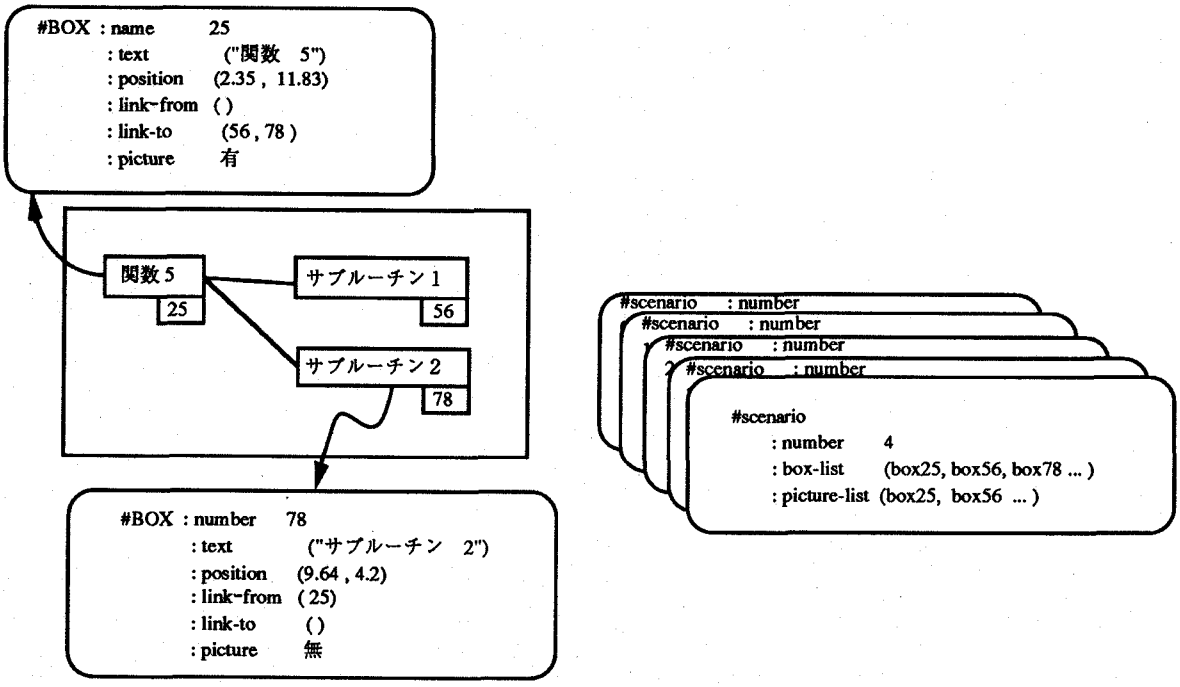


図４．５　ＣＯＭＩＣＳのソフトウェア構成



（ａ）ボックスのデータ
各ボックスは属性を記述した
フレームデータを持つ

（ｂ）シナリオのデータ
シナリオのフレームデータは、
点滅させるボックスと、表示する
説明図のリストから成る

図４．６　ボックスとシナリオのデータ構造

4．4．3　ソフトウェア構成

ＣＯＭＩＣＳは、Common Lispによりワークステーション上で実現した。ソフトウェア構成を図4.5に示す。

1）インタフェース部：ＣＯＭＩＣＳ全体に共通なウィンドウとマウス操作を提供する。

2）ボックス部：ボックスの作成と、コピー・修正・移動などの編集を行なう。

3）リンク部：ボックス間のリンク作成と、リンクの表示を行なう。

4）ピクチャー部：各ボックスの説明や、ステージに表示する画面を作成、表示する。テキストや作図は、外部の作図用ソフトを呼んで行なっている。

5）シナリオ部：シナリオの作成と、シナリオの実行機能を提供する。シナリオの作成は、ユーザがストーリィに従って動くボックスをマウスで選択し、その時、同時に表示させたい画面もマウスで選択していく。

6）データ部：データファイルの保存、読み込みを行なう。

図4.6のようにデータは構造で表現し、ボックスデータとシナリオデータの２種類ある。各ボックスにつき一つのボックス用の構造体が作成され、属性情報を格納する。各シナリオにも一つのシナリオ用の構造体があり、ボックスのリストを格納する。

4．5　ＣＯＭＩＣＳの使用と効果

ＣＯＭＩＣＳは、コミュニケーションをサポートするためのツールである。ソフトウェア開発のライフサイクルの中で、ＣＯＭＩＣＳは以下のような場面で利用できると考えられる。

1）システム設計者からプロジェクトメンバーへのコミュニケーション

システムの概要や具体的な動作イメージなどをメンバーに伝える時、ＯＨＰのように用いる。また、設計者の意図を理解してもらい、プロジェクトとしての統一を図るためにも使える。

2）プロジェクトメンバー同志のコミュニケーション

プログラマ間のディスカッションや、設計者とプログラマとのデスカッション時に、ＣＯＭＩＣＳの上で容易にソフトウェアを修正・変更し、シナリオを走らせられるので、黒板のように利用できる。

3）プロジェクトメンバーからメンテナンス要員へのコミュニケーション

保守に先立って、システムの理解が必要である。まず、ＣＯＭＩＣＳによりシステムの概要と具体的な動きを把握し、設計者の意図を理解する。その後、仕様書でソフトウェアの詳細を確認し、保守にあたることが効果的である。

ＣＯＭＩＣＳを開発後、その有効性の検証のために、ＣＯＭＩＣＳを試用した。

対象としたのは、開発に６人・年を要した実システムである。このシステムは、まず第一版を開発し、その後改善して第二版の開発が終了している。このシステムの概要と開発過程のデータを、ＣＯＭＩＣＳを初めて操作する者に入力してもらった。そして、以下のシナリオを作成した。

・開発プロジェクトが企画される過程を説明するシナリオ
・システム全体の概要を説明するシナリオ
・一つのモジュールの役割・フローを説明するシナリオ
・システムが第一版から第二版へ更新される過程を説明するシナリオ

すべてのデータの入力には、４人・日を要した。入力データの規模を表２に示す。入力操作そのものよりも、シナリオを考えるのに時間がかかった。シナリオは、伝えたいことを要領良くまとめ、簡潔でわかりやすいストーリィをつくらなければならない。そのため、シナリオを作成し、走らせてみては作り直すという作業を繰り返した。しかし、この試行錯誤によって考えがまとまるという利点もあった。

入力後、このシステムの開発プロジェクトマネージャーとシステム設計者に対し、ＣＯＭＩＣＳがシステムの意図伝達に効果的かどうかを評価してもらった。その結果、以下の評価が得られた。

○満足な点

・設計プロセスやバージョンアップの経過などが表現されており、設計者の意図がプログラマに伝達されると期待できる。
・ソフトウェア開発プロセスの一事例として保存すれば、開発チームから保守チームへの意図伝達や、他のプロジェクトを企画・管理するときの参考資料として利用可能である。

○不満足な点

・わかりやすいシナリオの作成をサポートしてほしい。
・設計者がアイデアを入力し、後から自由に階層化・詳細化できる機能がほしい。

表4．2　COMICSに入力したデータ

シナリオ	使用した構成要素ウィンドウ数	ボックス、ピクチャの総数	シナリオの場面数
①開発プロジェクト企画の過程	3	41	9
②システム全体の概要	4	93	14
③モジュールのフロー	3	46	6
④システム第一版から第二版への過程	4	55	11

このように、COMICSを用いると、比較的短時間でダイナミックな動きや相互の関係が記述でき、しかも設計プロセスの保存などにも有用なことが確かめられた。

4．6　結言

本章では、ソフトウェア開発プロジェクトの生産性向上のための、コミュニケーション支援方法を提案した。

実際の開発プロジェクトに対するインタビュー調査の結果から、メンバー間のコミュニケーションの実情と問題を整理した。そして、ソフトウェアの意図伝達方法として、劇場モデルに基づいた方法を提案し、それをワークステーション上で実現してコミュニケーション支援ツールを開発した。さらに、その有効性についても検証した。今後は、COMICSの有効性についてさらに検討を加えると共に、実用化に際して必要な機能を整備していく予定である。

《参考文献》

1) 岸本美江・西田正吾・後藤卯一郎：ソフトウェア生産プロセスにおけるインターラクションの分析、第35回情報処理全国大会、pp.1141-1142 (1987)

2) 岸本美江・西田正吾・後藤卯一郎：ソフトウェア意図伝達支援ツール　COMI

ＣＳ（１）・（２）、第37回情報処理全国大会、pp.857-861 (1988)
3) 大山　正・東　洋編：認知心理学講座１認知と心理学、東京大学出版会 (1984)
4) ポラニー，Ｍ．：暗黙知の次元、p.146、紀伊國屋書店 (1980)
5) Finzer, W. and Gould, L. : Programming by Rehearsal, Byte, Vol.9, No.6, pp.187-210 (1984)
6) Glinert, E.P. and Tanimoto, S. L. : Pict: An Interactive Graphical Programming Environment, Computer, Vol.17, No.11, pp.7～25, (1984)
7) Marks, P. : System Development as a Performing Art : A People-Centered View of Leonardo, MCC Technical Report Number STP-365-86, (1986)
8) Gibson, J. J. : Observation on Active Touch, Psychological Review, Vol.69, No.6, pp.477-491 (1962)

第５章　大規模プロジェクトにおける
トラブルコミュニケーション支援システムＣＡＣＴＵＳ

５．１　緒言

本章では、百～数百人のメンバーから構成される大規模なプロジェクトにおけるコミュニケーション支援について考える。前章では、ソフトウェアに関するコミュニケーションの認知的な側面を支援する一手法について述べた。ソフトウェアの意図を表現する難しさは、プロジェクトの構成人数にかかわりなく起こる基本的な問題である。これに対して本章では、人数が多いために生じる問題、コミュニケーションのボトルネックの問題について考える。

情報化社会では、システム開発の件数だけでなく、開発の規模も拡大している。しかし、現在のソフトウエア工学では、開発技術の向上が追い付かず、大量の要員を投入した開発作業になっている[1)]。もともと協同開発作業には、第２、３章で述べてきたようなコミュニケーションの難しさ、特にソフトウェアの場合では意図伝達の難しさがあるが、それに加えて、構成メンバーの数が１００人規模になると、次のような新しい問題が生ずる。

１）コミュニケーションが少数のシステム設計者、管理者に集中するため、過負荷となってコミュニケーションの遅れや漏れが発生する危険がある。

２）規模が大きくなると、ソフトウェア、プロジェクト、ともに全体を把握することが困難になり、知識、情報の分散化が進み、適切なコミュニケーション先を判断することが難しくなる。

一般に大規模プロジェクトは、単に人数が多いだけでなく、複数の組織の集合から構成されている。そのため、組織間のコミュニケーションは管理者を介して行なうことが多く、コミュニケーションの経路が複雑になる、管理者にコミュニケーションが集中する、などの問題がある。また、全体を把握できるメンバーが少なく、障害が発生したときの処理が数人の管理者に集中し、管理者は終日電話やＦＡＸの対応に追われる。その結果、情報伝達が遅れたり、漏れたりする危険がある。また、情報が分散しているため、問題解決の際に適切な情報を正確に収集することが難しいという問題も上げられる。

本章では、上記のような問題を踏まえてプロジェクト内コミュニケーションを見

直した結果、何らかのトラブルが発生した際のトラブル処理コミュニケーションに着目する。以下では、トラブル処理コミュニケーションについて定義した後、トラブルコミュニケーションモデルの提案を行ない、モデルに基づいたコミュニケーション支援ツールについて説明する。

5．2　プロジェクトにおけるコミュニケーション

大規模なシステムは、一般にウォーターフォールモデルに従って開発される。ウォーターフォールモデルとは、ソフトウェアの開発プロセスを「要求仕様・システム設計・ソフトウェア設計・コーディング・テスト・保守」という段階に分類したモデルである。プロジェクト管理は、このプロセスに従って工程や進捗を管理する。プロジェクト内で行なわれるコミュニケーションも、このプロセスにしたがって設定される。主なコミュニケーションには、以下のものがある。

①各種仕様書、ドキュメント類
②定期的ミーティング
③進捗報告
④技術レビュー

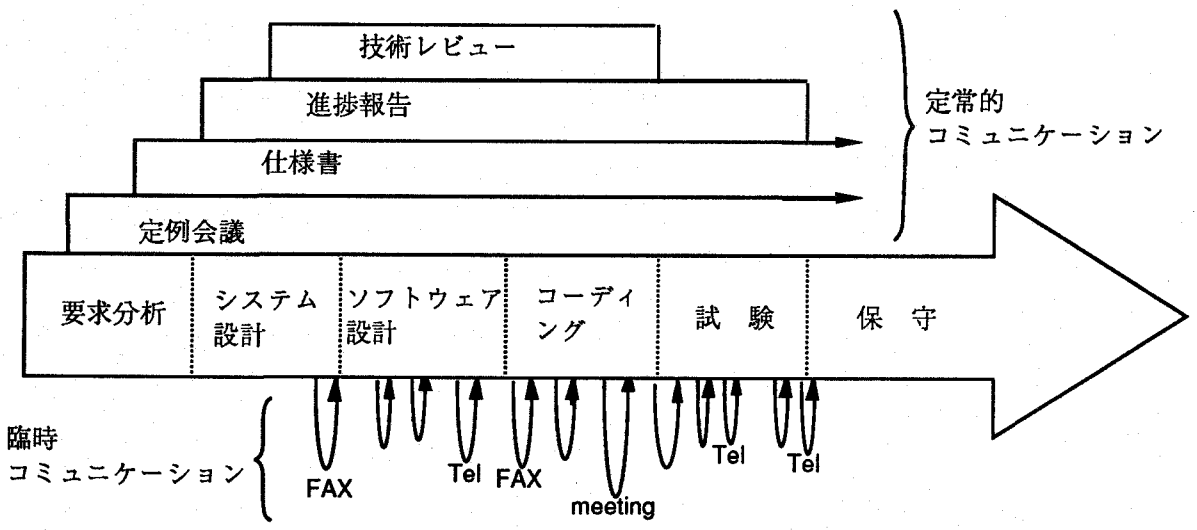


図 5．1　プロジェクト内で発生するコミュニケーション

これらはソフトウェア工学においても研究されており、様々な支援ツールや手法が開発されている。これらのコミュニケーションは十分に標準化され、その手順・内容がプロジェクト開始時から予め決められており、問題の有無にかかわらず遂行される。図5.1はこれらのコミュニケーションを開発プロセスに重ねて示したものである。ここでは、これらを仮に定常的コミュニケーションと呼ぶ。

一方、実際のプロジェクトを観察したところ、上記のコミュニケーションの他に、頻繁な電話、ＦＡＸ、打ち合わせが発生していた。これらのコミュニケーションの内容は多様で、必要に応じて随時行なわれている。例えば、仕様書の曖昧箇所に関する問い合わせは、数分の電話で処理される。客先からの急な仕様変更であれば、緊急会議が開かれる。ここでは、仮にこれらを臨時コミュニケーションと呼ぶ。開発作業中に何のトラブルもなく順調に進んでいれば、定常的コミュニケーションだけで十分な情報伝達ができるはずである。しかし、実際には予期せぬ大小様々なトラブルが発生しており、その度にトラブルに応じた臨時コミュニケーションが行なわれている。トラブル処理は、記録に残らないような些細なコミュニケーションから、緊急ミーティングを何度も繰り返すような大きなものまで多様であり、定常的なコミュニケーションの枠にはおさまらない。

ところが、臨時のコミュニケーションはその内容も方法も時期も不特定なため、これまで体系的な支援は行なわれていなかった。臨時コミュニケーションは人のネットワークで処理するものと考えられ、支援は、ツールやメディアの提供（ＦＡＸ、電子会議システム等）に限られていた。本研究では、以下のような理由から臨時コミュニケーションに着目し、支援方法を提案する。

（1）プロジェクトにとって、本当に重要な情報、緊急を要するものは臨時コミュニケーションで伝達されると考えられる。
（2）臨時のコミュニケーションは数が多い。特にプロジェクトのキーパーソンの所に集中し、処理の遅れや漏れの危険があり、支援の必要がある。
（3）プロジェクトが大規模になると、知識が分散される。プロジェクト全体、システム全体を把握するメンバーがいなくなり、メンバー同士の関係も疎になり、人のネットワークで問題解決できる限界を越えてしまう。

組織的な協同開発作業であれば、臨時コミュニケーションがまったくランダムな構造を持つものとは考えられない。トラブル処理にもなんらかの枠組みがあるはず

である。臨時コミュニケーションの根底にある規則性が明らかになれば、コミュニケーション支援が可能になる。次章では、臨時コミュニケーションのモデル化に先立って、ここで取り扱うコミュニケーションを明らかにする。

5．3　トラブルコミュニケーションの分析

プロジェクトの臨時コミュニケーションには様々な内容・形態があり、モデル化するためには何らかの視点、枠組みが必要である。本章では、新しい枠組みの提案と、それに基づいた分析を行なう。

5．3．1　トラブルコミュニケーションの定義

臨時コミュニケーションを、協同作業を進める上で他のメンバーと情報伝達する必要が生じたときに発生するコミュニケーションという枠組みで捉え、それをトラブルコミュニケーションと呼んで、以下のように定義する（図5.2）。

（1）第一に、プロジェクトメンバーが問題を認識していない状態をコンセンサス状態と呼ぶ。第二に、コミュニケーションを発生させるようなトラブルをトリガと呼ぶ。あるメンバーがトラブルを認識し、一人では処理できず、他のメンバーに連絡・相談する必要があると判断したとき、コンセンサス状態が崩れる。そしてそのトラブルがトリガとなってコミュニケーションが始まる。コミュニケーションによってトラブルが解決されると、プロジェクトは新たなコンセンサス状態に入り、コミュ

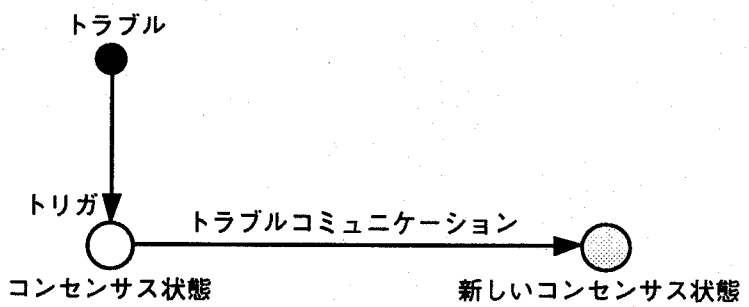


図5．2　トラブルコミュニケーションの定義

ニケーションは終了する。この、トラブルが発生してから解決するまでの一連のコミュニケーションをトラブルコミュニケーションと呼ぶ。

（2）トラブルを解決するためには、関係者への連絡・確認、会議、情報収集などのコミュニケーションが必要である。その内容も手段もまちまちであるが、トラブル解決のためのコミュニケーションはすべてトラブルコミュニケーションとする。このときのコミュニケーションが定常的なものか臨時かは問わない。定常的コミュニケーションと臨時コミュニケーションが並行、またはどちらか一方だけで処理される場合もある。

（3）一つのトラブルコミュニケーションは一つのトラブルを処理するための最小ユニットである。実際には複数のトラブルが同時に発生し、トラブルコミュニケーションはいくつも並行して起こるため、プロジェクトが完全なコンセンサス状態に入ることはないように見える。

5．3．2　トラブルコミュニケーションのプロセス

次に、トラブルコミュニケーションの開始から終了までのプロセスを分析する。まず実際に調査したプロジェクトのトラブルから、一例をあげる。

1）画面表示スピードに関して、客先要求仕様を満たしていないというトラブルが発生した。
2）試験担当者は、統合試験でのシステムの動作に要求仕様を満足していない箇所を発見する。
3）試験担当者が再試験をしてトラブルを確認し、試験チームの責任者に問題を連絡する。
4）試験チームの責任者は、画面表示速度を改良すればよいと判断し、画面表示モジュールの開発チームの責任者に連絡する。
5）開発チームの責任者はトラブルを分析し、表示速度の問題ではなく、他モジュールからデータ転送する方式に問題があると判断する。
6）関連モジュールの開発責任者とプロジェクトマネージャが集まり、関連仕様書とテストデータなどをもとに解決策を検討する。その結果、ディスプレイを高速のものに変更するか、プログラム言語を変更して処理速度を上げるかの２つの選択肢にまとまる。
7）納期、予算などの条件を考慮にいれ、プロジェクトマネージャが最終決定を下す。

8）関係者へ変更案が連絡され、実行に移される。
9）トラブル発見から処理までの経緯は、障害報告書としてマネージャから品質管理部門へ送られ、保存される。

この問題解決プロセスには、協同作業における問題解決の二つの特徴がある。一つは、問題の発見者、解決者、解決策の実行者が、それぞれ異なるという点である。単独作業の場合、問題の発見も解決も単独で行なわれるので、コミュニケーションの必要はない。しかし、一つのシステムを複数モジュールに分割して複数で開発する場合、モジュール単独ではエラーが出なくても組み合わせると動かない、担当者の判断で仕様を変更できないなど、単独作業にはない問題が発生する。それを解決するためにコミュニケーションが必要になる。トラブルの発見から解決までには、複数のメンバーが関わっているが、特に3種類のキーパーソンがあげられる。

①発見者：トラブルを発見するメンバー
②解決者：解決のための最終決定をするメンバー
③実行者：解決策にしたがって、具体的な解決処理を担当するメンバー

上記の実例では、発見者は試験担当者、解決者はプロジェクトマネージャで、関係者との合議で解決策を決定し、実行者は各モジュールの担当者、になっていた。この例ではトラブルが複数のモジュールに関連した広範囲なものであったため、多数のメンバーが関わった。もしトラブルの規模が小さく一つのモジュール内で閉じている場合にも、複数のメンバー間でコミュニケーションが必要になることもある。ソフトウェアのライフサイクルには、製作、試験、保守という複数の開発フェーズがあり、プロセスの進行にしたがって担当者が変わり、管理の方法も変わる。そのため、トラブルの発見が遅れるほど、より多くのメンバーが関わってくる。例えば、試験担当者から開発担当者へ、保守担当者から開発担当者へ、保守担当者から品質管理部門へ、などのコミュニケーションが発生する。

しかし、これは一般的な特徴であり、すべてのトラブルについて発見者、解決者、担当者の三者が別になるわけではない。トラブルの規模が非常に小さいものであれば、発見、解決、担当が一人のメンバーで閉じてしまい、コミュニケーションが行なわれない場合もある。情報収集のためのコミュニケーションだけが行なわれる場合もある。

もう一つの特徴は、問題解決プロセスに上記のような三者が関わってくるとして、

それぞれのコミュニケーション先をどうやって決定しているのか、という点である。発見者は解決者をどうやって判断し、解決者は実行者を何により判断しているのか。観察によれば、トラブルの起こったモジュールの開発担当者がその解決策を決定するとは限らない。プロジェクトでは、役割分担が明確になっている。多くの場合、組織は管理者、設計者、プログラマ、という階層構造になっていて、それぞれの権限範囲が決められている。プログラマは仕様の範囲内では自由なプログラミングができるが、仕様を変更する権限はない。システム設計者も、自由な裁量の範囲は限られていて、それ以上は合議、又はマネージャに意志決定を委ねなければならない。トラブルの解決策を決定する「権限」を持っているかどうかは、組織における意志決定に際し、重要なポイントになる。そして解決策が決まれば、その実行は該当モジュールの担当者が行なう。「意志決定の権限をもつこと」と「実際の処理を担当すること」は別のことである。

また、情報収集も適切な意志決定にとって重要なコミュニケーションである。上記の例では、関連モジュールの責任者らが集まり、各種データを用いて検討の上、２つの解決案を提案した。解決者であるマネージャは、その他の条件も考慮して最終決定している。解決策の決定には、どの情報を集めるかが影響する。特に大規模なプロジェクトでは、情報が分散しているため、必要な情報がどこにあるか、誰に相談するべきかを知っていることが重要になる。

最後に、プロジェクト特有のコミュニケーションとして問題解決後の報告コミュニケーションがあげられる。多くの場合、開発プロセス中に発生したトラブルは、客先への提出資料、又は今後のプロジェクト管理用資料として保存される。保存方法は様々であり、記録すべきトラブルの種類、内容、保管場所、などはプロジェクトによって決められている。例えば、プログラムのバグは、パッケージ製作中に発見・処理されれば記録に残らない。が、テスト要員によって発見されれば、障害連絡シートとして発見から解決までの経緯が記録され、保存される。

以上の分析より、ソフトウェア開発プロジェクトにおけるトラブルコミュニケーションプロセスには次の２点の特徴があると言える。次に、このコミュニケーションプロセスをシミュレートするモデルを提案する。

・ソフトウェア開発プロジェクトにおける問題解決プロセスには、発見者、解決者、実行者という複数のメンバーが関わり、これらのメンバーの間でコミュニ

ケーションが行なわれる。

・上記のコミュニケーションを決定するには、「意志決定をする権限の所有者に関する情報」「トラブル箇所ついての知識の所有者に関する情報」「トラブル報告に関する情報」の各要因が重要である。

5．4　トラブルコミュニケーションモデル

組織的な作業におけるトラブルコミュニケーションは、問題の発見者、解決者、実行者、の３者間のコミュニケーションプロセスであると言える。あるコンセンサス状態から新しいコンセンサス状態までは次のようなプロセスをたどる。

1）トラブル発生

2）トラブルの発見者は、問題のある箇所に対して意志決定の権限を持つメンバーを検索し、そのメンバーを解決者と判断してトラブルを連絡する。

3）解決者は、問題の箇所に関連する箇所を判断し、それらの箇所に関する情報を持っているメンバーをすべて検索し、情報収集する。

4）解決者は集めた情報に基づいて解決策を決定。

5）解決者は、解決すべき箇所とそれに関連する箇所を判断し、それらの箇所を担当するメンバーを検索し、解決策の処理を依頼する。

6）解決者は、トラブルの発見から解決までの処理が報告義務があるかどうかを判断し、報告義務があればその報告先を判断して、報告する。

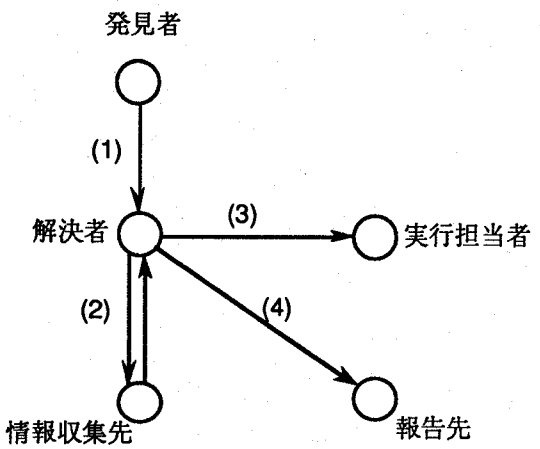


図５．３　トラブルコミュニケーションモデル

ここで提案するトラブルコミュニケーションモデルは、このプロセスにおけるコミュニケーションを4段階で表現したものである（図5.3）。

［1］発見者から解決者へのコミュニケーション

［2］情報収集コミュニケーション

［3］解決者から実行者へのコミュニケーション

［4］報告コミュニケーション

各段階には一つずつコミュニケーション先を判断するためのメカニズムが必要である。判断には以下の6種類のデータを用いる。

（a）プロジェクトのメンバー構成

（b）システムのソフトウェア構成

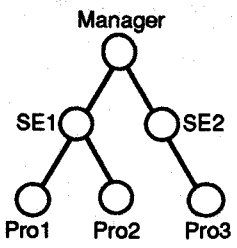
（c）各メンバーの担当範囲

（d）各メンバーの権限範囲

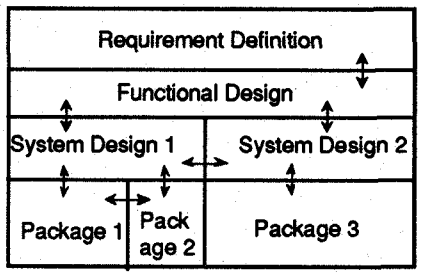
（e）各メンバーの知識範囲

（f）トラブル処理分類表

図5.4は小規模なプロジェクトの例である。この図を用いてデータを説明する。（a）はプロジェクトのメンバー構造を示したものである。一般にツリー型の階層構造になっている。このデータはメンバー間の関係とメンバーの階級情報を提供する。［4］の報告コミュニケーションにおいて、報告先が関係や階級で指定された場合に、このデータよりコミュニケーション先を検索する。（b）はシステムのソフトウェア構造である。ソフトウェアをモジュール分割したときの構造と、各モジュールの関係を示す。［2］［3］のコミュニケーションにおいて、トラブルの発生したモジュールと関係のあるモジュールを検索するときに用いる。（c）は各メンバーの担当モジュールの範囲を示したものである。（b）のソフトウェア構成データに（a）のメンバー構成データを対応させて、各モジュールの担当者を表示している。［3］のコミュニケーションにおいて、解決策の実行を依頼する担当者を検索する時に用いる。（d）は各メンバーの持つ権限の範囲を示したものである。（b）のソフトウェア構成データに（a）のメンバー構成データを対応させて、各モジュールに対する権限所有者を表示している。［1］のコミュニケーションにおいて、解決者を検索する時に用いる。（e）は各メンバーの持つ知識の範囲を示したものである。（b）のソフトウェア構成データに（a）のメンバー構成データを対応させ

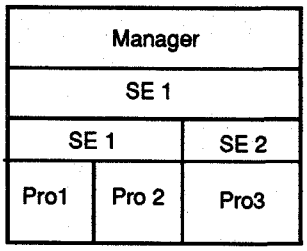


（a）プロジェクト構成

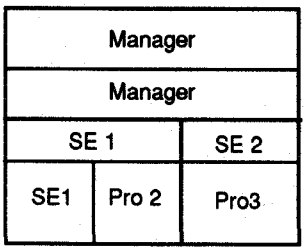


↔矢印は、モジュール間の関係を表す

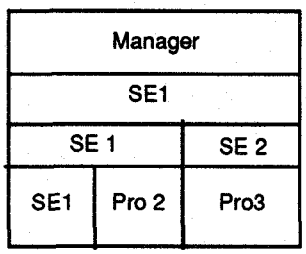
（b）ソフトウェア構成



（c）担当範囲



（d）権限範囲



（e）知識範囲

トラブルの種類	トラブルの発生条件		
	発見された時期	情報収集する範囲	トラブルを報告する範囲
タイプ1：仕様不良	単体試験まで	全関連モジュール	一階層上位の管理者
	統合試験開始後	全関連モジュール	一階層上位の管理者とプロジェクトマネージャ
タイプ2：仕様変更	単体試験まで	全関連モジュール	一階層上位の管理者
	統合試験開始後	全関連モジュール	一階層上位の管理者とプロジェクトマネージャ
タイプ3：仕様の解釈ミス コーディングミス	単体試験まで	必要なし	必要なし
	統合試験開始後	全関連モジュール	一階層上位の管理者とプロジェクトマネージャ

（f）トラブル処理分類表

```
[ System structure]
  Structure List = { :
                  (System Design 1
                     relatedD: Functional Design,
                               System Design 2,
                               Package 1, Package 2 )
                  :
                  }

[ Project organization]
  Member List = { :
                  (SE1  upM: Manager,
                        sameM: SE2,
                        lowM: Pro1, Pro2 )
                  :
                  }
```

```
[ "charge" map ]
  Member List = { :
                  (SE1  Char: Functional Design,
                              System Design 1 )
                  :
                  }

[ "competence" map ]
  Member List = { :
                  (SE1  Com: System Design 1,
                             Package 1 )
                  :
                  }

[ "knowledge" map ]
  Member List = { :
                  (SE1  Know: Functional Design,
                              System Design 1,
                              Package 1 )
                  :
                  }
```

図５．４　サンプルプロジェクトのデータ構造

て、各モジュールに対する情報所有者を表示している。［2］のコミュニケーションにおいて、情報収集先を検索する時に用いる。（f）はトラブルの種類とそれに対応する報告などの手続きを示したもので、［4］の報告コミュニケーション先を判断するときに用いる。

これらのデータ構造は基本的な枠組みであり、プロジェクトに合わせて変形できる。例えば、担当をチーム単位でおおまかに捉えようと思えば、メンバー構成データをチーム名で示し、ソフトウェア構成データもそれに対応させた概略のモジュール分割にできる。詳細なデータが必要であれば、メンバーを個人単位で記述し、モジュール分割も詳細に記述すればよい。ソフトウェア構成とモジュール間の関係をより詳細に記述すれば、情報収集先を検索するときに無駄のないデータを得ることができる。（f）のデータはプロジェクトによってその記述形式も内容も様々であり、図 5.4 の分類は一例である。

このデータ例を用いて、モデルの動作を説明する。今、図 5.4 （b）の Package1 にトラブルが発生したときのシミュレーション結果が 図 5.5 である。

［1］ Package1 の担当者である Pro1 がトラブルを発見するが、データ（d）により Pro1 には意志決定の権限がないので、権限を持つ SE1 を解決者と判断して連絡する。

［2］ SE1 は情報収集のために Package1 に関係のあるモジュールをデータ（b）より検索したところ、 Package2 、System Design1 であった。そこでそれぞれのモ

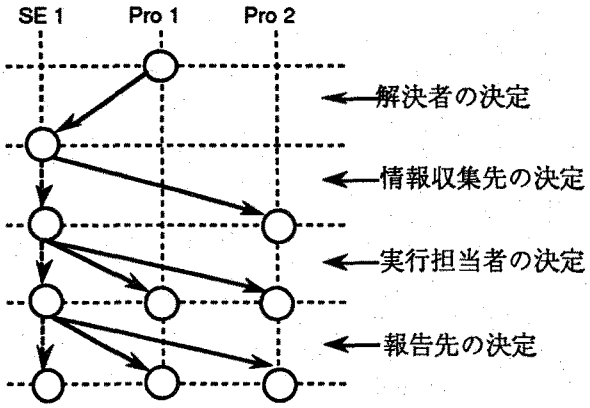


図５．５　シミュレーション結果

ジュールに関して情報を持つものをデータ（e）より検索し、SE1、Pro2から情報を収集する。

［3］SE1は集めた情報をもとに解決策を決定し、Package1とこれに関係のあるモジュールPackage2、System Design1の担当者をデータ（c）より検索し、SE1、Pro1、Pro2に解決策の実行を依頼する。

［4］SE1はトラブルの経過と処理の終了を、データ（f）より関係者に報告する。

次に、データを変更した場合のコミュニケーションプロセスの変化を見る。今、Pacakge1に関する権限と情報の所有者をSE1からPro1に変更する（図5.6）。上記

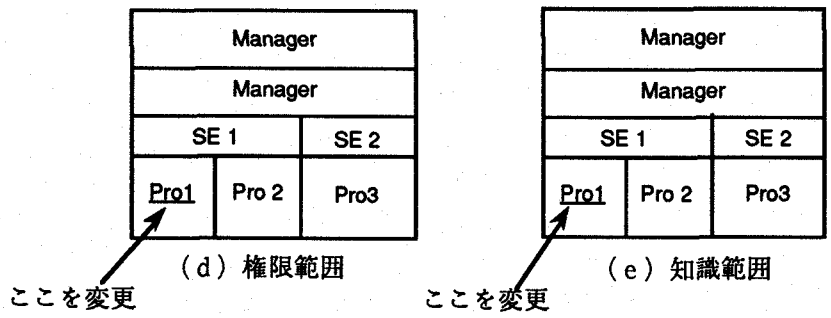


図５．６　　図５．４の一部を変更したデータ

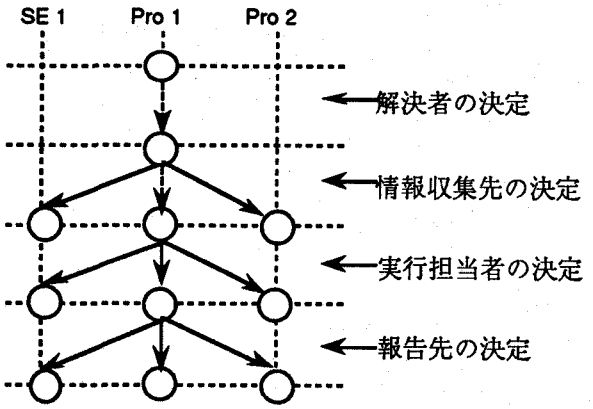


図５．７　データ変更後のシミュレーション結果

の例と同じトラブルが発生したものとしてシミュレーションした結果が図5.7である。解決者が変わり、コミュニケーションの方向が逆になっている。この場合、トラブルの発見から解決、処理を一貫してPro1が担当し、情報収集と解決策実行の依頼、報告のコミュニケーションだけが行なわれている。

本モデルは、トラブル解決プロセスを4段階のコミュニケーションで記述しているが、実際には4段階で順調に解決するケースは少ない。例えば、図5.4のプロジェクトにおいて、見かけのトラブルがPackage1に発生したため、発見者のPro1はSE1が解決者であると判断して連絡した。しかし、SE1がトラブルの原因はPackage2にあると判断した場合、SE1は真の解決者であるPro2に連絡する。この時、［1］の発見者から解決者へのコミュニケーションが2回行なわれることになる。コミュニケーションプロセスは［1］→［1］→［2］→［3］→［4］の5段階になる。また、一度決定した解決策が実行不可能であった場合、実行担当者から解決者へ解決策の変更が依頼されることもある。このときのコミュニケーションプロセスは［1］→［2］→［3］→［1］→［2］→［3］→［4］となる。

このように、本モデルの4段階のコミュニケーションは基本的なものであり、組み合わせによって多様なプロセス表現が可能である。

5．5　プロジェクトコミュニケーション支援システムＣＡＣＴＵＳ

5．5．1　ＣＡＣＴＵＳの概要と機能

トラブルコミュニケーションモデルに基づいて、プロジェクトにおける問題解決コミュニケーションをシミュレートするツールＣＡＣＴＵＳ（Computer Assisted Communication Tool for Urgent Support）を開発した。ＣＡＣＴＵＳの目的は、大規模なプロジェクトにトラブルが発生したとき、コミュニケーションが正確、迅速に行なわれるようにコミュニケーション先を示唆することである。上述のように、数百人規模のプロジェクトでは、情報の分散化とコミュニケーションの少数メンバーへの集中という問題がある。ＣＡＣＴＵＳは以下の点からこの問題を支援する。

・トラブルの解決策を決める際に参考にすべきすべての情報の所在を網羅して表示し、情報収集の漏れを防ぐ。
・トラブルの解決者、実行担当者を示唆して、適切な解決者を探し回るようなコミュニケーションの無駄と遅れを防ぐ。

・上述の支援により、コミュニケーション処理に追われる管理者のコストを軽減し、コミュニケーションの漏れや遅れを防ぐ。

本システムはトラブルコミュニケーションモデルを実現したもので、以下の特徴を持つ。

①コミュニケーションプロセスを4段階に分け、各段階にコミュニケーション先決定メカニズムを持つ。

②各決定メカニズムは、上述した6種類のデータベースからコミュニケーション先を算出する。

③トラブルが入力されると、解決者リスト、関連場所リスト、実行者リスト、報告先リストが順次表示される。

プロジェクトのメンバー構成やシステムの構成に関するデータは、あらかじめデータファイルとして準備しておかなければならない。本システムは、以下の機能を持つ。

1）メニュー操作で 以下の入出力を管理する。
- ・データファイルの読み込みと変更
- ・6種類のデータベースの表示
- ・トラブルの発生したモジュール名、発見者、トラブルの内容を入力
- ・下記2）～5）の計算結果をシミュレート結果として表示

2）トラブルモジュールと（d）権限範囲データから、トラブルモジュールに対して権限を所有するメンバーを検索し、解決者リストを作成する。

3）トラブルモジュールと（b）ソフトウェア構成データから、トラブルの発生したモジュールと関連のあるモジュールを検索し、関連モジュールリストを作成する。関連モジュールリストと（e）情報範囲データから、情報を所有するメンバーを検索し、情報収集先リストを作成する。

4）先に作成した関連モジュールリストと（c）担当範囲データから、担当するメンバーを検索し、実行担当者リストを作成する。

5）トラブルの内容と（f）トラブル処理分類表と（a）プロジェクト構成データから、報告義務が指定されているメンバーを検索し、報告先リストを作成する。

本システムの利用は、準備と使用の2段階に別れる。まず6種類のデータをテキ

ストエディタで入力し、データファイルを作成する。次にシステム上でトラブルに関するデータを入力し、コミュニケーションプロセスをシミュレートさせる。また、システム上でデータを変更し、それによるコミュニケーションプロセスの変化を見ることもできる。

5．5．2　ソフトウェア構成

ＣＡＣＴＵＳはＣ言語によりワークステーション上で実現した。システム構成は以下のとおりである。データファイルの新規作成は一般のテキストエディタで行なうが、データの変更はシステム上で行なえる。

1）インタフェース：メニュー管理、入出力管理
2）プロセス計算：入力されたトラブルとデータよりコミュニケーションルートを計算
3）データ処理：データファイル管理
4）エディタ：データ修正管理

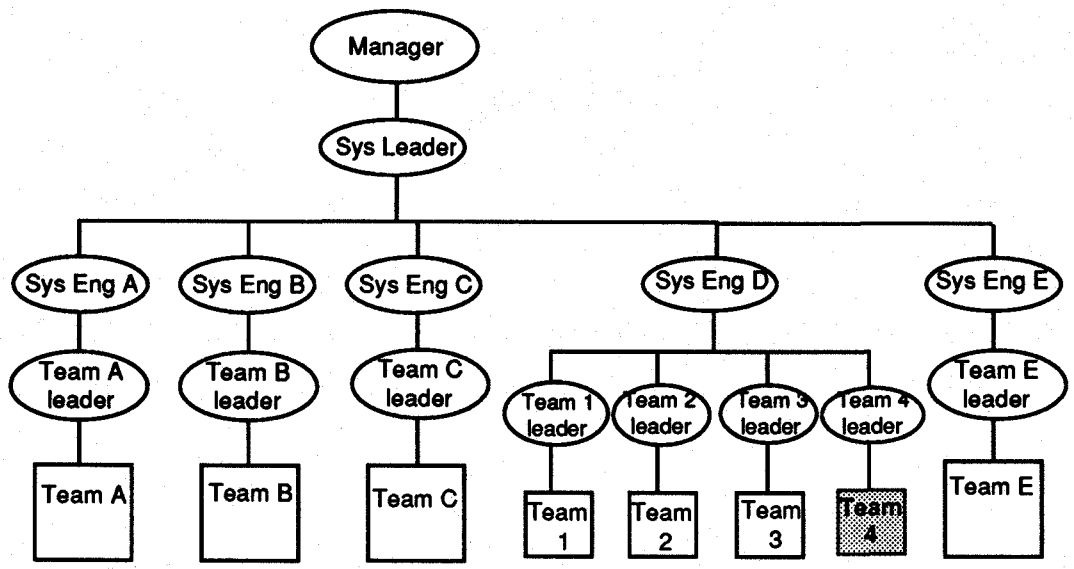
5．5．3　ＣＡＣＴＵＳの動作例

本システムによるシミュレート結果を示し、実際のプロジェクトにおけるコミュニケーション記録の結果と比較する。

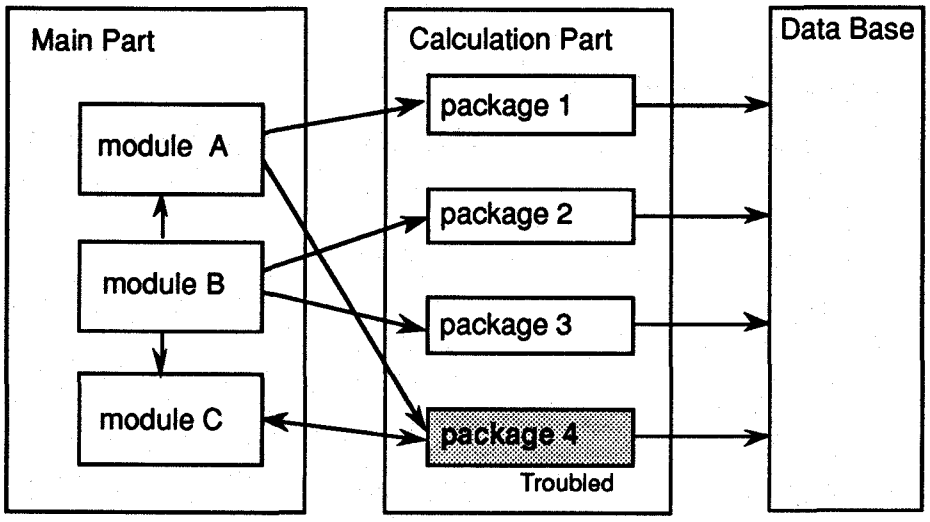
比較に用いるプロジェクトとソフトウェアの構成を図 5.8 に示す。6 種類のデータベースはここでは割愛する。今、図 5.8 の（b）でアミカケ表示した Package4 に、客先からの仕様変更が発生した。担当しているのは（a）でアミカケ表示した Team4 である。発見者を Manager としてシミュレートした結果を図 5.9 に示す。［1］で解決者を算出して Team4 Leader へトラブルを連絡する。［2］で関連場所 moduleA、moduleC、DataBase を算出し、各々の情報所有者である TeamA Leader、TeamC Leader、TeamE Leader から情報を収集する。［3］で関連場所の担当者 Team4、TeamA、TeamC、TeamE に解決策の実行依頼をし、［4］で Manager に報告する。

図 5.10 は記録されていた実際のコミュニケーション履歴である。図 5.9 との比較点をまとめる。

（1）発見者、解決者、報告先は同じである。
（2）情報収集先と実行担当者が、実際は２人だけなのに対し、シミュレーションでは３人になっている。
（3）シミュレーションでは各役割間のコミュニケーションが直接行なわれている



(a) Project organization



(b) System structure

網かけのモジュール Package4 に
仕様変更が発生した。
担当者は Team4 である。

図５．８　プロジェクト構成とソフトウェア構成

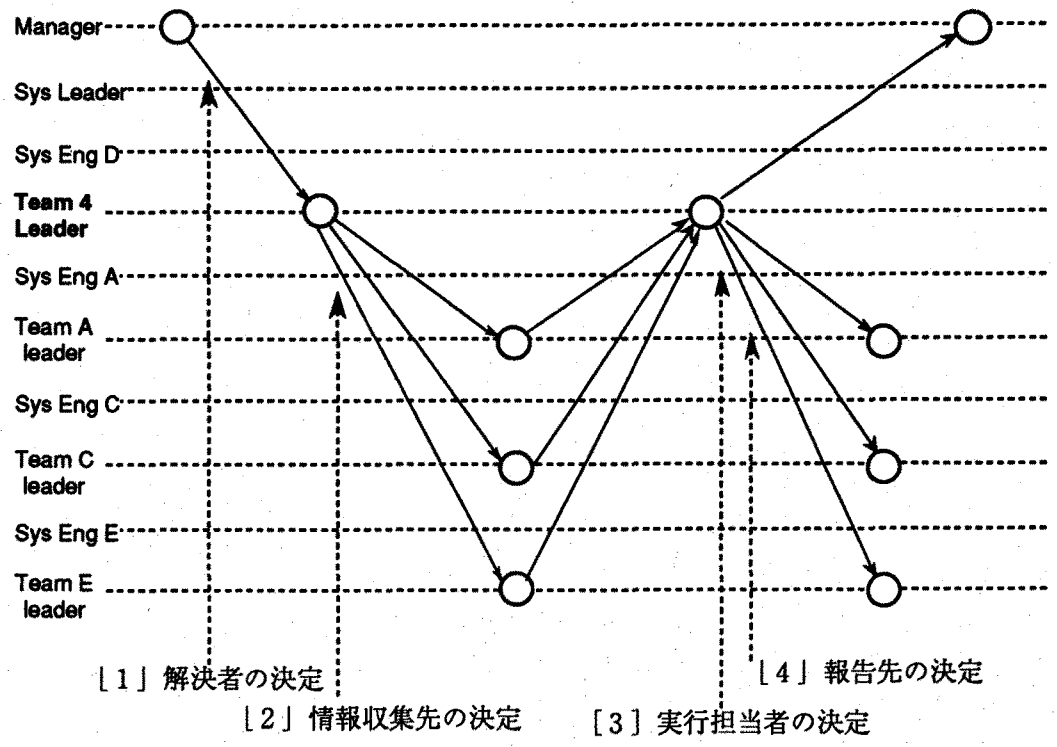


図５．９　　図５．８のプロジェクトにおけるシミュレーション結果

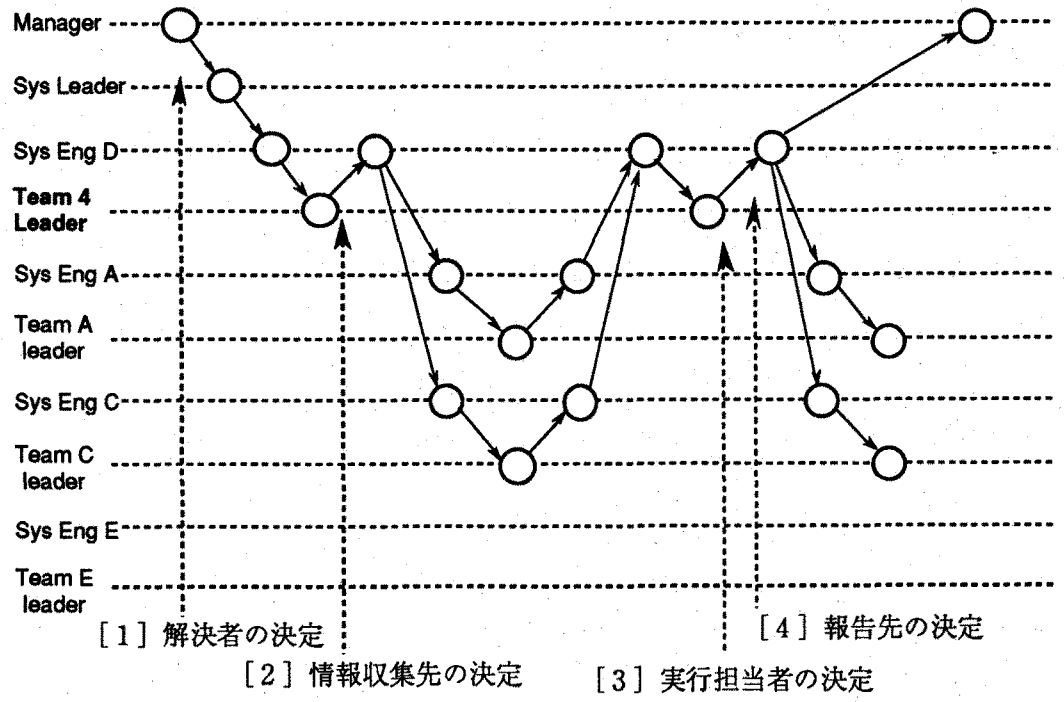


図５．１０　　図５．８のプロジェクトにおける実際のコミュニケーションプロセス

のに対し、実際には途中で他のメンバーが介在している。

（1）より、モデルの計算が妥当なことがわかる。（2）で見られた違いの理由は、本モデルでは、関連する場所すべてから情報を収集する設定になっているため、関連のある3箇所から情報を集めている。しかし、実際には、関連していても解決者が必要ないと判断した場所からは情報収集されない。この相違に関しては、二通りの評価ができる。一つは、解決者の判断が間違っている可能性もあるので、関連があるすべての場所から情報収集すべきであると考えると、すべての関連場所を網羅する本モデルが有効である。もう一つは、トラブルの内容によって関連の内容も変わってくるのだから、常にすべての関連場所から情報を得る必要はないと考えると、本モデルのデータ記述をさらに詳細なレベルまで分類し、トラブル内容別の関連データを用意する必要がある。ここでは前者の立場に立ち、モデルとしてはすべての関連場所を提示し、実際に情報収集が必要かどうかは、本システムを使用するメンバー自身が判断するものとする。（3）で見られた違いの理由は、プロジェクトのメンバー構成に起因する。ここで例としてあげたプロジェクトの各チームは、外部のソフトウェアハウスで構成される。ソフトウェアハウス間での連絡は、各チームリーダと担当システムエンジニアとを介して行なわれる。従って、発見者から解決者、解決者による情報収集などのコミュニケーションもすべてシステムエンジニアを介して行なわれている。ここで介在するメンバーはトラブル処理プロセスでは何の役割も持たないため、本モデルではあえて記述していない。しかし、データ記述を拡張することにより介在者のプロセスをシミュレートすることも可能である。プロジェクトメンバー構成データには、メンバー間の関係とメンバーの階級情報が記述されているが、ここに直接コミュニケーションできるメンバーの情報ユニットを追加し、システムにはコミュニケーション可能な相手を検索する機能を追加する。例えば、ソフトウェアハウスのメンバーはその担当システムエンジニアとしかコミュニケーションできないとし、システムエンジニア同士は直接コミュニケーションできると記述する。するとチーム間で連絡が必要になったとき、互いの担当システムエンジニアを介したコミュニケーションプロセスが表現できる。

以上のような相違点は見られたが、トラブル処理プロセスに関わる発見者、解決者、情報収集先、実行担当者、報告先、はモデルによりシミュレートされていると考えられる。

５．６　ＣＡＣＴＵＳの利用

ＣＡＣＴＵＳはソフトウェア開発プロセスにおいて、On-line で使用する場合と、Off-line で使用する場合が考えられる。具体的には、以下の利用例がある。

１）コミュニケーション先のアドバイス（On-line の使用）

開発プロセス中にトラブルが発生したとき、本システムによりコミュニケーション先をアドバイスすることができる。トラブルの発見者にとっては解決者を、解決者にとっては情報収集先や実行担当者が確認でき、コミュニケーションの漏れや遅れを防ぐことができる。また、客先の要請などで仕様変更の検討が必要になった場合、どの範囲まで影響が及ぶか、変更に際してどこまでの承認が必要か、などを判断する資料になる。

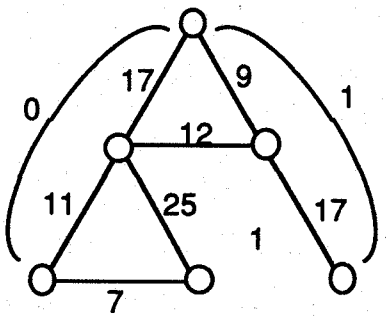
２）コミュニケーションの問題点の分析（Off-line の使用）

プロジェクトでは、権限や知識が管理者や少数のシステムエンジニアに集中するため、コミュニケーションも特定のメンバーに集中してしまう。本システムにより、様々な内容・場所のトラブルをランダムに発生させた場合のシミュレート結果を集計すれば、コミュニケーション頻度の分布がわかる。極端に集中しているメンバーには問題が起きやすいなど、分析資料になる。

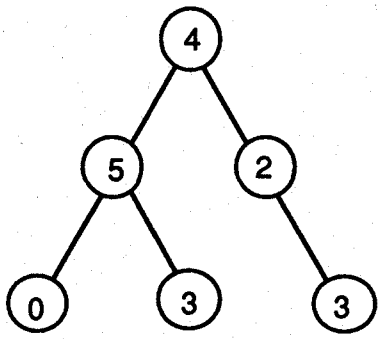
３）コミュニケーションプロセスの予測（Off-line の使用）

プロジェクトのメンバー構成、分担などを決めるときの参考資料になる。ランダムなトラブルを発生させてシミュレートし、特定のメンバーにコミュニケーションが集中した場合、権限や情報の一部を他のメンバーに分散させるだけでコミュニケーション分布が変化する様子がわかる。図 5.11 の（ａ）は図 5.4 のプロジェクトデータを用いたシミュレーション結果である。丸の中の数字は、解決者となって意志決定した回数で、線上の数字はその場所にコミュニケーションが発生した回数である。SE1 の意志決定回数が多いことがわかる。今、権限を図 5.5 のように変更したとき、同じトラブルを発生させたときのシミュレーション結果は図 5.11 の（ｂ）のようになった。意志決定の数が各メンバーで平均化され、コミュニケーション発生の分布も変化したことがわかる。

この他にも、新人にプロジェクトの構成や情報の分布を学習させるのに用いることもできる。また、メンバーによって異なる権限範囲データや情報範囲データを持っていることもあり、様々なメンバーが本システムを使用すれば、各自の持つデータの違いを認識することにもなる。

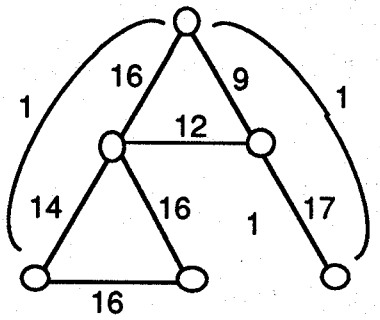


コミュニケーション頻度の分布

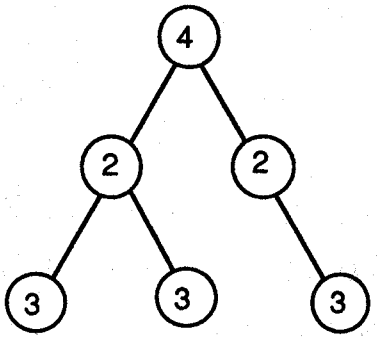


解決者の分布

（a）図5.4の権限範囲データに基づいたときのコミュニケーション分布



コミュニケーション頻度の分布



解決者の分布

（b）図5.6の権限範囲データに基づいたときのコミュニケーション分布

図５．１１　データを変更したときのコミュニケーション分布の変化

５．７　結言

本章では、協同作業の中でも問題解決型のコミュニケーションに着目し、そのプロセスをトラブルコミュニケーションモデルとしてまとめた。そのモデルをもとに、コミュニケーション支援システムを開発した。このシステムは、プロジェクトメンバーの知識、権限、担当内容などのデータを基に問題発生時のコミュニケーション先をアドバイスし、コミュニケーションの遅れや漏れの防止を支援する。今後、大量データ入力の問題と、システムの評価が残されている。

《参考文献》

1) 亀田靖浩：大規模ソフトウェア開発の現状と課題、電子情報通信学会誌、Vol.74、No.5、 pp.457-460（1991）

2) 海谷治彦・佐伯元司：ソフトウェアの仕様化過程における協調作業のモデル化、情報処理学会研究報告、91 - HI - 37、 PP. - （1991）

3) 福田由紀雄・井上敦子・津田淳一郎：遠隔地ソフトウェア開発の実験、情報処理学会研究報告、 90 - SE - 71、pp.49-56（1990）

4) Begeman, M., Cook, P., Ellis, C., Graf, M., Rein, G. and Smith, T. : Project NICK: Meetings Augmentation and Analysis, Proceedings of CSCW'86, pp. 1-6 (1986)

5) Bendifallah, S., and Scacchi, W. : Work Structures and Shifts: An Empirical Analysis of Software Specification Teamwaork, 11th International Conference on Software Engineering, Pittsburgh, pp. 260-270 (1989)

6) 仲谷美江・西田正吾：ソフトウェア開発プロジェクトにおけるトラブルコミュニケーションモデル――ソフトウェア開発プロジェクトにおける問題解決プロセス、第7回ヒューマンインタフェースシンポジウム、 pp.379-384 (1991)

7) Nakatani, M. and Nishida, S. : Trouble Communication Model in a Software Development Project, 電子情報通信学会英論文誌, Vol.E75-A, No.2, 196-206 (1992)

第６章　ソフトウェア開発プロジェクトにおけるコミュニケーションの評価手法

６．１　緒言

第４章、第５章ではコミュニケーション支援のためのモデルを提案したが、本章ではソフトウェア開発プロジェクトにおけるコミュニケーションの状態を評価する手法を提案する。

複数の人が一緒に仕事をするとき、情報を交換するためのコミュニケーションは欠かせない。しかし、コミュニケーションの役割は情報交換だけではない。加藤（1986）[1)]によれば、コミュニケーションには「伝達」と「交わり」の側面がある。「伝達」とは情報を人に伝えることを目的とし、コミュニケーションをその道具と考える側面である。「交わり」とは、コミュニケーションという行為自体を目的とする側面である。一つのコミュニケーションには道具的機能と目的的機能との両側面が存在し、二つの機能を切り離して考えることはできない。

従来のコミュニケーション研究はどちらかの側面だけに焦点を当てたものが中心であった。組織におけるコミュニケーションについて言えば、生産効率を上げるため、情報を正確、迅速に伝達することを最優先していた。近年、ＣＳＣＷ（Computer Supported Cooperative Work）など協同作業支援研究では、交わり的なコミュニケーション、いわゆるインフォーマルコミュニケーションの重要性が認識され始めている。しかし、組織におけるインフォーマルコミュニケーションの研究は非常に少ない。評価方法も確立していないため、コミュニケーションの問題を認識することも困難である。

以下では、コミュニケーションを「交わり」「伝達」の２側面でとらえ、協同作業に必要なコミュニケーションを５つに分類する枠組みを提案する。これに基づいて質問紙を作成し、「各メンバーによる主観的評価→集計→結果の視覚化→プロジェクト診断」という一連の手続きを開発した。これによってコミュニケーションの問題を、個人間のミクロな視点とプロジェクト全体のマクロな視点で理解することができる。特にマクロな評価の視覚化に顔の表情を用いたことで、コミュニケーション量や人間関係の雰囲気を直観的に表現することができた。この評価方法により、これまで敬遠されがちだったインフォーマルな部分も扱いやすくなり、プロジェクト間の比較も可能になって、プロジェクト管理の有効な資料となる。２つのプロジェクトに実施し、その妥当性を確認した。

6．2コミュニケーション評価研究の意義

コミュニケーションは古くから多様な分野で研究されてきた。その内容は、人間のコミュニケーション行動の分析から産業界への応用まで多岐に渡る。ここでは協同作業におけるコミュニケーションの理論、支援技術、現場の分析、という視点から従来の研究を分類した（図 6.1）。

まず理論的な研究は、主に社会心理学で行なわれている。社会心理学で集団のパフォーマンスが研究されるようになったのは、1924年にアメリカのホーソン工場で行なわれた実験がきっかけであった[2)]。この実験では物理的な環境条件と生産性の関係を調べようとして、環境よりもむしろ人間関係の方が労働意欲に大きく関与するという事実が発見された。以来、集団の成員間の人間関係がパフォーマンスに及ぼす影響について多くの研究がある[3)]。また、リーダーシップのＰＭ論[4)]ではリーダーの機能として

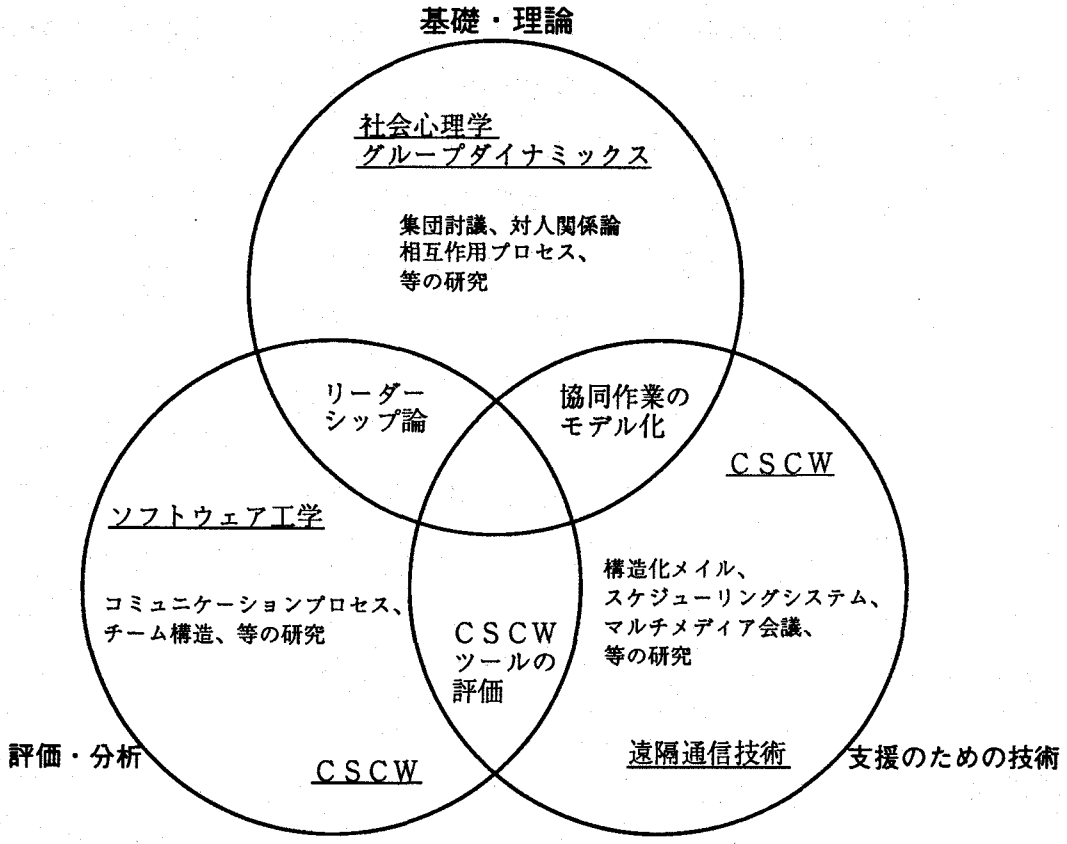


図６．１　協同作業におけるコミュニケーション研究の分類

「課題遂行機能」と「人間関係維持機能」をあげ、双方に優れたリーダーがもっとも高い生産性を達成できることを示している。このように、成員間の人間関係が集団の生産性に何らかの影響を及ぼすことは明らかである。

次に、コミュニケーション支援の研究はここ数年で盛んになってきた。その背景には、社会的ニーズの拡大や通信技術の発達がある。計算機でコミュニケーションを支援する研究はＣＳＣＷと呼ばれ、現在注目を集めている。ＣＳＣＷには、物理的、時間的に離れたメンバー間の自然なコミュニケーションを実現する研究[5) 6)]や、情報を構造化して知的作業を支援する研究[7) 8)]などがある。ＣＳＣＷをソフトウェア開発へ応用したコミュニケーション支援システムも研究されている[9) 10)]。

最後に、協同作業の現場では、調査・分析が行なわれている。協同作業におけるコミュニケーションプロセスの分析[11) 12)]や、ＣＳＣＷシステムを導入して失敗した事例の分析[13) 14)]がある。ソフトウェア工学では、分散開発プロセスの調査[15) 16)]、チーム構造やコミュニケーション構造の研究[17) 18)]が行なわれている。

以上、協同作業のコミュニケーション研究を基礎→支援技術→現場、というサイクルで捉えたとき、現場分析を理論にフィードバックさせるための評価研究が少ない。従来のフィールド研究は現状の把握に留まり、評価方法を体系的に扱ったものはない。このような現状を踏まえて、本研究は以下のような目的を持つ。

1）コミュニケーション評価の枠組みを提供する。

　コミュニケーションの成否は主観的な部分が多く、評価枠組みがなく比較が困難だった。本手法は第三者による調査を容易にし、コミュニケーションの問題をプロジェクト間で比較する方法を提供する。

2）問題点を視覚化する。

　これまでコミュニケーションの問題は生産性向上にとって２次的なものとして処理されがちだった。本方法は問題点を顕在化し、コミュニケーションの重要性を認識させるデータ表現方法を提供する。

以下では、まず協同作業における人間関係を向上させるコミュニケーションの重要性について述べた後、コミュニケーションを分類し、質問紙の内容と採点方法の説明、最後に適用事例を紹介する。

6．3　インフォーマルコミュニケーションの重要性

評価手法について述べる前に、協同作業におけるコミュニケーションの役割、特に

インフォーマルコミュニケーションの重要性について述べる。

ソフトウェア開発プロジェクトでは、仕様の情報を正確に伝えることがコミュニケーションの第一目的になる。しかし、協同作業には、いわゆるインフォーマルコミュニケーションも不可欠である。例えばFikes（1982）[19]やRoot（1988）[20]は、多くの仕事が水面化のコミュニケーションによってうまく行なわれていることを指摘している。また安田ら（1992）[21]の調査によれば、サテライトオフィスのように作業現場が離れていると、インフォーマルコミュニケーションが難しくなり疎外感がある、組織への帰属意識が薄れる、仕事ぶりを見てもらえない不満を感じる、などの問題が生じる。さらにソフトウェア協同開発では、標準化されたフォーマルなコミュニケーションだけでは意図が伝わりにくく[22]、伝わらない部分を日常のコミュニケーションで補っている。

我々が実働プロジェクトに対しインタビュー調査を行なったところ、管理者は経験的にインフォーマルコミュニケーションに日頃から気を配っていた。以下ではインタビューと観察をもとに、インフォーマルコミュニケーションと生産性との関係を個人効率への影響と情報伝達効率への影響とに大別して具体的に整理する。情報伝達効率は、さらに正確さと速さに分けている。これらの内容は表6.1にまとめている。表の第2列は各要因を構成する要素で、第3列では各要素の内容を説明している。第4列は各要素の内容が日常のどのようなコミュニケーションから形成されるかという実例をあげている。

1）個人の生産効率への影響

個人の生産効率を向上させる要因には、能力とモチベーションがある（物理的な環境は省く）。能力を向上させるものは学習、経験である。一般には社内研修、技術レビューなどの学習の場が設定されている。一方、インフォーマルなコミュニケーションの中でも学習は行なわれている。新人は、先輩の行動様式を観察して真似ることにより仕事の能率を上げていく。新人の行動に対しては、周囲からの指導や助言も与えられるだろう。雑談から新たな知識が得られることもある。ソフトウェア開発の場合、プログラム言語の知識は教科書から得られても、デバッグの仕方、マニュアルの読み方などは、他の人に相談したり、真似たりしながら覚えていく。開発に対する基本姿勢、思考方法、習慣などは日常会話の積み重ねによって伝わる。

モチベーションやモラルも生産性を高める要因である。これらを向上させるものには、報酬や人間関係がある。報酬は経済的なものだけでなく、他人から期待や称賛を

よせられる満足感も大きい。人間関係は、共通の目標をもつという連帯感、仲間意識、毎日出会う親密感、などによって形成される。同僚との人間関係が悪いとモチベーションは低下する。モラルは、仕事への満足感、誇り、リーダーシップ、集団への帰属意識などによって高められる。モラルが高い集団は活発で、能率も上がる。

2）情報伝達効率への影響

協同作業におけるコミュニケーションの基礎は、情報を交換することである。情報伝達効率は一見技術的に解決でき、インフォーマルコミュニケーションとは無関係のようだが、実際には次のような影響がある。

その1：正確さに対する影響

ソフトウェアを標準化された方法で記述しても正確に伝わらないケースとして、次の状況が考えられる。

i) 伝える意志はあるが伝えられない
- ・抽象的なイメージを表現できない
- ・全部書いている（話している）とキリがない

ii) 伝える意志がない
- ・常識だから言うまでもないと思うことは、意識的、無意識的に言わない

iii) 話し手は伝えたつもりでも聞き手が誤解する
- ・抽象的な表現なので、誤解しやすい
- ・互いに常識だと思っていることが違う

いずれの場合も、聞き手は伝わらなかった部分を補って理解しなければならない。正しく補う（推測する）ためには、その背景となる知識（相手の思考方法、経験、知識のレベル、意図、プロジェクトの背景、状況、等）を知っていることが必要である。これらは日常の些細なコミュニケーションの積み重ねで伝達される。また、ディスカッションやレビューなどで、互いの考え方について徹底的に話し合う機会も必要である。

その2：伝達の速さに対する影響

必要な情報を早く送ることは効率向上の重要な要件である。伝達にかかる時間は、距離やメディアに依存する。しかし次のようなケースでは、伝達が遅れがちになる。

i) 相手に話に行くのが気が重く、ついつい後回しになる。または直接話すのを避けてしまい、電子メイルなどですませようとして十分な伝達ができない。

ii) プロジェクト内のコミュニケーションが不足していると、誰がどのような知識をもっているのか知らず、必要な情報を探すのに手間取る。有効な情報がそばにあっても気がつかないために利用できない。

iii) コミュニケーションが不足しているためメンバー間に誤解があっても気がつかず、問題の発見が遅れる。

これらの問題は、メンバー間の心理的距離やコミュニケーション頻度に起因する。ソフトウェア開発の場合、問題の発見が遅れると解決に多大なコストがかかるため、問題の早期発見は特に重要である。そのため、問題があるときだけ連絡を取り合うのではなく、日頃からコミュニケーションをもち、問題を発見する機会を作ることが大切である。

以上のように、インフォーマルコミュニケーションも直接、間接に作業効率に影響している。しかし、人間関係が良くても緊張感がなくインフォーマルな会話ばかりでは、逆に作業の能率は下がってしまう[23]。重要なことは、「情報伝達」と「交わり」の両側面がバランスよく存在することである。

ここで提案する評価方法では、この2側面の程度を測定し、視覚化する。「交わり」要素は主観的なものであるから、主観評価を中心とした質問紙法を用いる。評価の最終目的はコミュニケーションの問題点を見つけることであり、**「個々のメンバー間の問題（ミクロな評価）」**と**「プロジェクト全体の問題（マクロな評価）」**の2つのレベルからで問題を見る。「ミクロな評価」は部分的な生産性低下の原因を示し、管理者やメンバー自身が個別に対処すべきものである。「マクロな評価」はプロジェクト全体のコミュニケーション傾向を示すものである。

次節では、実際に測定するコミュニケーションを定義する。

6．4　コミュニケーションの分類

「伝達」と「交わり」の2側面を直接測定することは困難なので、2つの要素を異なった割合で含むいくつかのコミュニケーションを定義し、それらを測定することで間接的に判断する。

1）評価コミュニケーションの分類

「伝達」「交わり」の2要素の占める割合によってコミュニケーションを5つに分類する（図6.2）。まず「交わり」要素が極端に多く「伝達」要素が極端に少ないもの

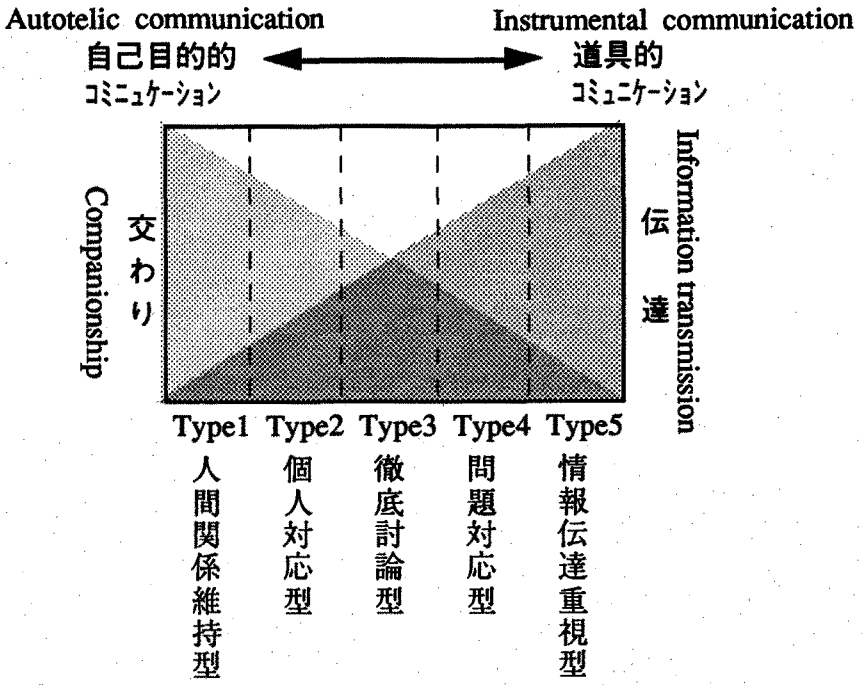


図６．２　コミュニケーションの２側面とそれに基づいた５類型

を人間関係維持型とし、逆に「伝達」要素が極端に多く「交わり」要素が極端に少ないものを情報伝達重視型とした。次に２つの要素を同等に含んでいるものを徹底討論型とした。この３つだけでは現状のコミュニケーションを詳細に分析できないため、その中間型として個別対応型、問題対応型を設けた。各タイプのコミュニケーションの内容を以下に整理する。

①Type1：人間関係維持型

目的：情報を伝達することではない。その人と会話すること自体が目的。目的がないこともある。

内容：不特定なもの。世間話。趣味、家庭の話。社内人事や仕事の話。

例　：雑談。挨拶。気晴らし。

②Type2：個別対応型

目的：ある特定の人と会話して、何かの情報を得る。

内容：情報の内容は漠然としている。仕事の話だったり、個人的な悩みだったり。

例　：遠回しにホンネを探る。仕事の遅れについて、原因と対策を話し合う。

③Type3：徹底討論型

目的：お互いの意見を出し合い、討論し、相互理解を得る。最終的に何かを決定することもある。

内容：仕事全般。関連分野のこと。

例　：設計の方針について議論。問題の解決策について関係者で集団討議。

④Type4：問題対応型

目的：ある情報を誰かに連絡する。必要があるときのみ行ない、形式は自由。

内容：仕事に関するもので、緊急のトラブル処理など非定型のもの。

例　：デバッグについて先輩に相談する。臨時会議を開いて関係者と打ち合わせる。

⑤Type5：情報伝達重視型

目的：ある情報を決められた人に連絡する。決められたフォーマットがある。

内容：仕事に関するもので、業務連絡など定型で一方向の伝達。

例　：定期連絡会。仕様書等ドキュメント類。

2）評価の指標

コミュニケーションの成否は定量的に測定しにくい。たとえば上司が人間関係維持のため部下とのコミュニケーション回数を増やしても、それを部下が嫌がっているようであれば、そのコミュニケーションは失敗しているといえる。反対にコミュニケーションの回数が少なくても、お互いの意図が誤解なく伝達されているようならば、そのコミュニケーションはうまく働いている。そこで、5種類のコミュニケーションがうまくいっている状況について評価指標をつくり、その指標の主観的達成度で評価する。

① Type1: 人間関係維持機能が働いているときの指標

・雑談をする

・気軽に話しかけられる

・接触する頻度や距離（毎日顔を会わせる、声が聞こえるほど近い、等）

② Type2: 個別対応機能が働いているときの指標

・1対1で仕事の話をする

・仕事で困ったときに相談できる

③ Type3: 徹底討論機能が働いているときの指標

・互いに納得するまで話し合う

・一方的な決定になっていない

・互いの意図が食い違っていたことはない

・仕事について相談できる

④ Type4: 問題対応機能が働いているときの指標

・必要に応じて迅速な連絡が取れる

⑤ Type5: 情報伝達重視機能が働いているときの指標

・特に用事がなくても定期的なミーティングを開催する

・必要な情報はもれなく伝達される

この指標をどれだけ満足していれば必要十分か、という合格ラインはない。それはプロジェクトの作業内容や組織の形態によって変わってくることで、理想のコミュニケーションを決めるのは難しい。本手法の目的は、上記５つのコミュニケーションのうち極端に不足しているもの、偏っているもののバランスを目に見えるようにすることである。

６．５　質問紙による評価手法の提案

６．５．１　項目の作成

質問紙はプロジェクトの全メンバーに配付し、記名式で回答してもらう。質問項目の作成に当たり、以下の点を考慮した。

・プロジェクトのメンバーを予め指定し、その人に対するコミュニケーションを評価してもらう。

・指標は主観的に判断してもらう。コミュニケーションの失敗・成功を客観的に判断するのは難しく、人間関係には主観的な部分が多いからである。

・不足はないか、失敗はないか、と言うように否定文を多くする。これは「コミュニケーションがどのくらいうまく行っていますか？」と肯定文で聞くと否定的な回答が出にくく[24]、「どのくらいの頻度で失敗しますか？」と聞いた方が問題が現われやすいと考えたからである。

質問項目はプリテストの後、１７項目とした（付録３に質問紙抜粋を添付した）。質問数は少ないが、数名の同僚に対して同じ項目数の回答をしてもらうため、時間がかかり過ぎないようにこの数に押さえた。５人に対して回答した場合、２０～３０分になる。それぞれの回答形式は、頻度に関するものは７段階評定、内容に関するものは複数選択肢である。

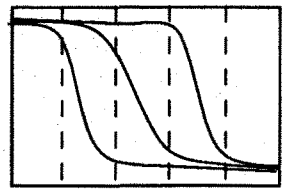
6．5．2　結果の集計方法

1）ミクロな評価の算出方法

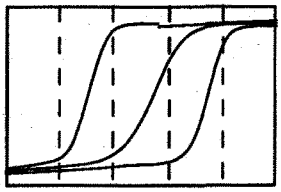
各回答の採点方法は指標に基づいて判断する。コミュニケーションを「多い－少ない」の７段階で評定する項目は、多い方から２段階の評価を肯定的評価と考えて+1点、少ない方から２段階の評価を否定的評価と考えて-1点、中間の３段階の評価をどちらとも言えない場合と考えて0点と採点した。選択肢の項目では、内容に応じて+1、0、-1点をつける。例えば相手との座席の距離を尋ねる項目について「席にいたまま話しができる」を選択した場合、十分に近接しているとみなして+1点、「交通手段を使わないと会えない距離」であればコミュニケーションに不利であるとみなして-1点とする。各項目は５つのコミュニケーションタイプに対応した質問なので、各タイプについてそれぞれの合計得点が出る。５つのタイプに関する質問数が異なり単純には比較できないため、合計得点を個人毎に－１～＋１の間に正規化し、グラフ化した。グラフからは、十分に機能しているコミュニケーション（＋の得点を取っている）と機能していないコミュニケーション（－の得点を取っている）のバランスが視覚的にわかる。図6.3のようにグラフの形には５種類のパターンが存在する。グラフの形はコミュニケーションの特徴を表わしており、メンバー間の人間関係をも表現していると考えられる。コミュニケーションの特徴がわかりやすいように各パターンに名前を付けた。人間関係重視型（Type1）に比べ情報伝達重視型（Type5）が少ない場合を交わり重視パターン、反対に情報伝達重視型（Type5）に比べ人間関係重視型（Type1）が少ない場合を情報伝達重視パターンとする。他に両端（Type1、Type5）が少なく徹底討論型（Type3）が多い場合を話し合い重視パターン、反対に両端が多く徹底討論型が少ない場合を話し合い軽視パターン、どのコミュニケーションも均衡しているパターンが考えられる。分類の基準は、グラフの高低差が0.75以上のものを差があるとみなし、差が見られないものは均衡パターンとした（図6.4）。さらに正規化した得点の高さで３段階に分け、得点合計が＋になる場合［＋］、－になる場合［－］、０になる場合［０］とした。質問紙の結果は、以下の５つのパターンのどれかに分類し、それぞれに［＋］［－］［０］の３段階を出す。

[1] 交わり重視（情報伝達軽視）パターン

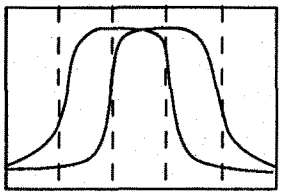
雑談や「調子はどうですか」的な目的のないコミュニケーションが主で、定期的な打ち合わせなどの形式的なものは少ない。



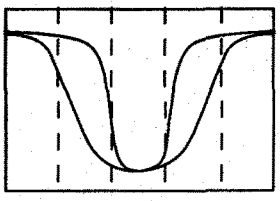
［1］交わり重視パターン
Type1が多く、
Type5 が少ない



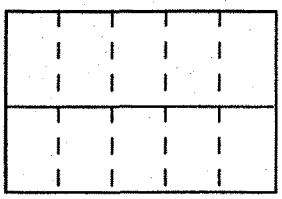
［2］伝達重視パターン
Type5 が多く、
Type1 が少ない



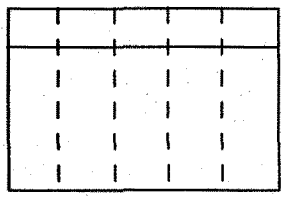
［3］話し合い重視パターン
Type1、Type5 が少なく、
Type3 が多い



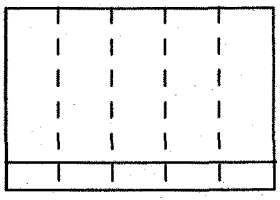
［4］話し合い軽視パターン
Type1、Type5 が多く、
Type3 が少ない



［5］均衡パターン

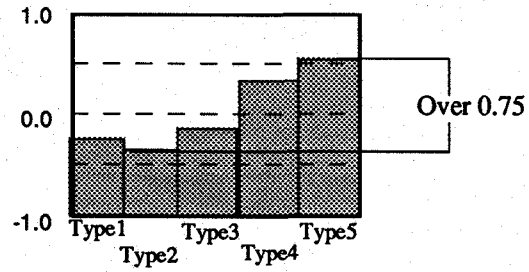


［A］ｺﾐｭﾆｹｰｼｮﾝ量［多］

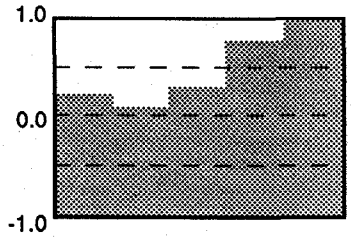


［B］ｺﾐｭﾆｹｰｼｮﾝ量［少］

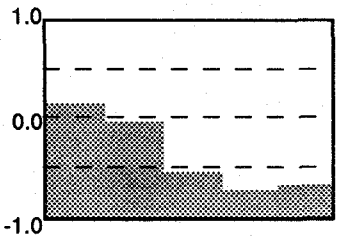
図６．３　評価パターンからみた対人関係の特徴分類



最高値と最低値 の差が 0.75 以上の場合、特徴的とみなす。
（この例では　伝達重視パターン［0］と表記する。）



評価の平均値が 0.0 を越える場合
満足度が高いとみなす。
（この例では伝達重視パターン［+］
と表記する。）



評価の平均値が 0.0 未満の場合
満足度が低いとみなす。
（この例では伝達重視パターン［−］
と表記する。）

［評価をパターンに分類する基準。］

(1) 最高値と最低値の差が 7.5 以上の場合、差があると考える。つまり特徴のある
パターンであるとみなす。
最高値と最低値の差が 7.5 未満の場合、差がない。つまりバランスが取れているとみなす。
(2) 評価の平均値が 0.0 を越える場合、コミュニケーションの満足度が高いとみなす。　［+］
(3) 評価の平均値が 0.0 の場合、コミュニケーションの満足でも不満でもないとみなす。［0］
(4) 評価の平均値が 0.0 未満の場合、コミュニケーションの満足度が低いとみなす。　［−］

図６．４　パターン分けの判断基準

2] 伝達重視（交わり軽視）パターン

会議や必要があるときだけコミュニケーションし、人間関係維持のためのコミュニケーションが不足している。

[3] 話し合い重視パターン

必要があるときだけコミュニケーションする。単に連絡だけでなく徹底的に議論するが、個人的な話題や定期的な打ち合わせは少ない。

[4] 話し合い軽視パターン

定期的な打ち合わせや必要最低限の連絡で会う他は、雑談や挨拶だけの付き合い。仕事の話についてじっくり話し合うことはない。

[5] 均衡パターン

どのコミュニケーションも同程度に行なっている。

このようにコミュニケーションを特徴づけることにより、人間関係の雰囲気を表現することができる。こうして、プロジェクトのメンバー間のすべてのコミュニケーションに、いづれかのパターンと3段階評価が割り当てられる。二人のメンバーの間では相互に評価した2つのパターンが出るので、それを矢印で表示した（図6.7 の具体例を参照）。色が濃い方が伝達重視の傾向にあり、色が薄くなるにつれ人間関係維持を重視する傾向がある。矢印の輪郭が薄く点線になっているものはコミュニケーションがない箇所（話したことがないので評価できないという回答）である。矢印がまったく書いていないところは、質問紙が回収されなかったところである。

2）マクロな評価の算出方法

個々の評価得点を単純に合計すると特徴が消えて平均化されてしまうので、特徴を残すためコミュニケーションパターンの出現頻度を集計して全体の評価とする。まず5つのコミュニケーションパターンと3段階評価［+］［0］［−］を集計し、全体の中での割合を見るためにパーセントで表示する（表 6.3 に具体例を示す）。評価を統合することによって、プロジェクト内のコミュニケーションの質（人間関係を重視している、情報伝達を重視しているなど）と量（各個人が十分だと思っているかどうかの主観的な量）のバランスがわかる。が、数字のままでは問題が見えにくく評価結果を管理者へフィードバックしにくいため、顔の表情を用いてコミュニケーションの特徴を視覚的に表現する（図 6.5）。表情は対人感情を表すので、コミュニケーションの問題を表現しやすい。顔の5つの要素（目、眉、口、傾斜、輪郭）にそれぞれの

パターンを対応させ、そのパターンの出現頻度に応じて要素を強調する。無表情を基準とし、「交わり」要素の多いものは表情を明るく、「伝達」要素の多いものは表情を険しくした。3段階評価［＋］の割合が多いものは要素の線を太く、［－］の割合が多いものは線を細く表現した。パターンと表情の対応は以下のとおりである。

[1] 交わり重視（情報伝達軽視）パターン

目の丸さ（笑っている雰囲気）で表現する。

［＋］［－］の評価は線の太さに対応する。

[2] 伝達重視（交わり軽視）パターン

眉の傾斜（険しい雰囲気）で表現する。

［＋］［－］の評価は線の太さに対応する。

[3] 話し合い重視パターン

口を開く大きさ（話している雰囲気）で表現する。

［＋］［－］の評価は線の太さに対応する。

[4] 話し合い軽視パターン

顔の傾斜（首を傾げている雰囲気）で表現する。

［＋］［－］の評価は顔の影の濃さに対応する。

[5] 均衡パターン

無表情。このパターンが増えても表情は変わらない。

［＋］［－］の評価は線の太さに対応する。

[6] 全パターンを合計したもの

［＋］の割合に応じて顔の面積を大きく、

［－］の割合に応じて表情を小さくした。

顔が大きく表情が小さいとアンバランスな顔になり、プロジェクトの中にコミュニケーションの活発な所とそうでない所があることを示す。

顔グラフには細かい情報は表せないが、プロジェクト全体の雰囲気を表現できる。表情からはコミュニケーションの性質がわかる。笑っていれば人間関係維持型が多い。眉を吊り上げていれば情報伝達型などである。

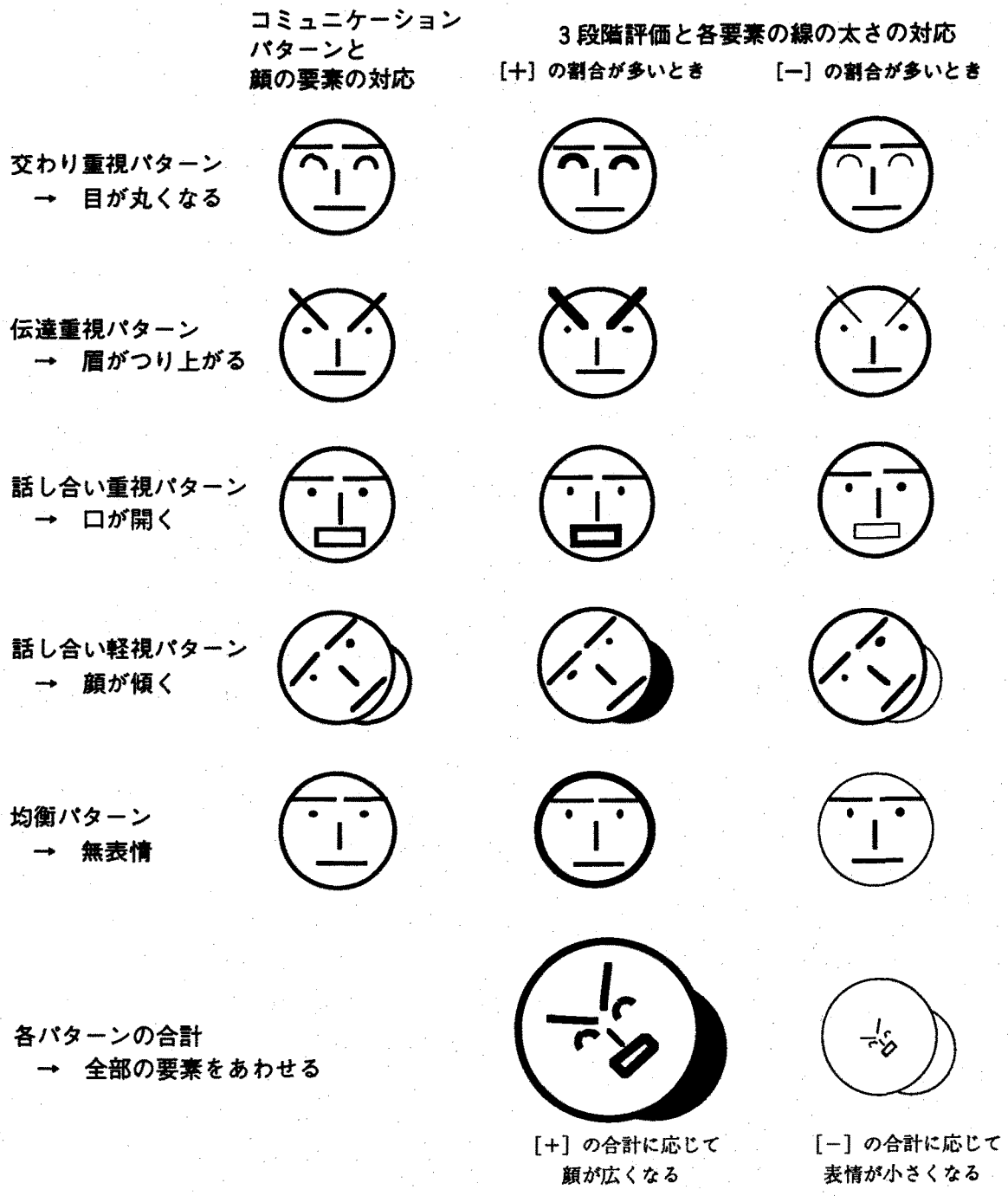


図６．５　顔の表情によるプロジェクトの雰囲気の表現

3）グルーピング評価

プロジェクトが複数のチームから構成されていたり、場所が離れていたりするとき、全体として一つの顔を作るより、いくつかのグループに分けて顔を出し、比較したほうが適切な場合もある。グルーピング評価では、グループ内の顔と他グループに対する顔を出す。顔の出し方はマクロな評価と同じである。具体的な計算の仕方を次節の実施例で示す。

これらの評価は、コミュニケーションの状況を視覚化するものである。図6.6 は評価手法の実施手順である。プロジェクトにどのような問題があるかは、ミクロ、マクロ、グルーピングの結果から読み取らなければならない。そのためには、プロジェクトの仕事内容や組織構造を予備知識として持っている必要がある。結果を解釈するさいの一般的なポイントを付録２の表 6.2 に添付する。

６．６　評価手法の適用と考察

上で提案した評価手法を実際のプロジェクトに適用し、その妥当性を検討した。実施したのは次の２つのプロジェクトである。今回はプロジェクトマネージャに質問紙

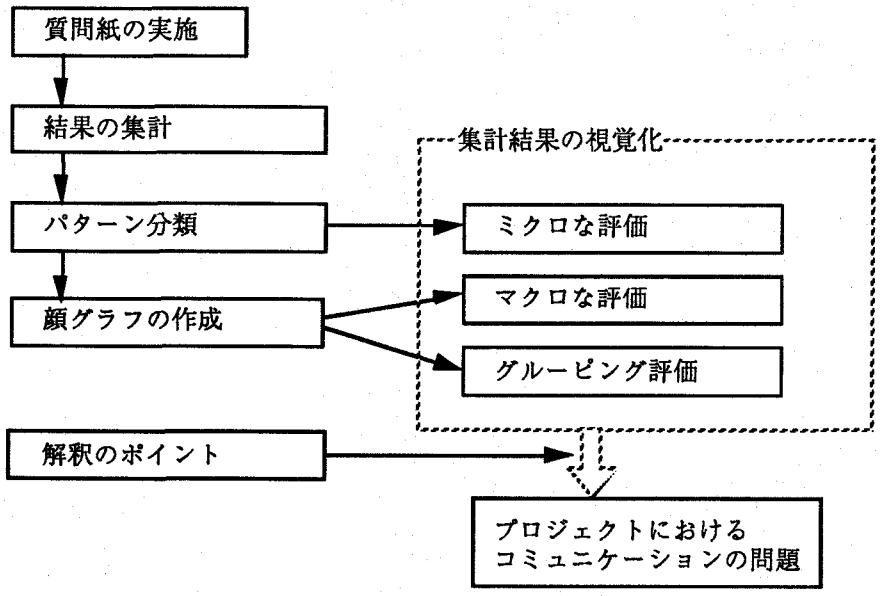


図６．６　プロジェクト評価の手順

の主旨を説明し、マネージャを介してメンバーに質問紙を配布してもらった。回収方法は、各メンバーから直接郵送してもらい、回答内容が他人にわからないようにした。

○プロジェクトⅠ：メンバー５人、回収率１００％、１年間のプロジェクトの半ば頃に質問紙を実施した。

○プロジェクトⅡ：メンバー８人、回収率６０％、３年間のプロジェクトがほぼ終了する時期に質問紙を実施した。

回答を集計した結果の解釈例を示す。

1）プロジェクトⅠ

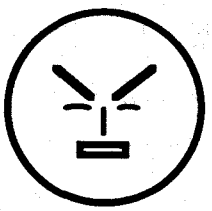
集計結果を図6.7 に示す。表6.3 は顔の算出に用いたもので、パターンごとに評価の高低（［+］［0］［−］）を集計し、その割合を計算している。メンバーの座席配置は、管理者Ａとプログラマ Ｃが近く、システム設計者Ｂとプログラマ Ｄが近い。プログラマ Ｅは外部のソフトウェアハウスにいて離れているが、Ｂ、Ｄとは仕事での付き合いが長く、頻繁に連絡を取っている。

［マクロな評価］（図 6.7 （ａ））

・伝達重視タイプ、話し合い重視タイプ、均衡タイプが多い。
・交わり重視タイプが少ない。
・顔の面積が広く、総じてコミュニケーション量は多い。
・管理者、設計者は伝達重視型コミュニケーションが目立つ。が、席が近い者とは（ＡとＣ、ＢとＤ）バランスよくコミュニケーションしている。
・Ｂ、Ｄ、Ｅの間ではコミュニケーションは多いが、Ｄ、ＥはＡ、Ｃとは席が離れていて、ほとんどコミュニケーションがない。Ｂは席が離れているが、打ち合わせのためか、ＡやＣとのコミュニケーションが多くなっている。

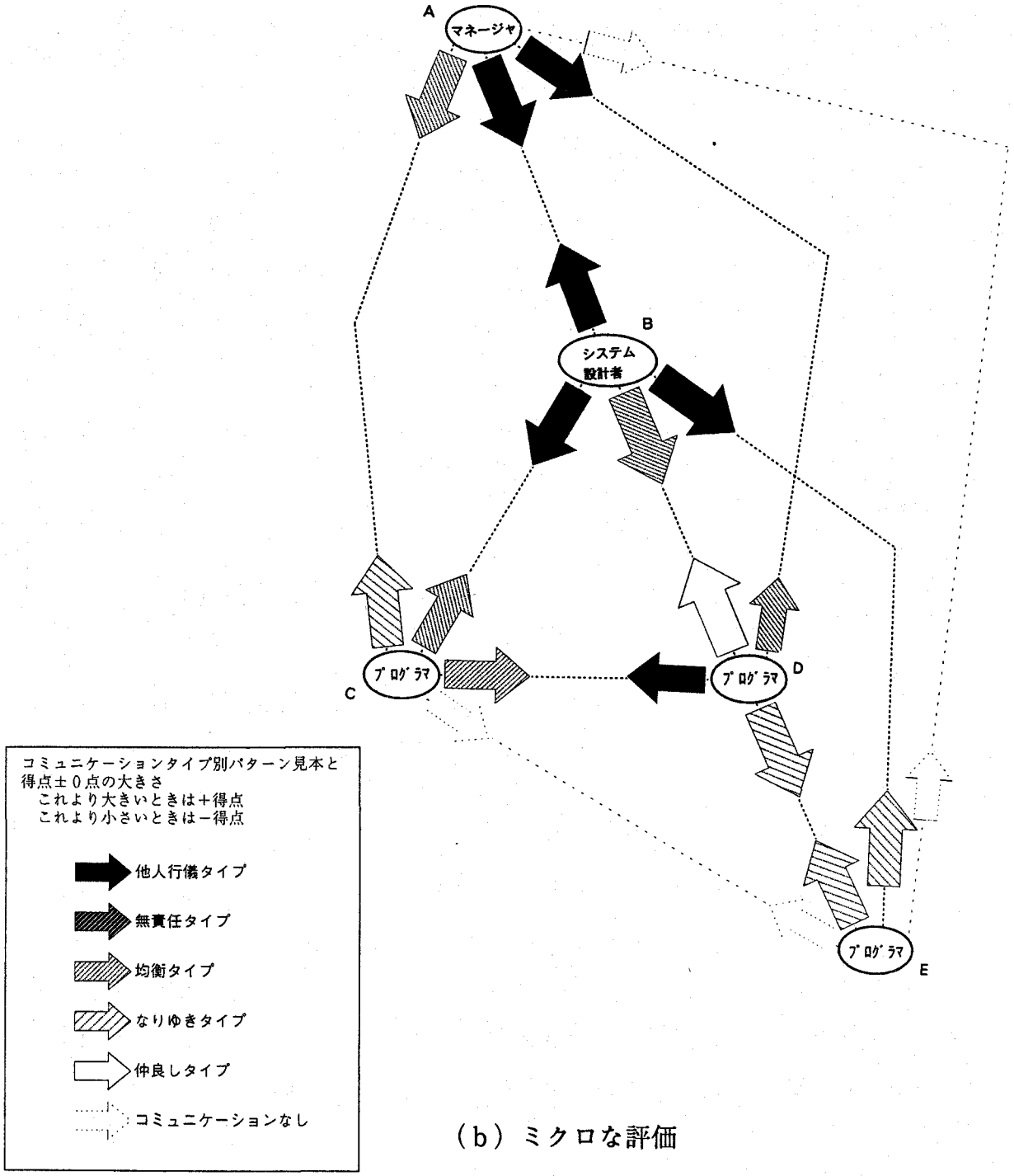
［ミクロな評価］（図 6.7（ｂ））

・管理者、設計者は伝達重視型コミュニケーションが目立つ。が、席が近い者とは（ＡとＣ、ＢとＤ）バランスよくコミュニケーションしている。
・Ｂ、Ｄ、Ｅの間ではコミュニケーションは多いが、Ｄ、ＥはＡ、Ｃとは席が離れていて、ほとんどコミュニケーションがない。Ｂは席が離れているが、打ち合わせのためか、ＡやＣとのコミュニケーションが多くなっている。



（a）マクロな評価

［解釈のポイント］
・顔が広い→　全体にコミュニケーション量が多い。
・顔の輪郭線が太い→　均衡パターンが多い。
・眉が太く、つり上がっている→　情報伝達重視型が多い。
・口が開いている→　話し合い重視型が多い。
・目が細いが丸くない→　交わり重視型は少し。



（b）ミクロな評価

図6．7　プロジェクトⅠの評価結果

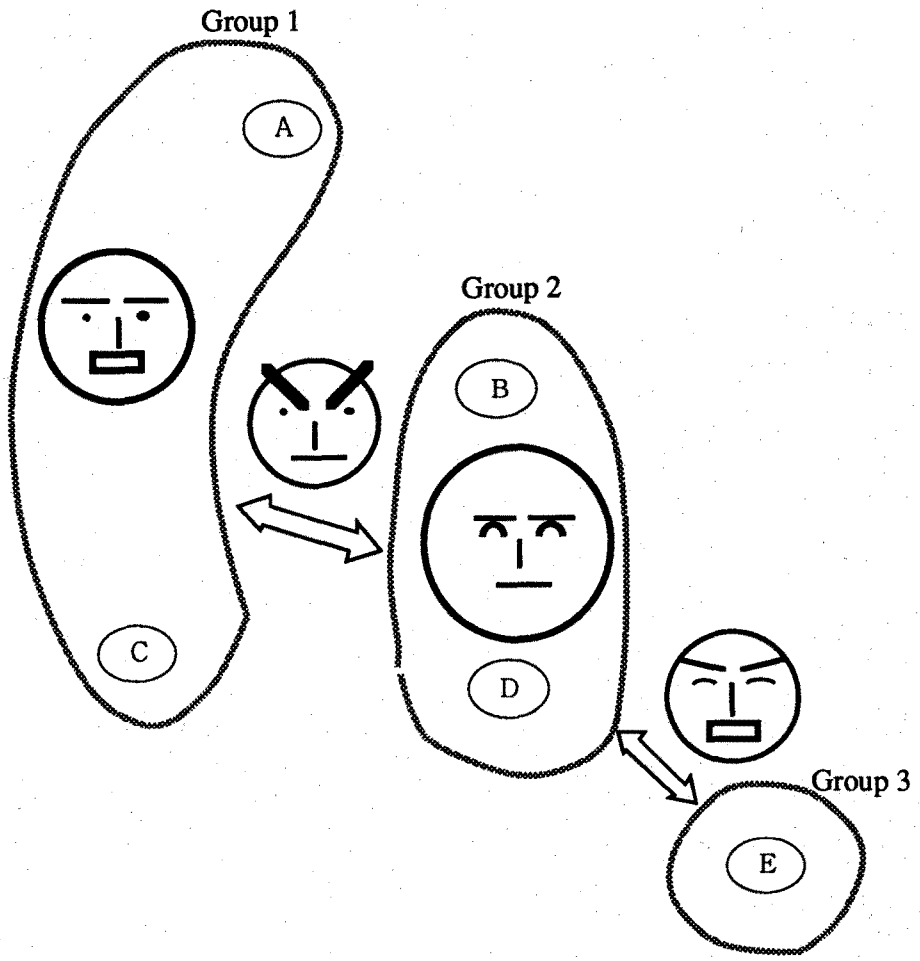
表6．3　プロジェクトＩのコミュニケーションパターンの分布

（a）各パターンの出現頻度

(回答数)	[+]	[0]	[-]	無	計
交わり重視パターン	1	0	0	--	1
連絡重視パターン	4	1	1	--	6
均衡パターン	4	0	1	--	5
話し合い重視パターン	4	0	0	--	4
話し合い軽視パターン	0	0	0	--	0
コミュニケーションなし	--	--	--	4	4
無回答	--	--	--	0	0
合計	13	1	2	4	20

（b）各パターンの出現する割合

(%)	[+]	[0]	[-]	無	計
交わり重視パターン	5	0	0	--	5
連絡重視パターン	20	5	5	--	30
均衡パターン	20	0	5	--	25
話し合い重視パターン	20	0	0	--	20
話し合い軽視パターン	0	0	0	-	0
コミュニケーションなし	--	-	--	20	20
無回答	--	-	--	0	0
合計	65	5	10	20	100



プロジェクトＩを座席の距離によって３つにグループ分けし、評価した。
グループ１、２はそれぞれ中では活発なコミュニケーションをしているが、
グループ間では情報伝達のみになっていることがわかる。

図 6.8　プロジェクトＩにおけるグルーピング評価例

［グルーピング評価］（図6.8）

・座席の距離でグループ分けした。

・このプロジェクトはメンバーの席が３箇所に分散し、グループ１（ＡとＣ）、グループ２（ＢとＤ）、グループ３（Ｅ）に分けられる。

・グループの内側の表情は笑ったり口を開けたりしているが、グループ１と２の間の表情は眉がつり上がっており、連絡だけのコミュニケーションになっている。

［問題点の総合解釈］

作業に必要な情報は伝達されるが、交わり要素が少なく、和気藹々ムードではない。管理者にはプロジェクト内の雰囲気作りや個別対応のためのコミュニケーションが不足しており、表面的な情報は伝達されるが、潜在的な問題の発見が遅れる危険がある。特に席が離れたメンバーとは、連絡は行なっているが日頃の接触がなく、意図が正しく伝わらない可能性もある。また個人的な問題による生産効率の低下や外部組織との軋轢なども認識されにくい。

２）プロジェクトⅡ

集計結果を図6.9 に示す。メンバーは８人だが、一番上の管理者は名目だけで、日頃の接触はない。プログラマＰ１とＰ２、Ｐ３とＰ４は、それぞれ別の外部ソフトウェアハウスに所属し、定期的に打ち合わせするほかは電話などで連絡を取っている。

［マクロな評価］（図 6.9 （ａ））

・顔が少し大きいのと同時に表情が非常に小さく、一部のものだけが頻繁にコミュニケーションし、全体としてはコミュニケーション量が少ないことがわかる。

・眉や口にも変化は見られるが顕著ではなく、均衡タイプが多い。

［ミクロな評価］（図 6.9（ｂ））

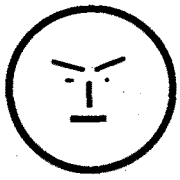
・コミュニケーションのないところ、無回答が多い。

・コミュニケーションが多いところは均衡タイプか話し合い重視タイプ。

・システム設計者ａ、Ｐ１、Ｐ２の間ではコミュニケーションが活発で、プロジェクトの中心的存在になっている。

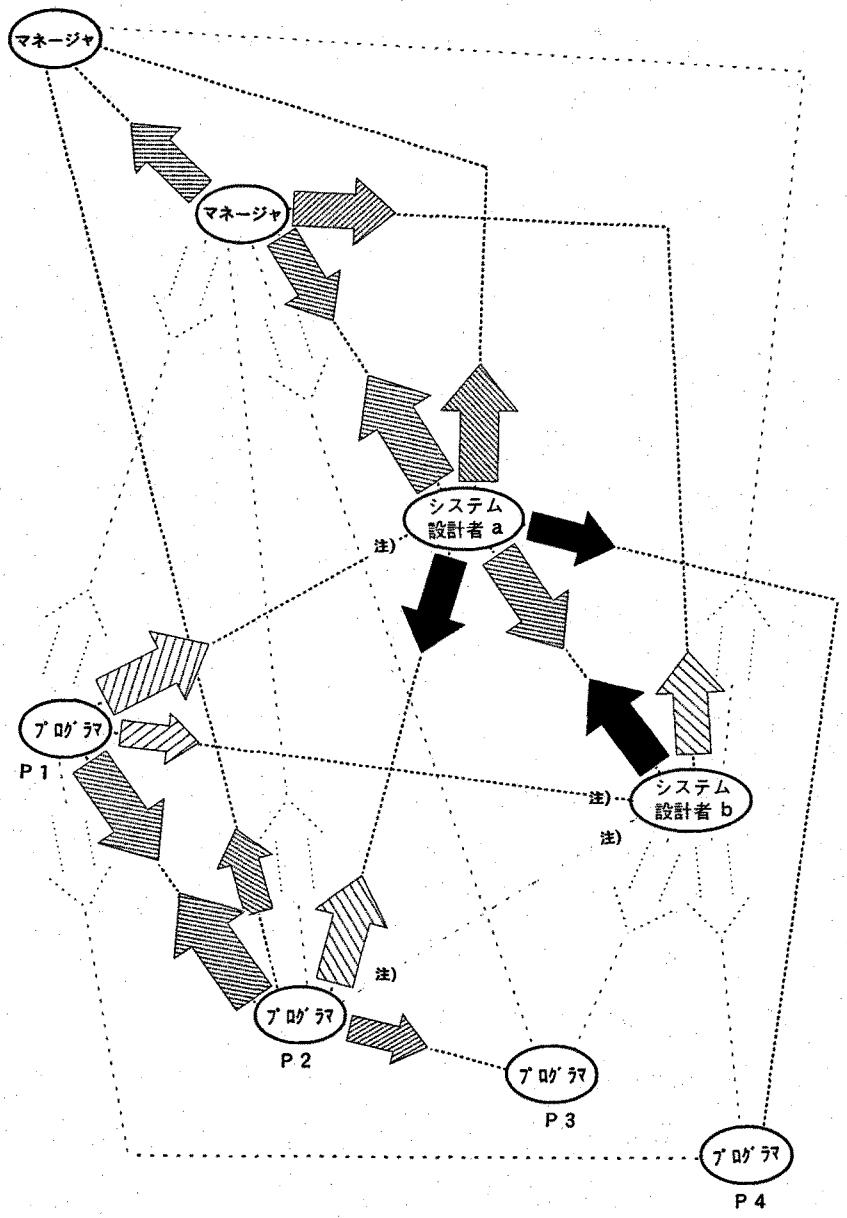
［グルーピング評価］

無回答が多いので、グルーピング評価は行なわなかった。



［解釈のポイント］
・顔が少し広く、表情が小さい→　コミュニケーション量が多い所と少ない所に分れる。
・顔の輪郭線が太い→　均衡パターンが多い。
・眉が細く、少し上がっている→　情報伝達重視型が少し。
・口が小さく開いている→　話し合い重視型が少し。

（a）マクロな評価



注）は、時間が長くなり過ぎて回答してもらえなかったところ。
無回答や回答の拒否ではない。

（b）ミクロな評価

図6．9　プロジェクトⅡの評価結果

[問題点の総合解釈]

コミュニケーションの多いところとないところが極端にわかれており、人間関係に問題があることを暗示している。席が離れていて日常のコミュニケーションがないという回答と無回答とは人間関係の質が違う。無回答の場合、軋轢があるため回答を拒否したということも考えられるので、要注意である。ここではプログラマが二つのソフトウェアハウス（Ｐ１、Ｐ２）と（Ｐ３、Ｐ４）で構成されており、特に（Ｐ３、Ｐ４）と設計者、管理者とのコミュニケーションが問題である。

質問紙の妥当性を検討するため、質問紙回答後に各プロジェクトのマネージャからコメントを得た（マネージャには結果は知らせていない）。

プロジェクトⅠ

「作業は若干遅れぎみだが順調に進んでいる。自分は進捗管理が主で、内容についてかかわることはない。質問紙に回答しながら、普段人間関係に気を使うようなコミュニケーションをしていなかったことに気がついた。（Ｂ、Ｄ、Ｅの３名を指して）このメンバー達は以前から一緒に仕事をしており、設計も話合いながらやっている。

プロジェクトⅡ

「自分は進捗管理にしか携わっていないが、プログラマたちの間では人間関係に摩擦があったとも聞いている。一部のメンバーからは質問紙を断られた。プロジェクトのメンバーの移り変わりもあり、外部（ソフトウェアハウス）では転職するケースも多いらしい。作業自体は複雑なものではなく、問題なく出荷した。」

これらのコメントの内容は質問紙による評価と一致する。それは主に次の点である。

・「伝達」要素は十分に機能しており、作業を進めるのに困難はなかった。
・「交わり」要素が一部のメンバーにしか見られず全体に少ないため、プロジェクトとしてのまとまりや仲間意識が育っていない。
・コミュニケーションの量が著しく少なく、人間関係に問題のある箇所が特定できる。

一般に心理検査の妥当性の検証には、他の資料や既存の検査との整合性の比較やエキスパートによる判定との比較などがある[25]。今回は他に類似の検査が存在しないため、該当プロジェクトのエキスパートであるマネージャの評価と比較する方法をとった。事例は少ないがマネージャのコメントは質問紙の結果を肯定するものであった。今後、大規模なプロジェクトなどへも適用し、評価事例を蓄積していく予定である。

本評価手法の活用事例として、コミュニケーション支援やプロジェクト管理への適用が考えられる。評価結果をどう解釈して対応するかは、プロジェクトのメンバー構成や作業内容と照らし合わせて検討する必要がある。今後評価事例を蓄積して、プロジェクトの内容・構成とコミュニケーションの関係を詳しく分析し、コミュニケーションの問題を提示するだけでなく適切なコミュニケーション支援についてのアドバイスができればさらに有効であろう。

また、コミュニケーション支援ツールを導入する前後で本調査を実施して、ツールの導入によるメンバーのコミュニケーション構造の変化を捉えることができる。ツールの直接的な評価ではないが、人間関係に及ぼすツールの影響を知ることができる。

最後に、プライバシーの問題を述べておかなければならない。実名をあげるので、人間関係に問題がある部分ほど回答しにくいと考えられる。今回はマネージャを介して渡したが、第3者である研究者が直接渡し、プライバシーの厳守を強調したほうがよいと考えられる。

また、大規模なプロジェクトになると全メンバー間の評価は無理になるので、事前調査によるグルーピングを行ない、質問紙を実行する必要がある。

6．7　結言

これまでソフトウェア工学では、いわゆるインフォーマルなコミュニケーションに対して正面から取り組んだ研究はなかった。インフォーマルコミュニケーションは心理的な部分、生産効率は工学的な部分として扱われていたが、インフォーマルなコミュニケーションも協同作業における生産性要因の一つといえる。その重要性を管理者に認識させ、問題に取り組むためには、プロジェクト内のコミュニケーションを評価し、わかりやすいカタチで表示する必要がある。従来インフォーマルコミュニケーションは評価できる形で定義されていなかった。本章では「交わり」と「伝達」の2要素を基にした5分類を提案し、生産性への影響を整理した。つぎにコミュニケーション評価の質問紙を作成し、ミクロな評価とマクロな評価の算出手法を開発した。今後はプロジェクトメンバーへのフィードバックと管理者へのアドバイスやコミュニケーション環境の研究への発展が考えられる。

《参考文献》

1) 加藤春恵子：広場のコミュニケーションへ、勁草書房（1986）

2) Roethlisberger, F. J. and Dickson, W. J. : Management and the woker , Harvard Univ. Press. (1939)

3) Cartwright, D. and Zander, A. (Eds.) : Group Dynamics : Research and theory, Harper and Row (1953) （三隅二不二・佐々木薫訳編：グループ・ダイナミクスⅠ・Ⅱ、誠心書房（1959））

4) 三隅二不二：リーダーシップ行動の科学、有斐閣（1984）

5) Goodman, G. and Abel, M. : Collaboration Research in SCL, Proceedings of CSCW'86, pp.246-252 (1986)

6) Fish, R. S., Kraut, R. E., and Chalfonte, B. L. : The VideoWindow System in Informal Communications, Proceedings of CSCW '90, pp.1-12 (1990)

7) Winograd, T. : A Language Perspective on the Design of Cooperative Work, Proceedings of CSCW'86, pp.203-220 (1986)

8) Conklin, J. and Begeman, M. L. : gIBIS : A Hypertext Tool for Exploratory Policy Discussion, Proceedings of CSCW'88, pp.102-114 (1988)

9) 古宮誠一：ソフトウェア協調型設計過程のモデル化と知的支援方法について、情報処理学会研究報告、92 - SE - 83、pp.113-120（1992）

10) 貫井春美・松尾朗・太田哲夫：ソフトウェア分散開発支援システムD^2、機械学会　設計工学・システム部門講演会論文集、No.920-103、pp.366-371（1992）

11) Suchman, L. and Trigg, R. : A Framework for Studying Research Collaboration, Proceedings of CSCW'86, pp.221-228 (1986)

12) 氏田博士・久保田龍治・池田宏治・河野龍太郎：プラント運転クルーのコミュニケーション分析、第７回ヒューマンインタフェースシンポジウム、Oct.23-25、Kyoto、pp.19-24（1991）

13) Grudin, J. : Why CSCW Applications Fail : Problems in the Design and Evaluation of Organizational Interfaces, Proceedings of CSCW'88, pp.85-93 (1988)

14) Markus, M. L. and Connolly, T. : Why CSCW Applications Fail : Problems in the adoption of interdependent work tools , Proceedings of CSCW'90, pp.371-380 (1990)

15) 福田由紀雄・井上敦子・津田淳一郎：遠隔地ソフトウェア開発の実験、情報処理学会研究報告、90 - SE - 71、pp.49-56（1990）

16) 海谷治彦・佐伯元司：ソフトウェアの仕様化過程における協調作業のモデル化、

情報処理学会研究報告、91 - HI - 37 (2)（1991）
17）Mantei, M. : The Effect of Programming Team Structures on Programming Tasks, Comm.ACM, Vol.24, No.3, pp.106-113 (1981)
18）Scott, R. F. and Simmons, D. B. : Predicting Programming Group Productivity --- A Communications Model, IEEE Trans. Softw. Eng., Vol.1, No.4, pp.411-414 (1975)
19）Fikes, R. E. : A Commitment-Based Framework for Describing Informal Cooperative Work, Cognitive Science, Vol.6, pp.331-347 (1982)
20）Root, R. W. : Design of a Multi-Media Vehicle for Social Browsing, Proceedings of CSCW '88, pp.25-38 (1988)
21）安田浩編：ヒューマンインタフェースに関する調査・研究報告書（平成３年度遠隔情報通信研究会分散オフィス推進のための技術課題研究）、日本サテライトオフィス協会（1992）
22）仲谷美江・西田正吾・坂口敏明・後藤卯一郎：劇場モデルに基づいたソフトウェア意図伝達支援ツールＣＯＭＩＣＳ、情報処理学会論文誌、Vol.31、No. 1、pp.125-136（1990）
23）Schacter,S., Ellerston, N., McBride, D. and Gregory, D. : An experimental study of cohesiveness and productivity , Human Relations, Vol. 4, pp.229-238 （1951）
24）西田春彦・新睦人：社会調査の理論と技法（Ｉ）、川島書店（1980）
25）安藤公平・大村政男・花沢政一・佐藤誠：心理検査の理論と実際、駿河台出版（1979）

《付録1》　表6．1　生産性要因とインフォーマルコミュニケーションとの関係

（番号は図 6.2 の Type1～Type5 に対応している。）

生産性要因	要因を構成する要素	要素の内容	各要素を実現するためのｺﾐｭｹｰｼｮﾝの例
個人効率	能力	学習（研修、模倣） 経験を積む	②（指導する、相談にのる） ③（ディスカッションする）
	モチベーション モラール （集団の規範）	心理的報酬 課題水準の向上 帰属意識 人間関係（仲間意識） リーダーシップ	①②（期待の言葉「君にならできるよ」、 評価の言葉「よくがんばってくれたね」など） ①（趣味や家族の話をする、経験談を話すなど 全人格的な付き合いをする、 メンバーだけの会合を開いて親睦を深め 仲間意識や対人魅力を形成する）
情報伝達効率 正確さ	メンバーの相互理解	メンタルモデル形成	③（共有経験を持つ、ディスカッションなどで 相手の思考プロセスを把握する）
	適切な方法で伝達	近接している 出会いの頻度が高い	④（音声、図面、ジェスチャーなどを総合して 相手が納得するまで説明する、確認も容易） ⑤（標準化記述の利用）
速さ	心理的距離	人間関係（気軽さ） 人間関係（信頼関係）	①（話しやすい雰囲気づくり、挨拶などの励行） ①②（指導する、相談にのる） ③（ディスカッションする）
	物理的距離	近接している 出会いの頻度が高い	④（伝達、確認が容易） ⑤（連絡網などの制度を充実させる） ②（メンバーが集まる機会を設定する、 問題の早期発見のため、普段から観察をする）

《付録２》　　表６．２　評価結果の解釈のポイント

プロジェクトの構成や地理的な条件を考慮にいれて判断する。

評価の着眼点	解釈
［ミクロな評価］	
コミュニケーションが行なわれていない、または極端に少ない所がある	席が離れている、組織が違うなど、 コミュニケーションしにくい理由がある。 そうでなければ、気軽に話せない人間関係になっている。
一箇所に集中して コミュニケーション量が多い	プロジェクトの中で一部のメンバーだけが仲がよい。 全体としてまとまりにくい。
コミュニケーション相手と まったく異なる評価がでている所	コミュニケーションが失敗している。 意図や気持ちが通じていない。
無回答が集中しているところ	人間関係に問題がある。
［マクロな評価］	
表情 ・どの部分が目だっているか ・変化のない部分はあるか	顔の雰囲気がプロジェクトの雰囲気になる。 目が笑っていないときはプロジェクトの「交わり」要素が少ない危険あり。 眉が平坦なときは「伝達」要素が少ない危険あり。
各要素の線の太さ	要素の線が細いときはそのパターンのコミュニケーション量が少ない。
顔の広さ、表情の狭さ	顔が広いときはプロジェクト全体のコミュニケーション量が多い。 表情が小さいときはコミュニケーションが少ない。 顔が広くて表情が小さいときは、多い所と少ない所があって仲間割れしている危険あり。
［グルーピング評価］	
グループ内の表情と外へ向いた表情の 比較	グループ内の問題、グループ間の問題をそれぞれ解釈。 内と外の顔が違っているときは、閉鎖的なグループと考えられる。
席のまとまり、組織上のまとまり、 上下関係、など様々なグループ構成で 比較	組織間の特徴、

《付録３》　質問紙抜粋

２．その人達の席とあなたの席とはどのくらい離れているでしょうか？
（一日のうち、もっとも長い時間座っている席について考えてください）

	[コミュニケーションの相手] Aさん	Bさん	Cさん	Dさん	Eさん
１．席についたままで話ができるくらい	1	1	1	1	1
２．相手の姿は見えないが、声は聞こえる	2	2	2	2	2
３．同じ部屋にいる	3	3	3	3	3
４．部屋は別だが、気軽に会いに行ける距離	4	4	4	4	4
５．部屋が離れていて、出かけるのが面倒な距離	5	5	5	5	5
６．交通手段を使わなければ会えない距離その１（例：大阪と神戸）	6	6	6	6	6
７．交通手段を使わなければ会えない距離その２（例：大阪と北海道、海外）	7	7	7	7	7
８．その他（　　）	8（　）	8（　）	8（　）	8（　）	8（　）

３．その人達とはどんな話をしていますか？
［Aさんとは］

	少ない						多い
１．仕事（方針、仕様など）							
２．仕事（資料の場所、技術的知識など）							
３．雑務（事務処理、コピーのとり方など）							
４．世間話（社内外の人事動向など）							
５．世間話（政治スポーツなど、新聞ネタ）							
６．個人的なこと（家庭、趣味）							
７．その他（　　）							

４．その人達との会話は予定されているものですか？

４－１．会話の始まりは

	[コミュニケーションの相手] Aさん	Bさん	Cさん	Dさん	Eさん
１．定期的に行なうことが多い	1	1	1	1	1
２．たいてい次回の日時を決めている	2	2	2	2	2
３．予定はないがだいたい推測できる	3	3	3	3	3
４．必要が生じたときに行なうことが多い	4	4	4	4	4
５．偶然居合わせたときに始まることが多い	5	5	5	5	5
６．その他（　　）	6（　）	6（　）	6（　）	6（　）	6（　）

４－２．会話の内容は

	[コミュニケーションの相手] Aさん	Bさん	Cさん	Dさん	Eさん
１．毎回決まっている（進捗報告など）	1	1	1	1	1
２．次回の話題を決めている	2	2	2	2	2
３．予定はないがだいたい推測できる	3	3	3	3	3
４．そのとき生じた問題について	4	4	4	4	4
５．その場の状況で変わる	5	5	5	5	5
６．その他（　　）	6（　）	6（　）	6（　）	6（　）	6（　）

１１．その人達とのコミュニケーションは今のままでいいと思いますか？

《いくつでも》	[コミュニケーションの相手] Aさん	Bさん	Cさん	Dさん	Eさん
１．今のままで十分だと思う	1	1	1	1	1
２．仕事に関する会話を増やした方がいい	2	2	2	2	2
３．仕事に関する会話を減らした方がいい	3	3	3	3	3
４．仕事以外の会話を増やした方がいい	4	4	4	4	4
５．仕事以外の会話を減らした方がいい	5	5	5	5	5
６．コミュニケーションのやり方に改善の余地がある	6	6	6	6	6
７．その他（　　）	7（　）	7（　）	7（　）	7（　）	7（　）

１３．思い通りに情報が伝わらなくて仕事上のトラブルが起こったことがありますか？
（例えば、依頼したことがこちらの意図していたとおりにならなかった、
逆に相手の意図するところがはっきりわからない、
話し合って納得していたつもりが誤解していた、　　など）

	少ない						多い
［Aさんとの会話では　］							
［Bさんとの会話では　］							
［Cさんとの会話では　］							
［Dさんとの会話では　］							
［Eさんとの会話では　］							

もし、さしつかえなければ、どのようなトラブルがあったのか、教えてください。

第7章　結論

本論文はソフトウェアの生産性向上を最終目標とした、ソフトウェア開発プロジェクトにおけるコミュニケーション支援に関するいくつかの研究をまとめたものである。

各章ごとに研究成果の要点をまとめる。

[第2章]

ソフトウェア開発プロジェクトを支援する従来の研究を整理した。

（1）ソフトウェア工学では、生産性向上研究は個人の生産技術の開発とプロジェクト管理手法の開発に大別できる。生産技術の研究では言語とツールの開発が一体となって進められてきた。近年は上流工程から下流工程までを一貫して計算機上で支援する動きが主流である。管理技術の研究では原価管理、品質管理など資源の管理の他に、外注管理、メンバー構造など人間管理の研究がある。メンバー間のコミュニケーション支援は、標準化により曖昧さ・個人差を排除することが主流になっている。

（2）協同作業支援研究（Computer-Supported Cooperative Work）の分野では、ソフトウェア開発プロジェクトを対象としたものには、開発プロセスをモデル化して支援機能を提案する研究と、マルチメディアを用いたコミュニケーションツールを開発プロジェクトに応用した研究がある。

（3）ソフトウェア工学では、プロジェクトの人間的側面に焦点を当てた研究は少なく、メンバー間の協同作業プロセスに関して体系だった支援は行なわれていない。ＣＳＣＷでは、協同作業プロセスを調査・モデル化した研究はあるが数は少なく、対象となるプロジェクトも限られていて、コミュニケーションの様々な側面をとらえるには不十分である。支援ツールの主流はマルチメディアの適用にとどまっている。またコミュニケーション評価に関する研究は行なわれていない。

[第3章]

実働の開発プロジェクトに対して行なった調査結果から、コミュニケーションの問題を整理し、第4章～第6章の研究の目的とアプローチを明らかにした。

（1）ソフトウェアの適切な表現方法がないため、プロジェクトのメンバーの意図の統一がはかりにくい。

（2）大規模なプロジェクトでは、小規模なプロジェクトには見られなかったコミュニケーションの問題がある。知識の分散の問題もその一つである。

（3）コミュニケーションの重要性は認識されているが、評価する手段がないため明確な問題として表面化せず、体系的な支援が行なわれない。

（4）以上の3つの問題は従来は扱われていなかった。各々に対し、認知科学・組織論・社会心理学のアプローチで支援を行なう。

[第4章]

ソフトウェアの意図を伝達するための表現方法を提案し、それを実現するシステムを開発した。研究の要点は以下のとおりである。

（1）人間の認知メカニズムには以下の特徴があり、ソフトウェアを理解するときにも機能と構造、部分と全体の関係を見せることや因果関係を示すことが重要である。

・文脈の中で現象を捉える

・まとまりを見ようとする

（2）上記のような特徴を持った表現方法として劇場モデルを提案する。ドラマとソフトウェアは構造が類似しており、ソフトウェアをドラマのように提示して説明できる機能が必要である。

（3）劇場モデルを実現したソフトウェア意図伝達支援システムを開発した。これはソフトウェアの構成要素をノードとリンクで表し、要素間の相互作用プロセスや設計プロセスを紙芝居のように1幕ずつ表示していく機能を持つ。

（4）本システムは設計者からプログラマへ、開発チームから保守チームへ、プログラマ同士のディスカッションで、など様々なコミュニケーション場面で説明ツールとして使用することができる。

[第5章]

大規模プロジェクトにおける障害発生時のコミュニケーションプロセスを分析し、コミュニケーション先をアドバイスするシステムを開発した。研究の要点は以下のとおりである。

（1）プロジェクトにおける障害発生時のコミュニケーションプロセスを分析し、障害発生から解決までには複数の役割（発見者・解決者・情報の保有者・解決策の実行者・報告先）が関わることが明らかになった。

（2）役割間の4つのコミュニケーション経路（①発見者→解決者、②解決者→情報収集先、③解決者→実行担当者、④解決者→報告先）を6種類のデータ

（メンバー構成、システム構成、担当マップ、権限マップ、知識マップ、障害分類）により決定するトラブルコミュニケーションモデルを提案する。

（3）トラブルコミュニケーションモデルにもとづき、障害が発生したときのコミュニケーションプロセスをシミュレートするコミュニケーション支援システムを開発した。

（4）本システムは以下の場面での利用が考えられる。

・プロジェクトに障害が発生したときに連絡先をアドバイスする

・権限や担当を変更したときのコミュニケーションプロセスの変化を予測する

・障害をランダムに発生させてコミュニケーションをシミュレートし、コミュニケーション分布の解析に用いる

[第6章]

プロジェクトにおけるコミュニケーションを分析し、その評価枠組みを提案して具体的な評価手法を開発した。研究の要点は以下のとおりである。

（1）プロジェクトにおけるコミュニケーションを整理し、コミュニケーションとソフトウェア生産性との関係を考察した。

（2）コミュニケーションには情報伝達を目的とする「伝達」的側面とコミュニケーションすること自体を目的とする「交わり」的側面がある。この2側面をもとにコミュニケーションを分類し、評価基準を作成した。

（3）上記の分類をもとに質問紙を作成し、集計方法と結果の表示方法を考案した。

（4）質問紙を2つのプロジェクトに実施し、集計例や結果の表示例を示すと共に妥当性を検証した。

今後に残された課題としては以下のものがある。

（1）組織間コミュニケーションの支援

本論文ではプロジェクト内のメンバー間のコミュニケーションを支援している。しかし近年では大規模な組織同士が協同で研究開発する場合も珍しくなく、今後は組織間コミュニケーション支援のニーズが高まると考えられる。このようなニーズに対し、社会学や組織学などを用いた新たなアプローチを開拓する必要がある。

（2）インフォーマルコミュニケーションの支援

本論文ではインフォーマルコミュニケーションの重要性を指摘し、その評価手法の提案を行った。今後、インフォーマルコミュニケーションに対してどのよ

うな支援をすべきか、その実現方法は何か、についての検討が残されている。

（3）他分野への応用

本論文はソフトウェア開発プロジェクトを対象としたコミュニケーション支援であった。今後は同様のアプローチを他の分野にも適用する、または他の分野で用いられているアプローチをソフトウェア開発にも応用する、という可能性が残されている。

謝辞

本研究は、筆者が所属する三菱電機（株）先端技術総合研究所（元中央研究所）で行なったものを大阪大学基礎工学部西田正吾教授の御指導のもとにまとめたものです。本研究は多くの方々のお力添えを得て初めて所定の結果をあげることができました。ここにお名前を書き記してあらためて感謝の意を表します。

本研究の機会をお与え頂くとともに、心温まる励ましを頂いた三菱電機（株）中央研究所元所長馬場準一博士、同研究所システム基礎研究部元部長上村勝彦博士、同研究所システム基礎研究部前部長武田捷一博士、先端技術総合研究所基礎研究部門統括兼システム情報研究部部長渡邊治氏、中央研究所システム基礎研究部第４グループ元グループマネージャ坂口敏明博士に深く感謝いたします。特にシステム基礎研究部第４グループ前グループマネージャ西田正吾博士には研究の立案、遂行方法を始め、終始並々ならぬご指導とご鞭撻を賜わりました。あらためて御礼申し上げます。

本論文をまとめるにあたり、詳細なご検討と貴重なご示唆を賜わりました大阪大学基礎工学部井口征士教授、宮原秀夫教授、馬野元秀助教授に厚く御礼申し上げます。本論文の内容について懇切なるご指導とご助言を賜わりました東京大学工学部原島博教授に厚く御礼申し上げます。

本研究の遂行および作成にあたって温かい御援助と御協力をいただいた同社先端技術総合研究所システム情報研究部の先輩、同僚諸氏に深く感謝いたします。さらに本研究を遂行するうえで多大な御協力をいただいた同社制御製作所の方々に深く感謝いたします。

最後に、研究生活、家庭生活ともに心身両面から支え続けてくれた同社産業システム研究所プラントシステム開発部仲谷善雄博士に心より感謝いたします。

著者の発表論文（本研究に関連する論文）

[論文]

1. 仲谷美江・西田正吾・坂口敏明・後藤卯一郎：劇場モデルに基づいたソフトウェア意図伝達支援ツールＣＯＭＩＣＳ、情報処理学会論文誌、Vol.1、No. 1、pp.124-135 (1990)
2. Nakatani, M. and Nishida, S. : Trouble Communication Model in a Software Development Project, 電子情報通信学会英論文誌, Vol.E75-A, No.2, pp.196-206 (1992)
3. 仲谷美江・原島博・西田正吾：ソフトウェア開発プロジェクトにおけるインフォーマルコミュニケーションの評価手法、電子情報通信学会論文誌D-II、Vol.J77-D-II、No.4、pp.823-837 (1994)

[国際学会発表論文]

1. Nakatani, M. and Nishida, S. : Computer-based Intention Communication System to Support Cooperative Work in Software Development, コンピュータワールド'８９, pp.151-158 (1989)
2. Kishimoto, M. and Nishida, S. : Computer-based Collaboration for Software Develpment, the Third International Conference on Human-Computer Interaction, Sept.18-22, Boston, Massachusetts, pp.43-50 (1989)
3. Nishida, S. and Nakatani, M. : Computer-based Intention Communication System For Software Development Based On Theater Model, Tenth European Meeting on Cybernetics and Systems Research, April 17-20, Vienna, Austria, pp.701-708, (1990)
4. Nakatani, M. and Nishida, S. : Communication Support System in A Software Development Project Based on Trouble Communication Model, WORK WITH DISPLAY UNITS '92 (1992)
5. Nakatani, M. and Nishida, S. : A Group Communication Support System Based on Organizational Approach, コンピュータワールド'９２ (1992)
6. Takeda, S., Nakatani, M. and Nishida, S. : Group Communication Support System for Software Development Project Based on Trouble Communication Model, the Fifth International Conference on Human-Computer Interaction, Vol.2, Aug.8-13, Orlando,

Florida, pp.961-966 (1993)

8. Nakatani, M., Harashima, H. and Nishida, S. : An Evaluation Method of Communication in a Software Development Project and its Application for Diagnosis, the Sixth International Conference on Human-Computer Interaction, Vol.2, July.9-14, Yokohama, Japan, pp.365-370 (1995)

[解説]

1. 仲谷美江・西田正吾：ＣＳＣＷ――その動向と研究事例――、画像ラボ、Vol.2、No.3、pp.33-36 (1991)
2. 仲谷美江・西田正吾：インフォーマルコミュニケーション研究の動向、計測と制御、Vol.33、No.3、 pp.214-221 (1994)
3. 仲谷美江：ソフトウェア開発プロジェクトにおけるインフォーマルコミュニケーションについて、計測と制御、Vol.34、No.12 (1995)（掲載予定）

[講演]

1. 岸本美江・西田正吾・坂口敏明：組織的なソフトウェア生産における意図の伝達の重要性について、認知科学会第４回大会、pp.94 (1987)
2. 岸本美江・西田正吾・後藤卯一郎：ソフトウェア生産プロセスにおけるインターラクションの分析、情報処理学会全国大会 第35会大会、pp.1141-1142 (1987)
3. 岸本美江・西田正吾・坂口敏明：ソフトウェア開発プロセスにおけるコミュニケーション・サポートにむけての一考察、第３回ヒューマンインタフェースシンポジウム、pp.403-406 (1987)
4. 岸本美江・西田正吾・坂口敏明：協同作業における意図の伝達の難しさとそのサポートについて――――ソフトウェアを対象にして、認知科学会 第５回大会、pp.80-81 (1988)
5. 岸本美江・西田正吾・後藤卯一郎：ソフトウェア意図伝達支援ツールＣＯＭＩＣＳ（１）理論的背景」情報処理学会全国大会 第37会大会、pp.857-858 (1988)
6. 岸本美江・西田正吾・後藤卯一郎：ソフトウェア意図伝達支援ツールＣＯＭＩＣＳ（２）ツールの開発、情報処理学会全国大会 第37会大会、pp.859-860 (1988)
7. 岸本美江・西田正吾・後藤卯一郎：ソフトウェア意図伝達ツールＣＯＭＩＣＳの開発」第４回ヒューマンインタフェースシンポジウム、pp.55-60 (1988)

8. 岸本美江・西田正吾・後藤卯一郎：ソフトウェア開発プロジェクトにおける協同作業の支援－－－ソフトウェア意図伝達支援システムＣＯＭＣＩＳの開発－－－、日本ソフトウェア科学会ソフトウェア研究会、pp.21-28 (1989)
9. 岸本美江・西田正吾・後藤卯一郎：ソフトウェア意図伝達ツール『ＣＯＭＣＩＳ』、情報処理学会ソフトウェア工学研究会64-19、pp.145-152 (1989)
10. 仲谷美江・西田正吾：『視点』を用いたソフトウェアのイメージ表現について、認知科学会 第６回大会、pp.46-47 (1989)
11. 仲谷美江・西田正吾：ソフトウェア意図伝達に対するＣＳＣＷの視点からの考察、第５回ヒューマンインタフェースシンポジウム、pp.331-336 (1989)
12. 仲谷美江・西田正吾：協同作業におけるコミュニケーション支援に向けての一考察、電子情報通信学会ヒューマンコミュニケーション研究会、HC90-22、pp.51-57 (1990)
13. 仲谷美江・西田正吾：プロジェクトの状況を考慮したコミュニケーションモデル、認知科学会 第８回大会、pp.56-57 (1991)
14. 仲谷美江・西田正吾：ソフトウェア開発プロジェクトにおける問題解決型コミュニケーションのモデル化、情報処理学会全国大会 第43会大会、Vol.5、pp.294-295 (1991)
15. 仲谷美江：ソフトウェア意図伝達支援システムＣＯＭＣＩＳ、第７回ヒューマンインタフェースシンポジウム講習会資料、pp.105-110 (1991)
16. 仲谷美江・西田正吾：ソフトウェア開発プロジェクトにおけるトラブルコミュニケーションモデル－－－ソフトウェア開発プロジェクトにおける問題解決プロセス、第７回ヒューマンインタフェースシンポジウム、pp.379-384 (1991)
17. 仲谷美江・西田正吾：インフォーマルコミュニケーションを考える、認知科学会第９回大会、pp.54-55 (1992)
18. 仲谷美江・西田正吾：トラブルコミュニケーションモデルに基づいたソフトウェア開発プロジェクト支援システム、電子情報通信学会ヒューマンコミュニケーション研究会、HC92-26 (1992)
19. 仲谷美江・西田正吾：グループコミュニケーション支援システムＣＡＣＴＵＳの開発、第８回ヒューマンインタフェースシンポジウム、pp.657-660 (1992)
20. 仲谷美江・西田正吾：ソフトウェア開発プロジェクトにおけるコミュニケーション支援システム、機械学会第２回設計工学・システム部門講演会論文集、pp.360-365 (1992)

21. 仲谷美江・原島博・西田正吾：ソフトウェア開発プロジェクトにおけるインフォーマルコミュニケーションの調査・分析手法、電子情報通信学会ヒューマンコミュニケーション研究会, HC92-64 、pp.41-46 (1993)
22. 仲谷美江・原島博・西田正吾：インフォーマルコミュニケーション評価に関する考察」電子情報通信学会ヒューマンコミュニケーション研究会, HC94-55 (1994)
23. 仲谷美江・原島博・西田正吾：ソフトウェア開発プロジェクトにおける評価手法の提案、第１０回ヒューマンインタフェースシンポジウム、pp.595-602 (1994)

著者の発表論文（本研究以外の論文）

［その他の発表］

1. 岸本美江・仲谷善雄・武田捷一：テレビゲームにおける問題解決プロセスの分析、第２回知識工学シンポジウム、pp.57-62（1984）
2. 岸本美江・仲谷善雄・武田捷一：コンピュータ機器の『使いやすさ』に関する一考察－－－アンケート調査より－－－、第３１回情報処理学会全国大会、pp.1473-1474（1985）
3. 岸本美江・仲谷善雄・武田捷一：ＯＡ機器の『使いやすさ』の心理的評価手法の提案、第１回ヒューマン・インタフェース・シンポジウム、pp.183-190（1985）
4. 岸本美江・仲谷善雄・武田捷一：ワープロのモード変換に関する考察、第３２回情報処理学会全国大会、pp.1835-1836（1986）
5. 岸本美江・仲谷善雄・武田捷一：ユーザの立場から見たワープロの訂正方法について、情報処理学会日本語文書処理研究会、Vol.7-3（1986）
6. 岸本美江・仲谷善雄・武田捷一：何をもって『使いやすい』と評価するのか？－－－エキスパートと初心者の差について－－－、日本認知科学会第３回大会、pp.48（1986）
7. 藤原良一・河野良之・岸本美江：バーバルプロトコルを用いた知識獲得実験、日本認知科学会第３回大会、pp.34（1986）
8. 仲谷美江・山岡孝行・細野義久・西田正吾：階層型組織における協調型意志決定支援－－－リアルタイム問題解決を対象として－－－、電子情報通信学会ヒューマンコミュニケーション研究会、HC93-21、pp.7-14（1993）
9. 仲谷美江・西田正吾：リアルタイム問題解決における協調型意思決定支援について、認知科学会第１２回大会、pp142-143 (1995)