03—R007

グループワーク支援システムの
研究開発報告書

平 成 4 年 3 月

JIPDEC

財団法人 日本情報処理開発協会

この報告書は、日本自転車振興会から競輪収益の一部である機械工業振興資金の補助を受けて平成3年度に実施した「グループワーク支援システムの研究開発」の成果の一部をとりまとめたものであります。

はじめに

近年、ハードウェア技術の進歩による価格の低廉化やソフトウェアの進歩により、ワークステーション(WS)やパーソナルコンピュータ(PC)の大幅な普及がもたらされてきた。

また、ローカルエリアネットワーク(LAN)や広域ネットワーク(WAN)等の通信技術の開発は、地理的、時間的に離れている数多くのWSやPCの保有者をお互いに結び付け、リソースの交換、共有を可能にさせている。

今迄のコンピュータの利用方法をみると大量データの高速処理といった作業が圧倒的に多かった。しかし、オフィスや生産現場等の集団社会では、データを中心にした処理のほかに、伝達・調整・合意といったグループの協調作業や会議・レビュー・協同執筆等の協同作業が頻繁に行われ、そのウェイトは非常に高いものになっている。また、近年における組織のグローバル化、地域分散化においては、こうした作業の効率化が極めて重要な課題となっている。したがって、従来の高速処理機能面に注目したコンピュータ利用やデータ交換的なネットワークの利用の他に、組織やグループの協調そのものを支援する技術を開発することができるならば大幅な生産性の向上が期待できる。

こうした認識のもとに、当協会では、平成３年度より２年計画で、グループの協調作業をコンピュータを用いて支援するための「グループワーク支援システムの研究開発」事業を実施した。

事業の実施に当たっては、グループワークを支援するシステムについて学会論文を中心に広く調査すると共に、各界の代表者からなるワーキンググループ(主査：斎藤信男　慶應義塾大学環境情報学部教授)を設置し、ソフトウェアの分散開発環境とグループワークの問題を取り上げ幅広く検討した。また、平成３年９月には海外調査員を派遣し、大学、先進研究機関の訪問調査を実施した。

さらに、グループワーク支援システムの構成方法、可能性、問題点等を明らかにするために実験環境を構築し、実験を進めると共に各種検討を行った。

本報告書は、これらの研究開発成果を中間報告として取りまとめたもので、３章から構成されている。

第１章では、ソフトウェアの分散開発の問題を取り上げ、プロセスモデル、プロダクトモデルの考え方、CASE環境におけるグループワーク支援機能、プロジェクト管理等について述べている。

第２章では、現在我が国で研究開発中のエージェント指向電子メールシステム、マルチメディア在席会議システム、分散開発支援環境等を取り上げ、そのグループワーク支援機能について述べている。

第３章では、グループウェアあるいはCSCWに関する海外の文献の中から代表的なものを２２編選び、その概要を紹介している。

最後に、本研究開発に当たって、ご指導ご協力を頂いた、斎藤主査を始め各委員の方々及び関係各位に深甚なる謝意を表わすものである。

平成４年３月

財団法人 日本情報処理開発協会

「グループワーク支援システムに関する調査研究ワーキンググループ」メンバー名簿

（敬称略）

主　査	斎藤 信男	慶應義塾大学　環境情報学部　教授
委　員	青山 幹雄	富士通（株）交換事業本部複合交換機事業部 第一ソフトウェア部第二開発課　技師補
委　員	筏井 幸夫	（株）SRA技術本部先端技術開発部　課長
委　員	井口 俊則	日本ナレッジインダストリ（株） 品川システムセンター　第２グループ　部長
委　員	小川　裕	日本電信電話（株）NTTソフトウェア研究所 ソフトウェア開発技術研究部　主幹研究員
委　員	黒沢　隆	日本アイ・ビー・エム（株）東京基礎研究所 対話アプリケーション担当　課長
委　員	阪田 史郎	日本電気（株）C&Cシステム研究所 ネットワーク研究部　部長代理
委　員	津田 道夫	（株）日立製作所　ビジネスシステム開発センタ 第二開発部　主任技師
委　員	中村 英夫	（株）東芝　システム・ソフトウェア技術研究所 システム・ソフトウェア開発推進部　課長
委　員	船川 浩伸	東京電力（株）技術開発本部システム研究所 制御研究室　主席研究員
委　員	三木 亮二	新日鉄情報通信システム（株）技術開発部 研究開発担当係長
委　員	向山　博	（財）日本情報処理開発協会　開発研究室　主任研究員
執筆協力	落水浩一郎	静岡大学　工学部　情報知識工学科　教授
執筆協力	松本 吉弘	京都大学　工学部　情報工学教室　教授
事務局	土川　潤	（財）日本情報処理開発協会　開発研究室　係長
事務局	中野 弘之	（財）日本情報処理開発協会　開発研究室

ワーキンググループ開催状況

第１回ワーキンググループ　（平成３年１１月１８日(月)）

主な議題

(1) ワーキンググループの進め方について

(2) CSCW及びGDSS関係の文献紹介

第２回ワーキンググループ　（平成３年１２月１０日(火)）

主な議題

(1) ヒアリング「ソフトウェア開発における協調作業支援についての課題」
（静岡大学 落水教授）

(2) 「ソフトウェア開発環境とグループウェア」についての検討

第３回ワーキンググループ　（平成４年１月２３日(木)）

主な議題

(1) ヒアリング「CASE環境におけるグループ・コミュニケーション支援」
（京都大学 松本教授）

(2) 「分散開発環境とグループウェア」についての検討

第４回ワーキンググループ　（平成４年２月１７日(月)）

主な議題

(1) 委員発表と討論「分散開発環境：新しい開発環境像を求めて、他」（青山委員）

(2) 委員発表と討論「ソフトウェア分散開発支援システム　D^2」（中村委員）

目　次

2．分散開発におけるグループワーク支援の実現例

1．分散開発とグループワーク支援

1．分散開発とグループワーク支援

1.1　分散開発の出現と発展

1.1.1　分散開発とは

『分散開発』とは、通常、図1.1-1に示すような2つの意味で用いられている。ひとつは、従来の集中ホストマシンに代わる、ワークステーション（WS）とネットワークから成る分散処理環境による開発であり、以下、『分散処理開発』と呼ぶ。一方、開発組織そのものが、地理的に離れた複数の開発拠点に分散する開発形態がある。これを、『分散組織開発』と呼ぶ。両者に共通の場合に、分散開発と呼ぶ。実際には、両方の分散形態が相互に推進力となり、分散処理環境上で分散組織による開発へ移行している [青山92]。

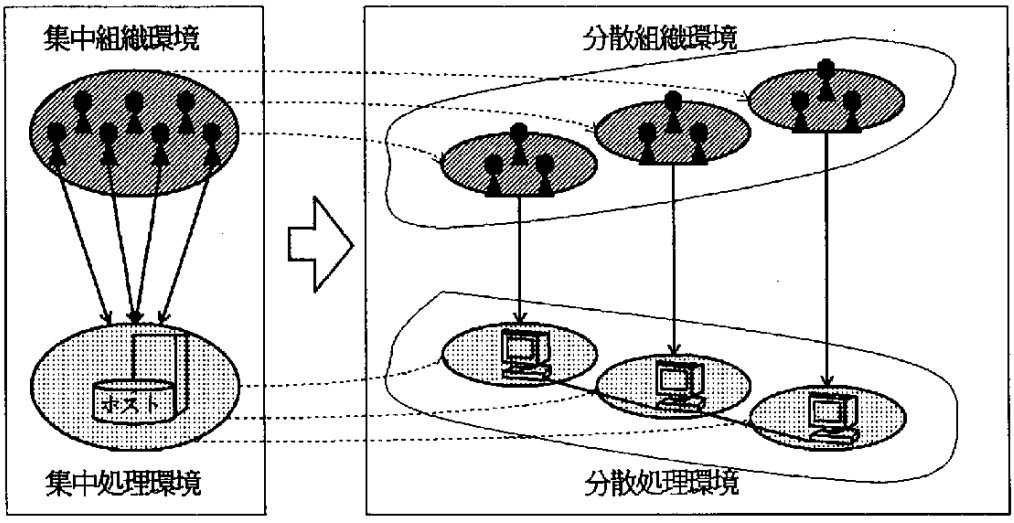


図1.1-1　分散開発環境

1.1.2　分散開発の出現の背景

ソフトウェア開発環境が分散化する背景には、次のような要因が指摘されている。

(1)　技術的背景

①　情報処理技術の変化

コストパフォーマンスの優れたWSと高速ネットワーク技術の発展を核として、今や、一人一人のソフトウェア技術者が一台のWSを占有する『一人一台のWS環境』の時代になっている [斉藤91, 松崎91, 長野92]。さらに、オフィス用、家庭用、携帯用と、一人で三台を使い分けることも可能である。このような分散処理環境の上に、CASE (Computer-Aided Software Engineering)、GUI (Graphical User

Interface)、グループウェアなどの優れたソフトウェア資産が蓄積されている。このようなソフトウェア資産を異なるハードウェアやOS上でも利用するために、基盤技術を標準化する重要性が認識されている。ソースレベルのインタフェースであるAPI (Application Program Interface) や各種プロトコルの国際標準やde facto standardが形成され、オープンアーキテクチャと呼ぶ、ハードウェアに依存しないソフトウェア環境が構築されるに至った。この結果、ソフトウェア開発の基盤環境の向上とその資産活用が推進できるようになった。

② 開発環境に対する要求の変化

従来のTSS (Time Sharing System) による集中処理環境におけるレスポンス時間がプログラム生産性に大きな影響を及ぼすことが実験的に指摘され、生産性向上のためにレスポンス時間の短縮が要求された [Thadhani84]。分散処理環境では、この問題を経済的に解消できる。さらに、WSの豊富な処理能力を活用し、開発者の能力を最高に発揮できる環境が望まれている。例えば、マルチウィンドウシステムは、個人レベルでの並行作業を可能にする。また、高解像度ディスプレイによるGUIの開発は、図形を活用した、いわゆる視覚的開発などの新しい開発方法の実現を促した。COCOMOモデルで比較した結果、WSの利用により30%の生産性向上を得たとの報告がある [Mital86]。

(2) 社会的背景

① ソフトウェア開発を取り巻く環境の変化

開発するソフトウェア規模の絶えざる増大 [亀田91] に伴い、ソフトウェア開発要員が依然不足している。多数の開発要員を集中して確保することが経済的にも、社会的にも困難となっているため、地域的に分散して開発する、分散組織開発が普及している。実際に、大規模ソフトウェア開発で、開発要員の50%以上が遠隔地に分散するような開発が行われている。

② ワークスタイルの変化

Druckerは、21世紀のワークスタイルは従来の階層的組織における指揮命令型からチームやグループを中心とするネットワーク型組織へ移行すると指摘している [Drucker88]。さらに、『テレコミューティング (Telecommuting)』[Cross88] や『テレワーク (Telework)』[大沢88] と呼ばれる、在宅勤務、サテライトオフィス、リゾートオフィスなどの新しいオフィス形態が導入されている [坂本91]。また、企業活動の国際化に伴い、ソフトウェアの国際分散開発も始まっている。分散組織開発は、ソフトウェア開発に、このような新しいワークスタイルをもたらすと期待されている。

1.1.3 分散開発環境の発展

分散開発環境は、技術面で、分散処理の発展に大きく依存してきた。ハードウェア面ではWSとネットワーク技術であり、ソフトウェア面では分散OSなどの基盤ソフトウェアの発展である。図1.1-2は、従来のTSSによる集中開発環境から分散開発環境への移行を示す。

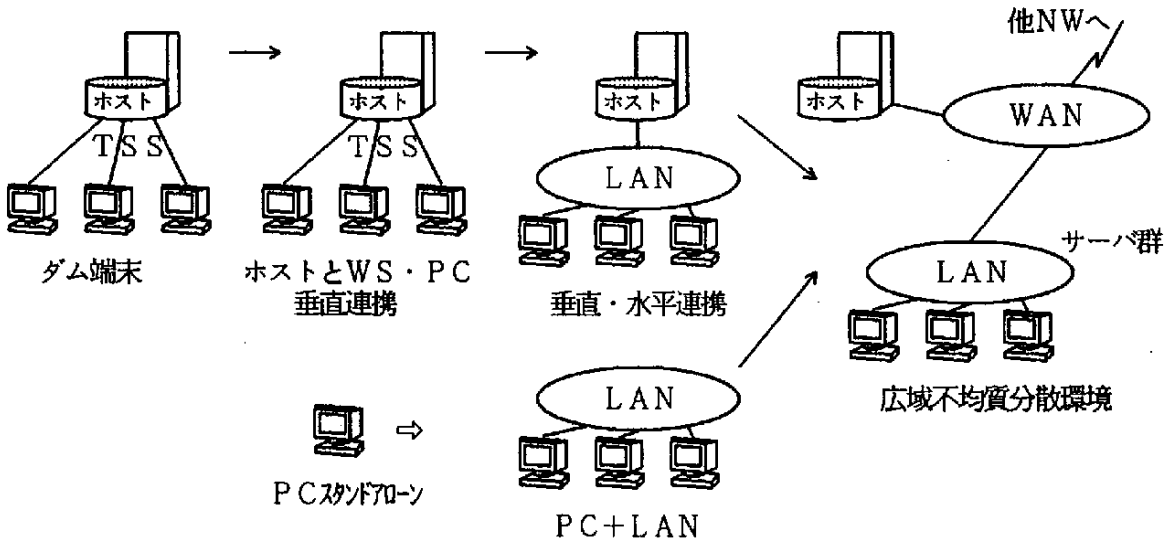


図1.1-2　分散開発環境への推移

(1)　ワークステーションの発展

前川らは、WS を「個人計算環境と分散ネットワーク環境を前提とする高機能計算機」と定義している[前川90]。このような WS のアイディアは1960年代に Engelbert らによって研究された [Goldberg 88]。その後、Xerox Palo Alto 研究所で1973年に開発された Alto により、現在の WS 原形が確立された。1982年には Apollo 社（現 Hewlett-Packard 社) や Sun Microsystems 社で、商用の WS が開発された。

一方、1981年に IBM 社が発表した IBM/PC は、いわゆるパーソナルコンピュータ（PC）が実務に広く利用される道を拓いた。初めはスタンドアローンで利用されていた PC も、近年、ネットワークを介してプリンタやファイルなどを共有するようになった。高機能 PC と WS とでは、性能の差がなくなりつつある。

(2)　ネットワークの発展

1969年から運用を開始した ARPANET は、パケット交換による「WAN (Wide Area Network)」の構築技術を確立し、分散開発環境における資源共有の道を拓いた。その後、多数の WAN が構築されている [Quarterman86]。パケット交換の考えは、Altoを接続するための「LAN (Local Area Network)」として開発された Ethernet に結実し、各種の LAN にも応用されている [Metcalfe76]。わが国でも、1980年代後半から大規模の LAN や WAN が構築されている [松方89, 村井91]。また、近年のいわゆるパソコン通信の普及により、電子メールが、郵便、電話や FAX に次ぐ通信メディアとして定着しつつある [山村87]。

(3) 分散開発環境の発展

WSとネットワークによる分散環境技術の可能性を探る目的で、いくつかの大学や研究機関において、大規模な分散環境を構築する試みが1980年代初めに始まった。例えば、カーネギーメロン大学（CMU）のAndrew（アンドリュー）プロジェクト [Morris86]、マサチューセッツ工科大学（MIT）におけるAthena（アテナ）プロジェクト [Champine90]、ケンブリッジ大学におけるCambridge Distributed Computing Environmentの開発などがある [Coulouris88, 村岡91]。1980年代後半には、このような実験的分散開発環境の成果がソフトウェア開発現場へ導入された。初期の事例として、Xerox社のCedar [Swinehart85]とMesa Programming Environment [Sweet85]やApollo社のDSEE (Domain Software Engineering Environment) [Leblang84]などが知られている。

近年は、オペレーティングシステムに止まらず、分散開発環境の基盤技術を統合した環境のモデルや製品が提供されている [IEEE91, OSF91, UI91]。

1.1.4 分散開発の利点と問題点

分散開発は、さまざまな利点をもたらすと考えられている。一方、開発者や開発資源が分散することによる問題点も指摘されている。分散開発を活用し生産性を向上するためには、分散による利点を伸ばしつつ、その問題点をいかに改善するかに懸かっている。

(1) 分散開発の利点

分散開発は、一人一人の開発者のレベルからプロジェクトや開発環境全体まで、次のような改善効果をもたらすと期待されている。

① 人に優しい環境

WSの強力な処理能力を活用し、一人一人の技術者の作業環境を改善する。レスポンス時間の改善、GUIによる操作性と情報表現力の向上、ネットワークによる情報の収集と交換の改善など。また、組織分散やサテライトオフィスなどは、通勤条件やオフィススペースなどの生活環境の改善も狙いとしている。

② オープンな環境

オープンな環境は、開発環境構築の導入コストや運用コストが低減でき開発規模の増大に応じて拡張や分散が可能となるなど、利用者にとって利点が多い。このため、次のような特性を満たすことが指摘されている [Hubley91, IEEE91, OSF91]。

(a) 互換性 (Compatibility)

国際標準などに準拠し、異なるインプリメンテーション（ベンダ）間や版数間でOSインタフェースが保証され、利用者の選択の幅が広がるとともに既存ソフトウェア資産やデータが活用できる。

(b) 可搬性 (Portability)

異なるシステムやOS間でアプリケーションの移植性が保証され、既存アプリケーション資産が活用できる。例えば、異なるアーキテクチャのシステム間でのAPIや同一アーキテクチャのシステム間でのABI (Application Binary Interface)の保証。

(c) スケーラビリティ (Scalability)

同一のアプリケーションが、WSからスーパーコンピュータまで異なる規模 (Scale) のシステム上で動作可能となる。

(d) 相互運用性 (Interoperability)

異なるアーキテクチャのシステム間でもネットワークで相互接続して、コミュニケーションや協調処理などが可能となる。

③ リライアブルな環境

組織や各種サービスの機能分散と負荷分散により、特定のサーバの障害などにより開発環境全体が停止するリスクが分散される。この結果、開発作業全体の中断を防止できる。

(2) 分散開発環境の問題点

分散開発環境では、開発作業の主体である人とWS、さらには成果物であるドキュメントや種々の情報が分散する。このため、開発者やグループ間での協調と情報の収集、交換などの作業オーバーヘッドと作業品質の低下が問題となる。Adamsらはこれを次の5つの問題に要約している [Adams89]。

① 複数の人間が関与することによる問題 (Multiple people problems)

大規模ソフトウェア開発では、多数の開発者やグループが共同して開発する。分散開発では、開発者やグループ間の情報の共有や作業の協調が困難となり、生産性や品質の低下を招くおそれがある。このため、組織や作業をどのように分散し管理するのか。また、分散組織間のコミュニケーションや情報共有をいかに支援するか、など多くの課題がある。

② 複数のオブジェクト（プログラムやデータ）を扱う問題 (Multiple object problems)

分散開発では、ソースコードやドキュメントなどの開発にかかわるオブジェクトが分散するため、変更の一貫性の保証や変更に伴う影響範囲の把握が困難となる。

③ 複数の版数を並行して開発保守する問題 (Multiple release problems)

大規模ソフトウェア開発では、しばしば、複数の版を保守する必要がある。また、分散した各開発者やグループが並行して複数の機能を開発する『分散並行開発』は分散開発の進展した開発形態として実行されている [青山90]。このような開発条件下では、複数の版の変更管理やシステムの構成管理が重要な問題となる。

④ 複数のマシンが関与することによる問題 (Multiple machine problems)

分散処理環境では、WS、サーバ、ホストなど多数のマシンが協調して動作する必要がある。これらのマシンの相互運用性やソフトウェアの移植性、保守性など、従来の集中型開発環境では問題となら

なかった新しい課題が提起されている。

⑤ 複数のツールを利用することによる問題 (Multiple tool problems)

ソフトウェア開発支援のためにさまざまなツールが開発され、実務に適用されている。既存のツール資産を無駄にすることなく分散開発環境へ移行したり、他のツールと円滑に統合したり連携できることが望まれている。

このような問題は、分散処理システムにおけるプロセスの同期や資源の管理問題と多くのアナロジがある。例えば、分散処理システムにおけるプロセッサへの機能分散と同様に、相互に密接な関係のあるプログラム群を異なる拠点で分散開発すると、拠点間のコミュニケーショントラヒックが増大する。また、分散組織環境においては、開発における情報の共有が困難となる。ドキュメント化されていない情報や、紙のドキュメントでは、相互に参照したり、リアルタイムに更新できない。このような、分散開発環境の問題点を実際の開発の中で調査した事例が報告されている。

1.1.5 分散開発の分類

分散開発の形態は、分散処理開発の形態と分散組織開発の形態の観点から次のように分類できる。

(1) 分散処理開発の分類

① ネットワーク形態による分類

(a) インハウス分散：集中組織で WS と LAN による分散処理環境を利用する形態。

(b) 都市（地域）内分散：同一都市内あるいはキャンパス内で複数の建物に分散する開発環境。このような環境では中距離の高速ネットワーク『MAN (Metropolitan Area Network)』により開発拠点が結合される。

(c) 広域分散：複数の都市あるいは国際的に分散した開発拠点を WAN で結んだ環境。

② 機能と負荷による分散

(a) 機能分散：作業の内容による分散。例えば、各サーバは、計算、ファイル管理、印刷など異なるサービス（機能）を提供する。

(b) 負荷分散：作業の負荷による分散。例えば、同一サービスを提供する複数のサーバの分散処理により実行時間を短縮する。

③ 均質分散環境と不均質分散環境

(a) 均質分散：同一アーキテクチャのシステムで構成される分散開発環境。

(b) 不均質分散環境：異なるアーキテクチャのシステムで構成される分散開発環境 [Notkin87]。

実際には、機能分散と負荷分散を組み合わせた、不均質な広域分散開発環境となる傾向にある。

(2) 分散組織開発の分類

組織分散における開発分担の観点から、次の２つに分類される。

① 組織機能分散

各拠点で、異なるプロセスやプロダクトを開発する。これは、更に次のように分類される。

(a) 垂直（プロセス）分散：開発プロセスによる分散。例えば、上流工程と下流工程を異なる拠点で開発する。

(b) 水平（プロダクト）分散：開発プロダクトやシステムによる分散。例えば、アプリケーションやサブシステム毎に異なる拠点で開発する。

② 組織負荷分散

各拠点で、同一工程や同一アプリケーションを分担して開発する。

一般的に、まず、下流工程を中心とする水平分散から始まり、その後、上流工程の分散へと発展していく傾向にある。

1.1.6 分散開発を支援するソフトウェア工学技術

ソフトウェア工学の観点から分散開発を支援するアプローチが研究、開発されている。特に、ソフトウェア開発活動を構成する主要素は開発プロセスとプロダクトであることから、分散開発の支援は次の２つのモデルで分類できる。

- プロセスモデル
- プロダクトモデル

(1) プロセスモデル

ソフトウェアの開発過程を形式的に表現する、『プロセスプログラミング』が提案されている[Osterweil87]。一方、グループによる共同作業を支援する枠組みとして『CSCW (Computer-Supported Cooperative Work)』[Greif88]や『グループウェア』[Johansen88, 石井89, 松下91]が提案されている。この２つのアプローチは、図1.1-3に示すように、分散開発を支援する上で、プロセス面から考える共通の視点を提供している。

まず、開発プロセスと開発環境に相似性があることから[Notkin88]、開発環境を構築するには、その開発活動を分析し、開発プロセスの構造を把握する必要がある。Winograd は、『デザインとはツールそのものを設計することではなく、そのツールの使用を通して引き起こされるワークスタイルの変化を設計することである』と指摘している[Winograd86]。開発プロセスと開発環境とが適合していなければ、所期の効果を達成できないだけでなく、生産性が低下する恐れさえある。

特に、分散組織開発では、各拠点の開発活動が自律的になるため、拠点毎の開発プロセスが異なる傾向にある。また、各拠点の開発活動は相互に関係するため、その動的相互作用は極めて複雑となる。

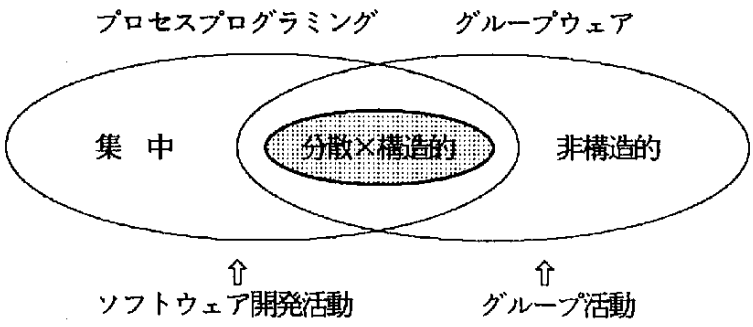


図1.1-3 プロセスプログラミングとグループウェア

このような問題を把握する上で、プロセスモデルは有効なアプローチを提供すると期待される。

一方、グループウェアは、ソフトウェア開発のみならず、グループによる広汎な活動を対象としている。このため、従来のグループウェアの研究、開発は、会議支援、意思決定、共同執筆など、ソフトウェア開発との関連はあるものの、ソフトウェア開発の抱える問題を直接扱うことは少なかった。ソフトウェアの分散開発ではグループ共同作業が極めて重要であることから、分散開発を直接支援するグループウェアの研究、開発の必要性が指摘されている [落水92, 垂水92]。

グループウェアによりグループの共同活動を支援するには、まず、グループの構造や活動をモデル化する必要がある。このモデルは、構造的モデルと非構造的モデルに分類される [石井91]。ソフトウェア開発の観点から解釈すると、それぞれ、定型的な開発作業と非定型的な開発作業に対応する。したがって、ソフトウェア開発におけるグループウェアの構造的モデルは、開発プロセスそのものに他ならない。

このようなアプローチに基づき、実際の開発プロセスをモデル化する研究が行われている [Aoyama 90, 稲田89, 望月90]。その結果、分散組織環境における開発プロセスの動的挙動が明確になるなどの有効性が報告されている。しかし、オフィス作業そのものがオープンであることから、モデルの適用範囲や限界も指摘されている [Hewitt86]。今後、分散開発を支援するために、実際の開発プロセスの分析や開発プロセスに基づく支援環境などの問題を検討する必要がある。

(2) プロダクトモデル

プロダクトモデルは、開発プロセスの入出力である成果物に着目する。分散開発環境では、プロダクトの生成、蓄積、参照、交換が分散する。さらに、プロダクトは相互に関係があるため、その管理や変更に伴う影響の波及は極めて複雑となる。

プロダクトには、ドキュメントやソースコードなどの恒久的なものと会議の議事録など一時的なも

のがある。分散開発環境におけるプロダクトの構造とその流れを分析することがプロダクトモデル構築の第一歩となろう。実際に、ソフトウェア開発における情報伝達を分析した事例 [福田90, 貫井90] や分散開発の管理支援システムの構築に適用した事例 [青山90] の報告がある。さらに、オブジェクト指向 DB により、プロセスとプロダクトを統一的に管理する環境も提案されている [Matsumoto90]。

また、日本的企業風土では、フォーマルな情報のみならず、インフォーマルな情報の共有も重要である [青木89]。

1.1.7 分散開発を推進するために

分散開発は実際のソフトウェア開発に急速に普及しつつあり、その問題を解決するためにさまざまな方面から研究、開発が進められている [坂下92]。特に、グループウェアや開発プロセスのアプローチは、組織や環境が『分散』する本質に関わるものであり、今後の活発な研究、開発が期待される。

【参考文献】

[Adams89] Adams, E.W., et al.: Object Management in a CASE Environment, Proc. 11th ICSE, pp. 154-163 (1989).

[青木89] 青木昌彦他: 日本企業グローバル化の研究, ＰＨＰ研究所 (1989).

[Aoyama90] Aoyama, M.: Distributed Concurrent Development of Software Systems: An Object-Oriented Process Model, Proc. IEEE COMPSAC '90, pp. 330-337 (1990).

[青山90] 青山幹雄他: 交換ソフトウェアの分散並行開発支援環境, 電子情報通信学会交換システム研究会, No. SSE90-25, pp. 7-12 (1990).

[青山92] 青山幹雄: 分散開発環境：新しい開発環境像を求めて, 情報処理,Vol. 33,No. 1, pp. 2-13 (1992).

[Champine90] Champine, G.A., et al.: Project Athena as a Distributed Computer System, IEEE Computer, Vol. 24, No. 9, pp. 40-51 (1990).

[Coulouris88] Coulouris, et al.: Distributed Systems, Addison-Wesley, (1988) [水野忠則（監訳）: 分散処理システム, 電気書院 (1991)].

[Cross86] Cross, T. B., et al.: Telecommuting: The Future Technology of Work, Richard D. Irwin, Inc. (1986) [松下　温 (訳): テレコミューティング, 近代科学社 (1988)].

[Drucker88] Drucker, P.: The Coming of the New Organization, Harvard Business Review, Jan.-Feb. 1988, pp. 45-53 (1988) [高度情報技術と情報ネットがもたらす未来型組織の構想, DIAMOND ハーバードビジネス, Apr.-May 1988, pp. 16-30 (1988)].

[福田90] 福田由紀雄他: 遠隔地ソフトウェア開発の実験,情報処理学会ソフトウェア工学研究会, No. 71-7 (1990).

[Greif88] Greif, I. (編): Computer-Supported Cooperative Work, Morgan Kaufmann (1988).

[Goldberg88] Goldberg, A. (ed.): A History of Personal Workstations, ACM Press (1988)[村井　純 (監訳): ワークステーション原典, アスキー出版局 (1990)].

[Hewitt86] Hewitt, C.: Offices are Open Systems, ACM Trans. Office Inf. Syst., Vol. 4, No. 3, pp. 271-287 (1986).

[Hubley91] Hubley, M.: Distributed Open Environments, BYTE, Vol. 16, No. 12, pp. 229-238 (1991).

[IEEE91] IEEE: Draft Guide to the POSIX Open Systems Environment, P1003.0, IEEE Computer Society (1991).

[稲田89] 稲田良造他: ソフトウェア開発過程の形式化とその詳細化による支援システムの作成－ＪＳＤを例として－,電子情報通信学会論文誌,Vol. J72-D-I, No. 12, pp. 874-882 (1989).

[石井89] 石井　裕: グループウェア技術の研究動向, 情報処理, Vol. 30, No. 12, pp. 1502-1508 (1989).

[石井91] 石井　裕: グループウェアのデザイン, bit, Vol. 23, No. 3, pp. 273-283 (1991).

[Johansen88] Johansen, R.: Groupware, The Free Press, (1988) [会津　泉 (訳): グループウェア, 日経ＢＰ社 (1990)].

[亀田91] 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).

[Leblang84] Leblang, et al.: Computer-Aided Software Engineering in a Distributed Workstation Environment, Proc. ACM Symp. on Practical Software Development Environments, pp. 104-112 (1984).

[前川90] 前川　守（編）：ワークステーション, 丸善, (1990).

[松方89] 松方　純: 大学における大規模ＬＡＮの構築, 情報処理学会論文誌, Vol. 30, No. 1, pp. 25-35 (1989).

[Matsumoto90] Matsumoto, Y., et al.: A Data Model in the Software Project Database KyotoDB, JSSST Adv. Software Sci. Tech., Vol. 2, pp. 103-121 (1990).

[松下91] 松下　温（編著）：グループウェア入門, オーム社, (1991).

[松崎91] 松崎　稔他: ホストなき世界の到来,日経コンピュータ,1991.10. 7号, pp. 48-106(1991).

[Metcalfe76] Metcalfe, R. M., et al.: Ethernet: Distributed Packet Switching for Local Computer Networks, Comm. ACM, Vol. 19, No. 7, pp. 395-403 (1976).

[Mital86] Mital, R.M., et al.: A Case Study of Workstation Usage During the Early Stages of the Software Development Life Cycle, Proc. ACM Symp. on Practical Software Development Environments, pp. 70-76 (1986).

[望月90] 望月純夫他: ソフトウェアプロセス－実時間システムにおけるケース・スタディ,情報処理学会ソフトウェア工学研究会,No. 71-18 (1990).

[Morris86] Morris, J.H., et al.: Andrew: A Distributed Personal Computing Environment, Comm. ACM, Vol. 29, No. 3, pp. 184-201 (1986).

[村井91] 村井　純他: 大規模分散環境ＷＩＤＥのアーキテクチァ, 情報処理学会マルチメディア通信と分散処理研究会, No. 51-4 (1991).

[村岡91] 村岡洋一他（編）: 知のキャンパス,bit, 1991年 4 月号別冊 (1991).

[長野92] 長野宏宣: 分散開発環境の基盤技術, 情報処理, Vol. 33, No. 1, pp. 14-21 (1992).

[Notkin87] Notkin, D., et al.: Heterogeneous Computing Environments, Comm. ACM, Vol. 30, No. 2, pp. 132-140 (1987).

[Notkin88] Notkin, D.: The Relationship between Software Development Environment and the Softwar Process, Proc. ACM Symp. on Practical Software Development Environments, pp. 107-109 (1988).

[貫井90] 貫井春美他: グループウェア支援機能の実験的考察,情報処理学会ソフトウェア工学研究会, No. 71-8 (1990).

[落水92]落水浩一郎: ソフトウェア開発におけるグループウェアの役割,ソフトウェアツールシンポジウム '92予稿集, pp. 83-92 (1992).

[大沢88] 大沢　光: テレワーキング革命,日本実業出版社 (1988).

[OSF91] OSF: Guide to OSF/1: A Technical Synopsis, O'Reilly & Associates, 1991.

[Osterweil87] Osterweil, L.: Software Processes are Software Too, Proc. 9th ICSE, pp. 2-13 (1987).

[Quarterman86] Quarterman, J. S., et al.: Notable Computer Networks, Comm. ACM, Vol. 29, No. 10, pp. 932-971 (1986), [村井　純 (訳): 世界のネットワーク, bit, Vol. 19, No. 3-7 (1987)].

[斉藤91] 斉藤信男 (編): ワークステーションと分散処理, コンピュートロール, Vol. 33,コロナ社 (1991).

[坂本91] 坂本幸保他: 志木サテライトオフィス第 2 期実験を振り返って,電子情報通信学会誌, Vol. 74, No. 3, pp. 240-244 (1991).

[坂下92] 坂下善彦: 分散開発環境の事例と今後の展望, 情報処理, Vol. 33, No. 1, pp. 32-39 (1992).

[Sweet85] Sweet, R.E.: The Mesa Programming Environment, ACM SIGPLAN Notices, Vol. 20, No. 7, pp. 216-229 (1985).

[Swinehart85] Swinehart, D. C., et al.: The Structure of Cedar, ACM SIGPLAN Notices, Vol. 20, No. 7, pp. 230-244 (1985).

[垂水92] 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).

[Thadhani84] Thadhani, A.J.: Factors Affecting Programmer Productivity during Application Development, IBM Syst. J., Vol. 23, No. 1, pp. 19-35 (1984).

[UI91] UI: UI-ATLAS Distributed Computing Architecture: A Technical Overview, UNIX International (1991).

[Wino86] Winograd, T., et al.: Understanding Computers and Cognitions, Addison-Wesley (1986) [平賀　譲（訳）：コンピュータと認知を理解する,産業図書 (1989)].

[山村87] 山村陽一他（編）：電子メールとグループ通信特集号, 情報処理, Vol. 28, No. 8, pp. 1014-1059 (1987).

1.2　グループの理解とグループウェア

グループワーク支援システム (グループウェア) 成功の鍵は支援対象であるグループそのものの理解にある。グループは個人、組織の中間に位置する。グループの多くは個人よりは社会的であるが、組織ほどフォーマルなものではない。このことから、グループは個人ともまた組織とも異なる特徴を持つ実体であると言える。したがって、個人に関する知識や組織に関する知識だけではグループワーク支援は成功しがたいと考えるのが自然である。グループとその内部のグループ・プロセスを明らかにし、そのプロセスと整合性をもつ支援システムが望まれる。グループとグループ・メンバーの間のコミュニケーション、グループ・メンバー間のコミュニケーション、リーダーシップ、コンフリクト管理などを支援する機能が求められる。

1.2.1　グループウェア

近年グループワークを支援するソフトウェアとしてグループウェアが一部で注目を集めている。グループウェアの定義は現状では必ずしも一定していないようである。ここでは、グループウェアはグループ間の協調を支援するソフトウェアではなく、グループ内の協調を支援するソフトウェアである、と考える。

1.2.2 グループ理解の重要性

グループウェアが近年注目されるようになった背景には、コンピュータ技術の発展、通信技術とコンピュータ・ネットワークの発展 (分散システムの発展)、ユーザ・インタフェース技術の発展、パーソナルなコンピュータ利用形態の進展、などがあることは否定できない。しかし、これらの技術は、グループウェア実現の手段を提供するにすぎなく、グループウェア技術の本質の問題ではない。

グループウェアに限らず、何かを支援するソフトウェアは支援対象の深い理解に基づくものでなければならない。会計システムは会計についての理解に基づくものであるべきことは当然であろう。応用分野知識の重要性が認識されつつある。グループウェアでの応用分野知識とはすなわち、グループ

に関する知識ということになる。また、システム開発の成否はそのシステムを導入する組織に関する知識の量にかかっている。その意味でもグループウェアではグループに関する知識が重要である。

ソフトウェア工学の世界では1970年代後半から1980年代にかけ、種々のソフトウェア開発方法論や支援ツールが提案されてきた。1980年代後半からはCASE (Computer-Aided Software Engineering) ツールが登場してきている。しかしながら、こういうソフトウェア開発方法論、ソフトウェア開発支援ツール、CASEツールは期待ほど効果をあげていない。このことを真摯にとらえている一群の人たちは、我々がそもそもソフト開発自体をよく理解していないことに気付き、ソフトウェア・プロセス理解の重要性を説いている。グループウェア技術の研究開発や発展にはグループの理解が中心課題になる。

1.2.3 グループワーク支援システム研究の中心課題

グループワーク支援システム研究の中心課題はグループの理解である。有効なグループワーク支援システムを開発するために、明らかにすべきは次のような事柄である。

(1) グループというものは一体何か？
(2) グループはどういう特徴を持っているか？
(3) グループ・プロセスはどうなっているか？
(4) グループ作業では何が問題になっているか？
(5) グループワーク支援システムが支援できるのはどの様な部分か？
(6) グループワーク支援システムを使うことでどの様な問題が解決できると期待できるか？
(7) グループワーク支援システムを導入するとグループ・プロセスはどう変化するか？
(8) グループワーク支援システムを導入したとき、その効果をどのように測定すべきか?

1.2.4 グループの特徴

グループは複数の個人が、最終的には統合された一つのあるいは複数の成果をめざして協調して作業する実体であり、組織ほどフォーマルなものではない。むしろ、組織レベルでは決められることだけが決めてあり、他の重要な部分はグループ・レベルに任されている、と考える方がよい。グループは単なる個人の集合ではないし、組織でもない。

(1) グループはソフト・システムである。

システム理論では、物理現象を支配ルールとするようなシステムはハード・システムと呼ばれる。ハード・システムではシステムの目的や評価基準は決めやすい。一方、人間を要素に含むシステムでは多くの人が同意できる明確な目的や評価基準を決めることが非常に難しい。このようなシステムはソフト・システムと呼ばれる。グループは人間中心のシステムであり、ソフト・システムである。グループの目的や評価基準を決めることは難しい。

(2) グループはダイナミックなシステムである。

① グループの課題は変化する。

② グループ・メンバーの役割はダイナミックに定まる。

③ グループ・メンバーの動機は時間とともに変化する。

④ グループはその外部環境の変化に影響を受ける。

(3) グループ・メンバーは多様である。

① グループ・メンバーの動機はメンバー毎に異なる。

② グループ・メンバーの能力や得意分野は異なる。

③ グループ・メンバーの経験は異なる。

(4) グループは成長する。

グループは徐々に成長する。グループ・メンバーも習熟し成長するが、グループも成長する。

(5) グループの運営はインフォーマルである。

組織レベルではフォーマルなプロセスが支配的であるが、逆にグループ・レベルではプロセスはインフォーマルである。メンバー間の合意を基本に柔軟に運営される。

1.2.5 グループ作業での問題

ソフトウェア開発現場でのグループ作業の問題を中心にして、グループ作業のもつ問題について記してみたい。グループ作業での問題は次のようなものがある。

(1) グループの課題が定まらない、あるいは不適切である。

グループ作業の最大の問題はこれであると思う。課題は明らかになっていると誤って信じているケースが多い。あるいは、課題は明確になっているべきとの考え方が世の中に行き渡っているので表向き課題は存在するが真の課題は不明なことが多い。

(2) グループ・メンバーの課題・役割が不明確である。

一致団結すれば何事も解決できる、に近い信念で運営されているグループが多すぎる。グループ・メンバーの能力を無視した分担、課題の割り当てが多い。グループ・メンバーの能力の把握、発見につとめているグループは少ない。グループ・メンバーに課すべき課題がグループ・メンバーの自発的参加により徐々に定まってくるのが理想である。

(3) グループの状態が把握できない。

グループの状態は表向き把握されていることになっているにすぎないことが多い。ソフトウェア開発プロジェクトが開発期間の終り三分の一以降で問題が明らかになってくることが多いことからも分かることである。

(4) グループ・メンバーの意志や動機とグループ課題の整合性が不明になる。

グループ・メンバーの能力の把握が弱いと同時に、グループ・メンバーの意志や動機に対する理解も弱いことが多い。表向きはグループ課題に向けて各メンバーが作業していることになっているが、実は各メンバーは必ずしもグループ課題を念頭において作業をしている訳ではないことが多い。グループ・メンバーの意志がすべてグループ課題の達成に向くべきであるという訳ではない。むしろ、グループ・メンバーの意志がグループ課題に向いてはいないことが多いことの認識や、そういうメンバーの作業をグループ課題につなげる仕組みの必要性の認識の少なさを指摘したい。

(5) コミュニケーションがよくない。

コミュニケーションすべき事柄が認識されていないことが多い。グループ内コミュニケーションはかなり原始的なものの積み上げが大切である。少しの疑問、確認、提案、コメントなどが自由に流れる仕組みが必要である。組織レベルではコミュニケーション・チャネルや内容がフォーマルに決定されており、強制されることが多いがグループ・レベルでは意外と落とし穴になっている。

(6) リーダーが不在である。

通常、グループにはリーダーが存在する。リーダーがリードしていないグループが多い。

(7) グループ・プロセスが不明である。

グループ作業の分担のプロセス、コミュニケーション・プロセス、グループ内の決め事のプロセスなどは最初不明の状態から始まる。しかし、不明のまま進んでしまうグループが多い。グループ・プロセスが柔軟に決まり、メンバーに理解されていることが必要である。

(8) グループ内のコンフリクトの管理ができない。

グループ・メンバーは多様である。したがって、グループ内でコンフリクトが発生するのが自然である。コンフリクトを統合・昇華する形でまとめる力がグループに要求される。卓越したグループ・リーダーがその力を発揮する場合もあろうが、グループ・メンバー内にその力が分散している場合もあろう。

1.2.6 グループワーク支援システムに対する期待

グループ作業の成否は当然のことながら、グループ・メンバーである人にかかっている。しかし、グループワーク支援システムが支援できる部分には次のようなものがあると期待される。

(1) グループの課題(とその細分化したもの)とメンバー作業の関係の表示

(2) グループ内コミュニケーションの円滑化

(3) グループ内問題点リストの提示とトラッキング支援

(4) グループ・プロセスのガイドと記録

(5) グループ進捗のとりまとめと表示

1.2.7 まとめ

グループワーク支援にはグループの理解が必要である。グループワーク支援は通常のマネジメント支援の課題と似ているが、マネジメントよりインフォーマルな形の支援が必要になる。グループ・メンバーの自発性、創造性、人間性を前提にし活かすような支援システムが望まれる。

1.3 分散開発のプロセスモデル

大規模なソフトウェア開発は、複数の人間がグループを構成し、協調して開発を進めなければならない。このようなソフトウェア開発を効率的に行うためには、ソフトウェア開発のあらゆる局面で、グループ活動（グループワーク）を支援する必要がある。この支援技術としてグループウェアが注目されている。このグループウェアを活用するためには、対象となるソフトウェア開発活動自身を把握する必要がある。ソフトウェア開発活動は、ソフトウェア作成のプロセスと、それぞれのプロセスで作成される成果物として把握できる。

本節では、ソフトウェア開発活動のモデルとして、ソフトウェアプロセスを取り上げる。まず、ソフトウェアプロセスモデルの概念について概観し、分散開発環境でのプロセスモデルの必要性について考察する。さらに、グループウェアを構築するにあたって、ソフトウェア開発のプロセスとの関連を考察する。

1.3.1 プロセスモデルの概念

ソフトウェア開発では、品質のよいソフトウェアを、できるだけ少ないコストで、約束した納期までに開発し、さらには、納入後のメンテナンスまで行ってゆくことが求められている。

また、開発作業自体が知的活動であるため人間的要素が多いうえ、常に一品生産で、開発期間が長く、多くの人の共同作業で行われることから、経験の蓄積がなかなかうまくゆかないという特徴がある。

ソフトウェアの品質向上とコストの削減を図り、かつ納期を確保しゆくためには、開発作業の進捗を管理するだけでなく、工数やマシンタイムなどのリソースや工程ごとの中間製品の品質を管理する必要がある。さらに開発作業のやり方を改善し、生産性や品質をさらに向上させてゆくためには、ソフトウェアのライフサイクル全体を見渡した管理をきめ細かく行ってゆく必要がある。

このため、ソフトウェア開発では、開発工程を細分化し、各工程ごとにリソース、成果物を管理することによって、製品の品質、進捗状況を把握することが重要である。

ソフトウェアの開発のプロセスの典型的な例を示す（図1.3-1）。

(1) 基本検討工程（BI）

業務分析を行い、システムの要求仕様を固め、開発計画を作成する。

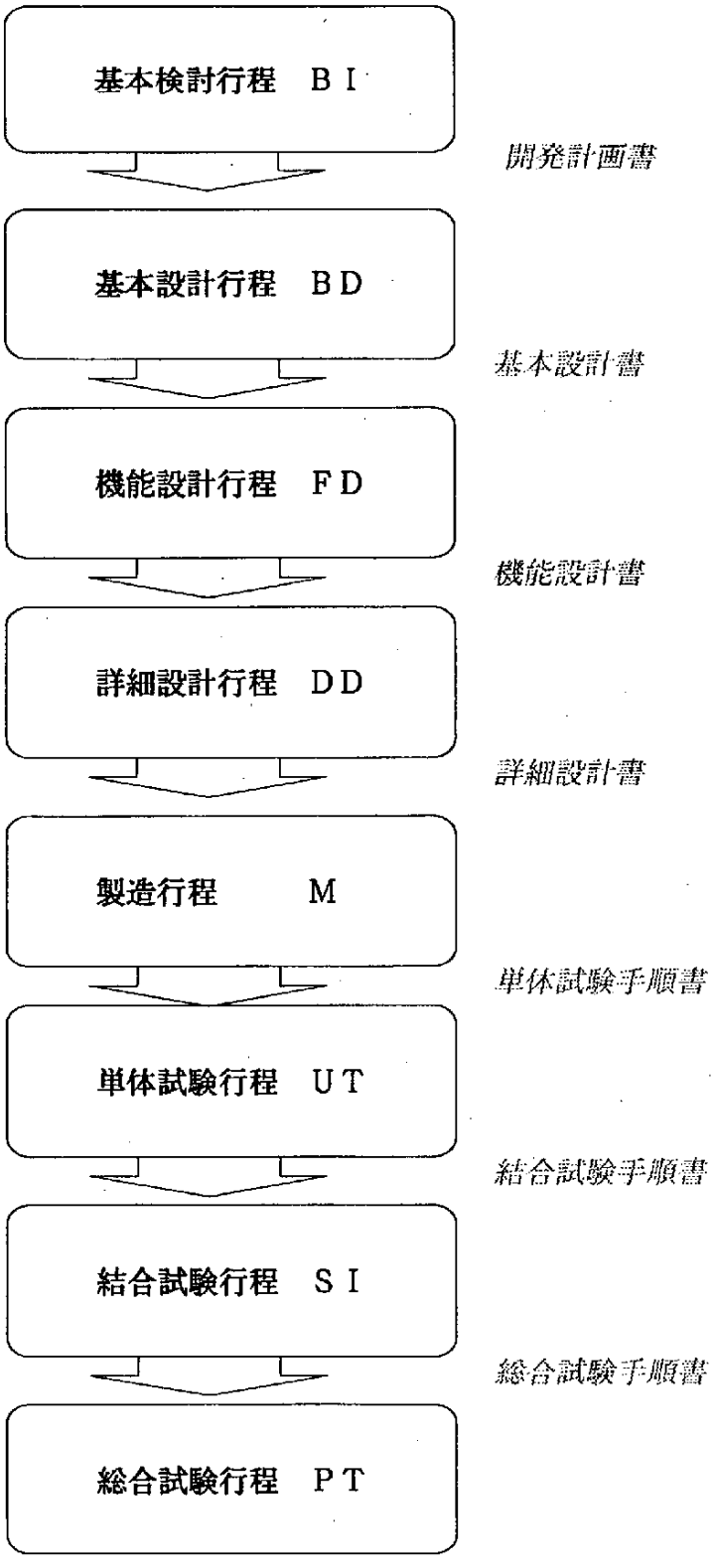


図1.3-1　　ソフトウェア開発プロセスの典型例

(2) 基本設計工程（BD）

システムの概略設計を行い、システム構成、ソフトウェア構成、開発工数など、基本条件を策定し、基本設計書を作成する。

(3) 機能設計工程（FD）

システムを構成する各機能単位の詳細化を行い、インタフェース条件、出力リストの形式、画面の構成などを設計し、機能設計書としてまとめる。

(4) 詳細設計工程（DD）

各機能ブロックを実現するプログラムの仕様を決定し、詳細設計書を作成する。

(5) 製造工程（M）

プログラムの論理設計、コーディング、コンパイル、机上チェックを行う。

(6) 単体試験工程（UT）

各ソフトウェアモジュールは所定の機能を満足するかどうか、マシンを使って確認する。

(7) 結合試験工程（SI）

各ソフトウェアモジュールを結合し、所定の機能を満足することを確認する。

(8) 総合試験工程（PT）

ハードウェア、ソフトウェアを含めた総合的な最終確認を行う。

1.3.2 分散開発とプロセスモデル

近年、ソフトウェアの開発量が著しく増大してきている。そもそも、ソフトウェア開発は人間の頭脳労働であるため、需要に見合う技術者を確保する必要がある。ソフトウェアの生産効率の面からは一箇所集中で開発するのが最適であるが、これ以上、首都圏で要員を確保するのにはもはや限界である。

そのため、地方で人材を集める開発要員の地方分散化が進められている。要員、開発設備を地域に分散させてソフトウェアを開発するためには、単に要員やWSなどの開発設備を地方に分散配置するだけではうまく行かない。

- 分散したチーム間での移動のロスとコミュニケーションのロスを最小限にするような環境を整えること
- 開発対象を分業しやすい単位に分割して配分し、地域のチーム単位に負荷バランスを均等化すること

などの課題を解決しなければならない。

本項では、特に、ソフトウェアプロセスの観点から、ソフトウェア分散開発における問題点を掘り下げてみる。

(1) 工程分離・専担化

設計の上流工程と、プログラム製造の下流工程を分離して、地域に分担して開発することがよく行われる。

これは、基本設計を首都圏で顧客の近くで集中して行い、ソフトウェアの仕様が固まったところで、地域のソフトウェア工場で生産する方式である[友野91]。

① 基本検討工程担当（BI）

開発要望元との窓口となり、具体的な要求仕様を検討、確定する。

② 基本設計担当（BD）

確定した要求仕様から基本設計を行う。

③ 設計（機能・詳細）、製造担当（FD−DD−M−UT）

プログラム設計、製造を担当する。

④ 試験工程担当（SI−PT）

要求仕様どおり、機能するか確認する。

このような各工程担当間で、プロダクトが順次引き渡されて行く。これにより、製品開発のスループット（単位期間あたりの製品供給数）を増大させることが可能となる（図1.3-2）。

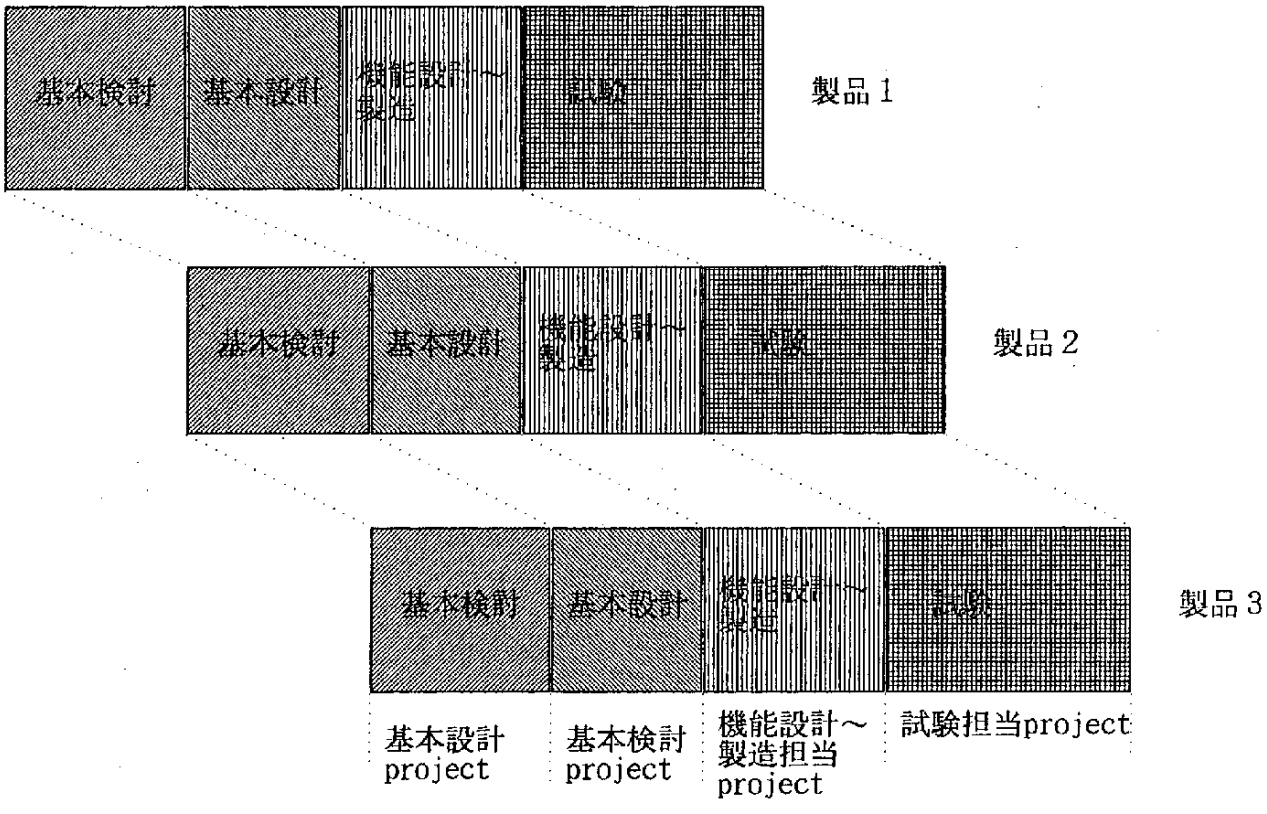


図1.3-2 工程専任による開発プロセス例

このような開発体制では、各工程の担当間で、設計書の引き渡しが頻繁に起こり、各チーム間でのコミュニケーションが大変重要になってくる。

また、ある工程だけ専任するため、たとえば、プログラム製造をもっぱら担当するチームでは、つねにプログラミング、単体デバッグに追われ、担当者のインセンティブが低下する恐れがある。

(2) サブシステム単位の平行開発

システムの規模が大きな場合には、システム全体をサブシステムに分割して、各地域の開発チームが設計から試験まで分担することにより、平行開発を行うこともよく行われる。

この場合には、サブシステムを担当するチームが地域に分散するため、

① 対象となるソフトウェア開発に必要な業務知識を持った専門家が分散してしまうため、分割損が生じる

② 各チームで独自に設計するため、同様なモジュールを重複して開発してしまう

③ サブシステム間のインタフェースなど分散したチーム間をまたがるような調整がうまく行かなくなる可能性がでてくる

などの問題が出てくる。

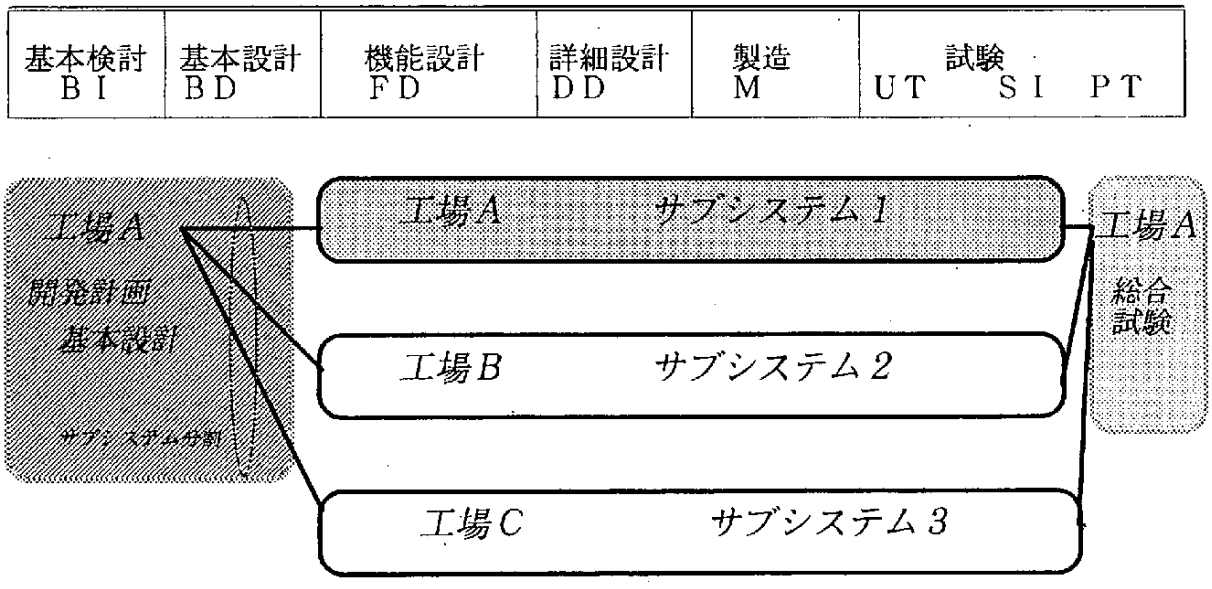


図1.3-3 サブシステム平行開発の例

(3) 専担化とシステム分割の組み合わせ

(1)、(2)を組み合わせることにより、各地域チーム（工場）の負荷を平順化する試みも行われている（図1.3-4）。

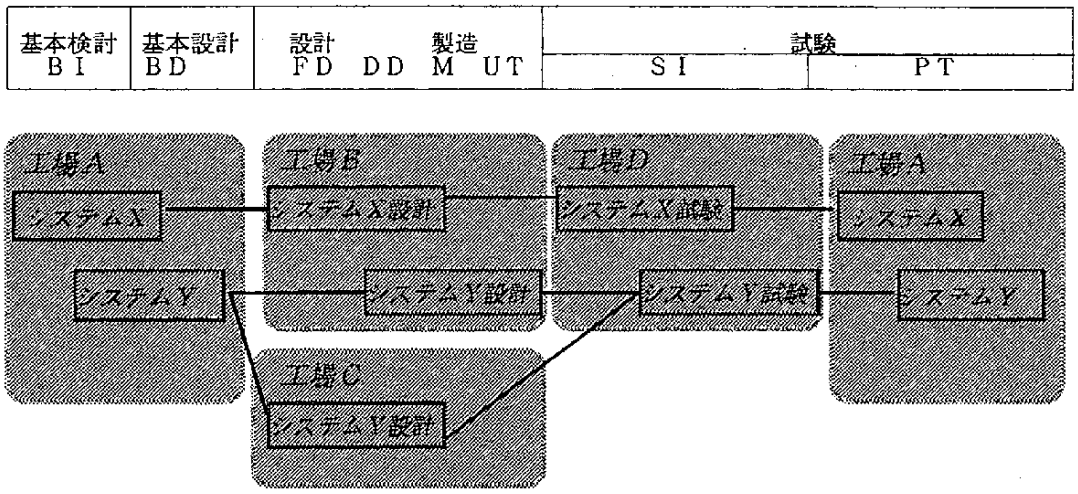


図1.3-4　サブシステム単位の分散及び工程分離の組み合わせ例

1.3.3　プロセスモデルとグループウェア

前項では、ソフトウェア分散開発における課題とその取り組みについて、ソフトウェアプロセスの観点から概観した。

本項では、ソフトウェアの分散開発におけるソフトウェアプロセスに関わるいろいろな試みで明らかになってきた問題点の中で、グループウェアと関わりがある、

① 地域分散したチーム間のコミュニケーションの支援

② プロジェクトの進捗の把握

③ 共通のデータベースの構築

について考える。すなわち、「地理的に分散した作業者間での協調作業と対話の支援」にグループウェアの技術をどう適用して解決できるか考察する。

(1) 地域分散したチーム間のコミュニケーションの支援

開発工程が地方に分散されると、工程間の引継ぎや問い合わせが分散したチーム間で行われるようになる。グループウェアの技術が役立つと考えられるのは、以下のケースである。

（支援業務）	（グループウェア技術）	（工程）
・ コメント・問い合わせ	半構造化メール	設計・製造・試験工程
・ レビュー	電子会議	全工程
・ プログラミング支援	共用マルチウィンドウ	製造工程
・ デバッグ支援（協調）	共用マルチウィンドウ	試験工程

(2) プロジェクトの進捗の把握

開発拠点が地方に分散すると、日頃から開発メンバーと顔を合わすことがなくなるため、進捗状況が十分把握できなくなる。

(支援業務)	(グループウェア技術)	(工程)
・ 工程進捗会議	電子会議	全工程
(グループ意志決定支援)		
・ 進捗報告	電子メール	全工程
・ スケジュール管理	電子メールなど	全工程

とくに、進捗を把握する際には、進捗状況を文書だけで報告するだけではなく、音声、顔の表情などのノンバーバルの情報も重要であろう。

プロジェクト管理については、チームメンバと実際に会って進捗状況を把握できないため、十分に行う必要があろう。「1.5 分散開発とプロジェクト管理」を参照。

(3) 共通作業データのデータベースの構築

地方に分散して作業をすすめる場合、システムとして必要な設計情報、各工程の成果物を、開発メンバがオンラインでアクセスできる共通作業データのデータベースとして用意する必要がある。これは、一箇所でソフトウェア開発する場合と同じ環境をつくることを意味しており、分散開発の”距離”の制約を解決するものである。「1.6 CASE統合環境におけるグループワーク支援」を参照。

(4) 今後の課題

ソフトウェア開発におけるグループウェアを考えるに当たっては、ドメインのモデル化といったトップダウンのアプローチと、グループウェア技術としてどこまでできるのかという、ボトムアップのアプローチの双方が必要である。ソフトウェア分散開発の概念モデルでは、次の内容をモデル化する。

- 開発工程
- 作業者（エージェント）
- 作業プロセス
- 作業者間の協調作業（インタラクション）
- 作業データ（プロジェクトデータベース）

概念モデルを構築したうえで、必要な機能の検討、シミュレーションによる事前評価など行うことが必要になると考えられる。

【参考文献】

[友野91] 友野, 竹内, 常世田 : 交換ソフトウェア開発における開発体制／技法, NTTR&D, 40, No.11 (1991).

1..4　分散開発のプロダクトモデル

1.4.1　分散開発環境の定義

この節で扱う分散開発環境の定義は、

『時間的にも空間的にも分散した環境（非同期・遠隔）において比較的独立性が高い組織によってコンカレントに行われるシステム開発作業』とする。

1.4.2　プロダクトモデルとは

(1)　プロダクトモデルの定義

システム開発におけるプロダクトとは『システム要求分析から設計・コーディング・テスト・運用（保守）に至るシステムライフサイクル全てにわたって発生する成果物』をいう。具体的にはシステム計画書・設計書・ソースコード・テスト手順書・ユーザマニュアル・開発スケジュール等が含まれる。広義には非公式な会話や個人的なスケジュール・思いつき・メモ等も含まれる。

プロダクトモデルとは全システムライフサイクルにわたってプロダクトに関する情報（プロダクトの構造、プロダクトそのもののライフサイクル、プロダクト間の関連）を表したものである。

(2)　プロダクトモデルの必要性

システム開発の各工程で発生するプロダクトは、見方を変えると、目標としているシステムの概念を各工程で必要とされる側面から見直した一つの論理ビューと考えられる。プロダクトの作成とは、前工程で作成されたプロダクトを次工程で必要とされるビューから読み替え、その工程で必要とされるプロダクトに変換する作業といえる。通常これらプロダクトはドキュメントとして作成されるが、その作成にあたっては人手によって前工程のドキュメントを読み替え、その工程で必要とされる情報を付け加え、再作成しているのが現状である。今後システムの大規模化に対応するためには、この読み替え・再作成のプロセスをいかにコンピュータで自動化できるかが開発成功へ向けてのキーポイントとなる。

このコンピュータによる自動化を行うためには、各ドキュメント（プロダクト）の変換をシステマティックに行えるように、ドキュメントの構造ならびに各ドキュメント間の関係を詳細に記述したプロダクトモデルが不可欠となる。

1.4.3　プロダクトの分類

(1)　共用プロダクトと個人用プロダクト

特定の管理機構（人であるかシステムであるかは問わない）のもとに、多数からの参照・操作を許すものが共用プロダクトであり、個人の責任の下で、個人の使用に限られたものが個人用プロダクトである。この定義に従った場合、共用プロダクトを個人環境のもとにコピーし、その個人の責任で操

作する場合には、共用プロダクトではなく個人用プロダクトに分類される。実際の開発現場においては、上記のような例は往々にして起こるものであり、使いやすさを損なわずに如何にうまく共用プロダクトを制御するかが、現状における課題の一つである。

(2) フォーマルデータとインフォーマルデータ

組織の階層構造に沿って上下に流れるデータがフォーマルデータであり、決定に至るプロセスで発生するデータよりも、その結果が重視される。これに対して組織を横断的に（水平方向）に流れるデータがインフォーマルデータであり、結果よりそのプロセスで発生するデータが中心となる。フォーマルデータは管理体制も整い、書式も定められている場合が大半であるが、インフォーマルデータはほとんどが書式も無く、また個人用プロダクトとして管理されている。今後は、このインフォーマルデータを、それが持つ自由度を損なわずに組織としてどう管理していくかが課題である。

1.4.4 プロダクトから見た分散開発の現状と問題点

(1) 現状

CASEツールの進歩に伴って、各工程におけるプロダクト作成の支援環境は整ってきているが、工程間を接続する技術に関してはまだ未熟である。最近はリポジトリによる全工程の一元的な管理を目指した技術も生まれつつあるが、製品開発が始まったばかりであり、実際の開発現場に根付く迄には今しばらく時間が必要である。

(2) ライフサイクルモデルに基づいた問題点

従来のウォータフォールモデルに基づくシリアルな開発モデルは、分散環境におけるコンカレントな作業環境と相入れない部分がある。分散開発に適した開発モデル（例えば各拠点毎のプロトタイピングの適用等）の作成ならびに分散開発に適したプロダクトの分割方法が必要である。

(3) コンピュータ技術に関する問題点

① 統一的なデータの記述能力

システム開発で扱うプロダクトは表現方法（文書、図、グラフ等）が多様であり、またその構造や操作に関しても複雑である。システム開発の各工程においては、その工程に最適な記述方法を使ってプロダクトを作成するものであり、各プロダクトを跨がる統一的なデータの記述方法を確立しない限り、プロダクト間のシステマティックな接続は困難である（一般の製造業に見られるようなJIS規格図面に相当する記述方法の確立）。

② 高速通信技術

一般的に、図やグラフ等を多用するプロダクトはデータ量が多い。分散化された環境にあっては、頻繁に参照されるような共用プロダクトに関して、通信に必要とされる時間がシステムの使い勝手の上で大きな問題となる。

③ アクセス制御

オンライントランザクションシステムのように、短時間でかつ少量のデータに対するアクセス技術は確立されているが、ドキュメント作成のように、大量のデータに対して長時間にわたる同期・排他制御を行う方法、ならびにその制御単位（ドキュメント全体・章・句・単語等）に関する制御技術が貧弱である。

④ ツールインターフェース

同一のプロダクトを複数の異なった環境からアクセスする場合、それぞれに異なるツールに対して統一的なインターフェースをどう与えるかが問題となる。現状、PCTE (Portable Common tool environment) にみられるように、ツールインターフェース標準化の動きもあるが、国際標準として確立されたわけではない。またリポジトリを使った解決策も考えられるが、分散環境においては使われるリポジトリそのものが異なる場合も考えられる。

⑤ セキュリティ

ネットワークを利用してプロダクトにアクセスを許す場合、集中環境下での場合と比べて、どうしてもセキュリティの面で弱くなってしまう。いかにシステムの使い易さを落とさずにセキュリティを守るかということが問題である。

⑥ バージョン管理

個人で使用するようなプロダクトは、更新された時間軸に沿ってバージョン管理を行えば問題はないが、複数の人間による更新の場合、それぞれの人間に関してバージョン管理ができるような機構が必要となる。単純なバージョン管理を行った場合、特定の人間が連続して更新を行うと、他の人間による更新が履歴として残らないという問題がでてくる。

(4) その他

① 文化の壁

経済のグローバル化に伴って開発拠点も国際化せざるを得ないが、そうなった場合、ドキュメントの表記言語や翻訳の問題等の言葉を中心とした文化の壁に突き当たる。現状では、上流工程から国際化して開発していくことは考え難いが、コスト削減を意識したコーディングレベルのみの国際分業化は充分に考えうる。

② ネーミングルール

同一概念に対して複数の名前付けを行うことは日常的に見られる現象であるが、集中環境にあっては、会話等の手段を通して大半の場合、名称の統一は計られている。しかしながら、分散環境ではお互いの意思を確認しあえる手段に乏しく、不統一な名称は後々まで続く大きな誤解の原因となり、ひいてはバグの温床となりかねない。

③ インフォーマルな情報の取扱

集中環境においては日常の会話等から何気なく得られたインフォーマルな情報も、分散環境においては何らかの手段で蓄積・交換する必要がある。また単に蓄積・交換するだけでは不要な情報による情報洪水を招く結果となりかねず、何らかの方法で情報の選別を行うことが必要である。

④ デザイン理由・変更理由の取扱

現状でもこれらの情報は必要欠くべからざるものとされているが、分散環境にあっては他の拠点との意志疎通が粗となりがちであり、理由のはっきりしない変更や設計は混乱を招くもとである。

⑤ プロジェクトマネージメント

これはプロダクトのみの問題ではないが、分散環境では各拠点毎に比較的独立性が高いという理由から、進捗状況の把握が困難であったり、また問題発見やタイムリな意志決定ができなかったりする。

1.4.5 分散開発のプロダクトモデリング

(1) モデリングのための指針

以下にプロダクトモデルを構築する際に必要と思われる技術、ならびにその選択理由を述べる。

① オブジェクト指向概念

種々の表記法、複雑な構造を持ったプロダクトを統一的に扱う能力に優れている。またクラス概念や継承機構を導入することで、複雑になりがちなプロダクトの構造を無理なく扱うことができる。

② 情報伝達のための半構造化メール

インフォーマルな情報に加え、フォーマルな情報も同一レベルで扱う能力を持つ。また意味（構造）を導入することによって情報の取捨選択が可能となる（情報洪水の防止）。

③ ハイパー構造

通常、組織とプロダクトはn対mの関係にある。またインフォーマルなデータは、その内容も関連するプロダクトも常時変化し続けるものであり、これをスタティックな構造で管理することは困難である。これらを無理なく扱うためには、ハイパー構造の持つリンク機構を使ったダイナミックな構造設定能力が必要である。

④ メタデータ概念

常に変化し続ける開発環境やプロダクトに対して、その変化の影響を最小限に止める能力に優れる。

(2) プロダクトモデリング例

上記の指針を基に、開発環境のモデリングを考えてみる。

① プロダクトとプロジェクトの分離

プロダクトは、プロジェクト全体に関わるものから、サブプロジェクトに関わるもの、さらに個人のみが関与するものまで、その関連が複雑多岐にわたる。これを一元的に扱うには、モデルの構造が複雑に成り過ぎるため、先ず開発環境全体をプロジェクトに関するものとプロダクトに関するものに分

離する（クラス概念の導入）。

② メタ化

分離したプロジェクトとプロダクトを図1.4-1に示すようにメタ構造化する。

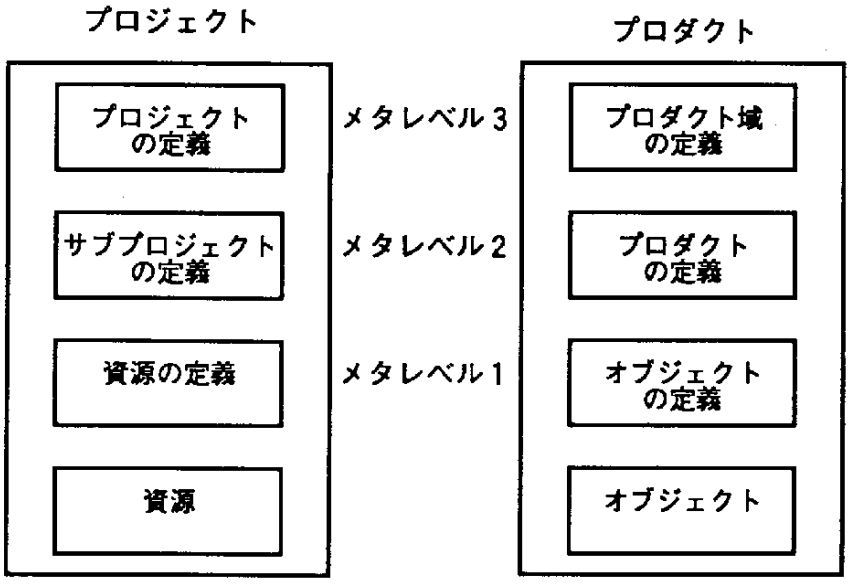


図1.4-1 メタ構造

各メタレベルに含まれるデータは以下に述べるようなものが考えられる。

(a) プロジェクトの定義(メタレベル３)

開発対象プロジェクトを定義する。定義される内容としては、

- プロジェクトを構成するサブプロジェクト(分散開発においては、各拠点がサブプロジェクトに対応する)
- サブプロジェクト間の関係（サブプロジェクトの流れ、任意時点での多重度)
- プロジェクトの目的
- 関連する部門
- プロジェクト全体のスケジュール
- プロジェクトの責任者 等

(b) サブプロジェクトの定義(メタレベル２)

各開発拠点単位のプロジェクト(サブプロジェクト)を定義する。定義される内容としては、

- 必要な資源(人、金等)

・ 関連する部門

・ サブプロジェクトの責任者

・ 資源間の関係(人員構成、組織構成等)

・ サブプロジェクトのスケジュール 等

(c) 資源の定義（メタレベル１）

当該環境において利用可能な資源を定義する。定義される内容としては、

・ 導入されているソフトウェアやハードウェアの情報

・ 個人的スケジュール

・ プロジェクトメンバの情報 等

(d) 資源

・ ソフトウェア、ハードウェア

・ 人、金 等

(e) プロダクト域の定義（メタレベル３）

全プロジェクトを通して作成される全てのプロダクトを定義する。定義される内容としては、

・ 全プロダクトの責任者

・ プロダクト間の関連

(f) プロダクトの定義（メタレベル２）

各プロダクトをを定義する。定義される内容としては、

・ プロダクトの名称

・ プロダクトを構成するオブジェクトの関係(ビューの定義)

・ プロダクトの責任者(所有者)

・ プロダクトに対するアクセス権 等

(g) オブジェクトの定義（メタレベル１）

プロダクトを構成する個々のオブジェクトを定義する。定義される内容としては、

・ オブジェクトを処理するためのプロセス（または標準インターフェースの提供）

・ オブジェクトの所有者

・ オブジェクトに対するアクセス権

・ オブジェクトのタイプ（テキスト、図形、イメージ等）

・ オブジェクトの分類（個人用／共用、フォーマル／インフォーマルの区分） 等

(h) オブジェクト

・ テキスト、図形、イメージ 等

テキストや図形はそれぞれ別々のオブジェクトとして管理される。こうすることで、それを処理す

るためのツールが単純化され、異なる環境での扱いが楽になる。

各レベルにおけるデータは、より上位にあるメタレベルのデータを引継ぎながら、各レベル固有のデータを付加していく形で全体のデータが構成される（継承機構）。これにより、不要なデータの重複定義を防ぐことができる。例えば、プロジェクトの定義（メタレベル２）で定義されているスケジュールは、サブプロジェクト（メタレベル１）に引継がれ、その全体スケジュールの枠組みの中で、サブプロジェクトのスケジュールが決定される（同様にメタレベル１の資源の定義内の個人スケジュールにも引き継がれる）。

③　プロジェクトとプロダクトの関係付け

最後にプロジェクトとプロダクトを関係付けるために、お互いの間にリンクを張る(ハイパー機構)。今回のモデルでは、図1.4-2に示すように、同一レベル間で各々必要とされる物同士にリンクを張り、プロジェクトとプロダクトを関連付ける。リンクの表現は、上記②で定義した各メタレベルにおける属性として新たに付け加える。

同一レベル間のみにリンクを張る構造にした理由は、異レベル間にリンクを張った時に起こる構造の複雑化を避け、下位レベルの管理を上位レベルにのみ行わせることにより、変更時の波及効果を最小限に抑えるためである。これによって、開発現場の作業者は自分の関係するオブジェクトにのみ集

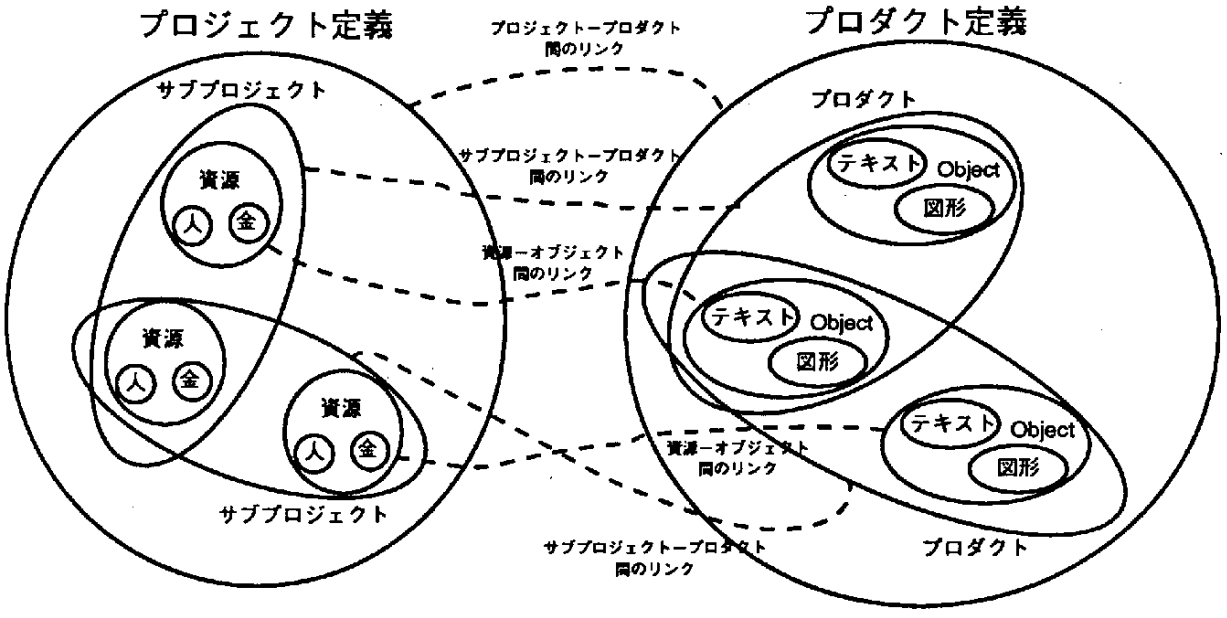


図1.4-2　プロジェクトとプロダクトの関連

中すれば良く、またプロジェクトマネージャーは個々のオブジェクトの内容は気にせずにプロジェクト全体の管理に気を付けていれば良く、管理の階層化も無理なく実現可能となる。

④ メッセージの流れ

最後に、上記モデルを使ったコミュニケーションを考えてみる。図1.4-3に見られるように、コミュニケーションのために必要なメッセージは自分のメタ定義を参照しながら、半構造化メッセージの形で、関連する上位／下位のメタならびにリンクを張られたデータに送られる。メッセージを受け取った側はメッセージの内容を解析して、必要なアクションを起動させると同時に、必要ならば自分のメタ定義を通して、関連する上位／下位のメタにメッセージを送ることができる。これによって何らかの変更がデータに加えられた場合、自動的に関連する全てのデータにメッセージを送る機構（ノティファイ機構）を実現することができる。

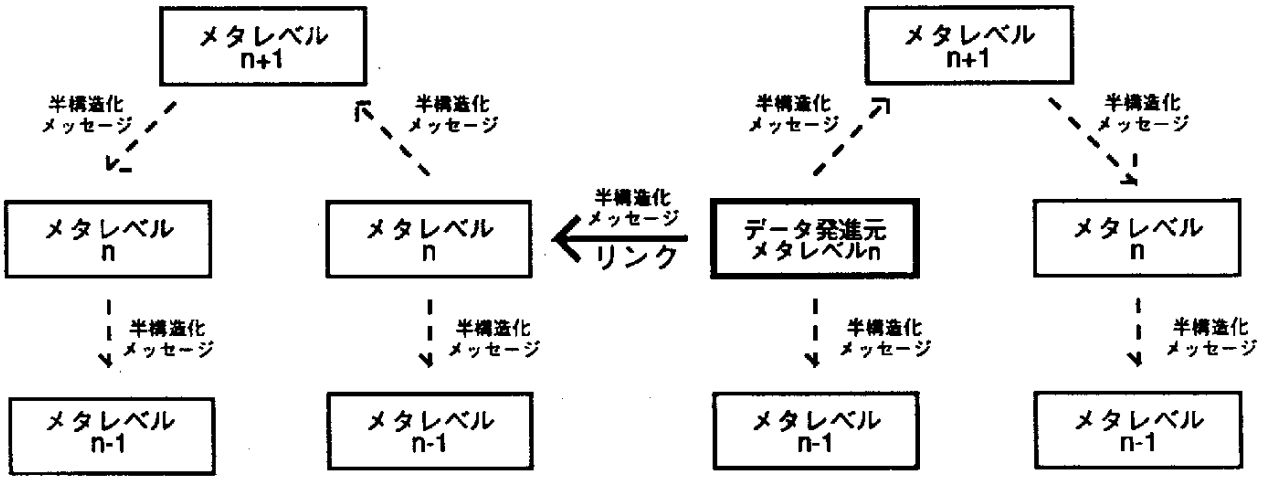


図1.4-3 モデル上のメッセージの流れ

(3) 今後の課題

以上がメタ構造、ハイパー機構、半構造化メッセージ、オブジェクト指向を使ったモデリングの例である。今回のモデルはプロダクトモデルの全体概念を示しただけであり、このモデルを制御するためのプロセス構造（プロセスモデル）や、オブジェクトをどういった単位で持つか（粒度の問題）、さらにどういった個々のプロダクトが実際のシステム開発で必要とされるか等には、一切触れていない。今後はこういった問題を検討していくことが必要となる。

1.4.6 Hypertextを使ったシステムの実例

(1) 概要

ここでは、Hypertextを使った実際のプロダクト管理システムの例として、GargらによるDIF (Documents Integration Facility) について説明する[Garg89, Garg90]。DIFは自然言語で記述されたドキュメントを大量に含む大規模なシステムのドキュメントの管理を目的としており、システムライフサイクルで発生する全てのドキュメントが対象となっている。DIFでは、ドキュメントを蓄積・処理・表示・改訂・再利用可能なオブジェクトと考え、それぞれのドキュメント間の関係はHypertextのリンクを用いて表される。物理的には、オブジェクトはUNIXのファイルとして、オブジェクト間のリンクはRDBで管理される。

(2) Hypertextによるドキュメント管理

システム開発を通して発生する全てのシステム関連ドキュメントは、Hypertextのノードとして登録される。各ノードには属性情報とリンク関係が付加され、これらを使って情報フィルタリングが可能となっている。またノードには何らの制限も加えていないため、任意のフォーマットのデータが登録可能である（これが従来のファイルシステムに対する大きな優位点である）。

(3) 管理対象ドキュメント

DIFで採用したドキュメンテーション方法は、南カリフォルニア大学で開発されたSystem Factory Documentation Methodを利用している。これはシステム開発プロセスをアクティビティに分解し、それぞれのアクティビティに対してドキュメント作成が行われる。DIFにおいては以下の8つのドキュメントが対象とされている。

① 要求仕様書(Requirements specification)

② 機能仕様書(Functional specification)

③ 構造仕様書(Architectural specification)

④ 詳細設計仕様書(Detailed-design specification)

⑤ ソースコード(Sourcel-code document)

⑥ テスト仕様および品質保証書(Testing and quality-assurance document)

⑦ ユーザマニュアル(User manual)

⑧ 保守マニュアル(System-maintenance guide)

(4) DIFの構造

① 操作モード

DIFにはスーパーユーザモードと一般ユーザモード二つのモードが存在する。スーパユーザはプロジェクトの責任者の決定や作成すべきドキュメントならびにそれに記載する内容を決定する。一般ユーザは、スーパユーザの指定したドキュメントの作成・変更を行う。さらに一般ユーザは、情報レベル

と情報構造レベルとよばれる二段階のレベルでDIFの操作が可能である。

② フォームとテンプレート

ドキュメントを規定するフォームならびにテンプレートの形式は、スーパユーザのみに設定する権限が与えられており、一般ユーザは変更することはできない。全プロジェクトに対し共通のフォームとテンプレートを設定することで、ドキュメントの標準化が図れ、かつ一般ユーザはフォームを気にせずにドキュメントの中身にだけ集中すれば良いというメリットがうまれる。

③ アクセス管理

スーパユーザはDIF上にプロジェクトのリストとそれに関連するエンジニアの情報を用意する。DIFはこの情報を基に、ドキュメントに対するユーザのアクセスを制御することができる。

アクセスを許された一般ユーザは、他プロジェクトまたは他人の作成したテンプレートにキーワードを付けたり、またリンクを張ったりすることができる。

④ 情報レベル

システム開発に関係する基本的な操作（例えばドキュメントの作成、プログラムのコンパイル等）はDIFを通して実行可能であり、DIFプラットフォームだけでシステムの開発を行うことができる。

⑤ 情報構造レベル

情報構造はデータベースでいうところのスキーマに相当し、ユーザは情報構造レベルで定義されるキーワードとリンクを使って、任意のドキュメントにアクセスすることができる。ただしデータベーススキーマと異なる点は、データベーススキーマでは構造が固定されており、情報構造レベルでは構造が実行時に動的に決定される点にある。以下に情報構造レベルを実現するための機構について説明する。

(a) リンク

リンクはテンプレート間の接続のために用いられる。

（例）C言語ソースコード ～ 実行モジュール

実際のリンクはテンプレートと他のテンプレート間、もしくはテンプレートの内容の一部と他のテンプレートの内容の一部の間に張ることができる。

(b) キーワード

キーワードは個別のユーザ毎に設定可能であり、必要とするノードを捜すために用いられる。キーワード自身はリレーショナルデータベースに蓄積される。

(c) フォームとコンポジション

フォームは従来の書式に相当するものであり、スーパユーザのみが変更可能なものである。これに対し、コンポジションはユーザ独自のフォームであり、Hypertextの機能をフルに使って任意の書式(ユーザビュー)を設定するための機能である。

⑥　ドキュメントの一貫性制御

複数のユーザから同時更新を許す機構をDIF は備えている。ただし実現方法については明記されていない。

⑦　ファイル構造

Hypertextの各ノードはUNIXのファイルとして実現され、リンクやキーワード等の情報構造レベルのデータはリレーショナルデータベースに蓄積される。

(5)　まとめ

以上みてきたように、DIF には以下に示すような分散開発環境におけるプロダクトモデルに必要とされる機能をある程度実現しており、非常に参考になると思われる。

- Hypertextを利用したドキュメント構造の動的な表現。
- 要求仕様作成から保守に到るまでの、全システムライフサイクルの一貫したサポート。
- メタ階層(情報構造レベル) の導入。
- ドキュメントの同時更新を許す機構の導入。

【参考文献】

[Bigelow88] Bigelow,J.:Hypertext and CASE, IEEE Software, pp23-27 (1988).

[Garg89] Garg,P.K.,Scacchi,W.: ISHYS Designing an Intelligent Software Hypertext System,IEEE EXPERT, pp52-63 (1989).

[Garg90] Garg,P.J.,Scacchi,W.: A Hypertext System to Manage Software Life-Cycle Documents, IEEE Software, pp90-98 (1990).

[原田91] 原田 実 : CASEの全て, オーム社 (1991).

[Malone87] Malone,T.W.,et al.: Semistructured Messages Are Surprisingly Useful for Computer-Supported Coordination, ACM Transactions on Office Information Systems, Vol.5,No.2, pp.115-131 (1987).

[松下91] 松下 温 : 図解グループウェア入門, オーム社 (1991).

[落水92] 落水浩一郎 : ソフトウェア開発におけるグループウェアの役割, ソフトウェア・ツール・シンポジウム'92 予稿集, pp.83-92 (1992).

[下平91] 下平丕作士 : 建築生産システム統合化のためのプロダクトモデルの研究と利用の展望, 情報処理, Vol32,No.7, pp839-848 (1991).

[Thomas89] Thomas,I.: PCTE Interface:Supporting Tools in Software-Engineering Environments, IEEE Software, pp15-23 (1989).

1.5 分散開発とプロジェクト管理

1.5.1 分散開発におけるプロジェクト管理の特長

最近のシステム開発プロジェクトでは、開発作業の一部もしくは大部分を外部の協力会社などへ発注するケースが多く、また大規模システムでは一ヶ所で集中開発するオフィススペースがなく、場所を分散して開発せざるを得ないケースも出ている。さらに、首都圏だけで開発要員を確保するのが困難になっており各地方拠点で分担開発する地域分散開発が既に現実化している。

プロジェクト管理とは「人」「時間」「物(開発環境、ツール等)」「金」のプロジェクト資源を有効に活用し、開発システムの要求機能と品質を確保し、納期と予算を守る作業であり、これまで多くの技法が開発されて来た[JIPDEC91]。

しかしながら分散開発におけるプロジェクト管理では、プロジェクトリーダが日常開発状況の実態を見ることができないため次のような点に留意して開発する必要がある。

(1) プロジェクト管理標準手順

各開発拠点の管理者のレベルもまちまちであり、作業レベルと品質に差異が起こりやすいのでプロジェクト計画の段階からプロジェクト制御、評価までの管理作業の流れを標準化する必要がある。これにより問題点の早期検出と対策を図る。

(2) プロジェクト管理項目

最終成果物だけでなく、管理手順に従った管理項目を設定する。管理項目として工程管理や品質管理のほかドキュメント管理、案件管理、仕様変更管理等がある。

(3) プロジェクト体制

プロジェクト管理グループにより、各管理指標の基準値、管理資料のフォーマット、プロジェクト運営方法、役割分担等を決め、プロジェクトメンバに徹底させプロジェクトの管理をする。

(4) プロジェクト管理の機械化

プロジェクトリーダが工程と品質の実態を把握する仕掛けと分散拠点との各種プロジェクト情報の配布、収集する仕掛けが必要になる。特に各種情報を迅速に伝達するネットワークシステムは不可欠である。

分散開発におけるプロジェクト管理は、プロジェクト組織(企業内開発、外部依託開発)、分割方式(開発工程による分割、機能による分割)、発注形態(分散開発を担当する会社との契約方式)によって異なるが、従来の成果物主体の管理に加え計画段階で設定した各種管理資料での管理が重要になる。

プロジェクトリーダは標準化した管理項目と管理基準によるプロジェクト情報をタイムリに把握し、また分散開発拠点側ではシステム開発に必要な管理情報が容易に入手できる事が分散開発におけるプロジェクト管理のキーポイントである。

1.5.2 プロジェクト管理標準手順

プロジェクト管理手順の目的は工程遅れと異常の早期発見と除去、当初の計画目標達成に向けての調整である。

プロジェクト管理手順は図1.5-1に示すように計画段階と実施段階があり、それぞれ次に示す作業を行う。

(1) プロジェクト計画段階

- 開発作業の洗い出し……基本構想書に基づいて最終成果物を決定し、それに対応した開発作業を決める。成果物と開発作業を階層構造に詳細化して体系化した技法としてWBS(Work Breakdown Structure)がある。
- 開発手順の標準化……詳細化した作業単位をベースにして開発手順を決める。分散開発では並行開発作業も考慮する場合もある。
- プロジェクト組織の計画…工程別、機能別のプロジェクト体制を設定する。プロジェクト計画に必要な職能(リーダ、分析者、設計者、プログラマ等)別に開発作業を対応させプロジェクト組織を計画する。
- 開発環境の設定……開発方法論、開発技法に加え分散開発環境を設定する。地域分散開発を実現するワークステーション型CASEを搭載したワークベンチの活用等を検討する。
- プロジェクト資源計画……開発工数、開発環境等の資源見積と配分計画を立てる。
- プロジェクトスケジュールの設定……大日程計画書を作成する。中日程計画書から下の詳細スケジュールはそれぞれの工程レベルのイニシェーション作業で設定していく。
- プロジェクト管理基準の設定……管理項目を決め、管理情報の収集タイミングを決める。また管理基準を設定しプロジェクトの状況判断に用いる。管理基準として進捗や品質の基準の他、障害管理基準、レビュー基準、連絡票伝達等の運用基準がある。また分散開発の立場から開発者に対する委託基準を決める。最終納入物件を規定するだけでなく、開発作業状況報告の内容、チェック方法も規定する。

(2) プロジェクト実施段階

- 管理情報の収集・分析……プロジェクト管理手順に従い、開発作業のマイルストン毎の管理情報を収集し、分析する。
- 原因究明……分析された情報をもとに問題点の原因を究明し対策方針を立てる。
- 対　策……対策作業方法を決め、作業の優先度、プロジェクト資源からの制約等を考慮し作業指示をする。

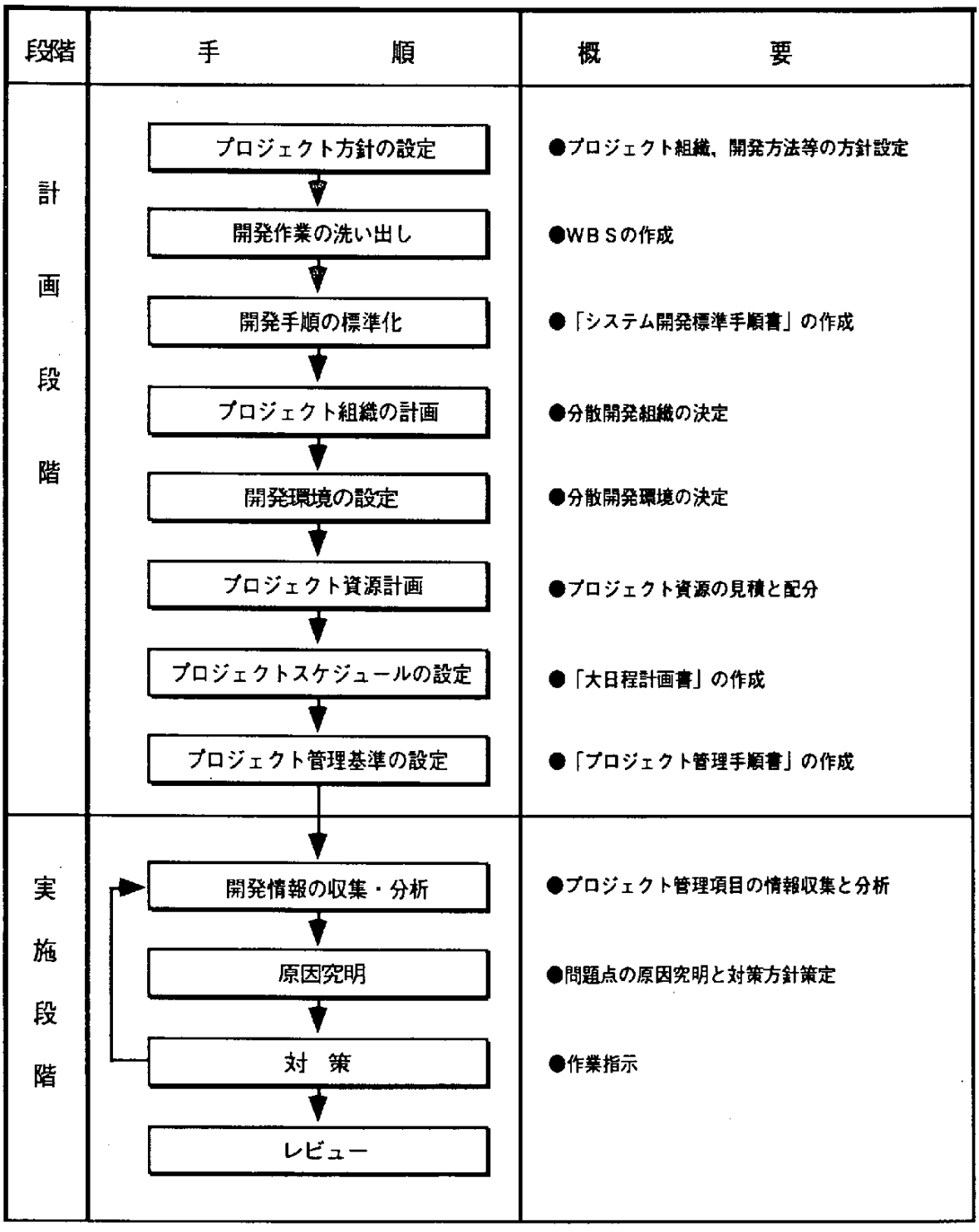


図1.5-1　管理手順の概要

・ レビュー……レビュー計画により着実なレビューを実施し、適切かつタイムリな評価と制御を図る。

1.5.3 プロジェクト管理の内容

プロジェクト実施段階におけるプロジェクト管理項目として次の項目がある。

(1) 進捗管理

開発計画に基づき作業の進行状況を把握し、遅れが発見されたら、体制、開発計画の見直し等の適切な対策を行うことがねらいである。進行状況の把握のためには、開発作業項目に対し、進行状況を客観的に数字で定量的に計測し定められた評価方法により判断する。

(2) 品質管理

ソフトウェア(プログラムおよびドキュメント)の品質を一定水準以上とするため品質上問題のあるもの、問題のある可能性が高いものを検出し、必要な対策を行うことがねらいである。

品質の評価のためには、品質の評価方法および合格基準をあらかじめ設定しておく。

(3) 開発規模管理

プロジェクト計画時の開発規模見積り値を実績値および各工程毎での見直し値と比較し、大きく相違するものについてはその要因を明らかにし、適切な対策をとる必要がある。

大きな仕様変更があったときには開発規模、要員、納期等の見直しを実施する必要がある。

(4) 発注管理(一括発注管理)

外注先に、プログラム開発を一括発注した場合、期待したとおりの機能、品質の成果物がスケジュールどおりにでき上がってくる様に管理することである。

特に、中間工程での進捗管理、品質管理が重要であり、予め外注先との間で進捗会議、レビューでの提出物件等詳細に定めておく。

(5) 案件管理

システム開発上の問題点や検討事項等の案件を一括管理し、対策のプライオリティ付け、スケジューリング、対策状況の把握を行う。

各案件には担当者を定め、スケジュール表に載せ管理する。

(6) 仕様変更管理

システム開発の過程では、一度決定された設計仕様が、種々の要因により変更されることが度々発生する。しかし、これらの仕様変更要求を余りコントロールせずに受け入れると、設計、プログラム作成、テストのやり直しとなり、工程遅延、品質低下、工数増大といった結果をまねくことになる。このため、工程の状況と、仕様変更の必要度を考慮し、仕様変更のスケジューリングを行う。また、仕様変更により開発規模が大幅に増加する場合、コスト負担、スケジュール等を再検討して進める必

要がある。

(7) 連絡票管理

エンドユーザあるいは外注会社との間での質問や依頼事項、仕様変更要求等のやりとりを口頭でやっていると、忘れたり、解釈の違い等の誤解によりトラブルの元となる。必ず文章にして連絡を行い、回答の必要なものについては、回答期限のフォローをすることが必要である。このため、連絡票の窓口、連絡ルートを明確にしておく。

(8) 障害管理

テスト中に障害が発生した場合、直ぐに原因が判明しない場合や判明しても対策が困難であったり、影響範囲が広いため、すぐ対策できないケースがある。管理者が確実に障害対策状況が見えるように、障害の発生日、原因判明日、対策予定日、対策日等を管理しておく。

(9) 開発環境管理

システム開発に必要な開発環境(ハードウェア、CASE ツール)、ライブラリ、テストファイル、テスト DB などのディスク容量、作業用の端末数、作業場所等を予め見積り、手当てすると共に、各開発グループへの割り当てを行い、使用状況をフォローすることで開発に支障がないようにする必要がある。常に先の工程で必要となるリソースを予め手配しておく。

(10) 性能管理

予め性能目標を立て、各工程で見積り・実測を行い目標値内に納めることを目的とする。

(11) バージョン管理

大規模な開発では、第１フェーズ、第２フェーズ等、段階的に開発し、リリースする開発形態がとられる。また、大規模な仕様変更や機能追加がある場合、これを修正前後のものと分けて管理する必要がある。

この様にプロジェクトを混乱させずに大規模な修正を行うため、各種リソースを分けて管理するのがバージョン管理の目的である。ライブラリ、ドキュメント等をバージョン毎に、異なる部分、同じ部分を明確にし、修正もれが発生しないようにプログラム修正時の反映方法を定めておく。

(12) 要員管理

プロジェクトのメンバを集める場合、いかなる技能を持つ人がいつ、何人必要かを見極め手配する必要がある。そのためには、スケジュールを立て、各作業項目に必要な要員の質と数を見積り、全体の作業工数を見積もるとともに要員の時期別投入計画を立てて置かなければならない。そして、スケジュールの進捗に合わせ実績をフォローし適宜人員の投入計画を修正していくことが必要である。

要員の計画を行う場合、予め仕事を頼む分散開発側の会社と折衝しておく必要がある。また、現実には必要とする要員を要求どおり手配できないケースがあり、プロジェクトのスケジュールに合せ、要員スキルアップの教育を実施する場合がある。

(13)　ドキュメント管理

システム開発で作成されるドキュメントは、後工程で仕様変更等により度々修正される。この修正変更を確実に設計ドキュメントに反映し、必要な部署／グループに配布する必要がある。このためには、各ドキュメントの責任者を定め、修正変更の方法、履歴管理方式、配布先管理方式を定めておく。

1.5.4　分散開発プロジェクト体制

分散開発プロジェクトを発足させる時、次の点に留意して体制を作る必要がある。

- 開発グループ及びメンバの責任分担を明確にすること。
- 大規模プロジェクトの場合、管理グループや共通技術グループ等を独立した組織として 設定すること。
- 工程ごとに体制が見直せる柔軟性を持たせること。
- プロジェクトリーダに全体が見えるようにすること。

これらの留意点のほかに、開発するシステム技術的特徴と課題、開発規模、開発方針及び分担開発グループの開発能力等によりプロジェクト体制を考える。図1.5-2にプロジェクト体制例を示す。

プロジェクトマネジャーの下に、プロジェクトリーダ、各グループのサブリーダがいてそれぞれの職能に対応したグループの作業を管理する。

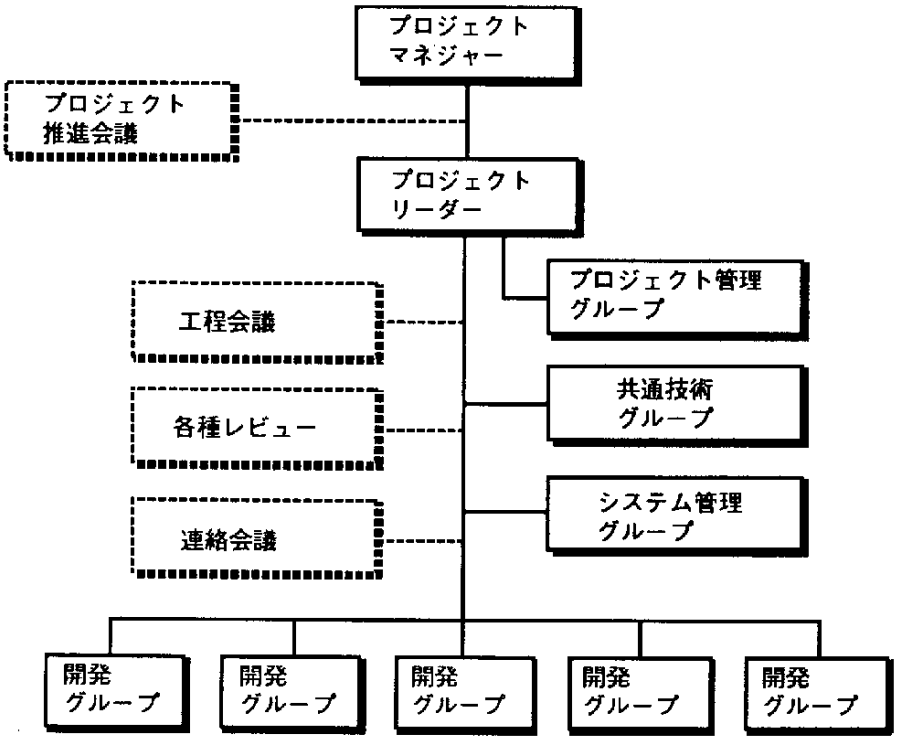


図1.5-2　開発体制の例

各グループの作業を次に示す。

- プロジェクト管理グループ…プロジェクト管理基準の設定と管理情報の収集、分析。
- 共通技術グループ……開発手順の標準化、設計仕様の整合性等の統轄、共通仕様の標準化(共通コード、プログラム部品、設計仕様)、システム全体の信頼性と性能の評価。
- システム管理グループ……開発とテストの環境整備、ライブラリ管理、組み合せテスト作業の支援。
- 開発グループ……各サブシステムの設計、製造、テスト。

1.5.5 プロジェクト管理の機械化

システム開発の分散化に対して管理は集中化しなければならない。ネットワークシステムを利用したプロジェクト管理の機械化を図る必要がある。図1.5-3は分散開発ネットワークシステムの例であるが、分散開発拠点のファイルに蓄積されたプロジェクト管理情報はネットワークシステムを伝送され中央側開発センタにあるリポジトリに収集される。分散拠点側では各サブシステムの開発の他、電子メールによる連絡票や障害票の送付、電子会議による設計レビュー等を行い、必要に応じて開発センタ、他分散開発拠点の設計情報を検索する。

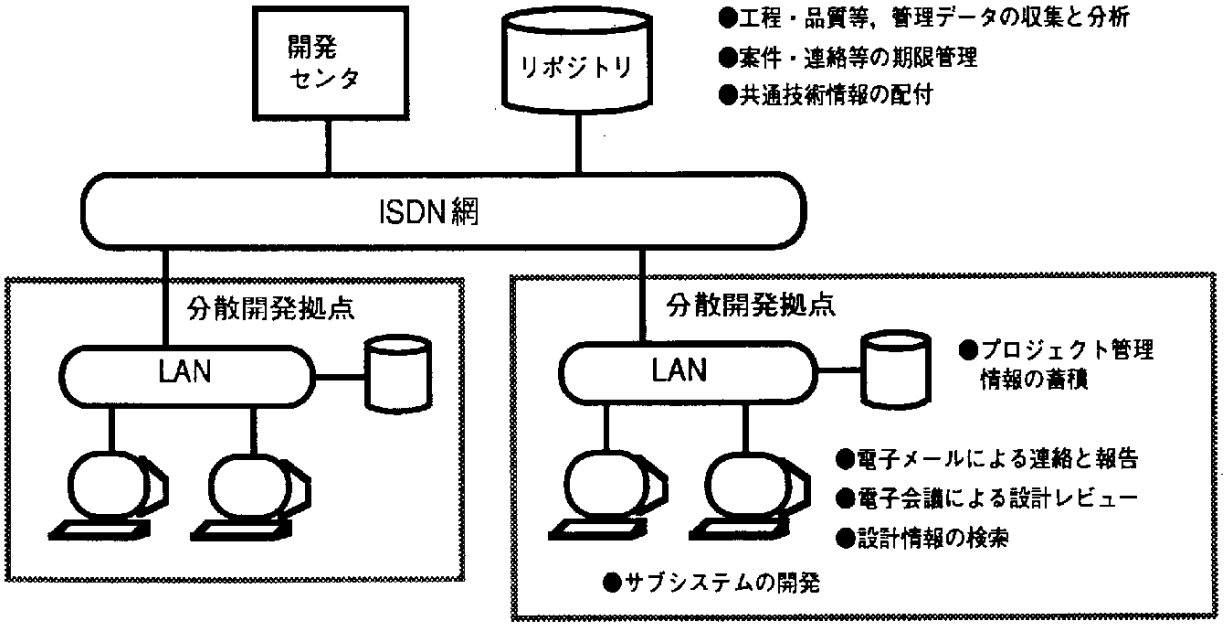


図1.5-3 分散開発ネットワークシステム

プロジェクト管理支援ツールの例として日立の SEWB／PJMS (Software Engineering Workbench ／ Project Management Support System)を紹介する。SEWB／PJMS はワークステーションで稼働するプロジェクト計画と進捗・品質管理を支援するツールで管理作業の標準化とレベルアップ、管理工数の削減を図っている。

SEWB／PJMS は工程管理表を階層管理しており、個人から開発グループ、プロジェクト全体までレベルに応じた進捗管理が行える。下位のレベルの管理情報は上位のレベルの管理情報に自動集計する。分散開発拠点での管理情報は自動集計され図1.5-4に示すような大日程計画表に状況が反映される。プロジェクトリーダは大日程計画表の進捗カーブから工程遅延の状況を把握し対策を立てる。

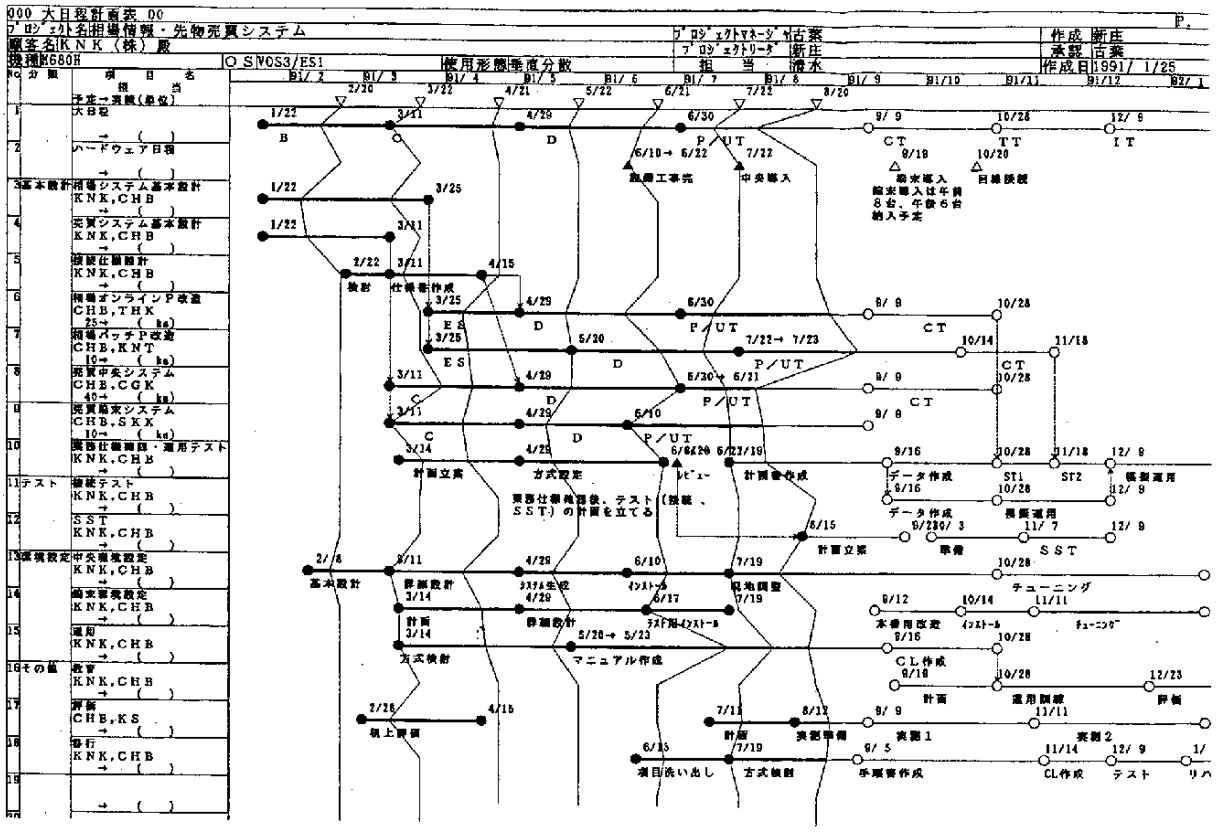


図1.5-4　SEWB／PJMS 大日程計画表の例

【参考文献】

[JIPDEC91]ＣＡＳＥに関する調査研究報告書,日本情報処理開発協会,pp. 97-111 (1991).

1.6 CASE 統合環境におけるグループワーク支援

1.6.1 CASE 統合環境とグループワークの接点

(1) CASE 統合環境の概略・現状

CASE ツールが市場に出始めたのは1984年ごろだと言われる。現在アメリカでは、100社を悠に超える企業が CASE ツールベンダとして名乗りをあげ、市場に多数のツールを提供している。だが、CASE ツールとはどのようなソフトをいうのか、論ずる人によりかなりの差異があり統一的な定義が存在しないのが実情である。ここでも CASE ツールの定義は特にせず、むしろ CASE ツールと呼ばれている一群のソフトウェアに関してその問題点を検討し、それらを克服するために打ち出された一つの方向、すなわち CASE 統合環境についてその現状を概観するとともに、「グループワーク」との関連を眺めてみる。

CASE ツールが市場に出始めても、市場でのソフトウェア開発の "Silver Bullet" としての歓迎ぶりとは裏腹に、CASE ツールの限界が認識されていた。一つの CASE ツールは一つの機能・サービスをサポートするにすぎず、ソフトウェア開発の上流工程から下流工程までの全ての作業をサポートするわけではない。したがってアプリケーションソフトウェアを開発しようとすると、開発の各工程にふさわしいツールを組み合わせて使う必要がある。しかし、組み合わせてもそれですぐにスムースなソフトウェア開発が実現できるわけではなく、開発方法論にしたがって、適切なタイミングで適切なツールを起動し、所定の成果物を生成していく開発プロセス制御機構が必要であるし、またツールの動き、ユーザの動き、プラットフォームの動き、成果物の動き等を、統合的、有機的に監視・制御・管理する機構も求められる。このように統合的・統括的な動きを行うのが CASE 統合環境である。ソフトウェア開発はある意味で総合技術であるので、個々の CASE ツールに統合的な動きができないのは無理からぬことではあるが、むしろ統合環境を構築するために必要なインタフェースが十分に整備されていないのが問題である。

CASE 統合環境がもつべき主要な機構を粒度を無視して列挙してみると、

- 開発対象アプリケーションにふさわしい開発方法論（プロセス）を設定し、それに従って開発作業を制御する機構
- 開発対象アプリケーションにふさわしいデータモデルを構築する機構
- 開発工程で関係する全てのデータを一元管理する機構
- プロセス記述に従って各工程にふさわしいツールを起動する機構
- 各工程にふさわしいプロダクトをツールに渡す機構
- ツール間コミュニケーションを保証する機構
- 環境を使用しているユーザの操作をモニタする機構
- ツールが排出する成果物をヴァージョン管理としてまとめる機構

・　成果物の動きを構成管理としてまとめる機構

これらいくつもの仕掛けは各々が独立に機能するものではなく、CASE統合環境として互いに有機的な連関を保ちながら動作することが求められるが、単独のCASEツールも次の3種類のインタフェースを備えることにより統合環境に組み込むことができる[松本90]。

① ユーザインタフェースの統合（日本語処理も含む）

② ツール制御の統合

③ ツールデータの統合

ユーザインタフェースの統合とは、人間系が関与する処理の方式を統一することで、具体的には画面形式の統一、画面操作形式（ウィンドウのオープン・クローズ、拡大・縮小、スクロール等）、メニュー方式の統一、日本語処理方式の統一等があげられる。ツール制御の統合とは、開発環境—ツール間あるいはツール—ツール間でコミュニケーションが可能なことをいうが、単にツールの起動・終了をツールの外部から制御できるだけではなく、種々のパラメータを介して、開発環境あるいはツールが必要とする情報・データを他のツールから取り出すあるいは注入することが可能なことをいう。このためには、各ツールにそのようなパラメータ付きの外部コマンド（あるいはメッセージ）を受け渡し、解釈し、適切なデータを出し入れする機構が備わっていなければならない。ツールデータの統合とは、ツール間でデータが自由に交換でき、しかもデータの解釈が全く等価でなければならない。すなわちシンタックスの同一性ばかりでなく、セマンティックスの等価性も保証されなければならない。このためには、同一のリポジトリを介して複数のツールがコミュニケーションを行うかあるいは個々のツール間でデータの翻訳を行うなどの方式が考えられる。

このような環境が実現すると、種々のレベルでの自由度が増し、それによって開発対象ソフトウェアの生産性および品質が向上することが期待できる。個人のレベルでは、ツールの選択が自由になる。セマンティクスまでの等価性が保証されているので、どのような機能のツールを環境に統合しようとも作業者が望むビューでデータを操作することができる。開発チームのレベルでは、開発のプロセスの自由度がある。その開発プロジェクトに最もふさわしいプロセスを記述し、それにしたがって作業を進めることができる。これがCASE統合環境のあらましである。

現在、CASE統合環境と言われるものが、いくつか発表されており[外山91]、

・ IBM AD／Cycle (Application Development／Cycle)

・ CIS ATIS (A Tool Integration Standard)

・ ISO IRDS (Information Resource Dictionary System)

・ ECMA PCTE (Portable Common Tool Environment)

などが挙げられる。

(2) CASE統合環境におけるグループワークの形態

ソフトウェア開発はすぐれてグループワーク的なものである。個人が趣味のために開発するソフトウェアは別にして、何らかの社会的活動に使用されることを前提としたソフトウェアは、グループワークによって開発される。仕様固めの段階で複数の人間の作業すなわちグループワークが行われることもあろうし、仕様・設計・製作・テストの全ての段階でグループワークが実施されることもある。すなわち、ソフトウェア開発それ自体が、本質的にグループワークである。現在ソフトウェア工学分野で行われている研究開発は、暗黙のうちにソフトウェア開発をグループワークであると捉えた上での研究開発に他ならない。開発方法論、ソフトウェアプロセス、品質管理技術、構成管理技術、開発支援環境等々、これらは全てソフトウェア開発はグループワークであるということから派生してくる考え方であるといえる。だが、開発作業者の手として機能する開発支援ツール、特に市販のCASEツールは作業者単独の作業を指向しており、ソフトウェア開発のグループワークとしての側面がほとんど無視されてきたといえる。むしろ、グループワークサポートは統合環境によってはじめて実現できたといえる。CASE統合環境をグループワークの面からみていくことにする。

まず、「ソフトウェア開発におけるグループワーク」とはどのようなアクティビティを指すのか、それを明確にしておく必要がある。が、これまでのところこれに対して明確な定義が与えられているとは言い難く、漠然としたイメージがあるにすぎない。そこで、作業仮説としてソフトウェア開発において起こる様々なアクティビティのうち、グループで作業するが故に必要となるアクティビティを列挙し、それらを整理してここでの「ソフトウェア開発におけるグループワーク」と定義することにする。それは表1.6-1のようになる。

これまでともすると、グループワークのサポートは「グループウェア」と呼ばれる一群の特定ツールによってサポートされてきたし、それで十分であるという認識が強かったと考えられる。また、CASEツール自体、もともとシングルユーザを指向して開発されてきたものであり、したがって、CASEツールにグループワークサポート機能が陽に取り込まれていることはほとんど期待できない。すなわち、グループウェアとCASEツールとは全く独立に、それぞれの考え方に基づいて発展してきたといえる。ところが、上記のようにソフトウェア開発そのものがグループワークであるので、CASEツールを使用している、いないにかかわらず、ソフトウェア開発のあらゆる工程、あらゆる場面において、グループワークサポート機能が提供されている必要がある。これはすでにそのサポートが個々のCASEツールの範囲を超えており、CASE統合環境がその構成機能要素の一つとしてもっていなければならないものであると考えられる。

表1.6-1に見るように、グループワークをCASE統合環境がサポートするためには、現在のリポジトリの持つ機能に加えて、各種イベントをトリガとして稼動するメール機能、さらには共通画面制御、ポインタ制御等のリアルタイム制御機能が要求される。これを概念的に図にすると図1.6-1のようになろう。

表1.6-1　ソフトウェア開発におけるグループワーク

ソフトウェア開発におけるグループワーク	通知	指示： ・プロセス実行指示 ・波及分析結果による指示等 通知： ・会議通知 ・ＤＲ通知 ・スケジュール管理 ・各種報告等 請求： ・審査請求（ドキュメント、ソースコード等々） ・購入請求 ・ベースライン化請求 ・テスト請求等
	対話	・会議、 ・設計レビュー ・相談（設計等） ・会話形式のプロトタイピング（GUIなどの設計等） ・会話形式のシミュレーション ・単純ヘルプ 等
	参照	・共同執筆 ・上流工程ドキュメント参照 ・他者設計参照（インターフェース設計等） ・方法論参照 ・各種管理データ参照 ・単純ヘルプ（キーボードなど）等

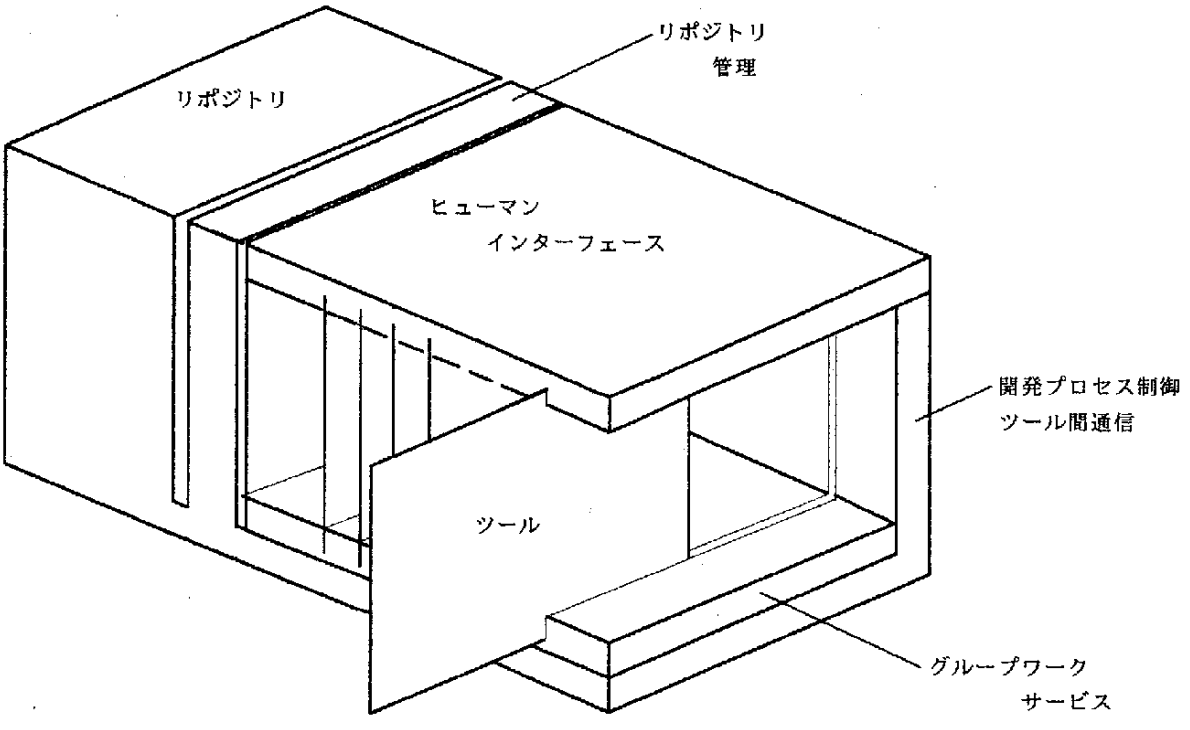


図1.6-1　グループワークを取り込んだCASE統合環境

1.6.2 CASE統合環境とグループワークのサポート状況

上記のように、現在CASE統合環境として仕様が公表されたり、インプルメントが行われたりしているシステムがいくつか存在する。この節では、それらのシステムついて、表1.6-1の3つの区分、すなわち「通知」「対話」「参照」について、具体的にどのレベルでどの程度サポートされているかを概観してみる。CASE統合環境それ自体はCASEツールを統合するためのフレームワークであるので、グループワークをサポートする個々のツールに言及している仕様はほとんどなく、したがってここでもソフトウェア開発におけるグループワークサポートのための「仕掛け」を主として見ていくことにする。

(1) AD／Cycle

AD／Cycleは、ソフトウェア開発の効率化を図りバックログを減らし、かつソフトウェアの品質を向上させるための切り札としてIBMが1989年9月に発表したアプリケーションソフトウェア開発のためのツールフレームワークである。これまでソフトウェア開発のライフサイクルを支えていた方法論やCASEツールの欠点は、データを共有できず、個々バラバラのツールが共存すること、開発作業のステップ間は人間が紙に書かれた情報を渡して次のステップが開始されること、ツール統合のための標準が存在しないこと、またたとえ統合環境として存在するものでも独自のインタフェースしかもっていないため、ツールの追加や他の環境との結合が難しかったりすることであると認識し、それを解決し生産性・品質を飛躍的に高め管理しやすい開発プロセスとするために開発したのがAD／Cycleである。そのねらいは、

- 最新のソフトウェア開発技術をツールによって提供する
- アプリケーション開発に関わる全てのデータを統合的に管理・制御する
- オープンアーキテクチャを実現して開発支援ツールの統合化をはかる

などであり、基本的に「開発支援用CASEツールを内臓したCASEツールのためのフレームワーク」である。

AD／Cycleのアーキテクチャは、もともとワークステーションの優れたGUI機能とメインフレームがもっている開発関連データの制御機能を融合するという発想を軸として設計された。マルチウィンドウを駆使したGUI機能はPersonal System／2で実現し、各種ツール間のコミュニケーション・連係・統合サービスを行うWSP／2 (WORKSTATION PLATFORM／2) 上ですべてのツールが稼動している。一方、開発関連のデータの共有メカニズム・制御機構はホストコンピュータがもつリポジトリサービスやライブラリサービスで実現している。リポジトリには、アプリケーション開発過程で発生するすべての開発関連のデータが蓄積され、どのツールからも自由にアクセスすることができる（現在のところ、上流から下流までの全てのデータが完全にリポジトリで管理されるところまではいっておらず、とくに下流工程のデータは、外部フォーマットで蓄積されている）。

AD／Cycle アーキテクチャの主要構成要素は、ツールの統合に必要な３つの要素（上記）をいずれも含んでおり、共通ユーザインタフェース、ワークステーションサービス、ワーク管理、インフォメーションモデル、ツール、リポジトリサービス、ライブラリサービスなどが挙げられる。統合されているツールのユーザインタフェースは、いずれも CUA (COMMON USER ACCESS) の規則・ガイドラインに沿った設計になっており、ツール間で互いに整合がとれたものになっている。またワークステーション上の表示にはプレゼンテーションマネジャが均一な表示フォーマットを提供している。ワーク管理は、ユーザの開発プロセスに合わせて作業をガイドする機能で、ユーザの方法論にあわせてツールの起動・終了が自動的に制御される。インフォメーションモデルでは、ユーザのソフトウェア開発に関連しているすべてのデータ、その属性およびデータ間の関連を定義することができ、それらの情報をツールが使うことになる。リポジトリサービスは、それらのデータを一元的管理するものであり、複数のユーザ・ツールに対して、データ定義、データ管理、データ共有、アクセス制御などを提供している。AD／Cycle アーキテクチャでは、統合ツールも重要な構成要素であり、企業のデータモデル構築用のツール、分析ツールなど上流工程支援のツールや、サードベンダ提供のツールも含めた下流工程支援のためのツールが整備されている。

このように広範なサービスを提供している AD／Cycle を、グループワークサポートという面からながめてみる。ソフトウェア開発におけるグループワークのうち、「通知」と「参照」の２つのサポートが実現されていることがわかる。まず、AD／Cycle の構成要素であるワーク管理がそれである。ソフトウェアの品質がコスト、スケジュール、生産性に根元的にかかわっており、品質を向上させるキーは開発プロセスの中にあるとの立場から、インフォメーションモデルを定義して生成物を明確にするとともに、その作業を実施するプロセスとそこで使用する手段を明らかにしていく。そのために個々のステップとその中で行う作業を定義し、つぎにそれらを総合した全体プロセスを定義して、ワークフローストラクチャを構成する。こうして定義されたプロセスは非常に複雑であり、中間生成物、最終生成物、チームのメンバ数などなどあらゆる要素がはいっている。これをプロセスモデルから読みとり、それを解釈し、人間系に表示し、開発作業のナビゲーションをするのがワーク管理サービスである。ただ、開発組織によってそのようなコンピュータによるナビゲーションを強く求める場合と、弱いナビゲーションの方が望ましい場合とがある。AD／Cycle では、そのあたりに柔軟性をもたせた運用ができるようになっている。

グループワークのうち「参照」もまた AD／Cycle においてサポートされている。開発作業社がツールを介してリポジトリ管理下にある任意のデータを参照する機能である。AD／Cycle でこの仕掛けを提供しているのがリポジトリマネジャである。リポジトリマネジャの機能は「仕様」と「ランタイムサービス」の２つに大きく分かれる。仕様では、データに対して３つのビューを定義している。ツール間共通の概念・意味を定義するビュー、採用しているデータベース依存部分を吸収するビュー、ツ

ール固有のビューである。一方、ランタイムサービスは、ツールのデータアクセスへの制御やユーザサービスであり、データのリード／ライト、アクセス制御、データの長期間のロック機能など、グループワークに必要な機能が提供されている。

AD／Cycle は、漸近的に環境を構築していくことを一つの方針としているので、仕様への記述が即インプリメントとはならないことはいうまでもなく、たとえば、現在のところ、ワークステーション上のツールからリポジトリ管理下のデータへのアクセスは、限られたツールでしか実現できないなどである。

(2) CIS ATIS (A TOOLS INTEGRATION STANDARD) (Software Backplane)

アメリカの CAD およびワークステーションベンダ Daisy Systems Corp.で働いていた数人の技術者が、新しい会社 Atherton Technology を設立したのは、1986年のことである。彼らは設立した会社を CASE ツールベンダとして育てるとともに、CASE ツールのフレームワークを提案しようと作業を開始した。これが、ATIS (もともと Atherton Tools Integration Standard) のはじめであり、ATIS インタフェースを設計するとともにそのインプリメンテーションとして Software Backplane を開発した。その後、Atherton Technology は、アメリカの CIS (CASE Integration Standard) Consortia に、ATIS をツールインタフェースの標準案として提案し採用された。これと前後して DEC 社が ATIS および Software Backplane の全面的なサポートを開始し、現在は、DEC と Atherton Technology が、ATIS と PCTE を融合しより多くのツールベンダが参画できるようなインタフェースをもった単一統合環境を構築しようと勢力的に活動している。

ATIS は、CASE統合環境を構築するためにリポジトリが持つべき機能を規定するとともに、CASE ツール統合のフレームワークを提供するものである。リポジトリ機能としては、

- ツールデータの管理
- データ操作の管理
- 共通データ形式の実現
- 継承関係の実現

を提案している。またCASE 統合環境のために、ツール制御のためのインタフェースを定義しているが、特徴的なのはそれを実現するためにオブジェクト指向モデルを採用している点であり、TYPE 階層（「通常の」オブジェクト指向のクラス階層）の形態で提供されている。ATIS は機能面から見ると、「モデル」という名の６つの独立した機能の階層構造を採っており、そのおのおのにインタフェースが提供されている。

- オブジェクトモデル

  最も低位に位置する基本となる層で、この中でツールが扱う基本的なオブジェクト、そのメソッド、アトリビュートを定義する。

・ バージョン管理モデル

ツールが扱うオブジェクトのバージョン管理を行う部分であり、複数ユーザが生成したバージョン等も管理している。このバージョン管理のインタフェースを標準化し、どのツールも同一の方法でバージョン管理ができるようにしている。

・ コンフィギュレーションモデル

複合オブジェクトなる概念を導入し、これを使って開発中のシステムの構成を記述し、管理を行う。

・ ワークフロー制御モデル

ツールが扱うオブジェクトを、開発ライフサイクルの中でどのように生成していくかを制御する機能であり、ユーザに特化したライフサイクルを柔軟に記述できるようなインタフェースが提案されている。ただ、ライフサイクルといっても、ツールが扱うオブジェクトの承認経路やオブジェクトと他のオブジェクトとの関係を定義できるわけで、ライフサイクルとはいえ、かなり限定された機能である。

・ ツール登録モデル

新しいツールを環境に導入する場合には、ツールそれ自身の登録と環境による認識、新しいツールが扱うオブジェクトの登録と環境による認識、新しいツールに対するメソッドやメッセージの認識等の手続きがあり、これらおのおのに対応するインタフェースを設定してやらなければならない。このモデルは、その設定のプロセスを記述したものである。

ATIS におけるツール間のコミュニケーションは、一つのツールがリポジトリの中のオブジェクトにメッセージを送り、それを他のツールが処理するという形態で行われ、ツールが直接他のツールを起動するというかたちではない。

・ IRDS モデル

ANS X3.138 の IRDS 用インタフェース

これらいくつかの層のうち、他の標準案と融合できそうなのがオブジェクトモデルであり、この部分を PCTE のオブジェクト管理システムと擦り合わせできないかと検討しているわけである。また、リポジトリとしては、IRDS モデル部分がそのためのインタフェースを提供しているといえる。

これが ATIS の概略であるが、これをグループワークをサポートするメカニズムが提供されているかどうかという面から見てみる。ATIS の機能の一つにワークフロー制御があるが、これは、ソフトウェア開発におけるグループワークのうちの「通知」サービスにあたる。たとえばあるユーザが一つのベースラインを参照して自分のオブジェクトの作業を行っているとき、そのベースラインに変更が加わった場合には、ベースラインを参照しているすべてのオブジェクトの担当者に変更通知が送られなければならない。これを実行するのが、ワークフロー制御である。一方、リポジトリを介して共同執

筆をしたり、他の作業者の生成物を参照する「参照」サービスを実現する機構も用意されている。バージョンモデル（機能）のチェックイン／チェックアウト機能は、バージョン管理に排他制御をとりいれたもので、作業者はRESERVEを発行しなければ目的とするオブジェクトに対して変更を行えず、変更後、REPLACEを発行しなければそのオブジェクトは登録されないので、RESERVE中はロックがかかっていることになる。

(3) PCTE (PORTABLE COMMON TOOL ENVIRONMENT) および ALF

1970年台に米国防総省(DoD)が作成したSTONEMAN文書は、既にプログラミング用支援環境の必要性を認識しており、基本的なツールのインタフェースを規定している。この文書から大きな影響を受けてスタートしたのがECのPCTEプロジェクトである。PCTEは、ハードウェアに依存せず、ツールベンダなどが採用できる標準のツールインタフェースを規定し、ツールのポータビリティを確保することをねらいとした標準インタフェースであり、PCTE全体としてツール統合環境の枠組みを提供している。標準ツールインタフェース設定に当たっての基本的考え方は、次の５つである。

- 特定の開発言語にとらわれないインタフェースとする
- 開発環境の中の全てのツールのインタフェースを規定する
  （ただし、例外はインプリメンテーションに依存する管理ツール）
- 既存のツールが容易に移行できるインタフェースとする
- 分散ハードウェア上で実現できるインタフェースとする
- 方針と具体的メカニズムとは明確に分離する

PCTEのインタフェースサービスはすべて、ライブラリファンクションの形で提供されており、PCTEのC言語バインディングで約260のファンクションからなっている。提供されるサービス内容は、

- データ管理

  OMS (Object Management Sysytem) により環境内のデータの定義、操作を行う。
- ツール実行管理

  プログラムをプロセスとして実行するとともに、プロセス管理、プロセス間同期、プロセス間コミュニケーションを行う。
- 分散環境管理

  ネットワーク管理などを行う。プロセス管理、プロセス間通信などが全ノードにトランスペアレントになっている。
- ユーザインタフェースサービス

  ツールベンダがツールを開発するためのインタフェースを提供している。

PCTEの機能をグループワークの面から見てみると、先ず、「参照」サービスがあげられる。環境内のデータの管理・操作を行うOMSが、データへのアクセス権限の設定、並行作業制御、オブジェクト

の分散化への対応などを実現しており、グループワークにとって必要なリポジトリ機能を備えている。データモデルは、ER モデルにもとづいたモデルを採用し、その上でデータの修正に対する一貫性の管理を実現しているが、スキーマ定義がファイルの粒度しかもち得ないので、グループワークによる共同執筆のような作業を念頭においた場合には、限定されたものになると考えられる。

その他のグループワークサービス、すなわち、「通知」および「対話サービス」をサポートする特別な仕掛については、仕様の中に見いだすことはできない。

PCTE の機能拡張を目的としたプロジェクトに ALF (Accueil de Logiciel Futur) がある。ALFプロジェクトは、PCTE の仕様には存在しなかったソフトウェア開発プロセスを解決するものである。このプロジェクトでは、プロセス記述言語を定義し、それを用いて MASP (Models for Assisted Software Processes) を記述し、実行するための環境を PCTE の上に構築するのがねらいである。特定の組織は、先ずこの環境を利用してその組織に適合した開発プロセスモデルを記述する。次に他の情報も組み入れてモデルの具現化手続きを行うと、ASP (Assisted Software Process) すなわち、開発プロセスが完成する。この開発プロセスと、PCTE が提供しているリポジトリの諸機能とを合わせれば、その組織に特化した IPSE (Integrated Project Support Environment) を構築することができる。

ソフトウェア開発のグループワーク面からみると、ALF によって実現される開発プロセスが PCTE に加わることで、「通知」サービスの一部が実現されたことになる。すなわち、開発作業者に対して、どのタイミングでどのようなツールを起動し、どのようなドキュメントを作成すべきかを指示していくなど、ソフトウェア開発の全てのフェーズにおいて、作業者に対して作業の「水先案内」と制御を行うものである。このようなサービスを行うためには、PCTE のリポジトリの諸機能と連係をとらなければならないことは言うまでもない。

以上見てきたように、現在公表されている CASE 統合環境においては、主としてリポジトリの機能としてグループワークをサポートする「仕掛け」がインプリメントされているにすぎない。ここでみた CASE 統合環境は、現在最も進んでいると考えられている統合環境であるが、まだいずれもユーザによる評価・フィードバックの段階であるといえ、大規模ソフトウェア開発への適用は少し先のことになると思われる。一方、グループワーク、特に「ソフトウェア開発におけるグループワーク」の研究はまだ歴史も浅く、十分な成果が出てくるのはこれからであると考えられる。ソフトウェア開発におけるグループワークという大きな側面を考えるなら、両者は独立に作業を進めるのではなく、本来協調して発展していくべきものであり、今後の融合が待たれる。

【参考文献】

[松本90] 松本正雄 : CASE統合環境のためのインターフェースの標準化現状, 情報処理, Vol.31,No.8, pp. 1086-1094 (1990).

[外山91] 外山勝保他：リポジトリ仕様の比較, 情報処理学会ソフトウェア工学研究会, 80-5, pp. 35-42 (1991).

[(社)日本電子工業振興協会] ソフトウェアエンジニアリングに関する調査報告書ーグループウェアへの期待ー, 平成３年５月, IBM System Journal, Vol.29,No.2 (1990).

[Beyer90] Beyer,H.R.: Proposal for Extending Dictionary Standards to Support CASE, Information Resource Dictionary System ATIS, ANSI X3H4 Working Draft, February (1990).

[Thomas89] Thomas, I.: PCTE Intrfaces:Supporting Tools in Software-Engineering Environments, IEEE SOFTWARE, November (1989).

[Derniame91] Derniame,J-C.,et al.: Development of Process-centered IPSEs in the ALF project, PCTE'91, Morning Sessions (1991).

[Soufflet91] Soufflet,D.: CASE INTEGRATION SERVICES(CIS) STANDARDS COMMITTEE, PCTE NEWSLETTER, No.6, pp. 11, April (1991).

[PCTE90] PORTABLE COMMON TOOL ENVIRONMENT (PCTE) Abstract Specification, STANDARD ECMA-149, December (1990).

1.7 グループワーク支援システムの適用分野

企業活動は、企業、事業部、部、チームなど一つの目的をもったグループの協調作業である。 グループウェアとはこれらのグループ間での情報や意見の交換を促す、協調作業（グループワーク）を効率よく支援するための道具である。

これらのグループワークに注目してみると、事務処理から折衝にいたるまで、どの仕事においても他のグループとの情報交換を頻繁に行い、そのために資料を作りコピーを配り、スケジュールを確かめ相手先の所まで足を運ぶといった作業の繰り返しをおこなっている。グループウェアは、これらの情報伝達の手段を見直しさらにコミュニケーションを円滑におこなわせることを主眼として研究・開発されている。

本節では、ソフトウェア開発業務を取り上げてこの中におけるグループの構成と実際の作業工程をみつめ、グループウェアの背景と適用分野を考えてみる。

1.7.1 グループワーク支援の背景

(1) 分散開発と地方分散

ダウンサイジングの時流を反映して、業務アプリケーションの主流はホスト一端末型の集中オンラインシステムからワークステーション主体の分散システムに移行しつつある。これに伴い、ソフトウェア開発も分散開発の環境下で行う必然性が高まっている。

また、大規模システムの開発を受託したメーカやソフトハウスは、集中的に発生する開発要員の不足を確保するために、コスト的にもメリットのある地方のソフトウェアハウスや系列会社などに分散して開発を行う傾向を強めている。一方、首都圏で従事する地方出身の技術者にもUターンを希望する者が潜在的なニーズも含めれば少なくない。メーカやソフトハウスは優秀な技術者を確保し、有効に生かす道を地方に求め模索している。

その反面、遠隔地であるが故の距離的・時間的なコミュニケーションの壁により次のような状況も多く存在する。

- 仕様変更の依頼／回答をFAXによる往復で交換するが書面の品質や保存性も悪い。
- 製造工程中も仕様の問い合わせや、障害原因の究明などにおいて電話やFAXによる対応では情報交換の効率が悪く、品質や生産性の低下を招きやすい。
- 進捗会議や仕様説明会（ウォークスルー）などにおいて対面型の会話が重要な要素をもつため、その会議のために出張がたびたび発生する。
- 開発業務の上流工程はコミュニケーションに依存する作業比率が高いため、遠隔地での開発にそぐわない。そのため、地方においては下流工程の仕事が占める割合が多く、技術や業務ノウハウの蓄積が難かしい。
- 上流工程の要員として首都圏に逆Uターンし長期滞在を余儀なくされる。

このようなコミュニケーション上のハンデは、首都圏－地方間のみでなく分散開発環境に共通の問題でもあり、グループウェアはこれらを解決するための重要な道具となる。

(2) 開発作業における情報量の増加

ソフトウェア開発の作業は、仕様情報の入力・加工・出力のサイクルを繰り返し、いくつかの工程をくぐり抜けながら、プログラムを主としたソフトウェアを作りだしていく。その各工程においては、さまざまな人々がそれぞれの役割と立場で参加し、その中で、加工の主体である仕様情報の他にも、プロジェクト管理情報、技術情報など非常に多くの情報を理解し・取込みながら、次の工程へと必要な情報を出力していく。

特に、ソフトウェア開発において品質管理が重視されるなか、開発プロジェクトの現場ではユーザ・メーカが変わる毎に、それぞれの開発支援、標準手順、開発ツールなどが適用され、担当者はその都度にそれらの情報を咀嚼し開発を進めているのが現状である。また、かれらの作業そのものもますます分業化され、部品化・ブラックボックス化されたコンポーネントを効率よく利用することが義務付けられている。

まさに、開発担当者はグループのメンバとして様々なコミュニケーションを交わしながら、品質を維持するためにそれぞれに関わる情報の渦のなかに身をおいて作業をしているわけであり、いかに「情報を咀嚼する」が生産性を決定づける大きな要因となっている。グループウェアはその道具とし

て、部品の仕様検索や担当者同士のコミュニケーションなどの手助けをする機能を提供することができる。

(3) 技術革新の土壌

グループウェアは電子情報を伝達情報として、ネットワークを介し複数の人間集団の協調作業を支援するツールである。 伝えたい情報を紙に書いたり読んだり（さらには見たり聞いたり）するごとくに電子情報を手軽にハンドリングできてこそ、グループウェアの利用価値がある。

(a) ワークステーション(パソコン)の普及

我々の仕事場では、パソコンやワープロがすでに日常のなかで身近な道具として活躍している。コンピュータ技術は飛躍的な発展をして、パソコンやワークステーションの高性能化・低廉価化が一人に一台という利用環境をわれわれに提供しつつある。

また、アイコンとマウス操作やマルチウィンドウによる画面表示、ビットマップディスレイの採用など画面を操作する上でのインタフェース、GUI (Graphical User Interface) 技術も非常に向上している。

(b) ネットワーク技術の発達

加えて、パソコン通信やLAN (構内ネットワーク) にみられるような通信環境の発達もめざましいものがあり、個人利用の目的で使われていたパソコン同士をネットワークで接続して情報を交換することが容易にできるようになった。

LANについては、イーサネットやトークンリングのようにネットワーク方式の業界標準が進んでおり、その上でLANを運用するネットワークOSも数々リリースされている。これらは、ネットワーク間で共用するファイルやプリンタなどのリソースを効率よく利用できるようなしくみを提供している。ビル内などの狭い地域で通信を行う場合はLANで、さらに全国を結ぶような広域的なものはWAN (広域ネットワーク) を用いて、異なるLANの相互接続も行えるようになってきている。

通信網については従来の電話網やパケット網が提供するサービスを統合したISDN (統合サービスデジタル網) の整備が進められており、回線速度・マルチメディア通信への対応・経済性など一段と優れた通信性能を提供する通信サービスに期待がよせられる。

(c) マルチメディアへのニーズ

「伝えたい情報」には、文字以外に絵・グラフ・写真さらには音・動画などのイメージ情報＝マルチメディアがある。グループウェアの背景には、マルチメディアを必要とする強いニーズがある。マルチメディアによりグループ間のコミュニケーション・インタフェースは向上し、加えて蓄積・参照情報の多様性が増すことにより利用者に与える恩恵も増大する。マルチメディア処理は以下のような技術の進歩によって実現され、また現在も急速に進歩しつづけている。

- イメージを処理するCPU処理性能の飛躍的な向上。
- CD-ROM、光ディスクなどマルチメディア対応の大容量データ蓄積装置の開発。

・ マウス、タブレット、イメージ・スキャナー、ビデオカメラ、高品質プリンタなどのイメージ入出力装置の技術進歩。

・ 画像圧縮技術の研究と標準化の発展。

・ ISDN (サービス統合デジタル網) によるマルチメディア通信。

(d) オープンシステムの進展

ワークステーションやネットワーク技術の発展に相まって、マルチベンダをキーワードとする異機種間コンピュータの相互接続やソフトウェアやデータの互換などを実現するオープンシステムが市場に普及しつつある。 オープンシステムではOSをはじめとして、ユーザ・インタフェースの機能を高めるGUI（上述）、アプリケーションプログラムの移植性を保証するAPI (Application Program Interface)、クライアント・サーバモデルを土台とした分散コンピューティング環境、SQLにみられるデータベース操作言語など、あらゆる角度から標準の設定が進められている。通信についてはOSI (Open Systems' Interconnection) により文書交換や電子メールなどさまざまな用途の通信に対応した標準が設定されている。

ソフトウェア開発に適用しようとするグループウェアは、異なる組織（企業も含め）の間で、異なる機種（ベンダ）を前提としたコンピュータやネットワークなどの資源を介して、共通の情報を自由に交換することを基盤とする道具であることから、オープンシステムは重要な時代背景となる。

1.7.2 情報の広場としてのグループウェア

理想的なグループウェアとは、異なる場所（部屋／階／ビル／地域／国）にいる複数の利用者が、その距離的あるいは時間（時刻）的なギャップを意識することなく、あたかも目の前（隣）にいて互いに会話やメモの手渡しを行うがごとく、望んだ情報を・欲しい時に・正確に・迅速に、伝達・共有（再利用）・整理することを助ける道具である。「目の前で」「望んだ情報を」とは相手の表情や声も意味し、文字やグラフ以外にも音声・映像（静画、動画）による情報交換機能も必要となる。その道具についてキーワードを列挙してみると以下のようなものがある。

・ 情　報　　：　文字、グラフ、絵、写真、ビデオ、音声、音楽　等

・ 機　器　　：　キーボード、タブレット、イメージスキャナ、ビデオカメラ、マイク、高品質ディスプレイ、ＣＤ－ＲＯＭ、光ディスク　等

・ システム　：　電子メール、ボイスメール、電子掲示板、電子会議室（共用スクリーン、共用ウインドウ）、遠隔会議（電話会議、テレビ会議、ワークステーション会議）、非同期会議支援、会議準備支援、協同文書執筆、メッセージ構造化、共用ファイル、ハイパーテキスト、オブジェクト指向データベース　等

・　ネットワーク：　LAN、WAN、FDDI、ISDN　等

グループウェアの広場には、このような多種にわたる情報や道具が用意されており、対象となる業務の内容に応じてこれらを統合し、グループ間の協調作業を支援しようとするものである。

現在、グループウェアについては各国であらゆる角度から研究が進められており、会議支援システムや協同作業支援システムなどさまざまな分類に属する製品が開発されている。その一部を以下に紹介する。詳細については別章を参照されたい。

(1)　同室会議支援：　GDSS、Colab、Project Nick

(2)　遠隔会議支援：　Cognoter、CRUISER、MERMAID、Rapport

(3)　電子メール、電子掲示板：　CC:Mail、GroupTalk

(4)　協同文書執筆支援（グループエディタ）：　Quilt、PREP

(5)　非同期討論支援（グループメモリシステム）：　gIBIS

(6)　メッセージ構造化支援（情報フィルタリング）：　InformationLens、ObjectLens

1.7.3　システム開発の現場におけるグループウェアの利用

分散開発によりソフトウェア開発を行う場合のパートナー同志は、おたがいの顔が見えなくなるほどに距離を隔てた時から、そのコミュニケーションのための時間や労力が増し、少しずつ協調作業に支障をきたしはじめる。

部屋、フロア、ビル、・・と離れるほどにその影響は大きくなる。その情景をいくつか思い浮かべながらグループウェアの適用例を想定し、その効果を探ってみることにする。

(1)　情景１：「進捗会議の日に」

【適用前】甲社のリーダは毎週金曜日になると、各チームから各モジュールの試験消化状況と不良発生状況の報告をうけてパソコンで集計を行い全体の消化・発生曲線と週間状況報告をまとめ、翌週の月曜日に発注元である乙社に向かう。片道40分程かけて定刻に着いたのだが、同じく開発を請け負っている３社分の進捗会議が軒並み遅れ、結局２時間待たされた後に会議に入った。資料のコピーを８部揃え出席者に配り進捗状況の報告を始めた。・・・・

【適用例】甲社において、グループウェアのメニューに登録されているプロジェクト管理の専用ツールを使い、プログラマは試験消化状況と不良発生状況などの進捗情報を自分のWS (ワークステーション) から入力する。プロジェクリーダとチームリーダはWSに向かいチームの状況を討議・対策をたて、会議の前日迄にこれに認証を行い、週間報告書をまとめ電子メールで乙社に送信する。会議の当日、甲社のリーダはWSから電子掲示板に表示された工程会議全体の進行状況を見ながら出番を待つ。連絡を受けWSから遠隔会議システムにエントリーし、WSのマルチウィンドウに映し出される消

化・発生曲線を説明しながらWSのマイクを通して説明を始めた。乙社では進捗責任者が同じ情報の表示されるWSに見入りながら報告を聞く。・・・

【効　果】

① ネットワークを介してプログラマにより入力された進捗情報が元となり、チーム単位や全体としての集計やグラフの作成は自動化され、集計のために手を患わすことがない。

② 作成した報告書は即時に相手に送信されペーパレス化が図れるうえ、過去の報告書の検索も可能である。

③ 会議には自席で参加することができ、往復時間が不要となり待ちの時間さえも有効に過ごすことができる。会議においては、テレポインティング機能などを利用しながら対面での距離感で会話をすることができる。

(2) 情景2：「原因不明の障害を追う」

【適用前】乙社のAは本番稼働した業務に発生した障害を調査することになった。障害は緊急を要する内容であった。ダンプリストをもとにマニュアルを取り出して調べたが、障害の切りわけがうまくいかない。甲社に委託したプログラムに疑いがあり、リストの必要箇所や関連メモをFAXで送信し、担当者Bと電話を使い前後の状況を説明し、意見を交わしながら調査を進めた。Bは手持ちの仕様書とFAX内容をつきあわせ、状況を推測しながら障害箇所の特定を急ぐが、情報量が不足しているためなかなか解明できない。プログラムには特に問題は見当たらず、サーバ側のファイルドライバとのインタフェースに疑いが出てきた。さっそく、サーバを開発した丙社の担当者Cに連絡を取り、同様の資料をFAXで送信し・・・。

【適用例】乙社のAは、その障害についてダンプリストなどのEDP情報をできるだけ集め連絡用ファイルに一時的に保存した。そして、WSからグループウェアの情報検索メニューにある障害状況診断サービスにアクセスし、障害の分類や現象コードなどからハイパーテキストを操作し障害の切りわけをする。Aは甲社の担当者Bと連絡をとり、ふたりはそれぞれのWSからグループウェアのワークステション会議にアクセスし、同期したウィンドウ画面上に、連絡用ファイルから調査経過も追加された障害状況を表示し、必要に応じてテレポインティング機能も使用し、互いに気になる箇所を指摘し合う。マニュアルや仕様書をマルチウィンドウに同時に参照することもある。必要であれば丙社のCにもこの会議に参加してもらう。・・・

【効　果】

① 障害状況の詳細な情報をお互いに共有でき、同じ場所で会話を交わしているように密度の高い協同作業を行うことができる。

② はじめて経験する障害であっても原因の究明を迅速に行うことができる。場合によって技術相談員にネットワークを通して指導してもらうことも可能である。マニュアル管理面でも省スペース化

が図れる。

(3) 情景３：「変更依頼書を管理する」

【適用前】乙社のＬは、担当するＸＸ照会サブシステムに仕様の変更が発生したために２件の仕様変更を甲社に依頼することになった。変更依頼用紙にサブシステム名、変更内容、変更期限など所定の項目を記入したあと、キャビネットより変更依頼書のキングファイルを取り出し、変更依頼NOをとりつけ仕様変更一覧に発行日、担当名、タイトルなどを記入する。上司の照査印をとりつけコピーをとった後で、甲社宛のメイル箱に投入し、依頼書の控えをキングファイルに綴じ込んだ。同時に、以前発行した３件の変更依頼について調べたところ、未回答のものが１件あることが判り、電話により甲社の担当者に催促をした。・・・

【適用例】乙社のＬは、グループウェアのメニューより入出状管理ツールを選び変更依頼書を作成する。このツールはメッセージの構造、業務処理手順の構造などが手続き化されている電子メールシステムであり、WS の画面指示に従って所定の項目を入力していけばよい。登録された書式イメージで入力内容は確認できる上、変更依頼NOも自動的に採番される。作成後、次工程指示をすれば登録された業務手順にそって上司の照査→甲社宛の送信と電子メールがリレーされ、甲社においても同ツールで管理される。また、Ｌが作成した変更依頼書はすべて一覧検索ができ、処理状況の遷移も把握できる。・・

【効 果】

① メッセージ構造化機能により依頼書の起票は簡単となり、また書式との合成により文書イメージで内容の確認ができる。また書類はペーパレス化され、省スペース化が促進される。

② 業務手順が手続化されるため、煩雑な入出状処理からも開放され的確な依頼・回答管理ができる。

(4) 情景４：「開発基準を学習する」

【適用前】甲社のＱは新しくプロジェクトに配属され、上司より開発基準を勉強するようにいわれた。 基準書の資料はキングファイルで４冊あり、保管棚に２セットある。

貸出しを受けて読みはじめると、文中に出てくる言葉が初めてのものが多く、先輩に尋ねたり専門書を調べたりしながら理解を進めた。途中、他の資料を参照する必要も多く閲覧するときの場所もかさむ。どうしても分からないところはメモにとどめ専任者に教えてもらうことにする。・・・

【適用例】甲社のＱは、グループウェアのメニューより文書検索ツールを選び、ハイパーテキスト化された開発基準関連ドキュメントを WS より参照する。同ドキュメントは各種手引き書やマニュアルとともにネットワーク下の共用ファイルに収められている。文中に不明な言葉が出てきたり、他文献の参考があればマウスでクリックをして、マルチウィンドウ上にその解説を表示して同時に理解をする。質問事項は別ウィンドウ上で入力して、あとでまとめて専任者に電子メールで送信しておく。

・・・

【効 果】

① 基準書などの共通情報はすべて共用ファイル化され、ペーパレスで管理できる。

② マニュアルや資料のあちこちをパラパラとめくるような煩雑さから開放され、自由に関連情報を理解しながら本体の書類を読み進むことができる。ハイパーメディアであればさらに利便性は高まる。

③ 不明点に関する質問は、急ぐのでなければ電子メールで質問／回答を交換すれば充分であろう。

以上、従来の業務形態について適用した例をいくつか想定してみた。さらに有効な適用方法もあろうが、いずれにしてもソフトウェア開発においては多くの適用分野があることがわかると思う。

1.7.4 今後の課題

上述したように、グループウェアはいろいろな背景をもって登場し、多くの研究と技術革新によって実現されつつある。遠隔テレビ会議に代表されるマルチメディア・サービスは設備・運用ともにまだ高価なものとなるが、さらなる技術革新がこれらを我々の身近に引き寄せてくれるものと思う。また一方では、費用対効果をみつめながら、必要なものが取捨択一される形でグループウェアは徐々に普及していくものと思われる。

グループウェアをソフトウェア開発の業務に適用するにあたっては、いろいろな課題が存在するが、ここでは一般的な問題も含めてそのいくつかを述べてみる。

(1) 電子メールにみる冗長情報

ソフトウェア開発においてグループウェアが適用される場合、同報通信も含めた電子メールの交換はかなりの頻度を占めるものと思われる。しかし、情報の交換が自由におこなえることから、取り交わされる情報には重要でない冗長なメールも混在する。その内容に関する重要性や緊急度については受取人が内容を見ないかぎり、要・不要を判断できず、自動的なふるいにかけることが難しい。また、メール到着を受信者に対し物理的にどう知らせるか、つまりいつ見てもらえるかが重要な意味を持つ場合に、受信者が確実に認知できなくてはならない。山積みになった電子メールの中に、極めて緊急な情報が埋もれ不測の事態も起きかねない。

(2) セキュリティ機能

グループウェアはネットワークをフルに活用した分散コンピューティングの一つの形態である。このサービス下では、ネットワークを介してあらゆるレベルのメンバ間で様々なメッセージが交わされることになる。また、ネットワーク下に管理されている情報・資源についても同様である。ネットワークや情報の利用が「オープン」であるがゆえの悪用を防ぐために万全のセキュリティ機能をサポートする必要がある。

(3) プロジェクト管理ツール

1.7.3 の適用例においてもふれたが、ソフトウェア開発においては開発の進捗状況や品質の管理などを行うプロジェクト管理ツールは欠かせない道具のひとつである。当面は独自のツールが用意され、個々に適用されていくものと思われるが、ソフトウェア業界に共通なツールが用意され普及すれば、ソフトウェア開発における業界全体としての生産性も飛躍的に向上するものと思われる。

(4) 上流工程の支援ツール

グループウェアは、分散システム開発の製造工程に多くの適用分野を見ることができる。しかし、1.7.1 において述べたように地方分散を本当の意味で実現させるためには、分析・設計などの上流工程を分散環境において支援するグループウェアのしくみを発展・充実させることが重要である。

(5) ソフトウェア開発のグループ特性

分散システム開発の体制においては、グループウェアが適用できる分野は幅広く考えられる。しかし、それに従事する人々のグループは、同一企業内とは限らず、むしろ発注する側とされる側、管理する側とされる側というケースが多く、請負い形態の立場が明確である。一般的に、発注側がグループウェアのツール環境を用意し受け手側がそれに応じた体制を整えることになろう。ここで大事な事は、グループウェアが「グループを管理する道具」ではないことをお互いに認識することである。グループウェアの目的は、グループ間の協調作業を支援することであり、発注側はこの点によく留意して運用すべきであり、受け手側も協力的に参加することが必要である。その時にグループウェアの機能を最大に引き出すことができる。当初は導入の効果というものが見えにくい部分もあろうが、充分な説明と理解なくしては、グループウェアも掛け声だけに終わってしまうことになろう。

(6) グループウェアがもたらす作業形態の変革

グループウェアは在宅勤務などさまざまな仕事の形態の可能性を我々に与えてくれる。また、さまざまなコミュニケーションの形態が可能になることからも、それに従事できる人の層も拡がってくる。つまり、グループウェアはそのひとが持つ個性や立場を生かし、有効にその力を発揮できる活躍の場を提供することになる。

【参考文献】

[松下91] 松下　温（編著）：グループウェア入門, オーム社 (1991).

[川面91] 川面　恵司（監修）：２００１年のコンピュータ, KKベストブック (1991).

[末松91] 末松 千尋（著）：オープンシステム入門, ダイヤモンド社 (1991).

1.8 ネットワーク環境における分散開発の将来

1.8.1 コンピュータネットワーク技術の最近の進展

コンピュータネットワークの技術は、最近もさまざまな面で急速に進展しつつある。これは、コンピュータを利用する環境を構築したり、利用したりするユーザにとっては、有利な事であり、それを踏まえたシステムの開発が重要になってくる。

ソフトウェア開発は、それを支援するコンピュータシステムに負うところが大きい。コンピュータシステムが十分高性能で利用しやすくできていれば、ソフトウェア開発支援環境もよいものができる可能性が大きい。ネットワーク技術が発展すれば、それに応じて、分散的なソフトウェア開発が更に強化されるわけであり、今日のような技術的な状況では、それを踏まえた検討、体系化などが必要になる。

最近のネットワーク技術で特に注目に値する技術には、次のようなものがある。

(1) ネットワークの高速化

ネットワークは、そのハードウェアの高速化が見られる。LANで言えば、100MBPSのFDDIが普及しつつある。これにより、マルチメディア化の進展も期待できよう。

一方、WANでは、ISDNが日本では普及しつつある。これは、65KBPSがかなり低価格で利用可能となり、従来のモデムを介した通信とは異なるコンピュータネットワークの構築が可能となる。さらに近い将来は、ネットワークの高速化の目標として1GBPSというレベルのネットワーク通信のサービスも可能となってくる。

(2) ネットワークの柔軟化

ネットワークの構築は、例えばLANではイーサネットでも10BASE5や10BASE2のような同軸ケーブル敷設を中心とした構造をとっていた。これに対して、10BASETのようなより線を利用したネットワークの敷設が普及しつつある。これは、端子ボックスなどを利用してそのトポロジを変更しやすい構造になっている。今後、インテリジェント化する建造物に対してそれを利用する組織に対応したネットワークを構築しやすいことは、重要であり、その意味ではネットワークトポロジの柔軟な構成を支援する技術は重要である。

(3) ネットワークの多様化

ネットワークを利用する場面やアプリケーションは、これから急増すると思われるが、そこでは、いろいろの応用が目標となろう。例えば、マルチメディアは一つの応用分野の拡大であるが、これをネットワークで支援するには実時間ネットワークが必要となる。これは、新しいネットワーク技術の分野であり、完全な解はこれからの研究成果を待つところが大きい。

(4) 統合的なネットワーク環境の実現可能性

ネットワークは、これからのコンピュータ利用環境では必須のものとなり、ワークステーションだ

けでなく、パソコン、ラップトップコンピュータ、メインフレームコンピュータなどさまざまな型の機種のコンピュータがそのノードとして混在している。また、その応用範囲も広範なものがある。

このような状況では、ネットワークを基盤とした分散環境がコンピュータシステムの中心になることは目に見えており、多様な応用、多様なコンピュータ、多様なネットワークを基盤とし、その上に多様なアプリケーションソフトウェアが稼働し相互作用を行っていくことになる。そこでは、十分な検討を加えたシステムの考え方と構築が必要となるが、その中核になるものは、ハード、ソフトを含めたシステムの統合化であろう。

(5) 分散環境基本ソフトウェアの進展

分散環境は、これから本格的な機能を持ったシステムが出現していく。それを実現する基本ソフトウェアとして、分散ファイルシステム、分散プロセスシステムなどが必要になる。既に、そのような製品も出現しているが、更に広域化への対応も必要となり、それらの基本ソフトウェアの開発が待たれる。

(6) 標準化の進展

ネットワークによる分散環境は、異機種間の接続が必須となり、それを支える標準化が必要となる。既に国際標準の場ではOSIを中心としたネットワーク機能の標準化が行われているが、これを更に推進し、相互接続性、相互運用性を向上させる必要がある。

1.8.2 アナログネットワーク技術の進展

ネットワークでも、TV系のネットワーク技術もかなりの勢いで進展しつつある。例えば、以下に述べるようなものが見られる。

(1) CATV系の発達

TVの難視聴区域への対策から始まったCATVは、いまや娯楽や教養を提供する地域CATV系の普及へと進展してきた。これは、双方向性を含めれば、十分効率のよいコミュニケーションの手段となる。

(2) 放送衛星の進展

人工衛星を利用した放送系により、効率よく大域的放送型コミュニケーションが可能となる。これは、アナログ型通信だけでなく、ディジタル系のコンピュータネットワークにも利用できる。

(3) TV電話

ある程度の高速の通信網を利用すれば、映像を送れるTV電話が可能となる。TV電話機器は、ビデオカメラの付随した個人用端末と考えることもできる。これが低価格で提供されれば、アナログ情報の電送も取り入れたコミュニケーションが可能となる。

(4) TV会議系

TV会議は、従来から特定の場面で利用されてきており、それなりの実績を上げつつある。これは、

ディジタル系のネットワークと組み合わせて、新しいコミュニケーションを作り上げることができる。

1.8.3 新しいコミュニケーションの形態

ネットワークシステムの進展により、それらを利用した新しいコミュニケーションの発展も可能となる。例えば、以下に示すようなコミュニケーションが可能となる。

(1) ディジタル系とアナログ系の混合したコミュニケーション

テキストなどの情報は、ディジタル系のネットワークで送信するのが能率がよいが、図形情報、動画などは、アナログ系のネットワークで送信した方が能率がよい。これらのネットワーク系は相互に補って動作すればそれぞれの長所を発揮できる。これを追究していけば、総合的なネットワーク系が実現できる。そのためには、それぞれの混在が自由にできるようなソフトウェアユーティリティ、エンドユーザ向けの低価格の装置の開発などが必要である。従来では、TV会議などは、アナログ系を中心に行われて来たが、融合した遠隔会議システムが考えられる。これは、従来のディジタル系ネットワークでおこなれていたグループウェアを利用したものに発展できるであろう。

(2) 高速な広域ネットワークの実現

従来、広域のネットワークはそれほど高速なデータ通信を実現できなかった。しかし、ISDNなどの出現により、広域ネットワークの高速化が進んでいる。そこでは、ローカルな世界でのネットワークの常識が適用できるようになる。

例えば、ファイル転送の操作をftpコマンドで行うのか良いのか、あるいは単にcopyコマンドで処理できればよいのか、当然後者の方がずっと能率がよい。広域ネットワークでも、copyコマンドでファイル転送が行える様にすべきである。また、このような方向を追究すると、広域の分散環境が実現できる。

また、米国で始まっている1GBのネットワーク構想は、将来の広域ネットワークのあり方を示している。これが実現できれば、広域のマルチメディアネットワーク環境が実現できるわけであり、さまざまなコミュニケーションが可能となる。

(3) マルチメディア環境とネットワーク

前記(1)にも関連するが、アナログ系のネットワークでは、各種メディアの操作もかなり容易にできる。これをディジタル系だけで行うとかなりの性能を要求される。特に、ネットワーク系では、マルチメディア環境で必須となる実時間処理がやや面倒になる。

マルチメディア環境がネットワーク上で保証されれば、従来のグループウェアがより有効に働く場面が多くなる。電子メール、電子ニュース、遠隔会議や討論など、画像や動画、あるいはユーザ同士の顔などを自由に扱えれば、面白い効果が現れる。

(4) 人工現実感とグループウェア

広域の高速ネットワークやアナログネットワークの混在により、広域でのマルチメディア環境が実現できれば、グループウェアのさまざまな発展も可能であろう。例えば、広域環境で人工現実感が実現できれば、遠隔地間でのコミュニケーションに幅を与えることができる。

2．分散開発におけるグループワーク支援の実現例

2．分散開発におけるグループワーク支援の実現例

2.1　D²による分散開発支援

本節では、ソフトウェア分散開発支援システムD² (D square: Distributed environment for software Distributed development) の開発思想と狙いどころなどについて紹介する。D²は現実のソフトウェア開発現場での利用を目指しているためにインフラに近いところに注力している。次いで同期及び非同期のエージェント指向コミュニケーションシステムと分散開発システムにおける基本的機密保持メカニズムとしてのユーザ認証システムについての現状と今後の展望を述べる。

さてCSCW (Computer Supported Cooperative Work) という言葉と共にソフトウェアの分散開発が注目されてきている。開発の分散化は本質的には効率が悪いが、それにも拘らず世界の動向は分散開発に向いている。この理由は次の２点に集約できると考えられる。

(1)　社会情勢による開発拠点の分散化

(2)　ワークステーション等の分散開発支援技術の進歩

第一の理由は、地方の時代と言う言葉に代表される傾向で、技術者が各地域に分散しつつあることに対応するためである。これは技術者獲得のためにやむを得ぬこととも考えられるが、むしろ積極的な観点から見て我々が取るべきグローバル化の必要性によるとすべきであろう。すなわち、ソフトウェア開発の資源である人とその技術をそこに存在するがままに有効活用することが社会あるいは組織のグローバル化につながるからである。

第二の理由は、旧聞に属するが例の情報化の波によるものであり、「組織と個人」の情報処理と言われるものが求められるようになって来たからである。このある意味で対立する二つの概念を組みとして扱う必要が出てきたということである。これは近年情報処理能力に対する需要が大きくなり、従来はホストマシンなどにより集中的に行ってきた情報処理の対象が飛躍的に広がったために、本来は効率よく処理できるはずの対象を処理できる状況に加工することが非効率的になったためといえる。すなわち、プログラム開発を集中的に効率よく行えなくなったためである。この結果、一部の情報処理は個人にまかされ、それらが効率良く操作可能にするためにワークステーションやLAN等の技術の発達が達成された。こういった処理の代表的なものはプログラム開発作業である。したがって、効率を落とさずに分散開発を行えるような支援環境を構築することが開発拠点の分散化に対応する目的となる。

この様な観点より、我々はメールを利用した分散開発の実験を行った [貫井90]。その結果によれば、情報の高度利用や共通化といった分散開発による利点がむしろ期待できることがわかり、D²システムはむしろこの点を狙って開発を進めている。以下、その理由を述べる。ソフトウェアの品質や生産性

の向上のためにはソフトウェアの再利用が進む必要があるが、現状ではほとんどシステムの極めて基本的な共通部分あるいは全体システムそのものと言ったように2極分化した形態での再利用のみが行われている。

この再利用をさらに進めるにはCASEツールなどの普及が必要だが、その普及状況は必ずしも満足のいく状態にはない。一方、ソフトウェア開発者の間で最近最も普及したツールと言えば電子メールである。ところが現状の電子メールにソフトウェア開発のための充分な支援があるのかと言えば、そうとは言えず現状の定番ツール群では物足りないであろう。したがって、CASEツールと単なる電子メールの間を埋める支援環境が、最も望まれているものであると考えられる[中村92]。本節では、以下こういった観点より現在研究開発中のD^{2}ソフトウェア分散開発のモデル及び支援システムを紹介する。

2.1.1 ソフトウェア分散開発モデル

本項では、ソフトウェア分散開発モデルについてその概要を述べる。まず、分散する主要なオブジェクトはつぎの四つであると考えている。それぞれについてどの様な形態をとるかを決定することが分散開発を実行するための問題となる。

(1) 開発拠点
(2) 開発作業
(3) 情報
(4) コミュニケーション

すなわち、(1)はじめにでも述べたように、まず拠点の分散が要求となるし、どの様に分散させるかが問題となる。(2)開発の開始から終了までの時間に沿って工程と作業を拠点毎に分散させるかが問題となる。(3)各工程で開発する成果物は何か、それはどこに保されるのか、その情報の共有化はどこまで許すのか最終的にどうまとめるのかなどが問題となる。(4)分散化された開発者達や組織、関係者がいつどこでどの様に情報を交換し、どの様な指示・報告がなされるべきかが問題となる。図2.1-1に制御システムの典型的な分散形態のモデルを示す。

次に、開発工程と情報の分散モデルについて図2.1-2にその例を示す。縦軸に工程をとり、横軸に分散された拠点を取る。交差した箇所にその工程での成果物とその共有範囲を記してある。なお、この図では、クライアントとの関連と保守工程については省略してある。また共有情報としてはこれらの成果物のほかに打ち合わせ記録やスケジュール、関連資料などが必要である。

このようなモデルに従って、各プロジェクト毎に分散開発形態を定めることが必要である。そのための手順として下記のものを考えている。

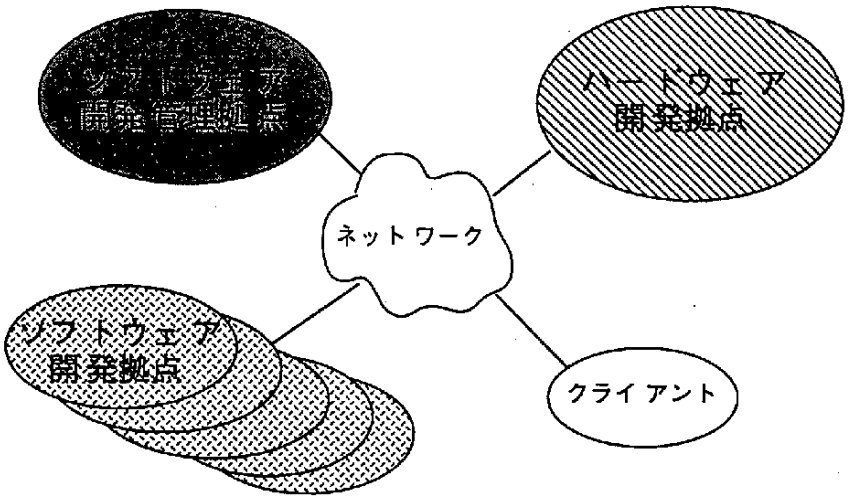


図2.1-1 開発拠点の分散モデル例

開発拠点 開発工程	ハードウェア 開発拠点	ソフトウェア 開発管理拠点	ソフトウェア 開発拠点
ｼｽﾃﾑ設計	ｼｽﾃﾑ設計書 ・H/W外部仕様書 ・S/W外部仕様書		
ｼｽﾃﾑ試験環境作成	ｼｽﾃﾑ試験仕様書		
ｿﾌﾄｳｪｱｼｽﾃﾑ設計		ｿﾌﾄｳｪｱｼｽﾃﾑ設計書 ・ﾌﾟﾛｸﾞﾗﾑ関連図 ・ﾌﾟﾛｸﾞﾗﾑ外部仕様書	
ｿﾌﾄｳｪｱｼｽﾃﾑ試験環境作成		ｿﾌﾄｳｪｱｼｽﾃﾑ試験仕様書	
ﾌﾟﾛｸﾞﾗﾑ設計		ﾓｼﾞｭｰﾙ関連図 ﾓｼﾞｭｰﾙ外部仕様書	ﾓｼﾞｭｰﾙ関連図 ﾓｼﾞｭｰﾙ外部仕様書
ﾌﾟﾛｸﾞﾗﾑ試験環境作成		ﾌﾟﾛｸﾞﾗﾑ試験仕様書	ﾌﾟﾛｸﾞﾗﾑ試験仕様書
ﾓｼﾞｭｰﾙ作成		ﾓｼﾞｭｰﾙ	ﾓｼﾞｭｰﾙ
ﾌﾟﾛｸﾞﾗﾑ試験		ﾌﾟﾛｸﾞﾗﾑ試験成績書	ﾌﾟﾛｸﾞﾗﾑ試験成績書
ｿﾌﾄｳｪｱｼｽﾃﾑ試験		ｿﾌﾄｳｪｱｼｽﾃﾑ試験成績書	
ﾊｰﾄﾞｳｪｱ設計	H/Wｼｽﾃﾑ設計書 ～ 試験仕様書		
ﾊｰﾄﾞｳｪｱ製造	ﾊｰﾄﾞｳｪｱ		
ﾊｰﾄﾞｳｪｱ試験	H/Wｼｽﾃﾑ 試験成績書		
ｼｽﾃﾑ試験	ｼｽﾃﾑ試験成績書		

図2.1-2 開発工程の分散モデル及びその成果物の共有モデル

(1) 開発拠点とその役割分担の設定

(2) 開発作業項目と手順の設定

(3) 各作業段階でのインタフェースの設定

(4) 拠点間でのフローの同期点の設定

(5) 情報の保管場所の設定

(6) データ／制御フローの記述とチェック

(7) 機密性のチェックとフローの修正

(8) 支援ツールの決定と利用方式の決定

この時、作業効率を落とす原因になるものとしては、拠点間のフロー自体の存在と経路の容量の不足によるのものやシステムの保守に拘るものなどがある。これらの影響をななるべく少なくするには、信頼性の高い大容量の回線を用いると共に伝達トポロジ [貫井90] のフロー解析を行い可能な限り独立な作業分担になるようにする必要がある。しかし現在、これらに関する方法論には研究中のものが多く定説はない。

2.1.2 ソフトウェア分散開発支援システムのモデル

前項では、分散開発のモデルについて述べた。そこで示したようにプロジェクト毎に分散開発のモデルは異なってくる。もちろん、メタ方法論としての分散開発のマニュアル的なものを考えることもできる。したがって、支援システムとしてはこれらの変化の影響が少ないモデルに基づいていることが望ましい。また、分散開発の実験の結果からグループウェアの支援系の満たす必要のある要件として、①支援サービスの連続性、②情報の利用容易性、を挙げている [貫井90]。これらを満たすモデルとして、図2.1-3に示すような階層型支援機能モデルを考えている。

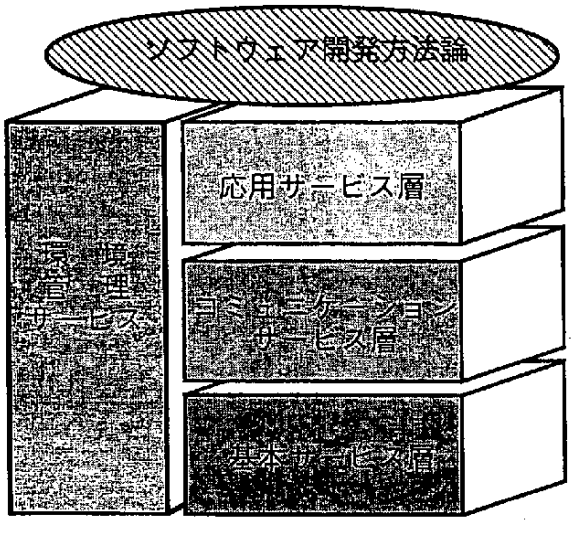


図2.1-3 階層型支援機能モデル

第一層（基本サービス層）は支援システムの基本となる個々の情報処理をサービスし、インフラ環境に相当する。PC／WS／サーバ／OS および付随するシステムコール、ネットワークといった要素が含まれる。

第二層（コミュニケーション・サービス層）は個人間の協調的な仕事を支援するプリミティブなサービスに相当するものである。電子メール、電子会議システム、電子掲示板、スケジュール管理、データベースといった要素が含まれる。

第三層（応用サービス層）は仕事の流れに沿ったサービスであり、第二層のサービスを組み合わせて構成される複合サービスに相当するものである。例えば、文書の共同作成などの単位業務はこの層の下位に位置し、ソフトウェア開発や EOA、OA などの実務はこの層の上位に位置する高度なものである。

環境管理サービスは全体の管理運用をサービスし、各層に対応する固有の機能と各層に共通の機能とが含まれる。例えば、PC・WS・ネットワークのアドレス管理は第一層に対応する固有の機能であり、データベース管理やネームサーバのディレクトリ管理は第二層に対応する固有の機能である。ユーザ管理やセキュリティ管理は各層に共通する機能である。

ソフトウェア開発方法論は、前節で述べたように分散形態に従った開発手順であり、各工程での成果物やデザインレビューなどの会議録などが含まれる。

このような階層モデルに準拠することによって、プロジェクト毎に異なった分散形態や開発方法論とそれに伴うインフラの相違などの影響の少ない支援システムを構築することが可能となる。

2.1.3 エージェント指向電子メール

ソフトウェア分散開発支援システムの第二層の例として、電子メールを中心としたコミュニケーションシステムについて述べる。コミュニケーションシステムには同期型（リアルタイム型）と非同期型とがあるが、ここではこれらを融合したシステムとしてとらえている。電子メールにおけるエージェントとは、業務に電子メールを用いる場合に発生する様々な手作業を支援する機能として考えられている。例えば、Andrew Message System や Object Lens などに典型的なものが見られる。そこでのエージェントはどちらかといえばオブジェクト的なものでメールにより起動されるサービス機能である[松尾91]。分散協調的な問題解決を実現しようとする試みも数多くあるが、進んでいるのはスケジューリングや負荷分散についてのものが多い。D^2 システムでは、ソフトウェア分散開発に必要な支援として汎用の単機能なエージェントとそれを組み合わせるエージェント記述スクリプトを用意している。汎用の単機能エージェントとしては、図2.1-4に示すように

・ 分類（フィルタリング）：サブジェクト、発信相手、日付等によりメッセージを分類・保管する機能。

・　配送（ルーティング）：自動的にメッセージを関係者に回覧、同報、発信し配送する機能。
・　起動（エグゼキューティング）：メッセージの受信・発信等のイベントにより指定したアプリケーションを実行する機能。

が用意されている。

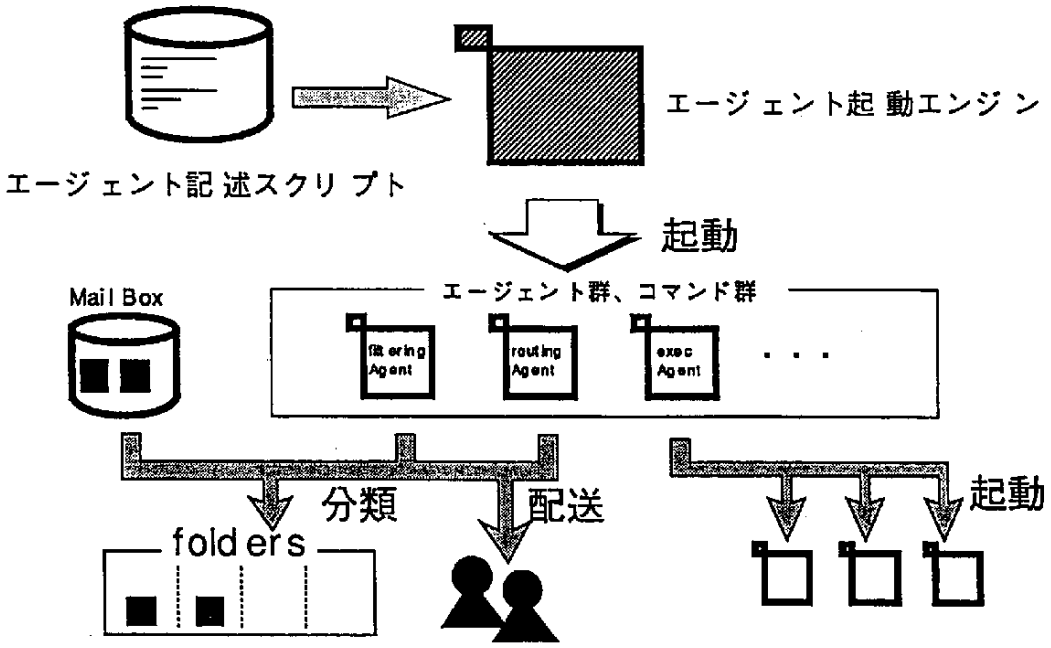


図2.1-4　エージェント指向電子メールシステム

これらのエージェントとその使用方法を記述したエージェントスクリプトを用いることにより、例えば図2.1-5に示すようなソフトウェアの成果物収集支援システムを容易に構築できる。図2.1-5において、プログラムは分散した各拠点にいる各担当により作成され、その後リーダのところに集められmakeすることになっている。このとき、ファイル転送などの方法を用いると転送先のディレクトリ構成を把握し意識して転送を行わねばならない。また、リーダが進捗を把握するのもそれなりの手間がかかる。

しかし、エージェント機能があればリーダが分類エージェントを定義しておくことにより、担当はメールにてプログラムをリーダに送れば良いことになる。また、進捗を見るには起動エージェントを用いて予定のディレクトリのプログラムの有無をチェックする。予定の期日に登録されていなければ配送エージェントを用いてフォローのメールを自動発信する。リーダ自身にフォローメールを発信することで定期的なフォローもある程度自動化可能である。

さらに、プロジェクト毎にスケジューリングエージェントを含んだ開発支援エージェントライブラリを関係者に送付することにより、開発に必要な支援機能と管理体制を確立することができる。この

場合、様々な設計変更やトラブルに対応した開発プロセスやスケジュールの変更が、各メンバのエージェント環境に一様に行われることが必要である。

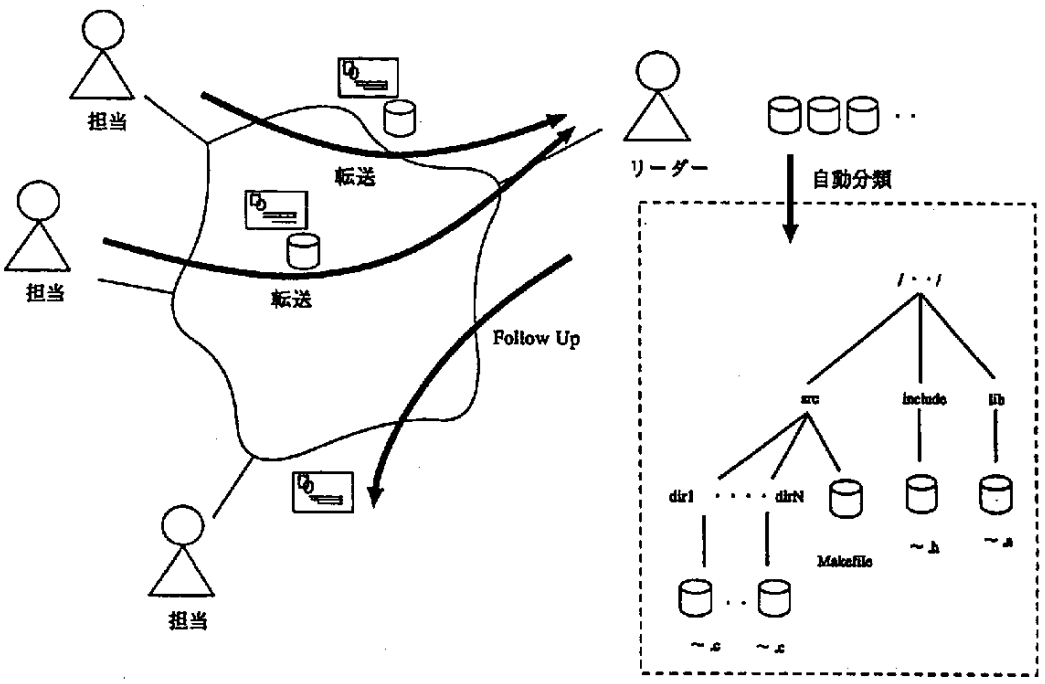


図2.1-5 エージェントによるソフトウェア成果物収集支援システム

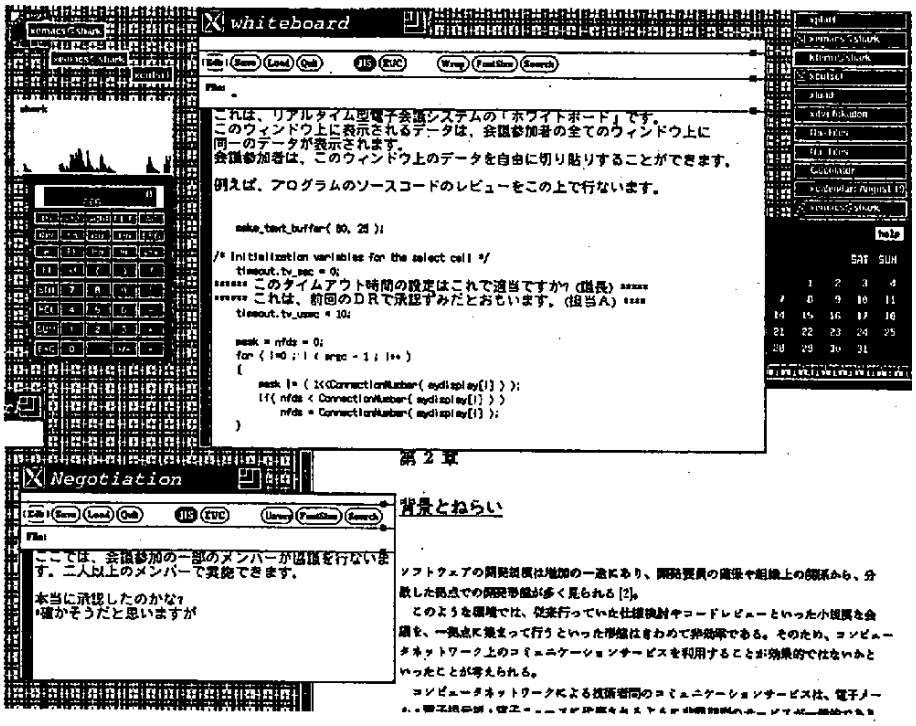


図2.1-6 同期型コミュニケーションシステムの表示画面例

2.1.4 リアルタイム電子会議システム

図2.1-6は同期型コミュニケーションを行うためのリアルタイム会議システムを組み合わせた場合の表示画面の例である [太田92]。参加者全員に共通に見えるホワイトボードと特定のメンバのみに共通するネゴシエーションボードとからなる。ソフトウェア分散開発の面においては、コードレビューなど直接対象となるソフトウェアの実体をチェックしながら会議を進めるような場合に特に有効であると考えられる。レビューの結果はメールで送られ分類エージェントにより格納保管される。なお、会議システムの各ボード自体もエージェントである。今のところ、エージェントと呼ぶにふさわしい分散協調メカニズムはインプリメントしていないが、実用的なものとしては会議予約システムなどを考えている。

2.1.5 ユーザ認証システム

階層型支援機能モデルの環境管理サービスのうち各層共通のものの例として、ユーザ認証システムについて述べる。ユーザ認証とは、ユーザがユーザ本人であることを確認することであり、通常のワークステーションなどではパスワードの入力で自分のユーザを認証している。ワークステーションが一台のときはこの方法でも比較的問題はないが、ソフトウェア分散開発の場合のように、ネットワークを用いて複数台のワークステーションが公衆回線などを用いて結合相互利用できる環境においては、単なるパスワード確認では不十分である。一つのワークステーションに入れたらすべての資源が許可されるのも問題であるし、関連するすべてのワークステーションに各々異なったパスワードを登録しておくことも非現実的であるし、公衆回線などの上をパスワードの文字列データがそのまま流れる方式にも機密管理上の問題が生じる。これらの問題を解決し、分散システムで中心的な役割を果たす各サーバ毎にユーザ認証を行う方式のシステム：D^2 Authentication System について述べる [佐波91]。本システムは分散環境でユーザ認証を行うため以下の手段を用いている。

(1) 認証サーバによる認証

(2) 相互認証による相手の確認

(3) 暗号化によるプライバシの保護

(1)の認証サーバによる認証は、従来の各ノード毎のパスワード（以下鍵という）管理を一括して認証サーバが管理する方式であり、これにより鍵の分散を防ぎ盗用を防ぐ。同時にシステムの大規模化の際の鍵管理も容易になる。(2)の相互認証は、サーバ側から正しいクライアントであると認証すると共にクライアント側からも正しいサーバであるということを確認する方式であり、名を騙り相手より情報を盗用することを防ぐ。(3)の暗号化は言わずもがなではあるが、ネットワーク上で直接鍵の情報が流れ、容易に盗用されるのを防ぐ。

このような方式を用いてユーザ認証を行うシステムの例として MIT の Kerberos があるが、本システ

ムはつぎの２点に特徴がある。

(1) 認証サーバに初期認証用のものとサービス認証用の２種類を設け負荷の分散を図っている。

(2) 暗号化方式に慣用暗号方式のかわりに公開鍵暗号化方式を適用し、相互認証時における鍵管理の問題と鍵長による安全性の改善を図っている。

図2.1-7に本システムの認証モデルを示す。

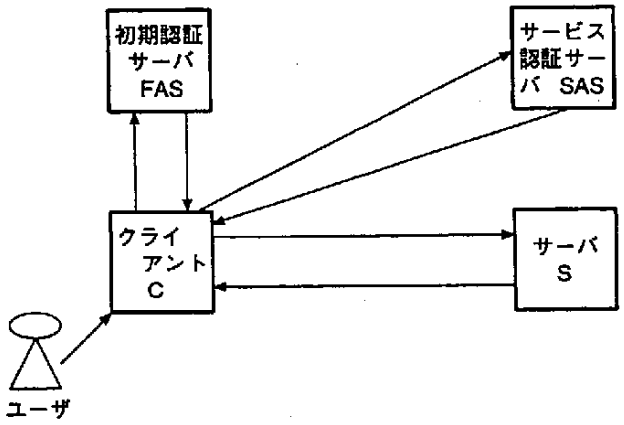


図2.1-7 D^2 Authentication systemの認証モデル

初期認証手順を図2.1-8に示すが、それは以下の手順で行われる。

(1) ユーザがユーザ名をＣに入力する。

(2) Ｃはユーザ名、ホスト名、ドメイン名をデジタル署名をつけて（F）ASに送り、初期チケットとユーザ情報を要求する。

(3) （F）ASはユーザ名を確認し、初期チケットとユーザ情報をデジタル署名をつけてＣに返送する。

(4) Ｃはユーザにパスワードなどを要求しユーザ情報と照合する。

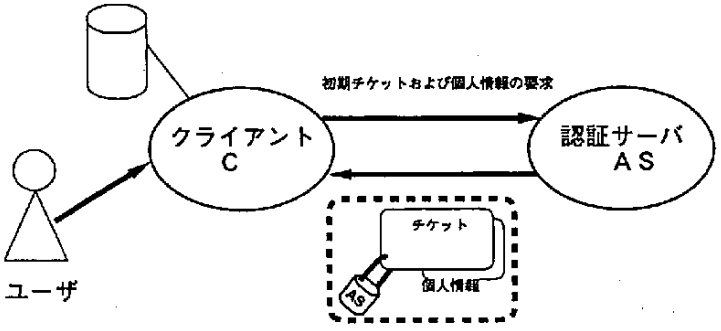


図2.1-8 初期認証手順

一方、サービス時の認証は以下の手順で行われる（図2.1-9参照）。

(1) Cは初期チケット、サーバのホスト名、ドメイン名、サービス名をデジタル署名して（S）ASに送る。

(2) （S）ASはサーバ用のサービスチケットをデジタル署名してCに返送する。

(3) Cはサービスを要求するためにサービスチケットをデジタル署名してSに送る。

(4) Sはサービスチケットを確認しサービスの承認をデジタル署名してCに送る。

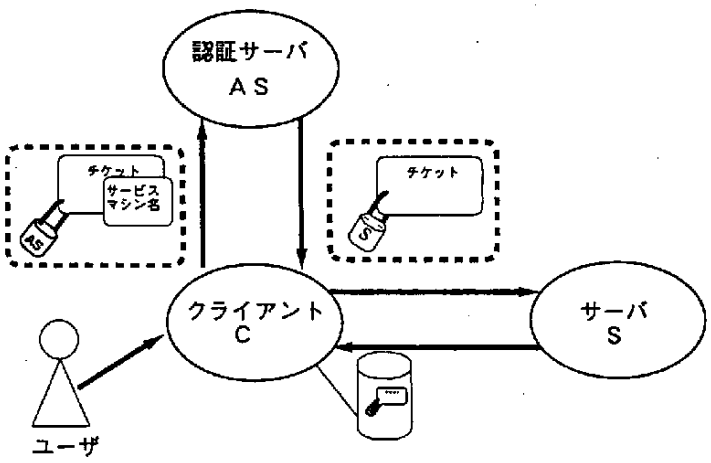


図2.1-9 サービス時の認証手順

このような初期認証手順及びサービス時認証手順により、システムのサービスを正しいユーザのみに与えるようにしている。

最後に本節全体をまとめる。本節ではD^2ソフトウェア分散開発支援システムについて、

(1) 分散開発の形態のモデルとその考え方

(2) 分散開発支援システムの階層的支援機能モデルとその考え方

(3) エージェント指向電子メールとその応用例

(4) リアルタイム電子会議システム

(5) 分散開発の基本的セキュリティ管理としてのユーザ認証システム

に関する概要を述べた。

D^2システムはソフトウェア分散開発のための支援システムとしては基本的な部分が多いが、従来のプログラミングツール、CASEツール、SEDBなどを統合し、人間の（知的）活動とその作業を支援する最も有効な枠組みではないかと考えている。すなわち、ソフトウェ開発のプロセスを規定する性格の支援システムであり、プロセスプログラミングに通じる一つの典型的な例がここで紹介したコミュニケーションシステムであると言えよう。

【参考文献】

[中村92] 中村英夫,貫井春美：ソフトウェア分散開発支援システムD^2,ソフトウェア・ツール・シンポジウム’９２予稿集(1992).

[貫井90] 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,ソフトウェア工学研究会71-8,pp.57-64(1990).

[松尾91] 松尾　朗,貫井春美,中村英夫：電子メールにおけるエージェントシステム,情報処理学会第43回全国大会予稿集2J-13(1991).

[佐波91] 佐波公夫,貫井春美,中村英夫：分散環境におけるユーザ認証システム,情報処理学会第43回全国大会予稿集2J-5(1991).

[太田92] 太田哲生,貫井春美,中村英夫：ソフトウェア分散開発支援向けリアルタイム電子会議システムの開発,情報処理学会第44回全国大会予稿集3M-4(1992).

2.2　MERMAIDによる分散開発支援

2.2.1　はじめに

高度情報化社会を迎え、企業や団体などの組織における業務が複雑・高度となり、グループで行う業務の重要性が増大している。そのため、CSCW (Computer-Supported Cooperative Work) と呼ばれるグループの協調活動のモデル化や支援方式、グループ協同作業支援システムの社会への影響などを研究する新しい学際的な研究分野が開拓され [Greif88,ACM90]、情報処理技術や通信技術を利用して人間の協同作業を支援するシステムであるグループウェア (Groupware) が注目されている [Ellis91]。また、近年のボーダーレスエコノミーやグローバリゼーションに対応して、例えば、互いに離れている本社や営業所、工場の人達を結んで生産計画を調整するといった人と人とのコミュニケーションの重要性が増している。

本節ではグループ協同作業支援システム構築のための基盤として研究開発され、実用化されているマルチメディア分散在席会議システム MERMAID (Multimedia Environment for Remote Multiple Attendee Interactive Decision-making) のソフトウェア分散協同開発等への利用経験について述べる。

2.2.2　MERMAIDシステムの概要

(1)　概要

MERMAIDシステムは、WSを使って、遠隔地にいる人達がテキスト（文字）、図形、イメージ、手書き、音声、動画を含むマルチメディア情報を広域網を介してリアルタイムで交換しながら、効率的に会議をはじめとした協同作業を行うことを支援する在席会議システムである [Watabe90,Watabe91,渡部91]。このシステムはグループ通信アーキテクチャ (GCA) [阪田89] を基礎として新規に構築したもの

であり、現在まで約1年半にわたり分散環境での実用実験を行っている。さらに、本システムを基盤として、情報処理システムによりグループを支援できる可能性の大きい部分である文書協同作成、ソフトウェア協同作成、グループ意思決定を含むグループ協同作業を支援することを目標としている。

(2) 利用者用装置の構成

各利用者のWSは図2.2-1のような構成となる。WS は NEC EWS4800（モデル220または260）を利用している。ディスプレイ（解像度:1280×1024）にはマルチメディア文書や動画が表示される。マウスは利用者がメニューを選択したり、画面に表示された文書を指す（ポインティング）ために利用する。タブレットは手書き文字・図形を画面の文書中に入力する場合に利用する。イメージスキャナは文書を読み込んで他の参加者に送るために利用する。ビデオカメラは参加者の様子を撮るために使われる。ワイヤレスマイクとワイヤレスヘッドホンは音声のやりとりのために利用する。他に動画を通信する際の動画情報の圧縮・伸張のための動画コーデックが必要である。なお、キーボードはテキスト（文字）の入力、編集のため以外にはほとんど使う必要がない。

図2.2-1 MERMAID利用の様子

(3) ソフトウェア構成

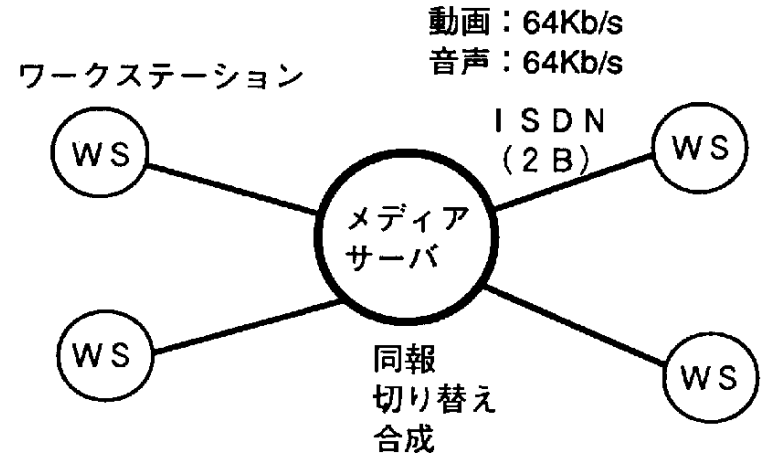
OS は UNIX で、C 言語を用いて構築した。サーバ・クライアントモデルに基づいてソフトウェアを作成した。ユーザインタフェースのためのウィンドウシステムとして X-Window、X-Window 上のユーザインタフェース構築ツールとして「鼎」[暦本90] を利用している。これらのウィンドウシステムを用いることにより、MERMAID システムはより視覚的に操作できるようになっている。その結果、利用者がこのシステムに親近感を持ち、操作も容易になった。

(4) MERMAID システムの通信環境

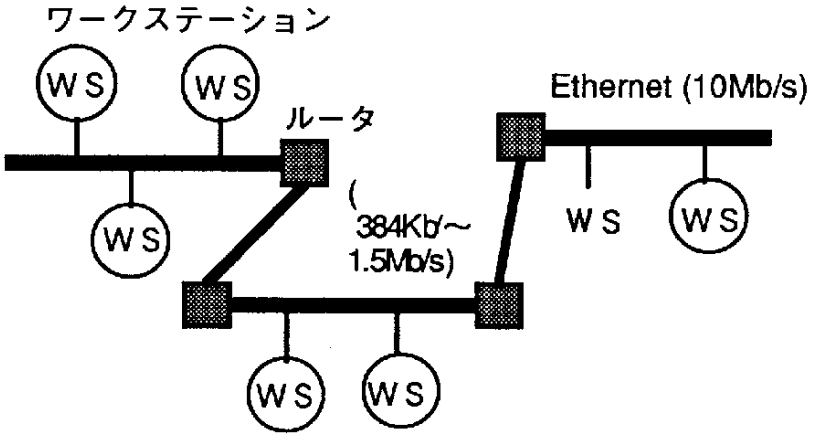
MERMAID システムでは動画、音声、データ（制御情報及び文書情報）の交換のために現在はISDN (2B+D) とブリッジで広域に接続された Ethernet を使用している。

(a) 動画、音声の通信

図2.2-2(A) に MERMAID システムにおける動画、音声の通信環境を示す。動画と音声の通信には ISDN のBチャネル (64Kb/s) を利用している。２本の B チャネルのうち、１本を動画通信、他の１本を音声通信に利用している。動画はビデオカメラから入力され、ビデオコーデックにより64Kb/s に圧縮され、通信路に送られる。音声はやはり64Kb/s に符号化され、通信路に入る。会議に参加しているクライアント（利用者の WS）からの動画はメディアサーバに一度集められて選択、合成され、全クライアントに同報される。全員の音声はメディアサーバでミックスされ、全クライアントに同報される。



（A）動画・音声通信



（B）データ通信

図2.2-2 MERMAIDの通信環境

(b) データの通信

図2.2-2(B) に MERMAID システムにおけるデータの通信環境を示す。制御情報や文書情報の通信にはブリッジで広域接続された Ethernet (Branch4680) を利用している。速度はローカル（LAN 内）で 10Mb/s、リモート（LAN 間）で384Kb/s～1.5Mb/s である。通信プロトコルは低位は TCP/IP を採用している。

(5) MERMAID システムの特徴

遠隔地間での協同作業を効果的に支援するには、(i)音声による情報交換を効果的に補足し得るメディアのサポート、(ii)リアルタイムに作業結果が相手に反映されること、(iii)計算機による作業の管理・支援、が必要である。そこで、MERMAID システムはこれらを含む以下のような特徴を持っている。

(a) 広域で多者間の在席会議を支援

広域に分散した人達がグループ協同作業を行えるように、WS を使って広域でしかも多者間の在席会議をサポートしている。ローカル通信と広域通信をつなぐ通信サーバにより、多地点間のリアルタイム情報通信が効率よく行える。

(b) 動画と音声を活用した臨場感の創出

円滑な会議の進行のためには臨場感が重要である。共有画面の操作権保持者が会議参加者のうち任意の 4 者までを動画で同時に映すように切り替えることができる。動画により会議の臨場感と参加者同志の親近感が増大する。音声は全員の音声がミックスされるため、共有画面の操作権とは関係なくいつでも誰でも話すことができる。

(c) マルチメディア文書のリアルタイム編集、交換

効果的なプレゼンテーションや議論、文書協同作成などのため、文字、図形、イメージ、手書きを含むマルチメディア文書を参加者が協同してリアルタイムで作成、編集、配布できる。

(d) 柔軟な共有画面操作権の移行制御

会議の性質や参加人数に応じて会議の進め方を変えたいことがある。そこで MERMAID システムでは共有画面の操作権移行のために 4 種類のモード（議長指名、要求順、バトン、フリー）を提供しており、よりフレキシブルに会議を進められる。また、会議中に個人作業を行えるよう画面のレイアウトを個人個人で変えられるようにしている。

(e) 親しみ易いヒューマンインタフェース

在席会議という形態から、誰でも簡単に操作できなければならない。そこで、マルチウィンドウ、メニュー表示、マウスを使ったメニュー選択を基本とし、理解し易く簡便なヒューマンインタフェースを提供している。

2.2.3　実験構成と利用評価

次のように利用実験を行い、評価した。

・　利用実験期間

1990年4月より1991年10月

・　接続地点

日本電気（株）の本社（東京）、中央研究所（川崎）、関西研究所（大阪）、筑波研究所（筑波）、神奈川サイエンスパーク（川崎）、北米研究所（アメリカ・ニュージャージー州プリンストン）の6箇所（図2.2-3）、計約２０台のWSを設置、会議の９０％は２地点間の接続、１０％は３地点接続、４地点以上はほとんどない。

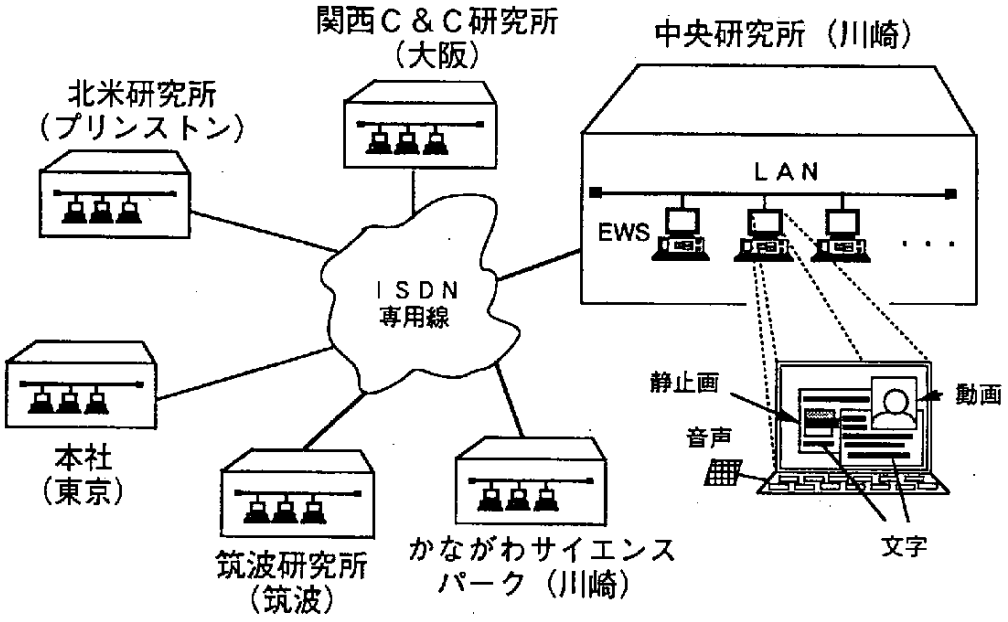


図2.2-3　現在のMERMAIDの接続地点

・　会議内容

業務の連絡、ソフトウェア仕様の検討、ソフトウェア開発の進捗報告、技術課題の討論などの技術的なものが多い。テレビ電話的な使い方による簡単な打ち合わせから、マルチメディア文書の交換を含む本格的な会議までである。

・　利用頻度

週に２～５回、平均２回、１回につき１時間～４時間、平均1.5時間。特に中央研究所と関西研究所では協同研究のために最低でも週２～３回、打ち合わせを行った。

・　利用人数

２～１５人、平均４人

・　利用ＷＳ数

２～６台、平均３台（１地点で２台利用することも多い）

現在、分散在席会議の特性を把握し、支援機能を向上させることを目的として評価を進めている。まだ途中ではあるが、現在までの主な主観評価を示す。

(1)　個人の自由の尊重について

(a)　ウィンドウのフリーレイアウトについて

個人ごとにウィンドウのレイアウトを変えられることは非常に便利である。会議参加者はみな自分の好みのレイアウトをとっている。個人作業用のスペースを空けたりするにはなくてはならないものである。

(b)　文書フリーアクセス、個人資料の処理について

共有文書の表示されていない部分をスクロールして個人的に見ることも多い。配布された文書のページ数を確認したり、ページをめくって図だけを見て資料の雰囲気をつかむことは普通の会議でもよくやることであるが、本システムを利用しているときも同じことをする人が多い。ただし、操作権を持っている人が黒板上の文書をスクロールすると、同じ文書の別の部分を見ていた人の黒板内容も操作権を持っている人に強制的に合わされてしまうため、不快に感じることがある。これは操作権保持者が説明する所を全員が注目できるようにしたためである（会議として成り立つために、個人の無制限の自由は認めていない）。会議資料と個人資料は同時に扱えるので、個人的に邪魔されずに文書を読みたいときは「ノート」に文書を表示すれば良い。

(c)　個人作業の同時実行について

会議中にメモをとったり、電子メールや電子掲示板を見たりということはよくある。これも時間を有効に利用するためには役に立つ機能と言える。

(2)　会議を越える会議について

(a)　サブグルーピング、他の人の呼び出しについて

会議中に他の人と連絡をとることはあるが、主に電話で済ませ、本システムにより連絡することはほとんどない。理由は、(i)少し意見を聞くだけならば電話の方が早くて手軽である、(ii)相手がWSを持っていない場合も多く、本システムが利用できない、などである。内緒話については相手と声で会話したいことがほとんどであり、電話が各人に２回線必要となる。ただし、あまり進んでやることはない。個人的に質問したり、裏でかけひきが必要な時には役立つ機能であろう。

(b)　会議中の資料検索について

会議中の資料検索は、現在のところ、過去の会議の議事録や討議資料を検索することが多い。大きなデータベースやシミュレーションのできるシステムと連結されていればかなり行われるものと思わ

れる。

(c) 文書編集について

黒板、ノートに表示した文書は自由にエディットできる。これは必ずと言って良いほど利用される。

(3) グループ作業と個人作業の連携について

個人の机の上で個人作業だけでなく、グループ作業までできるようになったことは非常に便利である（もちろん、現在のところ、WS だけでは仕事を遂行するために十分な機能を持ってはいない）。会議中に黒板とノートの間で文書の転記はよく行う。黒板の文書を個人的に保存しておきたいときは、操作権をもらって自分のファイルとして保存する。

(4) 多様な会議への対応について

(a) 共有画面操作権の移行モードについて

会議の性質に応じて異なった操作権移行モードが選択される傾向がある。司会者がおり会議構成員に職位階層の差があるときは指名モード、それ以外では要求順かバトンモードが多い。特に司会者を設けない時やブレーンストーミングのように全員が同時に書き込める必要がある時はフリーモードが利用される。

(b) 操作権について

操作権は会議で１つあればよいか、黒板ごとに１つ必要かは議論の分れるところである。後者は利用者インタフェースが煩わしいと判断し、現在は会議ごとに１つの操作権を設けている。

操作権の移行はどのモード（議長指名、要求順、バトン）でも煩わしさが伴う。操作権のやり取りをなくす（フリーモードを使う）ことが利用者にとって最も良い。ただし、全員のWSに届くメッセージの順序を何らかの方式（例えば、サーバによる順序制御）で同じになるよう制御する必要がある。

(c) 定例会議と突発的会議の割合

現在のところ、３対１程度で定例会議に利用することが多い。急に会議を行いたくなった場合でも、現在のシステムは音声や動画をつなぐためのダイアリングや機器のセットが多少面倒なため、電話で済ませることもある。文書を表示しながらの議論には本システムがかなり有効であるため、利用される。

(5) 分散会議の支援について

(a) 音声について

会議では最も利用しやすいメディアである。ただし、オフィスの自分の席で会議を行う場合は、回りの人達に迷惑がかからないように、スピーカフォンではなく、ワイヤレスマイクとワイヤレスヘッドホンを利用した方がよい。ただし、初めは会話が原因不明のまま途切れることがあった（すぐにワイヤレスマイクの電池切れと判明した）。

会議参加者が互いに知っている場合は問題ないが、初対面の人が多い会議では顔と声の対応関係を

把握し難いことがある。これは動画の4画面同時表示により補うべきである。また、参加人数が多い（概ね7人以上）場合は誰が発言しているか判別困難なことがあり、発言前に自分の名前を言うことが求められることもある。

(b) 動画について

動画は、資料や物を提示することにより説明を補助できる以外に、互いの顔が見えることにより、参加者に安心感を与え、参加者間の親近感を増大させるという効果があることがわかった。

現在、動画の伝送速度は64Kb/sであり、画面は1秒間に2～5枚の書き換えしかできず、音声に対して0.2～0.5秒の遅れが出る。また、コマ落としのようになるため、初めは違和感を持つ参加者が多い。会議で単に参加者の様子を送信するには平均して0.3秒以下の遅延になり、64Kb/sでもほぼ十分と言えるが、遅延が少なく、より滑らかな動きで、画質も良いものを求める利用者には、最低384Kb/s、できれば1.5Mb/s以上の伝送速度が欲しいところである。

現在はビデオプロセッサの解像度(360×288)が不十分なため、資料を動画で表示すると細かい文字は読み難くなる。縦横それぞれ今の2倍程度の解像度が欲しいところである。

(c) 利用人数とデータ伝送遅延について

現在までのところ、1つの会議ではWS4台以下で利用することが多い。WSの台数が多少増えてもデータ伝送遅延による不都合はさほど感じない。ただし、WS間で、文書の表示、スクロールのタイミングが2～3秒ずれることがある。

(d) 手書き（テレライティング）とキー入力について

タブレットを使った手書き入力は文書の補足説明によく利用され、効果も大きい。なかには議事録の作成に利用する人もいる。手書き文字・図形はベクトルデータとして保持・表示しているが、データ量が多くなりがちなため、黒板1枚の表示、再表示に数秒を要する。会議中にはキーボードはほとんど使われないが、人によっては文字入力／修正に利用することもある。

(e) テレポインティング

黒板の文書の一部を指しながらの説明や質問がしやすくなった。操作権に関係なく全員が常に利用できるにもかかわらず、操作権保有者以外には今のところあまり利用されていない。4～5人が同時にポインタを動かすと交換すべき情報量が一時的に増大し、ポインタの画面への表示がWSによっては数秒間遅れることがある。

(6) その他

(a) イメージ入力について

現在、イメージスキャナはパソコン用の低価格のものを利用している。しかし、読み取り速度が遅いため、A4版1枚をモノクロで読み込むのに10秒程度かかっている。カラーはさらに時間がかかる。高速読み取りのできる機種を利用する必要がある。

別の問題として、文書をイメージとして読み込む場合、スキャナの台上での位置決めと読み取り線密度を決めることが困難なことがよくある。何度か読み込んでやり直しをすることもあり、これが使い勝手を悪くする原因ともなっている。一度早くラフにスキャンして位置決めを行ってから、本格的に読み取るという必要がある。

(b) 大画面ディスプレイについて

初めは1台のWSを1人が利用するという環境を想定して設計した。しかし、実際にはWSは十分な台数なく、WSをオフィスのコーナーに置き、1台の前に数人が座って会議を行うことも多い。この場合は、WSの画面を拡大して投影する装置があると利用しやすくなる。実際、40インチのディスプレイを利用してTV会議のように会議を行ったことがある。出力は皆に見えるが、マウスやタブレットは交代で利用しなければならず、操作が困難であるという問題が生じた。入力装置を全員が持つようにするにはハードウェアの改造も必要となる。

(c) 透明デジタイザについて

手書き文字を入力するにはタブレットを利用するが、タブレットは画面の横に置き、画面と離れているため、描く所と表示される所が別々となり、利用しにくい。そこで、透明デジタイザのプラスチックフィルムを画面に張り、ペンで直接画面に描けるよう装置を自作した。しかし、ペンで描く際にキュッという音が出てガラスに傷をつけているようであまり気持ちが良くなかったり、視差（描く所と表示される所が少しずれる）が出たりで利用者には芳しい評価は得られなかった。将来的にはカラーの液晶ディスプレイに表面を加工した透明デジタイザフィルムを貼ることにより対応していきたい。

2.2.4 MERMAIDシステムの利用例

本節では、代表的な利用例を紹介する。現在、中央研究所（川崎）と筑波研究所（筑波）の研究員が協力して、筑波研究所内のスーパーコンピュータ(SX-2)を使って分子構造の解析システムを構築している（図2.2-4）。彼らはMERMAIDシステムの共有画面にソフトウェアの仕様案やモジュール分割案、分担案、進捗状況などを表示して検討したり、動画ウィンドウにプログラム実行結果を表示してチェックしている。スーパーコンピュータのカラーディスプレイは筑波研究所にしかないため、プログラムの実行結果（分子構造のグラフィックス）を見るために本来は中央研究所の研究員も筑波研究所に行かなければならないが、スーパーコンピュータの表示をMERMAIDシステムにより動画として中央研究所に送り、それをWSの画面に表示することにより、遠隔地でも結果を見ることができる。

このようにMERMAIDシステムを分散ソフトウェア共同開発に利用することにより、出張がほとんど必要なくなり、簡単な打ち合わせや質問のためでも手軽に相手を呼び出して画面を通して相談ができるようになったため、互いに離れていても短時間で質の高いソフトウェアが開発できるようになった。

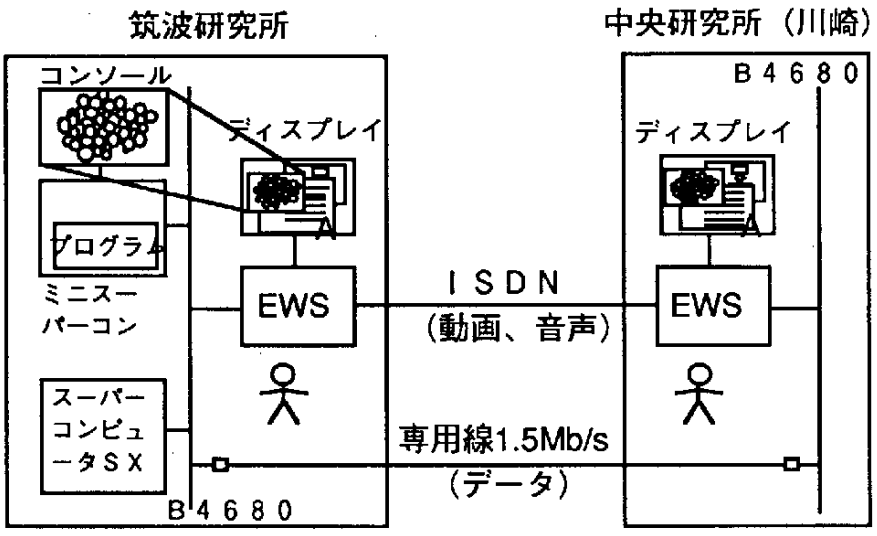


図2.2-4　MERMAIDシステムの分散ソフトウェア開発への利用例

2.2.5　MERMAID システムの応用分野

本システムはグループのコミュニケーションと情報処理を支援するものであり、より高度なグループ協同作業を支援する基礎となると考えている。本システムを応用して、次のようなグループ協同作業支援システムを構築したいと考えている。

(a)　分散ソフトウェア協同開発支援

ソフトウェア開発拠点が地方に分散しつつあるが、MERMAID システムを使って本社と地方拠点、ユーザを結んで共同でソフトウェアの仕様を検討したり、プログラミングやデバッグを行うことを支援する。

(b)　グループ意思決定支援

討議や投票をリアルタイムで行うだけでなく、根回しやグループで共同の案の選択を支援する。

(c)　遠隔教育支援

互いに離れている教師と複数の生徒間で、出題、回答、採点、質疑応答、講評をリアルタイムで行うことを支援する。

(d)　マルチメディア文書共同編集支援

１つの文書に複数の人が同時にコメントを付けたり、それについて話し合ったりしながら、その文書をリアルタイムで編集し、配布することを支援する。

(e)　在宅勤務、サテライトオフィス勤務支援

通勤時間が長くなる傾向や女性の社会進出への対応として、在宅勤務やサテライトオフィスが注目

され、実験されており [石割90]、これらの勤務形態を取る人が今後増加すると思われる。これらの人達と本社などのオフィスを結び、臨場感を持った打ち合わせや報告を行うことを支援する。

(f) 情報検索システムとの連動

データベース、ハイパーメディアシステムなどと連動し、会議中に外部の情報を検索したり、株価や為替の動きを見ながら相談ができるようにする。

(g) 他のアプリケーションと連動した業務支援

ワードプロセッサやスプレッドシート、フォーム処理システムなどのデータを MERMAID システムの共有画面に貼り付けて、報告、討議が行えるようにする。

2.2.6 おわりに

マルチメディア分散在席会議システム MERMAID の研究背景、概要、通信環境、特徴、評価、利用例、応用分野を述べた。MERMAID システムはグループの多様な協同作業の支援システムとして実用に対して満足する機能、性能が得られている。グローバリゼーションの傾向とは逆に、情報の偏在から国内では企業などの組織の東京への一極集中が進んでいると言われる。MERMAID のようなマルチメディアの電子コミュニケーション支援システムは、情報交換を促進するため、東京と地方との情報格差を縮小し遠隔地の人との協同作業を促進させ、一極集中を緩和するための有力なツールとなり得ると考えている。

【参考文献】

[ACM90] ACM Proc. Computer-Supported Cooperative Work, Los Angeles, Calif. (1990).

[Ellis91] Ellis, C., et al. : Groupware, Some Issues and Experiences, Commun. of ACM, Vol.34, No.1 (1991).

[Greif88] Greif, I. (編): Computer-Supported Cooperative Work: A Book of Readings, Morgan Kaufmann (1988).

[石割90] 石割寿郎: 志木サテライトオフィス実験に関する一 考察, 電気・情報関連学会連合大会講演論文集, 5-120--5-125 (1990).

[暦本90] 暦本純一ほか : エディタを部品としたユーザインタフェース構築基盤 : 鼎, 情報処理, Vol.31, No.5, pp.602-611 (1990).

[阪田89] 阪田史郎ほか : グループ通信アーキテクチャ ーマルチメディア分散会議システム構築のための基本概念ー, 電子情報通信学会オフィスシステム技報, OS89-4 (1989).

[Watabe90] Watabe, K., et al. : Distributed Multiparty Desktop Conferencing System: MERMAID, Proc. Conf. on Computer-Supported Cooperative Work, pp.27-38 (1990).

[Watabe91] Watabe, K., et al. : Distributed Desktop Conferencing System with Multiuser Multimedia Interface,

IEEE J. Selected Areas in Commun., Vol.9, No.4, pp.531-539 (1991).

[渡部91] 渡部和雄ほか: マルチメディア分散在席会議システムMERMAID, 情報処理学会論文誌, Vol.32, No.9 (1991).

2.3 KyotoDB による分散開発支援

2.3.1 はじめに

本節では、ソフトウェア・エンジニアリング・データベース KyotoDB と、KyotoDB における分散開発を支援するための協調活動支援環境 KAE について述べる。

KyotoDB [Matsumoto90,鰺坂92] は、オブジェクト指向パラダイムに基いたソフトウェア・エンジニアリング・データベース(KyotoDB カーネル) およびその開発支援環境である。KyotoDB カーネルは、ソフトウェア生産物とその開発プロセスを統合的に管理する。さらに、ソフトウェア生産物相互の意味的関係の管理と、ソフトウェア生産物のデータの変更に伴って発生する他のソフトウェア生産物への波及効果を解析する機能を持つ。

KAE は一種のグループウェアであり、人間の協調活動の過程をモデル化し、シナリオとして記述することにより、複数人にわたるプロセスプログラムの実行を支援する。

グループによるソフトウェア開発では、対象となるソフトウェアが大規模なものになるほど、あるいはグループを構成する開発作業者(以下「作業者」と表記)の人数が多くなるほど、作業者相互のコミュニケーションの必要性が急激に増大する。

現実に、作業者の時間配分のうちの半分以上が他の作業者への質問や情報提供であるという調査例も報告されている。このことからも明らかなように、グループ内部のコミュニケーションは開発作業において非常に重要な部分を占めている[垂水92]。

グループを構成する作業者が遠隔地に分散している場合には、通常の対面会議やミーティングの開催が困難になり、適切なコミュニケーションの手段なしでは円滑な作業が行えない。また、作業者は常にコミュニケーションが可能な状態にあるわけではない(例えば、コミュニケーションの必要な相手がミーティング中や勤務時間外であるなど)。このような場合にコミュニケーションを成立させるには、伝達される情報の蓄積などの手段(例えば、メモを机の上に置いておくなど)を用いる必要がある。前者をグループ・コミュニケーションにおける空間的障壁、後者を時間的障壁と呼ぶ。

また、ソフトウェア開発グループ(チーム)のリーダーや、小人数での高度なソフトウェア開発における各作業者は、複数の問題を並行して処理することを要求されることが多い。このような場合、どの問題に対して誰とどのように情報交換すればよいのかを適切に判断することが重要である。しかし、同時に扱っている問題が多くなってくると、問題の把握と判断、およびその解決のためのコミュニケーションに割かれる労力が無視できないものになり、迅速な問題解決が困難になってくる。例えば、

従来の電子メールを利用したソフトウェア開発を考えてみる。電子メールにより時間的障壁はある程度解消するものの、大量の電子メールの整理、内容の把握および処理に割かれる労力は小さくない。このような問題をグループワークにおけるコミュニケーション・オーバーヘッドと呼ぶ。

一般に、以上のようなグループ・コミュニケーションとグループワーク(協調活動)における諸問題の解消を目的とした、計算機ベースの支援システムをグループウェアと呼ぶ。グループによるソフトウェア開発は典型的な協調活動であり、それを支援するCASE環境はグループウェアとしての側面を持っている[垂水92]。ソフトウェア開発では、グループの作業の結果であるソフトウェア生産物の全てを計算機上で扱うことが可能なため、一般の協調活動に比べ計算機支援に適していると考えられる。

グループウェアは、支援対象となる協調活動のモデルを持つものと持たないものに分けることができる。前者のタイプのグループウェアを構造的アプローチに基づくグループウェア、後者を非構造的アプローチに基づくグループウェアと呼ぶ[石井91]。前者は、支援環境に組み込まれたモデルに従って、協調活動の構造を規定し制御する。後者はそのような制約を課さず、透明な媒体に徹する。構造的アプローチは協調活動の定型化によるコミュニケーション・オーバーヘッドの低減を、非構造的アプローチは空間/時間的障壁を解消する共有作業空間の提供をそれぞれ主目的にしたものとも考えられる。

プロセスプログラムの考え方は、作業者の振舞いを規定するという点で構造的アプローチに通じる。従来のプロセスプログラムは、作業者1名の作業プロセスを記述するものであった。それに対し、複数の作業者にわたる作業プロセスを取り扱う場合には、プロセスプログラムはそれぞれの作業者の振舞いと相互作用を規定する。このような複数の作業者のプロセスプログラムの実行を支援する環境は、構造的アプローチに基づくグループウェアに他ならない。

2.3.2 協調活動モデル KCAM

協調活動支援環境KAEは、構造的アプローチに基づいたグループウェアであり、KyotoDBカーネルとの連係により、複数人にわたるプロセスプログラムの実行と、ソフトウェア開発グループ内での一般的な協調活動を支援する。このためのモデルとして、KCAMを導入する。KCAMは、AMIGO Activity Model (AAM) [AMIGO88]をベースに、CASE環境における協調活動を意識した変更を加え、詳細化を行った協調活動モデルである。

KCAMでは、ある特定の1つの目的のために行われる協調活動をアクティビティと呼ぶ。アクティビティの粒度は細かいものから大きなものまで種々のものが考えられるが、KCAMでは比較的細かい粒度のものだけを対象とする。1つのアクティビティは以下の要素から構成される。

- 2つ以上の自律的に動作するエージェント

・　エージェント間で交換されるメッセージ

後述するアーキテクチャ・モデルにおける用語との区別のため、前者をKCAMエージェント、後者をKCAMメッセージと呼ぶ。

さらに、KCAMエージェントが果たすことのできる機能を抽象化したものをファンクション、KCAMエージェントの入力イベントに対する振舞いを規定するものをルールと呼ぶ。ファンクションがどう実行されるのかは、KCAMの対象外とする。

ルールの基本要素は動作部と条件部の2つである。動作部は、KCAMエージェントの動作をファンクションやその順序列の形で記述したものである。動作部は、無条件に実行される場合と入力イベントに対して実行される場合がある。後者の入力イベントの条件を記述する部分が条件部であり、入力イベントの種別とその内容に関する式を記述する。動作部のみあるいは条件部-動作部の組をルール節と呼ぶ。

また、アクティビティ全体の振舞いを規定するのが大域ルールである。大域ルールは、アクティビティの開始や終了、後述するアクティビティの連係を記述するために用いられる。ルールと同様に、大域ルールも条件部と動作部の組からなる。条件部には、各KCAMエージェントのフェーズに関する論理式を、動作部には大域ファンクションをそれぞれ記述する。後述するアクティビティの連係は、大域ファンクションにより実現される。大域ルールの条件部は、KCAMエージェントのフェーズ遷移の際に評価される。

2.3.3　協調活動支援環境KAEのアーキテクチャ・モデル

(1)　概要

実世界の協調活動をKCAMを用いてモデル化する場合、協調活動の参加者である人間そのものを1つのKCAMエージェントとして記述し取り扱うことは非常に難しい。なぜなら、実世界では1人1人の参加者が同時に複数のアクティビティに参加し、それぞれ異なった役割を演じている場合があるからである。

例えば、全く別の開発プロジェクトAおよびBに同時に参加している作業者Wを考える。Wは、Aにおいては電子メールの発信を、Bでは電子メールの受信を行っているとする。プロジェクトAとBの間に全く関連がないとすれば、作業者Wを1つのKCAMエージェントとしてモデル化することは、情報隠蔽という観点からも適切ではない。明らかに、プロジェクトAにおける役割W_AとBにおける役割W_Bをそれぞれ別のKCAMエージェントとし、プロジェクトAとBを別のアクティビティと考えてモデル化することが、この場合には適切である。

よって、アーキテクチャ・モデルではKCAMエージェントを以下のエージェントの組として取り

扱う。

・ 内部エージェント

アクティビティ毎の役割を演じるエージェント

・ 外部エージェント

外界のユーザまたは人工システム(外部存在)に1対1に対応するエージェント

KCAMエージェントは自律的に動作するエージェントであったが、これを分割した内部エージェントと外部エージェントも、それぞれ自律的に動作する。

アクティビティ、内部エージェント、外部エージェントとKCAMエージェントの関連を図2.3-1に示す。この図の例では、3つの外部エージェント U, V, W が存在し、それぞれ外界のユーザまたは人工システム(外部存在)に1対1に対応している。また、相互に関連のない2つのアクティビティ A と B が稼働し、KCAMメッセージがそれぞれの内部エージェント間で交換されている。さらに、各外部エージェントはアクティビティ A と B におけるそれぞれの役割を演じる内部エージェントと接続している。すなわち、アクティビティ A では、外部エージェント U, V, W に対応する外部存在がそれぞれ外部エージェント U, V, W を介し、内部エージェント U_A, V_A, W_A としてアクティビティ A における役割を演じている。また、アクティビティ B では、外部エージェント V, W に対応する外部存在が V, W を介し、内部エージェント V_B, W_B としてそれぞれの役割を演じている。したがって、外部エージェント V, W に対応する外部存在は、2つのアクティビティに別の内部エージェントを介して同時に参加し、それぞれのアクティビティにおける役割を演じていることになる。

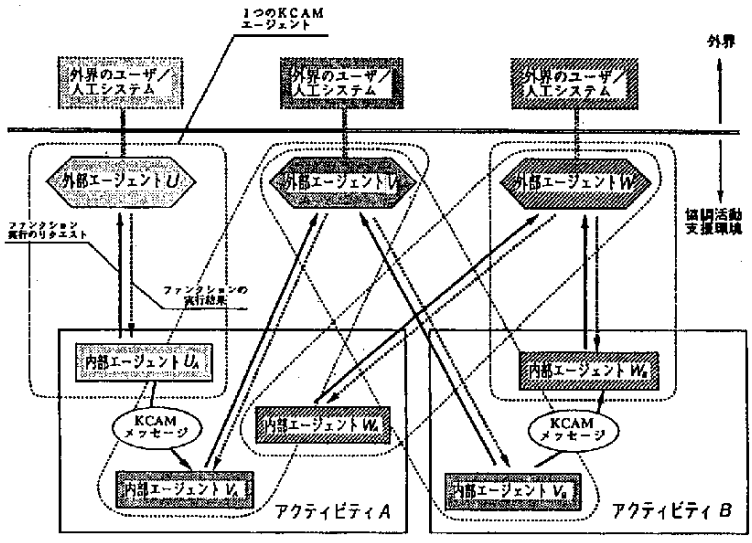


図2.3-1 アクティビティ、内部エージェント、外部エージェントの関連

KAEは協調活動の「支援」を目的とするものであり、「自動実行」を目的とするものではない。したがって、KAEはアクティビティのルール記述を解釈実行するが、必要な場合には外界のユーザと対話を行い、ユーザに実行を委任する。アーキテクチャ・モデルでは、このような実行の主体の違いに着目しファンクションを分類する。

(2) 内部エージェントおよびファンクションの分類

内部エージェントは、KCAMエージェントから特定の1つのアクティビティにおける役割だけを抽出したものである。外部エージェントは複数のKCAMエージェントに対応するが、内部エージェントはKCAMエージェントと1対1に対応するので、アクティビティにおいて内部エージェントを識別するためにKCAMアドレスを用いることができる。これは、図2.3-1において、点線で囲まれた1組の内部エージェントと外部エージェントが、1つのKCAMエージェントに相当することからも明らかである。

KCAMでは、アクティビティの稼働中にそれを構成するKCAMエージェントが増減することはない、と仮定した。したがって、内部エージェントも増減することはない。アーキテクチャ・モデルではさらに、アクティビティの稼働中は内部エージェントと外部エージェントの接続関係が変更されることがないとする。これは、アクティビティの開始時に参加する外界のユーザまたは人工システムが全て確定し、途中参加や退出が許されず、それぞれが演じる役割も最後まで変更されないことを意味している。

内部エージェントは、接続している外部エージェントがどのような外部存在に対応しているかによって、以下の3種類に分類することができる。

Role Agent (RA)

人間または非決定的な外部存在に対応する内部エージェント

Service Agent (SA)

決定性のある人工システムに対応する内部エージェント

Independent Agent (IA)

対応する外部存在のない内部エージェント

KCAMのレベルでは、ファンクションはKCAMエージェントの機能を抽象化したものであり、ファンクションが具体的にどう実行されるかなどの詳細はモデルの範囲外としていた。これに対し、アーキテクチャ・モデルのレベルでは、KCAMエージェントを内部エージェントと外部エージェントの2つに分割した。したがって、ルールは内部エージェントにより解釈実行され、ルール記述に現れたファンクションは内部エージェントから外部エージェントへのリクエストという形で実行される(図2.3-1)。アーキテクチャ・モデルでは、ファンクション実行の主体が何であるかという観点に基づいてフ

ァンクションを詳細化し、以下の3種類に分類する。

Role Function (RF)

人間または非決定的な外部存在に実行が委任されるファンクション

Service Function (SF)

外界の人工システムに実行が依頼されるファンクション

Independent Function (IF)

内部エージェント単独で実行が完了するファンクション

これらのうち、RFがKAEと外界のユーザとの対話に相当する。RFはSF, IFと異なり、その実行が非決定的に行われる。すなわち、RFはユーザによる判断や作業に対応するので、RFが実行されるかどうかはKAEの内部からは知ることができない。これに対し、SF, IFは決定的に実行され、エラー等の問題がなければ必ず結果が返される。

表2.3-1　内部エージェントが用いることのできるファンクション

実行する内部エージェント ＼ ファンクション	RF	SF	IF
Role Agent	○	○	○
Service Agent	×	○	○
Independent Agent	×	×	○

各内部エージェントが、どのファンクションをそのルール記述で用いることができるかを表2.3-1に示す。これにより、RAはRF, SF, IFの全てを、SAはSFとIFを、IAはIFのみをそれぞれルール記述の中で用いることができるということがわかる。

ファンクションはまた、アクティビティに特定の機能を果たすものと、一般的な機能を果たすものの2つに分類することができる。前者をActivity-Specific Function、後者をGeneral Functionと呼ぶ。この分類はRF, SF, IFの分類とは直交するものである。KCAMメッセージの送信、受信とフェーズの遷移は、典型的なGeneral Functionである。

(3)　外部エージェント

外部エージェントは、外界のユーザまたは人工システムに1対1に対応するエージェントである。外部エージェントは、対応する外部存在の性質の違いにより2種類に分けられる。ユーザまたは非決定的な性質を持つ人工システムに対応するものをUser Agent (UA)、決定的な人工システムに対応するものをExternal System Agent (ESA)と呼ぶ。UA, ESAはともに、KAE内で一意な識別子を持ち、対応するユーザまたは人工システムの状態に関わらず常に存在し動作している。

(4)　協調活動スキーマ, パラメータ, インスタンス

これまで「アクティビティ」という用語を、協調活動そのものを指すものとして用いてきた。アー

キテクチャ・モデルではさらに、協調活動スキーマ, インスタンス, パラメータ という概念を導入する。

協調活動スキーマ は個々の協調活動の枠組を定義するものであり、表2.3-2に示す要素からなる。さらに、一意な識別子(協調活動スキーマ名)を持つ。

協調活動パラメータ は協調活動スキーマを稼働させるための情報であり、表2.3-3に示す項目がある。既に述べたように、アクティビティの稼働中は内部エージェントと外部エージェントの対応関係は変更されないので、協調活動パラメータのRA およびSA のインスタンスの数と、それぞれが接続するUA およびESA の識別子はアクティビティ の終了まで変更されない。

協調活動スキーマとパラメータ から生成されて稼働しているものを 協調活動インスタンス と呼ぶ。これまで用いてきた「アクティビティ」はこれと同義である。協調活動インスタンスはKAE 内で一意な識別子を持つ。

(5) 管理エージェント

Internal Management Agent (IMA) は、1つの協調活動インスタンスに1つだけ含まれる。IMA はまた、自分の管理する協調活動インスタンスに含まれる内部エージェントの表を持っている。この表の主な項目は内部エージェントのエージェント型、KCAM アドレス および 最新のフェーズである。IMA がKCAM メッセージの配送を行う際には、受信者属性の KCAM アドレスを解釈し、この表を用いて対応する内部エージェントを特定してKCAM メッセージを分配する。さらに、内部エージェントがどのフェーズにあるかをモニタし、大域ルールを実行する。

Global Activity Management Agent (GAMA) は、KAE 内に1つだけ存在する。IMA と同様に、GAMA は KAE 内の全ての協調活動

表2.3-2 協調活動スキーマの構成要素

- メッセージ型の定義
- エージェント型の定義
  - 受信できるメッセージ型 (あるいは、そのリスト) の宣言
  - 発信できるメッセージ型 (あるいは、そのリスト) の宣言
  - 遷移できるフェーズの宣言
  - ファンクションの宣言 and/or 記述
    - * ルール記述に用いることのできる Role Function の宣言
    - * ルール記述に用いることのできる Service Function の宣言
    - * Independent Function の宣言と記述
  - ルール記述
    - * 変数宣言
    - * フェーズ毎の条件部 – 後続部 (複数) の記述
- 大域ルールの記述

表2.3-3 協調活動パラメータの項目

- RA のインスタンスの数
- RA のインスタンスのそれぞれが接続する UA の識別子
- SA のインスタンスの数
- SA のインスタンスのそれぞれが接続する ESA の識別子
- IA のインスタンスの数
- その他の初期条件

インスタンスの表を持ち、後述する協調活動インスタンスの連係において重要な役割を果たす。また、GAMA は協調活動スキーマのライブラリを管理しており、外部から協調活動スキーマ名とパラメータを与えられると、新しい協調活動インスタンス を生成する。

(6) 複数の協調活動インスタンスの連係

モデル化の対象になる協調活動が複雑なものであっても、協調活動全体を複数のアクティビティに分割することにより、KCAM によるモデル化が行える程度まで粒度を細かくすることが可能になる場合が考えられる。このような場合、分割してモデル化した個々のアクティビティを連係させることにより、最終的に協調活動全体のモデル化が行える。

例えば、グループによるソフトウェア開発において、作業者 Vは決定事項に対し、まずグループのサブリーダー L_1の承認を得て、次にリーダー L_2の承認を得なければならないという規則があるとする。これを電子メールを用いて行う協調活動のモデル化を考えてみる。全体を1つのアクティビティとしてモデル化することも可能ではある。しかし、承認を得る手順 Pが、相手がサブリーダーかリーダーかに関係なく同じだとすると、作業者 Vは以下の2つの手順を直列に実行することにより最終的な承認を得ることができる。

1. サブリーダー L_1の承認を得るための手順 P
2. (1.が成功した場合) リーダー L_2の承認を得るための手順 P

すなわち、承認を得る手順 Pを1つのアクティビティとし、このアクティビティを2つ直列に実行する。承認を得る必要がある相手の人数が多くなる場合を考えると、このようなアクティビティの分割とその連係によってモデル化が容易になることは明らかである。

KCAM では、複数のアクティビティ間の相互干渉はありえないとしていた。アーキテクチャ・モデルでも、稼働中の協調活動インスタンス相互の干渉はありえない。ただし、アクティビティの分割と連係を可能にするため、協調活動インスタンスの起動と終了においてのみ、2つの協調活動インスタンスが同期できるものとする。これは大域ルールにおいて、協調活動インスタンスの起動と終了待ちを司る大域ファンクションである *fork* と *wait* を用いることにより実現される。

fork は引数として協調活動スキーマ名と協調活動パラメータを取り、新しい協調活動インスタンスを生成する。*fork* で生成されたインスタンス(子アクティビティ)には、*fork* を実行したインスタンス(親アクティビティ)の識別子が渡される。同様に、親アクティビティには子アクティビティの識別子が返される。

wait は、引数で指定された識別子を持つインスタンス (子アクティビティ) の終了まで、*wait* を実行した側のインスタンス(親アクティビティ)の実行をブロックする。

fork, wait は、ともに IMA で解釈され、IMA から GAMA へのリクエストにより実行される。図2.3-2に、*fork* と *wait* による協調活動インスタンスの連係のパターンを示す。

Aは、ある協調活動インスタンスから別の協調活動スキーマを部品として利用する場合などに相当する。また、一時的に外部エージェントと内部エージェントの対応関係を変更する場合にも用いることができる。BはAに似ているが、親アクティビティがブロックされずに子とは関係なく進行する点が異なっている。Cは、外部エージェントと内部エージェントの対応関係を半恒久的に変更してしまう場合などに用いられる。

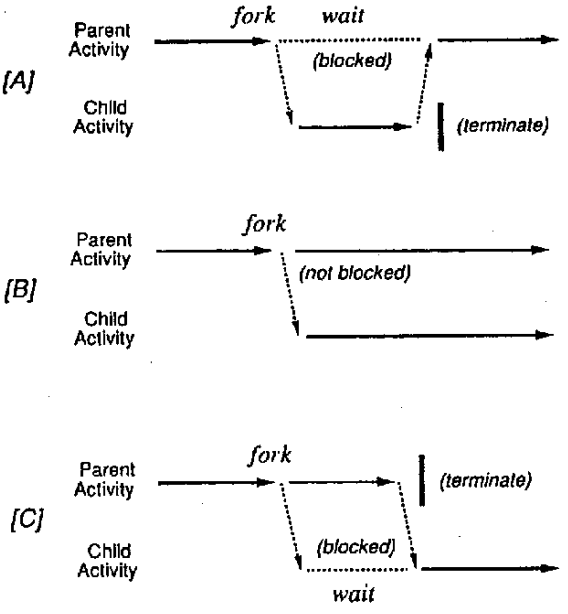


図2.3-2 協調活動インスタンスの連係のパターン

2.3.4 協調活動スキーマ記述言語 KCASL

協調活動スキーマ記述言語 KCASL は、協調活動スキーマを実行可能な形で記述するための言語である。KCASL は、2.3.6 で述べる UNIX 上での実装を前提に、C言語をベースにして設計されている。KCASL では 2.3.3の(4)の表2.3-2に示されたルール節や、その他の協調活動スキーマの要素を記述するための文法が定義される。

KCASL の設計上の要点を以下に列挙する。

(1) メッセージ型 と KCAM メッセージ の取扱い

メッセージ型 は `MessageTypes` 節で定義する。C言語の構造体と同様に、送信者属性、受信者属性、一般属性は、それぞれ構造体のフィールド `from`, `to`, `Contents` として扱う。よって、KCAM メッセージ の各属性の参照と代入がC言語の構造体に対する操作と同様に行える。さらに、一般属性 `Contents` も構造体であり、メッセージ型に応じて必要なフィールドを定義する。

(2) `select` 文によるルール節の記述

ルール節は、`RuleDescription` 節に記述される。C言語は並列に条件を評価する構文を持たないため、`select` 文を追加してこれを記述する。KCAM では入力イベントを KCAM メッセージの受信と 時間イベントの2種類としていた。KCASL では、Role Agent のルール 記述の場合のみ、Role

Function の実行を入力イベントとして条件部に記述できる。これにより、複数の Role Function の実行を一つのフェーズで待つようなルール記述が行える。

(3) 変数

変数は VariableDeclaration 節において宣言する。スコープは宣言したエージェント型のルール記述全体である。

(4) ファンクション の宣言と呼び出し

Independent Function は、IndependentFunction 節において、C言語の関数と同じ形式で宣言され記述される。Role Function, Service Function もそれぞれ RoleFunction, ServiceFunction 節において、C言語の関数と同じ形式で宣言される。関数の引数、返り値に関する規則もC言語と同じである。

(5) 制御構造

Independent Function の中や、select 文の各後続部では、C言語の任意の制御構造が使用できる。ただし、後続部から外部へ脱出したり、他の後続部や Indepent Function の中に分岐するような制御構造の記述は許されない。

2.3.5 KyotoDB における協調活動支援

(1) KyotoDB の概要

KyotoDB は、ソフトウェア開発のプロセスと生産物(プロダクト)を管理するソフトウェア・エンジニアリングデータベース(KyotoDB カーネル)と その開発支援環境全体の総称である。KyotoDB の目的とするところは以下の3つである[鯵坂92]。

(a) 生産物とその要素相互の意味的関係の管理

生産物(プロダクト)とは、仕様書,ソースコード,テストデータ等のソフトウェア構成要素である。意味的関係の管理とは、従来のファイル単位の版管理、構成管理ではなく、各生産物に含まれる意味要素を抽出し、それらの関係を種別化して管理することである。これにより、生産物に対する変更の波及解析などを可能にする。

(b) 生産物とプロセスの統合管理

生産物を作成していくプロセス(プロセスプログラム)の記述やその実働条件記述、生産指標を生産物と統合して管理する。

(c) 協調作業の支援

これが KAE の機能に対応する。

KyotoDB では、作業者1人の担当する作業単位を Unit Workload (UW) と呼ぶ[Matsumoto90]。また、UW によって作成される生産物を 目的生産物 と呼び、目的生産物を作成するために参照される生産物

を 参照生産物 と呼ぶ。1つのソフトウェア開発プロジェクトは、参照関係に従って構成されたUWの半順序集合(UWネットワーク、UWNet)という形で扱われる。UWNetはプロジェクトの開始時に作成されるか、必要な場合にはプロジェクトの進行に伴って変更、作成あるいは分割されていく。プロジェクト全体の再利用は、記録されていたUWNetを元に、必要な部分にのみ変更を加えることで行える。

(2) KyotoDB カーネル

KyotoDB カーネルは、オブジェクト指向データモデルに基づいており、以下の4つのクラスのオブジェクトが複合構造をなしている[鰺坂92]。これを図2.3-3に示す。

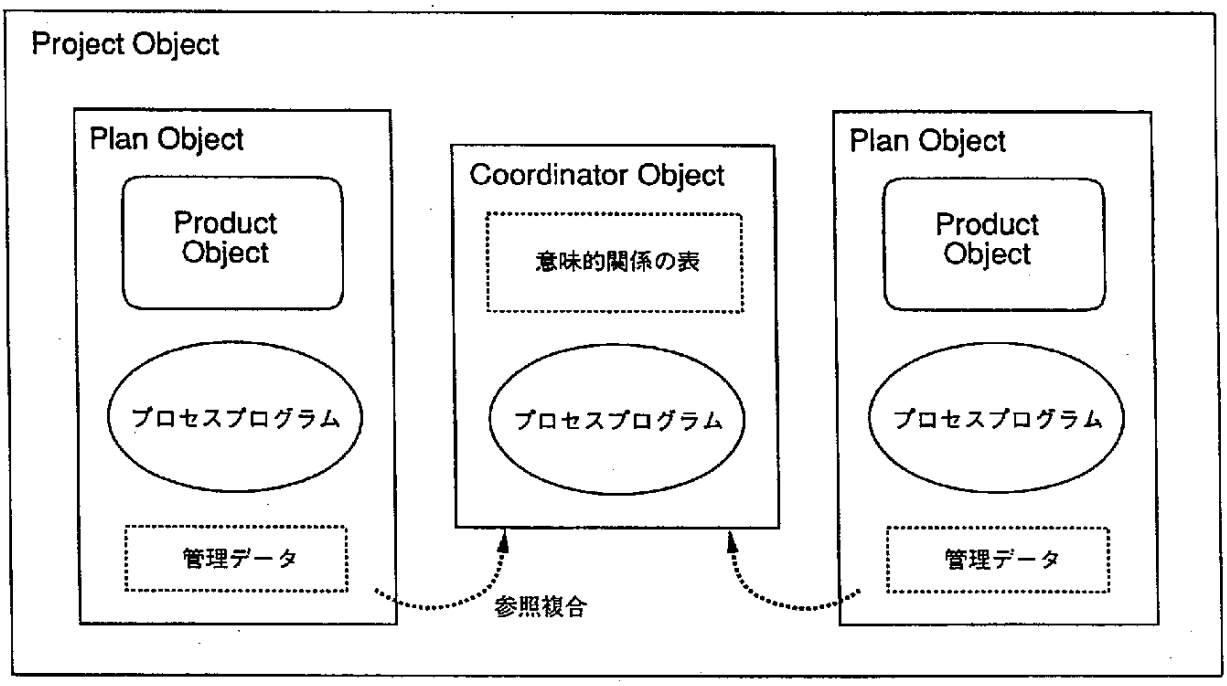


図2.3-3 KyotoDB カーネルにおけるオブジェクト

(a) Project Object

プロジェクト全体を管理するオブジェクトであり、KyotoDB カーネルと外界のインタラクションの窓口となる。すなわち、他のオブジェクトは全てProject Objectの中に隠蔽されており、外界からの直接のアクセスは許されない。

(b) Plan Object

1つのUWのプロセスプログラムと、その管理データを格納するオブジェクト

(c) Product Object

1つのUWの目的生産物を格納するオブジェクト

(d) Coordinator Object

2つの生産物の相互の意味的関係を管理し、生産物の変更による再作業の発生の可能性(波及効果)を検出して、その解決のためのプロセスプログラムを格納するオブジェクト

Plan Object は Product Object を構成複合の形で包含している。同様に、Project Object は Plan Object と Coordinator Object をそれぞれ構成複合の形で包含している。また、Plan Object はその参照生産物との関係を保持している Coordinator Object へのリンクを持っている(参照複合)。

Product Object 中の目的生産物や Project Object に含まれるオブジェクト中の全てのデータは、オブジェクト・ストリーム (O-Stream) という形式で KyotoDB カーネル外部と交換される。O-Stream は、KyotoDB カーネルにおけるオブジェクトの複合構造を忠実に反映しストリーム化したものであり、オブジェクトが保持しているデータと等価な情報を持つ。

(3) 作業者1名のプロセスプログラムの実行

KyotoDB カーネル中の目的生産物および参照生産物は、O-Stream に変換されて KyotoDB カーネル外部と交換される。また、目的生産物に対する作業者の作業過程は、操作列という形式で表現される。操作列は、目的生産物に対する意味的な単位操作の有限列である。

作業者1名のプロセスプログラム(UW *A* に対するものとする)は、以下のような手順で実行される。まず、全ての参照生産物が完成すると、KyotoDB カーネルは UW *A* を活性化する。作業者は、KyotoDB へのログインの際に活性化している自分の UW を選択し、KyotoDB からの指示や方針の示唆などに従って、参照生産物を閲覧しながら目的生産物を編集していく。編集に伴って蓄積された操作列は、コミットによって KyotoDB カーネルに送られる。送られた操作列はまず、その UW *A* に対応する Plan Object に渡される。Plan Object は次に、参照複合している全ての Coordinator Object に操作列を送る。Coordinator Object はさらに、参照生産物との意味的関係を用いて操作列を解析し、波及効果を調べる。問題になるような波及効果が検出されなければ、最終的に操作列は Product Object に送られ、目的生産物のデータが更新される。以上によりコミットが完了する。作業者は編集とコミットを繰り返し、UW A の作業が完了したと判断したらその旨 KyotoDB に通知する。

以上のプロセスプログラム (以下 "Plan1" と表記) を、KAE において実働化することを試みる。まず、Plan1 は KyotoDB カーネルと UW の担当者という2つの KCAM エージェントによるアクティビティであると考える。また、操作列や O-Stream は形式化されているので、KCAM メッセージとして取り扱える。したがって、KCAM によるモデル化と協調活動スキーマ記述を行えば KAE 上で稼働させることができる。以下、必要なエージェント型、メッセージ型およびファンクションについて述べる。

まず、Plan1 の協調活動スキーマにおけるエージェント型は以下の2種類であると考えられる。

(i) PLAN

UW での KyotoDB カーネルの役割を演じる SA

(ii) WORKER

UW の担当者の役割を演じる RA

主なファンクション には以下のものがある。

(a) 単位操作( )

作業者の単位操作に対応する WORKER の RF。この単位操作は次回のコミットまで蓄積される。

(b) コミット( )

作業者のコミットに対応する WORKER のRF。これの実行を受け、蓄積されていた操作列が PLAN に送られる。

(c) 作業完了( )

作業完了に対応する WORKER の RF。前回のコミットから単位操作が行われていない場合だけ実行が許される。理想的には、KyotoDB カーネルが UW の完了を検査する必要があるが、ここでは作業者の判断に任されるとしている。

(d) 目的生産物更新( )

Product Object に格納されている目的生産物のデータの更新に対応する PLAN のSF

また、主な KCAM メッセージは以下のとおり。

(a) productMsg

目的生産物を最初に WORKER に送る

(b) commitMsg

コミット( ) の際の操作列を送る

(c) doneMsg

作業完了( ) を PLAN に通知する

Plan1 の起動は、UW の活性化にそのまま対応する。すなわち、UW の参照生産物が全て完成したことを KyotoDB カーネル が検出し、KyotoDB カーネル から KAE (GAMA) に対して UW の協調活動スキーマ名 (この場合 "Plan1") とパラメータ (作業者に対応する UA のID など) が送られ、協調活動インスタンスが起動される。作業者のログイン,ログアウトに関わらず、作業完了までこのインスタンス は KAE 内で稼働し続ける。

(4) 協調活動支援

現在の KyotoDB カーネルでは、任意の1つの UW は特定の1名の作業者にだけ割り当てられる。すな

わち、複数の作業者に共有されるUW(目的生産物)は存在しない。したがって、目的生産物と参照生産物の関係の一貫性が損なわれた場合にのみ、複数の作業者の協調活動が行われる。このような関係の一貫性の破れ(波及効果)は、あるUWにおける作業によって目的生産物が変更された時に、それを参照しているUWが既に存在し、その間に関係が定義されている場合に発生しうる。

この節では、2.3.5の(3)で述べた作業者1名のプロセスプログラム "Plan1" に対し、コミットによる波及効果の解決のための機能を追加することを考える。すなわち、2.3.3の(6)で述べた協調活動インスタンスの連係を利用して、もう1つの協調活動インスタンス(子アクティビティ)を起動し、波及効果の解決を任せる。その際、波及効果の内容に応じた協調活動スキーマ名を決定し、パラメータを波及解析によって求める。協調活動スキーマ名は、波及効果の緊急度に基づいて決定できる。すなわち、緊急度の高い場合はリアルタイム型を、低い場合は非同期型のスキーマを選ぶことになる。協調活動パラメータは、波及効果の及ぶ他の作業者が誰であるかを示す。

また、子アクティビティとの連係のパターンは、波及効果の重要度に応じて変化する。すなわち、重要度が高い場合には、子アクティビティの結果によってはコミットが失敗することがありうる。一方、波及効果の重要度が低い場合は、子アクティビティの結果に関わらずコミットは常に成功する。これらは、2.3.3の(6)の図2.3-2に示したパターンAとBにそれぞれ対応する。

以上のように拡張されたプロセスプログラムの協調活動スキーマ名を "Plan2" とする。Plan2ではPlan1に加え、Coordinator Objectによる波及効果解析と、子アクティビティ形態の決定を行うために、以下のファンクションが必要になる。

(a) 波及効果解析( )

Coordinator Objectによる波及効果解析メソッドに対応するPLANのSF。

(b) 子スキーマ決定( ), 子パラメータ決定( ), 連係パターン決定( )

子アクティビティの協調活動スキーマ名、パラメータおよびPlan1と子アクティビティがどのように連係するかを決定する。これもCoordinator Objectのメソッドに対応するので、PLANのSFである。

また、波及解析や子アクティビティの結果を通知するため、以下のKCAMメッセージが追加される。

(a) noProblemMsg

問題となるような波及効果がなかった

(b) forkMsg

子アクティビティの協調活動スキーマ名など

(c) solvedMsg

子アクティビティによる解決が成功した

(d) notSolvedMsg

子アクティビティによる解決が失敗した

さらに、子アクティビティとの連係のための大域ルールが用意される。ここでは、WORKER と PLAN が波及効果解決フェーズに遷移した場合に、PLAN から forkMsg を受け取り、それに応じて子アクティビティを起動する。その後、子アクティビティの結果が必要ではなかった場合と子アクティビティによる解決が成功した場合には solvedMsg が、それ以外の場合には notSolvedMsg が WORKER と PLAN に対して送信される。WORKER と PLAN はいずれかの KCAM メッセージを受信し、動作を再開する。

図2.3-4に、Plan2 の協調活動スキーマ のグラフによる表現を示す。

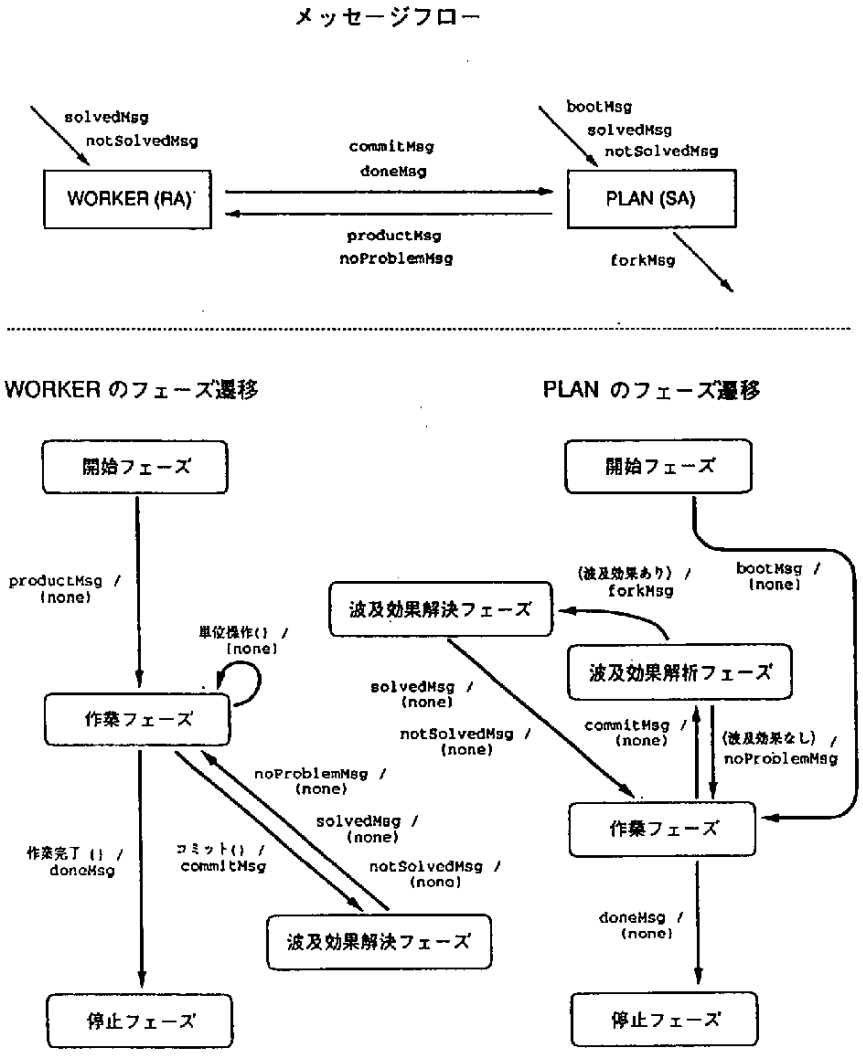


図2.3-4　Plan2 の協調活動スキーマ のグラフによる表現

2.3.6 KAE の実装

(1) アーキテクチャ・モデル の UNIX モデルへの変換

アーキテクチャ・モデルの構成要素と必要なサービスのうち、主なものについてそれが UNIX 上のどの概念に変換されるかを示す。

(a) 内部/外部/管理エージェントの実現

アーキテクチャ・モデルでは、これらのエージェントは全て自律的に並行動作する。よって、UNIX では各々をプロセスとして実現する。ただし、UNIX は種々のネットワークサービスを提供するが、プロセスの制御に関しては完全な位置透過性を持たない。すなわち、ネットワークを介して他のマシンの上のプロセスを制御する場合と、ローカルマシンのプロセスを制御する場合では、制御の方法が異なっている。したがって、KAE は単一のマシンの中で閉じているものとする。また、外部エージェント のうち UA は、外界のユーザ1名に対し1つ存在する。したがって、ユーザのワークステーションで稼働している Xウィンドウシステムのサーバ (X サーバ)を UA と見なす。

(b) KCAM メッセージの送受信

内部エージェントのインスタンスをそれぞれ別のプロセスとしたため、KCAM メッセージはプロセス間通信により送受信される。このための手段として、ソケット[Stevens88]を用いる。また、内部エージェント と IMA 、IMA と GAMA 間の通信にもソケットを利用する。プロトコルとしては一般的なコネクション指向プロトコルである TCP／IP を用いる。よって、プロセス間通信を利用するエージェント間では、起動時に接続が確立される。

(c) Service Function, Role Function の実行

SF とRF はそれぞれ、内部エージェントから外部エージェントへ実行が依頼または委任される。SF は外部エージェントから外界の人工システム へのリクエストにより実行される。KAE では、UNIX 上の他の受動的システムのみを人工システムとして取り扱う。RF は、前述したようにユーザの自発的な選択や作業に相当する。KAE では、ユーザ が RF を実行するための手段として、ツール を導入する。RF の粒度には種々のものがあるため、ツールには全体が1つの RF に相当するものや、ツールのコマンドがそれぞれ RF に相当するものなどがあると考えられる。つまり、あるフェーズにおいてルール節に現れている RF は、ユーザによるツールの実行やツールのコマンドの選択肢に対応する。ツールは内部エージェントから起動され、内部エージェントと双方向に通信する。ここでもソケットによるプロセス間通信を用いる。

(2) KCASL の実装方針

2.3.4で述べたように、KCASL はC言語をベースに設計されている。したがって、拡張された文法要素を変換し、C言語のみで記述できる形式に帰着させることにより、UNIX 上のC言語の言語処理系とライブラリがそのまま利用できる。このような変換を行う処理系を KCASL プリプロセッサと呼ぶ。

この項では、KCASL プリプロセッサによる変換の方針と、変換されたプログラムを実働化するために必要なライブラリの機能について説明する。

KCASL において拡張された文法要素のうち、重要なのは KCAM メッセージの取り扱いとルール節の記述の2点である。KCAM メッセージは、内部エージェント上ではメモリ上の構造体として扱われる。KAE は単一のマシンの中で閉じているものとしたので、KCAM メッセージを他の内部エージェントと送受信する際には特に変換を行う必要はなく、メモリイメージをそのまま送信すればよい。しかし、自己参照ポインタを含むリストや可変長データの取り扱いには工夫が必要になる。また、分散環境上に KAE を拡張することを想定すると、機種依存のメモリイメージをそのまま送受信することはできない。このような問題の解決のため、XDR (eXternal Data Representation) [Sun89] を用いる。XDR は異機種間の RPC (Remote Procedure Call) の際に、C言語で定義される主なデータ構造を機種に依存しない形式に変換するためのパッケージである。KCASL によるメッセージ型の定義は、そのまま XDR の定義に変換される。

ルール節では、複数の入力イベントを並列に待つ構文である `select` 文が拡張されている。入力イベントには、KCAM メッセージの受信、RF の実行および時間イベントの3種類である。KCAM メッセージの受信と RF の実行は、それぞれ IMA と接続した外部エージェントからのソケットを介してのデータ受信に対応する。したがって、この2つは UNIX のシステムコール `select()` により並列に待つことができる。時間イベントは、UNIX のタイマ機構を利用し割り込みにより検出する。`select()` の実行中も割り込みは有効なので、3種類の入力イベントを並列に待つことができる。

KCASL による記述から最終的に UNIX の実行ファイルを生成し、協調活動スキーマを実働化する過程を図2.3-5に示す。現時点では、図中の KCASL プリプロセッサを除く部分について実装が完了している。また KCASL による変換方針も確立している。

(3) ツールの実現と制御

ツールは内部エージェントから起動され、内部エージェントと双方向に通信する。ツールによる操作に対し、内部エージェントが受動的にだけ動作するのであれば、起動時以外はツールから内部エージェントへの方向にしか通信は行われない。しかし、ツールの動作時における他の入力イベントの発生に対応したり、ツールを介しユーザの作業を誘導するなどの機能を実現するには、ツールと内部エージェントは双方向に通信する必要がある。

UA として X サーバを利用するため、ツールのユーザインタフェースには X ウィンドウのツールキットを用いることができる。今のところ、ツールキットとしては 鼎[CANAE91] を用いている。

2.3.7 おわりに

最後に、KyotoDB と KAE の開発状況、および将来の課題についてまとめる。

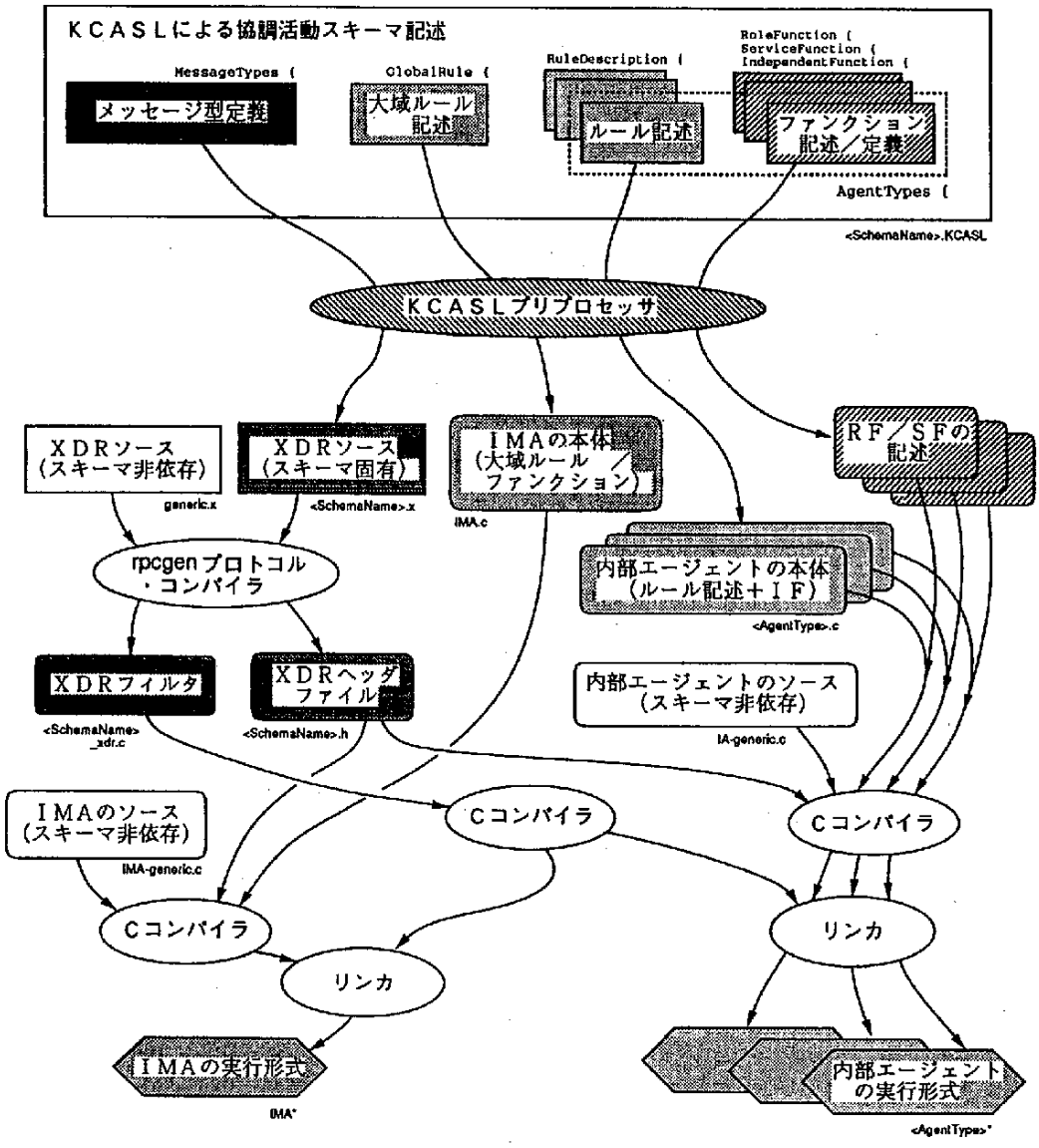


図2.3-5　協調活動スキーマの実働化の過程

KyotoDB カーネルは、設計と COB による実装がほぼ完了している。KAE の設計もほぼ完了しており、KCASL処理系を除きプロトタイプの実装に成功している。現在、KyotoDB カーネルと KAE を接続する作業と、各種ツールの設計と実装を進めている。

将来の課題としては、以下のものが挙げられる。

(a)　KCAM の拡張

現状では、KCAM でモデル化できる協調活動は非常にプリミティブなものに限られている。より複雑な現実の協調活動をモデル化できるような拡張が望まれる。

また、大域ルールは通常のルールに対するメタレベルのルールであると考えられる。現在は、大域

ルールは疑似コードのレベルでしか記述できていないが、このようなリフレクションの概念を導入することにより、厳密な記述が可能になると考えられる。

(b) モデル化とスキーマ作成の方法論の確立

実際の協調活動をKCAMによりモデル化し、協調活動スキーマとして記述する際に、何らかの系統的な手法なしには複雑な協調活動が取り扱えない。すなわち、KCAMエージェントやフェーズの同定、KCAMメッセージやファンクションの同定と定義、さらにルール記述の方法論を確立しなければならない。

また、協調活動スキーマの作成は一種の並行プログラミングであるので、適切にスキーマが記述されていないと、協調活動インスタンスの稼働中に容易にデッドロックが発生する。したがって、スキーマの検証を行う手法が必要である。

さらに、協調活動スキーマにおけるユーザの自由度が適切であるかを判断する基準を設け、KAEによる支援が逆にユーザの負担を増すことを防がねばならない。これはグループウェアに普遍的な課題でもある。

(c) 分散化およびオープン化

現在、KAEは単一のUNIX環境において実装されている。また、メッセージハンドリングを内部で行っているため、真に分散環境に対応できておらず、開放性も低い。これらの問題の解決のため、分散OSにおいてKAEを再構築するか、MHSなどの開放性の高いサービスを利用することを検討中である。

(d) UWの作成方法論の確立

KyotoDBにおけるプロセスプログラムでは、UWが適切に作成され、それぞれの生産物の担当者と参照関係が確定していることが前提になっている。現時点では、適切なUWNetを構成するための方法論が確立していない。このためには、現実のソフトウェア開発工程の分析とUWNetによる表現を試みることが欠かせないと思われる。

【参考文献】

[Matsumoto90] Matsumoto, Y. and Ajisaka, T.: A Data Model in the Software Project Database KyotoDB,JSSST Advances in Software Science and Technology,Vol.2, pp. 103-121 (1990).

[鯵坂92] 鯵坂 恒夫, 大森 匡, 松本 吉弘 :ソフトウェアエンジニアリング・データベースKyotoDBのオブジェクトモデルとその実現, オブジェクトテクノロジーの高度応用に関するObaseワークショップ論文集,pp. 89-96, (1992).

[垂水92] 垂水 浩幸: 分散開発環境 : 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).

[石井91] 石井 裕: グループウェアのデザイン --- 構造的アプローチと非構造的アプローチ,bit, Vol.23, No.3, pp. 273-283 (1991).

[AMIGO88] Pankoke-Babatz, Uta ,et al.: Computer-Based Group Communication --- The AMIGO Activity Model,Ellis Horwood Limited (1988).

[Stevens88] Stevens,W. Richard : UNIX(R) NETWORK PROGRAMMING,Prentice-Hall Inc. (1990).

[Sun88] The Portable ONC/NFS Group: Documentation for RPC, XDR and NFS, RPCSRC 4.0 distribution,Sun Microsystems Inc. (1988).

[CANAE91] 日本電気株式会社: UX ソフトウェア 鼎(かなえ) プログラミングの手引 (1991).

2.4 協調支援環境 Vela による分散開発支援

2.4.1 ソフトウェア開発における協調作業に関する問題点

ソフトウェア開発は、プロジェクトチームを構成する、必ずしもスキルレベルはそろっていない専門家集団の協調作業によって進められる。チームの構成員は、過去のシステム開発により経験的に得られた知識を再利用しつつ、CASEツールを使用することによって作業の自動化の度合を高めながら、各自に課された課題を解決していく。各自が割り当てられた仕事を首尾よく達成しつつ、全体としては、共通のゴールを共有・理解してプロジェクトを成功に導くためには、

(1) データベース技術が支援する成果物の管理機構

(2) 構成員間のコミュニケーションを支援するネットワークによる情報交換機構

(3) プロジェクト構成員の仕事の進め方にガイドラインを与え、また作業局面に応じたCASEツール群を統合的に利用することを支援するプロセスモデル

等の存在が必要であり、それらの要素技術が組合わせられ、適合されて、

(1) 各自の持つ不完全で部分的な知識の統合

(2) ソフトウェア中間生成物とそれに附随するデザイン・コンセプトを、関連する人々に正確に伝達し、確認すること

(3) 各作業単位毎に、必要な視野やコンテキストを設定すること

等に関する有効な支援が実現されなければならない[落水92-1]。

今日の分散開発環境の出現によりコミュニケーション支援の重要性はますます増加しつつある。たとえば、

(1) マクロなレベルでは、距離的に離れた場所に存在する開発チーム間のコミュニケーションの支援

が十分でなく、また、工程対応で分割されたソフトウェア開発プロセスが分散開発に適していない等の問題がある

(2) またミクロなレベルではEthernet-workstationsに代表される環境下で、アイディア開発やブレーンストーミング、問題解決と意志決定、意志伝達と合意形成等に関して、計算機のもつ能力をどのように活用すべきかが明確でない。

分散環境下にあっては、協調作業のための時空間の制約が厳しく、そこを緩和する方式が明確ではない。また、情報の表現メディアが十分でなくコミュニケーションに関する制約が大きい等の問題がある。このような状況では、以下のような、人間のもつ弱みが顕在化する。

(1) 人間は忘れやすい生き物である(時空間分散に弱い)

(2) 人間は不完全な生き物である(あいまいな情報や矛盾した情報をつくりだしつつも高度な判断をする)

(3) 情報の価値は役割と状況によって変化するがそれを適確に把握できない(状況の変化に弱い)

このような人間の弱点を補強することにより、チームや人間の能力を増幅する機構の開発こそ最も必要とされることであろう。

2.4.2 ソフトウェア開発環境の近未来像

ソフトウェア開発環境の近未来像として筆者は図2.4-1に示すようなサービスが必要であり可能であると考えている[落水91-1]。

図2.4-1において、環境利用者へのサービス機能としては、

(1) Coordination支援(ソフトウェアオブジェクトベース)

設計活動のゴール、すなわちソフトウェア設計活動を通じて生成される中間生成物やその依存関係および品質基準を定義し、また、並行的活動によって生じる情報の共有や分担(版管理、変更管理)の制御を支援する。

(2) Navigation支援(熟練者の作業プロセスの再利用)

より経験の少ない者が質のよい中間生成物を作成するための手順の支援。具体的には、開発方法論の枠組や詳細手順を与え、また、ソフトウェアオブジェクトの状態を変化させる操作(ツール機能)を統合化する役割をもつ。

(3) Communication支援(情報交換・伝達の支援)

ソフトウェア開発時に生じる、合意形成、意志伝達、技術情報の問い合わせ等によって生じるチーム構成員間の情報交換の支援。

(4) Control 支援（集団活動の制御）

資源割当てや開発スケジュール等のプロジェクト計画支援、および危機状態の検出や対応等のプロジェクト監視支援。

の4つが挙げられる。また環境開発者への支援機能も必要である。

(5) Adaptation 支援（ユーザ固有の作業形態に適合した開発環境の生成）

上記4つの機能はプロジェクトチームやプラットホームによってその必要度や実装法が異なる。例えば、数人からなるスキルレベルの高いチームの場合はCoordination支援が支配的となりCommunication支援も軽く実装すればよい。一方、100人以上の構成員からなるチームの場合はControl支援が支配的となり、プロジェクトの目標達成にあたって予想される問題点(開発過程で生じうる制約)をクリアしつつ大勢の人間の歩調をそろえさせるプロジェクト立案が重要である。Communication支援も合意形成や意志伝達確認に力点をおく必要がある。すなわち、プロジェクトの規模[Per88]、対象分野の特性、チーム構成員の分散度、データの安全性や機密度、利用する計算機設備等のパラメータに応じてメタ SDE (Software Development Environment) からスペシフィック SDE を生成する環境開発者の支援が必要である [Tul90]。

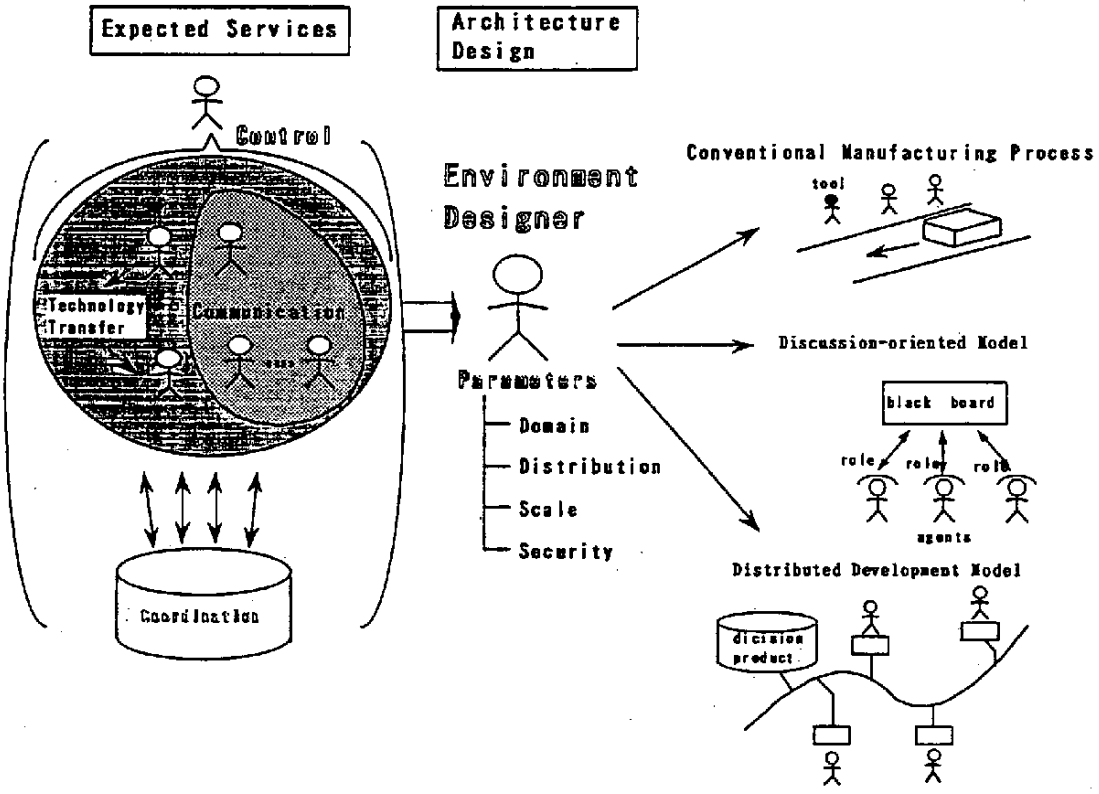


図2.4-1 ソフトウェア開発環境の近未来像

2.4.3 Vela 開発の背景と目標

図2.4-2 は、図2.4-1 における5つのサービス機能のうち Adaptation を除く4つの機能について、研究室で開発中のプロトタイプ Vela における機能とその相互関係を示したものである [落水90]。プロトタイプ開発の目的は、前節でのべた要請に対する有効な仮説を設定し、モデルを設計するための実験設備を構築することである。

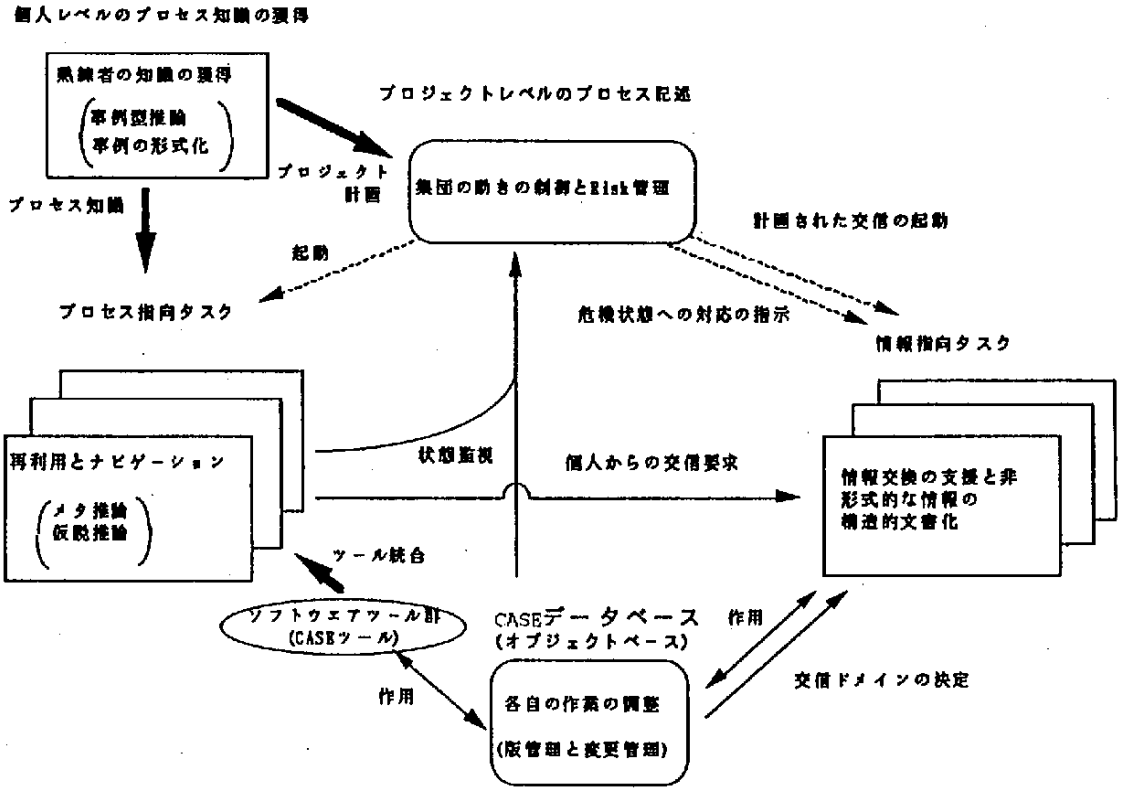


図2.4-2 Vela機能の相互関係

2.4.1 節で述べた問題点に対して以下のような問題点の切り出し方をすることにより、ナビゲーション支援とコミュニケーション支援に関して、現在の研究の力点をおいている [落水92-2]。

(1) 協調開発における個人の問題

「使い慣れた、良い言語と環境を使って、ちゃんとした常識を持った人が、ゆとりをもって作れば、良いソフトウェアができる」のが理想ではあるが、現状では、対象分野や開発経験に依存して形成される常識を取り扱う技術がなくソフトウェア設計活動を困難にしている。

(2) 協調開発におけるチームの問題

「多大な工数を費やして開発しなければならないようなアプリケーションにおいては、すべてを理解している人は組織のどこにも存在せず、組織として理解していることの全体は多くの経験豊富な専門家一人一人に分散している[シュレ90]」。この事実は開発規模の増大につれて顕在化し、分散環境下

においてはさらに増幅される傾向にある。

このような問題に答えるためには、(i) 経験豊富な人間の中に潜在するソフトウェア設計問題解決の経験的知識を獲得・形式化するモデルと機構を開発する、(ii) スキルレベルの異なる専門家集団が、各自に割り当てられた仕事を首尾よく達成しつつ、全体としては、共通のゴールを共有・理解してプロジェクトを成功に導くための協調支援のモデルと機構を開発することが必要である [落水92-2]。

ナビゲーション支援機構は、個人のスキルレベルにあわせた知性増幅器を提供することにより問題(1)に答え、コミュニケーション支援機構は、各自が有する不完全で部分的な問題解決知識の統合を図る一つの有力な手段であり問題(2)に答える。

このような組織モデルに支援された人間集団が図2.4-3に示すようなソフトウェア開発環境モデルに以下のようにアクセスする。

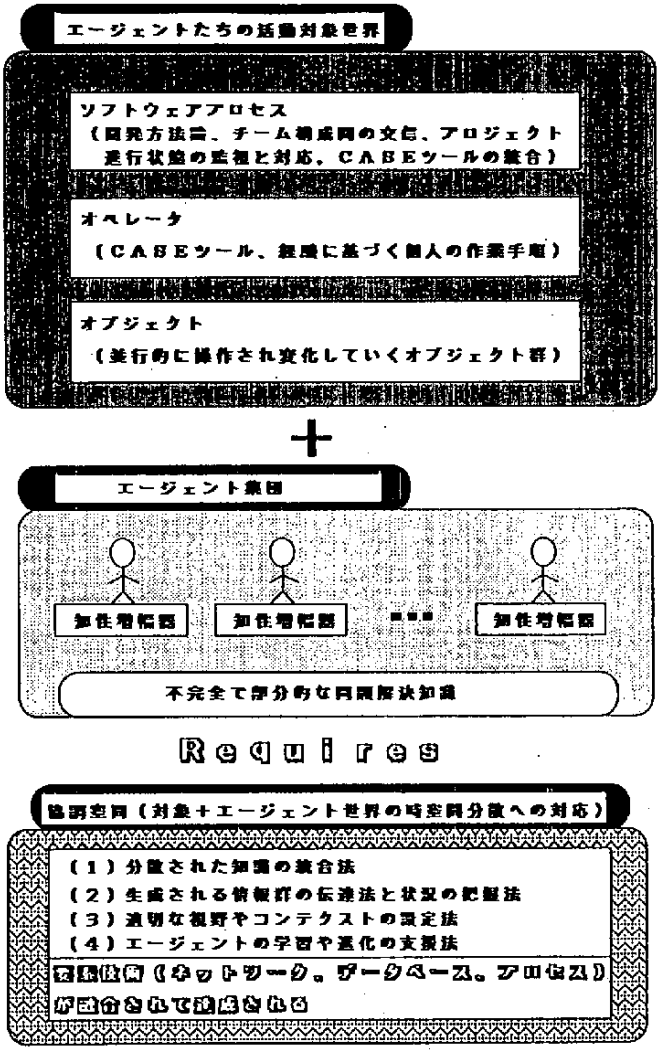


図2.4-3 開発環境と人間集団

(1) 並行的に操作され詳細化されるソフトウェアオブジェクト群を

(2) CASEツールや個人の活動により、その状態を変化させ、

(3) ツール群や各自の活動のプロセスモデルによる統合により、

- ツール統合による作業の連続性
- 開発方法論による作業の標準化
- プロジェクト管理による資源管理と危機管理

を保証されつつ操作する。

以下の項では、それぞれの問題に対する研究の進行状況をまとめる。

2.4.4 Navigation支援に関する研究の現状

ソフトウェアエンジニアリング諸活動に関する常識としては以下のものが挙げられる [Cur85]。

(1) 問題を分析しシステム・イメージを固め(対象分野の知識を統合し)つつ、システム設計へ写像する能力

(2) 新規システム特有の制約を考慮しつつ、設計問題解決の経験的知識を再利用することによって、問題を技術的解に変換する能力

(3) エラー現象と原因のつながりを発見するための戦略を状況に応じて選択する能力

経験豊富な人間の中にノウハウとして通常は存在しているこのような知識を、獲得・形式化し、普通の人間が利用することを可能にする機構の開発が本研究のねらいである。以下に研究経過と現状を示す。

① ソフトウェアプロセスに関する知識の分類

ソフトウェアプロセスに関する知識の型を、前向きの実行制御知識(仕事の進め方に関する知識)、オブジェクト知識(対象分野に依存する経験則の体系化された集合)、後向きの実行制御知識(仕事のある段階で行き詰まりが生じた時に利用される知識であり、開発経験に深く依存する知識である)に分類した[Ochi90,山口91-1]。

② 知識獲得・形式化機構の開発

前項で述べた知識の獲得・形式化を支援するプロトタイプを知識工学における事例ベース推論、モデル推論の技法を適用して開発した (図2.4-4)。

すなわち、事例ベース推論を利用して熟練者のソフトウェア設計プロセスに関する生データを収集・分類し、モデル推論の一つであるCIGOLのWオペレータとVオペレータを利用して、事例間に潜在する作業の共通性質や前提条件を抽出・整理する [山口91-2,山口92]。

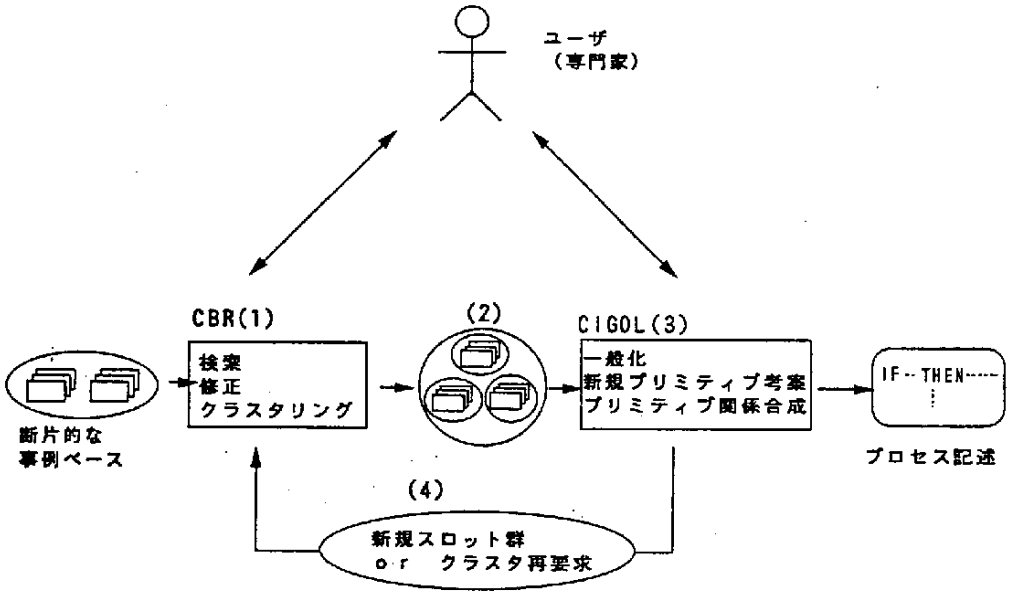


図2.4-4 Velaにおける知識獲得機構

③ 上記機構のSA/SD法への適用実験

ソフトウェア設計方法論には一般的記述が多く、例えばSA/SD法においては以下の点に、対象分野と経験に依存するノウハウを必要とする部分がある。この部分に関する知識を抽出する実験が進行中である。

- 外部入出力データの発生タイミングによる機能の分解
- データフロー図の構造上の特徴とバブル内の機能表現に基づく中央変換点の認識
- 第一次切断構造図の概略先読み評価に基づくボスの発見
- 設計品質基準による構造図の改良

現在開発された機構は知識工学技術の最新の成果をふまえて開発されたものである。機構の能力に関する評価実験を以下の観点から実施する予定である。

(1) 実際の生産現場における生データを使用して、潜在している作業ノウハウが抽出可能であるか否かを評価する。

(2) 大学にいる優秀なデザイナの行動分析をおこなう。

(3) また、知識利用機構に関してもプロトタイプの開発を進める

予定である。すなわち、知識工学におけるメタ推論、仮説推論の技法を応用して、ソフトウェアプロセスに関する形式化された知識の利用支援機構を開発する。

2.4.5 Communication 支援に関する研究の現状

分散開発環境下にあっては、協調作業のための時空間の制約が厳しく、そこを緩和する方式が明確でない。一般に協調支援機構は、データベース技術が支援する成果物の管理機構やネットワークによる情報交換機構を基に、2.4.1で述べた機能に融合することで開発可能である。最近 CSCW の技術が注目を集めているが、ソフトウェア開発固有の交信を支援するには、本項の①で述べるようにまだ十分とはいえない。

Vela においては、ソフトウェア開発に固有の交信形態を考慮した協調支援モデルの定義と支援機構の開発をめざしている。

(1) CSCW に関する技術現状のサベイ

① e-mail の効果と限界[Fel86]

一般に、e-mail は、空間的に離れていて関心が同じである人々の間に交信の場を形成することにより、ある種の空間分散に関する制約を緩和する。しかし、「ゴールを達成するために満たすべき制約をクリアしつつ中間生成物を生成していく専門家達のコミュニケーションを支援する」という観点からは、話題に応じて気楽に情報を交換できるという e-mail の効果だけでは必ずしも十分ではない。

また、プロジェクト進行中に生じる異常状態を検知し、それに対して適切な対応をとるためには、コミュニケーション支援単独ではなく、成果物管理機構やプロセス監視機構等も考慮にいれてモデル化することが必要である。

② 準構造化メッセージの効果と限界 [Mal87]

準構造化メッセージは、ある情報に対する処置が定型化しているような業務 (例えば、比較的簡単なオフィス業務) において、配送された情報(電子メール)に対して必要となる処理の自動化を支援することにより、人間の不完全さや忘れやすさをある程度補完する効果がある。しかし、ソフトウェア開発の世界では、ある人の中間成果物を次の人に渡す場合には、同時にデザイン・コンセプトを伝達する必要があるし、また伝達の度合を確認する必要もある。そのような作業は必ずしも定型化できないことから準構造化メッセージの効果も限定される。

③ ハイパーテキストの効果と限界 [Con88]

人間の思考過程の記録法としてハイパーテキストの効果を調査した。ハイパーテキストは構造的思考のツールや構造的コミュニケーションにおける情報表現媒体として有望であるが、作業内容に応じた適切な視野を設定したり、必要な情報を得るためのコンテキストを与えるという点に関しては現状ではまだ不十分であると判断した。

(2) 交信ドメインの概念

ソフトウェア設計の初期段階やデザインレビューにおいてはさまざまの検討事項が長期にわたって検討され、時間分散の問題が生じる。

これに対して、論点対応に、関係者の間に永続的な仮想リンク (交信ドメイン) を設定し、グループ内の検討内容を発言の型に応じて構造的に記録するような交信モデルを設計し (図2.4-5) 、TCP／IP のアプリケーション層の一つとしてプロトタイプを開発し、さらに、デザインレビュー支援機構を試作した [落水91-2]。

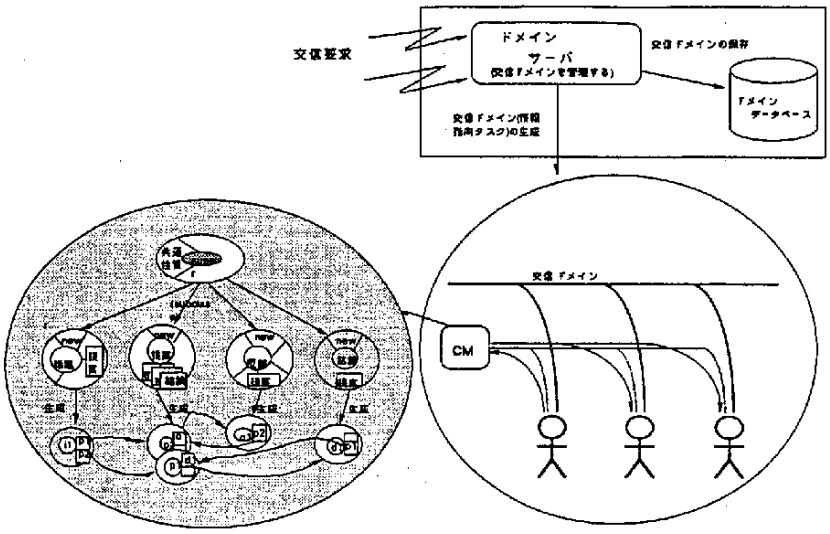


図2.4-5 交信ドメイン(論点対応の仮想リンク)

しかし、開発された機構は TCP／IP 機構の単なる抽象にすぎず、適用範囲が狭いことに不満が残り、以下にのべる方向にアプローチを変更した。

(3) 協調空間の概念の導入 [佐塚92]

ソフトウェア開発にたずさわる人々をその作業の型 (エージェントの型) によって、以下の3種に分類する。

① 部分問題解決型エージェント

情報源、中間生成物の入力先、出力先はほぼ決定されており、関連するエージェントとのコミュニケーションにより、与えられた課題を効率よく達成するための努力をはらう。

② 調整型エージェント

部分問題解決型エージェントの協調を支援する。すなわち、部分問題解決型エージェントの個々の活動や全体の状況を把握しながら、チーム作業の進行状況分析、二次的な問題の解決、異常状況の検出とその対応を行う。それに加えて、組織全体の状況を広く把握し適切な対応をとるために必要な情報源とその探索法を知っている。

③ 開放型エージェント

現在の課題を達成することより、むしろ新しい問題解決法を工夫するために、発想し、思考し、開

拓する。問題解決チーム間にまたがって情報交換を行い、また外部世界との情報のやりとりを行う。

各エージェントの相互関係を図2.4-6 に示す。

つぎに、各エージェントが必要とするデータや情報の型 (作業環境) を定義する。各型のエージェントのインターフェスにおける操作は、その型のエージェントが必要とする情報の型や生成する情報の型、その情報型に附随する操作として形式化する。

エージェント間の相互作用とは、ある作業環境から出ていく情報が別の作業環境に入る時におこる情報の蓄積と型変換の操作であると定義しこれを協調プリミティブとよぶ。現在抽出した協調プリミティブは以下の5つである。

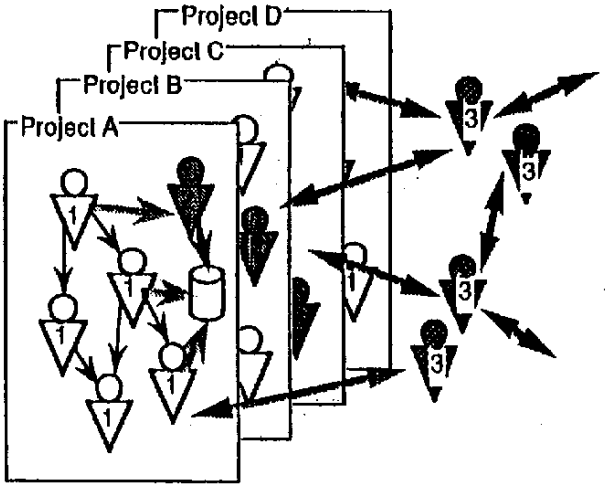


図2.4-6 各エージェントの相互関係

① 協調プリミティブ1 (生成物の蓄積と伝達)

情報の蓄積と伝達の機能を同時に実現し、プロジェクトに公開する。

② 協調プリミティブ2 (状況の収集・蓄積と検索時のフィルタリング)

状況を可能なかぎり自動的に収集し蓄積する。状況把握側は、必要な情報のみをフィルタリングして利用する。

③ 協調プリミティブ3 (計画情報の分配)

作業計画を関連するすべてのエージェントに分配する。

④ 協調プリミティブ4 (状況把握)

異常状況を可能な限り自動的に検知し、原因を推論し、対応する役目にあるエージェントに通知する。

⑤ 協調プリミティブ5 (情報パイプ)

仕事に不慣れな技術者への指導とナビゲーション。問題発生時に、情報の存在場所から必要場所に情報が迅速に流れる必要がある。

このような協調プリミティブの集合を協調空間とよび、協調支援の環境モデルとして採用する。これらの協調プリミティブを実行するエージェントが協調支援環境である。

2.4.6 おわりに

ソフトウェア開発を創造性豊かなものにするには、人間要因の考慮が重要であり、また、ソフトウェア開発環境はそれに対応できなければならない。

筆者が開発中の協調支援環境 Vela は以下の2点からその問題にアタックしている。

(1) 経験豊富な人間の中に潜在するソフトウェア設計問題解決の経験的知識を獲得・形式化するモデルと機構の開発

(2) スキルレベルの異なる専門家集団が、各自に割り当てられた仕事を首尾よく達成しつつ、全体としては、共通のゴールを共有・理解してプロジェクトを成功に導くための協調支援のモデルと機構の開発

前者に対しては、知識工学における事例ベース推論、モデル推論の技法を適用したプロトタイプが完成した時点であり、後者に関しては協調支援の環境モデルの定義が終了しプロトタイプ開発にとりかかろうとしている。

【参考文献】

[Con88] J.Conklin and M.Begeman: gIBIS: A Hypertext Tool for Exploratory Policy Discussion, CSCW'88, pp.140-152(1988).

[Cur85] B. Curtis 編:Human Factors in Software Development,IEEE Tutorial, ORDER No. 577(1985).

[Fel86] Martha S. Feldman:Constraints on Communication and Electronic Mail, CSCW'86,pp.73-90(1986).

[Mal87] T.W.Malone, K.R.Grant, F.A.Turbak, S.A.Brobst, and M.D.Cohen: Intelligent Information-sharing systems,CACM, Vol.30 No.5, pp.390-402(1987).

[Ochi90] K.Ochimizu and T.Yamaguchi:A Process Oriented Architecture with Knowledge Acquisition and Refinement Mechanisms on Software Processes, Proceedings of the 6th International SOFTWARE PROCESS WORKSHOP, pp.145-147(1990).

[落水90] 落水:ソフトウェアプロセスモデルに基づくソフトウエア開発支援環境Vela,日本ソフトウエア科学会第7回大会論文集,pp.205-208(1990).

[落水91-1] 落水:CASEとは,ソフトウェア科学会サマーチュトリアル「90年代のCASE」資料,pp.1-13 (1991).

[落水91-2] 落水:統合環境Velaにおけるデザインレビュー支援,情報処理学会,ソフトウェア工学研究会資料,ソフトウェア工学77-5(1991).

[落水92-1] 落水:ソフトウェア開発におけるグループウェアの役割,ソフトウェア・ツール・シンポジウム'92予稿集,pp.83-92(1992).

[落水92-2] 落水:ソフトウェア開発環境の動向とAI技術への期待,知能ソフトウェア工学の動向と展望シンポジウム論文集,pp.11-16(1992).

[Per88] D. E.Perry and G.E.Kaiser:Models of SoftwareDevelopment Environments,10th International Conference on Software Engineering,pp.60-68(1988).

[佐塚92] 佐塚,落水:協調作業におけるエージェント間の相互作用のモデル化,情報処理学会第44回(平成4年前期)全国大会予稿集,(1992).

[シュレ90] S.シュレイアー、SJ.メラー著,本位田、山口訳:オブジェクト指向システム分析,啓学出版(1990).

[Tul90] C.Tully:Session Summary SDE ARCHITECTURE ISSUES,6th International SOFTWARE PROCESS WORKSHOP,pp.31-36(1990).

[山口91-1] 山口、落水:Velaにおけるソフトウェアプロセスモデル構築支援環境(1)-枠組-,第43回情報処理学会全国大会講演予稿集,(1991).

[山口91-2] 山口、落水:ソフトウェアプロセスモデル構築における知識獲得と利用方式,人工知能学会研究会資料,SIG-FAI-9002-3,(1991).

[山口92] 山口、落水:事例ベース推論とモデル推論の相互作用に基づくソフトウェアプロセスモデル獲得支援環境,知能ソフトウェア工学の動向と展望シンポジウム論文集,pp.75-81(1992).

2.5 グループコミュニケーション支援技術

本節では、ＩＢＭの『リアルタイム・プレゼンテーション・システム（ＲＴＰ）』および『Person to Person／２（Ｐ２Ｐ）』を、グループコミュニケーション支援システムの事例として紹介する。ＲＴＰは試作システム、Ｐ２Ｐは製品（米国のみ）であるが、どちらのシステムも、現在手に入るハードウェアとソフトウェアを使い、現在のインフラストラクチャ（ＬＡＮおよびＩＳＤＮ基本インターフェース）で使用可能な、マルチメディアのワークステーション会議システムである。ＲＴＰとＰ２Ｐどちらも、オペレーティング・システム／２（ＯＳ／２）™のＰＭ（プレゼンテーション・マネージャー）上で動作するものであるが、特にそのアプリケーション・プログラムとの関係について考えることにより、分散開発環境での適用可能性について論ずる。

2.5.1 リアルタイム・プレゼンテーション・システム（ＲＴＰ）

ＲＴＰは、日本アイ・ビー・エム東京基礎研究所において開発された試作システムであり、遠隔地にいるパーソナルシステム／５５™のユーザー同士が、マルチメディアの会議資料を同時に共有しながら、リアルタイムの打ち合せができるシステムである。ネットワークとしてはＩＳＤＮ基本インターフェース（２Ｂ＋Ｄ、ＮＴＴ　ＩＮＳネット64）およびトークン・リングＬＡＮ（４または16メガ

ビット／秒）を使用する（システム構成については図2.5-1参照）。

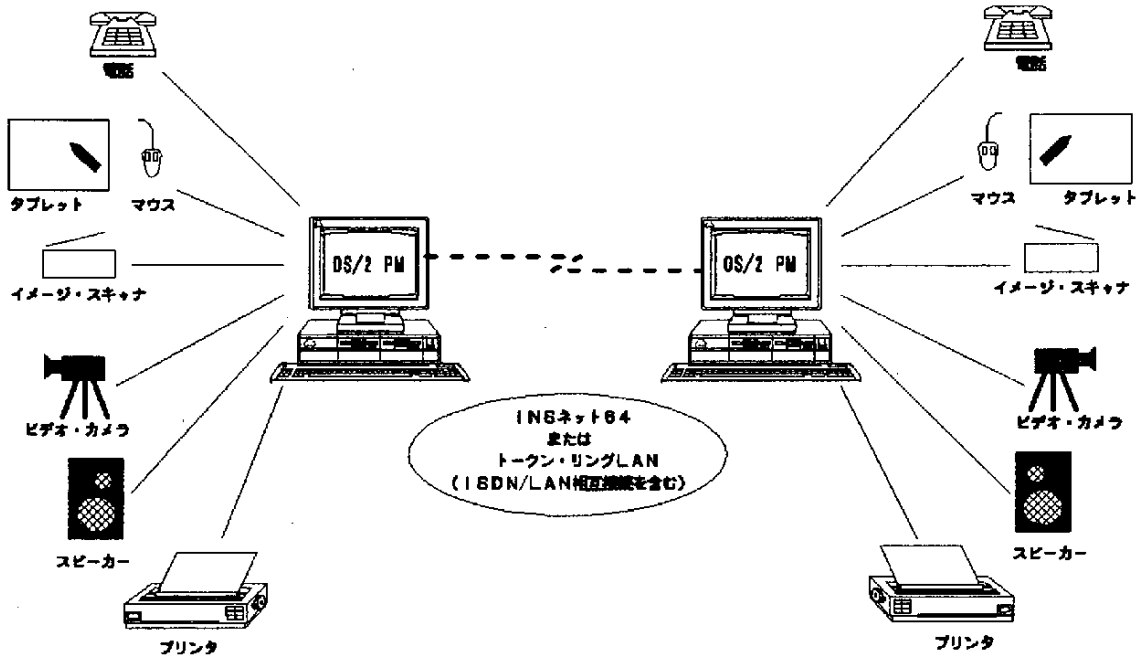


図2.5-1　ＲＴＰのシステム構成

ＲＴＰは、あたかもＯＨＰ（オーバー・ヘッド・プロジェクタ）を使ったプレゼンテーションを行うように、予め作成された『マルチメディア・フォイル』と呼ばれる会議資料を使って、遠隔地にいる相手と対話的な打ち合せを行うことができる（ＲＴＰの画面サンプルを図2.5-2に示す）。現在一般に利用できるＲＴＰ試作システムでは、二人のユーザーが、ＬＡＮあるいはＩＳＤＮを通じて一対一で使う構成のみをサポートしているので、今後この二人のユーザーを、しばしば『自分』と『相手』という表現で呼ぶことにする。Ｐ２Ｐも二人のユーザー構成のみをサポートしている。

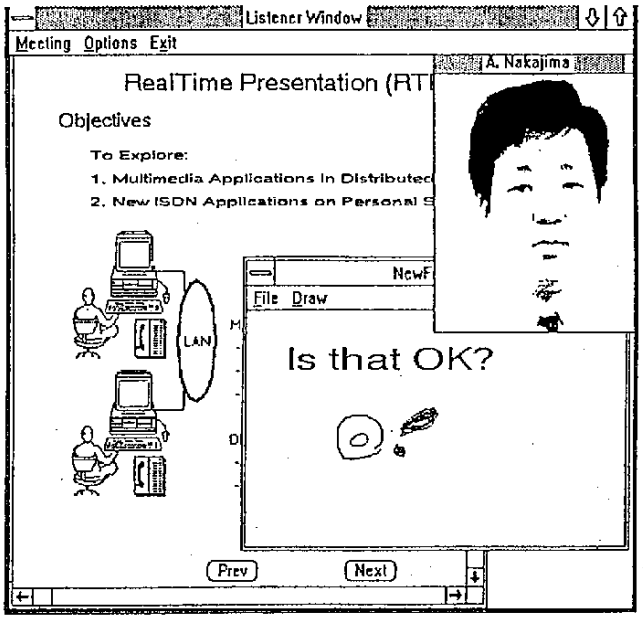


図2.5-2　ＲＴＰのサンプル画面

図2.5-2の画面サンプルで、画面の右上に表示されている相手の顔は、会議用電話帳と共に予め用意された静止画であるが、既にH.261（P×64）等のリアルタイムの動画も実験済みである。P2Pは『アクションメディアII』というボードを使って、DVI™(ディジタル・ビデオ・インタラクティブ)のRTV（リアル・タイム・ビデオ）の圧縮方法によるリアルタイム動画を実現している。

RTPのユーザーにとっては、会議資料としてのマルチメディア・フォイルを作成し会議の準備を行うモードと、作成したマルチメディア・フォイルを相手に転送した後、それを使ってRTPを実行するモードの二つのモードが存在することになる。以下にRTP実行時の主な機能について解説するが、最近の試作システムでは、会議の準備のモードは『相手のいないプレゼンテーション』と考えることにより、準備時と実行時の編集機能の差は小さくなりつつある。

(1) マルチメディア・フォイル表示機能

文字、グラフィックス、イメージ・データから成る複数ページのメタファイルを、OHPシートを順番にOHPの上に乗せていくように、相手との共有ウィンドウに順次表示していき、タブレットやマウスを使ってメタファイル上に注釈を付けることができる。図2.5-3にマルチメディア・フォイル・ブラウザのウィンドウの様子を示す。また、MAVC（マルチメディア・オーディオ・ビジュアル・コネクション）を使って取り込んだ音声と静止画を、相手と同時にプレイバックする『メッセージ・フォイル機能』も存在する。会議資料としてのマルチメディア・フォイルはデータ量が多いため、会議実行前に予め転送しておくことで、RTP実行時のデータ転送量を最小限とし、ISDN基本インターフェースの64キロビット／秒のBチャネルでもリアルタイム性を持たせることができる。この会議資料の転送はRTPのファイル転送機能を利用して行われる。

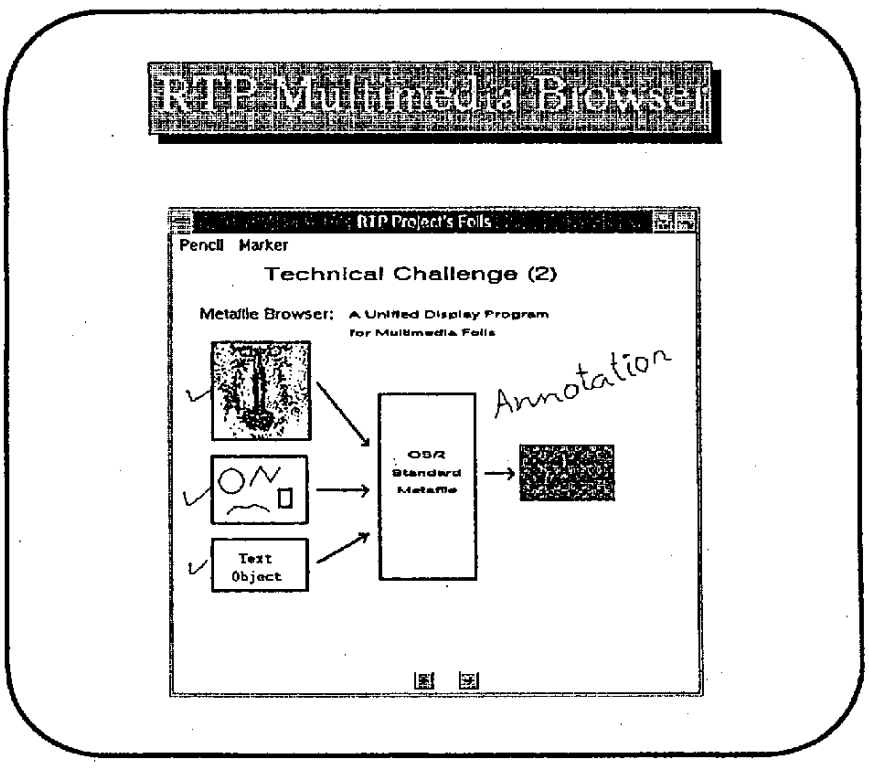


図2.5-3 マルチメディア・フォイル・ブラウザのウィンドウ

(2) リモート・ポインター

共有ウィンドウ上でのある参加者のマウスやペンの動きが、他の参加者にリアルタイムに伝わる機能で、一般的にグループウェアではテレポインティングと呼ばれている。我々はリモート・ポインターの制御方法として様々なものを検討したが、ここでは我々が考案した「疎結合（Loosely Coupled）方式』ついて紹介したい。その方式とは、自分がポインターを動かした時に、もし或るタイムアウト時間（ネットワークの遅延時間等から決められる）内に相手のポインターが動かなければ、自分がリモート・ポインターの制御権を取り（相手のポインターは自分のポインターの動きに従って動く）、もしこのタイムアウト時間内に相手も動いた場合は、自分のポインターも相手のポインターも独立に動く、というものである。図2.5-4にこの疎結合方式リモート・ポインターのメカニズムを示す。一方Ｐ２Ｐにおいては、自分と相手それぞれが別のリモート・ポインターを、『チョークボード（Chalkboard）』と呼ばれる共有ウィンドウ上で使うことができる。

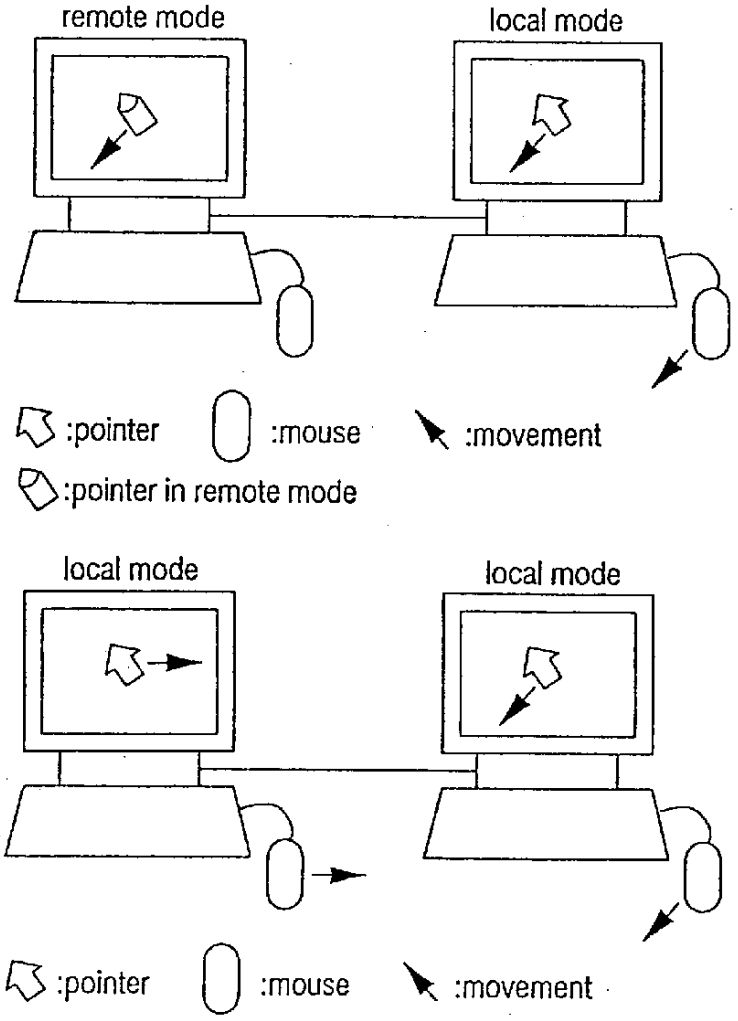


図2.5-4　疎結合方式リモート・ポインター

(3) 対話的なフォイルの作成

この機能は『ニュー・フォイル（New Foil）』と呼ばれ、会議中にＯＨＰ上のブランクのＯＨＰシートに会議内容を書き込んでいくように、ユーザー同士が打ち合わせをしながら、共有ウィンドウに文字やグラフィックス・データを対話的に書き込んでいけるものである。Ｐ２Ｐのチョークボードは、このニュー・フォイルと同様の編集機能を持ち、次に述べる静止画転送を含めて、ユーザー同士のデータの共有を統一的に行うための共有ウィンドウとなっている。

(4) 静止画の転送

ＲＴＰユーザーは、ＲＴＰ実行時に、スキャナーやカメラから取り込んだ二値あるいはカラーの静止画を相手に転送し、共有することができる。この静止画転送機能は、任意のＯＳ／２　ＰＭアプリケーションのウィンドウの内容をイメージ・データとして相手に転送できる機能（この機能は『ウィンドウ・コピー』と呼ばれている）をベースとして実現されている。静止画転送機能は大量のデータ転送を伴うので、ＩＳＤＮ、あるいはＬＡＮ同士がＩＳＤＮゲートウェイで結合されたネットワーク構成では、データの圧縮／伸張を行っている。

2.5.2 Ｐｅｒｓｏｎ　ｔｏ　Ｐｅｒｓｏｎ／２

米国ＩＢＭで1991年10月に発表された製品（顧客独自の必要性から開発された特殊製品）であり、パーソナルシステム／２™のユーザー同士が、トークン・リングＬＡＮ、イーサネット、またはＩＳＤＮ基本インターフェースを通じてリアルタイムの打ち合せができるシステムである。図2.5-5にＰ２Ｐのシステム構成図を示す。主な機能としては、上で言及した『チョークボード』と呼ばれる共有ウィンドウと、ＤＶＩの動画を利用した『ビデオ・ウィンドウ』があり、以下により詳細に解説したい。Ｐ２Ｐは、さらに『チャットライン（Chatline）』と呼ばれるテキスト・メッセージの交換機能、ファイル転送機能などを持っている。リアルタイムの音声については、構内ならばＰＢＸ、遠隔ならばデータ通信に使われるＩＳＤＮとは別の電話回線を利用してやり取りする。

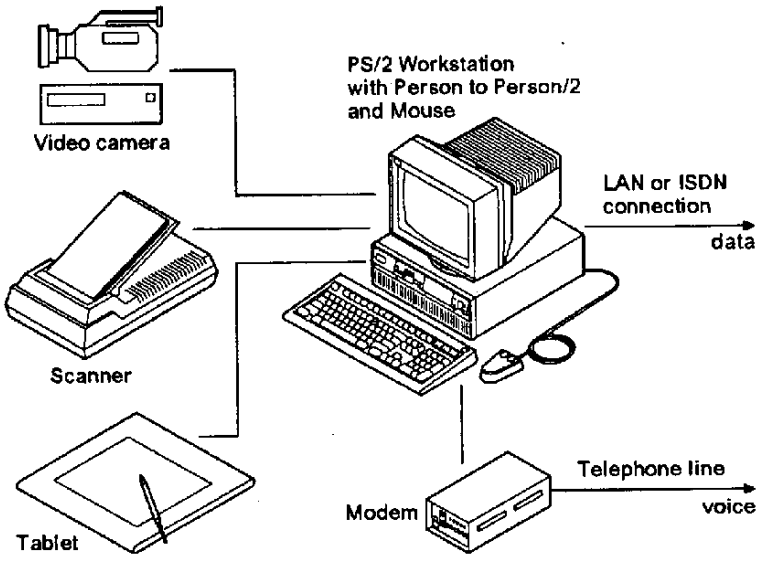


図2.5-5　ＰＴＰ／２のシステム構成

(1) チョークボード

チョークボードは、ＯＳ／２　ＰＭのウィンドウ上で、一般的にグループウェアで「共有黒板」と呼ばれているような機能を提供する。ユーザーは複数個のチョークボード・ウィンドウを画面上に開くことができ、それらを相手のユーザーと共有化することにより、遠隔地でネットワークを介した打ち合わせができる。図2.5-6にチョークボードのウィンドウと文字やグラフィックスのデータをチョークボードに書き込むためのパレットを示す。それぞれのチョークボードは、「フォアグラウンド」と「バックグラウンド」の二つの層が重なったもので、ユーザーは、バックグラウンドにはスキャナー、ファイル、他のＰＭウィンドウからイメージ・データをロードすることができ、フォアグラウンドに文字やグラフィックスのデータを上書きすることができる。作成されたチョークボード上のデータは、イメージ・データとして保管することができる。またリモート・ポインターをチョークボード上で利用できる。さて、他のＰＭウィンドウのビット・イメージをロードする機能は、「ミラー（Mirror）」と呼ばれており、これによって相手のユーザーと任意のＰＭアプリケーション・プログラムを、たとえ相手がそのアプリケーションを持っていなくても、ウィンドウのビット・イメージの形で共有することができる。あるチョークボードからミラーとして一度選択されたウィンドウは、そのチョークボードと関連付けられ、そのウィンドウが更新された時には、ウィンドウのタイトル・バーにあるアイコンをクリックするだけで、チョークボードには新しいイメージがロードされる。

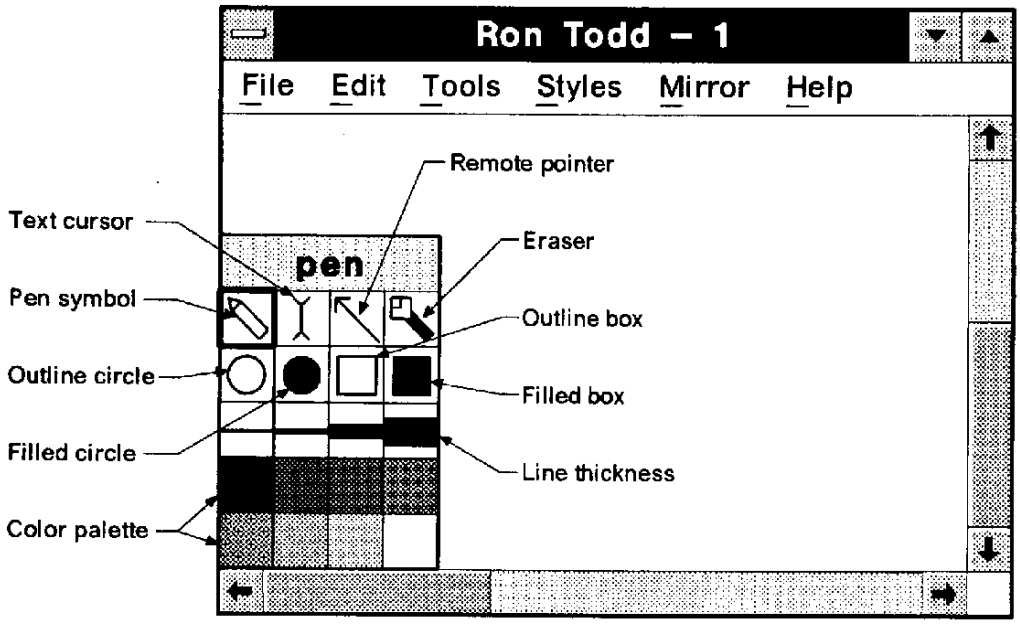


図2.5-6　チョークボード

(2) ビデオ・ウィンドウ

既に述べたように、Ｐ２Ｐは、ＤＶＩ（ディジタル・ビデオ・インタラクティブ）のＲＴＶ（リアル・タイム・ビデオ）の圧縮方法によるリアルタイム動画を実現している。この動画を表示するＰＭのウィンドウをビデオ・ウィンドウと呼び、自分の画像と相手の画像の両方、あるいはどちらか一方を表示することができる。画像の解像度は128 ×120 でカラーのビット数は９ビット／ピクセルで、相手画像の一秒あたりのフレーム数は（ネットワークのロードによって変化するので、最高のフレーム数は）、ＬＡＮの場合はおよそ15、ＩＳＤＮ基本インターフェースの場合にはおよそ１である。Ｐ２Ｐは、画像取込みオプション付きのアクションメディアII画像再生アダプターと呼ばれるハードウェア・ボードを使ってこの機能を実現している。Ｐ２Ｐシステムとしては、このビデオ・ウィンドウ機能はオプションである。

2.5.3 分散開発環境での適用可能性

これまでＲＴＰ及びＰ２ＰというＯＳ／２上のワークステーション会議システムの事例について解説してきた。これらのシステムは、コンピュータ・システムを通じて人間同士のコミュニケーションを支援する事ができるが、単に電話、ファックス、テレビ電話、ＯＨＰ、黒板などの機能を電子的に組み合わせただけでなく、ワークステーションのアプリケーション・プログラムとどのように組み合わせて使えるかという点が重要であろう。本項では、ＲＴＰまたはＰ２Ｐとアプリケーション・プログラムとの関係について考察し、実際にアプリケーションとしてプログラム開発環境を取り上げた時、その分散開発環境での適用可能性について議論してみたい。もちろんシステム構成としては、現在の二者で一対一の構成ではなく、将来複数ユーザーがサポート可能となり、分散している複数の会議参加者（プログラム開発者）が、Ｐ２Ｐのチョークボードのような共有ウィンドウを持てるという前提で考えてみたい。

ＲＴＰおよびＰ２Ｐと他のＯＳ／２　ＰＭアプリケーション・プログラムとのインターフェースは、それぞれ『ウィンドウ・コピー』および『ミラー』機能であり、任意のアプリケーション・プログラムがＰＭのウィンドウに表示しているデータを、ビット・イメージとして共有ウィンドウ上に取り込み、打ち合わせに利用することができる。このインターフェースは、個々のアプリケーション・プログラムのセマンティクスとは独立であるので、ＲＴＰやＰ２Ｐの存在を知らない任意のアプリケーションとインターフェースをとることができるという点で非常に便利である。したがって、アプリケーションとしてプログラム開発用のツール（ＣＡＳＥツール、デバッガ、ワープロ、エディタなど）を考えた時、そのウィンドウ・イメージをプログラム開発者同士で共有し、それについて話し合うことができるであろう（例えば、デバッガの画面を見ながらプログラムのバグについて議論する等々）。

しかし、このインターフェースは、ウィンドウのビット・イメージという、アプリケーションから

みて低いレベルのものなので、より効率的なデータの転送やデータの再利用（取り込んだデータを変更してツールに入力すること）などが要求される場合には、別の方法を考える必要がある。それでは次に、ファイルのレベルで、共有ウィンドウがアプリケーション（プログラム開発用ツール）とインターフェースする場合を考えてみたい。例としてワープロを考えてみると、ユーザーは、ワープロで作成した文書ファイルを共有ウィンドウに読み込むことによって相手に転送し、その共有ウィンドウに表示された文書を相手と同時に見ながら、打ち合わせをすることができることになる(図2.5-7参照)。

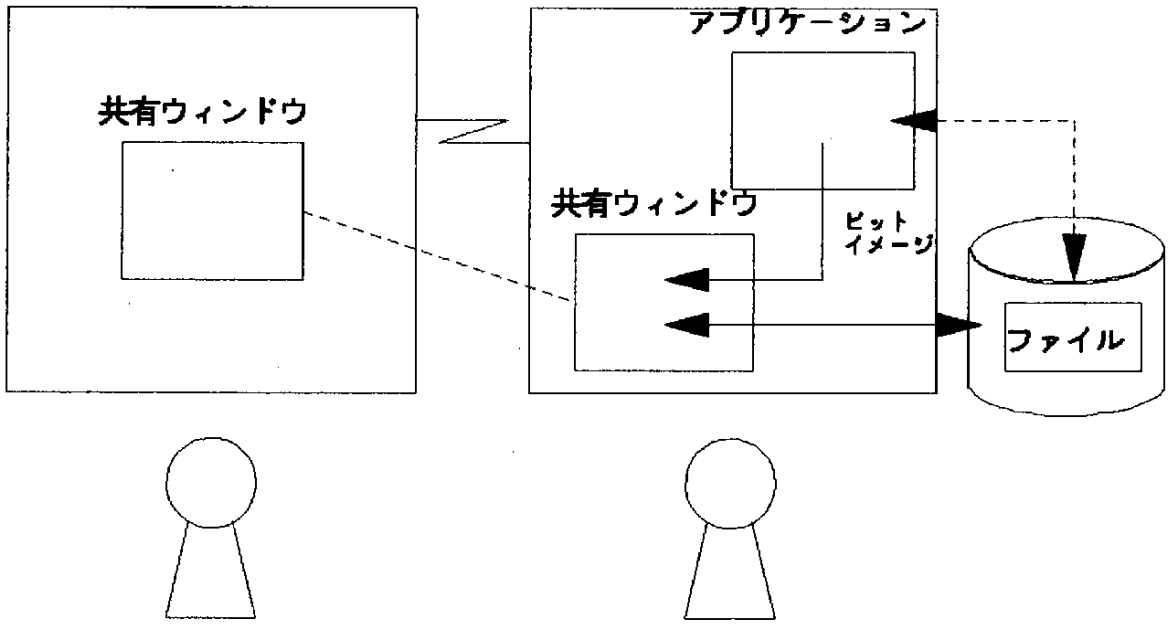


図2.5-7　共有ウインドウとアプリケーションとのインターフェース

この場合、文書ファイルのみ転送すればよいので、データの転送時間は短くなるが、共有ウィンドウは、ワープロの文書ファイルのフォーマットと、その文書の表示方法を理解している必要があり、共有ウィンドウ開発のワークロードは大きくなる。この方式は、特定のツールに特化した会議システムに有効なアプローチと考えられるが、共有ウィンドウをそのツールに対応させる方法と、それともツールそのものを共同作業用に書き換える方法のどちらを取るかを、まず議論するべきであろう。

今までは、会議参加者であるプログラム開発者が、Ｐ２Ｐのチョークボードのような共有ウィンドウを通じて打ち合わせをするという前提であったが、プログラム開発用ツールとしてのアプリケーションそのものを、どうしたら共同作業用に変更できるか、という点について考えてみたい。我々は既存のアプリケーションを変更することなく、そのまま共同作業に使えるようにするためのメカニズム開発を何度か行ってきたが、ＯＳそのものに共同作業用の仕組み（ウィンドウ、ポインター、ファイルなど必要なシステム資源をネットワークを通してマルチユーザー環境で共有できること）が組み込

まれていない場合、サポートできるアプリケーションの範囲、システムのバージョン・アップへの対応、ローカルとリモートの環境の相違への対応、ローカルとリモートでのオペレーションの同期など様々な問題を解決しなければならないことがわかった。

今後は、ＯＳを含めて共同作業用のシステム・プラットフォームを開発し、アプリケーション自体を変更するほうが、ユーザー・インターフェース等も含めて最適な機能を実現できるであろうが、その際にアプリケーションをどのように想定し、アプリケーションとの依存性をどの程度のものと考えるかが重要な設計ポイントとなるであろうと思われる。また、プログラム開発者の共同作業によって大量のマルチメディアの文書が作成されることなるので、ネットワーク環境においてこれらの文書を管理できると同時に、高速なアクセスや検索ができるような、データベースやセキュリティ・システムを包含したシステムを構築する必要があろう。

2.6　コンピュータによるグループワーク支援の可能性と利用実験

(財)日本情報処理開発協会では、ソフトウェア開発時における各種グループ作業をコンピュータを用いて支援するシステムの研究を進めている。そのために、既存のグループウェアあるいはCSCWで行われている研究内容を調べると共にLANを中心としたプロトタイプ環境を開発し、グループ協調作業の障害、問題点等の検討を行った。以下にその内容の一部を示す。

2.6.1　コンピュータを用いたグループワーク支援の可能性

大規模ソフトウェア開発は、一般に要求仕様の確定、設計、製造と手順を踏んで行われる。また、各段階では、多くの開発要員が相互に連絡を取り、協調し、中間成果物を生成しながら開発を進めていく。開発工程が厳密に定義できず、また、中間成果物にしてもあいまいさが残る現在の技術水準においては、それらを埋めるための要員間の協調作業は極めて重要なものである。しかるに、現状においてはこうした作業は、ほとんど個人の能力に依存し、有力な支援環境が構築されていない。ここで開発作業の種類を考えると、個人が自立的に進めることの出来る作業と、グループで協同して行う作業がある。図2.6-2に個人的作業を詳細化したものを、また図2.6-3にグループ作業を詳細化したものを示す。

実際の作業は、これらが組み合わせられて進行することにる。例えば、モジュール設計と言う作業を例にとると、各作業者は与えられたインタフェースに基づいてモジュール内の詳細化を行うが、その際必要に応じて疑問点の確認、レビュー等を行う。

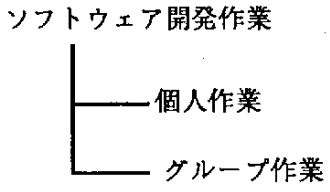


図2.6-1　ソフトウェア開発作業の種類

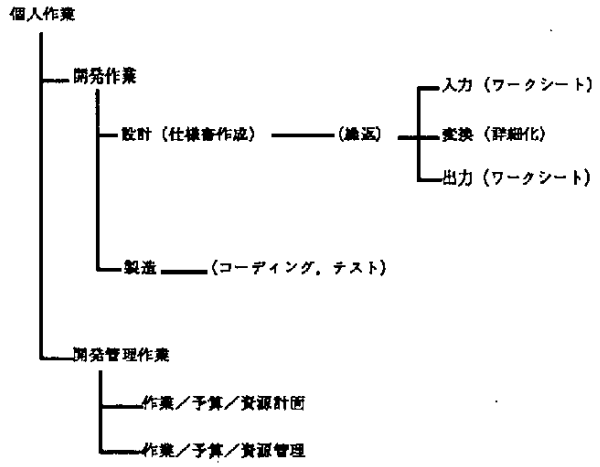


図2.6-2　個人作業の構成

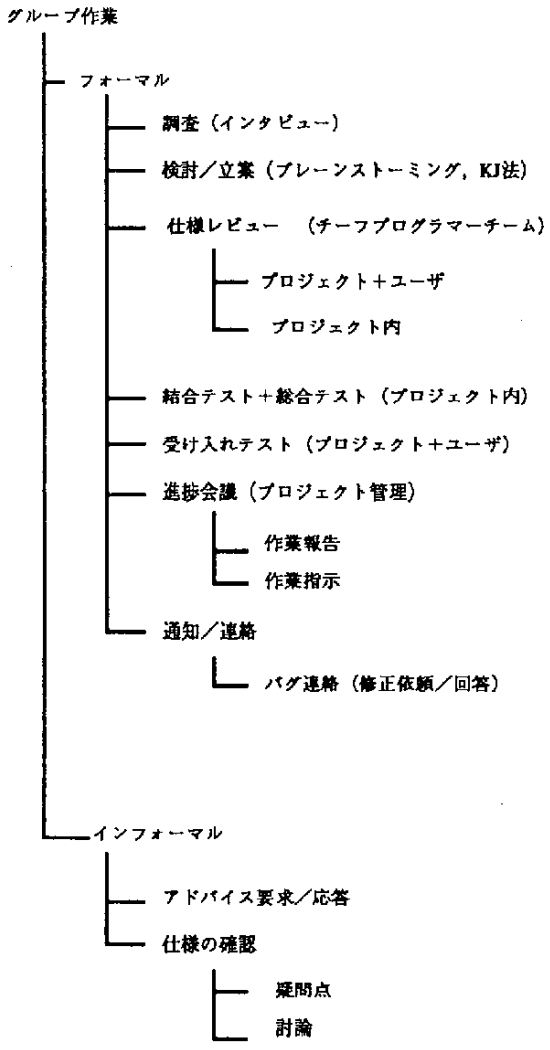


図2.6-3　グループ作業の構成

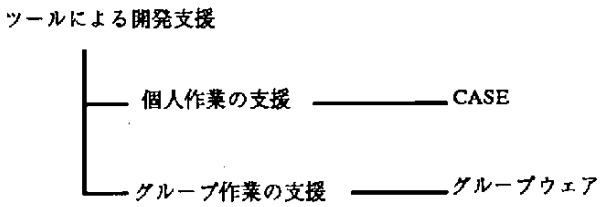


図2.6-4　ツールによる開発支援

近年、ソフトウェア開発を支援するためのツールとして、各種ツールが考案されCASEとして成長してきている。しかし、現在のところ多くのCASEツールは、データフロー図の作成、アクションダイヤグラムの作成等個人的に行える作業の支援に留まっている(図2.6-4)。

ここで、開発時におけるグループ作業がどのように行われているかについて考えてる。従来の開発作業においては、一ケ所にメンバーを集中させた対面型のグループ作業が圧倒的でしかも支援ツールはほとんど無いに等しい状態であった。

しかし、CSCWやCHIの会議では、グループ内での協調作業を円滑かつ効率良く進めるための多くの試作ツールやその利用経験が報告されている。これらをうまく応用するか、あるいはカスタマイズする事によって開発作業におけるグループ協調作業のツールによる支援が期待できる。以下にソフトウェア開発に関係の深い研究事例を挙げる。

(1) Information Lens

MITのT. Maloneらは、Information Lens [Malone87]と呼ばれる準構造化電子メールシステムの開発を行った。

電子メールは、ネットワーク上のユーザが最も簡便に通知、要求、了解等のメッセージを伝え合うことのできるツールである。

Maloneらは、この電子メールを構造化することによりコンピュータによる知的な支援ができるのではないかと考えた。ここで構造化とは、自然文テキストイメージをいくつかのフィールドから成るメッセージにすることである。準構造化とは、すべての内容を完全にフィールド化するのではなく、可能な部分のみをフィールド化するという意味である。こうすることにより送信ユーザは、いくつかのフィールドから成るテンプレートを用いて穴埋的にメッセージを作る事ができる。また、受信ユーザはフィールドの値を条件に、メッセージの選択、移動等ができるようになる。更に受信メッセージに応じて対応する返信メッセージの種類を示したり、自動的にアクションを起動できる。

例えば、Action Request(アクション要求)というメッセージに対しては、Commitment(了解)、Request for information(情報要求)、Action request(アクション要求)、Message(通知)が返信メッセージタイプとして提示される。

アクションの起動では、例えばミーティングの通知が来たならば、カレンダーに登録というアクションを取ることができるようになっている。

Information Lensでは、メッセージの構造化やメッセージの自動処理指定をユーザが自分で行えるようになっている。自動処理は、IF-THEN形式のルールとして指定できる。こうしてユーザは目的に合わせて構造化電子メールシステムを構築できる。

このように電子メールは、ユーザ間のメッセージ伝達媒体でありメッセージを構造化することにより何らかの協調支援の可能性が期待できる。

Maloneらは、こうした準構造化電子メールシステムの可能性を示すためにいくつかの応用システムを開発した。

ソフトウェア開発の観点からは、次に示すソフトウェアメンテナンス処理用の作業者割付システムを示している。

① ユーザは、作業割付者にProblem report(障害報告)の準構造化メッセージを送る。

② このメッセージに対してシステムは、返信メッセージのオプションとしてAcknowledge(了解)が取れるようにする。作業割付者が、了解を選ぶと障害報告の内容が了解メッセージに転記される。

③ 了解メッセージが送信されると障害報告は了解済みとなり、次のオプションはAssign(割付)になる。割付を選択すると割付メッセージが作成される。また、割付メッセージの送信先には、可能な作業者の名前が選択できるようになる。

このシステムは、ユーザ間でメッセージを構造化するが、各ユーザは自立的にメッセージ制御をすることができる点に特長がある。なお、Maloneらは、Information Lensシステムを拡張したObject Lensシステムの開発を進めている[Lai88]。

(2) ICICLE

BellcoreのV. Sembugamoorthyらは、ICICLE（Intelligent Code Inspection in a C Language Environment）[Sembugamoorthy90] と呼ばれるプログラムコード・インスペクションのための知的環境を開発している。コードインスペクションはプログラムコードの誤り、コード作成基準に従っているか否かのチェック、設計仕様とのチェックを行うことによりコードの品質を向上させることである。コードインスペクションは①ソースコード、関連設計書等のリソースの準備、インスペクターの割当等の事前準備、②コードに対するコメント作成、③コードインスペクション会議の開催、④コード作成者への修正指示という手順で行われる。この中で仕事量の多いのは②、③の作業であり、ICICLEもそこに焦点を当てている。

コードに対するコメント作成は、コードを表示するコードウインドウとコメントを作成するコメントウインドウを用いて行うことができる。コメントウインドウはコードウインドウに表示されているコードをクリックすることにより開くことができる。コメントを付けられた行には印が付けられコメントの有無、種類が分かるようになっている。

こうしてコメントが各インスペクターにより付けられると、次はインスペクター会議となる。会議ではメンバーがネットワークで結ばれたワークステーションを用いて各コメントのチェックを行う。ここで会議の参加者には、調整役、リーダ、書記の役割が割り振られる。調整役のみがインスペクション用のウインドウをコントロールできる。これは、ウインドウスクロールの混乱を避けるためである。コメントの内容が正当であれば、書記がそのコメントを正式なものとして採用する。この部分は従来であるとかなり大変な作業であったが単に採用するか否かのアクションのみで済む。なお、会議は、現在直接対面形式で行っているがオーデオ／ビデオリンクを用いた遠隔システムの検討を行っている。

ICICLEは、こうした機能の他にインスペクション作業の効率化のためにルールベースの静的デバッガ、ハイパーテキストベースの知識ブラウザ等を有している。

協調作業環境の開発に当たっては、個人作業とグループ作業のスムースな遷移が重要な課題となっている。その意味でもICICLEの統合化環境は注目される。

(3) gIBIS

MCCのE. J. Conklinらは、gIBISと呼ばれるシステム設計の過程を記録するツールの開発を行った[Conklin88]。このツールの中心となる理論はHorst Rittelが開発したIBIS(Issue Based Information Systems)である。この理論は、議論を問題 (Issue)と、それに対する見解 (Position)、更に見解に対する賛成か、

反対かの意見(Argument)という非常に簡単な要素で表現する。

設計過程における問題とは、例えばCPUを何にするかといったものであり、それに対して複数の見解を示す。見解とは、A社のCPUにする、B社のCPUにする等である。また、意見はそれぞれの見解に対する賛成、反対の意見表明である。

Conklinは、itIBIS、gIBISを実際のソフトウェア開発に適用し、次のような結果を得ている[Yakemovic90]。

① 記述の品質が一定に保たれた。

② 当初、獲得された情報はメンテナンスで利用されるだろうと考えていたが、プロジェクトの初期にも役立った。例えば開発担当者の退職等。

③ IBISは、従来の設計ドキュメントと違った見方を提供してくれた。そのため違った視点からのレビューができた。

④ IBISは、会議のアジェンダとしても利用できた。何が問題で、事前にどこまで議論されたか把握できた。

⑤ 要求分析は、ソフトウェア部門、マーケティング部門、製品管理部門等の部門間のコミュニケーションが非常に重要になる。IBISは、コミュニケーションの良い土台を与えた。

⑥ 過去の議論の経過が分かるため新しい要員の加入が容易になった。

⑦ プロジェクト管理の観点からどのような未解決の問題が残っているかを発見するに役立った。

地域分散した開発環境では、設計部門と製造部門が分かれる場合がある。こうした場合に、設計の根拠を構造化して伝達できるならば設計者の意図がスムースに製造者に伝わり、品質、生産性の向上につながると考えられる。

(4) その他

この他にも日本電気C＆Cシステム研究所では、同所で開発したマルチメディア在席会議システム（MERMAID）を用いたソフトウェアの分散開発の実験を進めている。また東芝システム・ソフトウェア技術研究所では、電子メールエージェントシステムを開発し分散開発への適用実験を進めている。

このように協調支援技術、とりわけソフトウェア開発作業の協調支援技術は、研究が始まったばかりである。しかし、ソフトウェア分散開発の動きと合わせて今後益々研究が活発になると予想される。また、CASE環境では、プロセスモデル、プロダクトモデルを中心に研究がされてきたが、今後は更に第3のモデルとして協調モデルが研究されるであろう。

2.6.2 グループワーク支援システム環境の構築

(1) ハードウェア環境

プロジェクトのメンバーが協調的に協同作業を行うために必要な技術的、社会的情報を得るために、コミュニケーション環境の構築を行い実証的に検討を行うことにした。協同作業を行うためには、意思の伝達、成果物の交換等のためのコミニュケーションチャネルが必要になる。このチャネルは、会議室のような空間であったり電話等の物理的媒体であったりする。

一般に、我々は紙等に書かれた資料を基に、相手の意見を聞いたり、述べたりして作業の合意を得ていく。また、直接の対面が困難な状況では、手紙や電話によるやり取りも行われる。

我々は①分散開発環境においては直接対面会議を頻繁に設けることができない、②紙を媒体とした場合は、安価であるがCASE環境と連係がとれない、③電話は即時性があるが記録が可視的でない等の理由により、コミュニケーションチャネルとして、LANを中心としたネットワークを用いることにした。

ただし、現状のLANは転送容量が少なくまた転送スピードも遅いので、LANはデータの転送のみに用いることとし、LANの他に電話回線を用いた音声の通信チャネルを用いることにした。また、同様な理由で、動画像についてはビデオ回線を用いることにした。ただし、机等の作業空間の制限から、データと動画像は同一の画面上にオーバーレイウインドウの形で表示できるようにした。

データ入力に関しては、文字データを入力するためのキーボードの他に、図、絵等の入力を考えタブレットを用意した。

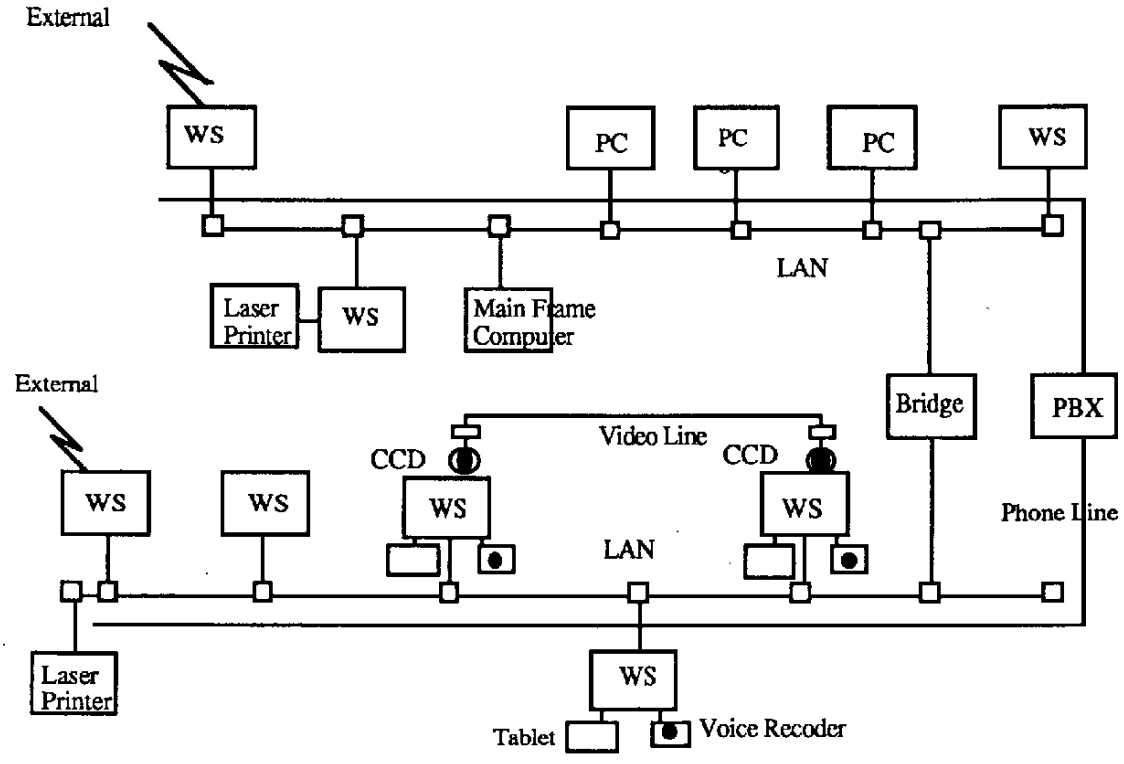


図2.6-5　実験用ハードウェア環境

(2)　ソフトウェア環境

グループ内における協同作業には、会議のように各メンバーが同一のものを見たり聞きながらリアルタイムで進めるものと、手紙等をやり取りしながら非同期に進めるものとがある。また、作業には全員が一箇所に集まって行うものと、遠隔地に各メンバーが分散した状態で行うものがある。したがって、コンピュータによるグループワーク支援はこれらの様々な形態を支援する必要がある。

一方、グループワーク作業には、伝達、連絡、サービス要求、合意形成、調整等様々なものが考えられる。ソフトウェア開発といった作業を取り上げると、仕様書のレビュー、コードのレビューが典型的なグループによる協同作業となる。

そこで、レビュー作業に役立つと考えられるツールを捜し、先のハードウェア環境で試用してみることにした。この目的はメンバーが分散しているソフトウェア開発環境における協同作業、とりわけレビュー作用を支援するツールを開発するための要求仕様を固めるためである。次にそれらのツールの概要を示す。いずれもMacintosh上で稼働するツールである。

(a)　リアルタイム型ツール

①　Timbukts

Timbukts™（Farallon）は画面全体の共有を可能にさせるツールである。このツールを用いることにより、あるメンバーの画面を他の複数のメンバーが自分の画面上で参照したり変更できる。この場合、自分の画面内の一つのウインドウに他のメンバーの画面が表示される。したがって、複数のメンバーの画面を自分の画面内に表示できる。

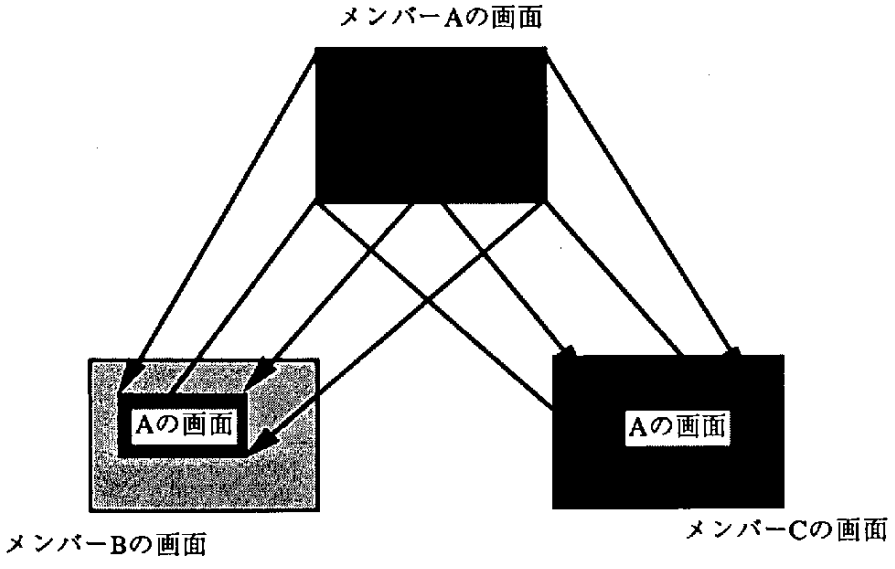


図2.6-6　Timbuktsによる画面共有

アクセス制御に関しては、参照のみ可、参照／変更とも可のオプションを持っており、各オプションにはさらにパスワードが設定できる。ただし、フロアーパッシングの制御は持っていない。したがって、フロアーパッシングの制御は、電話等の手段により行う必要がある。

この画面共有ツールを用いたグループによる遠隔分散レビュー作業は次のようになる。まず、レビューを受けるメンバーは設計仕様書等を自分の画面上に表示する。次にレビューするメンバーはTimbuktsを用いてその内容を自分の画面に表示する。その後、レビュー作業の調整者がレビューを進行させていく。

こうした画面共有システムの良い点は、既存のアプリケーションがそのまま使えるというところにある。

② Aspects

Aspects™（Group Technologies）は、リアルタイム型の協同執筆／編集ツールである。このツールは、ドキュメント作成のアプリケーションとしてライティングの機能、ドローイングの機能、ペインティングの機能を持っている。

このツールでドキュメントのレビューを行う手順は次のようになる。まずドキュメントを作成した担当者は、レビュー会議の調整者にファイル転送ツールを用いてドキュメントを送る。調整者は、このドキュメントに対する会議を設定し、Aspectsの会議パネルに登録する。また、会議参加の方法を指定する。参加には、調整者の承認を求める方法と、パスワードを要求する方法とがある。その後、調整者はドキュメントをAspects内のアプリケーションに取り込みレビュー会議を進める。各メンバーは、次々に会議に参加しリアルタイムでレビューを行うことになる。フロアーパッシングの方法は、フロアーパッシング無し、持ち回り、調整者によるコントロールの3種類がある。

Aspectsには、テレポインティングの機能があるため、各メンバーは各自のポインターを持つことができる。

(b) 非リアルタイム型のツール

① PREP

PREPは、カーネギーメロン大学で開発が進められている共同執筆及びコメント作成用のエディタである [Neuwirth90]。このツールの開発の大きな目的は、外部コメンテータが共著者に対する役割の調査にある。したがって、カーネギーメロン大学では、フィールドでの利用を進めながらプロトタイプ的に開発を進めている。図2.6-7にこのツールの利用例を示す。まず左端は、原稿に対する計画である。この計画と原稿を見ながらコメンテータはコメントを付ける事ができる。２番目のカラムは原稿であり、３番目のカラムはこの原稿に対して付けられたコメントである。複数のコメンテータがいる場合は、さらにその右にコメントカラムを追加していくことができる。また、コメントは、原稿の他にコメントに対するコメントといったやり方も行える。コメント間の関係はハイパーテキストのようにリ

ンク付けられ、そのリンクを実線で見ることもできる。

このツールは非リアルタイム型であるので、原稿作成者はPREPで作成したファイルをコメンテータに転送し、コメンテータはそのファイルに対してコメントを付け返すといったやり取りとなる。

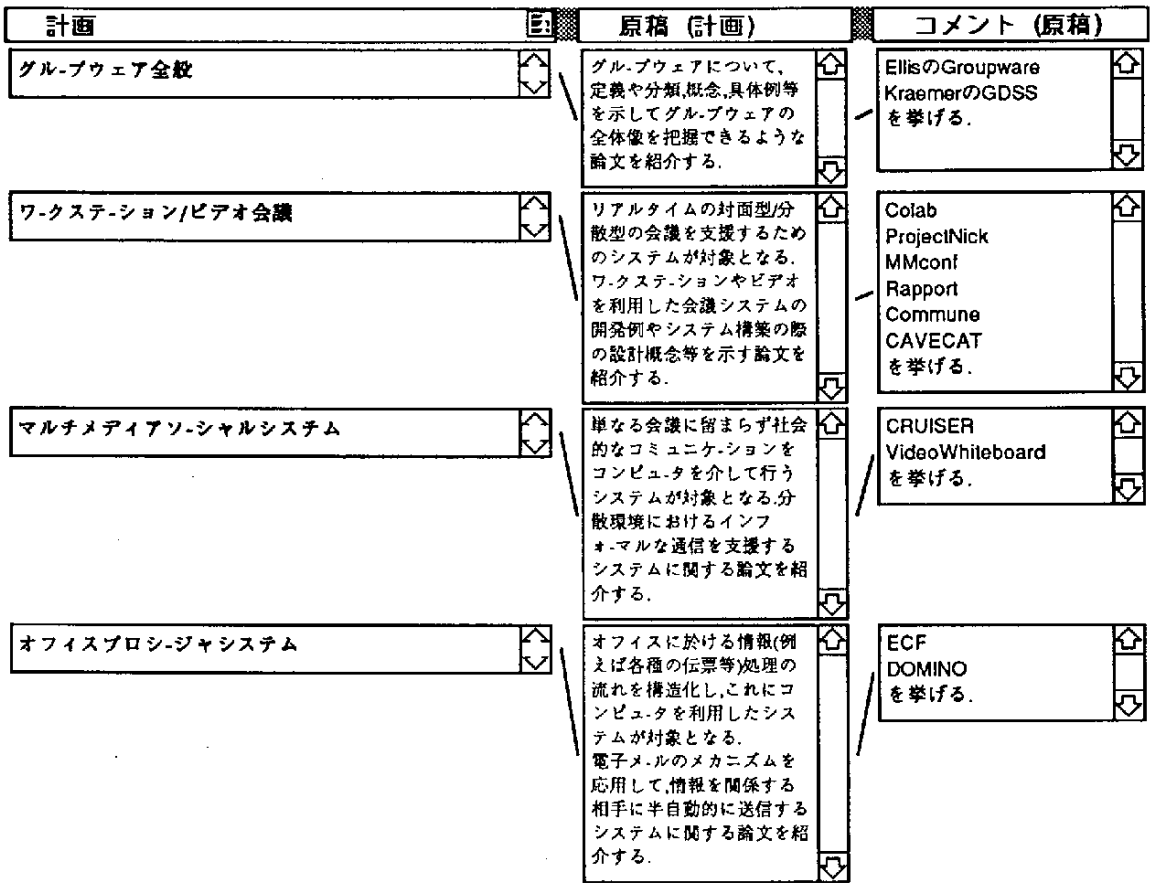


図2.6-7　PREPを利用したコメント作成

(b)　QuickMail

QuickMail™（CE SOFTWARE）は、サーバ／クライアント型の電子メールシステムである。この電子メールの特長としては、メール本文と共に複数のファイルを添付して送ることができることと、ボイスレコーダから音声を入力して送る事ができる等がある。

協同レビュー作業では、本文中にレビュー作成の期限、注意事項等を記入し、PREPのファイルを添付してメンバーに送るという使い方を行った。

2.6.3　利用上の問題点と解決策

先に述べたハードウェア／ソフトウェア環境を用いて協同レビュー作業の実験を行った。ここでは、その中から、画面共有ツールを用いたリアルタイムレビュー作業について述べる。

(1) ハードウェア／ソフトウェアの評価

① デスクトップの表示

- ポインターで場所が正確に指せる。
- ポインターがふらふらして目障りである。
- 目が疲れる。
- 会議室のOHPに比べて見やすい。
- 文字の拡大縮小がその場でてき、効果的である。
- 複数ウインドウを同時に見られる大きさが欲しい（→ 予め発表前に、画面の大きさ内で効果的に表示できるウインドウの大きさを設計しておくことが必要である）。
- イメージデータの表示に時間がかかる。
- カラーで表示したい（→ 利用した画面共用ソフトウェアがカラーをサポートしていない）。
- 網かけが複数できると良い（→ ドラッグすることにより特定の文章の表示を反転できるが、同時には一箇所のみしかできない）。
- 以前の表示内容に戻るために場所を探すのが大変（→ 紙又はOHPでは、すぐに探せる）。
- スクロールが遅い。

② 音声（電話）の利用

- 音声は必須である（→ フロアーパッシングはすべて音声で制御を行った）。
- レシーバー（受話器）を耳にあて続けなければならないため、手や耳が疲れる。

(2) オフィスの作業環境との調和

- 電話がくると応対せざるをえない（→ 通常の会議であれば、会議室に行くため緊急でない限り電話は来ない）。
- 他の人からの割り込みがある（→ 会議をしていることが外見からは分からないので、用事のある人が席に来てしまう）。

(3) 個人の作業環境との調和

- 必要な資料を取り出せる。
- 本やファイルを取り出して見る事ができる（→ 会議室の場合、取りに戻る必要がある）。

(4) 雰囲気

- リラックスして相手の話を聞ける（→ お茶などを飲みながら聞ける）。
- 上下関係の意識が少なくなる（→ あくびをしても気にならない）。

(5) その他

- 共有している文書ファイル等を、前に遡って個人的に見ることができると良い（→ 紙で事前に配付されている場合は、個人的に見ることができる）。

－　簡易で精密な作図ツールがあると良い（→　黒板にチョークで書くような感じで使えるものが欲しい）。

－　画面の内容がすべて相手に見えてしまうことは都合が悪い場合もある。

2.6.4　今後の課題

ソフトウェア開発上で最も高度で緻密な協同作業であるレビュー作業を取り上げ、ネットワーク環境での実現方法について、プロトタイプ環境の開発を進めながら検討してきた。

その結果、分散環境において各メンバーが分散されたままでもネットワーク技術を用いて、十分にレビュー作業が行えるとの確心を得ることができた。今後は、この作業をさらに進め、レビュー作業の構造化、ツールの開発、利用実験を行う予定である。

【参考文献】

[Conklin88] Conklin,J. et al. : gIBIS : A Hypertext Tool For Exploratory Policy Discussion, CSCW'88, pp. 140-152 (1988.9).

[Lai88] Lai,Kum-Yew et al. : Object Lens: A "Spreadsheet" for Cooperative Work, ACM Trans. of OIS, Vol.6, No.4, pp.332-353 (1988.10).

[Malone87] Malone, T. et al. : Semistructured Message Are Surprisingly Useful for Computer-Supported Coordination, ACM Trans. of OIS, Vol. 5, No. 2, pp.115-131 (1987.4).

[Neuwirth90] Neuwirth, C. et al. : Issues in the Design of Computer Support for C0-authoring and Commenting, CSCW'90, pp.183-195 (1990.10).

[Sembugamoorthy90] Sembugamoorthy, V. et al. : ICICLE : Intelligent Code Inspection in a C Language Environment, COMPSAC'90, pp.146- 154 (1990).

[Yakemovic90] Yakemovic,B. et al : Report on a Development Project Use of an Issue-Based Information System, pp.105-118 CSCW'90 (1990.10).

3．グループウェアに関する調査

3．グループウェアに関する調査

コンピュータを用いたグループワーク支援の研究は、大きく分けてグループの協調作業の社会的・認知科学的な研究と、コンピュータを用いていかに支援するかといった工学的な研究がある。この両者は、各々が独立しているのではなくお互いに密接に関係している。こうしたことから1984年にPaul CashmanとIrene GreifらはCSCW (Computer-Supported Cooperative Work) と呼ばれる新しい学際的な研究分野を開拓した。1986年には、第一回のCSCW会議が米国のオースチンで開かれ、その後は、2年ごとに開催されている。また、ACM (Association for Computing Machinery)のSIGCHI (Special Interesting Group on Computer & Human Interaction)が主催するCHI会議でもCSCWが大きく取り上げられるようになってきており、SIGOIS (Special Interesting Group on Office Information Systems)が主催するCOIS (Conference on Office Information Systems)やCOCS (Conference on Organizational Computing Systems)でも検討されている。ヨーロッパでは、1989年に第一回のECSCWが開催され、その後はCSCWと同様に2年ごとに開催されている。このように、近年急激にこうした問題に対する関心が高まっている。ここでは、そうした会議で発表されたものや、海外調査で入手したものの中から代表的なものを22編選び紹介する。

グループウェアあるいはCSCWの研究は、非常に多岐にわたっているため、次の分野に分けて紹介することとした。

① グループウェア全般
② ワークステーション／ビデオ会議
③ マルチメディアソーシャルシステム
④ オフィスプロシージャシステム
⑤ メッセージ構造化システム
⑥ グループエディタ
⑦ グループメモリシステム
⑧ ツールキット

取り上げた論文は、複数の分野に該当するものもあるが、システムの中心となる目的や機能に合わせて便宜的に分類してある。以下に各分類の内容について簡略に説明する。

① グループウェア全般

ここでは、グループウェア（およびその研究の上で重要な要素となるグループ意思決定支援）について、定義や分類、概念、具体例等を示して、グループウェアの全体像を把握できるような論文を紹介する。

なお、論文中にグループウェアの具体例として、グループエディタであるGROVEが紹介されている。

② ワークステーション／ビデオ会議

ここでは、リアルタイムの対面型／分散型の会議を支援するためのシステムが対象となる。ワークステーションやビデオを利用した会議システムの開発例やシステム構築の際の設計概念等を示す論文を紹介する。ここで紹介するシステムは、Colab、Nick、MMconf、Rapport、Commune、CAVECAT の6点である。

③ マルチメディアソーシャルシステム

ここでは、単なる会議に留まらず社会的なコミュニケーションをコンピュータを介して行うシステムが対象となる。分散環境におけるインフォーマルな通信を支援するシステムに関する論文を紹介する。ここで紹介するシステムは、CRUISER および VideoWhiteboard である。

④ オフィスプロシージャシステム

ここでは、オフィスにおける各種事務処理の流れを構造化し、複数の担当者とコンピュータが協調的に作業を進めるシステムが対象となる。電子メールのメカニズムとオフィスのプロセス記述を用いて、処理を半自動的に進めていくシステムに関する論文を紹介する。ここで紹介するシステムは、ECF および DOMINO である。

⑤ メッセージ構造化システム

ここでは、情報のフィルタリングに関するシステムが対象となる。電子メールで送受信されるメッセージを半構造化することにより、メッセージの分類や優先度付けを自動的に行えるシステムである Information Lens と、この概念を拡張して多様なオブジェクトの記述と操作が行えるようにした Object Lens に関する論文を紹介する。

⑥ グループエディタ

ここでは、文書の共同執筆とレビューを支援するシステムが対象となる。複数の著者間、あるいは著者とコメンテータ間のコミュニケーションと情報の共有を支援するシステムに関する論文を紹介する。ここで紹介するシステムは、Quit、PREP、DistEdit の3点である。

なお、DistEdit はツールキットとしても捉えられる。

⑦ グループメモリシステム

ここでは、討論の過程を構造化し、討論内容の整理と記録保持を支援するシステムが対象となる。IBIS と呼ばれる議論モデルに基づいて、グループによるソフトウェア設計の上流工程における討論を支援するハイパーテキストである gIBIS と SIBYL という2つのシステムに関する論文を紹介する。

⑧ ツールキット

ここでは、グループウェアの個々の機能要素を部品として、これらの部品の組み合わせとカスタマイズによって状況に合わせたグループウェア・アプリケーションを自由に構成できるシステムが対象となる。リアルタイムの8種類のグループ・ツールを提供する LIZA と、会話と行動の管理のための

支援システム構築用部品群からなるツールキットであるStrudelに関する論文を紹介する。

なお、取り上げた論文の出典は次のとおりである。

- Communications of ACM,Vol.30,No.1, January 1987
- ACM Transactions on Office Information Systems, Vol.5, No.2, April 1987
- ACM Computing Surveys, Vol.20, No.2, June 1988
- CSCW'88 Proceedings, September 1988
- CHI'89 Proceedings, May 1989
- ACM SIGOIS Bulletin, Vol.11, No.2, April 1990
- CSCW'90 Proceedings, October 1990
- Communications of ACM, Vol.34, No.1, January 1991
- CHI'91 Proceedings, April 1991
- ECSCW'91 Proceedings, September 1991

3.1 グループウェア全般

3.1.1 Groupware

原文タイトル：	Groupware: Some issues and experiences
翻訳タイトル：	グループウェア：問題と体験
著書：	Clarence Ellis (Software Technology Program, MCC, Austin, Texas, USA) Simon Gibbs (Centre Universitaire d'Informatique, University of Geneva, Switzerland) Gail Rein (Software Technology Program, MCC, Austin, Texas, USA)
出典：	Communications of ACM, Vol.34, No.1, January 1991, pp.38-58

本論文では、広義のグループウェアについて研究するとともにグループウェアの開発者が直面している設計上のいくつかの問題について概説されている。本論文は次のような5つの部分に分かれている。まず「概観」ではグループの共通の仕事と共有環境の必要性の見地からグループウェアを定義している。ここでのグループウェアの定義は多くのシステムに及んでいるため、次の部分では「グループウェアシステムの分類」について述べられる。さらにこれらのシステムを構築する者の「視点」について述べられる。4番目の「概念と例」では一般的なグループウェアの概念がいくつか紹介され、グループウェアの一例として新しいグループエディタとして開発されたGROVEにこれらの概念を適用

した結果が示される。5番目にグループウェアの設計者や開発者が直面している設計上の問題点についての議論が示される。ここではリアルタイムのグループウェアにおけるシステムレベルでの問題に焦点が当てられている。

(1) 概　観

グループのインタラクションの支援方法を考えるに当たって留意しなければならない点は、通信、協同作業、調整作業の3つである。

① 通信、協同作業および調整作業の重要性

電子メールのようなコンピュータベースの通信は、他の形式のコミュニケーションを完全には統合化していない。本来、電子メールや電子掲示板のような非同期のテキストベースの世界は、電話や対面の会話といった同期の世界とは区別される。電気通信とコンピュータ処理技術の統合はこれらのギャップの橋渡しになる。

通信と同様に、協同作業はグループ活動の礎石である。効果的な協同作業には、人が情報を共有することが必要となる。今の情報システムは、ユーザをお互いに大きく引き離してしまっている。CADデータベースシステムで作業する2人の設計者を例に考えて見ると、彼らは同時に同じオブジェクトの異なる部分を修正することや、相手の変更について知ることは滅多にできない。グループの最新の状況を把握し、必要な時には各ユーザの行動をはっきりと通知するような共有環境が必要である。

通信と協同作業の効果はグループの活動がよく調整されていればより強化される。たとえば、調整がなければプログラマやライタのチームは対立したり、何度も同じ作業を行っているような状況になるであろう。調整は、仕事そのものとも、また何人かの仲間が仕事を遂行するときに必要なオーバーヘッドであるとも見なせる。

② グループウェアの定義

グループウェアの目標は、グループ活動の通信、協同作業、調整作業を援助することである。したがって、ここではグループウェアを

「共通の仕事（または目標）に携わる人のグループを支援し、共有環境のインタフェースを提供するコンピュータベースのシステム」

と定義する。共通の仕事と共有環境の観念はこの定義にとって重要である。この定義は、ユーザが共通の仕事を共有できないタイムシェアリングシステムのようなマルチユーザシステムを除外する。この定義では、ユーザが同時に活動することが必須であるとは言っていない。

グループウェアという用語は、最初にJohnson-Lentzによって「コンピュータシステムに社会的なグループのプロセスを付加したもの」と定義された。Johansenはその定義をコンピュータベースのシステムに限定した。本論文は基本的にシステムレベルの問題に関係しているので、Johansenの論に従ってい

る。

(2) グループウェアシステムの分類

ここでは、グループウェアの様々な種類を見分けるために有用な２種類の分類を取り上げる。最初の分類は、時間・空間に基づく分類である。２番目はアプリケーションレベルの機能的なものによる分類である。

① 時間・空間的分類

グループウェアは、対面のグループまたは多くの場所に分散しているグループを手助けすると考えることができる。さらにグループウェアは、リアルタイムのインタラクションや非同時、非リアルタイムのインタラクションにおいて通信や協調を強化するものと考えられる。これらの時間・空間的な考慮事項は、図3.1-1 に示されるような２×２のマトリクスによって表現された４つのカテゴリになる。会議室技術は左上、リアルタイムの文書エディタは左下、物理的な掲示板は右上、電子メールは右下のマスに入る。

	同じ時間	異なる時間
同じ場所	対面対話	非同期対話
異なる場所	同期分散対話	非同期分散対話

図3.1-1 グループウェアの時間・空間のマトリクス

包括的なグループウェアシステムは、４つの全てのカテゴリの要求に応えられる必要がある。たとえば、(a) グループ内で、リアルタイムで文書を編集するのにコンピュータを使用する（同時／同じ場所、または同時／異なる場所）という場合と (b) 自分の家かオフィスで一人で編集する（異なる時間）という場合とで、基本的機能やインタフェースの Look and Feel が共通であることはたいへん重宝である。

② アプリケーションレベルの分類

２番目の分類は、アプリケーションレベルでの機能に基づいており、包括的という意味ではない。また、これらの定義したカテゴリは重複するものも多い。

(a) メッセージシステム

グループウェアの最も親しみ深い例は、コンピュータベースのメッセージ交換システムであり、非同期のテキストメッセージのユーザグループ間の交換を支援する。電子メールやコンピュータ会議、あるいは電子掲示板システムの例がある。こういったシステムの増大は「情報オーバーロード」現象をもたらしてきた。最近のメッセージシステムには、ユーザが処理する負荷を軽減することによって情報オーバーロードの管理を助けるものがある。「知性」がメッセージ配送システムにつけ加えられる場合もある。例えば、Information Lens システムやImail システム等の例が挙げられる。

(b) マルチユーザエディタ

グループのメンバは文書の協同執筆、編集にマルチユーザエディタが利用できる。ForComment のようなエディタは非同期に使用され、さまざまな批評のコメントと著者の作成したテキストをうまく分けることができる。リアルタイムのグループエディタは、同時に同じものをグループ内の人が編集できる。エディタは透過的にロックと同期を管理しユーザは個人用の文書があるかのように占有できる。このような例としては、Collaborative Editing System (CES)、Shared Book 及び Quilt 等がある。

また、マルチユーザエディタには、他のユーザの活動をはっきりと通知できるものもある。Mercury は、プログラムチーム用のエディタであるが、他人がプログラムを変更したことによって自分のコードの変更が必要になるときユーザに通報する。DistEdit システムは、マルチプルグループエディタを作ったりサポートするためのツールキットを用意しようとしている。

(c) グループ意思決定支援システム及び電子会議室

グループ意思決定支援システム（GDSS）は、体系化されていない問題をグループで検討するためのコンピュータベースのシステムである。目標は意思決定会議の効率化、意思決定会議の高速化、意思決定の結果の質の改善である。GDSSは代替案の順序づけや投票用のツール、発想支援や論点解析のような意思決定の構造化を補助できる。

多くの GDSS は、数台のネットワーク化された WS 、コンピュータに制御された大型共有ディスプレイ、オーディオ／ビデオ装置から構成される電子会議室として構成されている。これらの施設のいくつかは特別に訓練されたオペレータが必要であり、また他のものはグループのメンバの操作能力を前提としている。アリゾナ大学の Plex Center Planning and Decision Support Laboratory がよく知られた例である。

(d) コンピュータ会議

コンピュータは、さまざまな方法で通信の媒介として役に立つ。特に、会議進行の方法に 3 つの新しいアプローチ、すなわちリアルタイムコンピュータ会議、コンピュータ遠隔会議、デスクトップ会議を与える。

リアルタイムコンピュータ会議は、電子会議室に参集するか物理的に分散しているユーザのグループが、WS または端末を通じて同時にインタラクトできる。グループが物理的に分散しているとき、

コンファレンスコールといった音声リンクがしばしば設けられる。リアルタイムコンピュータ会議のソフトウェアをインプリメントするためにのアプローチとしては、単一ユーザ用のアプリケーションを修正せずにアプリケーションの出力を各参加者のディスプレイに表示できるような会議環境に組み込む方法と、複数のユーザの存在を考慮した特別のアプリケーションを設計する方法の２種類がある。前者の例としては端末接続や複製ウィンドウがあり、後者の例としては、会議スケジュールシステムのReal Time Calendar（RTCAL）や、リアルタイムグループ注記システムのCognoterなどがある。

コンピュータ遠隔会議は、グループの会議に対する電気通信による支援である。例としては、電話会議やビデオ会議がある。遠隔会議は特別な部屋や訓練されたオペレータが時々必要となるといった不便な点がある。最近のシステムはWSベースの会議用のインタフェースがあり、より利用しやすくなっている。例えば、ゼロックス社はポートランドとパロアルトに分割されたプロジェクトチームで使うための音声／画像システムを設置した。同様のシステムの一つであるCRUISERは、ビデオ回線をブラウズすることで電子的に廊下を歩き回ることができる。

デスクトップ会議は、遠隔会議とリアルタイム会議の欠点を減らし長所をまとめたものである。この方法はやはり会議用インタフェースにWSを使用しているが、参加者によって共有されるアプリケーションを実行することができる。現在のデスクトップ会議システムは、WSごとにマルチビデオウィンドウが出来るようになっている。これは、情報の動画像や参加者のビデオイメージが表示できる。例としては、MMConfシステムやRapportマルチメディア会議システムが挙げられる。

(e)　インテリジェント・エージェント

インテリジェントエージェントは、ある特定の仕事の型に対応し、ユーザインタフェースの動きを人間のユーザに似せたものである。代表例として、Lizaという名のエージェントを持つグループウェアツールキットがある。

(f)　調整システム

調整システムは、全体の目標といった状況の下で関連する他人の行動を見ることができると同時に、各個人は自分の行動を見ることができる。システムはたいてい、ユーザの行動や待ちの状態の現状を知らせたり、自動的に助言や警報でユーザの行動のきっかけをつくる。調整システムは、形式（フォーム）、手順、会話、通信構造の４つの内どれに基づくかにより、４つのモデルの型に分類できる。フォームに基づいたモデルの例としては電子循環フォルダ（ECF）があり、会話に基づいたモデルの例としてはThe Coordinatorがある。

(3)　視点

グループウェア成功のためには、少なくとも５つのキーとなる分野または視点がある。それは、(i)分散システム、(ii)通信、(iii)人間－コンピュータ間のインタラクション、(iv)人工知能（AI）、(v)社

会理論である。グループウェアとこれら５つの研究領域の関係は相互に有益なものである。各分野はグループウェアの理論や実践における理解を進めるだけでなく、グループウェアは５つの領域全部に対して研究のやりがいのある分野あるいはトピックスを提供する。こうしたトピックスはグループウェアがなければ研究されないものである。

同じく重要なことは、与えられたグループウェアシステムはたいていこれらの分野の２つ以上の視点が関係しているという点である。例えば、デスクトップ会議のパラダイムは、(i)通信技術に始まりコンピュータの能力及び受信機の表示デバイスでこれを強化するという方法と、(ii)パーソナルWS（分散システムの視点）に始まり通信機能と統合するという方法の、２つの方向から導き出されたことがわかる。

(4) リアルタイムグループウェアの概念と例

① グループウェアの用語

グループウェアにおいて具体化された用語と考えは進化している。ここでは、グループウェアのシステムの研究と比較に役に立つ重要な用語を説明する。

(a) 共有状況

共有状況とは一組のオブジェクトのことであり、オブジェクト及びそのオブジェクトの上になされる行動が一組のユーザによって見ることができるオブジェクトの集合である。例として、協同執筆システムにおける文書オブジェクトおよび電子教室におけるクラスノートがある。

(b) グループウインドウ

グループウインドウとは、そのインスタンスが異なるディスプレイに表示されているウィンドウの集合である。各インスタンスは接続される。例えば、あるインスタンスに円を描くと他のインスタンスに円ができ、また、あるインスタンスでスクロールすれば他のインスタンスもスクロールする。

(c) テレポインタ

テレポインタは、２つ以上のディスプレイに現れるカーソルであり、異なるユーザが動かすことができる。あるディスプレイで動かされれば、全てのディスプレイで動く。

(d) ビュー

ビューは、共有状況のいくつかの部分のビジュアルまたはマルチメディアによる表現である。異なるビューには、表現は違っていても（例えば、数列は表やグラフで表現できる）同じ情報を含めることができる、また、共有状況の中で参照される異った部分に対して、同じ表現を使用できる。

(e) 同期及び非同期のインタラクション

同期のインタラクションにおいては会話のようにリアルタイムで会話できる。非同期のインタラクションは郵便物による文通のように長い時間をかけて行われる。

(f) セッション

セッションとは、グループウェアシステムによって支援された同期インタラクションのある期間のことである。例として公式のミーティングや非公式のワークグループの討議がある。

(g) 役割 (role)

役割は、人またはシステムモジュールに対して与えられた特権及び義務である。役割は、公式または非公式に持たせることができる。例えば、多くの人を訪問して話したいと思い立った人は、自然に（非公式に）話題の提供者の役割が与えられるであろう。また、グループの長は、公式にマネージャの役割が与えられる。

② GROVE：グループウェアの一例

(a) GROVEの概要

The GRoup Outline Viewing Editor（GROVE）は、ここで紹介した概念を実現したグループウェアの一例である。MCCでインプリメントされたGROVEは、ワークセッション中にグループの人がアウトライン（概要）を同時に編集するために使うよう設計された簡単なテキストエディタである。

GROVEのセッションでは、各ユーザは各人のWSとビットマップディスプレイを持っている。各ユーザは、スクリーン上に何重にも重ねられたウインドウの上で1つ以上のテキストのビューを見たり操作できる。GROVEは、ビューとビュアとを区別している。ビューとは読み込みアクセス権によって限定されたアウトラインにおける事項の部分集合である。ビュア(viewer)とは連続したビューの部分集合を見るためのグループウインドウである。GROVEのビューとビュアは、プライベート、共有、公開に分類される。プライベートビューとは、ある特定のユーザのみが読むことができる事項である。共有ビューとは、列挙されたユーザの集合が読むことのできる事項である。公開ビューとは、全てのユーザが読むことができる事項である。

図3.1-2 はGROVEのグループウィンドウを示している。グループウィンドウはユーザの同期インタラクションのためのビュアを共有できる。

グループウィンドウは、ビューの表示だけでなく、誰がウィンドウを使用でき誰が与えられた時間のセッションで実際参加しているかを示す。この情報はビューのメンバの人のイメージ写真等（イメージが使用できない場合は単に氏名）をウィンドウの下のボーダーに沿って表示できる。ユーザがセッションに出入りすると彼らの写真がそのとき可能なグループウィンドウに表示されたり消されたりする。図3.1-2 のウィンドウは下のボーダー沿いに示されている3人のユーザのWSに表示され、各ユーザは他の人とセッションがつながっていることがわかる。ユーザは標準の編集操作（挿入、削除、カット、ペーストなど）を行ってグループウィンドウ中の編集中のアウトラインを修正できる。これが実行されると、3人全員ともすぐにその修正を見ることができる。スクリーンで灰色をしたアウトラインの項目（図3.1-2 の一番下の項目のように）はそのユーザによって修正することはできない（修

正は黒い色の項目に限られる）。ユーザは（マウスクリックで）アウトラインをオープンしたりクローズしたりでき、また読み書き許可を変更できる。

関係者はGROVEのセッションにいつでも参加／退出できる。ユーザがセッションに入った（または再び入った）とき、以前に保存したバージョンの再生を選択しなければ、最新の文書を受け取ることができる。たとえセッションに参加していない間に変更が行われたとしても最新の状態は保持される。セッションは参加者が誰もいなくなったとき終了する。

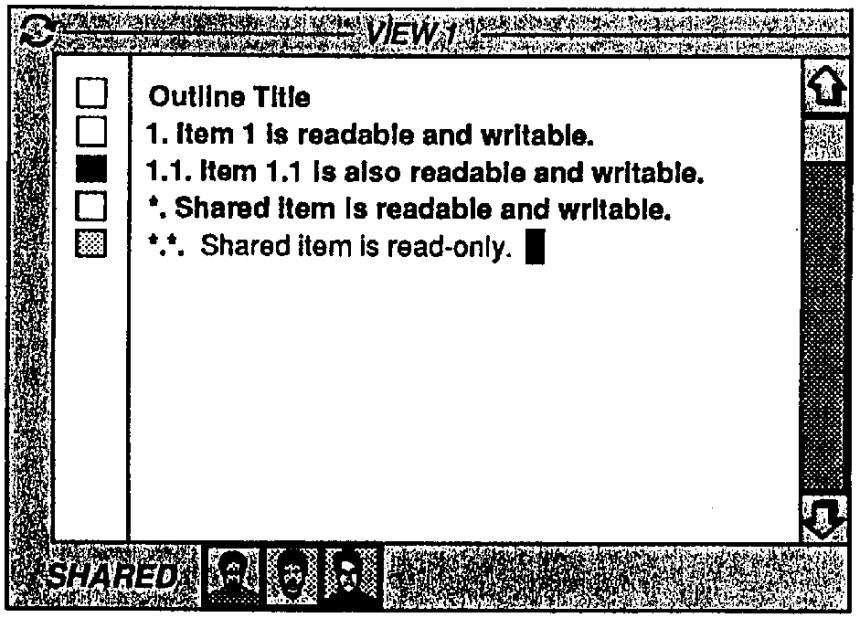


図3.1-2　GROVEのグループウィンドウ

(b)　GROVEの使用体験

GROVEは、数グループの協同論文の計画からブレーンストーミングの提示までさまざまな設計の活動に使われてきた。一般にセッションは３つの型に分けられる。

(i)　対面セッション：３台のSun WSと電子黒板のある研究所の電子会議室におけるセッション

(ii)　分散セッション：関係者が自分のオフィスでマシンを使って仕事をし、音声通信用のスピーカフォンで会議電話を行うセッション

(iii)　混合形式セッション：対面の参加者もいれば、分散した参加者もいるセッション

表3.1-1 は、15のグループセッションにおけるセッションの型、グループ規模と仕事を表にしたものである。分散及び混合形式のセッションにはしばしば５、６人の参加者がいる。

ユーザから見れば、分散編集セッションは対面の編集セッションとは明らかに異なった体験である。分散セッションについての観察結果として、次の点が挙げられた。

表3.1-1　GROVEセッションのまとめ

セッション形式	人数	仕　　事
分 散	3	プロジェクト記述の問題の確認
対 面	3	プロジェクト記述の問題リストの精選
対 面	3	技術報告のアウトライン
分 散	3	管理上の発表計画
対 面	3	管理上の発表計画（続き）
対 面	2	チュートリアルの計画
対 面	3	プロジェクト計画の討議
対 面	3	システムのソフトウェア拡張の討議
対 面	3	プロジェクト計画の討議（続き）
対 面	3	プロジェクト計画の討議（続き）
混合形式	5	２つの計画の類似点/相違点の確認
分 散	3	遠隔セッションのテスト
分 散	3	２つの関係したトピックのブレインストーム
分 散	5	論文のアウトライン
混合形式	6	論文のアウトライン

・　情報アクセスを増加させる。

・　グループ内の並行作業を促進する。

・　論議をより困難にする。

・　グループの焦点を集めることをより困難にし、気分を集中させる必要がある。

・　社交的インタラクションが減少する。

また、グループ編集（同期またはリアルタイム）の単一ユーザの編集に対する比較結果として、次の点が挙げられた。

・　インタラクションは曖昧で、焦点が定まらず、混沌としている。

・　他人との衝突は滅多に起こらない。

・　一人で実施した場合より効率的にできる。

・　情報の無駄を無くし、効果的なグループの成果に導く。

・　ツールの使用を自然な態度で習得できる。

(5)　設計上の問題

ここでは、小～中規模のグループが使うように設計されたリアルタイムのグループウェアを対象として、その設計者や開発者が直面している問題について述べる。

①　グループインタフェース

グループインタフェースは、グループの行動を表現しかつ、単一のユーザよりむしろ複数のユーザ

によって制御されることにおいて、単一ユーザのインタフェースとは異なる。グループインタフェースの一例は、GROVEのグループウィンドウである。他の例では、リアルタイムコンピュータ会議用のインタフェースやマルチプレーヤー用のゲームのインタフェースなどがある。
グループインタフェースでは単一ユーザのインタフェースでは出てこなかった設計上の問題が出てくる。基本的な問題は、複雑さの管理法である。複数のユーザは単一のユーザよりも、より活動のレベルが高く、高度の並行性を引き起こす傾向がある。インタフェースはこれらの複雑な作用を支援しなければならない。

また、単一ユーザインタフェースのどのような技術と概念が、グループインタフェースの構築に役に立つかということも重要な問題となる。単一ユーザインタフェースのままであるとどこでうまく行かなくなるかということを把握することにより、新しい概念の必要性が示される。例えば、スクロールバーは複数の人数で操作する場合役に立つか、あるいは単に気を散らせるだけであるかなどである。

その他、WYSIWISの問題、グループの焦点と散漫の問題、グループ力学に関連した問題、スクリーンのスペースの管理に関連した問題、グループインタフェースのツールキットに関連する問題などについても考慮する必要がある。

② グループプロセス

プログラムのウォークスルーのように明確に定められた仕事は、ユーザが集まって参加する必要があり、これはグループプロセスといわれる。グループプロセスはシナジー（協力作用）と並行性を増大させるが、必要となる調整のオーバーヘッドはグループの負荷となり、効果を低下させる。グループウェアの技術は、利益を増大する一方でオーバーヘッドを最小化しようとしている。

考慮すべき問題としては、グループプロトコル、グループオペレーション、組織的・社会的要因、例外と調整、行動支援の統合などの点がある。

③ 同時実行制御

グループウェアシステムは、参加者の同時操作による競合を解決するために同時実行制御を必要としている。GROVEのようなグループエディタの場合、一人が文に単語を挿入する一方で、もう一人がその文を削除することも有り得る。グループウェアには独特の同時実行の一連の問題がある。明確なロックやトランザクション処理のようなデータベースのアプリケーションにおける同時実行処理の多くのアプローチは、グループウェアにとって不適当であり、緊密に結合したチームワークを妨げてしまう。

考慮すべき問題としては、応答性、グループインタフェース、広域分散、データの複製、堅牢性(rubstness)等がある。

また、同時実行制御方法についても考慮すべきである。制御方法のアプローチとしては、単純ロック(Simple Locking)、トランザクションメカニズム、交替(turn taking)プロトコル、集中化コントロー

ラ、独立性検出、可逆実行、オペレーション変換などが考えられる。

④ 他のシステムの問題

グループウェアは、電子メールシステムそのものに近いものから、最新技術のリアルタイムのマルチユーザツールまで幅広いシステムを含んでいる。グループウェアの設計者は共通のインプリメント上の問題に直面する。ここで考慮しなければならない問題点は、通信プロトコル、アクセス制御、（リアルタイムの）通知などである。

3.1.2 GDSS

原文タイトル： Computer-Based Systems for Cooperative Work and Group Decision Making

翻訳タイトル： 協調作業およびグループ意思決定のためのコンピュータ利用システム

著書： Kenneth L. Kraemer
(University of California, Irvine, California, USA)
John Leslie King
(University of California, Irvine, California, USA)

出典： ACM Computing Surveys, Vol.20, No.2, June 1988, pp.115-146

本論文では、グループが作業する際のニーズに合うように作られた「グループ意思決定支援システム（GDSS）」についてレヴューされ、これらのシステムの現在までの導入結果について評価されている。まず、GDSSの基本的概念がレヴューされ、米国において現在稼働中あるいは開発中のシステム例が示され、また、現状におけるGDSSの導入結果が評価される。さらに、意思決定を始めとして、コミュニケーションや情報処理を含むグループ活動を支援するGDSSの今後の発展について述べられている。

(1) GDSSの定義づけ

さまざまな論文において、GDSSを構成するものは何かという点についての一致した見解は、ほとんど見つけられない。DesanctisとGallupeは、GDSSを「ひとつのグループとして一緒に作業を行っている意思決定者が、込み入った問題を解決する際に手助けとなる、対話形式のコンピュータ利用システム」と定義した。この定義は、グループ活動の手助けとなる方法の発展のために、隠されていたニーズを理解するという点で、最初の適切な定義であると言える。

Huberは、ニーズとして次のようなジレンマがあることを示した。意思決定者にとっては、検討すべき多くの問題があって長時間の会議が増える一方であるが、彼らには他にも差し迫った問題があるた

め、会議に出席する暇が無い。このジレンマを解決するためには、会議をより効率的に行うことであり、このためのキーコンセプトがGDSSなのである。この方法は、決定内容の質を低下させることなく決定のスピードアップを図り、さらに決定内容の質を向上させることまで期待している。

GDSSは、このような目的をどのようにして達成できるのか。Huberは、論文中で次のように簡潔に述べている。

(a) グループ意思決定を効果的に行うためには、出席者が決定に満足し、後々出席者がうまく一緒に仕事ができるように、状況のニーズに合致することが必要である。

(b) 討論が特定の少数者によって仕切られてしまうことや、下位のメンバが上位のメンバのいいなりになってしまうこと、グループが各メンバの意向・考えを統制してしまうこと、さらに、普段からメンバ間のコミュニケーションがうまくいっていないことや、問題点を究明し別の策を講じる時間が不十分であることなどが原因となって、グループ意思決定の生産性が落ちてしまう。

(c) GDSSはこのような問題に対して、次のような支援を行うことができる。すなわち、出席者一人一人がPCを持ち、全員が一度に見ることのできるディスプレイスクリーンを備え、データベースにアクセスしたり、スクリーンを通してグループリーダとコミュニケーションを行ったり、ワープロソフトを使用したり、あるいはグループ共用のデータやグラフィクスにアクセスしたり、メンバ相互のコミュニケーションをコントロールしたりといったことができるシステムを提供する。

Huberが他の論文で指摘しているように、意思決定時におけるGDSSの役割は、特に「理性的（rational）」に意思決定を行えることを目指したものである。つまり、意思決定会議の出席者は、グループ全体の役に立つために情報を理性的に使用しなければならない。

(2) GDSSの分類

① GDSSの分類の基礎

ここでは、グループが意思決定を行わなければならない状況において使用できる技術によって、GDSSを分類する。この場合、コンピュータ、通信、意思決定技術、あるいはこれらの組合せが考えられる。表3.1-2に示すようにGDSSのベースとなる6つの技術について述べる。

② GDSSの要素のパッケージ

GDSSを理解するためには、これを(a)ハードウェア、(b)ソフトウェア、(c)オーガニゼイションウェア、および(d)要員の4種類の要素から成る社会技術的なパッケージ（組合せ）と考えた方が分かりやすい。

表3.1-2　技術を基準としたGDSSの分類

技　　術	要素技術の組み合せ
電子会議室	コンピュータとＡＶ
遠隔会議設備	コンピュータと通信
グループネットワーク	コンピュータネットワークと相互会話
情報センタ	コンピュータ、データベース、及び検索ツール
意思決定会議システム	コンピュータと意思決定モデル
協同研究所システム	コンピュータと協同作業用ツール

(a)　ハードウェア

ハードウェアとは、会議用施設、コンピュータ、通信設備、AV機器等を指す。会議用施設としては、最低限会議用テーブルと附属設備を備えた個室でなければならない。会議室には、会議のオブザーバ用のスペースを設けることもある。また、事務局や出席者が休憩できるようにラウンジルームやAVを備えた会議室が必要な場合もある。

コンピュータ機器としては、グラフィック処理が可能なコンピュータで、通常はグループ全員が見ることのできるような大きなディスプレイあるいはプロジェクションスクリーンを備えていなければならない。もう少し高度なシステムになると、各メンバに一つずつPCあるいは端末を備えたり、メンバ相互の通信の制御や、データベースやモデルにアクセスができる大規模な処理装置を置いたり、ローカルメンバ間の通信を行うためにLANを敷いたり、外部のメンバやデータベースとリンクするための遠距離通信システムを備えたり、複数の大スクリーンあるいはマルチウインドウの表示ができるスクリーンを備えたりする。また、高度なシステムでは、コンピュータ会議用やテレビ会議用に、マルチポイントの相互通信ができたり、メンバ全員が見たり修正できるグループ共有のファイルやディスプレイと、各メンバのみが見たり修正できる個人用のファイルやディスプレイとに分かれていたりする。

(b)　ソフトウェア

ソフトウェアは、GDSSの技術的機能のキーとなるものであり、一般的な情報処理、意思決定モデル化、そして通信を支援するために利用される。

一般的な情報処理のためのソフトウェアとしては、個人用／グループ用に、データベース管理システムや高級プログラム言語処理、あるいはグラフィック処理／スプレッドシート／統計分析等が可能な統合型パッケージなどがある。一方、意思決定モデル化用のソフトウェアは、特にグループの意思決定を支援することを目的にしており、モデル化用言語（例えば、SIMSCRIPTやDYNAMO等）、意思決定構造化技術、ブレインストーミング、一般的なグループ活動のための技術、デルファイ技術、意思決定分析技術などのためのものがある。通信用のソフトウェアは、グループワークを協同的に行う

ことの支援を目的としており、ローカルあるいは遠距離間においてテキスト、データ、音声、ビデオ画像等の交換を行うためのツールなどがある。

(c) オーガニゼイションウェア

オーガニゼイションウェアには、組織として利用するデータ、グループとしての活動、グループの協同作業を管理する活動などがある。各メンバのアイデアや意見をまとめ、各々の重み付けや判定を行ってデータを統合するようなGDSSもある。また、組織内のデータだけでなく、外部のデータベースも利用できるGDSSもある。これらのデータベースには、戦略的・作戦的プラン、予算、マーケットや顧客のデータ、生産データが含まれる。また、オーガニゼイションウェアには、グループとしての活動が含まれる。これは、グループのメンバ間の関係が、独裁的であるか民主的であるか、友好的であるか闘争的であるか、あるいは政治的であるか論理的であるかによって変わってくる。グループとしての活動は、グループリーダの選ばれ方や議題の選ばれ方によって、物事の流れやグループ意思決定会議が左右される。また、グループとしての活動は、各メンバの組織の中での位置づけによって規定されることもある。さらに、オーガニゼイションウェアには、グループの協同作業を管理する活動もある。これは、グループ活動に積極的に参加させたり、決定事項について秘密を守らせたり、グループ活動を円滑にさせるといったように、グループ活動の業務と社会性との対立を緩和させることを目的としたものである。

(d) 要員

要員とは、グループの参加者（出席者）およびグループ活動を支援するスタッフ（進行役）を指す。「進行役（facilitator）」の存在がGDSSのキーとなる。この役目は、非常に幅が広い。進行役は、最低限グループ活動を支援する技術のオペレータであることが要求される。例えば進行役は、計算を行ったり、ディスプレイを操作したり、会議のドキュメントをとったりする。また進行役は、実際に会議を進行させたり、個々の意思決定やグループ活動のリーダ役を務めたりもする。新規のグループにおいては、トレーナ役や障害復旧役を務めることになる。この場合進行役は、メンバにハード／ソフトの使用法やグループ活動の技術を教え、問題が起こったときに助言したり、グループにフィードバックさせたりする。

表3.1-3に、上に述べた4つの要素とGDSSの6つの技術との対応を示す。

(3) GDSSシステムの実例

現在あるGDSSの実例を全てあげることは不可能に近い。電子会議室は、何百というオーダで存在する。AV業者の技術的アドバイスを受けて、建設業者やエンジニアリング企業が新設あるいは改装したものである。現在でも、電子会議室は増えつつある。これに比べて、ビデオ遠隔会議設備は少ない。

表3.1-3　GDSS の代表的な要素

要　素	電子会議室	遠隔会議設備	ｸﾞﾙｰﾌﾟﾈｯﾄﾜｰｸ
ﾊｰﾄﾞｳｪｱ	会議室、AV、ｸﾞﾗﾌｨｯｸﾃﾞｨｽﾌﾟﾚｲ、ｺﾝﾋﾟｭｰﾀ	会議室、AV、AVとｺﾝﾋﾟｭｰﾀの通信制御装置	ｵﾌｨｽ、ﾌｧｲﾙｻｰﾊﾞとWS、電話、ｺﾝﾋﾟｭｰﾀﾈｯﾄﾜｰｸ
ｿﾌﾄｳｪｱ	会話的ｸﾞﾗﾌｨｯｸｽ	通信	同時／非同時のｺﾝﾋﾟｭｰﾀ会議、端末接続、ﾘｱﾙﾀｲﾑの会議のｽｹｼﾞｭｰﾘﾝｸﾞ、ﾋﾞｯﾄﾏｯﾌﾟﾃﾞｨｽﾌﾟﾚｲの共有
ｵｰｶﾞﾆｾﾞｲｼｮﾝｳｪｱ	AV、規格化されたﾚﾎﾟｰﾄ、標準的な会議運用規定	AV、遠隔会議用運用規定	議長主導型ﾐｰﾃｨﾝｸﾞ
要　員	出席者、AV技術者	出席者（複数の場所）会議のｱｼｽﾀﾝﾄ	出席者（ﾛｰｶﾙでの複数の場所）、ﾘｰﾀﾞ
例	組合せ式のAVﾙｰﾑもあるが通常ｶｽﾀﾑﾒｲﾄﾞ	静止画電話ｻｰﾋﾞｽ	MIT Lab for Computer Science RTCAL and Mblink EIES, NOTEPAD, PARTICIPATE CONFER II

要　素	情報センタ	意思決定会議ｼｽﾃﾑ	協同研究所ｼｽﾃﾑ
ﾊｰﾄﾞｳｪｱ	会議室、大ｽｸﾘｰﾝのﾋﾞﾃﾞｵﾌﾟﾛｼﾞｪｸﾀ、ｺﾝﾋﾟｭｰﾀ、端末	会議室、大ｽｸﾘｰﾝのﾋﾞﾃﾞｵﾌﾟﾛｼﾞｪｸﾀ、表示端末、票決用端末	会議室、電子黒板、WS、electem
ｿﾌﾄｳｪｱ	DB管理、統計用ﾊﾟｯｹｰｼﾞ、検索・ｸﾞﾗﾌｨｯｸｽ・ﾃｷｽﾄ処理用ｿﾌﾄ	意思決定分析用ｿﾌﾄ、ﾓﾃﾞﾙ化用ｿﾌﾄ、投票集計・表示用ｿﾌﾄ	ﾏﾙﾁﾕｰｻﾞｲﾝﾀﾌｪｰｽ、WYSIWIS、ｱｳﾄﾗｲﾝ化（COGNOTER）、評価（Argnoter）
ｵｰｶﾞﾆｾﾞｲｼｮﾝｳｪｱ	組織内外のDB、標準的な会議運用規定、標準的な会議（年次総会等）	民主的な意思決定規定（１人１票、重要な議題を全て検討すること、多数決等）	標準的な会議運用規定
要　員	参加者、ｺﾝﾋﾟｭｰﾀ専門家ﾓﾃﾞﾙ化専門家	出席者、決定分析者、ｸﾞﾙｰﾌﾟ活動のｱｼｽﾀﾝﾄ	出席者
例	HOBO System, SYSTEM W, EIS,EXPRESS,XSIM	Group Decision Aid,DDIのDecision Conferences, SUNY AlbanyのDecision Tectronics,Arizona大のPlanning Lab, Minnesota大のGDSS Lab	Xerox PARCのColab, MCCのNICKﾌﾟﾛｼﾞｪｸﾄ

米国においては、私設の遠隔会議設備は30～40の数であり、公共のものは15～20に過ぎない。

また情報センタは、比較的新しく誕生したケースであり、米国においてはコンピュータのメインフレームメーカが、今までの情報処理機器供給の一環として始めたものである。情報センタの供給は、最初は組織の情報処理部門によって行われるが、次第にその情報処理部門にコンセプトや機器やソフトウェアを納入するコンピュータベンダが行うようになる。

意思決定会議システムは、他のGDSSに比べて明確な特徴を持っているため、市場に出回っている製品をあげることができる。表3.1-4に、主なGDSSの供給元と、供給しているシステム、およびそのシステムが持つ機能を示す。

表3.1-4　主要なGDSSの供給元

供給元	システム	機能
Applied Futures,Inc.	CONSENSOR（ハード／ソフト）	投票の集計・表示
Compshare	SYSTEM W（ソフト） EXPRESS（ソフト）	ﾃﾞｰﾀ管理、ﾓﾃﾞﾙ化、統計分析、ｸﾞﾗﾌｨｸｽ、ﾚﾎﾟｰﾄ作成、PC通信
Decisions and Designs,Inc.	Decision conference（完全なパッケージ）	会話的決定分析(6ﾓﾃﾞﾙ)、会議機能、決定分析、ｺﾝｻﾙﾃｨﾝｸﾞ
EXECUCOM	IFPS（ソフト）	会話的会計ﾓﾃﾞﾙ化、ﾃﾞｰﾀ管理、ｸﾞﾗﾌｨｸｽ
Institute for the Future	研究のみ	遠隔会議の研究・ｺﾝｻﾙﾃｨﾝｸﾞ
MIT Laboratory for Computer Science	MPCAL,CDS,RTCAL, MBlink（研究のみ）	ｶﾚﾝﾀﾞ管理・ﾘｱﾙﾀｲﾑの会議・協同文書編集など物理的に離れて散らばるﾛｰｶﾙｸﾞﾙｰﾌﾟの支援
Perceptronics,Inc.	GROUP DECISION AID（ハード／ソフト）	会話的意思決定ﾂﾘｰ、分析、ﾁｭｰﾄﾘｱﾙ、文書作成
SRI International	QUICKTREE,APLTYER（ソフト）	個人向けの会話的意思決定ﾂﾘｰ分析
SUNY,Albany Design Techtronics Group	公共サービス（完全なパッケージ）	会話的決定分析(6ﾓﾃﾞﾙ)、ﾃﾞｰﾀ管理、ｸﾞﾗﾌｨｸｽ、決定ｺﾝｻﾙﾃｨﾝｸﾞ
UCLA Cognitive Systems Laboratory	研究（数学的モデル／ソフト）	ｸﾞﾙｰﾌﾟ意思決定ｾｯﾃｨﾝｸﾞと分析
University of Arizona Planning Laboratory	PLEXYS（ソフト） 研究・公共サービス（完全なパッケージ）	電子的なﾌﾞﾚｲﾝｽﾄｰﾐﾝｸﾞ、stakeholder分析、組織の分析、知識管理、ｸﾞﾗﾌｨｸｽ、ﾚﾎﾟｰﾄ作成、PC通信
University of Minnesota GDSS Laboratory	CAM－研究のみ（ソフト）	Snow card技術、一般的なｸﾞﾙｰﾌﾟ技術、stakeholder分析、ｽﾌﾟﾚｯﾄﾞｼｰﾄ・割付ﾂｰﾙ、ﾃﾞｰﾀ管理
Xerox PARC	COLAB－研究（完全なパッケージ）	対面式のｸﾞﾙｰﾌﾟﾜｰｸに対するｺﾝﾋﾟｭｰﾀの支援

これらのシステムについて調べてみると、いくつかのことが分かる。まず第一は、ほとんどの意思決定会議用 GDSS が次の３つの機能を持っていることである。

① 意思決定ツリーやmultiattribute utility分析といった構造的な意思決定分析機能

② 社会的な判断分析やデルファイ技術、一般的なグループ活動のための技術といった構造的なグループ処理機能

③ データ管理、グラフィックディスプレイ、意思決定経過のドキュメンテーション、チュートリアル、意思決定分析のコンサルテーション、グループ活動支援、会議支援、票決集計・表示といった協同作業支援機能

これら全ての機能を全ての要素（ハードウェア、ソフトウェア、オーガニゼイションウェア、要員）で満たして販売されている例はほとんど無い。現在全要素のパッケージとして購入できるものは、Perceptronics社の GROUP DECISION AID だけである。その他のシステムは、各組織の内部向けシステムであり、外部からは研究用として使用料を支払って利用できるのみである。Arizona大学のPLEXYSというソフトウェアは販売されており、約１ダースの大学と企業が導入して、各組織向けにアレンジしている。

第二は、一つの GDSS の中にいろいろな種類の機能を組合せる傾向が見られることである。例えば、SYSTEM W のような情報センタ用の製品は、一つのシステムの中にいくつかのソフトウェア機能を装備している。従来このような各機能は、各々別のソフトウェア製品として分かれていたものである。同様に、GROUP DECISION AID には Deccisions and Designs社によって意思決定モデルが一覧できるように機能追加された版や、グループ活動にとってさらに役立つようにマーケティングプラン機能を追加した版等もある。Minnesota大学のCAM と Arizona 大学の PLEXYS は、グループ意思決定とグループ活動の研究の両方を支援するシステムを目指している。

第三は、GDSS は様々な形式をとりうるということである。一般に GDSS の技術が進歩するほど、GDSS がグループ活動へ介入する度合はより深くなる。

(4) GDSS システム導入後の評価

① GDSS 研究の発展経過

GDSS 研究の発展経過には、大きく二つの流れがある。一つは、個人レベルおよびグループレベルでの意思決定そのものの研究である。この方面では、まず、個人およびグループにおける心理学的あるいは認知科学的な働きを見つけることや、小グループ内の相互関係の社会学といった研究がある。また、組織としての開発という観点から、グループにおいて意思決定までの時間を早め、そして友好的な合意に達せられるように、グループの活動を支援する方法を見つけるといった研究もある。これ

らの研究は、技術という面には係わらない。むしろ、意思決定を行う人々が何を考え、どのように行動するかということを問題にしている。

もう一つの流れは、意思決定に役立つ情報の収集・管理・提供を行う技術に係わる研究である。この方面は、エンジニアリングという面が中心になる。意思決定の新しい方法や技術を生みだし、これに基づいて意思決定を行う人の役に立つツールを開発することを目的としている。この方面における主な研究領域は、意思決定過程の分析や意思決定のモデル化、およびコンピュータや電子的なデータの記録・通信・情報表示等を対話的に利用した場合の人間の情報処理のあり方等である。

意思決定会議システムや協同研究所システムは、社会心理学と技術の両方の面から開発されたものである。この技術は、情報の獲得と共有を容易にするために、他のGDSSの技術を集めたものである。グループ意思決定活動の研究成果は、その活動を円滑にするのシステムの技術開発のために適用されている。今のところ、意思決定会議システムや協同研究所システムは技術の面に片寄っている。それは、実際の意思決定がどのように行われるかということが、依然としてよく分かっていないからである。グループ意思決定活動についての社会心理学的な見方は広く指示されているものの、依然として「理性的な」モデルの方から影響を受けている。

② GDSSを使用して得られる効果

(a) 気分的な効果

GDSSは、会議を活発にし、グループ間の結束を固めるという気分的な効果をもたらす。コンピュータを使ったグラフィックス表示や投票、通信、データ分析、モデル化のシステムによって、グループ活動に関心を持つようになる出席者もいる。これらの技術の中には、その動きを見ているだけでおもしろく、エキサイトさせるものがある。

ただし、GDSSのこのような気分的な効果が、意思決定の結論の品質に寄与するかどうかは、まだはっきりしていない。

(b) 議事進行の手助け

GDSSの技術は、グループ意思決定の議事進行の手助けとなる。グループ意思決定活動には、決定すべきキーとなる問題に注意を向けさせたり、出席者から問題に対する考えを引出したり、どのような行動を起こすべきかについての出席者の合意の糸口を得るための、議事進行のルール（プロトコル）がある。

GDSSの技術は、決定すべき問題の中で重要な項目は何かということを、出席者により早くより正確に理解させることを目指していると考えられる。簡単な例としては、コンピュータによるグラフィックディスプレイがあげられる。多くの人々にとって、文章や表からではデータの特徴を読み取ることは難しいが、チャート図やグラフを用いれば、特徴のある部分についてその違いを際立たせることができる。投票システムはこれに似ているが、さらに明確な効果がある。電子投票システムは、合意

があるか無いかをはっきりさせ、さらに討議すべき項目をはっきりさせ、討議のために新たな情報を提供するといった手助けを目的として設計される。

このようなGDSS技術が、グループ意思決定の議事進行ルールを確立させる。このようなルールが常によりよい意思決定の役に立つか否か、研究からははっきりとは分からないが、研究だけでなく実際の経験を通して、このルールには何らかの知恵があるだろうということは分かっている。とは言っても、確立したルールは、広い範囲で適用できるのか否かは明確ではない。

(c) 意思決定の際に利用できる情報の品質向上

GDSSは、意思決定者が利用する情報の品質を向上させることがある。これは、次の二つの場合である。まず、意思決定会議において、他の方法では出席者が入手できない情報を、オンラインデータベースを利用して得ることである。もう一つは、モデル化やシミュレーション機能を利用して、意思決定活動において発生する情報を切り詰めたり、評価したり、統合したりすることである。

GDSSのこのような手助けによって、会議の出席者の一人一人が情報を取捨選択したり、意思決定における問題点を理解できるような、品質の向上が望めるであろうか。単に多くの情報を提供するだけでは意思決定の向上にはつながらない、ということは広く認められている。多くの情報がどんどん与えられると、出席者の能力を超えてしまい問題を検討できなくなり、意思決定活動が混乱してしまうこともある。意思決定会議の場において、単純にオンラインデータベースを利用するだけでは、意思決定の活動やその品質を劇的に向上させるというわけにはいかないようである。とは言え、出席者が選択できる様々な仮定をテストできるGDSSの効果は、GDSSの技術の中でも重要なものであるだろう。

(5) GDSS導入の障害となる問題

① 技術的な問題

GDSSは、コンピュータの情報処理や記録、グラフィックディスプレイ、データベース管理システム、統計処理、意思決定分析やモデル化のプログラム、通信、分散入力装置等の様々な技術に支えられている。技術開発の現状において、これらの技術がGDSSで用いられる際に、以下のような問題点がある。

- コンピュータを使用する際のアクセスのし易さや柔軟性（ネットワーク通信上の問題）
- 表示技術（ディスプレイの解像度の問題）
- グラフィックスの能力（表示速度と表示可能な大きさの問題）
- モデル化や意思決定分析用のソフトウェア（高度なモデル化の問題）
- データベースとモデル化を統合する能力（モデル化機能のリンクの問題）

② GDSS のパッケージの問題

前述したGDSS 技術を構成する４つの要素がパッケージとしてうまく組み合わされなければ、GDSS のアプリケーションはうまく働かない。ハードウェアやソフトウェアの技術が無ければ、GDSS は夢のようなもので実現不可能である。しかしそれだけでなく、技術的な熟練や組織としての安定、会計上の安定、クライアントや顧客に対するデモ効果等がパッケージとしてまとまらなければならない。このような意味で、GDSS は単に個々の作業を達成するためのツールではなく、生産と結び付いた大規模なコンピュータシステムとして捉えられる。

③ 意思決定活動に対する理解不十分

個人やグループの意思決定活動に係わる研究は、比較的新しい分野であり、意思決定の振舞いについて未解決の問題が多い。認識活動に対する理解が不足していることや、複雑な意思決定状況にある曖昧さ等が原因となって、意思決定の研究を難しいものにしている。これが、意思決定のための支援技術を開発する際に、比較的狭い範囲で合理性を主体に意思決定活動を捉えてしまう大きな理由である。現在のシステムで採用されている理性的なモデルによる捉え方は「現実の世界」の意思決定活動の中では限られたものでしかない。なぜなら、人々が時々見せる、訳の分からない非理性的な振舞いというものを考慮していないからである。たしかに、理性的な捉え方も重要な役割を果たしており、現実の意思決定活動において当てはまることもある。しかしそれは、多くの場合、現実とは似ていない。

Huberによれば、理性的なモデルは単純であるという点で役に立つが、この他に、意思決定に係わる文献から得られた三つの重要なモデル（「政治的／競合的モデル」、「ごみ箱モデル」、「プログラムモデル」）があると述べている。そしてGDSS の技術が、この三つのモデルにどの様に対応できるかについて述べている。この三つのモデルにおいて、GDSS の技術は問題や機会を明らかにし、選択した結果についてよりよく把握させ、問題を扱う際の意思決定者の役割を明確にする。「政治的／競合的モデル」では、GDSS は他のグループの存在と、それらのグループの強みと弱みを評価する手助けとなる。また「ごみ箱モデル」では、GDSS は問題が起こったときに、それまで棚上げされていた解決策を取り出す手助けとなる。そして「プログラムモデル」では、GDSS は問題や解決策に係わるさまざまなグループ（あるいはサブグループ）の振舞いを明確にする手助けとなる。意思決定に役立つこのような種類のGDSS は、開発して導入することは可能と考えられるが、今のところは、このようなことができるシステムは無い。

GDSS がもっと効果的に発展していくため（特に理性的な意思決定の振舞いのモデルとは異なる場合に対応していくため）には、現実の意思決定活動について研究を続けていくことが必要である。とは言っても、必ずしも新しいGDSS を開発していく前に意思決定活動の研究が行われなければならないということではない。逆に、GDSS のツールを開発して導入することが、意思決定活動を研究する

上で重要であると考えられている。これらのツールは、正しく設定されて上手にコントロールされれば、意思決定の実験的な研究に大いに役立つ。この問題について討論する最近の研究者会議において、こういった実験的研究はますます一般的になってきている。

(6) GDSS の近い将来

① GDSS の成長する度合

意思決定会議システムや協同研究所システムの高度な適用という面での GDSS の成長は、今後数年間は比較的緩やかであると思われる。これらのシステムは、ユーザベースではまだ少数であり、現在のシステムが持っている機能からみれば、短い間に広く適用されるということはまだ充分保証できるものではない。しかし、政府や企業の予算に見られるように、この方面への関心はまだ続いているので、開発は引続き行われると思われる。

② 技術

GDSS の技術面で現在抱えている問題の多くは、数年後には解決されていると思われる。コンピュータのプロセッサ技術におけるコスト／パフォーマンスは向上し続け、これにより多くの組織が GDSS 技術を導入するようになると予想される。ディスプレイ技術についても、コスト／パフォーマンスは向上するであろう。データのアクセス・分析、意思決定分析、モデル化等のためのソフトウェアは、今後も開発が続けられると思われる。これらのソフトウェア製品は、マイクロコンピュータ市場への投資が続く中で、開発に拍車がかかるであろう。エンドユーザのアクセスやデータベースの利用が便利になるように、もっとユーザフレンドリなインタフェースが開発されると思われる。

③ 意思決定活動の知識

意思決定活動を把握するという点では、さほど目ざましい発展は望めないと思われる。この方面の発展は、認知科学や管理工学などの分野の発展に依存している。この分野においては、ブレークスルーがあるかも知れないが、急速な進歩は望めそうもない。意思決定活動の知識の面では、緩かな成長となるであろう。ＡＩ技術が人間の認知機能の研究に役立っているように、GDSS 技術の実験的な導入が意思決定活動について新しい知識を得るために役立つと思われる。

3.2 ワークステーション／ビデオ会議

3.2.1 ビューの共有

原文タイトル： Sharing views and interactions with single-user applications

翻訳タイトル： シングル・ユーザ・アプリケーションによるシェアリング・ビューとインタラクション

著書： Saul Greenberg
(Advanced Technologies, Alberta Research Council, Alberta, Canada)

出典： ACM SIGOIS Bulletin, Vol.11, No.2, April 1990, pp.227-237

本論文では、元来シングル・ユーザ用に設計されたアプリケーションを、特別な「ビュー・シェアリング・ソフトウェア」を通じて物理的に異なるワークステーション間で共有するというコンセプトの基で、ビュー・シェアリング・ソフトウェアの設計と評価において考慮しなければならない役割と責任（ビュー管理、発言権制御、参加者の会議への参加登録、メタ・レベル通信の処理等）について述べている。また背景として、既存のビュー共有システムに関する概観を述べている。

(1) ビューの共有による協同作業能力の向上

ビューの共有とシングル・ユーザ・アプリケーションとのインタラクションによって、人間の協同作業能力は大きく向上する。可能性としては次のようなものが考えられる。

① 参加者は地理的に分散していられる。十分な帯域幅とボイス・チャネルのネットワークがあれば、物理的に離れた小人数のグループが、コンピュータ・ベースの作業をリアルタイムに容易に共有できる。

② 高度なテキスト及びドローイング・ツールがグループで利用できるようになる。紙やホワイトボードのような従来のメディアに比べて、シングル・ユーザ・アプリケーションの共有によって機能が豊富になるばかりでなく、グループがディスプレイ上のオブジェクトを作成、操作、セーブする際の柔軟性が高まる。

同様に、（アイデア・アウトライナのような）専門的なアプリケーションによって、参加者はより作業に適したツールを利用できるようになる。

③ 物理的に互いの邪魔にならないという単純な理由から、ワーク・サーフェスへのアクセスを交代するのが容易になる。

④ 共有ビューは、「肩越し」の相談に有益である。経験者は、新人に問題を指摘させたり、問題解決に「ユーザを導いたり」、新人が自分で行おうとするのを見守ったりすることで、容易に援助を

行うことができる。

⑤ 共有ビューは、聴衆に資料を示したり、聴衆が会議中に資料を参照したり利用したりするためのメディアとして利用できる。

さらに、情報シェアリングや調整活動、協同設計、共同監督、進行記録など、あらゆる協調プロセスが共有ビューの恩恵を受けることができる。

(2) 既存の共有ビューシステムの概観

本論文では、60年代中頃から続いている共有ビューシステムに関わる研究開発の一部について、簡略に述べてられている。ここで取り上げられたシステムは、次のとおりである。

- NLS（1968年、Engelbartと English）とこれを改良したAugment（1982年及び1984年、Engelbart）
- RTCALとMBLink（1985年、Sarin と Greif）
- Cantata（1987年、Chang）
- Shared X（1989年、Garfinkel他）
- Dialogo（1986年、Lantz）
- Diamond Multimedia Conferencing System（1989年、Crowley と Forsdick）
- Rapport（1988年、Ensor他）
- Xerox PARC（1987年、Stefik他）
- Capture Lab（1988年、EDS）
- EMCE （1988年、Garcia-Luna-Aceves他）
- Timbuktu（1988年、farallon）
- VideoDraw（1989年、Tang）
- Telecollaborationプロジェクト（1989年、Corey他）とCruiser（1988年、Root, 1989年、Fish）
- SharedArk（1988年、Smith）

(3) ビュー・シェアリング・ソフトウェアの役割と責任

総合的なワークステーション・ベースのビュー・シェアリング・システムを構築する場合、設計上考慮しなければならない決定が数多く存在する。システムの責任は、次の4つの役割によって果される。

① ビュー・マネージャ

アプリケーションの分散ビューは、WYSIWISの考えに（多少とも）従わなければならず、誰もが自分の画面かウィンドウで同じ画像を見なければならない。ビュー・マネージャは、これらのビューを

同期、伝達する責任を負う。

② チェア・マネージャ

全ての参加者が妥当な方法でアプリケーションとインタラクトできなければならない。アプリケーションは1つの入力ストリームを処理するためにのみ構築されているため、一般に、一度に1人だけに制御を与える発言権制御メカニズムが望ましい。チェア・マネージャは、発言権制御方針の設定や変更の責任を負い、参加者間の発言交代を調整、実行したり、選択した入力ストリームをアプリケーションに送ったりする。

③ 登録マネージャ

登録マネージャによる会議登録は、会議の開会と閉会、会議中の参加者の入場と退席、アクセス・コントロール、会議状況のフィードバックという4つの問題を取り扱う。

④ メタ・マネージャ

参加者は、アプリケーションに実際に影響を与えることなく、ジェスチャや注釈によってアプリケーションの「周辺」で話し合いができなければならない。メタ・マネージャは、このようなメタ・インタラクションの分離と制御に責任を負う。

各マネージャの役割と関係を、図3.2-1と図3.2-2に示す。図3.2-1は、同機種のターミナルベースのコンピューティング環境を示している。この場合、WYSIWISなビュー・マネージャを作ることは比較的簡単である。アプリケーションの出力ストリームを分岐して、ストリームのコピーを他参加者のターミナルに送るだけでよい。接続された全参加者が最初のターミナルや画面構成と互換性のあるターミナルを走らせていれば、各画面は同じ画像を表示する。

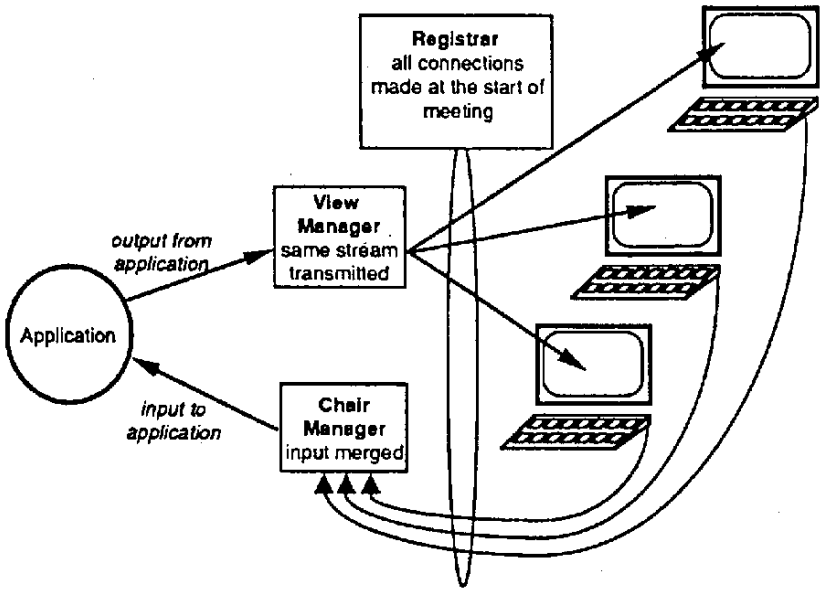


図3.2-1 同機種のターミナルベースのコンピューティング環境用の簡単なビュー・シェアリング・システム。明示的な発言権制御や参加者の自由な入場や退席、メタ・レベルのコミュニケーションはない。

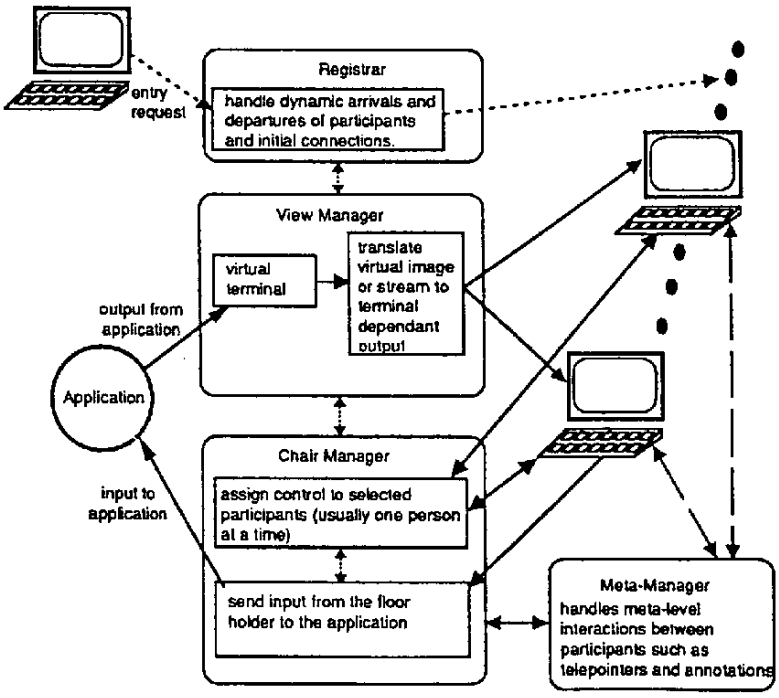


図3.2-2　異機種のターミナル、発言交代、ダイナミックな登録、メタ・レベル対話を処理するより複雑なビュー・シェアリング・システム。

これに対して図3.2-2は、異機種のターミナルで、互換性がないコンピューティング環境を示している。この場合、ビュー・マネージャはアプリケーションに「仮想ターミナル」宛てに書かせてから、出力ストリームを翻訳し、異なるターミナル全体に同じ（又は近似の）画像が現れるようにする。

(4)　障害と問題

共有ビュー・システムに関わる障害や問題点には、次のようなものがある。

①　規格

あらゆるビュー・シェアリング・システムや、ひいてはマルチユーザ・システムで最大の制約となるのは、プレゼンテーション規格がないことである。テキストベース・ターミナルのビュー・シェアリングは、ASCII規格とターミナル・ドライバの総合データベースがあるためかなり単純だが、現在のワークステーションが採用しているウィンドウ・システムの多くは、1つか2つ注目に値する例外はあるが、独自のものである。オフィス内やオフィス間のワークステーションの多様性を考えると、全ての参加者が同じウィンドウ・システムを使ったりこれに精通したりすることはなさそうである。

②　ネットワークとマルチメディア

既存の共有ビュー・システムの多くは、ローカルエリア用や専門化された高帯域幅ネットワーク用に構築され、遠隔地の参加者を一堂に集めるという目的にはそぐわないように思われる。通常の電話

回線では遅すぎるし、特別な高帯域幅接続は利用できないかコストがかかりすぎる。技術が発展して高速回線が安価になるまで、マルチメディアを含むリアルタイムのビュー・シェアリングは、遠隔オフィス間の重要なコミュニケーション・デバイスとはならないだろう。

③　ウィンドウ・シェアリングの技術的問題点

効果的なウィンドウ・シェアリングの設計には、様々な技術的問題がある。例えば、カーソルは普通ウィンドウ・マネージャの考えで動かされるので、1つのカーソルの共有や複数のカーソルの表示は、システムの根幹に大きな変化を与えることになる。また、画面に個人のウィンドウや1つの会議に接続された多重共有ウィンドウ、そして複数の会議が混合して表示されることになれば、システムは所属の同じウィンドウを識別するだけでなく、関連ウィンドウをまとめて管理するためのスキーマを持たなければならない。

④　データ・シェアリング

共有ビューを通じてどのようにデータを共有、操作するかという問題もある。アプリケーションが1人の参加者のデータ空間（すなわちその人のファイル空間内）で動くとすると、その人のファイルが他の会議参加者の故意又は不注意でダメージを受ける危険性がある。さらに、他の参加者はその人のアクセス許可によって制約を受けるが、このため他参加者が会議に参加したり会議へ自分の作業結果を持込んだりすることが防げられる可能性がある。アクセス・コントロールは、アプリケーションがグループの所有するデータ空間内で動く場合でも、問題として残る。

⑤　人間的な問題

効果的なビュー・シェアリング会議については、様々な非技術的問題がある。まず、すべての参加者にその共有アプリケーションを使った経験があるとは限らないことである。エディタを共有する際の問題を考えてほしい。何人かがシステムにいるとして、それぞれが自分のエディタがよくて、他の人のエディタを知らない場合である。自分のエディタが共有用に選ばれなかった参加者は、会議では低レベルのメンバになってしまう。1つのエディタで合意が得られたとしたとしても、カスタマイズされたバージョンに慣れていれば問題が起きる。

二つ目は、共有ビューを個人化する方法がない点である。WYSIWIS は、参加者の役割や知識、能力が大きく異なる場合、制約過剰になってしまう恐れがある。技術者が使うビューは、（例えば）管理職が理解するには細かすぎることがある。このような基本的な制約は、アプリケーションに協同作業認識を持たせることによってしか解決できない。

三つ目は、会議のダイナミックスについて我々はほとんど何もわかっていないことである。フェース・トゥ・フェースの会議における発言権制御は、非常に微妙ではあるが自然なボディ・ランゲージによるところが大きい。会議に招いたり参加したりする際の人間的な慣習も微妙である。これに対して、共有ビュー・システムの明示的な発言権制御機構や登録スキーマは、不自然で制約が多い。

3.2.2 Colab

原文タイトル：	Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings
翻訳タイトル：	黒板を越えて：会議における共同作業・問題解決のコンピュータ支援
著書：	Mark Stefik (Xerox Palo Alto Research Center,Palo Alto,CA) Gregg Foster (Xerox Palo Alto Research Center,Palo Alto,CA) Daniel G.Bobrow (Xerox Palo Alto Research Center,Palo Alto,CA) Kenneth Kahn (Xerox Palo Alto Research Center,Palo Alto,CA) Stan Lanning (Xerox Palo Alto Research Center,Palo Alto,CA) Lucy Suchman (Xerox Palo Alto Research Center,Palo Alto,CA)
出典：	Communications of ACM,Vol.30,No.1, January 1987,pp32-47

本論文では、フェース・トゥ・フェース（対面型）の会議における協調問題解決のためのコンピュータ支援を研究することを目的として、Xerox PARCにおいて開発されたColabと呼ばれる実験的会議室について概観されている。

Colabで利用するために開発された会議ツール（Boardnoter、CognoterおよびArgnoter）、そのために開発したコンピュータの要素技術及び言語サポート、Colab会議から得られた観察結果及び現在追求中の研究問題について述べられている。

(1) Colabの概観

Colabでは、コンピュータは対面会議における協同作業を支援する。Colabは、LANで接続されたパーソナルコンピュータを用いた２～６人の小さいワーキンググループ用に設計されている。設計する際には、従来の会議室で馴染んでいる要素が使われた。Colabプロジェクトの主目的は、コンピュータ科学者である本研究者自身の会議をより効果的にすることにあり、コンピュータツールが会議の進行にどの程度影響を及ぼすかについて広範囲な研究を実施する機会を提供することである。

(2) 協同作業用のツールの要件

会議ツール（会議におけるグループの対話及び問題解決を支援するプログラム）の基本的必要条件は、参加者全員に対して共有できるインタフェースを供給することである。このようなマルチユーザ・インタフェースは、会議参加者がコンピュータメディアを通じて、容易かつ即座に互いに対話できることを意図したものである。

ここで、会議ツールの重要な概念を表現するために、WYSIWYG（What You See Is What You Get）と似た表現で、WYSIWIS（What You See Is What I See）という用語を定義した。これは、参加者全員に共有情報の一致したイメージを提示するということである。会議参加者全員が全く同じものを見て、他の者が指し示している箇所を見るならば、会議ツールは、まさにWYSIWISである。

① 緩和されたWYSIWIS

厳格なWYSIWISは全員のディスプレイに同じイメージを表示することになるが、実際上、これはあまりにも制限となることがわかり、代りにWYSIWISの概念を緩和して使用した。例えば、グループの全員がアクセスできるパブリックの対話型ウィンドウと、アクセスが限定されるプライベートウィンドウ（例えば、電子メールなど）が区別できることは便利である。

また、ポインタ表示を画一化しないことも厳格なWYSIWISの概念に反している。会話中にものを指すことは効果的な方法であるが、参加者全員のカーソルを表示するとかえって注意散漫になりがちである。必要な場合のみポインタを見えるようにすることは、WYSIWISからは外れるが効果的である。

WYSIWISの緩和の別の例として、パブリックウィンドウが各自のスクリーン上で違う場所に表示されることを認めている。この場合パブリックポインタは、ウインドウ内の相対的な位置を判断して表示される。これは、スクリーン上の位置によってものを参照する可能性を犠牲にするが、個人ごとのスクリーンレイアウトを可能にする。

② 並列処理の支援

他のプロセスと同様に、会議は、いくつかのことが同時に実行されることにより効率的になる。Colabツールは同時行動の支援を目指しているので、並列処理を行うためには活動をどのように分解して支援するかという問題が、ツール設計の際の鍵となる。並列処理を行うためには、タスクは、グループの各メンバがある程度独立して実行できるような適切なサイズの操作に分割されなければならない。操作が小さすぎるとあまりにも相互依存し、その干渉によって基本的な並列処理までが妨げられてしまう。例えば、共有テキストを作る場合には、個々のキー入力の操作のレベルまで分解すべきでない。とは言っても、操作が不必要に大きいと、協同で作業することができなくなってしまう。

また共有するオブジェクトを同時に扱えるようにすることも、衝突の原因となる。その際には衝突解決の方法が必要になることもあるが、人為的な強制で解決できる場合もある。衝突発見システムやビジーシグナル機能は、誰かがある項目を既に編集しているとか使用中であるということを利用者にグラフィック的に警告する（例えば、使用中の項目は他の全てのスクリーン上では灰色になる、など）。

(3) Colabにおける会議ツール

① Boardnoter

Boardnoterは打ち解けた自由形式のスケッチを重視したインフォーマルな会議のためのものであり、黒板の機能を厳密にまねている（図3.2-3）。Boardnoterを使って描くためには「チョーク」を使い、消すには「黒板消し」を、文字を書くためには小型の「タイプライタ」を使い、そして、指すには「ポインタ」を使う。Boardnoterで四角を描くには、単に「チョークをつまんで」４つの点をストロークすればよい。Boardnoterの新しいバージョンでは、複写、移動、拡大／縮小、ゴムヒモ状の線で囲む、グループ化、推敲（整理）などの機能を追加し、あらかじめ準備されたイメージを使ったり、拡大縮小をすることができるようになるので、通常の黒板よりも優れたものになる。

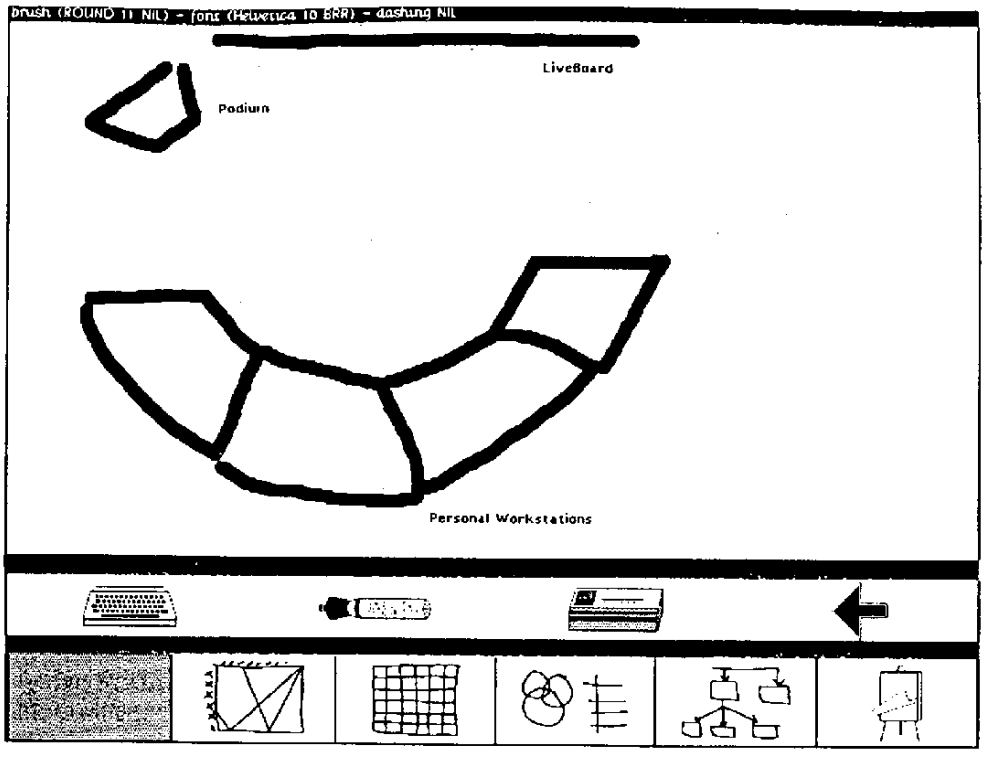


図3.2-3 Boardnoter のスクリーンイメージ

② Cognoter

Cognoter は、一緒にプレゼンテーションの準備をするためにアイデアを整理するColabツールである。そのアウトプットは、注釈が付いたアイデアのアウトラインとそれに関連した文章である。

プレゼンテーションの計画には、アイデアは何で、どのアイデアを一緒にして、どのアイデアが先にきて、プレゼンテーションの順番はどうすべきか、最後に、どのアイデアを除去するのが正当かをグループが決めることが必要である。

Cognoterは、１つの会議を、各々異なった活動である３つのフェーズ（ブレーンストーミング、整

理、評価）に分ける。ただしグループは各フェーズを進める際に、そのフェーズに与えられた活動しかできないわけではない。例えば、ブレーンストーミング（第１のフェーズで重視される）は、最後のフェーズにおいても行うことができる。

多くの点で、Cognoterは、ThinkTankなどのツールとは全く異なるプロセスを支援する。Cognoterが複数の参加者が同時に利用できるよう設計されているという最も明白な違いもあるが、それ以上の特徴として、Cognoterは思考プロセスをより小さく、段階的に効率よく行えるステップに分割するという点がある。Cognoterがアイデアの生成と整理のタスクを分けているのに対して、ThinkTankでは、アイデアはアウトラインでしか整理できない。Cognoterはまた、リンク形成操作によってアイデアの部分的整理が一歩一歩洗練されていくような段階的な整理ができる。アイデアの移動操作やグループ化操作は、わずかのリンクで能率的にアイデアを整理することができる。

ただし、Cognoterではプレゼンテーション計画プロセスの重要な部分について明瞭ではないところがある。例えば、Cognoterは、適切な専門的レベル、論文の目標、アイデアの削除、整理の主張については関与しない。Cognoterを修正してそのような点を明瞭な状態にすることはできるだろうが、これは本ツールの有効範囲外である。

(i)　ブレーンストーミング

ブレーンストーミングのフェーズの目的は、プレゼンテーションのための最初のアイデアを生成することにあるので、グループ内の対話での協力作用を促進し、アイデアが出てくることに干渉や禁止をしないことは重要である。したがってCognoterでは、このフェーズにおいてはアイデアの評価や除去はされず、それらを整理することは重視されない（図3.2-4 を参照）。

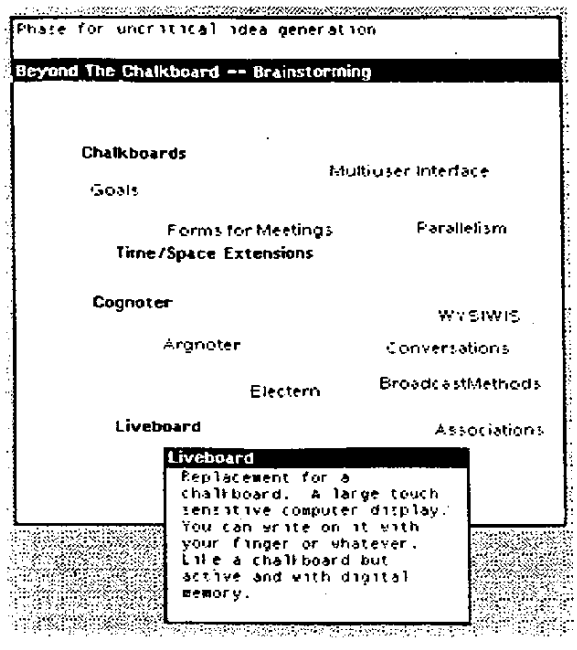


図3.2-4　Cognoter によるブレーンストーミング

その代りに、参加者はパブリックウィンドウの空いているところに、アイデアを特徴付けるキャッチワードまたはキャッチフレーズを入力する。参加者は、いつでも同時にアイデア項目を追加したり文で補足説明できる。項目を移動させることはできるが、一旦書かれた項目は（自分のでさえも）削除できない。文章で補うことは項目の意味を明瞭にし、プレゼンテーションの用語としてはっきりさせるために使われる。一旦入力されれば、全参加者にパブリックに表示され、編集することができる。

(ii) 整理

整理フェーズでは、グループは、ブレーンストーミングフェーズで生成されたアイデアの順序付けを行おうとする。Cognoterを使えば、アイデアの順序は、プレゼンテーション順にアイデアをリンクすることと、サブグループに分割することによって円滑に行うことができる。さらに、項目移動操作は、整理やリンクする前にお互いに項目を動かすことによって、それらを実際に実行する前に討議できる。

図3.2-5 に示されるように、順序は項目の矢印リンクで視覚的に示される。リンクは、プレゼンテーションの完全な順序を決定するために集団で行われる。いくつかの項目を１つのグループにまとめることができ、図3.2-6 に示されるようにグループごとのウィンドウに移動される。グループが形成されるとき、ウィンドウに表示される括弧付き項目（図3.2-6 中の[Tool Dimensions] など）はグループ全体を表している。グループ化された各項目は別のウィンドウに表示される。グループ間でもリンク付けを行うことができる。括弧付き項目からの、あるいは、括弧付き項目へのリンクは、そのグループ全体からの、あるいは、グループ全体へのリンクのように扱われる。このようにグループ分けやグループ間のリンク付けの操作によって、少数のリンクでアイデアの全体的な順序を整理することができる。

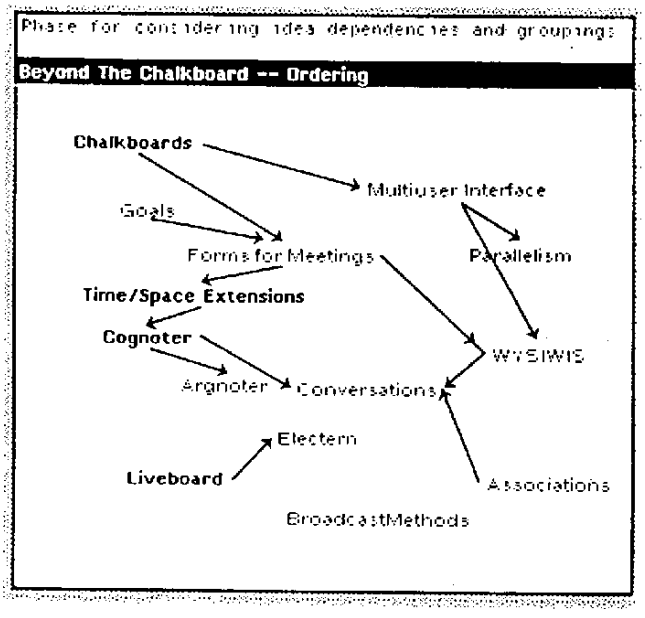


図3.2-5　アイデアの順序付け

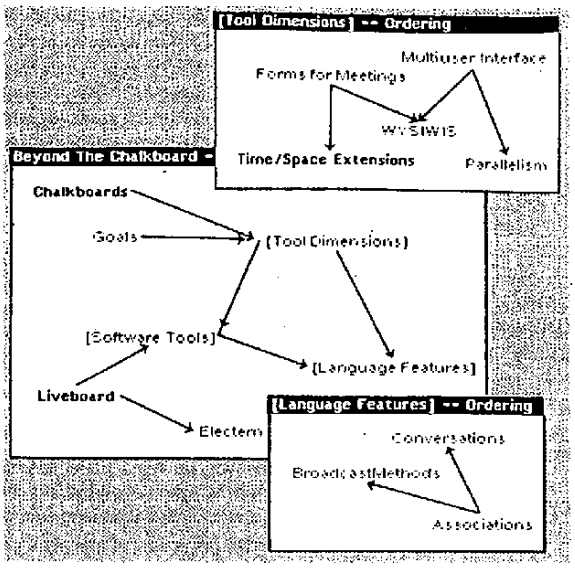


図3.2-6　項目のグループ化

(iii) 評価

3番目の評価フェーズでは、プレゼンテーションの最終形式を決定する。参加者は、アイデアを再編成するために全体の構造をレビューし、欠けている詳細を記入し、枝葉で関係のないアイデアを除去する。

整理フェーズで行われるリンク操作は、通常、2つのアイデアに対する局所的な考察に基づくので、評価フェーズにおいて、リンク付けされたプレゼンテーションの全体を見ることによって、ユーザが全体の構造を評価し、より広範囲の関係（例えば、バランスがとれているかどうか）を考察することができる。

Cognoterは、例えば「何を第I.A.1項に入れるべきであったか」などという質問に答える系統的なプロセスを提供する。プレゼンテーションの出発点は、系統的に確認できる。出発点は、入ってくるリンクがない項目である。そしてCognoterは、アウトラインの準備をし、任意にどのアイデアを整理するかを示すことにより、アイデアを最終的に整理することを助ける。項目の関連図をよく検討することによって、テキストを追加しながらアウトラインが作られる。

③ Argnoter

Argnoter はプロポーザルの呈示・評価のためのColabツールであり、本論文執筆時点では設計及び実装の初期の段階にある。

グループの一人以上のメンバーが何らかのプロポーザル（代表的なものは、プログラム計画または研究進行計画）を持っているとき、プロポーザル会議は開かれる。そのとき会議の目標は最も良いプロポーザルを選ぶことになる。白紙状態で始まるCognoter会議に対して、プロポーザルは、会議の前に少なくともある程度は案出されている。Argnoterの参加者は、すでにこれらのプロポーザル作成にエネルギーを注ぎ込んでいるで、会議にはより大きな争議及び不一致の可能性がある。したがって、これらの会議では、反対意見の発見、理解、評価が意思決定の本質的部分となる。

プロポーザルを展開する際、通常その提案者は、何が要求され、何が可能であるかを正確に知らずに始める。彼らは、その時点での目標に基づいて提案を展開し、何が可能であるか学びつつ目標を明確にしていく。協同考案作業においては、メンバ間のコミュニケーションにより、目標と提案の間のこのような相互作用や緊張は、最後まで続けられる。このとき、提案目標は必ずしも最初から一致するとはかぎらない。通常は目標を推敲していくことも協同プロセスの一部であり、提案を段階的に考案・選択していくこともまた協同プロセスの一部である。

Argnoterによる会議は3つの明瞭なフェーズ（提案、議論、評価）によって構成される。

(i) 提案

提案フェーズでは、プロポーザルは明確に定められる。各プロポーザルは、短いテキスト、また必要であれば図で記述され、その特徴または機能に応じて名前が付けられる。Argnoterでは、プロポーザ

ルは、プライベートでもパブリックでも可能なプロポーザル「フォーム」と呼ばれる一まとまりのウィンドウ上に作られ、表示される。パブリックのプロポーザルフォームは、WYSIWISである。一方、プライベートフォームは、それをコントロールする参加者のマシンにのみ表示される。プライベートフォームは、作業を共有せずに、各参加者が新しいプロポーザルを見たり作成することができる。このとき、会議で考察中のプロポーザルは、別のウィンドウで見ることができる。新しいプロポーザルは、現存するものを修正したり、複数のものから特徴を結合することによって作られる。新しいプロポーザルは、その親プロポーザルからテキスト、図及び判断理由を自動的に継承する。

(ii) 議論

議論フェーズでは、個々のプロポーザルを選択する／しない理由の呈示を行う。これらの理由は、書き留められなければならない。Argnoterの場合には、ただ気に入ったプロポーザルに賛成、残りは反対というのではなく、参加者が全プロポーザルに関して賛否両方の判断理由を書く。したがって、全てについて賛否の判断理由を見て熟考することになるので、参加者は注意深くそれらを考案する時間をとるようになる。

議論フェーズでは比較を目的として、互換性、コスト、開発期間、効率、実現可能性、簡素性、及び、有用性などのカテゴリで、各プロポーザルを賛成反対に分類分けできる。コンピュータ支援で、これらのカテゴリに基づいてプロポーザルを比較した補助用の表を自動的に作り出すことができる。

議論段階では、参加者は、判断理由を追加したり、現プロポーザルを修正しても良い。そうすることにより、アイデアを協力して出すことが促進され、プロポーザルや理由付けを協同で行う役に立ち、プロポーザル間で共通した理由をシステマチックに見つけだすことができるようになる。

(iii) 評価

評価フェーズでは、まず個々の議論の背後にある前提を考察する。Argnoterにおける前提は、判断理由に関する判断理由として表現される。たとえば、「これは労働コストを無視できる」という判断理由は、「このプロポーザルは安価である」という判断理由に基づいている。Argnoterを使えば、これらのさまざまな判断理由を結び付ける議論の構造を見ることができるようになる。

会議参加者は、判断理由の妥当性についてしばしば意見が合わない。Argnoterでは、明確な「blief set（信念セット）」によってこれらの差異のモデル化を行おうとしている。blief setは一組の判断理由を、有効（信じられる）または無効（信じられない）に分類する。

blief setを明確にすることにより、Argnoterが一種の議論スプレッドシートとして使えるようになり、プロポーザルを特定の信念のセットに関連付けて見たり評価することができるようになる。プロポーザルの表示は、プロポーザルに関する議論を整理し、前提を調べ、特定のblief setに基づいたそれらの議論を表示することによって生成される。複数のblief setを共存させることができ、どんな参加者でもblief setを作る（または特殊化する）ことができる。blief setは、異なる観点（<例>保守派対自由主義

者、開発担当対マーケティング担当）の特性を示すことを意図したものである。

(4) インプリメント

Colabのツールをインプリメントするに際して、以下の点を考慮した。

① データベースを維持するための制御方式

Colabデータベースを維持するためにいくつかの制御方式（集権モデル、集中ロックモデル、協力モデル、依存発見モデル、移動ロックモデル）の実験を行い、この結果、協力モデルによるアプローチを採用した。

協力モデルでは、各マシンはデータベースのコピーを持ち、変更を行う場合には、同期をとるのではなくその変更をブロードキャストすることで行われる。このアプローチは次のような競合状態を本質的に持っている。２人の参加者が同時に同じデータを変更すると、どちらの変更が最初に有効かを決める競合が起こり、各マシンで結果が異なってしまうことがある。

これらの欠点を和らげる要因として次のものがある。第１は、データベースを変更する順序は、ほとんどの場合、行われた順番に無関係であるということである。さらに、Colab参加者はその問題を知っており、言葉による合図を使って彼らの行動を調整する。したがって参加者が同じデータを同時に変更することはまれである。

第２の要因は、分散データベースの完全性を保証するメカニズムとは別に、Colabにおいては、参加者の活動を調整し、会議が本来持っている作業の分割と合意達成の両方のための社会的な仕組みを支援するメカニズムを提供する必要があることである。そのようなメカニズムの１つとしてビジー信号が挙げられる。この信号は現在使用中のスクリーン項目を灰色に変え、その項目を変更しないように他の参加者に警告する。しかし、誰かがその項目を作業し始める瞬間とビジー信号が他の人に伝達される時間との間に必然的に遅延があるため、次の参加者が矛盾した修正を開始してしまう恐れがある。この場合でも、ビジー信号は２人の参加者が矛盾した作業していることをすぐに発見できることを保証する。

② 言語サポート

Colabソフトウェアは、Ethernetで接続されたXerox Lispマシン上に構築されている。ソフトウェアはLoopsで記述されている。LoopsはLispにオブジェクト指向を持たせたものであり、データを持つことができるオブジェクトでプログラムが構成されているという点でSmalltalk-80に似ている。計算結果はオブジェクトとしてメッセージを相互に送る。Loopsはパーマネント・オブジェクトの概念をサポートしている。パーマネント・オブジェクトとは、マシン間でユニークであることを保証された識別子によって指定できるオブジェクトである。これらのパーマネント・オブジェクトのバージョンはいくつかのマシンに同時に存在し得る。アソシエーションは複数のマシン上で同じオブジェクトを示す一組

の表現である。個々の表現はアソシエートと呼ばれ同じユニークな識別子を持っている。

Colabでは、マシン、Colabツール、および問題解決のために一緒に働いている参加者の３つを一まとめにして、会談(conversation)という言葉で表現している。新しい参加者が会談に加わったとき、参加者全員は新しい参加者が加わったことが分かり、新参加者は他の参加者の存在が分かる。新参加者のマシンはデータベースを表すオブジェクトのコピーをもらう。

会談では、通信はシステムの設備及びプログラミング抽象の組合せにより実現され、Ethernet上のプロトコル階層によってサポートされる。このメカニズムはリモートプロシジャコールのプロトコル上に実現された。この最上層に、リモートマシンのオブジェクトにメッセージを送信する仕組みと、会談でのあるオブジェクトの全アソシエートにメッセージを送信する仕組みを付け加えた。

(5) 予備観察

① 記述プロセスの構造

Cognoterの設計は、発想の早期段階から方針または概要の生成および展開を通じて最終プレゼンテーションに至るまでの記述プロセスに関する一連の推測を反映している。しかし、実際にCognoterを利用してみると、設計とプロセスの適合の点ばかりではなく、その間の微妙な不連結が明らかになった。

Cognoterを使った初期のセッションでは、整理フェーズに移行する前に、メンバーがブレーンストーミングウインドウでアイデア間の関係を表示しようとして空間的なグループ分けを始めてしまうことが分かった。

整理及び評価プロセスは、ブレーンストーミングで生成されたアイデアの組み合わせが完全であったかどうか見るのを容易にした。設計する際の最初の前提では、アウトライン作成ツールは評価フェーズが完了した後で使用すべきものであったが、実際には参加者は、アウトライン作成ツールが現在の構造の中間状態を表示する際にも便利なツールであることに気づいた。これらの観察により、もともと前提にしたこととは異なったプロセス間の「繋がり」があることが、ほのかに分かった。

② 協同作業の維持

全ての参加者がディスプレイや共有データへ平等にアクセスすることによって、Colabのインタフェースは役割の柔軟性を高め、１人の参加者によって活動が規制されてしまうのを防止している。

しかしながら初期のセッションでは、当時の技術上の制約条件は単なる協同作業の限界ばかりではなく、いくつかの点でリソースの限界をも示した。とくに記述技術は同時に１人しかテキストを入力できないことで、グループで共通の文章を維持する際に１つの（すなわちその人の動作による）視点を共有することになる。

初期のCognoter会議において、共有視点を維持する作業は会議活動の活発度合いの変化において明

白だった。とくにアイデアが練られる整理フェーズの間、参加者は数分の間、当面の目標の説明や行動の短期計画の作成を言葉で対話し、グループが彼らの「宿題」にした後、しばらくの間熱心にタイプする傾向があった。並行編集を始めて数分後、人々は他人が何をしていたか分からなくなり、他人が次に何をするかを見失う。するとグループはシステムとのインタラクションを止め、再び彼らがどこにいて何をすべきかを議論するといった状況が見られた。

3.2.3 Project Nick

原文タイトル： Project Nick: Meetings Augmentation and Analysis

翻訳タイトル： Nick プロジェクト：「打合せ」の支援と分析

著書： Peter Cook (Software Technology Program, MCC, Austin, Texas, USA)
Clarence Ellis (Software Technology Program, MCC, Austin, Texas, USA)
Mike Graf (Software Technology Program, MCC, Austin, Texas, USA)
Gail Rein (Software Technology Program, MCC, Austin, Texas, USA)
Tom Smith (Software Technology Program, MCC, Austin, Texas, USA)

出典： ACM Transactions on Office Information Systems, Vol.5, No.2, April 1987, pp.132-146

Nick プロジェクトは、MCC の Software Technology Program の中の Design Interface Groupによって行われたシステム設計の領域における打合せの分析と支援に関する研究である。

本論文では、Nick プロジェクトで現在検討中のいくつかのアイデアとそれらの実現方式が紹介されている。まず、いくつかの定義と研究目的が述べられ、つぎに、本研究の構成要素について説明される。ここでは打合せ進行モデルを提示し、構成要素はこのモデルに沿って議論されるが、このモデルは Nick プロジェクトおよびシステムの本質となっているものである。さらに、打合せという概念およびポットの理論について正式な定義をおこなうとともに、この定義を用いて打合せを分析し、典型的な「打合せの問題点」を発見する方法が概観される。また、プロジェクトの打ち合わせ内部での側面が説明される。電子的な打合せ支援機能が説明され、打合せ情報マトリックスについて議論されている。

(1) 定義と研究範囲

打合せとは、二人以上の人間が協力して行う、構造化されたコミュニケーション活動であると定義される。これらの集まりは、法人団体におけるあらかじめ予定された正式な会合(取締役会等)から、特定の問題を話し合うため二人の同僚が非公式にアドホックな形で打合せをすることまで、幅広い範囲

を持つ。打合せの参加者は空間的及び時間的に同じ場所にいる場合もあるし、いない場合もある。つまり、分散会議はここでの定義の中に含まれている。また、非同期の電子会議およびメールシステムは、会議参加者が実際に同時点において打合せ資料を見ていなくても良いような打合せの形式である。

本研究では直接の面談による打合せを対象としている。その理由の一つは、MCC における一般的な関心と資源である。また、ここでは主として小規模の設計打合せを対象としている。つまり、最小限二人の設計者が解決策を検討しているような場合から、最大 50 名程度の人々が集まり、ネゴシエーション、問題解決、情報交換を行う場合までを想定して、打合せ活動を支援することを対象とする。

また、本研究は主として、設計チームの打合せを対象とする。設計チームの活動には、ブレーンストーミング、探索、設計構造の定義、問題の分析、タスクの割当て、問題解決などがある。設計に関する打合せは、効率性と有効性についてはよく分かっていない。これは特に、責任がまだ明確になっておらず、組織構造が変わりやすい設計の初期段階において言えることである。

(2) 主要構成要素

本研究では、理論の構築とシステムの構築の 2 つの面のアプローチをとっている。この両面は、並行して進められ、また互いに相補いあうものである。

ここでは、まず理論構築の側面から、設計プロジェクトのビデオテープを観察することや筆者自身のグループにおける打合せにおいて実験することによって得られた洞察を基に、打合せの主要要素を取り出しモデル化を行っている。これを以下に紹介する。

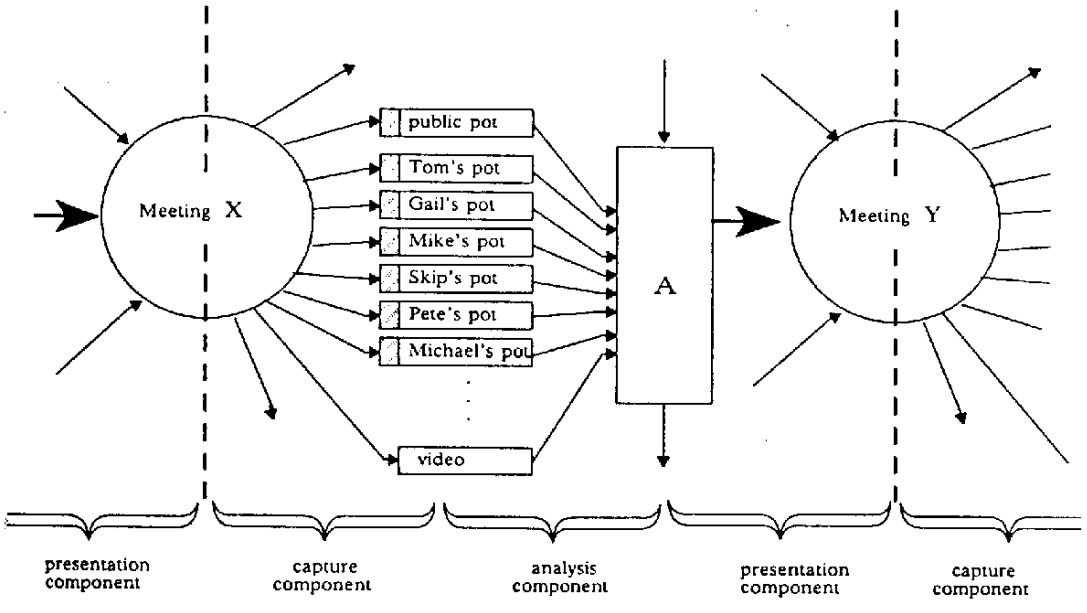


図3.2-7 Nick プロジェクトの概要

図3.2-7 は、本研究の主要要素であるところの獲得コンポーネント、分析コンポーネント、プリゼンテーション・コンポーネントを描いたものである。一般的に、ある打合せ（図3.2-7 では打合せ X）で生成された情報は、獲得され、分析され、次の打合せ（図3.2-7 では打合せ Y）で利用されたり、プリゼンテーションされたりする。図中の円は打合せを意味し、箱は情報の記憶装置（および処理）を意味する。矢印は、情報の流れを意味する。この図では、典型的な打合せ（打合せ X）は多くの情報源を持つことが示されている。打合せ X の円への矢印の出所が示されていないのは、これらの情報源が指定されないことを示している。この円から出る矢印は、出力情報を示し、矢印の終端にある箱は、情報のリポジトリである。情報リポジトリのそれぞれは、打合せ情報の或る一部を獲得する。避けられないことであるが、打合せで生成された情報の一部は、獲得されないままで終わる。打合せを撮影したビデオテープでさえも、参加者全員の表情までは記録しない。したがって、打合せ X の円から出る矢印の中には、いかなるリポジトリにも届かないものがある。

このモデルでは、知的で高度な情報解釈選択機構を持つリポジトリとそうでないものとの間を区別していることが重要である。高度なフィルタを持つリポジトリを区別するために、ポット（瓶）という用語を作った。図においてポットは、入り口に斜線を付けた箱によって示されている。「ビデオ」というラベルを持つ箱は、技術的な支援機構の例（ビデオテープもしくはビデオディスク）である。これは打合せにビデオテープもしくはビデオディスクを利用したことを示している。「パブリック・ポット」というリポジトリは、打合せの過程でホワイトボードに記入された情報、あるいは公式な議事録に記録された情報を示す。これは、パブリック・メモリとも呼ばれ、会議の過程ですべての参加者が見ることができるものである。

図3.2-7 の「A」とラベルされた大きな箱は、分析コンポーネントを示す。このコンポーネントは多くの異なったリポジトリからの情報を総合し要約する方法を研究するものである。現在、この分析は打合せの途中ではなく、打合せが終了した後に行われる。このコンポーネントには、打合せのシンタックス（打合せの進行過程）およびセマンティックス（技術的内容）を獲得し、理解し、改善するための情報分析が含まれる。また、効果的なプレゼンテーションを行うための、入力情報の変換方法の研究も含まれる。例えば、ビデオテープは有効な獲得手段であるかも知れないが、入力情報を注意深く再編集しない限り、プレゼンテーション手段としては時間がかかり不便であるように思われる。打合せの過程の特定の部分にランダム・アクセスできるビデオディスクは、より優れたテクノロジである。典型的な変換手段としてはタイプ入力から音声出力への変換がある。

箱 A から出ている太い矢印は、分析され変換された情報を示す。この情報は、電子的な形で入手可能であり、図では打合せ Y として示されている次の打合せにおけるプレゼンテーションのための情報源として利用可能である。プリゼンテーション・コンポーネントは、コンピュータ化された情報の表示だけでなく、ブレーンストーミングや、黒板の前での創造的かつアドホックな「おしゃべり」のた

めの、構造化されたプリミティブを利用可能にすることも考慮している。設計者たちはしばしば、箱、円、リスト、マトリックスなど、或る種の構造体を利用する。

(3) 打合せの分析

本研究では、打合せ要素の間のスタティックおよびダイナミックな関係を説明するために、ポット理論を開発し洗練しつつある。打合せの概念を定式化する定義と、ポット理論の説明を以下に述べる。

打合せとは、8つの要素の組 M = < G, R, P, I, A, S, C, E, >であると定義する。それぞれの要素は次のとおりである。

G: 打合せの目標の集合
R: 打合せで利用する資源の集合
P: 打合せの参加者の集合
I: 打合せで操作される情報
A: 打合せの過程で発生する活動の集合
S: 打合せのタイプ、議題、議事進行規則等、打合せの構造
C: 打合せが行われる状況
E: 打合せに周辺的な影響を与える外部要因

集合Gは打合せに参加する人たちが心に抱いている目標ではなく、組織としての目標である。前者は、参加者集合(P)の一部分である。集合Pの個別要素は、参加者のプロフィールであると考えることができる。つまり、各参加者の目標、能力、および信念の記述である。集合Pについてもう一つ注意しておくべきことは、参加者とは人間である必要はないということである。

資源Rは、打合せの間に利用され、あるいは利用可能である実体的なオブジェクトである。例としては、ハードウェア、ディスプレイ装置、あるいは会議室の椅子やテーブルである。他方、情報Iは、文書、議事録、および打合せの多種多様な入力データ、出力データである。

Aは打合せの間に発生する活動の集合であり、コミュニケーション、知覚、相互作用を含む。これらにはエピソードのようにマクロなレベルで記録することができるものがあるし、あるいは打合せ相互作用グラフのようにより詳細なレベルで記録されるものもある。また、身振りなどのような非言語的行動もある。

打合せの構造Sは、議題、議事進行規則、あるいは会議の予定時間などを含む。また、議長、書記のような役割と、どの人がどの役割を務めるかという指定が含まれる。

打合せの状況Cとは、非常に豊かな構造を持つものである。これはその打合せに関係する組織的構造、

社会的構造、および歴史的、政治的、経済的要因を含む。集合Eは、その打合せに直接関係しない外部的要因を含む。

上記の要素の多くは、打合せの進行と共に、進化発展する。集合I（情報）とP（参加者）は特に変化しやすい。打合せは、I, P, Rを主要入力とする一つの関数と考えることができる。この関数は、G, S, Cの影響のもとに、入力に対して、Aによって記述される或る種の変換を加える。打合せの出力のうち最も目に見えやすいものは、生成されたり変換されたりする情報（I）、および、参加者（P）の知識の変化、態度の変化である。しかし、多くの場合、他の集合も変化する。目標は再検討されるかもしれない。議題は変更されるかも知れない。参加者間の関係は変化するかもしれない。あるいは組織構造について重要な決定がなされるかもしれない。

ここでは打合せにおける抽象的かつグローバルなリポジトリNの存在を仮定し、それをヌーメノン(noumenon)と呼ぶこととする。ある特定の打合せのためのA（活動）およびI（情報）のすべての要素は、このヌーメノンの中において存在すると考える。ヌーメノンへのアクセスは、現象(phenomena)、つまり打合せにおける活動と情報を通じてのみ行われる。そして、如何なる個人もそれらのすべてを吸収することはできない。したがって、個人のリポジトリ（個人は打合せでの出来事に高度な解釈を施すから、そのすべてはポットである）は、すべてヌーメノンの中からフィルタをかけられ取り出された部分集合もしくは抽象化されたものである。二人の参加者の個人的ポットP1, P2の内容を考えて見よう。それは、フレーム、つまり打合せの過程においてこれらの参加者が重要であると認識したところの、構造化された要素を含んでいる。さらに、個人的ポットの要素からヌーメノンの要素への写像（P1→NおよびP2→N）を考えて見よう。これら二つの写像の組合せから、二つの集合P1, P2の間の写像を作ることができる。この合成写像の良さが、これら二人の参加者間のコミュニケーションの良さの尺度であると仮定する。

(4) 打合せの支援

ここでは、一回の打合せの中における支援と分析について考察する。まず、打合せの途中における強力な情報獲得能力の存在について注目することが重要である。この情報獲得機能とフィルタリング機能は、図3.2-8に図解されている。

本プロジェクトの電子的打合せ支援設備は、当初次のような構成であった。

(a) 打合せの参加者一人一人に提供されたパソコン、キーボード、マウス、プライベートな表示画面。

(b) 会議室の中でパソコンを結ぶLAN。このネットワークは、建家全体をカバーする大きな電子ネットワークに接続されており、会議室の中には準備されていない情報に対しても必要に応じて即座にアクセスすることができる。

(c) 参加者全員が見ることのできる共同作業用表示装置。打合せ支援システムの出力を表示するため

のソフトウェアが用意されており、これによってメンバは打合せの結果や進行状態を正確に知ることができる。

(d) 議題情報を表示し、参加者間の意味のあるコミュニケーションを提供し、グループ・メモリへの情報入力を可能とするソフトウェア。

(e) 参加者間の二つのサブチャネル。一つはグループ・プロセス情報用であり、他方はグループの情緒的状態を登録するためのものである。

(f) 打合せに関連した統計情報を獲得し、処理するソフトウェア。これによって打合せの有効性についてのいくつかの側面を定量化することが可能になる。

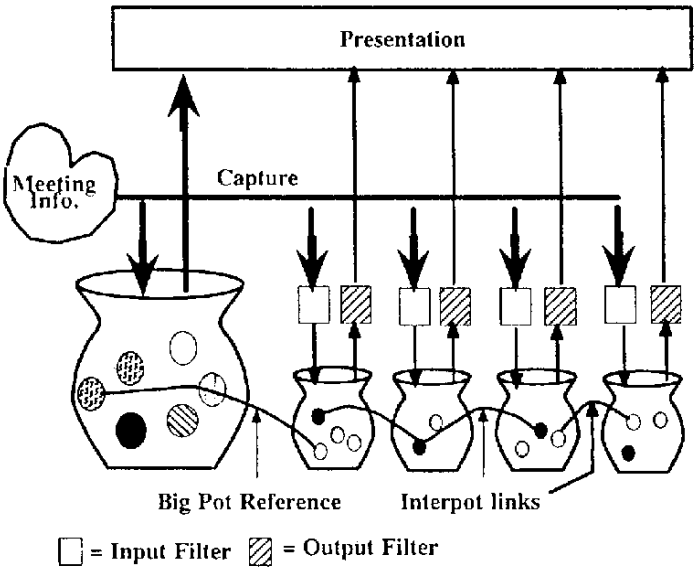


図3.2-8 ポットの理論

多様な打合せの構造のそれぞれにおいて、打合せの継続中に表示され、操作され、獲得される情報のタイプは、どの程度の構造化がなされているか（バイナリ、構造化、非構造化）、また情報の共有がどの程度許されるか（完全なプライベート性保護、サブグループ内部での公開、完全な公開）に対応して分類されるであろう。このカテゴリを図3.2-9のマトリックスに示す。

プライベート情報とは、会議の間に参加者個人によって獲得される情報であり、打合せの席上では他者に開示されない。例えば、これは非常にカジュアルで、正直なメモである場合があろう。これは会議中自由に使用できるパーソナル・ワークステーションを介して獲得される。サブグループ情報とは、会議の間、二人以上の個人によって共有される情報である。通常、会議中秘密のメモを回覧することは悪いマナーであると考えられている。しかしながら、パーソナル・ワークステーションを利用して、一人または複数の他者にメモを送る可能性は、より高度の並列性を持つ新しい打合せスタイル

の開発へ導くかもしれない。

バイナリ情報とは、イエス／ノウ、真／偽など1ビットで伝達することのできる情報である。このタイプの情報は、参加者にとって作成、送信、受信が特に容易である。図3.2-9に示した打合せ情報マトリックスの1行3列目には「投票」とある。これはバイナリのパブリック情報チャネルをサポートする電子装置を利用して行える打合せ活動の一つの例である。打合せ参加者は、自分の意思に応じて、自分のワークステーションのボタンを押すだけで、現在討論中の問題に関する無記名投票を行うことができる。イエス／ノウは、ネットワークを介してパブリック・マシンへ伝達される。投票集計結果は即座にパブリック表示画面に表示される。

	プライベート情報	サブグループ情報	公開情報
バイナリ情報	リストのチェックマーク	指示フラグ	投票
構造化情報	リスト	促進用メッセージ	プレゼンテーション、リスト
非構造化情報	スケッチ図	ノートとコメント	電子黒板上のスケッチ図

図3.2-9　打合せ情報マトリックス

構造化情報とは、定形化されており、コンピュータが即座に解釈することのできる情報（通常は非バイナリ情報）である。このタイプの情報には、リスト、マトリックス、テンプレート、フローチャートなどがある。図3.2-9のマトリックスの2行2列目には、「促進用メッセージ」とある。これは、参加者と進行役の間でやり取りされる定形メッセージの一つである。これはサブグループ・コミュニケーションの一例である。参加者全員がこのメッセージを見る必要はないからである。進行役は、「そろそろ次の議題へ進むべき時ですよ」というメッセージや、「現在の話題について是非発言させて欲しい」というメッセージを受け取る。これらのメッセージは事前に用意されており、一つのキー・ストローク、マウス・クリック、タッチ・スクリーンへの指の接触、あるいはその他簡易なユーザ・インタフェース技術によって起動されるべきである。本システムでは、気分表示メッセージをインプリメントしている。これは誰でも何時でも起動することができる。これは、「退屈したよ」「この打合せは非常に興味がある」「そろそろコーヒーブレークが必要だ」などのメッセージである。これらのメッセージによって進行役は会議を生産的、効果的に運営するための判断をおこなうことができる。或る種の条件のもとにおいては、電子黒板あるいは何らかのパブリック表示媒体上のムードメータを介して、これらの入力メッセージの平均値を公開して表示することも役立つかもしれない。もしそうすれば、これは構造化された公開情報（2行3列目）の一例となる。

非構造化情報とは、定形化されておらず、コンピュータが即座には解釈できない情報である。設計者にとっては、これはしばしば、パーソナル・ワークステーションの画面上に描くプライベートなスケッチ図（3行1列目）あるいは、電子黒板上のパブリックなスケッチ図（3行3列目）の形を取る。

電子的媒体上でスケッチすることの利点は、スケッチによって作られるビットマップは保存、再生、連続的編集（構造体に変換された場合）が可能であるという点である。グラフィカルなソフトウェア設計支援ツールや打合せ用のグラフィックスとアニメーションの開発には特に重点が置かれている。ソフトウェア設計者のためのグラフィカル言語の開発が進歩すれば、これらのスケッチ図をコンパイルし、構造化情報として取扱うことが可能になることが期待される。これによって、打合せ分析のフェーズで利用できる構造化情報が拡大されるであろう。

3.2.4 MMConf

原文タイトル：	MMConf: An Infrastructure for Building Shared Multimedia Applications
翻訳タイトル：	MMConf：共有マルチメディア・アプリケーション構築のためのインフラストラクチャ
著書：	Terrence Crowley (Bolt Beranek and Newman Inc. Cambridge, MA, USA) Paul Milazzo (Bolt Beranek and Newman Inc. Cambridge, MA, USA) Ellie Baker (Bolt Beranek and Newman Inc. Cambridge, MA, USA) Harry Forsdick (Bolt Beranek and Newman Inc. Cambridge, MA, USA) Raymond Tomlinson (Bolt Beranek and Newman Inc. Cambridge, MA, USA)
出典：	CSCW'90 Proceedings, October 1990, pp.329-342

本論文では、MMConf プロジェクトの目的と、採用されたアーキテクチャ、そのインプリメント、さらにこれに付随するアプリケーションに関して紹介されている。

(1) MMConf プロジェクトの目的

MMConf プロジェクトは、分散したリアルタイムのグループ・インタラクションをコンピュータでサポートしようとするものである。本プロジェクトは、投票やブレーンストーミングのためのツールのような会議プロセスそのものを支援するアプリケーションではなく、会議に内容をもたらすためのアプリケーションを取り上げ、共有環境で動作するシングル・ユーザ・アプリケーションのサポートと、本質的な共有アプリケーション開発基盤の提供を目的としている。

MMConf プロジェクトの最終的な目標は、協調作業の種類を制限している距離による障壁を、コン

ピュータによって乗り越える方法を見つけることである。

作業の指針となった想定には次のものがある。

① MMConf は、広範囲な帯域幅と待ち時間特性を持つ広域ネットワークで実行できること。

② グラフィックとインタラクティブ性が高度に発達したアプリケーションをサポートすること。アプリケーションの会議での対話能力は、単一で使った時と同じくらい高いこと。また、公平に全てのサイトでほぼ同じ能力が達成されなければならない。

③ MMConf は、長時間の会議も、電話に似た短い「軽量」会議も同様にサポートすること。

④ コンピュータ遠隔会議には、従来の音声やビデオによる遠隔会議も伴うことになる。MMConf では、従来の電話会議と広帯域ネットワークを使ったパケット交換の音声ビデオ会議システムと共に用いた。

(2) 設計上の問題

従来のグラフィック・ユーザ・インタフェースは、シングル・ユーザを想定している。このようなインタフェースを何人かのユーザに分散させるアーキテクチャでは、コンピュテーション・モデルを集中型にするか複製型にするか、参加者のスクリーン全体を使うかウィンドウのみ使うか、メタ討議を可能にしたまま、どうやって複数のポインティング装置を一致させるかという問題を解決しなければならない。

① 集中型のアーキテクチャと複製型のアーキテクチャ

コンピュータ会議アーキテクチャを設計する上で重大な決定となるのが、集中型にするか複製型にするかである。これによって、アプリケーションの実行が変ってくる。

集中型アーキテクチャでは、アプリケーションには1つのコピーしか存在しない。中央サーバが、ローカルなスクリーンを含む全会議サイトに出力を分配し、各サイトのサーバが中央サーバに入力を送り返して、これがアプリケーションに転送されることになる。

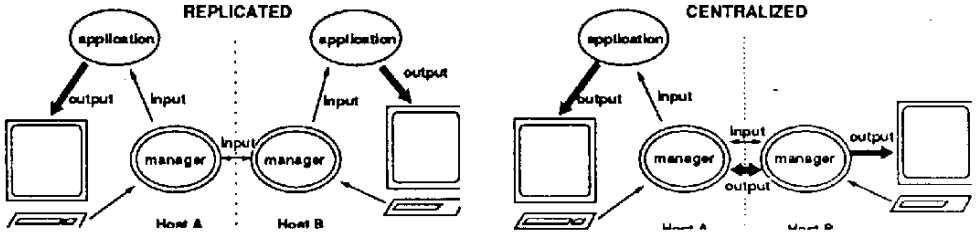


図3.2-10 複製型及び集中型アーキテクチャ。縦の点線がマシンの境界を表す。
複製型アーキテクチャは、境界を越えては入力事象しか送らないが、
集中型アーキテクチャは入力、出力共に分配しなければならない。

複製型アーキテクチャでは、会議の全サイトでアプリケーションのコピーが動作する。各サイトのサーバは、全コピーが入力事象を同じ順序で見られるようにするための何等かの制御メカニズムを使って、コピーに入力を分配する。全てのコピーが同じ入力を見て、状態を同期したまま各サイトで同じ出力を発するため、コピーの出力はローカルにしか現れない。

2つのアーキテクチャでは、集中型の方が本質的に単純である。アプリケーションの変更もいらず、アプリケーションとウィンドウの間に新しい会議ウィンドウ・サーバを挿入するだけでよい。

MMConf では複製型アーキテクチャを使っている。複雑さは増すが、複製型にすることで次のような大きな利点が生まれる。

(a) 性能

複製型アーキテクチャの2つの特性によって、大きく性能が向上する。1つ目は、入力と出力両方ではなく、入力のみをサイト間に分配すればいいので、帯域幅が低くてすむことである。2つ目は、複製型システムは、ネットワーク待ち時間の変化に対する感受性が低いことである。会議の全参加者は、アプリケーションのローカルのコピーとやり取りするので、インタラクティブな性能が高い。

(b) 汎用性

集中型アーキテクチャは、サーバ・ベースのウィンドウ・システムで最もうまく動く。サーバ・ベースのシステムでは、入力事象と出力要求はネットワーク接続を通るので、分配する出力を得るためのハンドルができる。

一方、複製型アーキテクチャもサーバ・ベースやカーネル・ベース・システムを使った場合に同じくらいうまく動く。カーネル・ベースのウィンドウ・システムでは、ライブラリやシステム・コールが入力事象や出力要求を扱うので、出力のハンドルがない。複製型アーキテクチャは、1つの入力ハンドルしか必要としないので、割込みがはるかに容易になる。ウィンドウ・システムには、入力を得るための入口点は1つだが、出力を発生する入口点はたくさんあるのが普通である。

出力のローカルな発生も、異種環境での実行を容易にする。ウィンドウ・システムのまったく異なるサイト間での複製型遠隔会議では、入力事象を翻訳するだけでよい。

(c) 容易なサイト指定インタラクション

複製型アーキテクチャでは、新しいアプリケーションでも共有環境でのローカルなインタラクションを容易に実行できる。集中型アプリケーションでは個々のインタラクションができるが、会議出席者1人1人について別個に事象処理スレッドを維持しなければならない。複製型アプリケーションは、状態の通知を他の会議出席者にマルチキャストするだけで、シングル・スレッドのままでいられる。複製型アーキテクチャの大きな欠点は、アプリケーションの全てのコピーについて同じ状態を保つことが難しい点である。この場合障害となるのは、初期アプリケーション状態の相違、順序の誤った入力事象、非決定的なアプリケーション、会議遅参者の4つである。

また、これ以外に、プロセッサとファイル・システムの速度の差のため、複製型アーキテクチャは一時的な矛盾を経験する。会議出席者がエディタに文書を読みこませた後に、全てのスクリーンで文書が見えるかどうか口頭で話し合う場合（「もう見えた？」）などがそれである。集中型アーキテクチャにも、ネットワークとローカルなウィンドウ・サーバの待ち時間によってディスプレイ出力が矛盾することがあるが、遅れは小さいものであることが多い。

② スクリーン・ワークスペース管理

コンピュータ会議アーキテクチャでは、ユーザのスクリーン全体を管理するか、一定のウィンドウのみを管理するかを選択しなければならない。後者の場合には、どのウィンドウ制御がローカル・スクリーンだけに影響を与えるかも選択しなければならない。ウィンドウ制御には、ウィンドウ・サイズ、スクリーン上の位置、他会議ウィンドウとの相対位置、スタッキング順、アイコン化などがある。

会議ウィンドウのみを管理することで、ユーザは自分のワークステーションで独自の活動を行うことができる。これは、軽量会議が既存ウィンドウに影響を与えないで現れたり消えたりするためにも重要である。MMConf はこのモデルに従っている。

参加者のプライベート・ウィンドウのレイアウトがそれぞれ異なっていて、会議中もそのプライベート・ウィンドウで作業を行っている場合、ウィンドウ位置とスタッキング順をローカル決定とすることが重要であると考えた。アイコン化を独立して制御することで、共有状況が減り、アプリケーションは、アイコン化されているかオープンかに応じて、異なった行動を取る。MMConf では、アイコン化はアプリケーションの全コピーに影響を与える。アプリケーション事象の解釈にウィンドウ・サイズが重要なことが多いが、MMConf では、ウィンドウのコピーは全て一緒にサイズを変更する。

③ メタ討議

分散グラフィック・ユーザ・インタフェースの問題には、メタ討議能力を保持しながら、会議の要素に注意を向けるためのポインティングやスケッチング等複数のポインタを一致させる点がある。コンピュータ遠隔会議システムでは、1つの会議にポインタが1つしか使えないか、会議出席者と同数のポインタを同時に使えるかのどちらかで、いずれのアプローチも先例がある。

MMConf のテレポインタは、会議ウィンドウ内にしか現れない非常に大きな目立つ矢印である。現在、使用中のユーザが真中のマウスボタンを押すことでテレポインタを呼出す。既存のアプリケーションの機能性を排除することのないよう、このテレポインタは通常のポインタ事象も送る。

また、スケッチングについては、透明のウィンドウなら、どんなウィンドウにも重ねられる単純で一貫したスケッチング機能が得られるが、残念なことに、本システムがサポートされるウィンドウ・システムに透明なウィンドウはない。そのため、MMConf ではいくつかの討議向きアプリケーションの中に直接、スケッチング・サポートを加えている。

(3) MMConf アーキテクチャ

MMConf アーキテクチャは、次の 5 つの部分から構成される。

(i) 会議管理と通信

(ii) アプリケーション・プログラミング・モデル

(iii) アプリケーションへの入力に割込む方法

(iv) プログラミング・モデルをウィンドウ・システムに合うようにインプリメントする方法

(v) ウィンドウ・システムに特有の入力事象と MMConf の一般的な事象記述との間のマッピング

MMConf の UNIX 上のインプリメントは、遠隔会議アプリケーションにリンクされたユーザ・インタフェース・ツールキットの会議管理と通信を除いて、その他の全てを盛りこんでいる。

MMConf のユーザ・インタフェース・ツールキットには、会議中に行動を変更する機能が多く含まれている。MMConf のアプリケーションにも、会議認識コンポーネントが入っている。会議認識には、同期確保、サイト間のデータ共有、副作用の制限、並列活動の容易化の 4 つの目的がある。

このツールキットによって、アプリケーションには次のような基本的な会議認識メカニズムが得られる。

① 会議情報

最も単純な照会は、アプリケーションが会議の下で動作しているかどうかである。他にも、どの会議出席者が最新の事象を発したか、ローカルなアプリケーション・ピア（アプリケーション・プロセスの各サイトにおいて対応するプロセス）が発言権を持っているかどうか、会議には何人参加しているか等の照会がある。

② 発言権管理

発言権管理が稼働していると、アプリケーションは発言権をロックして一定のインタラクション中に変更できないようにすることができる。アプリケーションは、発言権がロックされている間、システムが事象を分配すべきかどうかも決定できる。アプリケーションは、ファイル名収集等、1 つのサイトでのプライベートなインタラクション中は発言権をロックするのが普通である。

並列活動をサポートするため、ツールキットによってアプリケーションが発言権管理を禁止して、事象を自動的に分配するかどうか指定できるようにする。発言権を保持しないアプリケーションでは、全会議出席者が同時にアプリケーションとインタラクトでき、各アプリケーション・ピアは、異なる順番で事象が到着するのを見ることになる。これは通常のアプリケーションでは混乱の原因となるが、会議認識アプリケーションは、会議出席者ごとに制御スレッドを維持するため、各事象の起点を使うことができる。

③ メッセージ転送

アプリケーションに固有のプロトコルのための基本的な転送メカニズムは、メッセージを全てのアプリケーション・ピアにマルチキャストする機能である。この基本的な機能の1番上位として、任意のデータを引数としてパスして、登録機能のファミリの1つを呼出す機能がある。この機能は完全な遠隔手続き呼出し機能ではなく、呼出された機能は値を戻すことができない。

どちらの転送メカニズムも非同期で、送信プロセスは応答待ちをブロックしない。受信プロセスは、メッセージを入力待ち行列の事象の1つとしか見ない。

④ ファイル転送

複製型アーキテクチャの欠点に、データ・ファイルも複製しなければならない点がある。会議が始まる前にファイル分配が必要とされることは、自発的な会議や、計画された会議に新しい資料を導入する妨げとなる。このため、アプリケーションが新しいファイルを容易に導入でき、全ての会議出席者が同じファイルを入手できるようなファイル転送機能を MMConf に持たせた。

この基本的なメカニズムによって、1つのサイトのアプリケーションがファイルの転送を要求できるようになる。遠隔サイトは、転送ファイルの名称かエラー表示を含んだメッセージを受取る。

(4) インプリメント

UNIX 上では、MMConf は各アプリケーションにリンクされたユーザ・インタフェース・ツールキットと、ツールキット内でルーチンが通信する会議マネージャ・プロセスからなる。会議マネージャは、会議の出席者を表し、会議出席者間の全ての通信を取り扱って、出席者が会議に参加したり退席したり、新しい会話を開始するためのユーザ・インタフェースを提供する。

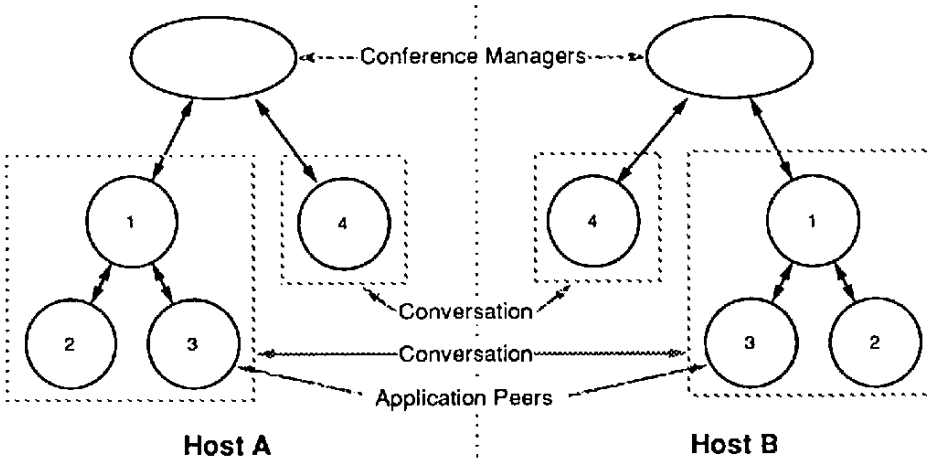


図3.2-11 MMConf のプロセス構造。この図は、会議の基本的なプロセス構造を示す。会議マネージャは、マシン間の通信を扱い、ツリー内の対応するプロセスはアプリケーション・ピアで、対応するツリーは会話である。

現在TCP／IP上にインプリメントされている1つのデータ・ストリームが、ローカルな会議マネージャを遠隔会議マネージャに接続し、もう1つのデータ・ストリームは、ローカルなマネージャをローカルなアプリケーション・ピアに接続する。アプリケーションは、全ての会議トラヒックを会議マネージャに送り、マネージャはそのコピーを進行中の会議ストリームに書込むことでこれをマルチキャストする。

① 会議開始

会議を開始するには、会議発起人である1人のユーザが会議マネージャを呼出し、ユーザと他会議出席者のホスト名と最初の会話のトップレベル・アプリケーション名を指定する。発起人の会議マネージャは、他会議出席者のホストの周知のポートに接続して、ホストが会議マネージャのコピーをスタートさせる。

遠隔会議マネージャは、発起人が誰であるかと、ユーザが会議招待を受けるか拒否するかの応答を要求した対話ボックスを、全出席者のスクリーンに表示する。被招待者全員が応答するか時間切れになると、発起人の会議マネージャが他のマネージャ全員に全会議出席者のリストを送り、最初の会話を開始する。

② アプリケーション開始

会話をスタートさせる会議マネージャは、まず会話に独自の識別子(ID)を割当てる。そしてマネージャはIDとトップレベル・アプリケーション名、コマンドライン引数をマルチキャストする。

すると各マネージャは、アプリケーションのコピーを生成して、IDとマネージャが接続要求待ちしているIPCポート名の入った変数をアプリケーション環境に入れる。アプリケーションが第1回目にツールキット機能を呼出すと、その機能は環境変数を通知して、自動的にマネージャのポートへのIPC接続を開く。

③ 発言権管理

会話には1つの発言権しかなく、これが会議に同時に事象をもたらす出席者を識別する。発言権は、トークンと順序カウントからなり、カウントは、参加者2人以上の会議での発言権譲渡中、事象の順序が誤るのを防ぐために利用する。

MMConfは、発言権管理の概念をメカニズムと方針に分け、メカニズムでは発言権トークンの受渡しと全参加者のための同期事象ストリームの保持という低レベルの活動を扱う。方針では、会議出席者がどのように発言権を要求してこれを認めるかを決定する。会話毎に発言方針を異なるものとすることが可能で、出席者はいつでもこれを変更できる。

④ 入力事象の分配

ツールキット内のコードはアプリケーションの全ての入力事象を受取る。アプリケーションが発言権を持っている場合、ツールキットはまず事象をアプリケーションの待ち行列に入れ、会議マネージ

ャとのインタラクションを省いて、インタラクティブな性能をよくする。次に、ツールキットは事象をウィンドウ・システムに依存しない形式に翻訳して、これを会議マネージャに転送すると、会議マネージャはアプリケーション・ピアIDをスタンプして、他会議出席者にマルチキャストする。

アプリケーションに発言権がないと、事象を受取った時のツールキットの行動は、その時の発言権要求方針で決ることになる。暗示的要求方針では、ツールキットは発言権要求を出して入力待ち行列をブロックし、明示的要求方針では事象を放棄する。

会議マネージャは遠隔の入力事象をバッファし、一連番号で並べる。発言権譲渡中は、受取の順序が狂うことも有りえる。するとマネージャは、IDで順序付けた事象をローカルなアプリケーションにディスパッチする。

マネージャから事象を受取ると、各アプリケーション内のツールキットは事象をウィンドウ・システムに対応する形式に翻訳し直し、ローカルに発生したかのように待ち行列に入れる。しかし、会議認識アプリケーションは、事象を最初に発生した参加者を常に認識することができる。

(5) アプリケーション

MMConfプロジェクトにとって大きな目標は、有益な共有アプリケーションを構築することである。コンピュータ遠隔会議に用いて、遠隔会議アーキテクチャの要件を考察したアプリケーションとしては、次の5つのものが挙げられる。

① BBN／Slate Document Editor

元々はシングル・ユーザ・アプリケーションであるBBN／Slateを遠隔会議に使うことにより、ユーザはテキスト、グラフィック、画像、スプレッドシート、音声の入った文書を見て編集することができる。

② ViewShell

MMConfアーキテクチャの柔軟性を持つ端末エミュレータである。エミュレータ自体は、ユーザ入力の管理に発言権管理と事象分配を使う通常の複製型プログラムだが、コマンド・インタープリタ（シェル）とこれが起動させる全てのプログラムは1つのサイトでしか動作しない。

ViewShellは本来、端末ベース・プログラムのための集中型会議アーキテクチャを形成し、集中型アーキテクチャの利点を全て持っているため、MMConfユーザは、修正を行っていないアプリケーションやハードウェアに依存するアプリケーションにさえアクセスできる。

③ ビデオ・ツール

各種ビデオ・アプリケーションによって、近接したワークステーション間でビデオ資源を共有するための強力な環境が生まれる。ユーザ・インタフェース・ツールキットでのファイル名収集機能同様、ビデオ・ツールキットにおける機能は、ビデオ・アプリケーションに必要な全ての会議認識を持って

いる。

④ Sketch

発言権管理や自動的な事象分配なしに動く単純なグラフィック・エディタで、全会議出席者が同時に図とインタラクトすることができる。Sketch は、アプリケーションに固有のメッセージを使って、各出席者が図に加えた変更を告示する。Sketch にはまた、会議参加者がプライベートなウィンドウから画像やテキストを取込むためのスクリーン・グラブやテキスト・グラブ機能がある。

⑤ Presenter

遠隔会議でのスライド・ショーを実現するものである。スライドにはすべてアプリケーションと関連データ・ファイルの名前がついていて、これを選ぶことでアプリケーションがスタートし、データを読みこんで表示する。

3.2.5 Rapport

原文タイトル：	A Comparison of Application Sharing Mechanisms in Real-Time Desktop Conferencing Systems
翻訳タイトル：	リアルタイム・デスクトップ会議システムにおけるアプリケーション共有メカニズムの比較
著書：	S.R. Ahuja (AT&T Bell Laboratories, Holmdel, NJ, USA) J.R. Ensor (AT&T Bell Laboratories, Holmdel, NJ, USA) S.E. Lucco (AT&T Bell Laboratories, Holmdel, NJ, USA)
出典：	ACM SIGOIS Bulletin, Vol.11, No.2, April 1990, pp.238-248

本論文では、リアルタイムのコンピュータ・ベースの会議においてアプリケーション・プログラムを共有する方法を比較するために、Rapport というマルチメディア会議システムの 3 つのバージョンにおける、異なった手法が比較検討されている。

(1) リアルタイム会議システムの分類

リアルタイムのコンピュータ・ベースの会議システムは、次のように分類できる。

① クローズド・システム

クローズド・システムは、会議参加者のコンピュータに分散される単一の専門化されたプログラム

として開発されたシステムである。この場合、各参加者のコンピュータに存在する分散プログラムは、それが受け取るコマンドを解釈し、そのコンピュータ画面上に表示を生成する。これらの分散プログラムは、すべての参加者にとって状態の整合性をとるために、お互いにメッセージを送り合う。

② オープン・システム

オープン・システムとは、既存の一般に利用されるプログラム（アプリケーション・プログラム）を、修正することなく会議中で実行できるようにしたシステムである。例えば、それぞれの参加者の自分の気に入りのエディタ、VLSI CAD システム、財務計画エキスパート・システム等が、コンピュータ・ベースの会議の最中に利用できるのである。クローズド・システムと同様に、会議参加者のコンピュータの上のオープン・システム・モジュールは、状態の整合性を保持するため、お互いにメッセージを送り合う。また、オープン・システムは、アプリケーション・プログラムの出力表示を作り出すために、入力あるいは出力コマンドを各参加者のコンピュータに送る必要がある。かくして、オープン・システムは、一般的に言って、クローズド・システムよりも多くのメッセージ・トラフィックを作り出す。しかし、オープン・システムは拡張の際クローズド・システムのような必要性がない。新しい機能は、会議の間に、追加のプログラムを実行するだけで得られるのである。

オープン会議システムは、どの会議コンピュータの上でアプリケーション・プログラムを実行すべきであるかという基本的な問題を解決しなければならない。このためのアプローチとして、次の2つがもっとも一般的である。

(a) マルチ・サイト・アプローチ（複製実行方式）

このアプローチでは、会議の中の２つ以上のコンピュータが、同期と入力制御に関する或る種の制約条件のもとで、それぞれのアプリケーション・プログラムのコピーを実行する。プログラムの複数のコピーは、すべて同期をとって開始され、正確に同一の入力を与えられることによって、歩調を合わせる。種々の入力制御プロトコルをインプリメントするために、入力にはフィルタをかけることができる。このようにして、プログラムのコピーは、同一の出力を生成し、それぞれが会議参加者のコンピュータの画面上に表示される。

このアプローチの利点は、会議コンピュータの間で入力コマンドだけが伝達されるのであるから、ネットワークのトラフィックが相対的に少なくて済むという点である。これは、コンピュータが広域ネットワーク（WAN）によって接続されているような会議に対しては、ネットワークへの負荷要求が少ないマルチ・サイト・アプローチが適しているということを示唆するものである。

このアプローチの大きな欠点は、アプリケーション・プログラムの複数のコピーの間で、整合性を保つことの困難さである。このような同期性の喪失を検知し、会議参加者の間の整合性を回復する技術が必要であるが、殆どのアプリケーション・プログラムは、以前の整合性のある実行状態を回復することができない。さらに、この技術は、会議参加者のそれぞれが、それぞれのアプリケーション・

プログラムを同一のハードウェアおよびソフトウェア環境で実行することを必要とする。したがって、このアプローチは、共有アプリケーションが状態保持を殆ど必要とせず、再同期を取ることが容易である場合、つまり「スライドショー」アプリケーションである場合にのみ適しているように思われる。

(b) シングル・サイト・アプローチ（中央集中実行方式）

このアプローチでは、単一のサイトのみがそれぞれのアプリケーション・プログラムを実行する。会議の参加者からアプリケーションへの入力は、実行サイトへ送信される。アプリケーションからの出力コマンドは会議参加者全員のコンピュータへブロードキャストされる。このアプローチの主な利点は、会議中に同期と整合性を保持する重荷の大部分が、アプリケーション・プログラムから会議システムへ移されることである。それぞれのアプリケーション・プログラムについて、ただ1つのコピーしか実行されないので、整合性を取る問題は避けられる。さらに、このアプローチでは、会議参加者は自分ではプログラムを実行しなくても実行結果を見ることが出来る。したがって、自分の環境におけるソフトウェアとハードウェアに、他者とは幾分かの相違があっても、大勢の人々が会議に参加することができる。これによって、プログラム・コードを共有することなく、プログラムを共有することができ、プログラムのセキュリティおよび所有権が保持される。このアプローチの不利な点は、入力メッセージと出力メッセージの両方が通信されるため、マルチ・サイト・アプローチよりも、ネットワークのトラフィックが大きいことである。

(2) Rapportのインプリメントの概要

Rapport マルチメディア会議システムは、一群の人々がオフィスにあるコンピュータと電話を使用して、音声、データ、イメージを共有しながら、リアルタイムの議論ができるようにしたものである。比較のために3つのバージョンが開発され、会議においてアプリケーションの共有をサポートするためのもっとも一般的な方法がインプリメントされた。Rapport の異なったインプリメントについての測定結果は、上記のアプローチについての、定量的な比較を提供するものである。

Rapport システムには2つの基本機能がある。まず、呼び制御機能が提供されている。これは人々が会議を開始し、実行し、終了するためのメカニズムである。第2に、Rapport は、会議参加者の間で表示を共有するための機能、アプリケーション・プログラムの実行を制御する機能、それらの間の整合性を保持する機能、およびそれらに関連したウィンドウを管理する機能を持っている。図3.2-12 は、Rapport による会議のコンピュータ画面のイメージである。Rapport の呼び制御機能は、通常の電話の呼び制御機能に基づくものであるが、画面ベースの電話からアクセスされる。アプリケーション・プログラムの管理とそれらの表示は、仮想会議室のメタファに基づいている。Rapport は電子会議室を表示し、参加者は自分のアプリケーション・プログラムを携えてその仮想的会議室に入ることができる。図では、「ホワイトボード」グラフィック作成プログラム、ファクシミリへのメモ書きプログラム、

および端末エミュレーション・プログラムの3つのアプリケーション・プログラムが、会議 sid.0 に対応した仮想的会議室で共有されている。8人の参加者のうち3人がテレポインタを利用して、ホワイトボード上の項目を指し示している。

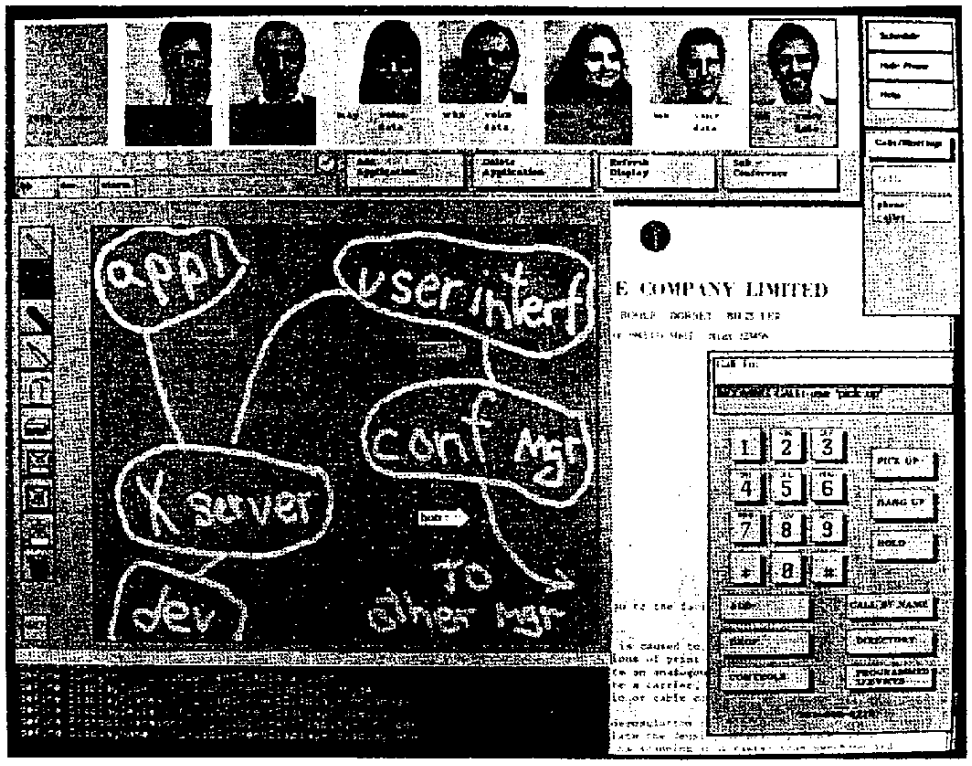


図3.2-12　Rapportのスクリーン・イメージ

Rapport の3つのバージョンにおいては、それぞれ異なったアプリケーション・プログラム実行環境が提供されている。1つのバージョンにおいては、アプリケーション・プログラムに対するマルチ・サイト実行環境がサポートされている。そこでは、それぞれの共有アプリケーションの入力コマンドが会議参加者へマルチキャストされる。他の2つのバージョンでは、シングル・サイト実行アプローチが取られており、会議参加者にはそれぞれの共有アプリケーション・プログラムの出力コマンドがマルチキャストされる。Rapport の3種類のインプリメントのすべては、特別に設計されたボードによって Sun ワークステーションと電話を結合したノードに基づいている。ワークステーション間のデータ通信には、10M ビットの Ethernet が利用される。音声と制御信号は別のネットワークで通信される。このネットワークの集合体は、ISDN の通信チャネルを真似たものであり、本開発においてはISDN への移行を容易にするために活用された。本論文執筆時点では、データおよび音声ネットワークによって結合されたこれらのワークステーション上で、Rapport 機能が完全にサポートされている。Rapport は、UNIX および X Window システムによってサポートされたソフトウェア環境で実行される。X Window サーバとそれぞれのアプリケーション・プログラムとの間の通信は、Rapport の会議マネージ

ヤによってフィルタされる。アプリケーション・プログラム、X Window サーバ、および会議マネージャ間の関係は、3 つのバージョンの間で異なっている。

① マルチ・サイト・アプリケーション実行環境（MULTI）

Rapport の最初のバージョンは、アプリケーション・プログラムのマルチ・サイト実行をサポートしており、MULTI と呼ばれる。このシステムの主要な構成要素およびそれらの間の通信経路を図3.2-13に示す。それぞれのコンピュータは、X Window サーバ (X-Server)、会議マネージャ (C-Mgr)、およびアプリケーション・プログラム、P1 と P2 を実行する。それぞれのアプリケーション・プログラムは、X Window サーバに接続されている。X Window サーバは、会議マネージャに接続され、会議マネージャはお互いに接続されている。すべての接続はソケットを介している。このバージョンにおいては、X Window サーバは、会議マネージャとの間でメッセージをやり取りできるよう改造されており、これによって Rapport は、アプリケーション・プログラムへの入力をフィルタし、マルチキャストすることができる。図で説明されている状況においては、アプリケーション P1 への入力がコンピュータ B にいる参加者によってなされている。B にいる会議マネージャは、この参加者がこのアプリケーションへの入力制御を持っていることを認識し、B の上の X Window サーバに入力ストリームを与える。B 上の X Window サーバは、その入力ストリームをそこにおけるアプリケーションに渡す。また、会議マネージャは、その入力イベントをコンピュータ A および C の上で実行されているマネージャに送る。これらのマネージャは、入力をそれぞれのローカルな X Window サーバに与え、そこで実行されているアプリケーションにわたるようにする。コンピュータ画面に表示をおこなうための X Window システムの出力コマンドは、アプリケーション・プログラムのそれぞれのコピーによってそれぞれのローカルな X Window サーバに送られる。

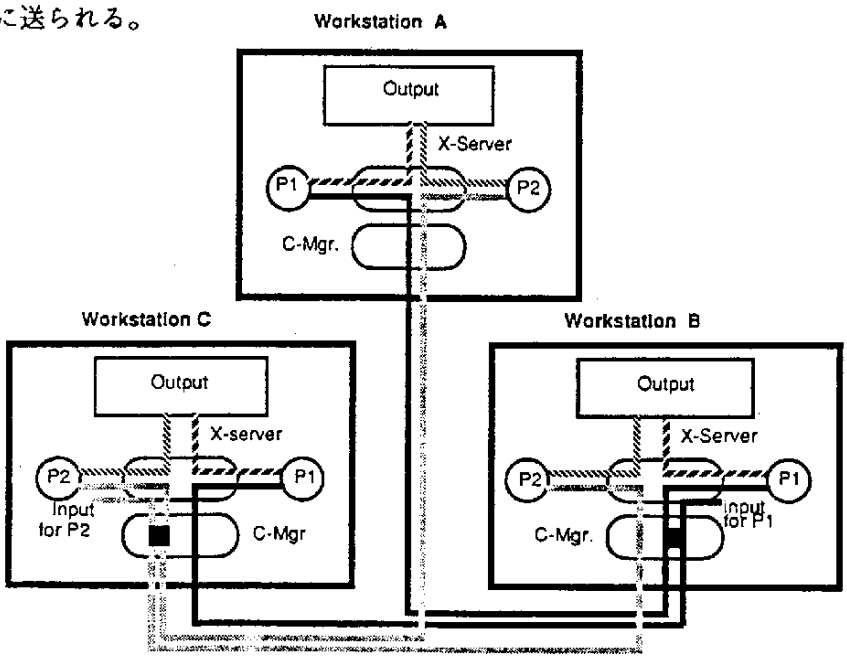


図3.2-13　Rapport におけるアプリケーションのマルチ・サイトによるインプリメント

② シングル・サイト・アプリケーション実行環境（SINGLE）

Rapport の第 2 と第 3 のバージョンは、アプリケーション・プログラムのシングル・サイトでの実行をサポートしている。ここで説明するインプリメントでは、LAN にも WAN にも適しており、SINGLE と呼ばれる。このシステムの主要な構成要素およびそれらの間の通信経路を図3.2-14 に示す。それぞれのコンピュータは、X Window サーバ（X-Server）および会議マネージャ（C-Mgr）を実行する。しかし、それぞれのアプリケーション・プログラムは、ただひとつのコンピュータの上でのみ実行される。このインプリメントにおいては、アプリケーション・プログラムは X Window サーバには接続されておらず、会議マネージャに接続されている。 会議マネージャは、擬似的な X Window サーバとしての役割を果たしている。X Window サーバは改造されていない。それぞれの会議マネージャは、自分自身のコンピュータの上の X Window サーバに接続されているが、リモートな X Window サーバにはまったく接続されていない。会議マネージャは互いに接続されている。ここでもすべての接続はソケットを介している。コンピュータ A は、アプリケーション P1 の実行サイトであり、コンピュータ C は、アプリケーション P2 の実行サイトである。アプリケーション P1 への入力はリモートな会議ノードであるところのコンピュータ B において生成され、アプリケーション P2 への入力はローカルに生成されている。B における会議マネージャは、この参加者がこのアプリケーションへの入力制御を持っていることを認識し、入力ストリームをアプリケーションへ送る。アプリケーションが出力を生成したとき、それはローカルな会議マネージャに直接与えられる。会議マネージャは、それをローカルな X サーバに与える。また、会議マネージャは、この X の出力要求をコンピュータ B および C の上で実行されている会議マネージャに与える。X のリソース識別子の変更が必要であるが、これは、B および C の会議マネージャによっておこなわれ、これらの会議マネージャは、それぞれのローカルな X

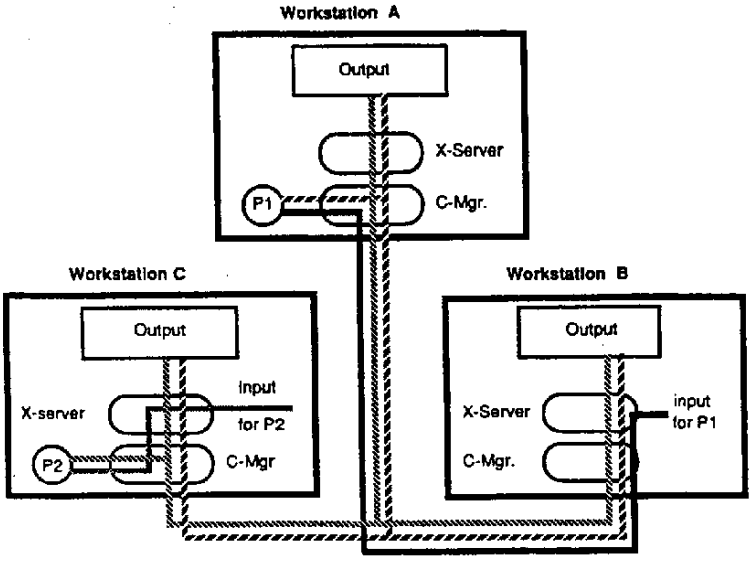


図3.2-14　Rapport におけるアプリケーションのシングル・サイトによるインプリメント

Window サーバに適切な X の出力要求を与える。それぞれの会議マネージャには同一のメッセージが送られるので、この通信は真にマルチキャストである。このマルチキャストの特性によって、このアプローチは WAN に適したものとなっている。WAN においては、メッセージの解釈は期待出来ないからである。さらに、このアプローチでは、1 台のコンピュータ当たり、ただ 1 つのコネクションしか必要ではない。必要なのは、会議マネージャと何らかのネットワーク・スイッチの間の接続だけであり、これはWAN にとって単純な構成である。

③ LAN のためのシングル・サイト・アプリケーション実行環境（SINGLE／LAN）

ここでは、②に対する改造を説明する。これは、ある種の X Window システムのプロトコル・メッセージに対する経路の長さを短縮するものである。このバージョンでは、X Window サーバの遠隔通信機能が利用されており、現在のところでは、LAN 環境でのみサポートされている。このため、このバージョンは SINGLE／LAN バージョンと呼ばれている。このシステムの主要な構成要素およびそれらの間の通信経路を図3.2-15 に示す。それぞれのコンピュータは、X Window サーバ (X-Server) および会議マネージャ (C-Mgr) を実行する。しかし、それぞれのアプリケーション・プログラムは、ただひとつのコンピュータの上でのみ実行される。ここでも、会議マネージャは、擬似的な X Window サーバとしての役割を果たしている。X Window サーバは改造されていない。X Window サーバのリモート・ホスティング機能を利用して、アプリケーション・プログラムに対応した会議マネージャは、会議コンピュータのそれぞれの X Window サーバに直接接続される。他の図と同様に、現在、アプリケーション P1 への入力はコンピュータ B にいる参加者によって生成されている。B における会議マネージャは、この参加者がこのアプリケーションへの入力制御を持っていることを認識し、入力ストリームをコンピュータ A の上のアプリケーションへ送る。アプリケーションが出力を生成したとき、それはロ

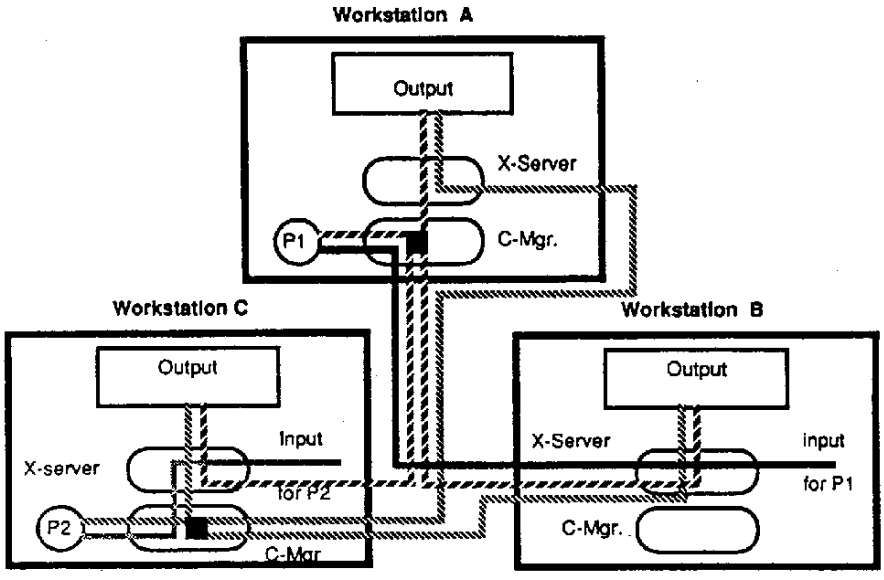


図3.2-15　Rapport におけるアプリケーションのシングル・サイト/LAN によるインプリメント

ーカルな会議マネージャに与えられる。会議マネージャは、それを、コンピュータA、B、Cの上のX Windowサーバに渡す。マネージャは、必要なリソース識別子の変更を行い、その後、コマンドをX Windowサーバに送る。会議マネージャがこれらの変更を行うことによって、Xサーバは会議を意識しなくて済む。しかし、マネージャは、すべてのリモートなX Windowサーバへの直接の接続を必要とする。図には示されていないが、会議マネージャは、相互に接続され、会議の状態情報を通信しあっている。ここでもすべての接続はソケットを介している。

(3) 性 能

Rapportの3つのバージョンである、MULTI、SINGLE、SINGLE／LANの性能を比較する。比較は、会議コンピュータの間のメッセージの数に焦点をあてる。これは通常の会議活動の間にカウントされたものである。

① 比較の環境

3種類のインプリメントの比較の要素となる、会議マネージャとアプリケーションとの間のメッセージは、3つのカテゴリに分けられる。管理メッセージ、入力メッセージ、出力メッセージである。管理メッセージとは、会議を設定し、参加者の追加削除を行い、会議のインタフェースのプロセスと通信するものである。入力メッセージとは、X入力イベントをアプリケーションへ送るもので、会議マネージャのプロセスの間に送られる中間的なメッセージもこれに含まれる。また、このカテゴリのメッセージには、情報問い合わせへの応答も含まれる。出力メッセージとは、アプリケーションの出力要求をX Windowサーバへ送るものである（中間的メッセージも含む）。

また、3種類のインプリメントの各実験環境において、アプリケーション、およびアプリケーションの実行サイトとその入力元の関係を合わせるために、会議「スクリプト」を作成することで対応した。すなわち、測定においては下記の三者会議のシナリオを使用した。

AがBとCを呼び出す。BとCは受話器を取り上げる。Aは端末エミュレータ・アプリケーションを開始し、テキスト・エディタviを起動し、事前に決められた編集操作を実行する。Aは、端末エミュレータの制御をBに渡す。Bは別の一連のコマンドを実行する。Cは対話的図面作成プログラムを開始し、事前に決められた図を作成する。Cは制御をAにわたし、Aは図を修正する。

② 比較結果（メッセージの数と応答時間）

ここでは、SINGLEとMULTIの比較に焦点を当てる事にする。SINGLE／LANの測定結果については簡単に触れるにとどめる。

(a) バッファリングのない場合のメッセージ数

3つのバージョンすべてが同一の管理メッセージを会議マネージャに送信した。会議の間に、98個の管理メッセージが発生した。その大部分は会議インタフェース・プロセスからの状態問い合わせに

よるものであった。すべての場合において、これらのメッセージは、メッセージ・トラフィック全体の15%以下であった。メッセージ・トラフィックの大部分はディスプレイからの入力と出力メッセージであった。MULTIは、全体で636個のメッセージを送り、そのうち538個はXの入力イベントを運んでいた。SINGLEは、全体で3875個のメッセージを送り、そのうち87%はXの出力イベント、11%はXの入力イベントであった。SINGLEが送るメッセージに入力イベントを運ぶものが少なかったのは、SINGLEの場合、入力はポイント・トゥ・ポイントで送るだけで良いからである。LANの送ったメッセージはSINGLEのそれのおよそ半分であった。これは会議マネージャとX Windowサーバが直接接続されることが多いからである。

会議シナリオにおいては、アプリケーションから出される出力要求の数は、生成されるX入力イベントの数のおよそ5倍であった。全体的に見れば、バッファされない場合においては、シングル・サイト・システムは、マルチ・サイト・システムに比較して、およそ4倍のXイベント・メッセージを生成している。

出力メッセージの平均パケット・サイズは、25バイトであった。管理メッセージは98バイト、入力メッセージは24バイトであった。入力メッセージは殆ど常に単一で来る（50ミリ秒以上の間隔）。入力メッセージに関して観察された最大の突発は、10ミリ秒以内に生成された6個のメッセージであった。出力メッセージは、時間的に良く分離されている場合が多い（50ミリ秒以上の間隔）。しかし、出力メッセージは、入力メッセージよりも、束になって発生する傾向がある。出力メッセージの最大の突発は、10ミリ秒以内に生成された89個のメッセージであった。これらの突発は、出力メッセージの数を入力メッセージの数よりずっと多くする原因となる。

(b) バッファリングを行った場合のメッセージ数

Rapportの3つのインプリメントのすべては、ペンディングとなっているすべての到着メッセージを読んで処理し、生成された出力メッセージをバッファの中に集めるメッセージ・バッファリング・パッケージを利用するよう構成することができる。このパッケージは、その後、バッファを単位としてそれぞれの目的地へ送る。UNIXのソケットは、部分的メッセージも送ることができるから、受取側では、バッファリングなしの場合と同じ振舞で良い。バッファリング・パッケージは、無視できるほどのソフトウェア・オーバヘッドしか持たないが、どのインプリメントにおいて会議スクリプトを実行する場合にも、必要なメッセージの数を劇的に減少させる。出力メッセージは、入力メッセージよりも、束になって発生する傾向が強いから、バッファリング・パケットを利用することの効果は、SINGLEの方がMULTIよりも大きい。SINGLEのインプリメントにおいて、バッファリングが使用された場合、出力メッセージの入力メッセージに対する比率は、3.4対1である。さらに、会議スクリプトを実効するために、バッファリングを使用するとき、SINGLEは、MULTIに比較して、3.6倍のノード間メッセージしか使わない。バッファリングをしたときのSINGLEの平均パケットサイズは、出力

メッセージが58バイト、入力メッセージが26バイトであった。これらのパケットサイズは、ナローバンドISDNおよびブロードバンドISDNの予想されるハードウェア容量の範囲内に十分おさまっている。したがって、SINGLEの場合のソフトウェア／ネットワーク・パケット通信速度のピーク要求は、MULTIの場合のそれのおよそ3.5倍であると結論する。

(c) 応答特性

端末エミュレータ(xterm)を実行するすべてのバージョンについて、このアプリケーションをローカルで操作しようと、リモートで操作しようとにはかかわりなく、本質的には即時の応答時間を観測した。すべての表示は、1秒以下の時間内で同期が取られた。

黒板アプリケーションで、細い線をリモートで描く場合、すべてのバージョンで即時の応答時間が得られた。太い線をリモートで描く場合、SINGLEのユーザは、若干の遅延時間を観察した。これは、太い線を描くために必要な余分の出力メッセージが原因で発生したものである。バッファリングを行った場合には、すべてのバージョンにおいて、最大応答時間は1秒以下であった。すべての平均応答時間は0.1秒以下であった。X Windowシステムに基づく典型的なアプリケーションによる測定は、SINGLEとSINGLE／LANは、MULTIの場合よりも、6倍のバンド幅のネットワークを必要とすることを示している。しかしながら、静的な分析および、動的性能計測の両者によって、3種類のインプリメントはインタラクティブな応答性能においては、同程度であると結論する。

3.2.6 Commune

原文タイトル： Commune: A Shared Drawing Surface

翻訳タイトル： Commune：共有ドローイング・サーフェス

著書： Sara A. Bly
(Xerox Palo Alto Research Center, Palo Alto, California, USA)
Scott L. Minneman
(Center for Design Research, Mechanical Engineering Department,
Stanford University, Stanford, California, USA)

出典： ACM SIGOIS Bulletin, Vol.11, No.2, April 1990, pp.184-192

本論文では、遠隔作業を行う設計者同士がドローイング・サーフェスを共有し、フェース・トゥ・フェースで作業する場合と同じインタラクションの多くを共有することができるプロトタイプのツール、Communeについて紹介されている。

(1) Commune の初期の研究

Commune は、共有ドローイング・サーフェスの設計と利用に関して行われている研究の一部である。本研究のテーマの1つは、ビデオテープで記録した作業状況と、その作業を支援する新技術の設計開発との関係を理解することである。本研究は、共有ドローイングに関して、設計中にドローイングスペースで起こる活動の特性を明らかにして、技術で支援できるドローイング活動分野を特定しようとするものである。このような活動には、話し合いやドローイング・サーフェスの利用、参加者のジェスチャなどが含まれる。Commune では、ドローイング・サーフェスそのものに関連する非言語的な活動に焦点を合わせた。

本研究の初期の段階で、Bly は2人の人間による設計セッションを、①フェース・トゥ・フェース、②オーディオ／ビデオでリンクされているが地理的には分離、③電話でのみ接続という3つの異なる状況で実験を行った。この研究から、設計者は共同設計プロセスの一部として広範囲にドローイング・サーフェスを利用し、ドローイングされたものだけでなく、ドローイング活動そのものが重要であることが結論づけられた。このような結論はデータの観察に基づくもので、どのセッションでも、設計者たちは数秒間の内にドローイング、ライティング、ジェスチャをしばしばすべて実行しながら迅速に行っており、状況が許せば規則的にドローイング・サーフェスを参照していた。これはフェース・トゥ・フェースのケースでは特に顕著で、当然のことながら電話の場合は不可能だった。どのセッションも、強調のためのドローイング・サーフェスの頻繁な利用が特徴となっており、設計者たちがお互いのドローイング・サーフェスやジェスチャを見ることができない電話セッションの場合でさえ、話し合っていることについて図を書いたりジェスチャを行って話し合いの内容を強調し続けていた。設計者たちが1つのドローイング・サーフェス上で指示したり書き入れたりすることのできるフェース・トゥ・フェース・セッションでは、全体の時間の約50%で同じドローイングに関してやり取りを行っていた。

(2) Commune

Commune は、これまで研究してきた同時に2人から3人が作業するグループによる設計活動の支援を目的としたものだが、特に参加者が異なる場所に存在する場合に焦点を当てている。初期の研究から、次の設計目標を満たすことが、共有ドローイング・システムの効果的な利用に役立つことがわかった。

- 各設計者の書いたものやジェスチャを、他設計者が同時に見ることができること。
- 設計者は、ドローイング、ライティング、ジェスチャの迅速な切換えができること。
- 全設計者が、共有スペースで同時に書いたりジェスチャを行えること。
- ツールはできるだけ「ナチュラル」なものであること（よく使われているライティング／ドロー

イングツール、入力した位置にその入力した図や文字が現れることなど)

このような目標を満たすCommuneのモデルとして、次のような形態を考えた。すなわち、各ユーザは、指示やドローイング、ライティング用のペンを持っており、そのペンで書かれたものは各々他と識別可能とする。システムとしては、ユーザが同一の面に書くと、その面に書いたものが現れる。面は水平に設置され、ライティング・ツールはペンのようなものとした。

① インプリメント

初期プロトタイプは、MS-DOSの'386 PCのプラットフォーム上に構築し、スタイラス付き透明デジタイジング・タブレットをライティング面とする。各デジタイザは、スクリーンが水平面を向くようにしたフラット・スクリーン・ディスプレイ・モニタ上に直接搭載する。Communeの各ワークステーションは、図3.2-16のように配置したモニタとデジタイザからなり、モニタは、同じ画像が全ユーザに同時に現れるようデイジーチェーン式になっている。ここで述べる設計用Communeの構成図を図3.2-17に示す。

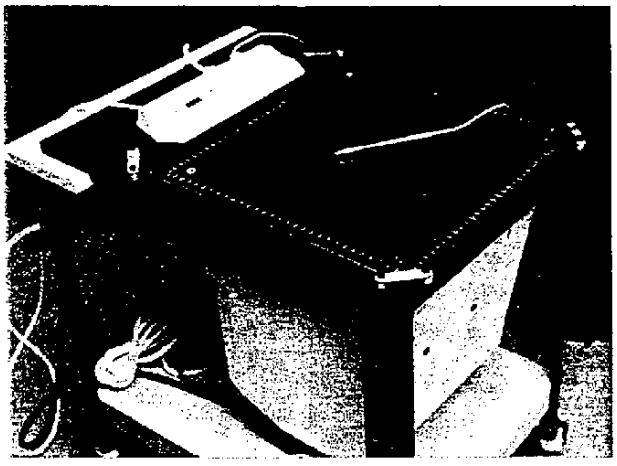


図3.2-16 デジタイザ付き水平モニタからなるCommuneのワークステーション

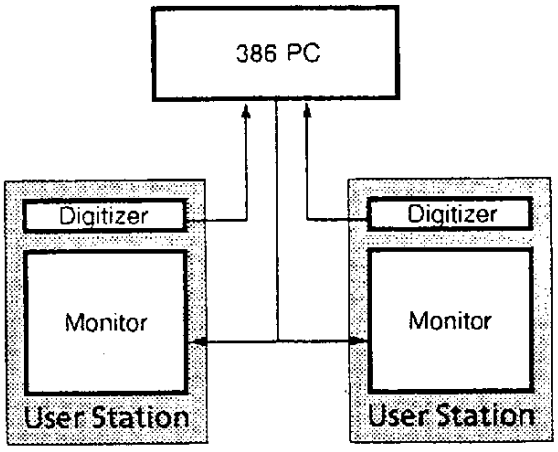


図3.2-17 PCで接続された2つのステーションを示す研究システムの構成図

各デジタイザ・タブレットは、それぞれのスタイラスの位置を連続してプロセッサに知らせ、Communeは各ユーザからのスタイラスの現在位置にカーソルを表示する。スタイラスで押し下げると、カーソルの跡に線が引かれる。このように、設計者はスタイラスを動かすだけで指示したり、スタイラスを押し下げて動かすことでドローイングやライティングを行うことができる。全設計者は、このような動きを、知覚できる遅れなしに同時に見ることができる。

各ユーザのスタイラスは、スクリーンの上ではそれぞれ異なった色の鉛筆形のカーソルとして表れ

る。書かれたものはピクセル（これも色で識別される）で、直線や曲線、テキストといった区別はない。最初のプロトタイプでは、Commune は3つのメニュー・オプションしか持っていない。ドローイング・サーフェス上に書かれたものを全てクリアする「ERASE」と、現在書かれたものをセーブしながら1ページ先に進む「→」、以前にセーブしたページに戻る「←」である。

② Commune の利用結果

Commune を使って、2人による3つの設計セッションに関する初期の研究と同様の研究を行った。遠隔協調作業をシミュレーションするため、2つのオフィスにそれぞれ2台の Commune ドローイング・ステーションを設置し、各オフィスのモニタとカメラが、もう一方のドローイング・ステーションにいる人間の顔を映す。音声のやり取りには電話を使用した。被験者である本論文著者の1人とその同僚の製品設計担当者の2人の設計者は、初期の研究にも参加している。製品設計者が現在従事している問題について、この2人が Commune を使って2時間の設計セッションを行った。

この設計活動全体を完全に記録するため、数種のビデオが集められた。各ワークステーションを肩越しに撮ったビデオは、各設計者がいつどのようにシステムとやり取りをするかをはっきりと記録していた。

ビデオテープ上の行動は、「アクション」、「用途」、「インタラクション」、「リアクション」という4種類のカテゴリに分けて記録した。「アクション」については、ユーザが面に英数字を書いたり（ライティング）、図面を描いたり（ドローイング）、書かれたものにカーソルでジェスチャを行うごとに事象を記録した。1つの事象は、ユーザがアクションを変更したり現在の書かれているものからカーソルを移動させたりした時に終了したものとみなされた。各事象は、図解や強調、おぼえ書き、参照、その他といった、「用途」によっても分類された。

観察結果は、上記の4種類のカテゴリのいずれの面でも、遠隔セッションよりもフェース・トゥ・フェースの設計セッションに近い傾向を見せた。図3.2-18 は、初期の研究でとりあげられた3つの状況とCommune を利用した状況におけるドローイング、ライティング、ジェスチャというアクションと用途を相対的に示したものである。

(a) アクション

Commune では、ユーザは他の遠隔状況よりジェスチャを多く行った。Commune は、設計上ドローイング・サーフェスに関連するジェスチャがカーソルを使った2次元の動きに限定されるため、ジェスチャの利用は阻害されると思われたが、設計者はスクリーン上に書かれたものを参照したり（「この点」、「ここ」等）、対応する話し合いを強調するため、カーソルによるジェスチャを頻繁に利用した。参照や強調という予想されたジェスチャの利用に加えて、設計者はカーソルによるジェスチャを、実際に面上に書かずにポイントを示すために使うことがあった。このような一時的な図解の実行と機能については、さらに研究が必要である。

	Draw	Write	Gesture
Face-to-face	29%	11%	61%
Media link	31%	38%	31%
Telephone	43%	46%	11%
Commune	39%	15%	46%

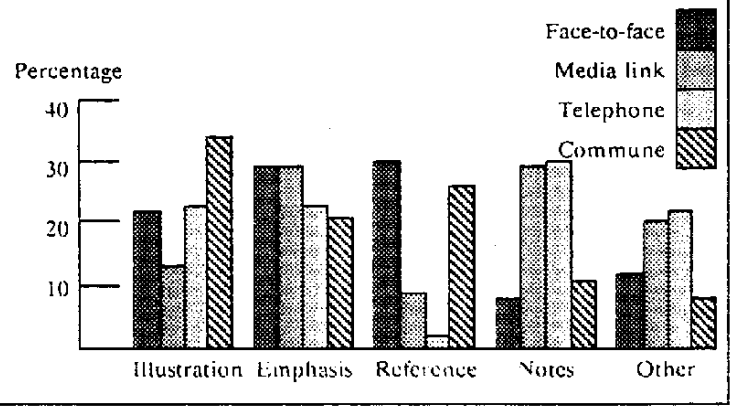


図3.2-18　4つの設計セッションのデータが、ドローイング・サーフェスの利用方法を示す

(b)　用途

ジェスチャ、ドローイングとも、ドローイング・スペースが初期の遠隔セッションより広く参照された。設計者は、初期のオーディオ／ビデオ・リンク・セッションでも互いのドローイング・サーフェスを見ることができたが、書かれたものやコンセプトへの参照はこれほど行われなかったようだ。Communeのセッションでは、フェース・トゥ・フェース・セッションのように、互いのドローイングに何度も参照が行われた。Communeでは、おぼえ書きや、いたずら書きその他無関係な書き込みも大幅に減少した。

(c)　インタラクション

設計者は、同じ場所で互いに書いたものを使ってやりとりすることができ、アクションの75%近くが、参加者双方が使う書かれたものに関連したものだった。セッション中、設計者たちは何度か同時に同じ場所に書いたり差し示したりした。どのセッションでもそうだが、設計者たちは、ライティング、ドローイング、指示を頻繁かつ迅速に混合していた。

(d)　リアクション

概して、設計者たちはシステムがうまくいったと述べている。

この他、新しい能力や問題も見つかった。Communeには、初期の3つの状況のどれにも不可能な2つの特長がある。まず1つは、設計者が常にドローイング・サーフェスに対して同じ向きを向いていられることである。そのため、書かれたものはどちらの設計者にとっても常に上向きとなる。これはオーディオ／ビデオ・リンク・セッションにも当てはまるが、相手の図面はビデオモニタに現れるのであって、ドローイング・サーフェスではない。2つ目は、設計者が文字通り同じ場所に同時に書き込みや指示ができることである。フェース・トゥ・フェースでも理論的には可能だが、実際には物理

的に手が邪魔になる。

Commune の利用においては、3 つの問題が特に目についた。1 つは、特殊なスタイラスが、短い線からなるスケッチの書き写しや作成には利用しにくいことである。2 つ目は、ドローイング・スペースが小さいため、ユーザが頻繁にページを変えなければならないことである。本論文の研究では、設計者は、当初のフェース・トゥ・フェース・セッションでは大きな紙 3 枚（ビデオ・リンク・セッションで 5 枚、電話のみのセッションで 3 枚）に対して12ページを使用した。3 つ目は、Commune ではジェスチャが指示アクションに限られることである。フェース・トゥ・フェース・セッションで見られる表情豊かな手や指の動きはできない。討議に関連したカーソルの動きと、無作為な手の動きも常に区別できるとは限らない。設計者がスタイラスを持たずに手のジェスチャを行う場合もあった（この場合、当然、Commune の相手側には見えない）。

3.2.7 CAVECAT

原文タイトル： Experiences in the Use of a Media Space

翻訳タイトル： Media Space の利用経験

著書： Marilyn M. Mantei
(Department of Computer Science, University of Toronto, Ontario, Canada)
Ronald M. Baecker
(Department of Computer Science, University of Toronto, Ontario, Canada)
Abigail J. Sellen
(Department of Computer Science, University of Toronto, Ontario, Canada)
William A.S. Buxton
(Department of Computer Science, University of Toronto, Ontario, Canada)
Thomas Milligan
(Department of Computer Science, University of Toronto, Ontario, Canada)
Barry Wellman
(Department of Sociology, University of Toronto, Ontario, Canada)

出典： CHI'91 Proceedings, April 1991, pp.203-208

Media Space とは、ビデオ、オーディオ、コンピュータを統合利用し、空間的および時間的に分散した複数の個人あるいはグループが協力して仕事することを可能にするものである。本論文では、このような Media Space の 1 つである CAVECAT (Computer Audio Video Enhanced Collaboration And Telepresence) について、その初期の利用経験を要約するとともに、未解決の技術的問題点、およびこの技術の心理学的、社会的影響について考察している。

(1) CAVECAT システム

CAVECAT システムは、デジタル＋オーディオ＋ビデオのネットワークによって接続された多くの高機能ワークステーションから構成される。それぞれのワークステーションは、パソコン、TV モニタ、TV カメラ、一対のスピーカ、マイクロフォンから構成されている。1 台につき 4 枚のビデオ・ボードがあり、最大 4 ヶ所までの映像を表示する（図3.2-19 参照）。場所によっては、ビデオ・ボードは、ワークステーションの画面上に低解像度のビデオイメージを直接表示することができるので、独立したモニタは不必要である。

このシステムの心臓部は、Rank Xerox EuroPARC で開発されたIIIF サーバを参考に開発されたスィッチング・ネットワークである。オーディオおよびビデオ通信はアナログであるが、ワークステーション上に存在する IIIF サーバのソフトウェアによってデジタル的にスィッチされる。それぞれのオフィスにあるパーソナル・ワークステーションは Ethernet を介して、接続要求メッセージを IIIF サーバへ送る。また、IIIF サーバは、それぞれのオフィスのプライバシ設定を調べ、他のオフィスからのアクセス要求を許可できるかどうかを決定する。

それぞれのパーソナル・ワークステーションには、サーバ・エージェントが常駐している。それぞれのオフィスの人々は、このエージェントへのユーザ・インタフェースを介して多くのコミュニケーション・メタファを選択することができる。タスク指向のメタファ（例えば、会合の呼び掛け）、空間指向のメタファ（例えば、誰か他の人のオフィスへ入る）、オブジェクト指向のメタファ（例えば、自分のオフィスのマイクロフォンを切る）などである。

図3.2-19　CAVECAT による会議のビデオ・イメージ

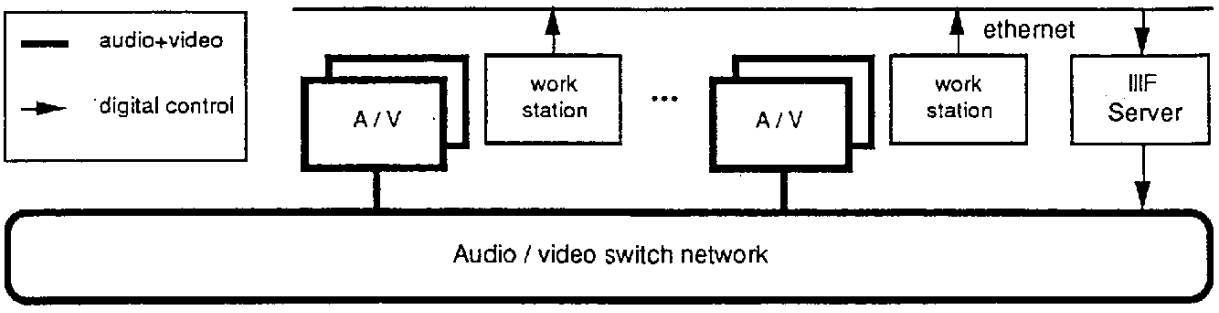


図3.2-20　CAVECAT ネットワークのレイアウト概念図

(2)　CAVECAT の予期しなかった利用方法

Media Space のユーザに対するインパクトを理解するために、2人の教官のオフィス、システム・プログラマのオフィス、および大学院生の作業場所に CAVECAT のノードを設置し、自分自身で利用することとした。コミュニケーションのインタフェースとしては、関係するオフィスのレイアウトからなる空間的メタファを使用した。CAVECAT ユーザ各人のビデオ・イメージをデジタル化し、それらのミニチュア化されたイメージを画面上でそれぞれユーザが所属するオフィスの内部に置いた。これらのユーザ・イメージをある仮想オフィスから別の仮想オフィスに移動させることによって、選択したオフィスとのビジュアルなリンク、および音声リンクが確立される。

これらの利用実験によって、次のような予期せぬ利用方法が発見された。

①　グループによる会議

当初の意図は、このシステムがまず第一に、それぞれのオフィスにいるひとりひとりの人間によるコミュニケーション装置として利用されることであったが、実際には、ビデオ会議の自然な需要が存在しており、このシステムはビデオ会議装置としても利用された。グループのメンバを訪問したお客があったとき、物理的に歩きまわってそのお客を他人のオフィスに連れて行って紹介するのではなく、CAVECAT を利用して紹介したのである。

②　鏡としての機能

このシステムは会議の他のメンバを表示するよう設計されたのであるが、自分自身を表示することには予期しない利点が発見された。利用者は、この「鏡」機能を利用して、自分がカメラに対して正しくフレームされているかどうかを確認したのである。鏡機能は4地点会話用の分割スクリーンに自動的に組み込まれていた。

③　モニタリング機能

もう一つの予期せぬ利用法は、自分自身のオフィスに仮想的に存在させることである。自分自身を仮想的に他人のオフィスに置くためにこのメディア・リンクを利用するのではなく、自分が自分のオ

フィスにいないときに、自分のオフィスを見るためのウィンドウとして、このリンクを利用したのである。利用者は、だれが自分を探していたか、あるいはいつ電話が鳴ったかをモニタすることができた。また、このシステムをセキュリティ目的に利用することができた。

(3) 技術的問題点

本実験において、以下のような技術的な障害が問題となった。

① システムの応答時間の遅れ

仮想オフィス間のコネクションを実現するために、最初に開発したプロトタイプのソフトウェア・パッケージでは、接続の確立、進行中の会議への参加、あるいは会議から抜けるための時間はおよそ2秒であった。2秒の待ち時間とは非常な遠距離通信におけるネットワーク切り替え時間および衛星通信による遅れに相当し、利用者にとってはもどかしいものであった。

② 音声のレベルと雑音

話者のオフィスにおける周辺雑音は音質にとって大きな問題となった。家具や設備の置き方の違い、オフィスにいる人数、ドアが開いているかどうか、オフィスの所有者の座っている場所とスピーカやマイクロフォンとの相対的位置関係、これらすべての要因が音質をさらに劣化させる潜在的要因となった。この結果として、音のレベルは常に調整されなければならなかった。現在、各参加者に自分自身のオーディオを制御する能力を与えることによって、システムを修正しつつある。しかし、そのような調整を簡単にする方法を考えなければならないし、個人による調整がフィードバックによって全体の音質を劣化させないことを保証する方法を考える必要がある。

③ 音声の局所化

CAVECATの参加者は、共有コミュニケーションの音声が、話し手の方向からではなく、空中から聞こえるような感じがするとコメントした。複数の参加者がコミュニケートする場合、参加者が音声を局所化できないとき、だれが話しているのかわからなくなることがある。また、他者のオフィスでの電話のベルがネットワークを介して伝達されてしまうため、それとの混同によって、自分の電話が鳴っていることに気がつくのが困難になることもしばしばであった。

④ ライティングとカメラのアングル

カメラの設置場所、ライティング、部屋の背景の色、カメラと人間との距離などすべての要素が他のオフィスへ送られるイメージの大きさと品質に影響した。カメラ・アングルが悪い場合には、話者の印象が傷つけられる恐れがある。自動焦点機能のあるカメラは、オフィスの中を歩きまわる人間に対して常にズーム・インしたりズーム・アウトしたりする。これによって他のオフィスにいる参加者は、多少乗りもの酔いになったような感覚を感じる場合がある。カメラと人間の位置関係を注意深く検討し、望ましいビデオイメージを提供するための適切な制御を提供しなければならない。

(4) 心理的および社会的影響

① オフィス間および内部での会議

グループ間の会議の管理は困難な課題である。同じオフィスにいる人々が互いに感じる「存在感」は、メディアを介してコミュニケートしている仮想オフィスにいる人々に感じるそれよりも強いからである。同じオフィスにいる人々の物理的な近さは、ビデオによる隣人よりも、物理的な実際の隣人をより強く意識させる。この事実、およびネットワークを介した音声の品質の悪さによって、別のオフィスにいる人々との会話よりも、同じオフィスの人々との会話の方が盛んになるという傾向が見られた。

しばしば、2つのタイプの会話が同時に進行した。ひとつはパブリックなもので、人々はカメラに向かって発言する。また、各オフィスにいる人々の間でプライベートな会話が進行した。これら2つのタイプの会話をコーディネートし、適切なときに、パブリックな討論を主体とすることは、難しい課題であった。

このような大きな会議でのもう一つの問題は、多くの個人の表示サイズが非常に縮小されてしまうため、インタラクションの細かな点がしばしば見えなくなってしまうことであった。表情や非言語的な身振りなどが目立たなくなってしまう。システムを介した交流は、同じ部屋に居る人々の間の交流ほどの「現実感」を失ってしまう。

② 見つめることと目を見ること

参加者たちは相手のビデオ・イメージを見ているため、TVカメラを直接見つめることをしない。したがって、互いの目を見つめることは実現されなかったのである。

通常の対面コミュニケーションにおいては、相手を見つめたり、互いに見つめあったりすることは重要な一部分である。会話の61% は相手を見つめるし、31% は互いに見つめあうと推定されている。相手を見つめることは少なくとも5つの機能を果たす。すなわち、(i) 会話の流れを整理する、(ii) コミュニケーションが聞き手によってどのように受け取られているかのフィードバックを提供する、(iii) 感情を伝える、(iv) 人間関係の性質を伝える、(v) 地位の上下関係を反映する、の 5 つである。

これまでに得た最善の解決策は、広角レンズを持つカメラを人物の前、上、およびモニタのすぐ上に設置することである。カメラは人物にあまり近付け過ぎてはいけない。人物を近く見せるためにはズーミングを利用する。

③ 会議参加者の地位

もう一つの興味ある観察は、地位の情報を伝達する通常の空間的および非言語的な手がかりを失わせることによって、CAVECAT が社会的な地位の上下関係を変化させたことである。

フェース・トゥ・フェースの会議では、通常部屋の中の着席順が人々の地位の順序を反映し、地位

の高い人ほど中央よりあるいは上座に席を占める。

CAVECATの設計は、意図しないで独自の社会的な地位の手がかりを導入していた。4人の参加者による会議では、CAVECATは参加者のイメージを2×2の枠の中にデタラメに配置する。CAVECATは、また、会議を招集した人を基準にして会議のためのビデオ・イメージを構成した。つまり、短い休会の後別の人が会議再開を要求した場合、前とは異なったイメージの配置になった。これは、参加者にとって非常に気になることであった。それはあたかも、一旦出たあと部屋へ戻ったとき、みんながテーブルの新しい位置に付かなければならないようなものであった。

④ 会議のコーディネーション

このシステムにおいて、地位への伝統的な手がかりが失われたことと、新しい手がかりが生成されているという観察は、討論の制御という、もっと一般的な問題に関係している。重要な手がかりが失われたり、わかりにくくなったりすると、仲介役が発言順序やグループ意思決定のプロセスをコントロールする必要性が大きくなる。例えば、会話で主導権を握りたいと考えている人々は、しばしば発言したいという希望を示すため上体を伸ばすものである。この手がかりはビデオでは発見しにくい。

⑤ イメージのサイズと個人への影響

参加者が会話において効果的にふるまうことができるかどうか、および、各参加者が他者からどのように認識されるかは、一部分、ビデオ・イメージのサイズによって決定されるようである。大きなイメージで表示される参加者は、より大きな影響を討論に与えるように思われた。小さなイメージの参加者は、会話から離れており、効果的にふるまっていないと思われた。

ビデオ・イメージのサイズは、4つの要因によって決定される。すなわち、(i) モニタの画面サイズ、(ii) 見る人とTVモニタとの距離、(iii) 人間とカメラとの距離、および(iv) カメラのズームの設定である。しばしば参加者のイメージのサイズは異なったままであった。これらの変数は、めったに調整されないからである。

⑥ ビデオ・イメージと社会的距離

イメージのサイズと角度は、人々が自分と他の参加者との社会的関係をどのように認識しているかということと関係がある。不適切なイメージ・サイズは、人々が会話において過度に個人的であるか、あるいは過度に非個人的であるという印象を与えた。

お互いの間の距離が、その人間関係にとって不適切であると知覚されたとき、人々はたちまち居心地悪く感じることは、よく知られた事実である。接近し過ぎているならば、人々は自分の空間に侵入されたように感じる。離れ過ぎていても、居心地悪く感じるのである。

Media Spaceの場合には、問題なのは人々の間の「知覚された」距離である。物理的な距離よりも仮想的な距離である。観察によれば、一般的に言って、ビデオ・イメージは個人性が低く、控えめに受け取られる。Media Spaceについて、もう一つ通常と異なっていることは、、コミュニケーションに参

加しているグループのメンバのそれぞれについて、異なった人間間距離が同時に与えられているということである。これは、実世界における物理的な距離の場合と異なっている。実世界においては、人々の間の物理的距離は、ある意味ではネゴシエーションされ、共有されるのである。CAVECAT では、参加者の個人的な空間は、侵害者がそれと意識しないままに侵害される可能性がある。

⑦ プライバシと監視

自分が他のオフィスに接続されたときそれを「知る」こと、およびコネクションを拒否することができることは、非常に重要な必須機能であることは非常に初期の段階から認識されていた。最初の Media Space には、他者が突然自分のオフィスに入り込んできたことを示すフィードバック機能が不足していた。さらに、IIIF サーバにはプライバシ保護機能があるにはあったのであるが、使いにくいものであった。適切なフィードバックを提供するための良いアプローチの一つは、人の声ではないオーディオ・キューを利用することである。

3.3 マルチメディアソーシャルシステム

3.3.1 CRUISER

原文タイトル： Design of a Multi-Media Vehicle for Social Browsing

翻訳タイトル： 散策会談のためのマルチメディア装置の設計

著書： Robert W. Root
(Bell Communications Research, Morristown, NJ, USA)

出典： CSCW'88 Processings, September 1988, pp.25-38

本論文では、分散化された共同作業を支援するための、コンピュータを利用した通信技術への新しいアプローチについて述べられている。ここでは、事前に計画されたのではないインフォーマルな社会的インタラクションを可能とするツールに焦点が置かれている。開発されたCruiser と呼ばれるシステムは、マルチメディア通信チャネルを介して、他者への直接的アクセスを低コストで提供する「社会的インタフェース」であり、これまでの研究論文および仕事場についての 3 つの基本的コンセプト（散策会談、仮想的作業場、インタラクション・プロトコル）に基づいて設計されている。これらの設計要素を利用して、新しいシステム・コンセプトが提案され、さらに、デスクトップ・ワークステーションを介して社会的インタラクションを自動化したことの CSCW にとっての意味が検討されている。

(1) 「散策会談 (social browsing)」の重要性

職場における人間関係と共同作業の形成と保持の鍵となるのは、個人の間の良好なコミュニケーションである。オフィスの研究によれば、オフィスワーカの活動の65%は個人間のコミュニケーションを含んでおり、32%は直接対面によるインタラクションとなっている。物理的な近接性は、個人間のコミュニケーションへ与えるその効果を介して、人間関係の形成と保持にとって重要である。しかし、分散業務のもっとも顕著な特性は、関係者が物理的に離れているという事実である。この物理的な分離の与える影響を克服するために、ユーザはコンピュータによって仲介されたコミュニケーションを隣のオフィスに歩み入るのと同じくらい効果的で容易なものにするようなツールを必要とする。

接近性の効果は、構造的要素とダイナミックな要素の組合せとして現れる。構造的要素は、建築設計的および組織的な考察から引き出される。それらは、環境の物理的な特性および共同作業の相手となる人々を定義することによって、社会的インタラクションの舞台を設定する。ダイナミックな要素とは、単に、人々は日常の活動の中で職場の中を動きまわるという事実である。人々は昼食をとりに外出したり、コーヒーを飲んだり、仲間と雑談したり、コピーを取ったりする。これらオフィス内小旅行の過程において、彼らは同様の用事をしている他人に出会ったり、それぞれのオフィスで勤務している人たちの横を通り過ぎるであろう。職場の中を歩きまわることは、それぞれの場所での仕事仲間との予定された、あるいは予定されていないインタラクションの機会を頻繁に提供する。ここでは、人間のオフィス内での動きに基づいて発生する、このインフォーマルかつ直接的な社会的インタラクションのダイナミックなプロセスを「散策会談(social browsing)」と呼ぶことにする。散策会談は、職場内での社会的関係を形成し保持するための基本的メカニズムであると考える。

(2) CRUISERの概要

CRUISERを設計するに当たっての前提は、次のとおりである。

- デスクトップの完全動画ビデオ・コミュニケーション
- 高品質な全2重のオーディオ設備
- ローカルなコンピュータによって制御された交換マルチメディア・ネットワーク
- ビデオ・イメージとコンピュータによって生成されたグラフィックスの統合が利用可能

実際においては、プロトタイプ・システムのユーザは、本質的にはビデオ電話の設備、つまり、カメラを同軸的に装備されたビデオモニタ、モニタ上中央に設置されたハンドフリーなオーディオ設備、および交換マルチポイント・マルチメディア・ネットワークを介したオーディオ・リンクとビデオ・リンクに対するコンピュータ制御を持っていることを仮定する。

① 仮想的作業場

マルチメディア・ネットワークによって、物理的にはわれわれと同じ場所にいない人々も含むよう、

われわれの社会的インタラクションの範囲を拡大することができる。本質的には、それによって、高忠実度の仮想的環境を創造することができるのである。その環境の住人は、ネットワークへの加入者に限られる。この仮想的世界は、柔軟性に富みダイナミックである。物理的な近接性という制約条件は取り払われ、ユーザは社会的な環境を、建築設計上の制約ではなく、人間関係と仕事上の任務を中心に組織することができるようになる。

CRUISER は、仮想的な社会環境へのインタフェースのためのモデルであり、インフォーマルで自然発生的かつカジュアルな社会的インタラクションを支援するために設計されている。このネットワークは、他者へのマルチメディア接続（社会的リンク）を提供する。インタフェースは、いくつかの仮想的な通路を提供し、ユーザはそこを自由に歩きまわることができる。これらの通路には、ユーザが選択した、あるいはシステムがランダムに選択した人々が歩いていると考える。ユーザは仮想的空間を歩きまわり、自然発生的な他者との出会いを経験することができる。これは物理的な現実世界において、われわれがコーヒー・ルームへの道すがらだれかに出会うのとまったく同様におこなわれる。

CRUISER システムのユーザ・インタフェースは、ブラウジング・インタフェースとビデオ・ウィンドウという２つの基本的要素を持つ。ブラウジング・インタフェースは、CRUISER ソフトウェアとのインタラクションを可能とし、ビデオ・ウィンドウは遠隔地のカメラからの映像を表示する。実際のインタラクションは多くの形を取り得るが、例として仮想的世界の地図が、ブラウジング・インタフェースによってユーザのワークステーション上に、２次元のグラフィックスとして表示されていると仮定しよう。ユーザは、マウスあるいはその他のポインティング装置を使って、ブラウジング・インタフェースによるインタラクションを行い、場所や機能を選択する。この地図は、仮想的な通路を表現するから、それは任意の便宜的なレイアウト、例えば円周形でも良い。この地図が現在のコミュニティを表現する。その中には、プリンタなどの装置があってよいし、共通エリアや会議室のような「実際の」場所があってもよい。また、他の人々（コミュニティのメンバ）が含まれる。ユーザはマウスを利用して経路を選択したり、特定の目的地を選択することによって、この仮想的世界の中を旅行するのである。

② 「社会的交流インタフェース」の要素

CRUISER システムは、その力の大部分を２つのアイデアから得ている。まず、ユーザは社会的な出会いを求めて仮想世界を歩きまわるための多様なメカニズムを持つことができる。次に、仮想的世界は物理的世界からは独立しており、ユーザのニーズに応じて組織し、その中の住人を選択することができる。

(a) 仮想世界を歩きまわるためのメカニズム

自然発生的な社会的交流を求めて歩きまわる（social browsing）というメタファは、実世界においても歩きまわるという行動が、自然発生的な社会的インタラクションの基礎になっているという考え方

に基づいている。仮想的世界において、計画されたインタラクション、自然発生的なインタラクションの両方を支援するためには、少なくとも3つの異なったメカニズム（ジャンプ、計画された経路、ランダム・ウォーク）が必要である。ジャンプとは、ビデオ電話をかけるのと同じことである。ユーザが目的の相手をマウスで指定すると、その場所のイメージがビデオ・ウィンドウに表示される。これは直接的な計画された動きのためのメカニズムである。このアイデアを拡張して、CRUISERシステムでは、仮想的世界における経路というものを設定する。経路とは、いくつかの場所（オフィス、共通エリア等）およびそれらをアクセスする順序の集合である。つまり経路は、ユーザが仮想的通路をクルーズするときに訪問する場所の順序を表わしている。これが自然発生的なインタラクションを生成するための基本的メカニズムである。経路をアクセスする2つの方法をCRUISERは提供している。計画された経路においては、経路はユーザによって事前に定義されており、ユーザがそのオプションを選択する度にCRUISERはその経路を生成する。ランダム・ウォークでは、CRUISERは、仮想世界において計算されるある種の関数によって経路を作成する。関数には、完全にランダムな方法、共通の関心に基づく方法、あるいは、ある特定の人がオフィスに在席しているかどうかなどなんらかの条件に基づく方法などがある。これらの関数はユーザが定義する場合もあるし、ソフトウェアにまかせる場合もある。どちらにしても、それらの目的は、社会的インタラクションのリンクをカジュアルな形でアクセスすることであり、可能な多くのインタラクションに対して予測のつかない機会を提供することである。

経路とは、通路を歩きながら多くのオフィスのドアを通り過ぎるようなものである。また、たまたま立寄った形での出会いのサポートも考えられる。つまり、通路で進行中の会話（立ち話）に第三の人物が一寸立ち寄る形で参加することである。

(b) 視点

ビジタが仮想的通路をクルーズしているときに、事前に設定された経路をたどっているならば、彼は経路に従って、ドアの開いているオフィスを一つ一つ自動的に「覗き込む」。ビジタが開いているウィンドウを覗くと、そのオフィスのカメラが撮影したイメージが見えるし、内部のマイクロフォンが捕らえた音声が聞こえる。クルーズする人がオフィスからオフィスへ動くにつれ、映像は変化する。まず移動しているというイメージ（つまり、仮想的通路のグラフィカルなイメージ）が与えられる。つぎに、経路上の次の場所の映像が与えられる。一方で、オフィスの住人のウィンドウは、仮想的通路に向かって開かれている。ビジタがオフィスを覗き込むと、そのオフィスの住人には、ビジタのオフィスのカメラからのビデオ・イメージが見える。そして、クルーズする人が通り掛けに言うことはすべて聞くことができる。通路が無人である場合には、オフィスの住人は通路のニュートラルなイメージを見るだけである。このような設計は、日常のオフィス生活の観察から得られた社会的対称性の哲学から導出されたものである。実世界においては、自分が見られることなく、自分だけが相手を見

ることは一般的に言って不可能なことである。物理的世界のこのような特性を保存することによって、コンピュータによって仲介されたインタラクションの世界に、ある種の社交的な礼儀正しさを導入でき、人が勝手に他者から観察されないことを補償することによってプライバシ保護の要素を導入できるのである。

(c) 仮想的世界のイメージ

CRUISER システムにおいては、他者は社会的なリンクとして示される。一人のユーザと他のあるユーザとの間の、あるいはすべての他のユーザとの距離は、正に1単位、つまり1ノードである。仮想的な社会空間のすべての住人は、機能的には正確に同一の距離しか隔たれていない。この事実は2つの重要な意味を持つ。まず、空間的近接性の概念は仮想的空間においては意味がなくなる。すべてのユーザは同一の近接性を持つ。第2には、他者へのすべてのリンクはコンピュータによって制御されているから、仮想的空間の部分世界、つまりはビューを無限の多様性をもって扱うことができる。つまり、特定の目的のための部分世界（コミュニティ）を作成することができる。例えば、組織のメンバ、社交クラブ、作業グループなどである。コミュニティという考え方は、社会的背景を形成し、散策会談のための参加者のサブセットを定義するうえで有用である。

③ インタラクションのプロトコル

CRUISER は、実世界とのアナロジーによって設計されている。社会的なインタラクションを開始するための強力なツールを提供するとともに、仕事場での社交上のニュアンスを保つことを試みる。設計に当たって最も重視したことの一つは、オフィスでのプライバシと社会的インタラクションを管理するための適切なインタフェース機能を提供することであった。ここでは、仮想的職場の住人へのアクセスを規制するための、1組のプロトコルを提案する。このプロトコルは、職場での規範についての観察に基づいている。

現実の世界においては、視覚的・音声的な手がかりを利用して、訪問したい相手の状況を把握することができる。このような潜在的な手がかりを知覚するのは、われわれが社会的環境において経験をつんでいるからであり、われわれ自身が他の人の仕事中に割り込んで良いものかどうかを常に考えるからである。また、実世界でのオフィス住人が自分の面会可能性について明らさまに制御を行おうとする場合には、彼らはオフィスのドアを閉めたり、電話に出なかったりすることによって割込みを排除することができる。これらの行動は、明確なシグナルとなるとともに、オフィスに流入流出する情報の流れに対する物理的な障壁としての機能も果たす。これらの手がかり、シグナルおよび障壁は、一纏まりのプロトコルを構成している。われわれはこれらのプロトコルを利用して、職場の社会的構造を定義したり強化しているのである。

一方、仮想的世界においては、手がかり、シグナルおよび障壁は独立に操作可能であり、オフィス

の住人に割り込んで良いかどうかや、オフィスと通路の間の情報の流れに対して、広い範囲の制御能力を提供する。例えば、ビデオはONにしておき、音声の方はOFFにしておくなどの仮想的世界独自のプロトコル条件を提供することが可能である。さらに、システムはビジタが誰であるかを知っており、また、彼らのユーザ・プロフィルからのなんらかの情報を持っていると仮定するならば、ビジタあるいはビジタのクラスごとの個別の割込み許可状態を設定することも可能である。例えば、組織上の地位をもとにしてデフォルトの割込み特権を指定しても良いし、あるいは、ある友人には普通よりも高い割込み特権を与えても良い。CRUISERシステムの仮想世界においては、ユーザが割込み許可（つまりプライバシ）特権状態の階層に従って、インタラクションを制御できるようにすることを提案する。それぞれの状態には、あらかじめ定義した情報チャネルあるいは通信チャネルの構成、割込みレベル、およびオーバライドのための必要条件が付属しており、これがオフィスの住人およびビジタの双方に影響する。ユーザは、自分の利用可能な変数を用いて、新しい状態を定義することができる。しかし、ユーザ間の規範の整合性を保存するためには、システムの提供する標準的状態を変更することはできない。情報／通信チャネルの利用は、オフィス住人にとってもビジタにとっても対称的である。住人はビジタがチャネルを使えないようにしておいたままで、しかもチャネルをオープンにしてオフィスの外の世界をモニタすることはできない。

これらの状態は、ビデオ・ウィンドウ上のグラフィックス・オーバレイ（例えば仮想的なブラインドなど）を利用して、ビジタに対して知らされる。例えば、もし住人が「ビジタ受入れ可能」であるならば、ウィンドウはオープンであり、クルージングする人は、住人のオフィスの中の音を聞くことができるし、覗き込むことができる。このウィンドウは完全に閉めてしまったり、部分的に閉じることもできる。閉じられたウィンドウと切断された通信チャネルは、シグナルとしての役割も障壁としての役割も果たすことができる。

プロトコルに加えて、CRUISERシステムは、インタラクションを求めるためのドアベルと、外出中の人々を扱うためのメカニズムを提供している。住人のビデオ・ウィンドウを覗き込むだけで、その人がいるかいないか、自分と接触可能であるかどうかをチェックすることはできる。しかし、プライバシ上の理由から、CRUISERシステムでは、住人の活動についての連続的なチェックが出来るような機能は、表立ってサポートしていない。人々は仕事の上で動きまわるものであるから、クルージングする人がオフィスを訪ねてもその住人は外出中であるかもしれない。現実世界でのこのような状況では、われわれは椅子の上や端末の画面と言った目につきやすいところにメモを残す。CRUISERシステムの仮想的世界では、ビジタはオフィス住人のインタフェース・ウィンドウに、デジタル版の「メモ用紙」を貼り付けておくことができる。同様に、オフィスの住人もインタラクション・ウィンドウにビジタに宛てたメモを残しておくことができる。自動的な「お返し訪問」機能を提供することによって、メモを残すことの有用性を拡張することができる。メモにマウスを置くことによって、住人は直接的にメ

モを残したビジタのオフィスへジャンプすることができる。これらの機能は自然発生的なインタラクションを支援する上で有用である。

(3) 社会的インタラクションと仕事との統合

CRUISER は、職場での社会的インタラクションを自動化するためのツールである。このシステムが設置されると、社会的インタラクションは一般的なコンピュータ環境に機能的に統合される。これによってわれわれは、社会的インタラクションをまったく新しい観点から見ることができるようになる。つまり、社会的インタラクションは、職場のリソースの一つであり、仕事を進めるためのツールであり、作業が実行される背景の重要な要素と見ることができるのである。このような統合は、多くの形を取り得る。

基本的な統合コンセプトは、CRUISER のリンクとアプリケーション・プログラムの間の結び付けである。リンクは、個人、経路、あるいはコミュニティに対応する。1つのリンクがあるアプリケーションに結び付けられると、そのリンクは、ユーザが事前に記述しておいた条件に従って、対応するアプリケーションによって自動的に起動されるようにできる。また、リンクとアプリケーションとの間の結び付きは、そのアプリケーション環境の一般的な背景の一部分として保持しておくこともできる。第2のコンセプトは、適応である。社会的関係が静的なものであることはめったにない。それらが共同作業の一部分である場合には特にそうである。CRUISER にメモリを与えておけば、そのユーザの間のコミュニケーションのパターンのモデルを作るであろう。コミュニケーション・パターンを時間軸の上で比較し、結び付きのモデルを利用状況に適応させることによって、CRUISER システムは、社会的リンクの利用を、ある作業の領域におけるあるユーザ集合の中における活動に合わせて調整する、ダイナミックなシステムに変身するであろう。

また、リンクとアプリケーションとの結び付きというコンセプトを利用して、CRUISER において偶然的な出会いといったインタラクションを可能にすることができる。例えば、ユーザは、計画された経路あるいはランダム・ウォークを、文書の印刷のような、ある典型的なワークステーションでの活動に結び付けることができる。このようにしたならば、文書を印刷するという活動は、仮想的な散歩に出るための機会としての役割を果たすこととなる。文書がプリンタに送られるときには常に、CRUISER は指定された経路を起動する。ときどき、CRUISER は計画されたインタラクションに予測不可能な要素を取入れ、ビジタが最終目的地へ行く途中で彼をどこか別のところへ連れて行く。現実世界を真に模擬するためには、ビジタはときどきは会話ノードに衝突するようにすべきであろう。これは現実世界では、人々が通りすがりに通路での会話にぶつかることがあるようなものである。結び付けの可能性は殆ど無限である。重要な点は、CRUISER の社会的インタフェースが、電子的作業環境の社会的インタラクションへの拡張をサポートし、コンピュータによって仲介されたコミュニケーシ

ョンの新しい機能と力を増大させるためのメカニズムを提供していることである。

3.3.2 VideoWhiteboard

原文タイトル：	VideoWhiteboard: Video Shadows to Support Remote Collaboration
翻訳タイトル：	VideoWhiteboard：遠隔共同作業を支援するためのビデオ影絵
著書：	John C. Tang (System Sciences Laboratory, Xerox Palo Alto Research Center, Palo Alto, California, USA) Scott L. Minneman (System Sciences Laboratory, Xerox Palo Alto Research Center, Palo Alto, California, USA)
出典：	CHI'91 Proceedings, April 1991, pp.315-322

本論文では、VideoDrawという共同図面作成用プロトタイプ・ツールと、これを改良して開発されたVideoWhiteboard（遠隔共同図面作成を支援するためのプロトタイプ・ツール）について紹介されている。まず、VideoDrawについてレビューを行い、VideoDrawについての経験がVideoWhiteboardの開発に導いた経過が説明され、その後VideoWhiteboardとそれを利用する人々の反応について述べられる。最後にVideoWhiteboardのこの最初の利用経験で観察されたいくつかの問題点について、プロトタイプの設計および共同作業による図面作成活動一般の両面から検討されている。

(1) VideoDrawの概要

VideoDrawは、共同作業者がビデオ・スケッチパッドを共有できるようにしたプロトタイプ・ツールである。2人用のVideoDrawシステムの概念図を図3.3-1に示す。このツールは、ビデオ表示画面に向けられたカメラを接続している。ユーザはビデオ・ディスプレイの画面に対して絵を描き（乾式消去可能ホワイトボード用マーカを使用）、これらの図形とそれにともなう腕の動きがカメラによって撮影され、他方の画面に表示される。このメカニズムによって、合成された共有図面作成領域が作成され、その上で共同作業者たちは互いの描いた図形、それに関係する相手の腕の動きを観察することができる。

VideoDrawを利用する人々の観察によって、彼らはしばしば腕のジェスチャを使ってシミュレーションをしたり、相互の間のインタラクションの効果を上げていることが明らかとなった。また、しばしば、これらの身振りは図面作成スペース中の関係したスケッチを参照しながら行われている。VideoDrawを利用することによって、遠隔地の共同作業者たちは、（結果としての図面を見るだけで

なく）図面の作成およびそれへの参照のプロセスを見ることができる。これは、共同図面作成作業において重要な資源である。共同作業者たちは、ときどき、同時に同じ場所へ書き込むためにVideoDrawを利用することさえあった。これによって単一のドローイング・サーフェスでは不可能なインタラクションが可能になったのである。さらに、VideoDrawは、遠隔地に居る共同作業者たちの間に、共存感覚を提供するための新しい方法について示唆を与えた。一緒に作業している相手の腕のビデオ・イメージがドローイング・サーフェスの上に見えることは、例えばコンピュータ化されたスケッチパッドの上でカーソルが動く場合とは別の種類の存在感覚を与えるのである。

VideoDrawの利用において観察された限界は次のとおりである。

ビデオ表示画面が相対的に小さい（20インチ対角）ので、画面一杯にしても、描くことの出来るテキストとグラフィックスの量が制限される。ディスプレイの蛍光体レイヤ（相手の図形が表示される部分）とディスプレイのガラス面（自分の図形が描かれる部分）との間のガラスの厚みによる視差によって、双方のユーザの作成した図形を適切に位置付けることが困難となった。ユーザは自分の画面上に自分が作成した図形を消去することができるのみであり、ときどき相手の図形を消去するよう相手に要求する必要があった。上向きに置かれたCRTディスプレイを相手に作業し、しかも自分の頭でオーバヘッドカメラの邪魔をしないようにするのは、居心地の悪いものであったし、視差の問題をさらに難しくした。これらの観察結果は、VideoDrawの設計を修正することを示唆するとともに、VideoWhiteboardの設計へ導いたのである。

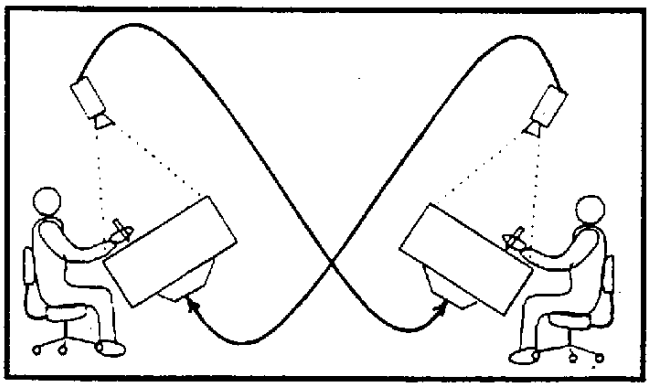


図3.3-1　2人用VideoDrawの概念図

(2)　VideoWhiteboardの概要

VideoWhiteboardは、遠隔地の間で共有される大領域の図面作成スペースを提供するビデオベースのプロトタイプ・ツールである。2つのサイトで共有されるVideoWhiteboardシステムの概念図を図3.3-2に示す。

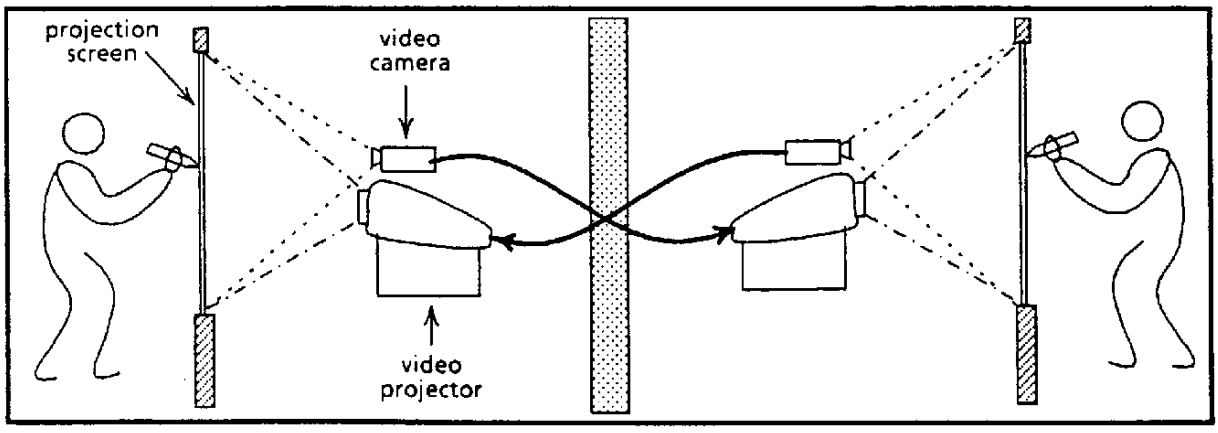


図3.3-2　2つのサイトにおけるVideoWhiteboardシステムの概念図

それぞれのサイトには、壁掛け型の背面投射型スクリーン（およそ4.5フィート(= 137 cm) × 6フィート(= 183 cm)）、ビデオカメラ、ビデオプロジェクタ、および適当なオーディオ装置が設置されている。ユーザは、スクリーンの前面のなめらかな表面に、標準の乾式消去可能ホワイトボード・マーカを使って絵を描く。ビデオカメラは、ユーザから見てスクリーンとは反対側に設置されており、画面全体のイメージを撮影し、他のサイトのビデオ・プロジェクタに送信する。送信されたイメージは、スクリーンに投影される。それぞれのユーザがスクリーン上に図を描くと、それらの図形は、カメラによってイメージされ他方のスクリーンに投影されるのである。図形のイメージのほかに、カメラは遠隔地の共同作業者の影絵を送信する。共同作業者は、自分で書いた図形と「ビデオで投影された」図形の合成像、および遠隔地の共同作業相手の身振りや行動の影絵を見ることになる。音声のコネクションによって、作業者同士は互いに会話することができる。ユーザは、ホワイトボードを共有して共同作業している場合とまったく同様に、VideoWhiteboardのスクリーンに対して、書き込み、描き、消去し、ジェスチャを行うことができる。

それぞれのサイトのユーザは、表示画面に対して正しい方向感覚を共有する。「右」と「左」は、どちらのサイトにいる人にとっても同じ意味を持つ。カメラはユーザから見てスクリーンと反対側に設置されているため、実際には右左が逆転されたイメージが撮影されるのである。ビデオ・プロジェクタは、実際にはスクリーンの裏側に向かって投影しているのであるが、フロント・プロジェクション・モードで投影することによって、この鏡像は補正される。

VideoWhiteboardにおいては、図形を描いたり、影絵を投影したりするための入力スクリーンは、遠隔地の共同作業者の図形や影絵を投影するための出力スクリーンと同一である。したがって、ユーザは、遠隔地の共同作業者の描いた図形の上に直接自分の図形を追加して描いたり、ジェスチャを加えることができる。

(3) VideoWhiteboard と VideoDraw の比較

VideoDraw と同様に、VideoWhiteboard は、それぞれのユーザが図面作成スペースを共有し、その共有スペースにおいて、自然な方法で描き、身振りし、インタラクトすることを可能にする。それぞれのユーザに対する図面作成スペースの見え方は共通であり、図面作成スペースの中の対象に対する口語的表現での参照が意味を持つ（例えば、「これ」「ここの」など）。ユーザは遠隔地にいる共同作業者が作成したスケッチを直接補正したり、インタラクトすることができる。彼らはスケッチに対してジェスチャし、遠隔地の相手は、参照しているスケッチに対するそのジェスチャを理解することができる。遠隔サイトのユーザは、図面作成スペースに対して同時並行的にアクセスすることさえできるので、一度に同一の領域に対して複数の人間が作業することが可能である。異なったサイトにいる2人の共同作業者の影絵のイメージは互いにスーパインポーズされる。

VideoDraw とは異なって、VideoWhiteboard は、大きなスクリーンを持っており、ユーザはより多くの図面作成領域を利用できる。この大画面によって、同じサイトでも、複数の作業者が作業に参加することが可能になる。VideoDraw では、ジェスチャのフルカラーの画像が表示されたのに対して、VideoWhiteboard は、遠隔地の共同作業者の影絵を見せる。VideoDraw では主として腕の動きを伝達したのに対して、VideoWhiteboard は、上半身のジェスチャを伝達する。VideoWhiteboard では、カメラとユーザはスクリーンの反対側に位置しており、ユーザの頭部がカメラの画面撮影を妨害するかもしれないという VideoDraw の問題点を解決している。VideoWhiteboard では、視差の問題も軽減されている。VideoDraw における視差の問題点は、軸から外れたオーバヘッド・カメラから画面作成表面を撮影しなければならないことによってより困難なものとなっていたのである。このようにして、VideoWhiteboard は、VideoDraw における限界のいくつかを解決したのである。

(4) VideoWhiteboard の利用状況の観察

同一ビルディングの2つの部屋に設置された VideoWhiteboard を2人の共同作業者が約1時間にわたって遠隔作業するのを2回観察した結果、次のような観察結果が得られた。

① VideoWhiteboard の利用における特徴

VideoWhiteboard の利用に対する最初の共通的反応は、遠隔地の共同作業者が、遠隔のサイトではなく、スクリーンの反対側にいるように感じるというものである。遠隔地の相手の影絵をスクリーンで見るというビジュアルな効果は、いく人かのユーザにとっては、自分が反透明なガラスのスクリーンを介して相手に話しかけているような印象を与えるのである。あるユーザは、相手の言うことを明瞭に聴くことができなかったので、耳をスクリーンに対してそばだて、もう一度言って下さいと相手に呼び掛けた。音声を拾っているスピーカフォーンは、口の反対側に置かれていたにもかかわらず、その身振りが目的の反応を相手から引き出し、相手はコメントをもう一度大声で繰り返したのである。

半透明のスクリーンの反対側にいる相手とインタラクトしているという印象は、本当ではないのにかかわらず、この錯覚が相手とインタラクトするためのユーザの能力の邪魔をすることは殆どない（むしろ、助けとなることがある）。ユーザは、自分が遠隔地の相手とスクリーン上の図形を共有しているのであることを、すばやく認識する。また、彼らはスクリーンの上に投影される相手の影絵をそれとして認識することができる。

VideoWhiteboard についての興味ある特徴の一つは、遠隔地の共同作業者の影絵が、図形やスケッチの表示されたスクリーン・イメージの上に直接スーパーインポーズされることである。この機構によって、注意の分散が大きく軽減される。他の共同作業支援システムにおいては、参加者は、相手を見るか、作業領域を見るかを常に選択しなければならないのである。ホワイトボードを囲んでの直接対面によるインタラクションの場合には、仲間の作業者はホワイトボードに向かって図面を描いている自分の横あるいは後にいる。VideoDraw のプロトタイプを利用する場合には、共同作業者は、うつむいて図面を見るか、顔を上げて相手を見るかを交互に繰り返す必要があった。VideoWhiteboard は、図面作成領域と遠隔地にいる相手の影絵を同じ観察角度に置くので、相手の動作と図面作成領域に同時に注意を向けることを可能にする「ヘッドアップ・ディスプレイ」効果が提供される。

VideoWhiteboard は上半身全体のジェスチャを伝えるのであるから、ある意味では、VideoWhiteboard は、腕のジェスチャを伝達するという VideoDraw の能力をさらに拡大したものである。
VideoWhiteboard は両腕を含む大きなジェスチャや、人々がインタラクションにおいて相手の反応を求めたり、反応を表明する際に自然に使っているボディランゲージ（肩をすくめるなど）さえも伝達することができる。

VideoWhiteboard は、フルカラー・イメージを表示する VideoDraw のようには、3 次元感覚を与えないが、スクリーンの周囲のライティングによって、投影された影絵にシェードの違いをつけ、物体がどの程度スクリーンに近づいているかの感覚を与えることができる。したがって、ユーザは、投影された影絵の密度および鮮明さによって、遠隔サイトにいる共同作業相手、あるいはその他の物体がどの程度スクリーンに近いかを知ることができる。

② VideoWhiteboardの利用における限界

VideoWhiteboard の利用におけるいくつかの限界も観察された。影絵はある種のジェスチャ情報を伝達するには効果的であるが、フルカラーのビデオ・イメージの持つ豊かさを欠いていることは、明らかである。特に、影絵では、遠隔地に居る共同作業者間のアイコンタクトは不可能である。遠隔地の相手の影絵だけを見ていて、注意の集中した、長期間のインタラクションが可能であるかどうかは、不明である。

影絵によるマスク効果は、1 つのサイトに 2 人以上の共同作業者がいる場合に特に問題となる。これはVideoWhiteboard のインフォーマルな利用において、何回か実際に発生したことである。複数の遠

隔共同作業者とインタラクトし、しかも彼らが影絵でしか見えない場合には、その判別がしばしば困難となる。幾人かのユーザは、このような状況においてはだれとインタラクトしているのかわかりにくいため、居心地悪く感じたと報告した。この問題は、ステレオ・オーディオ装置を利用し、どの声がどの影絵に対応しているかについての位置的手がかりを提供することによって軽減することができるかもしれない。

ユーザが実際に行うことと、遠隔地の共同作業者が投影された影絵によって見ることができるものとの間の本質的な非対称性は、インタラクションにおけるいくつかの問題を引き起こす可能性がある。1つの位置を正確に差し示す腕のジェスチャ（例えばポインテイング）と、スクリーンから多少離れたところで行われるあいまいなジェスチャ（例えば、首肯くこと）は、認識することが困難である場合がありえる。同様に、もし、ユーザがスクリーンから十分離れたところに立ち、認識可能な影絵を投影しなかったならば、遠隔地の共同作業者は、彼女との接触できなくなったように感じるかもしれない。影絵は大部分スクリーンの背後に置かれた光学機器によって生成される人工的な現象であるから、ユーザは、自分がどのような影絵を遠隔地の共同作業者に投影しているのかについては、視覚的フィードバックを持たないのである。この点については、VideoWhiteboardを介したインタラクションを保持するため、ユーザがいろいろの方法でこの潜在的問題を補っている様子が観察された（例えば大げさなジェスチャ、スクリーンに十分近いところに留まり、互いに常に影絵を見ることができるようにするなど）。

VideoWhiteboardには、大きなスクリーン領域があるが、使用する特定のビデオ技術の解像度による制約は存在している（プロトタイプにおいては、およそ330×240本のテレビ走査線である)。したがって、ユーザは、テキストとグラフィックスを解読可能にするため、やや大きめに書く必要があった。また、ビデオ投影技術は、フリッカ現象を起こす傾向がある。これは、このシステムの場合のように接近した位置で見る場合（すなわち、スクリーン上に描く場合）には特に問題である。それぞれのカメラによって撮影される領域は、それぞれのビデオプロジェクタによって投影される領域と正確に一致していなければならないから、VideoWhiteboardシステムを構成するには、光学機器の注意深い調整が必要である。スーパーインポーズされたイメージの光学的調整は、スクリーンの中央でのみ最適化されており、スクリーンの端に行くに従って段階的に劣化する。

VideoDrawの場合と同じく、ユーザは、自分のサイトのローカルなVideoWhiteboardスクリーンに描かれた図形を消去することができるのみであり、相手側のスクリーンに描かれた図形を消すことはできない。また、ユーザは、スクリーン上のイメージを保存、検索、印刷するための便利な手段を与えられていない。また、以前に作成したイメージに対するアクセス手段も与えられていない。

3.4 オフィスプロシージャシステム

3.4.1 ECF

原文タイトル： Support of Cooperative Work by Electronic Circulation Folders

翻訳タイトル： 電子循環フォルダによる協調作業の支援

著書： B. Karbe, N. Ramsperger, P. Weiss

出典： ACM SIGOIS Bulletin, Vol.11, No.2, April 1990, pp.109-117

本論文では、ProMInanD (Extended Office Process Migration with Interactive Panel Displays) プロジェクトで開発された電子循環フォルダ (Electronic Circulation Folders : ECF) について、その機能やインプリメント、および評価が紹介されている。

(1) 循環フォルダ (CF) と電子循環フォルダ(ECF)

オフィス作業はいくつかのステップからなり、オフィス・ロール（オフィスにおける役割）を果すオフィスの人間がこれを実行する。これはより高いレベルの活動の一部と理解することのできる包括的な活動であり、判断や失敗、予期しない反応などによって、予想できない、あるいは非決定的な行動を取ることのある人間がかかわることで問題は難しくなる。

オフィス・タスクの処理を支援するツールとして、これまで広く用いられてきたものに循環フォルダ(CF) がある。これは、オフィス・ワーカが作業を行うべき、任意ではあるが作業に関連した文書を内容としており、CF の表紙にはオフィス・ワーカの宛名が記載されている。社内のメッセンジャ・サービスが、あるオフィス・ワーカの発送書類入れの密閉した CF を、そのオフィス・ワーカが CF の表紙に書いた宛先の到着書類入れに運ぶのである。CF の内容に対する作業と、CF の組織内部での移送ははっきり分れている。

CF は、オフィス・タスクのための作業を移動するツールとしては非常に柔軟性の高いものだが、欠点もいくつかある。まずメッセージ・サービスに時間がかかる点があり、また宛先のオフィス・ワーカが不在である可能性が考慮されていない。さらに、ファイルの行方が重大な問題となることもあり、これは関連するオフィス・タスクが形式化されていないとさらに大きな問題となる。

ProMInanD システムは、CF の代りに電子循環フォルダ (ECF) を使って、オフィス・ワーカが CF で慣れている柔軟性を提供するものである。ProMInanD システムは、従来の CF と電子 CF 間の関係を示すことでオフィス・ワーカにその全内容を説明できるよう設計されている。あるタスクに関連したタイプの ECF インスタンスが、移送仕様に基づいて、そのタスクにかかわるオフィス・ワーカに自動的に移送される。ECF の内容に対する作業は、いわゆるステップ・プログラム、すなわち ProMInanD で

はない任意のシステムに支援され、その自動呼出は移送仕様で決定される。移送は組織の情報によって決り、オフィス・ワーカが影響を与えることもできる。ひとつのオフィス・タスクを分割した複数のサブタスクに対する並行作業は、1つのECFを先祖とし、いわゆる従属ECFをその子孫とするECFファミリによって支援することができる。

(2) ECFの仕様

① 表現方法

ECFは、説明部と内容部という大きな2つの部分からなっている。説明部は、ECFの移送仕様とECFのタイプ別の状態情報、システム全体で一意の識別、他ECFとの関係、進捗状況及び実際にオフィス・ワーカがこれまで実行したステップの履歴に分れる。内容部は、ECFの作業内ステップを実行するのに必要な文書やフォームを保持するワークパート、オフィス・ワーカが編集できるオプションのフォルダ・スリップ、オフィス・ワーカが追加した文書を入れるためのオプションの付録からなる。

オフィス・ワーカが関心を持つのはECFの内容部で、オフィス・タスクに対する本質的な作業、例えば「請求オーダ」タスク用のECFであれば、請求オーダ・フォームの記入などの作業を行う部分である。行うべき作業の種類は移送仕様が決定するが、ECFがどこからきてどこへ行こうとしているかはオフィス・ワーカも関心を持つことなので、移送仕様をグラフィック表示で探索できるようになっている。

② 移送仕様

ECFの移送仕様は、実行すべきステップの形で移送ルートを決定する。簡単に言うと、どのオフィス・ワーカがステップを実行しなければならないか、どの文書を使うか、そのためにどのアプリケーション・プログラム、いわゆるステップ・プログラムを呼出すかをステップが定義するのである。オフィス・ワーカは、組織としての機能、いわゆる「オフィス・ロール」によって定義される。個々のステップに関するこのような情報以外にも、移送仕様は次のような情報を持っている。

・ 全体的に見てタスクを実行するのに必要なすべての文書
・ かかわりのあるすべてのオフィス・ロール
・ すべてのステップの順序と、それがオフィス・ワーカによって実行されるのか、自動的に実行されるのかどうかの区別
・ 複数のステップのサブタスクへのグルーピング
・ ステップ・プログラムの実行結果によるそれ以後の移送方法
・ 従属ECFの自動起動

ECFの各タイプについてそれぞれ移送スキーマが存在する。あるECFのタイプの新しいインスタン

スを生成すると、その移送仕様は対応する移送スキーマを適正に加工し、パラメータを定義することによってつくられる。

「パフォーマ・ロール（実行者の役割）」も移送仕様を決定するパラメータの1つで、一部がECFの生成時に、一部が移送中に設定される。「組織説明」は、どのオフィス・ワーカがどのオフィス・ロールを果すことができ、そのロールでどのECFのインスタンスが生成されるかを指示する。具体的なオフィス・ワーカ（すなわちパフォーマ）が、適したロールになってECF、例えば請求オーダのインスタンスを生成すると、その生成時、オフィス・ワーカがシステムに知らされる。移送仕様は、この例では、次に部門長による「認可」が実行されなければならないと記述されていることもあり得る。このような場合、実際に誰が実行しなければならないかは、ECFのインスタンスを生成したオフィス・ワーカが、組織説明から具体的な部門を探索し、その部門の長の役割を果す具体的なオフィス・ワーカを探すことで行う。このような宛先の解明は、移送中できるだけ後の方で行われる。

ステップとECFの特性が、例外処理に関する情報を与える。一定の状況下で省略できるステップもあるが、「認可」のように強制的なもので省略できないステップもある。

ステップ・プログラムの結果と、そのECFの以後の移送との間には従属関係が存在し得る。上の例で認可が行われなかったら、そのECFをそれ以上通常に処理することは意味を持たない。このような従属関係は、アプリケーションに固有の決定プログラムによって評価される。このプログラムは移送仕様の中でも参照される。

複雑なオフィス・タスクは、組織の各部門に関連するサブ・タスクに分割できることが多い。サブ・タスクをどのように実行するかは、その部門の責任となる。他部門のオフィス・ワーカは、そのサブ・タスクがどのようなステップで処理されたかということを知る必要はないので、移送仕様のグラフィック表示では、そのサブ・タスクはステップのない1つのアイテムとして示される。サブ・タスクの概念を取り入れることにより、特に移送仕様の定義を簡略化できる。

ECFの移送仕様のスキーマを作成あるいは修正するため、ProMInanDはオフィス作業のオーガナイザが利用するグラフィック移送仕様エディタを備えている。

③　ステップ・プログラム

オフィス・ワーカの本質的な作業、すなわちECFのワークパートに記述された文書に取り組むことによるオフィス・タスクの処理を支援するには、いわゆるステップ・プログラムがアプリケーション開発者によって提供されなければならない。標準的なステップ・プログラムは、テキスト・エディタ又はフォーム・ハンドラである。これらの支援するアプリケーションは、より専門化されたステップ・プログラムであるのが普通で、ProMInanDがインストールされたオペレーティング・システム上で動くプログラムは、すべてステップ・プログラムとして使うことができる。移送システムのデータベースに適切な情報を入力するだけで、新しいステップ・プログラムを移送システムに知らせること

ができる。

(3) ProMInanD の機能

ProMInanD の機能としては、オフィス・ワーカの自分自身の作業を支援するオペレーションとその結果を移送する作業を支援するオペレーションの2通りがある。

オフィス・ワーカがProMInanD にログインすると、電子デスクが表示される。電子デスクによって、自分のオフィス・ロールによる選択を行ったり、他のツール、すなわち到着ECF の入っている到着書類入れや、作業を行ってデスクを離れたがまだ完全に終了していないECFに関する情報を入れた発送書類入れ、作業の中断したECF バイルや、スタートできる「空の」ECF の入ったフォーム・ボックスなどを起動することができる。

① ローカル・ワークを支援するオペレーション

どの時点でも、オフィス・ワーカのデスクにあるECF は、ツールの中にあるか表示されているかのどちらかである。オフィス・ワーカが役割上実行を許されていれば、到着書類入れやフォーム・ボックス、バイルの中のECF を選択することができる。選択されるとECF は表示され、オフィス・ワーカはECF の履歴と状況に関する情報を得る。ECF をオープンして内容に関する作業を開始することができる。作業を行うことのできるステップ・プログラムがいくつかある場合、オフィス・ワーカは好きなステップ・プログラムを選択できる。オフィス・ワーカは、ECF をバイルにしまうことで、作業を現在のステップで中断することができる。作業を完了したい場合は、いくつかあるモードの1つで転送することで、これが発送書類入れに入れられる。

デスク上のECF に、通常のローカルなワークフローからの逸脱とも考えられる影響を与えることもできる。cancel step と delete ECF がそれで、オフィス・ワーカがステップをキャンセルすると、それまでそのステップで実行された作業はすべてキャンセルされ、ECF は発送されたばかりと同じ状態になる。オフィス・ワーカがECF を削除すると、スタートさせたことのなかったかのようにデスクから除去される。しかしECF の削除は、ECF が最初のステップにある場合にのみ可能である。

② 移送を支援するオペレーション

ECF の通常の移送を取り扱うオペレーションは、forward、postpone、inform である。これらは、表示されたECF あるいは発送書類入れに対して、オフィス・ワーカが利用することができる。

ステップの作業が完了すると、オフィス・ワーカはECF を forward（転送）することができる。これによってこのECF はデスクを離れる。ステップが最後まで行っていなければ、ProMInanD は次のステップに定義されたオフィス・ロール ×× を果すオフィス・ワーカに、ECF を配送しようとする。この役割を果すオフィス・ワーカが2人以上いる場合は、ProMInanD が選択を行う。この役割を果すものが誰もいない場合は、ECF は「×× ロール のオフィス・ワーカに移送中」という状態に遷移し、後

にオフィス・ワーカがこの役割に入った時に配送される。配送することができないでいても、次のオフィス・ワーカについて詳しいことがわかったら、「××ロールを○○オフィス・ワーカ に指定」という状態に遷移することもある。

ステップの作業が中断した場合、オフィス・ワーカはECFをpostpone（延期）することができる。ECFは、「××ロールの○○オフィス・ワーカ の再提出まで延期」という状態に遷移し、オフィス・ワーカが指定したタイム・リミット後に提出される。

オフィス・ワーカのデスクを離れたECFは、システムを離れない限り発送書類入れに記録されている。発送書類入れのECFにinform（通知）オペレーションを行うことで、オフィス・ワーカはECFの現在の所在に関する情報を得ることができる。オフィスでよく発せられる「今ファイルはどこ？」という質問に対する答えを得られることは、オフィスの作業スタイルと協調にも影響を与えることになる。

通常の移送ルートから逸脱させるためのオペレーションもあり、これらは、not me、refer back、append、delegate、shortcut、shift、fetch back、cancel ECFである。どれも、表示されたECFあるいは発送書類入れに対してオフィス・ワーカが利用することができる。

オフィス・ワーカが、現在のステップで作業しなければならないのは自分ではないと苦情を言う場合には、not me（自分ではない）モードでECFを転送することで、前のオフィス・ワーカに返送される。

さらに情報が必要な場合があるが、オフィス・ワーカはその要求をフォルダ・スリップを記載するよう促される。ECFをrefer back（差戻し）モードで転送することで、ECFは前のオフィス・ワーカに配送される。

append（追加）によって、オフィス・ワーカは、移送仕様の指定に加えて、現在のステップの後にさらにステップを追加することができる。新しいステップのステップ・プログラムは、現在のものから引継がれるが、これを受取るオフィス・ワーカか組織のユニットを指定しなければならない。これは、オフィス・ワーカが現在のステップを実行あるいは補足するため他のオフィス・ワーカ（通常は部下）を必要とすることがあるという考え方によるものである。これを簡単に一般化したものがフリースタイルのECFとなる。これはたった1つのステップで始まり、続くステップが全て追加となるものである。これによって形式化されていないオフィス・タスクがすべて支援できることになる。

delegate（委任）オペレーションによって、現在のステップに2つのステップを追加することができる。1つ目のステップはappendと同様に追加されるが、2つ目は委任するオフィス・ワーカに戻される。これは、委任したオフィス・ワーカが結果を取り戻して責任を負うという考え方によるものである。

一定の状況下では、ステップを後ろへshift（シフト）したりshortcut（近道）したりすることができる。シフトされたステップは後に作業を行うが、近道したステップは排除される。シフト・オペレーショ

ンは、シフトしたステップ担当のオフィス・ワーカがしばらく不在の場合などに、間のステップを一時的に優先させるものである。

発送書類入れに記録されているECFをfetch back（取戻し）したいオフィス・ワーカもいる。ECFが延期されていると、これによって不完全な再提出が行われることになるが、そうでない場合、ProMInanDは、次のオフィス・ワーカが到着書類入れから選択していない限り、ECFを取り戻そうとする。

進行中のオフィス・タスクが陳腐化することもある。そのような場合、対応するECFのイニシエータは「ECFをキャンセル」したいと望むが、ECFはキャンセルによる放棄が既に不可能な状態に達していることがある。これは移送仕様で決定される。放棄がまだ可能な場合でも、放棄までに既に実行したステップの作業を補償するためアプリケーションに依存したステップを追加して行う必要がある場合もある。

(4) ProMInanDのインプリメント

ProMInanDの現行バージョンは、TCP／IPを有するUnixとSun Viewのウィンドウ・システムで走るSunワークステーションで実現される。ProMInanDは全てObjective-Cで記述されている。

ProMInanDのユーザ・インタフェース部分は、他の主要部分から完全に独立するよう開発された。さらに、Objective-Cのグラフィック・ファウンデーション・ライブラリであるICpak201を使って開発されており、特にグラフィック及びウィンドウ向きユーザ・インタフェースに適している。ICpak201の最下部コンポーネントである、いわゆる「アースベース」が、ワークステーションにあるウィンドウ・システムにインタフェースをマップする。そのため、ProMInanDのユーザ・インタフェースを他の目的システムに移植する場合は、アースベースを取替えるだけでよい。

ProMInanDの移送システム(MS)は、各オフィス・ワーカのワークステーションで走る複数のローカル移送サーバ(LMS)と1つのグローバル移送サーバ(GMS)からなる。簡単に言うと、LMSはワークステーションに配送されたECFを扱い、GMSはワークステーション間のECFの移送を制御することになる。

移送仕様は複雑なオブジェクトで、ECFを指定すると同時にこれを制御するが、そのためには、移送仕様を解釈し、特に例外処理に関してはダイナミックに修正しなければならない。Objective-Cで書かれているためオブジェクトの受け渡しや、格納、読取り、再起動ができるので、移送仕様はECFに追従できる。そのため、オブジェクトを異なるプロセス間で簡単に共有することができるのである。

ProMInanDは、分散型データベースシステムに保持される電子的組織ハンドブックに広範囲に依存しているため、スロット・メカニズムの拡張と組合わされて、ECFを配送すべきオフィス・ワーカの宛先が配送直前にわかるようになっている。このように、組織説明がMSのアルゴリズムと別になっ

ているため、指定されインスタンス生成までされたECFは、組織のある程度の変更には影響を受けない。

(5) 評価

ProMInanDは本論文執筆時点で比較的成熟した状態にあるが、さらに付け加えられれば望ましいものや開発中のものは次の通りである。

- ECFの寿命は数日あるいは数週間以上であるため、その間、ECFの新バージョンが開発されたり、組織が変更されたりすることがあり得る。このようなケースを適正に処理するため、ECFのバージョン管理が必要である。
- 終了したECFの内容は記録することができる。この記録を活用するため、有効な記録システムが必要である。
- 大きな組織では、グローバル移送サーバが1つでは大人数のオフィス・ワーカを処理できない場合が十分に考えられる。そのため、広域ネットワークも統合できるクラスタ・コンセプトを開発中である。

3.4.2 DOMINO

原文タイトル：	Experiences with the DOMINO Office Procedure System	
翻訳タイトル：	DOMINOオフィス・プロシージャ・システムの利用経験	
著書：	Thomas Kreifelts	(GMD, Germany)
	Elke Hinrichs	(GMD, Germany)
	Karl-Heinz Klein	(GMD, Germany)
	Peter Seuffert	(GMD, Germany)
	Gerd Woetzel	(GMD, Germany)
出典：	ECSCW'91 Proceedings, September 1991, pp.117-130	

本論文では、GMDにおいて開発され利用実験されたDOMINOと呼ばれるオフィス・プロシージャ・システムについて紹介されている。

(1) オフィス・プロシージャのモデル化

オフィスにおける事務処理の流れの円滑化を図るためにDOMINOと呼ぶシステムを開発し、これをGMD内で試験的に運用した。

DOMINOシステムは、オフィス内の各個人の物品購入願や出張・休暇願等を該当する管理者に提出

し、管理者がこれを処理するといった事務処理を半自動化して、オフィスのコミュニケーションの円滑化を図るものである。

本システムの設計の際に前提となった点は、次のとおりである。

① オフィスワーカは、各々プライベートな作業ドメインを持っており、これらのドメイン間をメッセージが流れることによって共同作業が成り立つこと

② これらのメッセージは、共同作業においてスピーチアクトの理論に基づいて処理されること

③ 組織化されたグループにおける共同作業は、各作業ステップが特定の入出力によって結び付けられて処理されること

④ 各作業ステップにおけるこの特定の入出力は「理想的」に行われること

このような前提に従って、本システムではオフィスにおける様々な共同作業をPetri netの考えに基づいてモデル化を行った。モデル化の例を図3.4-1に示す。これは物品購入願が受理されるまでの処理を表わしている。

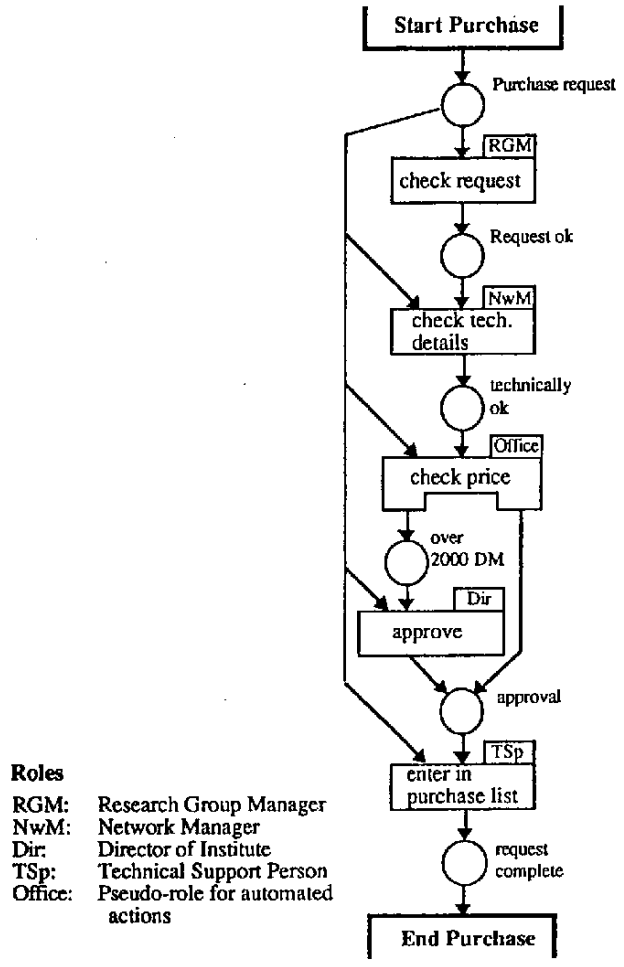


図3.4-1　DOMINO オフィス・プロシージャの例

(2) DOMINOシステムの構成

システムの構成は、各個人が所有するMacとホストであるUNIXマシン（SUN）とをEthernetで接続している。ホスト上には「仲介役（Mediater）」と呼ばれるエージェントが動作しており、これが電子メールの自動的な疑似ユーザとなって各ユーザとのメールのやり取りを行う。この電子メールはCoPlanXと呼ばれる独自のプロトコルで通信が行われる。「仲介役」は組織やプロシージャに関するデータベースを管理しており、このデータベースを解釈して共同作業の処理の流れに沿ったメールの送信を行う。図3.4-2にシステム構成を示す。

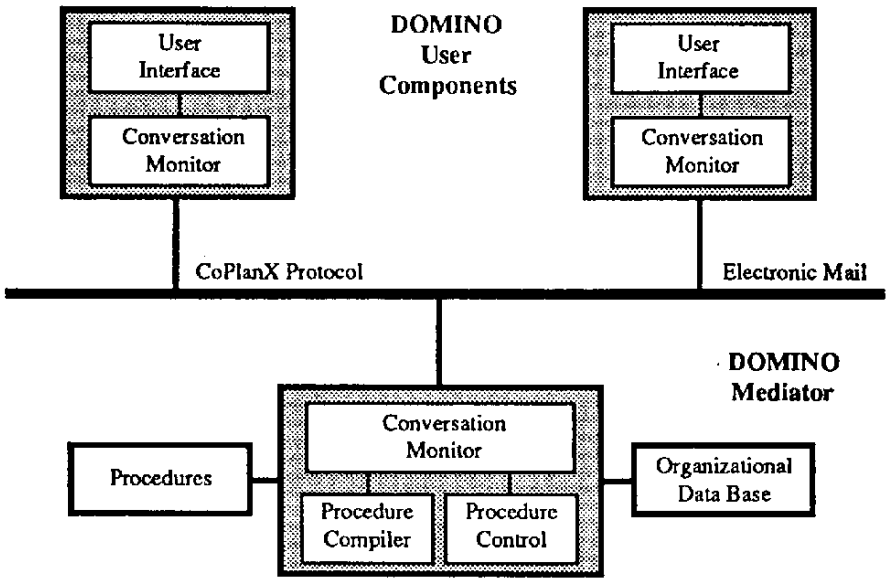


図3.4-2　DOMINOシステムの構成

(3) DOMINOシステムのインタフェース

ユーザインタフェースは、MacのHyperCardを利用してインプリメントされている。
購入願などのウインドウは定型のフォーマットで、必要な箇所だけをキー入力すればよいようになっているが、不定型のデータ（図形など）もクリップで留めるようなイメージで一緒に送受することが可能である。図3.4-3にユーザのウインドウ上に表示された購入伝票のイメージを示す。

(4) DOMINOシステムの利用結果

本システムをGMD内で運用してみた。ユーザとして、5つのグループに分かれた120人の研究員を対象とした。この利用結果として、次の点が判明した。

- ユーザインタフェースの点では、簡単で使いやすく、処理の流れが把握しやすい。
- 流れが構造化されていない処理に対しては、対応しきれない。

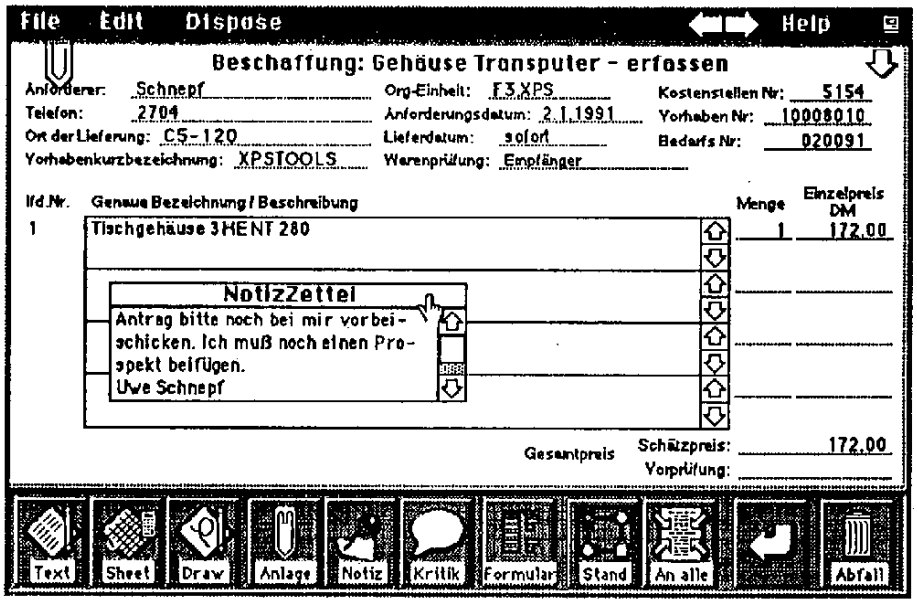


図3.4-3　DOMINO システムのインタフェース例（購入伝票）

・　オフィスの事務処理環境は、各エンドユーザに合わせて変更することは難しい。

・　組織の役割あるいはグループの役割といったものは、完全には表現できない。

今後は、DOMINOシステムでの経験を踏まえて、さらに各メンバの要求に合わせて柔軟に対応できるシステムを開発する予定である。

3.5　メッセージ構造化システム

3.5.1　Information Lens

原文タイトル：　Semistructured Messages Are Surprisingly Useful for Computer-Supported Coordination

翻訳タイトル：　コンピュータ支援協同作業にとって非常に有益な「半構造化されたメッセージ」

著書：
Thomas W. Malone　(MIT, Massachusetts, USA)
Kenneth R.Grant　(MIT, Massachusetts, USA)
Kum-Yew Lai　(MIT, Massachusetts, USA)
Ramana Rao　(MIT, Massachusetts, USA)
David Rosenblitt　(MIT, Massachusetts, USA)

出典：　ACM Transactions on Office Information Systems,Vol.5,No.2,April 1987, pp.115～131

本論文では、組織内部で情報の共有を図るためにInformation Lensというシステムを開発・導入したことで得られた成果が紹介されている。まず初めに、グループのコミュニケーションや協同作業用のシステムを開発する際に、半構造化されたメッセージのテンプレートを応用するというアイデアが紹介され、このアイデアを取り入れると、メッセージを作成したり処理したりする一般的な機能を設計することが簡単になることが述べられている。つぎに、この機能は作業割付（task tracking）やカレンダ管理（calendar management）、コンピュータ会議などのためのさまざまなアプリケーションに応用できることが示されている。最後に、あるコミュニティがその内部で使われるメッセージのタイプをどのように決めていくかという問題について若干述べられている。

(1) 半構造化されたメッセージとは

ここでは「半構造化されたメッセージ」を、次のように定義する。すなわち、さまざまなタイプに識別できるメッセージであって、各々のタイプはいくつかのフィールドを含んでいる（その内のいくつかのフィールドには、構造化されていないテキストやその他の情報が入っているものもある）。たとえば、「セミナーのお知らせ」の場合には、「日時」、「場所」、「講演者」、「演題」といったフィールドが通常のヘッダフィールドや講演内容の記述の他にあるテンプレートが当てはめられる。

このアイデアが重要であるというのは、次のような理由による。

① 半構造化されたメッセージは、他のものよりもコンピュータが自動的に処理できる情報の範囲が広い。

② 半構造化されたメッセージは、厳密な構造化に比べて自由に情報の交換ができる。

③ 情報処理を行う人々は、今までもメッセージをある程度構造化されたタイプのものとして扱ってきている。

④ メッセージが自動的に処理されない場合でも、準構造的なテンプレートはメッセージを作る人にとって役に立つ。

⑤ 半構造化されたメッセージは、段々と拡張していくシステムの設計をやりやすくする。

このような一般的な理由に加えて、⑥一般的な性質を持ついくつかのタイプを継承して固有のタイプを作れるようにメッセージのタイプを継承できる機能を持たせることや、また⑦メッセージや、メッセージを処理するルール、そして新しいメッセージテンプレートをディスプレイから編集できるエディタ群を用意することにより、半構造化されたメッセージをより簡単に使えるようになる。

(2) Information Lens システム

Information Lensは、半構造化されたメッセージの特長を活かした知的な組織内情報共有システムで

ある。このシステムは、人々がすでに受け取ったメッセージについて取捨選択したり、並べ変えたり、優先順位を付けたりする際に手助けとなる。また、受け取っていないメッセージの中で役に立つものがあるかどうかを知る際の助けにもなる。

さらに本システムは、電子メールシステムに応用するときに、次の４つの重要な機能を提供することができる。

① メールの送信者は、メッセージを作る際に構造化されたテンプレートを使うことにより、そのメッセージにどんな情報を入れたらよいかがわかり、さらに個々の情報について選択していくだけでほとんどメッセージができてしまう。

② メールの受信者は、送信者が構造化して送ってきたメッセージを、自動的に取捨選択・分類して該当するフォルダに入れるようにルールを指定することができる。

③ 送信者は、通常の宛先（個人あるいは受信者リスト）の他に、ある特別なメールボックス（現行では「Anyone」と名付けられている）に送ることができ、この場合このメッセージに関心を持つと思われる人に宛てて自動的に送られる。

④ 受信者は、Anyone宛であって自分のところには送られてこないメッセージを受け取るようにルールを指定することができる。

宛先の中にAnyoneを含むメッセージは、Anyone以外の宛先にはメールサーバが直接送信し、それと同時にそのワークステーション上の自動メールソータに送られる。自動メールソータは、ルールを指定している受信者にそのメッセージを送る。こうしてメッセージは、関心を持つと思われる人に対してのみ送られる。もし機密保持の問題があるときには、「Anyone-in-受信者リスト」という宛先にしておけば、指定した受信者リストの中でルールを指定した人に対してのみ送られる。

本システムの全てのアプリケーションは、Loopsという、LISPのオブジェクト指向版を使用して、Xerox 1100シリーズのワークステーション用のInterlisp-Dのプログラミング環境で作成された。これらのアプリケーションは、既に人々が使っている機能も一緒に使えるようにすることを考慮して、既存のメールシステム（Lafite）の上に構築されている。ユーザは今まで通りの使い方で、メールをやり取りしたり、受信者リストを利用したり、受信したメッセージを手作業で分類してフォルダに入れたりすることもできる。このようにして本システムは、本研究グループの５人のメンバにより１年以上の間継続して使用された。

(3) 半構造化されたメッセージで可能になること

① メッセージを作成する際の自動的な補助機能

さまざまなタイプの構造的なメッセージが用意されているため、メッセージを作成する際に、いろ

いろと自動的な補助機能を提供することができる。たとえば、Information Lens システムでは、メッセージのタイプごとにさまざまなテンプレートを用意し、そのテンプレートはそのフィールド特有の情報を扱うことができる。このようなテンプレートを利用してユーザがメッセージを作成していくグラフィックインタフェースの例を図3.5-1に示す。ユーザは、マウスで希望するメッセージのフィールドを選択した後、そのフィールドの「既定値」か、そのフィールドの「意味の説明」、あるいはそのフィールドに記述されると考えられる「内容の選択リスト」のいずれかをマウスで選択できる。内容の選択リストの中から希望する項目を選ぶと、その内容が自動的にメッセージのテキストに挿入される。

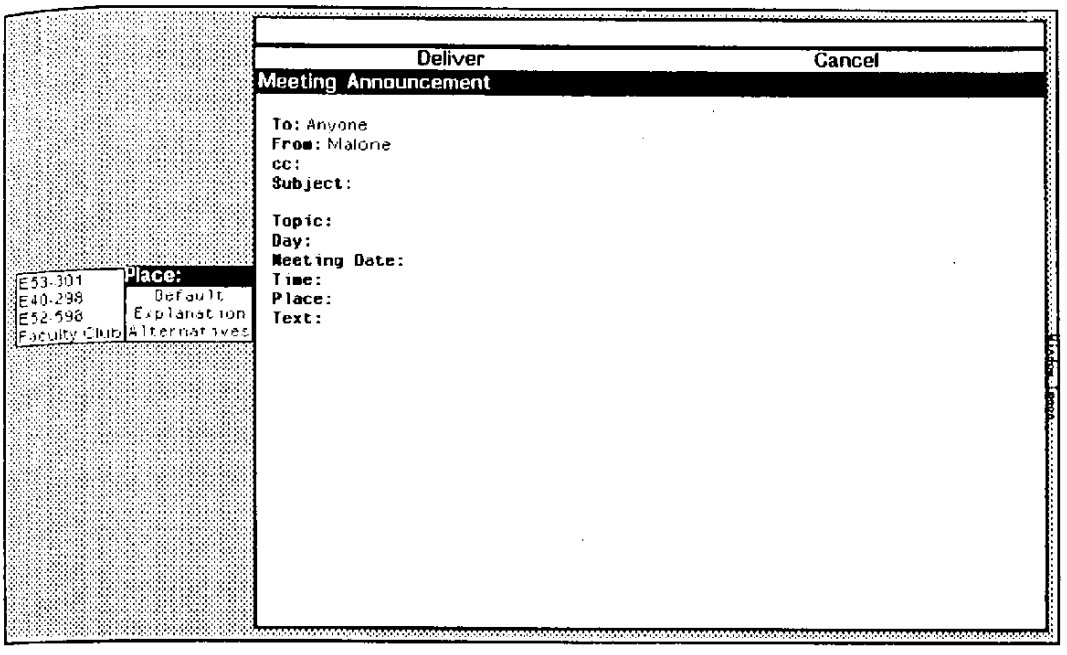


図3.5-1　ディスプレイ向きエディタを利用してメッセージを作成しているところ。指定したフィールドのテンプレートがポップアップメニューの形で表示されている。

メッセージのタイプごとに、既定値や選択リストに関する知識が豊富になってくるにつれて、システムはもっと簡単にメッセージの作成ができるような機能を提供できる。たとえば、図3.5-2に示したように、毎週の定例ミーティングのお知らせの場合、ほとんどの項目が既に既定値で埋められている。ここで見られるように、そのフィールドの既定値や意味の説明、あるいは内容の選択リストは、個々のユーザが、普段の自分のコミュニケーションのやり方や好みに合わせて設定することができる。

現行のシステムでは、全てのタイプのメッセージに Topic というフィールドが存在する。このフィールドは、そのメッセージのトピックを表わすキーワードを記述するためにある。このフィールドの選択リストは、他のフィールドのような単純なリストではなく、ネットワーク構造化されたさまざまなトピックの中から特定のトピックを得られるようになっている。

このようなメッセージ作成補助機能を望まないユーザは、message という名の一般的なメッセージ

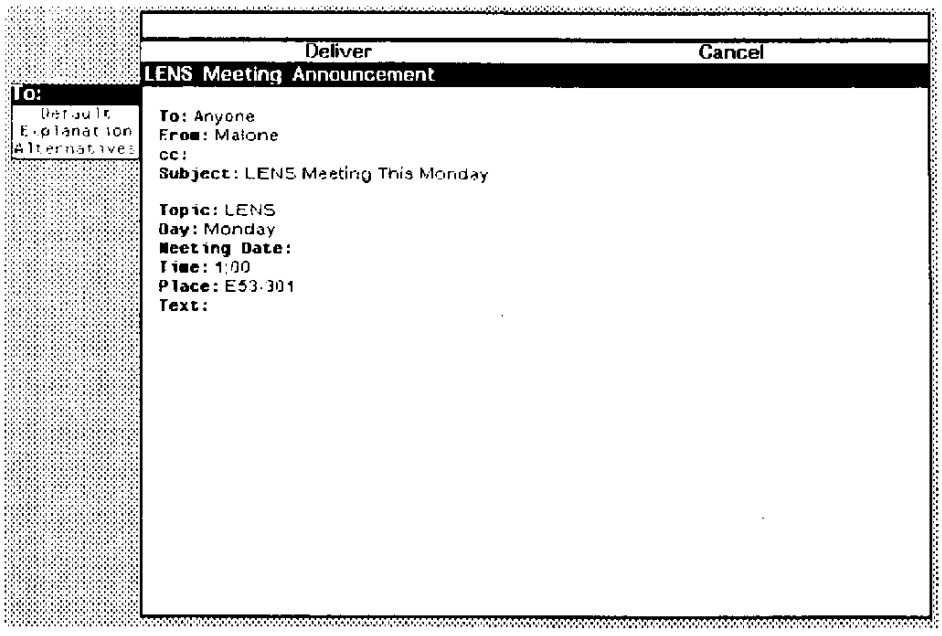


図3.5-2　さまざまなフィールドが既定値で埋められたテンプレートの例

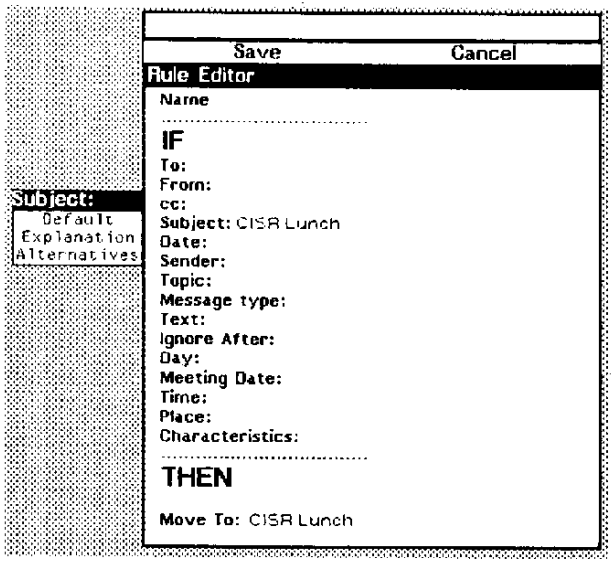


図3.5-3　メッセージを作成する際と同じ種類のエディタとテンプレートを利用して作成されるメッセージ処理用ルール

タイプを選択しておけば、既存のテキストエディタを使って、メッセージの各フィールドを直接編集することもできる。たとえば、メッセージの中に希望するフィールドを自由に付け加えることもできる。

② メッセージを自動処理させるルール

メッセージを構造化することで、届いたメッセージを処理するルールを作成する処理も簡単になる。たとえば図3.5-3は、テキストエディタを使ってInformation Lensシステムのルールを作成しているところを示したものである。このエディタは、メッセージを作成する時に使用したものと同じ種類のメッセージタイプを利用してルールのテンプレートを作っている。そして、メッセージ作成時と同じような操作方法で、各タイプごとに、既定値、意味の表示、選択リストを選択することができる。

本システムには、現在ローカルルールとセントラルルールという2種類のルールがある。ローカルルールは、メッセージがメールサーバからユーザの使用しているワークステーションに送られてきた時点で適用される。よく使われるルールとしては、特定のフォルダへメッセージを移動（move）する指示（図3.5-4a）やメッセージを消去（delete）する指示（図3.5-4b）、他の人へメッセージを再送（resending）する指示（図3.5-4c）などがある。メッセージの「移動」と「消去」という指示は、どちらも消去（delete）することであるが、物理的に抹消（expunge）してしまうことではない。だから、同じメッセージをいくつもコピーして異なるフォルダへ入れるという複数のルールも書けるのである。

「再送（resending）」は「転送(forwarding)」と似ているが、後者が元のメッセージを全てコピーして新しいメッセージとするのに対し、前者は宛先（To）とカーボンコピー(cc)のフィールド以外をコピーして新しいメッセージを作るところが異なっている。このとき宛先（To）フィールドには、ユーザがルールに記述しておいた宛先が適用され、またルールに記述した送信者（Sender）のフィールドが付け加えられる。

フィールド間あるいはフィルド内の条件を記述する際には、ブール関数を使うこともできる。さらに、メッセージの「特徴（characteristics）」を定義するルールを使用することによって、複雑な組み合わせのルールを記述することもできる（図3.5-4d）。このルールにさらに発展性を持たせるために、「LISP code」という名のルールも用意している。このルールが指定されていると、

(a) **IF** Message type: Action request
Action deadline: Today, tomorrow
THEN *Move to*: Urgent

(b) **IF** Message type: Meeting announcement
Day: Not Tuesday
THEN *Delete*

(c) **IF** Message type: Meeting proposal
Sender: Not Axsom
THEN *Resend*: Axsom

(d) **IF** From: Silk, Siegel
THEN *Set Characteristic*: VIP
IF Message type: Action request
Characteristics: VIP
THEN *Move to*: Urgent

(e) **IF** Message type: Request for information
Subject: AI, LISP
THEN *Show*

(f) **IF** Message type: NYT Article
Subject: Computer
THEN *Move to*: Computers
IF Message type: NYT Article
Subject: Movies
THEN *Move to*: Movies
IF Message type: NYT Article
Article date: < Today
Characteristics: Not MOVED
THEN *Delete*
IF Message type: NYT Article
Characteristics: Not MOVED and not DELETED
THEN *Move to*: NYT Articles

図3.5-4 ルールの例

ルールが発火したときにそこに記述されたLISPの関数名が呼ばれるというものである。

一方、セントラルルールは、個々のユーザがAnyone宛のメッセージの中から見たいと思うものを指定するために記述される。セントラルルールとしては、「show」と「set characteristic」という2種類の指定が可能である。「show」を指定すると、（宛先としてAnyoneが含まれている）メッセージの宛先の中にそのユーザが含まれていない時に、そのメッセージが送られる。こうして送られてきたメッセージは、その後他の通常のメッセージと同じように、ローカルルールによって処理される。

ルールに関しては、本システムを使用してみた結果に基づき、次の点が改良されている。まず、メッセージタイプに対するルールは、ルールセットエディタに現われた順序に従って適用される。また、特定のメッセージタイプに関するルールは、一般的なメッセージタイプから継承されたルールよりも先に適用される。さらに、何らかのアクションを起こすルール（メッセージの移動や削除など）については、「特徴（characterisstic）」として定義しておく（たとえば、MOVEDやDELETEDという名で定義する）。こうすれば、たとえば「そのメッセージがまだ移動も削除もされていなければ」といった条件のルールを設定することができる（図3.5-4fを参照のこと）。また、ユーザがルールを理解したり修正する際の手助けとなるように、指定したメッセージに対してどのルールが発火したかを示す履歴を見ることができる機能を用意した。

③　メッセージを返信する際の知的な補助

半構造化されたメッセージでそのタイプが認識できると、ユーザがそのメッセージを見た後でしたいと思われる選択肢を、システムが知的に判断して提供することができる。

(a)　返信するタイプの提示

メッセージに対して返事を出したり転送したりする際には新しいメッセージが作られる。そこで、本システムでは、受け取ったメッセージのタイプごとに、作成する新しいメッセージのタイプについて選択肢を提示する。たとえば、ユーザが「バグフィックスの要望」というメッセージに対して「返信」の選択をしたときに、「バグフィックスの約束」、「詳しい情報の要望」、「その他」という3つの選択肢を持つポップアップメニューが現われる。ここで「約束」を選ぶと、このメッセージに答える新しい「約束」タイプのメッセージが作られる。

(b)　対応するアクションの提示

さらに重要な機能として、本システムでは、受け取ったメッセージを見てユーザが対応するであろうと思われるアクションの選択肢を提示することができる。たとえば、ユーザが「ミーティングのお知らせ」というメッセージを読んだときには、通常の「返信」や「転送」等の選択肢のほかに、「カレンダに登録」という選択肢が自動的に付け加えられる。ここでユーザが「カレンダに登録」を選ぶと、メッセージの中の構造化されている項目（日付、時刻、議題など）がそのユーザのオンラインカレンダに編集されて付け加えられる。また、もう一つの例として、ユーザが「ソフトウェアリリース」

というメッセージを読んだときには、「ファイルをロード」という選択肢が自動的に付け加えられる。ユーザがこの選択肢を選ぶと、メッセージの中に記述されたファイルがそのユーザのシステムに自動的にロードされる。

(4) メッセージタイプの特徴の継承

メッセージテンプレートを作成したり利用したりする際に、テンプレートがいくつかの子タイプのテンプレートのネットワーク構造として提供されれば、もっと使いやすくなる。テンプレートの子タイプは、親テンプレートを継承することにより、その親テンプレートのフィールド名や特性を自動的に継承する。そして、各子タイプは、そのタイプ独自のフィールド名や特性を、継承した親テンプレートのフィールド名や特性に追加することができる。たとえば、図3.5-5はプロトタイプシステムでのメッセージタイプのネットワークの例を示している。図中の seminar notice というテンプレートは、親テンプレートである meeting announcement には無い speaker というフィールド名を追加している。また、LENS meeting announcement（図3.5-2）というテンプレートには、親テンプレートには無いさまざまな既定値が付け加えられている。継承してネットワーク化することにより、既にあるテンプレートと似たタイプのテンプレートを作成する際に、重複する情報を記述する手間を省くことができる。また、テンプレートを体系化することになるので、メッセージの送信者は適切なテンプレートを容易に選ぶことができるようになる。

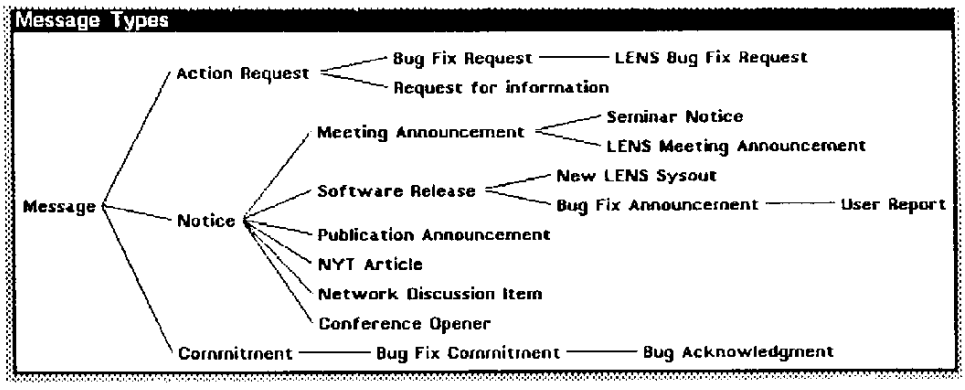


図3.5-5 ネットワークを構成しているメッセージテンプレート。トップレベル（図の左側）へ行くとより一般的なタイプとなり、ボトムレベル（図の右側）へ行くとより特殊なタイプとなる。

(5) アプリケーションの例

半構造化されたメッセージのアイデアが一般的であることを示すために、ここではアプリケーションの例をいくつか挙げてみる。ただし、ここでもっとも重要な点は、協調作業を支援するさまざまな

アプリケーションシステムが、(i)ユーザの立場に立ってスムーズに組み合わせられること、および(ii)構造化されたメッセージを支援する基本的な機能を利用することによってインプリメントが容易になること、の2点である。

① コンピュータ会議

コンピュータ会議システムが持つさまざまな機能は、「Information Lensシステム」に容易に組み込むことができる。まず、(a)会議名と（オプションとして）親会議の名というフィールドを持つ「conference opener」という新しいメッセージタイプを加える。また、(b)このメッセージタイプの返信用の選択肢として「join」という項目を加える。この項目が選ばれると、自動的に(i)この会議名を持つ新しいフォルダが作られ、(ii)トピックフィールドにこの会議名が記述された「Anyone」宛のメッセージを取り込むようにルールが記述され、(iii)このトピックのメッセージを新しく作成したフォルダへ格納するようにルールが記述される。

このようにして本システムは、コンピュータ会議システムが、予め定義されたさまざまなトピック（あるいはサブトピック）を用意することによってコミュニケーションの構造化を図っているような機能を実現できるのである。

② カレンダ管理

本システムは、「ミーティングのお知らせ」というメッセージに対してそのユーザのオンラインカレンダに自動的にスケジュールを登録するという選択肢を提供して、カレンダ管理の簡単なやり方を実現している。さらに、本システムでは、半構造化されたミーティングスケジュール用のもう少し複雑なプロトコルを実現した。このプロトコルでは、meeting proposals（ミーティングの提案）やmeeting acceptances（ミーティングの承諾）などの新しいメッセージタイプを利用している。「提案」や「承諾」のタイプは、meeting announcement（ミーティングのお知らせ）タイプのフィールドを全て持っている。ユーザは、ミーティングの提案が合意するまで、提案（あるいは再提案）のやり取りを続ける。

本システムでは、次のようにしてこの処理の自動化を図っている。まず、あるタイプ（たとえば、meeting proposals など）のメッセージに対して返信するため、全て他の人（たとえば、秘書）に送りたいと考える人がいるであろう。本システムは、このような処理を自動化する。また、本システムは、これらのメッセージに対して返信メッセージを作成する際に支援する。たとえば、meeting proposal に対してmeeting acceptance を返信しようとすると、全ての内容（時刻、場所、議題など）が自動的にmeeting acceptance にコピーされる。meeting acceptance を受け取った際には、そのメッセージに対するアクションとしてメッセージの「confirm（確認）」という選択肢が用意されている。この項目を選択すると、ユーザのカレンダにそのミーティングが登録され、開催時刻を確認するために、他の出席者に meeting announcement メッセージが送られる。

③ プロジェクト管理と作業割付

構造化されたメッセージを利用するシステムは、誰にどんな作業を割り付けたかを把握することを支援できるので、プロジェクト管理などの調整作業の手助けとなる。本システムでは、アクションの要望とそれに対する約束のメッセージを全て特別のフォルダに格納させるというルールを用意することによって、この処理を簡単に実現している。たとえば、あるユーザは、受け取ったアクション要望のメッセージを、自分の関係するプロジェクトごとに分類したフォルダに格納しているという場合もあるし、他のユーザは、自分の送ったアクション要望メッセージを、そのメッセージの送り先ごとに分類したフォルダに格納しているという場合もある。

ここではもう少し複雑な応用例として、ソフトウェアメンテナンス処理用の簡単な作業割付システムを開発した。この例では、まず、ソフトウェアシステムのユーザが作業割付担当者に対して problem reports（バグレポート）メッセージを送る。作業割付担当者は、ユーザに対して acknowledgement（了解）メッセージを返すとともに、開発担当者に対して problem reports メッセージを転送する。開発担当者はバグをフィックスすると、fix report（バグフィックスレポート）メッセージを作業割付担当者に送り、受け取った作業割付担当者は、元のユーザに対して、バグがフィックスしたことを示す user report メッセージを送る。

この応用例では、本システムは３つのメッセージタイプを定義し、新しい対応オプションをいくつか設定して実現している。作業割付担当者となったユーザには、problem reports を読んだ際に取ると思われる２種類のオプションを用意した。１つは、そのメッセージに対してまだ acknowledgement を返していない場合に選択できる acknowledge オプションである。このオプションを選択すると、problem reports の内容を基にして該当するフィールドの内容が自動的に記述された標準的な acknowledgement が作成される。acknowledgement を送ると、problem reports の中の characteristics（特徴）フィールドに acknowledged（了解済み）というフラグが立てられる。acknowledged フラグが立てられた problem reports を読んだ際には、assign（作業割付）が対応オプションとして追加される。このオプションを選択すると、作業割付担当者が problem reports メッセージを転送する際に宛先を選択できるように、開発担当者の一覧が表示される。

(6)　個々のグループに特化したメッセージタイプの定義付け

構造化されたメッセージタイプをどのように定義するかという点は重要である。各々のグループやアプリケーションで、自分達に関係のある特別の情報を扱える詳細な構造を作ることができる。たとえば、商品企画チームでは、商品という面から見たさまざまなメッセージタイプ（市場規模の評価、レスポンスタイムの評価、同業者のパワーなど）のネットワーク構造を作ることができる。

このアプローチの面白い点は、定義したメッセージタイプの数が増えるにつれて、これらのメッセージタイプを使う人の数も増えるという点である。たとえば、最初に情報共有システムを使い始めたユ

ーザは、全てのメッセージに共通のフィールド（To、From、Subject、Dateなど）だけを利用してルールを作成する。ユーザのグループが一般的なメッセージタイプのセットを使い始めると、さらに特種化したメッセージタイプを扱えるもっと複雑なルールを作成して便利にしようとする。たとえば、ワークグループの個々人は、手始めに構造化されていない電子メールを使って、さまざまなミーティングをスケジューリングしようとする。このシステムを利用することが一般的になり、重要性を持ってくると、メッセージを適切に分類したり優先順位を付けることができるように、特別のメッセージタイプ（meeting announcements や meeting proposals など）が定義されるようになる。そして最終的には、ミーティングのスケジュールを自動的に処理できる機能が付け加えられる。組織の原理という立場から見ると、「組織内部コード」というものが組織の生産するものの中でとりわけ重要であると言える。結果的には、本システムは、組織におけるこの一まとまりの言葉を定義、再定義できる道具であると言える。

3.5.2 Object Lens

原文タイトル： Object Lens: A "Spreadsheet" for Cooperative Work

翻訳タイトル： オブジェクト・レンズ：共同作業のための「スプレッドシート」

著書： Kum-Yew Lai (MIT, Massachusetts, USA)
Thomas W. Malone (MIT, Massachusetts, USA)

出典： CSCW'88 Proceedings, September 1988, pp.115-124

本論文では、Information Lens システムのコンセプトを拡張して開発されたObject Lens と呼ばれるシステムの初期のプロトタイプについて紹介されている。Object Lens は、初心者ユーザでも広範囲の共同作業活動のための（限定されてはいるが）非常にフレキシブルで有用なコンピュータ処理機能および通信機能にアクセスすることを可能にしている。本論文では、(1) Object Lens の設計において利用したキーとなるアイデアが説明され、(2) Object Lens の機能においてそれらのアイデアがどのように実現されているかが示され、(3) このフレームワークの中で開発したいくつかの単純なアプリケーションの例（タスク・トラッキング、知的メッセージ・ルーティング、データベース検索）が紹介されている。

(1) Object Lens のキーとなるアイデア

Object Lens のもっとも重要な特性は、それが、セミフォーマルなシステムであることである。ここで、セミフォーマルなシステムとは、以下の 3 つの性格を持つコンピュータ・システムであると定義する。①ある種の情報をフォーマルな形で指定された方法によって表現し、かつ自動的に処理する。

②①の情報、あるいは別の種類の情報を、フォーマルに指定されたのではない形で表現し、人間が処理することを容易にする。③コンピュータによるフォーマルな処理と、人間によるインフォーマルな処理の間の境界を容易に変更できるようにする。これらの特性は共同作業アプリケーションにおいて、特に重要である。共同作業においては、人々の行動のいくつかのパターンについては、十分理解されているけれども、それ以外に潜在的に重要ではあるが、特定することの困難な知識が大量に存在しているからである。

そのような柔軟かつセミフォーマルなシステムを開発するためには、システムの中に埋め込まれる知識は、可視的 (visible) かつ可変的 (changeable) な形でユーザに提示されなければならない。つまり、ユーザが、情報やシステムに含まれた処理ルールを見たり変更したりすることが簡単にできなければならない。Object Lens においては、知識をユーザに提示する方法に関して 2 つの中心的なアイデアがある。

(i)「受動的 (passive)」な情報は、半構造化オブジェクトとして提示される。

(ii)「能動的 (active)」な情報処理ルールは、半自律的エージェントとして提示される。

以下、これら 2 つの構成要素によって、如何に可視的かつ可変的な形で知識がユーザに提示されるかについて説明する。

① 半構造化オブジェクト

Object Lens システムのユーザは、メッセージ、人々、会合、仕事、製造された部品、ソフトウエアのバグなど、物理的あるいは概念的に親しみのある物を示すオブジェクトを作成、検索、表示することができる。この「オブジェクト指向データベース」においては、それぞれのオブジェクトは、フィールドとフィールド値の集合を含む。例えば、PERSON オブジェクトは、Name（名前）、Phone（電話番号）、Address（住所）、および Job Title（職位）などのフィールドを持ち、TASK オブジェクトは、Requestor（要求者）、Performer（実行者）、Description（内容記述）、Deadline（期日）などのフィールドを持つであろう。Information Lens システムにおけるメッセージと同様に、これらのオブジェクトは、ユーザがどのフィールドにも任意に情報を記入でき、また書き込まないことも任意であり、さらにフィールドの中の情報は、特定のタイプに当てはめる必要がない（例えば、「I don't know」のように、自由形式のテキストであって良い）という意味で半構造化オブジェクトである。

(a) テンプレート・ベースのユーザ・インタフェース

ユーザはこれらのオブジェクトをテンプレート・ベースのユーザ・インタフェースという、非常に自然な形式で見たり変更したりすることができる。このインタフェースにはいくつかの利点がある。例えば、(i) ユーザが既に馴れ親しんだフォームに似ている。(ii) オブジェクトに含まれるフィールドについての情報、および異なったフィールドの間の区別や比較等の情報をユーザに提示するのに便利で

ある。(iii) 多くの異なった種類のフィールドを一括して整合性のある使い方が可能である。

(b) オブジェクト間の関係

ユーザは、オブジェクト間にリンクを挿入したり削除することによって容易にオブジェクト間の関係を見たり変更することができる。例えば、TASK オブジェクトの Requestor フィールドと Performer フィールドには、それぞれその仕事を要求した人と、それを実行する人を示す複数の PERSON オブジェクトへのリンクが含まれるかも知れない。その場合、例えば、ユーザが TASK オブジェクトを見る時、その仕事に関係した人間についての情報（例えば電話番号など）を得ることは、もっと簡単になるであろう。

(c) 変更可能な表示フォーマット

ユーザはいくつかのオプションを用いてオブジェクトの表示方法を変更することができる。(i) ユーザは、例えば、表示するフィールドを選択したり、特定のフィールドの名称を変更したりすることによって、個々のオブジェクトの表示フォーマットを簡単に変えることができる。(ii) オブジェクトの集合から特定のフィールドを選択し、表形式で簡単に表示することができる。(iii) ユーザは、オブジェクトの集合の間の関係を選択し、オブジェクト間の関係が 1 つの図形的なネットワーク構造として示される、ツリー形式で表示させることができる。

(d) オブジェクト間の継承階層関係

タイプ定義の作成と修正は、オブジェクトのタイプを継承階層関係の中でアレンジすることによって簡略化されている。オブジェクトの新しいタイプは、既存のオブジェクト・タイプを特定することによって定義され、特にオーバーライドされた部分を除いて、既存オブジェクトのすべての性質が自動的に継承される。新しいオブジェクトについての情報の殆どは、毎回再入力されるのではなく、既存のタイプから継承されるので、新しいオブジェクト・タイプの作成は簡単になる。また、後になってオブジェクト・タイプの定義を変更する場合にも、その修正は自動的に下位のオブジェクト・タイプに継承される。

② 半自律的エージェント

Object Lens システムのユーザは、他のユーザのために自動的に情報を処理するルールベースの「エージェント」を作成することが出来る。これらのエージェントは、システムが自動的に実行するタスクを分離するための自然な方法を提供している。エージェントは、新しいメールの到着、指定したフォルダ内への新しいオブジェクトの出現、事前に指定した時刻の到来、他のエージェントからの陽表的な起動、ユーザによる陽表的な選択などのイベントによって起動される。起動されたとき、エージェントは、指定されたオブジェクト集合に対して、ある一纏まりのルールを適用する。あるオブジェクトがルールに指定された判定基準を満たすとき、そのルールは何らかの事前に指定された処理を実

行する。これらの処理は、検索、分類、メールの発送、オブジェクトの削除のような一般的な処理である場合もあるし、ファイルをロードしたり、カレンダにイベントを追加するなど、特定のオブジェクトに対応した処理である場合もある。

Object Lens のエージェントが「自律的」であるのは、それらが一度作成されたならば、人間のユーザが特に注意を払わなくても、処理を実行することができるからである。ただし、それらが「半自律的」であるに過ぎないというのは、(i) エージェントは常に人間のユーザによって制御されていること（つまり、人間のユーザはそのすべてのルールを容易に見ることができるし、変更することができる）(ii) エージェントは、独自の処理を実行するよりも、人間のユーザにオブジェクトを提示し、処理を求めることがしばしばであること、などの理由によるのである。

エージェントもルールもそれ自体がオブジェクトであるから、ユーザは他のすべてのオブジェクトに対して利用されるのと同じテンプレート・ベースのユーザ・インタフェースを介してそれらを見たり修正したりすることができる。与えられたオブジェクトに対してあるルールが何時発火すべきであるかの判定基準を指定するためには、ユーザは、ルール適用対象オブジェクトの「記述 (description)」を作成する。記述とは、特定のタイプのオブジェクトのための部分的に記入済みのテンプレートである。また、記述には「埋め込まれた記述 (embedded description)」を含めることができる。これは、もとのオブジェクトがリンクされたオブジェクトによって満足されなければならない特性を指定するものである。例えば、Task の記述には、そのタスクを実行する Person についての「埋め込まれた記述」が含まれる場合があるであろう。

(2) システムの特徴

Object Lens システムは、Ethernet によって接続された Xerox 1100 シリーズ・ワークステーション上に、Interlisp-D によってインプリメントされている。このシステムは、Loops の提供するオブジェクト指向プログラミング環境を活用して開発された。本論文の執筆時点では、基本的なメール処理機能が開発グループの 3 人よって 3ヶ月間本格的に利用されている。

① インスタンスの編集

図3.5-6 に、Person インスタンスのテンプレートを示す。内蔵されたテキスト・エディタを利用して、ユーザは、任意のフィールドに値を挿入したり、変更することができる。さらに、ユーザがマウスであるフィールドの名称部分をクリックすると、そのフィールドの値の候補のリストがポップアップ・メニューで表示される。これらの候補は、他のオブジェクトへのリンクである場合もあるし、単なるテキスト・ストリングである場合もある。これらの候補の1つを選択すると、その値が自動的にそのフィールドに挿入される。例えば、図には、Kum-Yew Lai のスーパバイザを示す Person オブジェクトへのリンクが含まれている。候補リストに現れていないオブジェクトへのリンクを挿入するには、ユ

ーザは (i) テンプレートの中のリンクを挿入すべき位置にカーソルを位置付け、(ii) ウィンドウ上部のメニューから「Add Link」オプションを選択し、(iii) リンクを付けるべきオブジェクトをポイントする。リンクが挿入された後で、マウスでその上をクリックすると、それがポイントしているオブジェクトが画面に現れる。

図3.5-7 は、もう少し複雑なテンプレートを示している。これは、Bug Fix Request メッセージのためのテンプレートである。このテンプレートのフィールドの 1 つは、修正すべき Bug であり、そのフィールドの値は、Bug オブジェクトへのリンクである。この場合、単に Bug オブジェクトへのリンクを表示するのではなく、テンプレートには、Bug オブジェクトそれ自体への、埋め込まれたテンプレートが含まれている。この埋め込まれたテンプレートのフィールドは、テンプレートの他のフィールドとまったく同様に編集することができる。

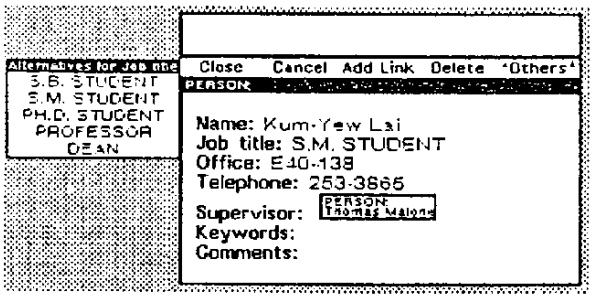


図3.5-6 オブジェクトは、簡単なテンプレート・エディタで編集することができる。

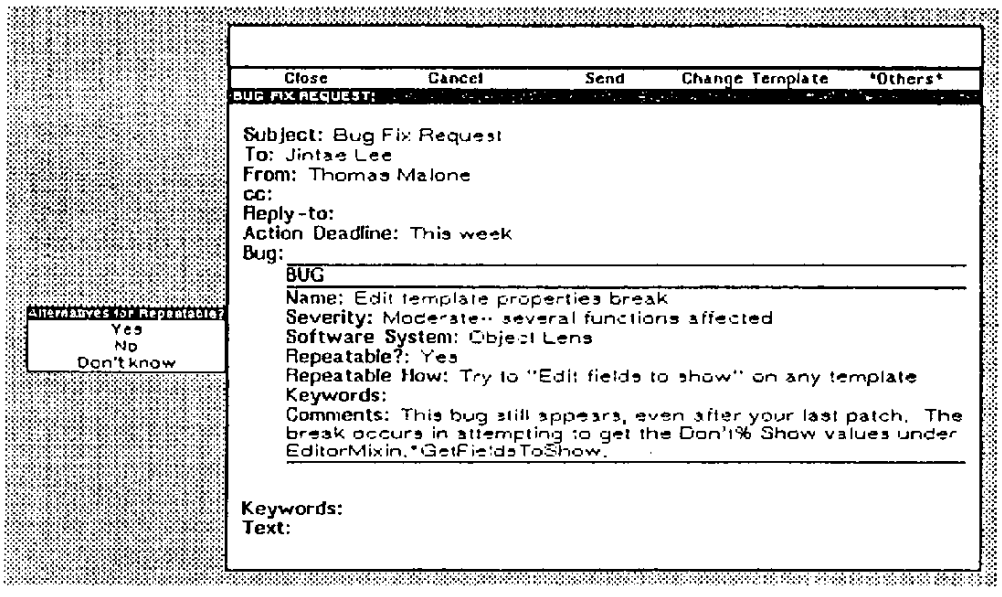


図3.5-7 埋め込まれたテンプレートによって、関係するオブジェクトを同時に見たり編集することができる。

② 新しいインスタンスの作成

ある既存のオブジェクト・タイプの新しいインスタンスを作成し表示するには、ユーザは、マウスでそのオブジェクト・タイプの定義（「Template」）をクリックする。図3.5-8 に現在われわれのシステムに含まれているテンプレートを示す。例えば、新しいメッセージを送るには、ユーザは、作成したいメッセージ・タイプに対応したテンプレートをクリックする。また、新しい Person オブジェクトを作成するには、ユーザは Personテンプレートをクリックする。そうすると、図3.5-6 や図3.5-7 に示されたような、オブジェクト・インスタンスが画面に表示され、ユーザは、それに値を記入することができる。

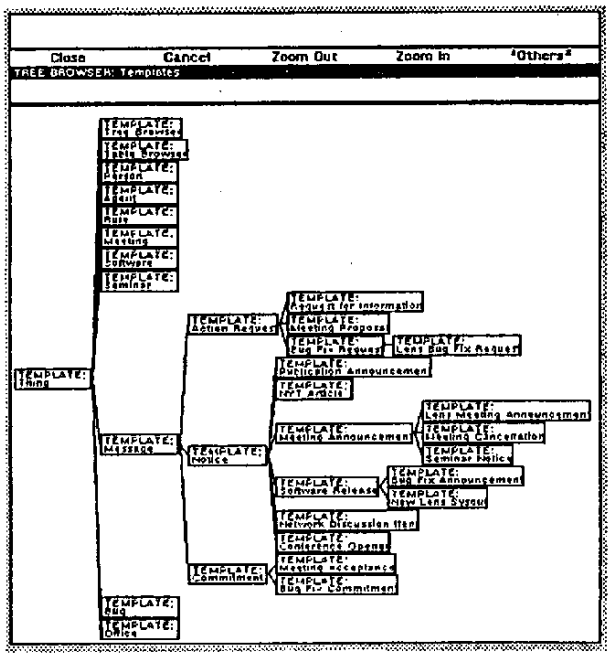


図3.5-8 オブジェクト・タイプは、テンプレートの集合によって定義することができる。

③ 新しいオブジェクト・タイプの作成

新しいオブジェクト・タイプを作成するには、ユーザは、親オブジェクト・タイプに対応するテンプレートをクリックする）。これによって、テンプレートへの操作の選択枝を示すメニューが現れる。これらの操作の 1 つは、「Create a subtemplate （サブテンプレート作成）」である。ユーザがこの操作を選択すると、親テンプレートのすべてのフィールドと性質を持った新しいテンプレートが作成される。その後、ユーザは、この新しいテンプレートにフィールドを追加したり、その表示フォーマットを変更したり、その他の性質を変更したりできる。

④ オブジェクト・タイプの表示フォーマットおよびその他の性質の変更

オブジェクト・タイプの表示フォーマットおよびその他の性質を変更するには、ユーザは、そのオブジェクト・タイプを定義するテンプレートを「編集」する。例えば、ユーザは、以下の変更を行う

ことができる。(i) オブジェクトのどのフィールドを実際に表示するか、(ii) 表示されるフィールドの名称、(iii) それぞれのフィールドに表示されるべき値の候補。また、ユーザは、それぞれのフィールドに対して、そのフィールドにおけるリンクがリンク・アイコンによって表示されるべきか（図3.5-6参照）、あるいは埋め込まれたテンプレートとして表示されるべきか（図3.5-7参照）を指定することができる。

⑤ エージェントとルールの作成

エージェントは、ルールの集合および、何時何処でそれらを適用するかの指定から構成される。ルールは、それを満足するオブジェクトの記述を含み、それらのオブジェクトに対して適用する処理を指定する。ルールには、IFフィールド（条件部）とTHENフィールド（処理部）がある（図3.5-9を参照）。ルールのこの両方の部分は、他のオブジェクトへのリンクを含んでおり、これらのリンクは、埋め込まれたテンプレートとして示されている。ルールのIF部は、「記述」であり、これは、フィールド値によって一連のインスタンスを記述する特殊なテンプレートである。ルールのTHEN部は、Actionオブジェクトである。

ルールのIF部を作成するには、ユーザは、(i) IFフィールドをクリックし、(ii) 提示されたメニューから「Descriptions」を選択し、(iii) 提示されたオブジェクト・タイプのツリー・フォーマットから目的のオブジェクト・タイプを選択する。これによって、適当なタイプの記述をルールの中に、埋め込まれたテンプレートとして、挿入することができるようになり、ユーザは、この記述の中のフィールドに記入することにより、オブジェクトがルールを満たすために特定のフィールドに現れるべき値を指定することができる。Information Lensの場合と同様に、文字列をand、or、not、あるいはカッコによって組み合わせることによって、フィールド値に対するもっと複雑な指定を作成することもできる。

ルールのTHEN部を指定するには、ユーザは単にTHENフィールドをクリックし、そのとき提示される処理の候補リストのメニューから選択するだけでよい。いくつかの場合には、ユーザは埋め

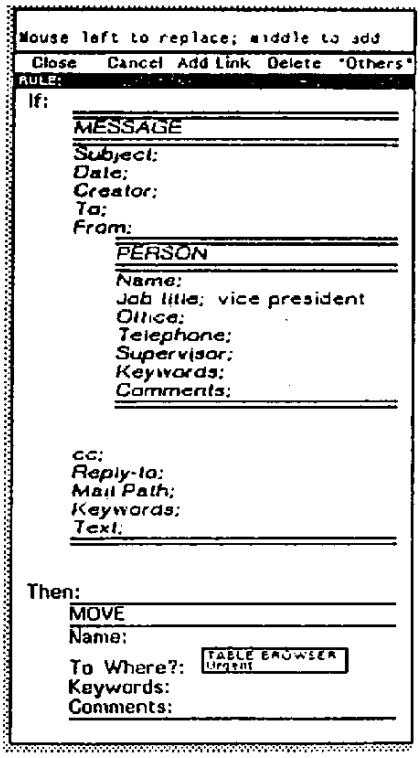


図3.5-9 複雑な問い合わせを作成するため、ルールには埋め込まれた記述を使用することが可能である。

込まれたテンプレートの中のいくつかのフィールドに記入して処理を指定する必要がある（例えばオブジェクトをどこへ移動するかなど）。

複雑な問い合わせを作成するため、ルールには、埋め込まれた記述を使用することが可能である。図3.5-9 に示したルールは、「副社長」を職位として含む人間からのメッセージすべてによって発火される。このルールを適用するにあたっては、システムは、メッセージの From フィールドの文字列が、知識ベースの中の、記述を満足するどれかの Person オブジェクトの Name と同一であるかどうかをチェックする。

(3) アプリケーションの例

① タスク・トラッキング

共同作業アプリケーションにおいてしばしば指摘される機能は、人々の仕事の進行度合いをトラッキングする能力である。例えば、このようなシステムは、「他の人々が自分に要求した仕事は何か」「それらの中で期日を過ぎているものはないか」「自分は他の人々にどのような仕事を要求したのか」等の質問に答えることができる。

Object Lens において、このような機能をサポートすることは、簡単である。例えば、このシステムには、既に、行動の要求や約束に対応したオブジェクト・タイプが含まれている。Information Lens においてさえも、これらのメッセージを自動的にソートして、誰がその仕事をすることになっているか、どのようなプロジェクトを含んでいるか、等によって分類してフォルダに入れることが可能であった。しかし、Information Lens においては、フォルダの内容の要約表示は、From、Date、Subject 等の標準的なメッセージ・ヘッダ・フィールドしか表示できなかった。タスクについてのより多くの情報を見ようとすれば、個別のメッセージを一つづつ表示しなければならなかった。Object Lens においては、ユーザの選択した任意のフィールドを表示することによって、フォルダの中のメッセージを簡単に要約することができる。

② 知的なメッセージ・ソート: 技術的変更通知

共同作業問題における興味ある例の 1 つは、製品設計仕様における変更の情報を組織の中の適切な人々に伝達する問題（しばしば技術的変更通知と呼ばれる）である。Information Lens においても、技術的変更通知を、「part affected（影響される部品）」、「type of change（変更のタイプ）」、「severity（重要度）」などのフィールドの内容によってソートすることが既に可能であった。Object Lens においては、追加の知識を利用してもっと知的なソートを行うことが可能になった。

③ データベース検索

個人としての仕事においても、共同作業においても、データベースからある種の条件を満たすオブジェクトを自動的に検索する機能が役立つ場合は多い。Object Lens においては、このための単純な方

法を提供している。ユーザは単に、1つのフォルダの中のオブジェクトをスキャンして、選択されたオブジェクトへのリンクを別のフォルダに納めるエージェントを作成するだけでよい。このエージェントのルールは、オブジェクトを選択するための判断基準を指定する。

Object Lens では、メッセージの受け手を決定するための特別の機能をインプリメントした。この機能によれば、「記述」をメッセージの To フィールドや、cc フィールドに埋め込むことができる。こうすれば、メッセージが送られたとき、これらの記述は知識ベースの中のすべての Person オブジェクトに自動的に適用され、検索された人々が To フィールドや、cc フィールドに挿入される。この機能によって、メッセージの送り手は、メッセージの送信時に、データベースの中の人々に関する現在の情報に基づいて、配布リストをダイナミックに計算し、作成することができる。

④ ハイパーテキスト

Object Lens においてハイパーテキスト・システムの機能を利用することは簡単である。例えば、本システムには Text というオブジェクト・タイプがある。これは Name と Text という2つのフィールドだけを表示する。Text オブジェクトの Text フィールドは、他のオブジェクトへのリンクを必要なだけ持つことができる。

ハイパーテキスト・システムの通常の利点に加えて、Object Lens では、ハイパーテキストを他のデータベース、メッセージング、および計算能力と統合しており、さらに有用なシステムとなっている。例えば、ハイパーテキスト・システムの別のノードへのリンクを挿入するには、ユーザは、まずリンクを張るべきノードを見つけなければならない。これまでのハイパーテキスト・システムにおいては、すべてのノードはリンクを追いかけることによって人手で発見しなければならなかった。Object Lens においては、上記のデータベース検索機能を利用して、ある種の基準を満たすオブジェクト（人々や、履歴情報など）を自動的に発見することができる。その後、それらのオブジェクトへのリンクをテキストに挿入することができる。

3.6 グループエディタ

3.6.1 Quit

原文タイトル： Collaborative Document Production Using Quilt

翻訳タイトル： Quilt を使用した共同文書作成

著書： Mary D.P. Leland (Bell Communications Research, Inc., NJ, USA)
Robert S. Fish (Bell Communications Research, Inc., NJ, USA)
Robert E. Kraut (Bell Communications Research, Inc., NJ, USA)

出典： CSCW'88 Proceedings, September 1988, pp.206-215

本論文では、文書の共同執筆を支援するコンピュータ・ベースのツールとして開発されたQuiltについて紹介されている。まず、共同執筆を支援するために必要な要素について述べられ、つぎにQuiltシステムの概要が示される。そして、共同作業におけるQuiltシステムの潜在的な利用法を示す1つのシナリオを提示して、Quiltシステムの機能と制限が解説されている。

(1) 共同執筆を支援するための要素

共同執筆や文書は非常に広い意味で捉えることができる。共著になる文書は非常に多数に上り、たとえ最終版の著者名が1人であっても、その文書が共同作業の成果であるケースは多い。この共同作業は、同僚にコメントをメモしてくれるように頼む非公式なプロセスから、より構造化された規則に基づく審議や承認のプロセスまで、実に様々な形を取り得る。後者のプロセスは、しばしば、提案書、論文、仕様書、その他の「公式」文書を発行したり、出版したりするときに必要とされるものである。このように文書の種類や協力の程度は様々であるが、そこでは必ず共同執筆者間のコミュニケーションが必要となる。共同執筆者は文書の内容を読み、それにコメントを付け加えることができなくてはならないし、また、文書を効率よく完成させるためには、調整、計画、情報の共有などが不可欠である。たとえば、文書の校閲はタイムリに行う必要があるし、この校閲がいつ終るかを編集者や執筆者たちに通知して、彼らが効率よく必要な訂正を行えるように取り計らう必要がある。また、これらのコミュニケーションは、文書の作成に協力者が果す社会的役割に応じて、適切な人々だけがアクセスできるように保護されなければならない。さらに、共同執筆者間に良好で生産的な作業関係を保つためにも、コミュニケーションは欠くべからざるものとなる。

(2) Quiltシステムの概要

Quiltは、ハイパー・テキストと直接操作インタフェースの概念を導入し、共同執筆の社会的局面に関する研究を結集して設計・開発された、共同文書作成作業の幾つかのコンポーネントを支援するシステムである。特筆すべきは、Quiltが、テキストまたは音声のどちらかを使って、文書に注釈を付ける構造化されたメカニズムを提供することである。デフォルトの注釈タイプとしては、訂正の提案、公のコメント、宛先指定メッセージまたはプライベート・メッセージなどが用意されている。共同執筆者たちは、必要に応じて、このデフォルトのセットを拡張することができる。この他、共同執筆者たちは、作成者や作成日時などを初めとして、様々な属性を注釈に定義することができる。これらの属性は、文書やその注釈のビューを選択するための述語を定義するときに使用できる。また、Quiltには、共同作業を円滑に調整するためのメカニズムとして、アクティビティのログ、通知、および、トリガの諸機能が用意されている。アクティビティのログには、利用者が文書に対して行った活動が、リテラルなマシン生成レベルと、より抽象的な人的生成レベルの両方で記録される。また、通知メカ

ニズムが電子メールと注釈プロセスを統合しているので、共同執筆者は、文書に直接アクセスしなくても、彼ら宛の注釈を見ることができる。さらに、共同執筆者は、トリガ・メカニズムを使って、所定の述語が適用されたときにQuiltに実行させる特定の動作を定義することができる。

Quiltのインプリメントは、比較的一般的で入手の容易なソフトウェアを使用することを前提としている。共同作業のアイテム、アイテム間のリンク、および、Quiltの共同執筆者および利用者に関する情報を格納するための基礎的なデータベース・システムとしてOrionが利用されている。また、Quiltのユーザ・インタフェースとしては、X WindowとXt Tool-kitが使用されている。

(3) Quiltの使用例

Quiltの機能を解説するために、数人の人々が、Quiltを使って、1つの文書を計画し、作成し、訂正し、吟味し、読むケースを想定して1つの潜在的なシナリオを説明して行くことにする。このシナリオには、2人の共同執筆者AnneとBillが登場する。この2人は、会議で数回顔を合わせたことがあり、Log Cabinという名称の文書作成プロジェクトを協力して実行することにした。彼らは同一の機関に属している訳ではないので、頻繁に会うことは難しい。しかし、彼らは共にQuiltを動作させるマシンにネットワークを介してアクセスすることができる。この共同作業を行っている間、彼らはプロジェクトの様々な局面について討論し、彼らが書こうとしている論文のアウトラインを決定し、草案段階の原稿をオンラインでやり取りしたり、互いの原稿にコメントを付けたり、原稿を訂正したりしながら、個別に著作作業を進め、最後に彼らの原稿を1つにまとめなければならない。さらに、論文を発表する前には、Billのボスである Cherylの承認を受けなければならない。Cherylのコメントに基づいて、彼らはさらに検討と訂正を重ね、論文をCherylの要求水準を満たすものに仕上げなければならない。

予備的な計画作業の大部分は、Anneの大学で、二日間に渡って、AnneとBillが一日中顔を付き合せて会議を行ったことで完了できた。しかし、内容や手順の詳細の多くがこの二日間の会議では決定し切れなかったので、彼らはQuiltを使ってこれらの問題点について討議することにした。この討議のテーマはすべてQuiltシステムの中の文書となり、AnneとBillは両方とも、これらを読み、訂正し、これらに注釈を付けることができる。この初期段階で、Quiltは、伝統的な会議システムに似たコンピュータ・ベースの同期会議システムとして使われた。しかし、Quiltには優れた注釈機能があるので、討議で出されたアイディアを、従来のように間接的に関連ポイントを参照して指示するのではなく、直接的に関連ポイントに付加することができる。この直接操作によって、会議の筋道を非常に容易にたどることができるので、AnneとBillはこの論文の共同執筆を継続できると判断した。

ここで、AnneとBillが展開した共同執筆作業のアウトラインをたどり、Quiltの下でどのような展開を経て最終文書ができ上がったのかを見てみることにする。まず、Anneは、図3.6-1に示す手順を使って、QuiltにLog Cabinプロジェクトの共同作業をセットアップした。Anneは、彼女自身とBillと

彼女のStudentsを初期メンバとしてリストした。StudentsはQuiltの利用者名であり、ここでは論文を読んで注釈を付けることができるAnneの生徒を意味する。次に、Anneは共同作業のスタイルを選択した。共同作業のスタイルによって、文書に付けることができる注釈のタイプや、共同執筆者が果す社会的役割が決定される。このスタイルは、次に示す様々なスタイルの中から選択することができる。

① 論文の共同執筆のためにQuiltに予め用意されているスタイルのセット
- 排他的スタイル　－　その章の著者だけがそれを修正できる。
- 共有スタイル　－　共著者はどの章でも修正できる。
- 編集者スタイル　－　任命された編集者だけがすべての章を修正する権利をもち、他の共著者は編集者に提案を行う権利しかもたない。

② Anneが既に定義している数種類のカスタム・スタイルのいずれか
- Anneが彼女の協力者のために既に定義しているデフォルトのスタイル。
- Anneが既に定義しているもう1つの（自由）スタイル　－　協力者全員がすべてのアイテムを修正、付加、または、削除できる。

③ Anneが、この共同作業のために、これから新たに定義する新しいスタイル。

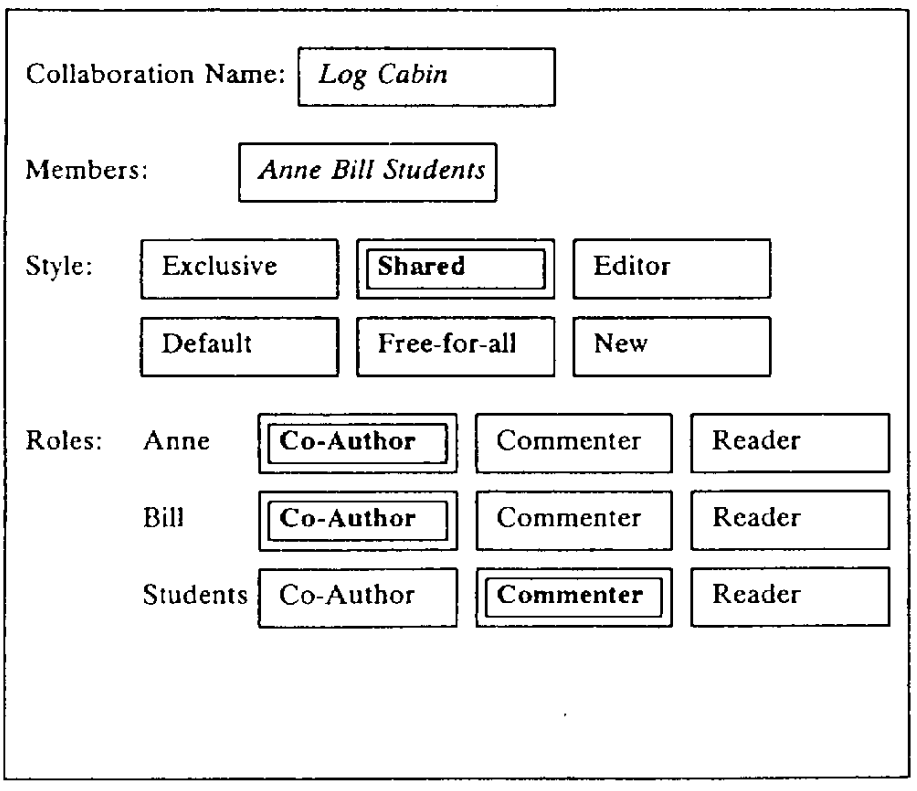


図3.6-1　新しい共同作業のセットアップ

ここで、Anneは、共同執筆者が互いに著作部分を交換できる「共有」スタイルを選択した。表3.6-1に、「共有」スタイルの表を示す。この表は、最上段に注釈のタイプ（基本文書、訂正の提案、公のコメント、宛先指定メッセージ、プライベート・コメント、ヒストリ）をもち、役割の階層を指定している。コメンテータは読み手に与えられるすべての権利と認可（並びに、認可テーブルで許諾されたその他の権利）をもっている。同様に、共著者は少なくともコメンテータの権利をすべてもっている。最後に、Anneは、メンバの一人一人に、選択したスタイルに応じて定義された社会的役割を1つまたは2つ以上割り当てた。この結果、AnneとBillはLog Cabin共同執筆プロジェクトの共著者となり、Anneのすべての Studentsはこのプロジェクトのコメンテータとなった。

表3.6-1　共同作業における役割と権利の定義

	Base Document	Suggested Revision	Public Comment	Directed Message	Private Comment	History
Create	Co-Author	Co-Author	Commenter	Commenter	Reader	Co-Author
Modify	Co-Author	Creator Co-Author	Creator	Creator	Creator	No One
Delete	Co-Author	Creator Co-Author	Creator	Recipient Creator	Creator	No One
Attach Revision	Commenter	Commenter	Commenter	Creator Recipient	Creator	No One
Attach Comment	Commenter	Commenter	Commenter	Creator Recipient	Creator	No One
Attach Message	Commenter	Commenter	Commenter	Creator Recipient	Creator	No One
Attach Private Comment	Reader	Reader	Reader	Creator Recipient	Creator	No One
Read	Reader	Co-Author	Commenter	Creator Recipient	Creator	Original Permissions Apply

AnneとBillは同一機関に所属していないので、互いの存在を刺激としてプロジェクトを進めることができない。そこで、彼らは、作業の調整と完成を助けるために、Quiltの通知およびトリガ・メカニズムを利用することにした。彼らは、パートナが文書に大きな変更を加えたときはいつでもメッセージを送信し、論文の提出期限が近付くにつれて間隔を狭めて頻繁に「注意を喚起」するようにQuiltに指示した。Anneは、彼女とBillが彼らの原稿のために開発したアウトラインをQuiltに入力した。これは大きな変更に当たるので、Quiltはその旨を電子メールでBillに通知した。一方、このメールを読んだBillは、Quiltのリンクを伝ってアウトラインにたどり着き、彼が妥当だと考える変更を加えたり、彼がAnneの意図を理解できない箇所を示すコメントを付け加えたりすることができる。AnneとBill

は、2つのメディアを使って、討議を続け、アウトラインを洗練した。メディアの1つはQuiltである。彼らはQuiltを使って、現在のバージョンのアウトラインを読み取り、アウトラインに変更を加え、互いに照会し合い、訂正の意図を説明した。もう1つのメディアは電話である。彼らは、電話を使って、現在のバージョンのアウトラインに変更を加えた。討議を行う度に、Anneはアウトラインを修正し、アウトラインに付属しているアクティビティ・ログに修正の理由を記録した。さらに、Anneは、彼女が討議中に考えついたアイディアを、彼女自身に宛てた音声コメントの形で、アウトラインの注釈として記録することができる。Quiltと電話を使ってコミュニケーションを行い、かなりの討議を重ねた結果、彼らのアウトラインは、まだルースな形ではあったが、再編成され、どちらがどの章を書くかについて合意に達することができた。彼らは訂正済みのアウトラインをQuiltに格納し、論文の中の各自が分担する章をLog Cabinの各アイテムとして作成した。

Quiltの利用者に与えられている認可のレベルに応じて、利用者は、(i) アイテムを読むことができる、(ii) アイテムを読み、注釈を付けることができる、または、(iii) アイテムを読み、注釈を付け、修正することができる。その他の動作（アイテムを標準テキストに変換する、アイテムを削除する、アイテムの名称を付け変える、アイテムをコピーするなど）も、利用者が得ている認可のレベルに応じて、実行することができる。

実際に論文の作成を進めている間、AnneとBillは、各自が分担する章に修正を重ね、文章の洗練に努めた。たとえば、Anneはテキスト・エディタを使って彼女の章のテキストを編集した。利用者とエディタの間にモニタ・プロセスを挿入すれば、Quiltでテキスト・エディタを使用することができる。このプロセスは、注釈リンクへのQuiltの内部参照を、人間が読み取れる形に変換する。論文作成中にも、Anneは彼女のテキストに注釈を付けることができる。このケースの注釈の1つのタイプは、たとえば「この参照をチェックせよ」というような自分自身への（パーソナルな）コメントである。この種の自分自身へのコメントは、目的を達したら、削除すればよい。もう1つのタイプは、たとえば「ここでもっと詳しく説明したほうがよいだろうか？」というようなBillへのノートである。この種の注釈は、Quiltによってオリジナル・テキストにリンクされ、電子メールでBillの元へ送られる。各ワーク・セッションが終るごとに、Quiltは、BillまたはAnneが実行したコマンドに関する基本情報と、彼らが見た文書や文書に加えた変更に関する情報をアクティビティ・ログに記録し、彼らが行った作業を文書化するように求めるプロンプトを表示する。

AnneとBillは共同作業のスタイルとして「共有」を選択しているので、BillはAnneのテキストを直接変更することができる。選択したスタイルによっては、Billの権限がAnneのテキストにコメントを付け、訂正を提案するだけに制限されることもある。Quiltは、共著者が互いのテキストを変更する際に、共著者によって選択された規則を強制的に適用する。共著者の1人が訂正を提案した場合、Quiltを利用して、2つのバージョンの相違を容易に調べることができ、改訂バージョンを交換するこ

とができる。1つの章にアクセスしたBillは、その章のテキストを読む傍ら、公の注釈とAnneからBillに宛てたメッセージを読み取ることができる。文書を修正する場合、Billはこれらの注釈のどれにでも応答することができ、さらに、基本文書またはその注釈のどれにでも彼自身のコメントを付け加えることができる。

AnneとBillは論文をCherylに見せる段階に達したと判断したので、Cherylをコメンテータとして共同作業のメンバ・リストに追加し、Cherylに文書の承認を求めるメッセージを送った。このメッセージはCherylの電子メールに表示され、Cherylはそこから文書にアクセスし、文書を審査することができる。Quiltは、Cherylが文書を読んだり、公の注釈やAnneとBillからCheryl宛てに送信されたメッセージを読んだりすることを許可する。Cherylはコメンテータとしての認可しか得ていないので、AnneとBillがやり取りしているプライベート・コメントを読むことはできないし、文書のテキストを変更することもできない。論文を読み終ったCherylは、Quiltの中で幾つかのプライベート・ノートを作成して、総合的な感想を記録し、幾つかの訂正を提案した。Cherylは、いくつかのテキスト・コメントを付け加えて、二、三のミスを指摘し、参照資料を提示した。しかし、音声でコメントを述べた方がより詳しく説明でき、理解されやすいと判断したCherylは、応答の大半を音声コメントとして記録することにした。

Cherylが論文に関するコメントを作成し終えると、QuiltはAnneとBillにCherylのコメントが入手可能になったことを通知する。また、Cherylは、AnneとBillがCherylの注釈の一つ一つを処理し終わったときにその旨を記録し、すべての注釈の処理が終ったときにその旨をCherylに通知するようなトリガの設定をQuiltに要求した。AnneとBillはCherylの注釈を吟味し、これに応えて論文を適宜訂正した。あるケースでは、彼らはCherylに宛先指定メッセージを送って、コメントの内容を詳しく説明するように求めた。Cherylは（電子メールを介して）直ちにこのメッセージを読み、即座にこの要求に応えた。しかし、Cherylは、AnneとBillが1つの注釈を処理するたびに、これに煩わされることはなかった。Cherylがセットアップしておいたトリガのお陰で、Cherylは彼女が行った提案の総合的な効果を見ることができた。このような再検討プロセスを数回繰り返した後、Cherylは論文の発表を承認し、AnneとBillは論文を雑誌に提出した。

(4) Quiltにおける制限

Quiltは、共同作業に係わるすべての問題の解決を目指している訳ではない。まず第一に、Quiltは共同作業の開始を支援するものではない。共同作業は、通常、非公式なコミュニケーションを重ねている内に開始されるものであり、物理的に近接しているところで発生する。しかし、Quiltの会議システムは、通常協力者たちが初期計画を立てるために集って行う会議を補うことができる。共同作業の社会的足場が固まってしまえば、共同執筆プロジェクトの強力な要素としての役割をQuiltに期待するこ

とができる。

第二に、Quilt は同期コミュニケーションのためのメカニズムを提供しない。協力者たちは作業を進めながら互いに直接話し合い、話し合いながら、彼らが共同で執筆している Quilt 文書を拾い読みしたりする。現行のプロポーザルでは、共同執筆者たちは、恐らく Quilt をオンラインで使いながら、あるいは、Quilt のプリント出力を手にもちながら、実際に集って会議をしたり、電話を使ったりして、このタスクを遂行することになる。

第三に、Quilt は、理念、思考、調査結果などを獲得し、これらを文書にまとめるタイプの著作システムとしては設計されていない。ただし、Quilt の中で、これらのシステムを文書エディタとして使用することは可能である。

最後に、Quilt の基盤になっているデータベースは現在のところ集中制御されるので、地理的に遠く離れた共同執筆者たちは Quilt を使用するために同一マシンへのアクセス権をもたなければならない。

3.6.2 PREP

原文タイトル：	Issues in the Design of Computer Support for Co-authoring and Commenting	
翻訳タイトル：	共同執筆および注釈のコンピュータ支援の設計に関する問題点	
著書：	Christine M. Neuwirth	(CMU, Pittsburgh, PA, USA)
	David S. Kaufer	(CMU, Pittsburgh, PA, USA)
	Ravinder Chandhok	(CMU, Pittsburgh, PA, USA)
	James H. Morris	(CMU, Pittsburgh, PA, USA)
出典：	CSCW'90 Proceedings, October 1990, pp.183-195	

本論文では、共同執筆と注釈の関係を調査するために開発されたPREP エディタのプロジェクトについて報告されている。このプロジェクトの一環として、共同執筆および注釈に対するコンピュータ支援を設計するに当たって次の 3 点が重点項目とされた。すなわち、①共著者とコメンテータとの間の社会的インタラクションの支援、②共同執筆と外部注釈における認識面の支援、③2 種類のインタラクションの両方の実際的な面での支援である。本論文では、これらの重点項目のそれぞれと取り組むために PREP エディタが採用した方式について説明されている。

(1) 共同執筆および注釈の支援に関する問題点

① 共同作業の社会的局面の支援

一般に、我々は長期間の共同執筆関係の社会的局面についてはほとんど何も知らないも同然と言える。現在のところ、共同作業の社会的局面に関する知識は非常に限られているが、共同作業の社会的

局面の1つとしての調整作業の問題については、ある程度の理解を得ている。調整の問題は、「著者が、単独ではなく、共同で著作作業を進めるに当たって、どのような活動をする必要があるか？」という疑問に対する答えとして発生してくる。

(a) 社会的役割の定義の支援

調整の問題に対する1つの答えは、社会的役割の定義を支援することである。様々な協力者の「適切な職務分担」（責任、対話のパターンなど）を規定することによって社会的役割を定義すれば、調整の問題は軽減される。Quilt システムはこの戦略を採っている。Quilt では、3種類の社会的役割、6種類のオブジェクト、1セットの動作を定義している。これらを定義したことで、Quilt の中では、様々なオブジェクトに対して、どの動作をどの役割が実行できるかを指定できるようになった。Quilt は、さらに、共同作業のタイプを3種類（排他タイプ、共有タイプ、編集者タイプ）定義している。利用者にタイプを定義する機能を与えたことは、Quilt の特筆すべき特色である。この機能がなければ、「著者」とか「コメンテータ」とかいった抽象的な概念に対して、オブジェクトに実行可能な動作を定義する際に多くの問題が持ち上がったと思われる。

社会的役割の定義を支援するシステムには、潜在的な問題がある。社会的役割の定義が「未熟」であると、予期せぬ結果を招く恐れがある。たとえば、プロジェクト開始直後には、誰がプロジェクトに「大きな貢献」をするか常に分かっているとは限らず、したがって、誰が著者としての役割を取るべきか分からないことが多い。しかも、開始時点で著者を定義してしまうと、「著者になれなかった」メンバの意欲が削がれ、この人はあまり貢献しなくなるかも知れない。したがって、執筆の社会的局面についてさらに研究が必要があるのと同じように、社会的役割を定義するシステムを実際に使用する著者たちについても、さらに研究を深める必要がある。

(b) 計画に関するコミュニケーションの支援

執筆作業の分担についての話には、論文の計画が出てくることが多いが、著者はプログラムのような感覚で計画を「実行」する訳ではない。その代わり、著者は執筆中に実行する事柄を決定するための資源として計画を使用する。このプロセスでは、部分的に完了した成果が重要な役割を果すケースが多々ある。部分的に完了した成果がタスク環境の一部となり、設計の後続過程を拘束するのである。さらに、著者たちは、言いたいことが見えてくるにつれて、彼ら自身に新しい目標を設定する。著者が単独で執筆活動を進める場合、彼らが自らに課した制約や、彼らが設定した目標を調和させる必要はないが、共著の場合には、彼らが設定した目標に関する見解を練り直したり、目標に見合った成果を得るために、しばしば、制約について話し合う必要がある。さらに、展開し続ける計画や制約についてコミュニケーションをもつことで、共著者やコメンテータが他人の計画を推測する必要がなくなるので、共同作業の性能が向上する可能性もある。

本プロジェクトの方式では、計画が執筆を制御しないことや、実際、執筆に先立って完全な計画を

立てるのは不可能であることを受け入れた上で、その代わりに、計画に関するコミュニケーションの支援に努力を集中している。

(c) 注釈に関するコミュニケーションの支援

著者たちは注釈を理解せず、自分のテキストに問題があると考えるよりは、読者の読み方が混乱していると受け取りがちで、注釈間に一貫性が欠けていたり、相い反する注釈を受け取ったりするとイライラする。著者とコメンテータの関係の問題は、著者が複数の読者に注釈を要請した場合に、一層急を要する問題になる。著者とコメンテータの関係について、経験から発せられた興味深い疑問は、著者グループが押し寄せる注釈をどのように管理し、有効利用すれば最適であるかというものである。

本プロジェクトでは、コメンテータ間のコミュニケーションにたくさん見られる混乱や困難を認める注釈方式を採用している。コメンテータ間のコミュニケーションを促すシステムを提供することによって、コメンテータ間のインタラクションがより有用で活発なものとなるよう期待している。

② 共同執筆における認識面の支援

(a) 執筆における認識的活動に対するタスク特定型支援

共同作業で行われるタスク特定型の活動（メモ書き、製図、執筆、身振りによる意思表示など）を理解して、これらのタスクを支援する特殊なツールを設計しようという試みは、これまでに幾つか行われているが、執筆の実質的作業と作成された草稿とを同等視する傾向があるため、ほとんどのテキスト注釈用ツールは作業中の草稿または草稿のアウトラインについてのコミュニケーションだけを支援している。しかし、経験豊富な著者は、テキスト自体とは直接関係ない中間的な外部表現を作成するのが普通である。アイデア間の概念的関係を差し示す矢印、ボックスなどの図形を作成する機能や、詳細部の可視表示を抑止する機能をサポートしていない環境で働く場合、フラストレーションを感じたり、重要な計画活動が省略されたと感じたりすることがあると、著者たちが報告している。著者たちは、通例、この種の中間表現を迅速に作成（および、廃棄）することを好み、これらの中間表現を、自らの論旨を把握するため、または、自らの論旨と他の共著者の論旨を調整するための、ほんの一時的な「スケッチ」、「いたずら書き」、または、「落書き」として使用することが多い。また、彼らは、ドローイング・エディタはこの種の「調整」機能に使用するには速度が遅過ぎ、結果としてこれらを調整のためには使用していない、と報告している。

執筆プロセスを研究し、注釈ツールを使用している著者たちを観察したところによれば、著者が執筆を進める際に行うメモ書き、製図、ノート取りの支援などの認識的な問題は、執筆において特に重要であり、共同執筆および注釈のためのコンピュータ支援を設計するときには、是非ともこれを考慮にいれなければならない。

外部コメンテータは、通常、作業中の草稿の作成に至る非公式な計画に関与していない点が、共著

者と異なる。さらに、外部コメンテータは既存の作業を行った草稿や未来の如何なる草稿についても所有権を主張することができない。にも係わらず、草稿よりも執筆プロセスの方をより強力に支援するツールを利用して、外部コメンテータと著者グループとの潜在的な新しい様々な関係を定義することができる。本プロジェクトの重要な目標は、外部コメンテータが共著者に対して果たす興味深い役割の調査を手助けする1セットのツールを構築することにある。この興味深い役割の中で最も重要と思われる点は、(i) 計画対象へのコメンテータのアクセスを可能とすること、(ii) コメンテータによる「著作テスト」を可能とすること、(iii) コメンテータに「著作テスト」の実行をリクエストまたは要求すること、の3点である。

(b) 共同執筆に係わる認識的活動に対する支援

共同執筆に関連した認識的活動は非常にたくさんあるが、ここではその中の1つである注釈へのアクセスだけを採り上げることにする。ほとんどのテキスト注釈システムは、ハイパーメディア・モデルに基づいている。これらのハイパーメディア・システムは、主要な情報アクセス方式として、ノードからノードへとリンク・アイコンをたどる方式を採用している。利用者は、通常、ノードをスクリーン上に提示し、その内容を読み、いずれかのリンクに着目し、その後、幾つかのリンクを選択的にトラバースする。この種の集中的なリンクの追跡はブラウジング作業には適しているが、その他の作業には問題が多い。ハイパーメディア・システムの誘導リンキング・システムを利用者の執筆ニーズに合わせてカスタマイズする研究を進めている研究者もいるが、まだアクセスの問題が未解決のままになっている。

本プロジェクトの方式においては、プログラムを利用者の認識的活動とマッチするようにカスタマイズする必要がある。

③ 共同作業の実際的な面に対する支援

著者用の注釈システムのほとんどは、基本的に、すべての協力者が同一のテキストを見られるものと仮定している。この仮定は、作業グループのメンバが遠隔地で作業を進めているときには実際的でない。多くの学術的共同執筆グループは遠く離れて作業を行っているので、遠隔共同作業の実際的な障害について心配する必要がある。共同執筆を実際に行うに当たっては、互換性と浸透性の2つが重要な問題となる。

(a) 互換性

共同作業のコンピュータ支援を行う既存システムの多くは、原稿へのマルチ・ユーザ・アクセス機能を採り上げているが、互換性のないシステムからのアクセスの問題を解決しているシステムはまだない。しかも、潜在的な共同作業資源（時間、金、コンピュータ・コンサルタントやプリンタなどの支援環境）がネックとなって、この種のシステムはなかなか採用されにくい。

(b) 浸透性

共同作業のための現在のコンピュータ・ツールは比較的浸透性が低い。つまり、電子テキストとハード・コピーの間の境界を越えるのが難しい。この結果、システムの有効利用や作業習慣に対する要求と、利用者の要求の間にミスマッチが存在する。たとえば、研究者が飛行機で旅行している間にハード・コピー原稿に注釈を入れるのは、そうめづらしいことではない。しかし、忙しい研究者が帰宅してから、電子的な形でこれらの注釈を入力する余分な手間を喜んで引き受けるとは思えない。近い将来、参加者がハード・コピーの入出力をあまり必要としなくなるとはとても思えないので、この問題を解決し損った場合には、互換性の問題に失敗したのと同じような否定的結末が予想される。したがって、共同作業を支援する次世代のコンピュータ・ツールは、システムの浸透性を高める必要がある。すなわち、電子媒体からハード・コピーへ、ハード・コピーから電子媒体への移動にかかる利用者の手間を最少限度まで減らす必要がある。

(2) PREP エディタ

PREP エディタは、MacApp が動作する Mac II の上にインプリメントされている。また、学生たちがロー・エンドのマシン上でもこのシステムを使えるように、モノクロディスプレイだけに限定して研究を進めている。

① 基本構造

構造レベルでは、PREP エディタは、Intermedia 、Neptune 、NoteCards などの文献に報告されている多くのハイパーメディア・システムと基本的特徴を共有している。このシステムは、アイデアにほぼ相当するチャンクを定義する。チャンクはテキスト、表、ツリー図、または、任意のイメージを含むことができる。このシステムは、著者が議論を構築するときに便利に使える外部表現として、幾つかのチャンク・タイプ（合成表、合成ツリー図）を特に標的としている。このワーク・スペースには利用者が初期にアイデアや議論を構成するときに可視的な理解を助ける作図ツールも含まれている。このシステムはチャンク間のリンクも定義するので、概念のネットワークを構築することができる。

チャンクは、共同作業者間で共有されるデータベースに格納される。しかし、アイデアのネットワークへ単に共有アクセスできるだけでは、共同作業には十分でない。共同作業者のスクラッチ・スペース全体を通読することは、通常の場合に、机の上に散らばっている本、ファイル・フォルダ、および、論文を厳密に調べることにほぼ相当する。相互理解を促進するために、PREP エディタはワーク・スペースの各部分についてコミュニケートするための規則を提供する。特に、このシステムでは、著者が「草稿」を定義できるようになっている。草稿というのは、著者が他のパートナーにアクセスさせることを目的として作成したワーク・スペース内の一領域のことで、所々が埋ったチャンクのグリッドで構成されている。グリッドの各カラムは、ワーク・スペースの様々な内容を格納するために使

われるのが普通である。カラムは互いに（または、メインカラムに）関連付けることができる。たとえば、1つのカラムには論文の内容を、もう1つのカラムにはその内容の構成するための計画を格納する。コメンテータは、カラムを読むときに、独自の注釈をカラムに付け加えることができる。図3.6-2に、3つのカラム（論文の計画、論文の内容、共著者の注釈）をもつ草稿を示す。草稿のカラムとワーク・スペース内の幾つかのチャンクでは、ワーク・スペースの内容の順序が異なるが、これはカラムの要求によるものである。慣れないうちは、PREPエディタの草稿は伝統的な直線的なテキストに基づく草稿とは大いに違って見えるかも知れないが、作業を進めるうちに、だんだん伝統的な草稿に似てくるのが普通である。これは、内容が、通常、計画や注釈よりも重要なものとして扱われるためである。

共有データベースの上に構築されたワーク・スペースと草稿が提供する構造を使って、共同執筆者や注釈者たちは、論文の内容に単に注釈を付加するに留まらず、計画を作成したり、計画や注釈についてコミュニケートしたりすることができる。

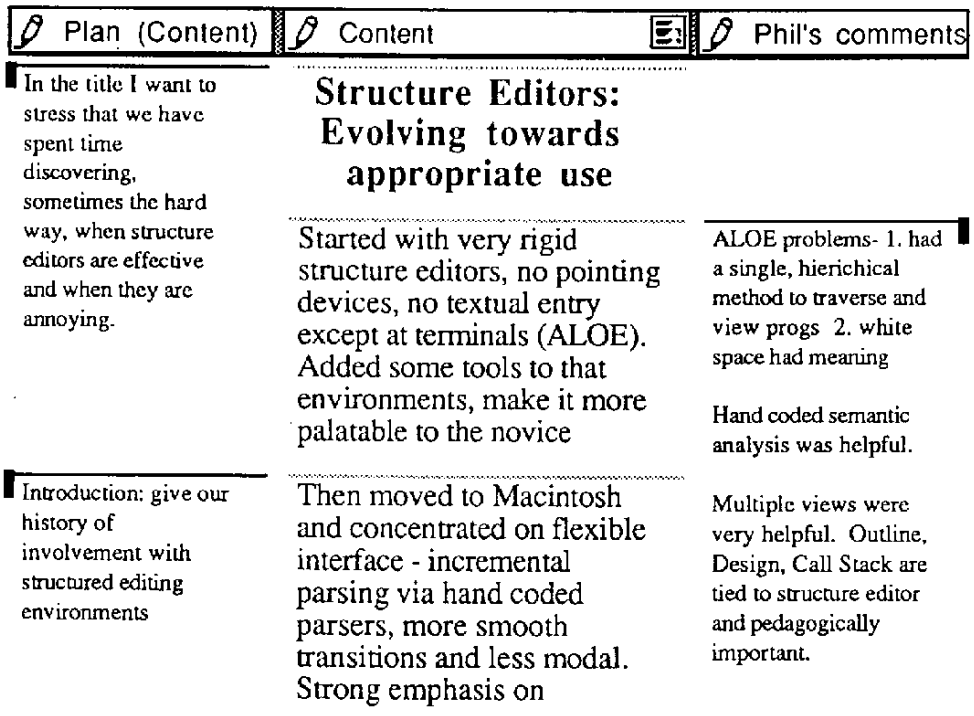


図3.6-2 「草稿」の中に三つのカラムをもつPREPエディタ

② インタフェース

草稿の可視表現については、現行のシステムでは注解の多い古いタイプの学術テキストに似たスタイルの注釈をサポートしているが、「ダイナミック注解」とでも呼ぶべき経路を追求中である。これは、フォント・サイズや空間関係を使ってシステム内のチャンク間の相互接続状態を示すことができ

るコンピュータの動的性質の利点を取り入れたものである。注釈の提供や注釈へのアクセスを容易にするのに役立つ可視システムを作るためには、可視グラマが著者のニーズをサポートできなければならない。たとえば本システムでは、コメントを可視的に整列すると協力者は「一目で」注釈を見分けられることを発見したが（図3.6-2参照）、フレキシブルなシステムにおいては、一般的に、任意の形状と、動的に選択されたアイテム間の複雑な相互接続状況を扱うことができる制約ベースのレイアウト・アルゴリズムが必要である。

③ バージョン化

共同執筆関係で最も共通なイベントの1つは、まず著者が草稿を他のメンバに提出し、次にこれを審査し、最初の著者が新しい資料を組み込む「編集→審査→組込み」のサイクルである。幾つかのシステムは、「変更バー」または他のヒストリ・メカニズムをサポートして、テキストが変更されたポイントを指示することによって、このプロセスを支援している。PREPエディタはさらにこれを推し進め、草稿とは別個のバージョンとして既存テキストへの訂正を可能としている。さらに、固有の計画スペースがあるので、訂正の背後の論拠もコミュニケートできる。このバージョン化による訂正機能を利用すると、著作の専門家たちは、既存資料を失う心配なく、草稿のスペースを操作することができる。

④ PREPエディタと既存システムの関係

PREPエディタは、一般的な意味で良好なハイパーテキスト・システムを提供するものではない。その代わりに、PREPエディタは、やや新しい可視的な構造を介して、リンクされたチャンクをサポートしている。これらの構造は、他のハイパーテキスト・システムの上に構築することが可能である。

3.6.3 DistEdit

原文タイトル：	DistEdit: A Distributed Toolkit for Supporting Multiple Group Editors
翻訳タイトル：	DistEdit：マルチプル・グループ・エディタをサポートする分散型ツールキット
著書：	Michael J. Knister (Software Systems Research Laboratory, Department of EECS, University of Michigan,Ann Arbor, MI, USA) Atul Prakash (Software Systems Research Laboratory, Department of EECS, University of Michigan,Ann Arbor, MI, USA)
出典：	CSCW'90 Proceedings, October 1990, pp.343-355

本プロジェクトは、分散型環境での協同作業をサポートするアプリケーション構築のためのツールキットの提供を目的としている。本論文では、そのようなツールキットとして、分散型環境でのイン

タラクティブなグループ・エディタ構築に使えるDistEditについて紹介されている。ここでは、まず本ツールキットの要件が述べられ、次に関連する他の研究との比較が行われ、さらに、試験システムのオペレーション、ツールキットの設計、このツールキットを使ったグループ編集に適合させるためのエディタの修正について述べられている。また、ツールキットの利用で得られた結果を提示し、設計の決定に関与したトレードオフについて述べられている。

(1) DistEditツールキットの要件

DistEditツールキットの要件は次のような非常に単純なものである。

① マルチユーザ協同作業

DistEditツールキットを使って構築したエディタは、ユーザが物理的に離れたまま協同してテキスト・ファイルの編集を行えるものでなければならない。

② リーズナブルな性能

DistEdit内で使用する通信プロトコルは、グループ編集を不便に感じないくらい、少ない遅延で全てのユーザが一貫してファイルを見ることができなければならない。

③ マルチプルな既存エディタとの互換性

1つのグループ・セッションで異なるエディタを利用することができなければならない。誰でも自分の好きなエディタがあるので、グループ・セッションに参加する時、それ以外のエディタの利用を強制されないことが望ましい。

④ 耐故障性

グループ・セッションは、マシンのクラッシュや人の参加、退席にかかわらず、スムーズに継続されなければならない。

⑤ 通信プロトコルの隠蔽

エディタをDistEdit を使ったグループ編集に適合させる際、通信プロトコルのような分散型システムに関する知識が必要であってはならない。

(2) 関連する研究との比較

グループ編集のために特別に設計されたエディタとしては、GROVEとShrEditがある。DistEditはこれらのシステムとは異なり、エディタではなく、シングル・ユーザ・エディタをグループ編集タスクに適合させるために使うツールキットである。DistEditツールキットを使えば、1つのグループ・セッションで、さまざまなハードウェア上のさまざまなエディタを利用することが可能である。例えば、あるグループ・セッションで、1人がEmacs、もう1人がvi（どちらも端末上で稼働）、さらにもう1人がxedit（Xが稼働するワークステーションが必要）を使うことも、それぞれのエディタのコードを

ほんの少し修正するだけで可能となる。

さまざまなエディタをフロントエンドとしてDistEditを使うので、Lantzが述べている構成可能ユーザ・インタフェースに似ている面もある。Lantzは、ユーザ・インタフェースを、単一のバックエンドとインタフェースする個別の交換可能なモジュールにすることで、カスタマイズ可能なユーザ・インタフェースを作ることを提案している。同様に、DistEditではさまざまなエディタが交換可能なフロントエンドとして作動するが、Lantzの述べたモジュラー・フロントエンドがかなり小型で開発が簡単なものを意図していたのと対照的に、DistEditはサポートするフロントエンドより実際には大幅に小さくなる。

DistEditツールキットの設計によって、非常に高度な耐故障性が提供される。グループ・エディタを設計する際、単純な方法として多くのエディタや協同作業用ツールに利用されているものに、クライアントーサーバ・モデルがある。これは、エディタ・バッファの状態を保持する1つのサーバが存在し、すべての編集コマンドが共有サーバを通るようにすることで、ユーザのビューの相互一貫性を確保している。この設計は単純だが、サーバのクラッシュに弱い。さらに、1人の人間がファイルを編集している場合でさえ、更新にはサーバを通らなければならないため、編集が遅くなる。これに対して、DistEditは各ユーザのエディタ・バッファ状態のコピーを保持する。

グループ用ツールキットに対するもう1つのアプローチとして、LIZAを挙げることができる。LIZAは、グループウェアを迅速にプロトタイプ化するための高度なツールの集合体で、メッセージ送信やムード表示、協同編集、スライド展示、グループのモニタ等のサポートを行う。このようなフレームワークと基本的なツールセットを持ったLIZAは、主に協同作業の新しいアイデアを迅速かつ容易に試験することを目的としている。DistEditは、これよりもかなりレベルの低い、アプリケーションが特定されるツールキットであり、ユーザが既に慣れている状況でのグループ編集の可能性を深く探求するために設計されている。

密接に関連した協同作業システムとして、より非同期つまり非リアルタイムなインタラクションをサポートするものがある。この例としては、CESやQuilt等のエディタが挙げられる。これらは、ユーザが同じ文書に対して作業できるが、通常は異なる時期に異なるセクションで行うものであるため、システムのインタラクションの持続期間は非常に長く、何日もかかることさえある。このようなシステムでは、耐故障性や更新のリアルタイムな伝播に関する問題の多くは重要ではない。DistEditツールキットはより密接に連結されたリアルタイムのインタラクションを提供することに焦点を当てている。

(3) エディタ例：DistEditベースのMicroEmacs

ここでは、DistEditを使えるように修正したMicroEmacs試験システムのオペレーションを、ユーザの立場から検討する。GNU Emacsのオペレーションも、編集オペレーションが強力になることを除い

て、同様である。一般に、グループ編集セッションの各参加者は、DistEdit ツールキットを使うために修正が行われているのであれば異なるエディタを使うことができる。

MicroEmacs エディタのオペレーションは、DistEdit システムを使わない通常の利用とほとんど変らない。ユーザはプログラム名を指定してエディタを起動してから、編集するファイル名を指定する。エディタは最初のユーザのために動くので、すべては通常と同じで、状態表示行にこのユーザが「マスタ」であることを示すインジケータが出る点だけが異なる（図3.6-3 参照）。

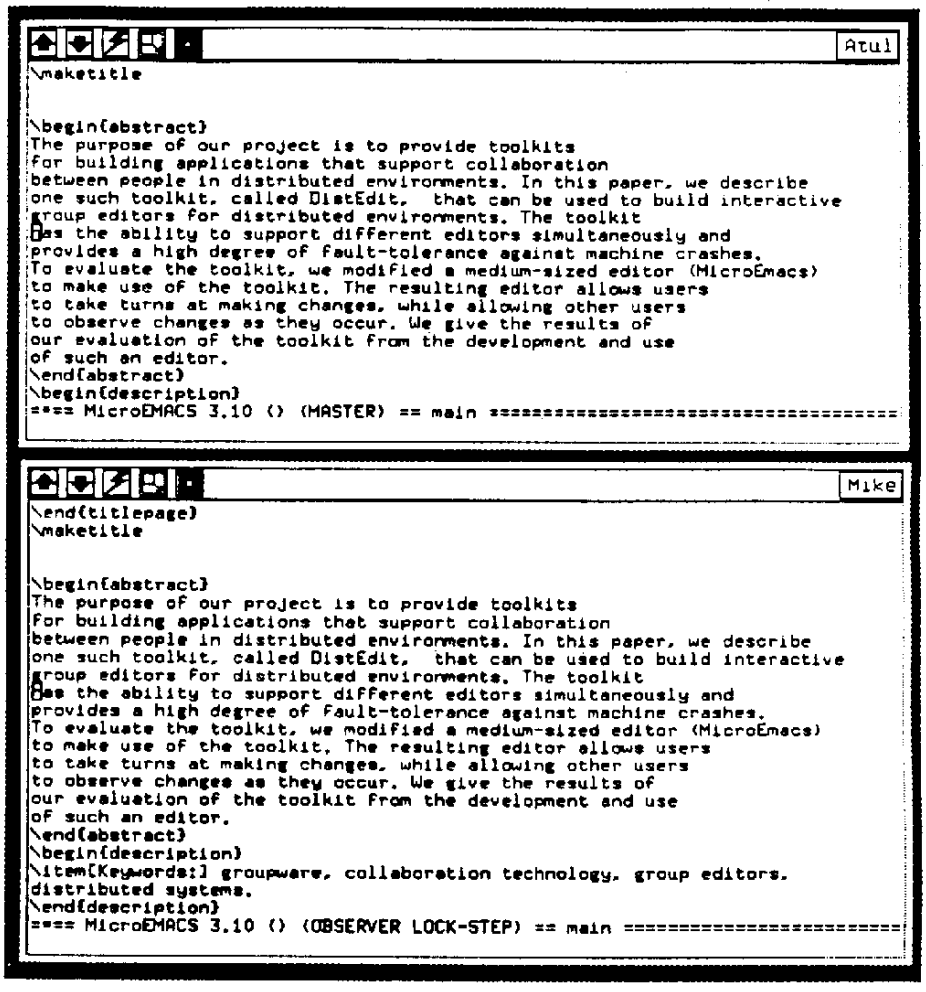


図3.6-3　DistEdit をベースにした MicroEmacs グループ・エディタのグループ・セッションにおけるマスタのウィンドウとオブザーバのウィンドウ。ウィンドウはそれぞれのワークステーションに現れる。各ウィンドウ下部の、セッション制御保持者を示す状態表示行に注意。

別のユーザが同じファイル名でプログラムを動かした場合には、エディタ・バッファにロードされたテキストは最初のユーザのものとなり、最初のユーザがすべての変更を行うと共に終了する。最初

のユーザが「マスタ」で、すべての編集機能を持っており、2番目のユーザは編集機能のない「オブザーバ」である。オブザーバである2番目のユーザは、マスタの行うすべての変更とカーソルの動きを見るが、オブザーバのカーソルはここでマスタのカーソルに「ロックステップ（連動）」されている。

オブザーバは、テキストを変更するようないかなるオペレーションも行えず、変更を行おうとしてもそのようなオペレーションは無効である。しかし、ファイルをセーブしたりウィンドウの大きさを変えたり、テキストを変更しない編集コマンドを発することはできる。

オブザーバが自分のカーソルを独自に動かしたいと思ったら、自分のカーソルをマスタのカーソルの従属から解放する「ロックスイッチ」コマンドを呼出すことができる。この「フリーカーソル」モードでは、オブザーバは自分のカーソルをどこへでも好きなところに動かすことができるが、やはり変更を行うことはできず、マスタの行った変更は全て同時に更新される。ロックスイッチとフリーカーソルは、オブザーバが自由にオン／オフ切換えができる。

常に1人のユーザがマスタで、他の人はオブザーバである。マスタは、「コントロールスイッチ」コマンドを呼出していつでも制御を放棄することができる。全てのエディタの状態表示行が更新されて、マスタがいないことが示される。マスタがいなくなると、どのオブザーバもコントロールスイッチ・コマンドを使って制御を得てマスタになることができるが、コントロールスイッチを発した最初のユーザがマスタになる。そうすると状態表示行が更新されて、マスタ／オブザーバ状態が表示され、新しいマスタが上記のオペレーションを行うことになる。マスタがいない時は、誰も変更を行うことができない。制御は、電話のような外部ラインやオンラインのトーク・セッションでの話し合いで譲渡する。

1つの編集セッションにユーザは何人でも参加することができる。ユーザは、他ユーザに影響を与えずに、いつでもセッションに入ったり抜けたりすることができる。1人のユーザのマシンが故障しても、そのユーザがセッションから抜けるだけである。マスタがセッションを抜けたり故障に遭遇したりすると、他ユーザはマスタなしという状態を見て制御を得られることになる。編集中のテキストは、全てのユーザがセッションを抜けて誰もローカルにセーブしておかなかった場合にのみ失われる。

1つのセッションのユーザは全員同じテキスト・バッファを持つ。見え方は、ウィンドウのサイズによってユーザごとにやや異なる。ロックステップの場合、オブザーバのカーソルはウィンドウ内を動いて常にテキストの同じ論理的位置にあるが（同じ行及び桁）、実際のスクリーン上の位置は異なることがある。

キーポイントは、サポートされているエディタすべてで、通常の編集コマンドが普通通りに動くことである。ユーザ側から見た変化は、ロックステップとコントロールスイッチ・コマンド、マスタ／

オブザーバとロックステップ／フリーカーソル状態インジケータだけである。

(4) DistEdit ツールキットの設計

DistEdit システムは、多種のビジュアル・エディタに容易に適用できるモジュラ・ツールキットとして設計されている。適用の要件としては、エディタがテキスト・バッファを直接修正する少数のサブルーチンを持っていることのみである。これを持っていないと、エディタには広範囲な修正が必要となる。エディタは、テキスト・バッファから読み取るルーチンについては修正する必要はない。

① DistEdit プリミティブ

DistEdit プリミティブとは、エディタのコードから呼出すことのできるオペレーションである。プリミティブは、エディタからの 1 つのファンクション・コールで開始されるが、実行場所に着く前に、何層かのコードやネットワークを進むことができる。この意味で、DistEdit プリミティブは、多くの遠隔実行を生じさせることになる点を除けば、遠隔手続き呼出し (RPC) のようなものである。

DistEdit のプリミティブには、(1) 更新プリミティブ、(2) カーソル位置通知プリミティブ、(3) 制御プリミティブの 3 種類がある。DistEdit には、すべてのテキスト更新オペレーションを実行する 更新プリミティブが 3 つ（ストリング挿入、ストリング削除、ストリング置換）と、カーソル位置の変更を同報通信するカーソル位置通知プリミティブが 1 つ、状態と制御情報の譲渡のための制御プリミティブが 2 つ（制御交代、ロックステップ切替え）ある。

更新プリミティブとカーソル位置通知プリミティブを発することができるのは、マスタ・エディタのみである。制御プリミティブは、オブザーバでもマスタでも使うことができる。

② 基礎となる通信ソフトウェア

同報通信機能と誤り制御が高品質であり、プロセス・システムが軽いという点から、通信パッケージとして分散型アプリケーションをプログラムするためのツールキットである ISIS が選ばれている。

ISIS の同報通信機能によって、DistEdit が低レベルの通信を取り扱う必要がなくなった。メッセージは失われず、すべての同報通信がグループの全参加者に同じ順序で到着することが保証される。

ユーザは、いつでもグループ・セッションに入ることができる。ユーザがセッションに入ると、エディタの状態が現在のユーザの 1 人から、新しいユーザのエディタに転送される。ISIS には、新しいユーザの通知メカニズムと状態転送のための耐故障性のある通信プロトコルがある。

(5) エディタに必要な修正

DistEdit ベースのグループ編集セッションで通常のエディタを使うには次のコード修正が必要となる。

① エディタ・バッファの全てのテキスト更新オペレーションを、3 つの DistEdit 更新プリミティブへ

の呼出しにマップしなければならない。

② 3つのDistEdit更新プリミティブを、エディタのテキスト更新ルーチンにマップし直さなければならない。これらのマッピングでは、エディタの全てのテキスト更新オペレーション（テキスト・バッファを直接更新する全てのルーチン）が機能が5個から15個の小さなルーチン・セットに入っていることが理想的である。

③ DistEdit制御プリミティブへの呼出しは、エディタのユーザ・インタフェース部分に挿入して、ユーザがロックステップ・コマンドとコントロールスイッチ・コマンドを実行できて、ユーザにマスタ／オブザーバとロックステップ／フリーカーソル状態を示せるようにしなければならない。

④ DistEditがカーソルの移動や照会を行って、テキスト・バッファの行を読み取れるよう、アクセス・ルーチンを備えなければならない。DistEditがテキスト・バッファの状態を参加ユーザに転送するためにテキスト・バッファの行を読むルーチンが必要になる。

⑤ テキスト・バッファの変更をスクリーンに表示するため、DistEdit内から呼出せるルーチンを備えなければならない。

システムの通信プロトコルやマルチプル・プロセス、耐故障性機能を扱う部分はすべてDistEditツールキット内に隠されている。上記の変更はどれも分散型プログラミングについて何も知らなくても実行できるようになっている。

(6) 結果

MicroEmacsとGNU Emacsのエディタを、DistEditツールキットを使った分散型グループ・エディタに変換し、両方のエディタを同じグループ・セッションで使った。

オリジナルのエディタ・コードの変更はほとんど必要なく、性能は十分に受入れ可能なものだった。

① MicroEmacsとGNU Emacsに必要な変更

(a) 変更量

DistEditを使うためのMicroEmacsの修正では、オリジナル・プログラム本体内で、9行のCのコードの追加と6行の変更が必要だった。マッピングと特にMicroEmacsのためのサポート機能を入れた追加ファイルには、約60行のコードが必要だった。

DistEditを使うためのGNU Emacsの修正では、オリジナル・プログラムの本体内で10行の追加と12行の変更が必要である。GNU Emacsのための追加サポートファイルには、約300行のコードが必要だった。

(b) 変更の困難さ

エディタの修正では、コード内の次の3つの領域の位置決定と修正を行った。(i) バッファ修正ルー

チン、(ii) トップレベルのユーザ・インタフェース、(iii) 入力ブロックポイント。

このうち入力ブロックポイントの位置決めが最も難しい作業だった。これは、ユーザからの入力待ちをエディタがブロックするポイントである。入力ルーチンは、DistEdit プリミティブが ISIS を通じて同報通信しながらキーボードかマウスの入力を待つようにするため、修正しなければならない。コードのこの部分の位置決めと修正は、入力抽象とマシン従属性が多レベルにわたるため困難だった。

② 性能

本試験では、それぞれ 8MB のメイン・メモリを持ち、Ethernet を介して接続された Dec 3100 ワークステーションを使ってエディタを動作させた。ユーザが数人いても性能は妥当なものだった。マスタ・エディタは、全体として大きくスローダウンすることはなかった。オブザーバのエディタでは、おそらく同報通信にかかる時間が常に同じではないためか、カーソルがジャンプすることがあった。最初、ISIS 通信パッケージのスピードがタイプについていけないのではないかとの懸念があったが、このような場合には、キーストロークごとに更新を送るのを防ぐため、何等かのキャッシングを追加することになっただろう。

(7) 考察

① マルチプル・エディタ・インタラクション

上の 2 つのエディタは 1 つのグループ・セッションでうまく動いたが、オペレーションやテキスト・バッファの見え方はエディタによって異なる。ウィンドウ表示とテキスト・マークがその例で、エディタがそれぞれ互いの機能を補完することもできる。

DistEdit では、全ユーザが同じテキストを見るのではなく、テキストが常に一貫していることが保証されるだけである。表示のされ方はエディタによって決り、長い行の処理などでは見え方が大きく異なる。（規定値の）GNU Emacs では長い行はスクリーンの次の行に表示するが、MicroEmacs は水平スクロールを使っている。ユーザの選ぶウィンドウ・サイズも大きく異なることがあり、同じテキストでもユーザによって見え方が変ることになる。

異なるエディタを使う大きな利点として、互いに補完しあうよう複数のエディタの機能を散在できる点がある。例えば、MicroEmacs（「取消」機能がない）ユーザの実行したオペレーションを、制御を切換えた後、GNU Emacs（「取消」機能がある）のユーザが取消すことができる。このように、異なるエディタと異なるユーザを組合せて、1 つのエディタよりも強力なパワーを生むことができる。

② 通信層構成

セッション内のエディタ間で通信を行うにはいくつかのレベルがある。本システムでは、通信用に基本的な更新と制御プリミティブを持たせることにした。各エディタの低レベルのオペレーションが、これらプリミティブへの呼出しに翻訳される。

代替案としては、ユーザのキー入力に直接割込んでこれを全ユーザに同報通信する方法もあったが、これには2つの問題がある。1つは、エディタが同じグループ・セッション内で異なっていると、他エディタからのキー入力を理解するよう全エディタを修正しなければならないため、実行が難しいことである。2つ目は、セッション内のエディタを同一にして単純化しても、同時更新を可能にするための拡張が簡単にはできないことである。異なるユーザからのキー入力は、予期せぬ行動につながる可能性があるため、1つのストリームに単純に混合してしまうことはできない。

③ プリミティブの選択

通信プリミティブを選ぶ際、いくつかの選択肢があったが、エディタから独立した非常に基本的なプリミティブとした。代りに複雑で強力なプリミティブ・セットを選ぶこともできた。特定のエディタ、例えばGNU Emacsで利用できる機能として通信プリミティブを選択することだが、このような強力なプリミティブには、ストリングの挿入、削除、置換ではなく、全体的な置換やブロック・インデンテーションのようなオペレーションを含むことになる。

基本的なプリミティブを採用した要因は次の3つである。1つ目は、強力な基本プリミティブ・セットだと、システムに新しいエディタが加わるごとに、プリミティブ・セットを更新ルーチンの呼出しに翻訳するためのコードを追加しなければならない点である。第2に、非常に強力な基本プリミティブ・セットの大半をあまり頻繁に使わない場合の性能低下の可能性である。長いメッセージ・フォーマットが必要になるため、システムの帯域幅ニーズが増すことになる。エディタ・オペレーションの大半は、シンプルな挿入や削除であろうと予測して、これらをプリミティブとした。第3は、エディタ・プリミティブの数が多くて複雑だと、同時更新の取扱いが複雑になる可能性である。エディタの提供する各オペーレション間の矛盾を解決するアルゴリズムが必要となる。単純なエディタ・プリミティブを使うことで、同時更新の解決タスクは容易になる可能性が大きい。

3.7 グループメモリシステム

3.7.1 gIBIS

原文タイトル：	gIBIS: A Hypertext Tool For Exploratory Policy Discussion
翻訳タイトル：	gIBIS：調査方針の議論のためのハイパーテキスト
著書：	Jeff Conklin (MCC, Software Technology Program, Austin, Texas, USA) Michael L. Begeman (MCC, Software Technology Program, Austin, Texas, USA)
出典：	CSCW'88 Proceedings, September 1988, pp.140-152

本論文では、初期の設計審議状況の獲得を支援することを目的として設計された用途特定型のハイパーテキスト・システムについて紹介されている。このシステムは、MCC Software Technology Program の中のDesign Journal と呼ばれるハイパーテキスト・プロジェクトの一環として開発されたプロトタイプ・ツールである。本論文ではまず、このシステムのベースとなるIssue Based Information Systems (IBIS)（問題点に基づく情報システム）と呼ばれる特殊な方式について説明され、つぎにgIBIS (graphical IBIS)と呼ばれるこのシステムの機能について述べられ、最後にこのツールに関する予備的な観察について説明されている。

(1) IBIS 方式

Horst Rittel によって開発されたIBIS 方式は、複雑な問題の設計プロセスが、基本的には関係者（設計者、顧客、開発者など）が設計上の諸問題を解決するために各自の専門知識や観点を伝え合う関係者間の会話そのものである、という原則に基づいている。問題、関心、疑問などはいずれも設計上の「問題点 (Issue)」と成り得るし、設計を進めるためには（合意が得られていなければ）これらの問題について議論する必要がある。実際、IBIS モデルにおいては、設計プロセスを構成しているのはこの「議論」に他ならない（自分自身との「議論」を排除するということではない。gIBIS は、対話的にも独白的にも、どちらにも使用できる）。

IBIS モデルは、設計上の問題の主要な Issue を互いに結びつけることに焦点を当てている。1 つ 1 つのIssue が多くの「解決案 (Position)」を持ち得る。Position というのは、Issue を解決するステートメントつまり主張のことである。Position は互いに排他的であることが多いが、IBIS 方式は Position が排他的であることは要求していない。つぎに、Issue の各 Position ごとに、その Position を支持する「議論 (Argument)」または、その Position に反対する Argument が、1 つまたは 2 つ以上存在し得る。したがって、独立した 1 つ 1 つの Issue がツリーのルートとなり（空のツリーの場合もある）、Issue の子供としての Position と、Position の子供としての Argument を持つことになる。

IBIS には、9 種類のリンクがある。たとえば、Position は、Issue との間に、Responds-to リンクをもつ。この Responds-to リンクは、Position と Issue の間にしか使えない。Argument は、その親 Position との間に、Supports リンクまたは Objects-to リンクをもたなければならない。Issue は、他の Issue との間に、Generalize リンクまたは Specialize リンクをもつことができる。また、Issue は、他の Issue、Position、および、Argument との間に、Question リンクまたは Be-suggested-by リンクをもつことができる。

代表的な IBIS の議論は、誰かが「X をどのように処理すべきか？」といった疑問を含む Issue ノードを提示することで始まる。その人物は、同時に、X の処理方法を提案する Position ノードを提示することもできるし、その Position を支持する Argument ノードを提示することもできる。これに応答し

て、あるユーザは別の Position を提示し、これを支持する独自の Argument を提示するかもしれない。その他のユーザもまた、それぞれの Position を提示したり、いずれかの Position に対する支持または反対の Argumentを提示したりすることができる。これに加えて、Argument によって発生した新しい Issue が提示され、直接この Issue の元となったノードにリンクされたりする。図3.7-1 に、IBIS 方式の中で有効とみなされる動作をすべて規定した状態遷移図を示す。

IBIS 方式には、停止のための規則は存在しないし、幾つかの Position に合意に達して Issue が解決されたことを記録する特定の方法も存在しない。むしろ、議論の目標は、関係者のそれぞれが、互いの提案の特定の要素を理解すべく努めること、および、個人の観点を他の関係者に説きつけるべく努めることにある。さらに、この方式は、中心的な Issue の探索、量的に片寄りのない質疑応答、1 つの観点を証拠を挙げて支持するなどの、より建設的な動作を支援する。

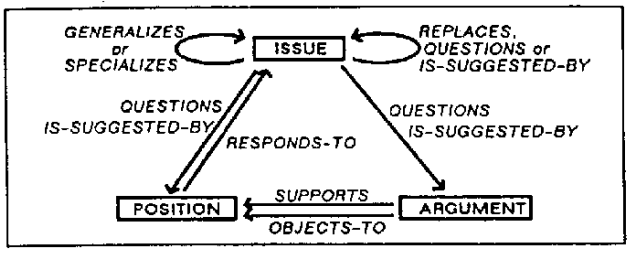


図3.7-1　IBIS において有効な修辞的動作のセット

gIBIS をインプリメントするに当たって、IBIS 方式に次のような変更と拡張を加えて、必要なフレキシビリティを確立した。(i) IBIS のフレームワークの中で自身の思考を表現する方法を発見できなかったユーザのための「エスケープ」メカニズムとして、ノードとリンクに Other というタイプを追加した。(ii) 要求文書、設計スケッチ、コードなどの非 IBIS マテリアルを含むノードのために、External というタイプを追加した。(iii) 他の Position を specialize（特殊化）または generalize（一般化）する機能を Position に与え、同様に、Argument にもこれらの機能を与えた。

(2)　gIBIS ツール

基本的な gIBIS インタフェースを図3.7-2 に示す。図に示されるとおり、インタフェースは4 つの独立したウィンドウ、すなわち (i) 画面左側のグラフィカル・ブラウザ、(ii) 右側上のノードへの構造化された索引ウィンドウ、(iii) 索引ウィンドウのすぐ下のコントロール・パネル、(iv) ノードの属性と内容およびリンクを表示する検査ウィンドウから構成されている。ユーザがノードまたはリンクを選択すると、その内容が即座に検査ウィンドウに表示される。

①　ブラウザ

ブラウザは、IBIS のネットワーク構造（すなわち、Issue、Position、Argument の各ノードとその間のリンク状態）を目に見える形で表示するためのウィンドウである。ノードとノード間の相互接続リンクは、実質的にサイズが無制限のキャンバスに表示される。ブラウザのほとんどの部分は、ネット

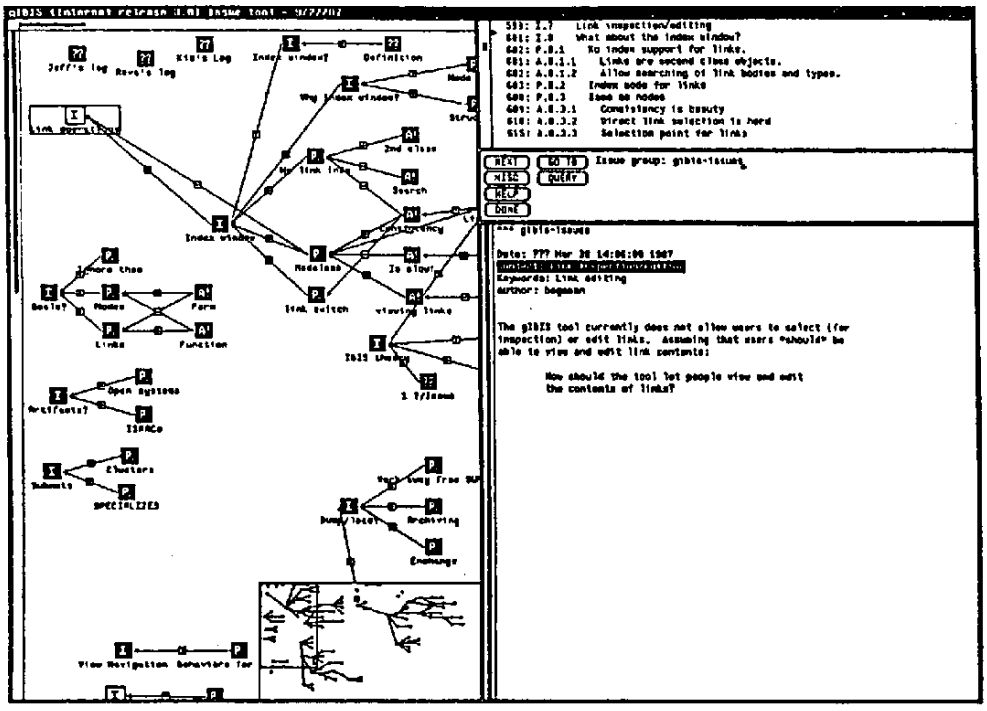


図3.7-2　gIBIS のインタフェース

ワークのローカル・ビューとして使われる。この部分には、現在関心をもっているエリアの「ズームイン」ビューを表示して、ノードとリンクの完全な詳細を見ることができる。ブラウザの右下の部分は、データのグローバルな概観を表示するための領域として予約されている。この部分には、ノードのラベル、リンクタイプのアイコン、ネットワークの細かい構造を作り上げている2次リンクなどを一切排除して、ネットワーク全体の概観を表示することができる。また、グローバル・ビューは、長方形の囲みによって現在のローカル・ビューの範囲と位置を示している（図3.7-2の例では、ローカル・ビューは、グローバル・ビューの左上隅から下に向って伸びている）。ブラウザのウィンドウ領域内で、キャンバスはスクロールすることができる。

ブラウザは、ノードやリンク（すなわち、表示したいオブジェクト）への直接操作式のインタフェースをサポートしている。マウスクリックで表示したいオブジェクトを選択すると、ブラウザの中でそのオブジェクトがハイライト表示されてボックスで囲まれ、その内容が検査ウィンドウに表示され、その索引ラインが索引ウィンドウの一番上にスクロールされる。

また、マウスクリックにより表示されるメニューから、オブジェクトの作成、編集、削除、移動などの操作を指示することができる。例えば、タイプがIssueのノードを選択すると、Issueに対して実行できる有効操作を反映したメニューが表示される。このメニューには、「従属ノードの生成」、「ノードの移動」、「ノードの編集」、「他のノードへのリンク付け」、「ノードの削除」などの項

目がある。ここでユーザがタイプがPositionである従属ノードの生成を選択すると、この従属ノードは選択されているIssueの右側に配置され、タイプResponds-toのリンクによって自動的にIssueとリンクされることになる。このメニュー選択を行うと、検査ウィンドウ（右下のウィンドウ）が半分に分かれ、下部にノード作成ウィンドウが表示される。このノード作成ウィンドウは、構造化されたテンプレートをウィンドウ領域一杯に表示するように予め設定されている。そこでユーザは、テンプレートの構造化されたフィールド（Subject、Keywordsなど）にデータを入力し、次に、ノードのトピックに関する任意の記述を入力する。ユーザがノードに関するデータを入力し終えてコントロール・パネルのSUBMITボタンを押すと、ノードが解析されて格納され、これに伴ってブラウザ・ウィンドウと索引ウィンドウが更新される。

また、ユーザが「他のノードへのリンク付け」というメニュー項目を選択すると、現在選択されているノードに有効な1セットの出リンクタイプのメニューが表示される。このメニューには、「質問」、「一般化」、「特殊化」などの項目がある。ユーザは、これらのリンクタイプの中から、いずれかを選択した後、ブラウザ・ウインドウ上でリンクすべきノードをマウスクリックで指示する。

gIBISは、カラー・モニタ付きのSUNワークステーションで使用するように設計されており、ノードやリンクタイプの情報、さらに「currently selected（現在選択中）」や「matches the current query（現在の照会と一致）」などの特殊なノードの状態が、色分けされて示される。白黒モニタを使用しているユーザのためには、色分けされた情報をアイコンで表示する機能を設けている。gIBISツールは、デフォルト設定のままでは、ユーザにカラー情報とアイコン化情報を両方とも提示するが、どちらの情報もユーザの手で抑止できる。

② ノード索引ウィンドウ

ノード索引ウィンドウには、現在のIBISネットワーク内のノードを、その順序に従って階層構造に配置したビューが表示される。Issue、Position、およびArgumentには通し番号が振られる。たとえば、次の図3.7-3はまずIssue 8 (I.8) のSubject行を示しており、これは子をもたない。次のI.9は、その最初のPositionノード（P.9.1）が子として2つのArgumentノード（A.9.1.1およびA.9.1.2）をもっている。Issueは単純に作成順に並べられる。ユーザは、ビュー構成パネルを利用して、Subjectだけでなく、Author、Keyword、または、ノードのLabelに基づく索引情報を、自由に構成することができる。

ノードは、ブラウザからだけでなく、索引からも選択できる。ノードの索引ラインの上でクリックすると、そのノードが現在のノードになる。これに伴い、そのノードのアイコンがブラウザ・ウィンドウの中でハイライト表示され、そのノードの内容が検査ウィンドウに表示される。

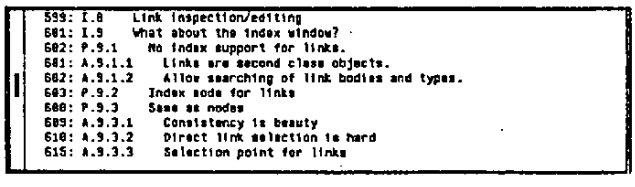


図3.7-3 ノード索引ウィンドウの例

③ コントロール・パネル

コントロール・パネルは1セットのボタンで構成され、これらのボタンを使用することによってgIBISツールの機能性を単なるノードとリンクの作成機能を越えたものに拡張することができる。各ボタンをマウスクリックするとメニューが現われ、このメニューを使って基本機能を拡張したり、基本機能をニーズに合わせて変更したりすることができる。たとえば、NEXTボタンを選択すると、通常、gIBISはユーザが読み終った現在のノードを記録してから次のネットワーク・ノードを表示する。しかし、NEXTボタンの上でマウスの右ボタンを押すと、mark current node as unread（現在のノードに未読マークを付けよ）という項目を含むメニューが表示される。

また、MISCボタンを選択すると、さらにユーザのニーズに合わせてインタフェースの特定の状況を作り変える機能や、ユーザから開発者へツール・グリップ／提案を送る機能、すべてのノードに読了／未読マークを付ける機能、IBISネットを直線化およびプリントする機能、Issueグループに名前を付ける／消す機能などを選択することができる。

QUERYボタンを選択すると、小さな照会・構成ウィンドウが表示され、その中の項目（Type、Date、Author、Subject、Keywordsなど）から照会したいものを選択してタイプインする。照会の結果は、ブラウザ・ウィンドウ（ローカル・ビューとグローバル・ビューの両方の中で、選択したノードが明るい黄色に変わる）と索引ウィンドウ（照会を満足するノードの索引ノードだけが表示される）の両方に表示される。

④ リレーショナルDBMSの利用

gIBISをインプリメントするに当たっては、工数が非常に限られていたため、既存のリレーショナルDBMSであるUNIFYの上にgIBISを構築することにした。データ蓄積マネジャとしてリレーショナルDBMSを選択したことで、同時制御、レコード・レベルのロッキング、信頼性の高いデータ蓄積、高速アクセス方式、および、妥当な検索エンジンを手に入れることができた。さらに、このDBMS(UNIFY)は無解釈データタイプ — 基本的には、任意の長さのテキスト、デジタル化した音声、グラフィック・ビットマップ等々を書き入れることができるフィールド — を提供しており、データベースの記録に不可欠な一部としてノードのボディを格納することができた。

しかし、DBMSをデータ蓄積マネジャとして利用したために、閉鎖的なシステムになってしまうという大きな不利益も生じた。本質的に、Issueネットワークが参照するすべてのオブジェクトは、DBMSの中に置かれていなければならないのである。残念ながら、Issueベースの議論に出てくる多くのオブジェクト（要求書、構造説明書など）やネットワークから得られる結果（コードやドキュメントなど設計の成果物）は、このデータベースの範囲外にあるので手が付けられない。このため、特殊な代理タイプのノードを作成して、「盲信的な」方法でgIBISが外部オブジェクトを参照できるよう

なメカニズムを作っている。この代理ノードは、外部オブジェクトを指示するポインタ（通常は、ファイルへの完全パス名）と、gIBIS がオブジェクトを表示するときに呼び出す任意のディスプレイ・プログラムの２つのパートをもつ。デフォルトのディスプレイ・プログラムを呼び出すと、外部オブジェクトはテキスト・ファイルと仮定され、gIBIS の標準検査ウィンドウにロードされる。一方、ユーザがディスプレイ・プログラムを指定すると、このプログラムが呼び出され、引数として外部パス名をパスする。この機能を利用すれば、外部データおよびプログラムを gIBIS にスムースに統合することができる。

(3) 観察所見

ここでは、IBIS 方式と gIBIS ツールの両方について、その長所・短所と研究上の問題点を述べる。

① ネットワークの構造

1987 年の半ばから 1988 年の半ばにかけての一年間の gIBIS ツールの利用状況を、ネットワークに関して統計を取った。32 人の人々が、全部で 33 の Issue の作成に参加した。1988 年 2 月現在で、作成されたノードの数は 2,091 個、すべての Issue グループがほぼ同数の Issue、Position、および、Argument をもっていた。

Issue ノードの 31% は Position をもっていないが、残りの Issue ノードは平均 1.9 個の Position をもっていた。一方、Position ノードの 59% は Argument ノードをもっていないが、残りの Position は平均 1.7 個の Argument をもっていた。

また、これらのノードを接続しているリンクの数は 2,214 個であった。リンクのタイプによる分類結果から、ユーザが反論 Argument よりも支持 Argument をより多く提示する傾向をもっていること、および、ノードを一般化するよりは特殊化する方が自然であることが示された。

さらに、この試験期間中の大きな Issue グループ 15 個における個人参加のレベルについての調査から、多くの Issue グループが主として 1 人の参加者によって構成され、参加者のバランスが取れている Issue グループは極くわずかであることが判明した。

② 明示修辞構造の有効性

幾つかの Issue グループに協力して参加していた何人かのユーザは、gIBIS ツールが議論に強制的に課す構造が非常に有効であり、「胸に秘めた一物や、密かな合図や、巧妙な修辞法」を白日の元に晒す役割を果すと報告してきた。彼らは、また、仮定や定義が明らかになる傾向も評価している。

これらの利点の幾つかは、IBIS ネットワークの半構造化された性質に、その源をたどることができる。この性質のお陰で、書き手は表現に何の制約も受けずに完全なメッセージを構造化することができたし、読み手は調査と総合的理解の両方を支援するテキストの反復構造を読み取ることができたのである。読み手と書き手の両方が、IBIS の明示修辞構造の恩恵を受けており、この構造は、少なくと

も、言葉による議論の一般的な構造を明らかにしている。

③　ツールと方式の相互促進作用

コンピュータ化されていないIBIS方式は扱いが厄介なので、高速gIBISツールによるサポートがなければ、本研究所で現在得ているような人気を博することはなかったと思われる。また、gIBISは本研究所の環境で使用できる唯一のハイパーテキスト・システムという訳でもないのに、短時間のうちに、より広く長時間使用されるようになった。これは、IBIS方式に対する要求とgIBISツールのハイパーテキスト機能の間に、特に良好な整合性が存在したためであると思われる。

④　未熟なセグメンテーションの危険

ハイパーテキスト・システムに一般的に見られる微妙な問題の一つは、個人の思考を独立した単位に分割するのがしばしば不自然なことである。特に、問題がよく理解されていないと、その思考は不明瞭で、混乱した、移ろいがちなものとなる。IBIS方式はユーザにノードタイプとリンクタイプの厳格な選択を強制するので、gIBISを使用するとこの傾向が顕著になる。特に、設計の会話は、「Xを試してみよう —Yの利点がある」といった形のコミットメントを特徴とすることが多い。これはPositionと、その支持Argumentであるが、PositionにIssueが結合されていない。ユーザの中には、IssueとPositionを即座に見分けることができない、アイディアを構造化する前に単に記録しておく「プロトノード」が欲しい、というような苦情を寄せた人もいる。

⑤　Issueの解決の把握

審議プロセスには、Issueの解決をどのように表現し、表示するかという、やや補足的な問題が存在する。IBIS方式は、「正しい回答」または少なくとも「我々が現在コミットしているPosition」としてIssueに応答するPositionの1つを選択することによって（選択の方法は問題でない）、Issueが解決されるとしている。このためには、Positionノードに「SELECTED」マークを付けて表現すればよいし、ブラウザ・ウィンドウの中でそのPositionノードをはっきり区別して表示すればよい（たとえば、選択されていないPositionとは異なるカラーで表示するなど）。

⑥　非直線的文書の文脈の問題

gIBISの実験の重要なエレメントの1つは、共同作業の媒体としてのハイパーメディアの利用性を調査することにある。数人のユーザが、Issueグループを共有して、共同で作業を行っている幾つかのケースで、予期せぬ問題が出現した。書き手が細心の注意を払って明瞭かつ完全に文を書かなかったために、読み手は、個々のノードを理解できたと感じたものの、数十ものノードにまたがって曲りくねっている書き手の思考の糸をたどることができなかったのである。つまり、ハイパーテキスト・ツールは、書き手のアイディアを、強制的に、細かく分割してそれぞれの粒子を独立して表現させるので、書き手の頭の中で展開しつつあるより大きなアイディアが曖昧になってしまう、ということである。

この問題の改善には、幾つかの技術が有効であろう。本論文執筆時点では、1セットのノードを結集

させる高レベルの構造の実験が進められている。これは、IBISサブツリーのすべてのノード（Issue、Position、Argument）の表示を単一のノードに結集させる新しいタイプのIPAノードで、このノードにIPA特定テキストを添付することもできる。この新しい構造は個々のIssueの議論を直線化するので、gIBISネットワークを読む時によく感じられるような分裂した感じが減少すると思われる。

⑦ 注釈的なまたは「上位（メタ）の」議論

たとえば、「しかし、それはここでは問題でない」など、「上位（メタ）」、つまり議論の（内容ではなく）プロセスにコメントを付けることは、人間の会話によく見られることである。同様に、IBISの議論においても、Issueグループの参加者たちが、仲間の誰かがIBIS構造を十分に使いこなしていないとか、アイディアの出し方が不正確であるなどと感じたときには、メタの議論が必要となることがある。たとえば、BがAのIssueノードの内容について、実は、これが2つのIssueと、一方のIssueに対するPositionであると感じたとすれば、Bは何等かの方法でこれを表現し、Issueグループの状況の中で、Aとこの「メタIssue」に関する議論を始める必要がある。

この問題を解決する方法は幾つかあるが、本研究では各ノードに独自の「メタ層」をもたせる方法を採用しており、ノードの内容がそのIBISタイプと一致しているかどうか、といった議論は、ノード内のこの特殊化された部分で追跡するようにし、リクエストに応じて可視表示できるようにしている。現行バージョンでは、どのユーザも任意のノードのボディ末尾に「メタ・ライン」を付加することができる。メタ・ラインを追加したら、コメントを入力し、サインすることによって、ノードのこの部分の中で、注釈的または手続き的議論を開始することができる。ノードの書き手は、ノードの末尾に応答を付加してもよいし、構造エラーを訂正するためにネットワークを修正してもよい。

⑧ ブラウザ・スペースのマクロ・レベル組織

ハイパーテキスト研究の重要なテーマの1つは、ネットワーク内のノード数が数十以上のケースで、グラフィカル・ブラウザを如何にして有効利用するかという問題である。現在の形のgIBISブラウザは、ノード・ビューイング・ウィンドウとコントロール・パネル・ウィンドウがスクリーンのスペースを共有しているので、スクリーンの半分以上を占めることはできない。この大きさのブラウザ・ウィンドウに一度に表示できるのは、せいぜい40個から50個のノードである。ブラウザには各ノードについてほんの一語か二語のごく短いラベルだけしか表示されないので、ノードに関する詳しい情報が欲しいときは、その上でマウスクリックして、ノード・ビューイング・ウィンドウの中でその内容を読まなければならないのである。

⑨ 展開を続けるネットワークの変化に対処することの重要性

すべてのデータベースは、そこに内包されるデータの変化を管理するためのメカニズムをもたなければならない。「Issue base」の本質は、古いマテリアルの方が正確で重要性も高いとか、いや、不正確でヒストリとしての関心しかあり得ないとか、あるいは、その中間であるとかいった、展開し続け

る議論のためのツールとなることにあるので、gIBIS のようなアプリケーションでは、変更が非常に大きな意味をもつ。

このため本システムでは、ネットワーク・マテリアルの年齢と関連性をシステマティックに指示するメカニズムを必要としている。たとえば、最近議論の対象となったノードや、更新されたノードを除いて、古いノードは黄色で表示するとか、端部をギザギザにして表示するとかいった具合に。先に、重要性、やま場、信頼性に関して示したメカニズムと同じように、年齢と関連性もまた、部分的にしか自動化できない主観的な尺度であると言えよう。変化を管理するもう 1 つのメカニズムは、言うまでもなく人間である。Issue ネットワークのサイズと重要性が増すにつれて、組織に Issue ベースの通用性と健全性を管理するスタッフがどうしても必要になってくる。

3.7.2 SIBYL

原文タイトル： SIBYL: A Tool for Managing Group Decision Rationale

翻訳タイトル： SIBYL：グループ意思決定原理の管理ツール

著書： Jintae Lee
(Center for Coordination Science, and Artificial Intelligence Laboratory, MIT, Cambridge, MA, USA)

出典： CSCW'90 Proceedings, October 1990, pp.79-92

本論文では、グループの意思決定支援システム、SIBYL について紹介されている。まず、SIBYL の基本となる動機について述べられ、次にSIBYL が意思決定過程を表現するために使う言語 DRL (Decision Representation Language) について簡単に述べ、SIBYL の模擬セッションでの用法が示される。次にSIBYL と同じような目的を持つ他のシステムとの比較が行われている。特に、設計原理を記録することを目標とするハイパーテキスト・システムである gIBIS と比較されている。

(1) 動機

SIBYL は、大きな 2 つの動機が基本となっている。知識の共用と、質的な側面からの意思決定支援である。

① 知識の共用

通常、意思決定を行うには、ある基準に沿って選択肢を評価する際に関係する知識の断片を集めたり関係づけたりということを行う。こうした知識は、いったん明白に表現されると、いくつかの方法によって、他の者と共用することができる。グループの意思決定では、特に表現された知識は、新た

な知識を取り込み、それを支持する主張、拒否する主張、条件付与などによってその知識を膨らませることができる。

また、意思決定プロセスの一部として表現された知識は、同じような意思決定を行う他のグループにとっても、有効であることが多い。過去の意思決定は、必要とする実際の情報だけでなく、ある意思決定にいたるまでの方法、違った目標にも到達しうる方法、選択肢の評価の対象となるべき属性について理解する際の手助けとなる。さらに、過去の意思決定は、選択したアプローチが成功したか否かという知識も提供してくれる。知識を共用する方法としては、1つのグループの中でも時を隔てることで行われることがある。意思決定がどのようになされたかという記録をドキュメントとして提供すれば、これが転じて公正化と学習の基本となる。

② 質的な意思決定支援

意思決定過程の質的構造を表現する言語を持つことができれば、そのシステムは、意思決定を支援する多くのサービスを提供することができる。たとえば、対象間の依存性をシステムで管理することができる。追加される知識の効果や不確実性について伝達することができる。そして、現在の意思決定に有効な知識の断片を過去の意思決定から取り出すことができる。

(2) SIBYL

SIBYLは、3つの部分からなる。(i) DRL (Decision Representation Language)、(ii) DRLで表現されたものを利用して質的な意思決定支援を行うサービス群、(iii) DRLをより簡単に利用できるようにするユーザ・インタフェースである。ここでは、SIBYLの模擬セッションという条件のもとでこれらを図示することとし、すなわち、ユーザ側に立って見たSIBYLについて説明する。

① DRL (Decision Representation Language)

DRLの語を形成する対象と関係について、以下に述べる。その図式を図3.7-4に示す。また図3.7-5は、意思決定グラフ、すなわち、DRL知識ベースをネットワークに表したもので、SIBYLの模擬セッションで使用することになる。

Alternative（選択肢）は、選ぶべきオプションを表わす。Goal（目標）は、理想的なオプションが持つべき特性を表す。Decision Problem（意思決定課題）は、目標を最適に満足させる選択肢を選ぶ問題を表す。各Alternativeは、Achieves（達成）という関係を通してGoalと関連づけられ、Achieves (Alternative, Goal) と表す。DRLでいう関係とは、Claim（主張）のサブクラスであり、特に、関係Achieves (A, G) は、選択肢Aが目標Gを達成するという主張を表す。選択肢の全体的な評価は、関係のもっともらしさ、すなわち、主張Is-the-Best-Alternative-For (Alternative, Decision Problem) で表わす。逆に、この関係のもっともらしさは、選択肢とすべての目標との間の関係Achievesのもっともらしさに比例する。これらの目標の重要性にも比例する。ひとつの選択肢は、その選択肢を各Goalsと結び

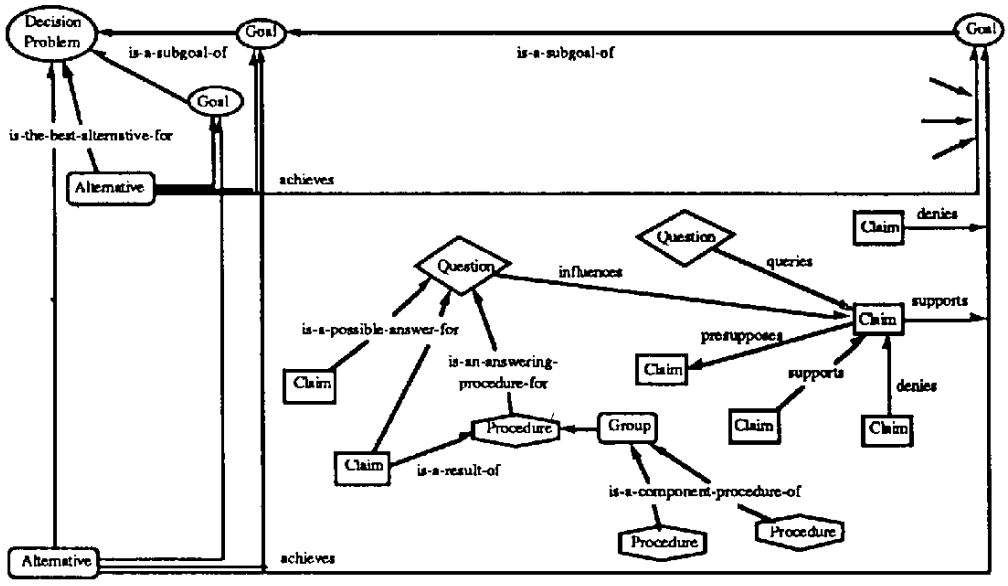


図3.7-4　DRL 用語

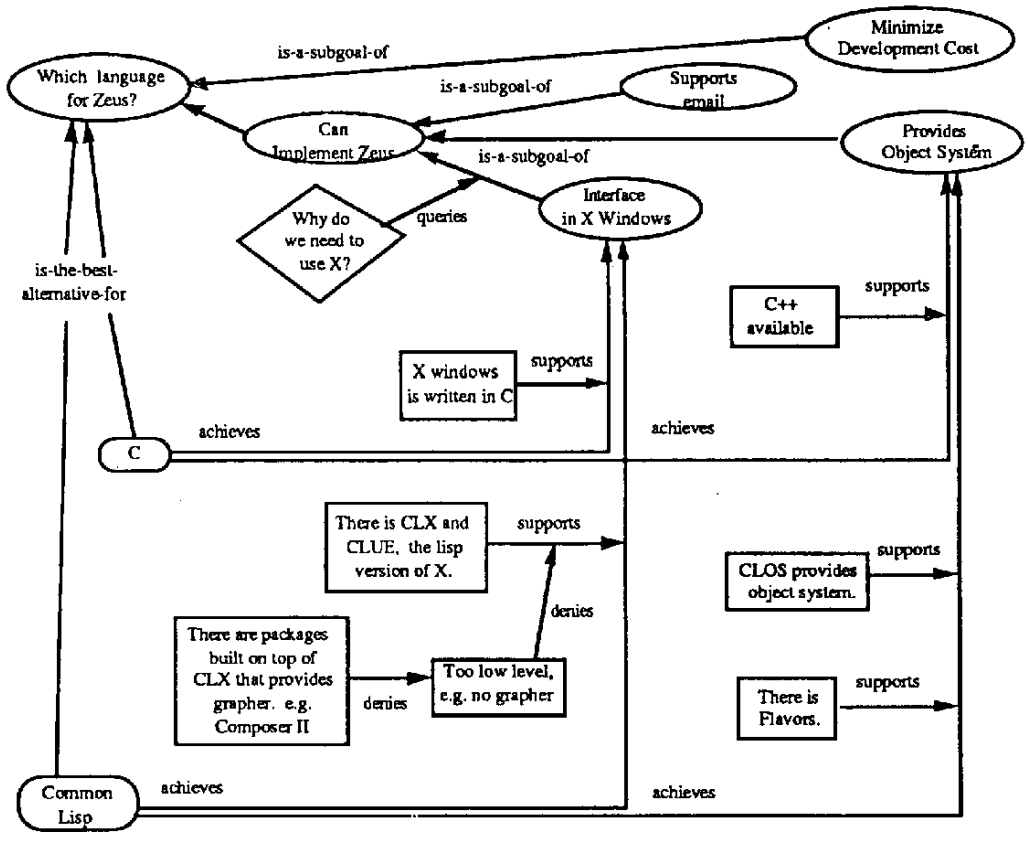


図3.7-5　意思決定グラフ例

付ける主張 Achieves のもっともらしさと、Goals の重要性について論じることによって評価される。分かりやすくいうと、Claim（主張）を作ることによって、他の Claims（主張）に対して Supports（賛成）、Denies（反対）、Presupposes（前提）を行う Claims を作り出し、DRL で議論するということである。これらの関係（Supports（賛成）、Denies（反対）、Presupposes（前提））は、すでに述べたように「主張」であり、それらもまた、議論の対象となるものである。あるClaim（主張）のもっともらしさが Qestion（疑問）にどう答えるかによるとすれば、その Question は Claim に影響を与える。

② SIBYL とのセッション

ここでは、SIBYL が利用する知識を、人間がどう与えるかについて説明する。この知識は、図3.7-5に示したような意思決定グラフで表される知識ベースを構成する。

(a) 課題

あるグループで、Zeus というシステムの開発に使用するプログラミング言語を何にするかを決定しようとしている。グループ・マネージャの Jane は、Decision Problem（意思決定課題）事例を作成することで意思決定過程を開始する。彼女は、画面の枠の中に全タイプを表示しているタイプ・ブラウザの中から、対象タイプの Decision Problem のところでマウスをクリックして作業を始める。新しい画面とメニュー項目を表示するテンプレート・エディタが呼び出されて、その対象について規定に従って実行することのできるアクションが示される。図3.7-6 には、そのエディタを示すが、これから述べる方法に従ってユーザがすでにいくつかのフィールドをエディットしてある。

(b) 当初の目標と選択肢の指定

Add An Alternative というオプションをクリックして、考慮すべき選択肢を表す Alternative という対象を作成する。ここでは、C と Common Lisp である。同様に、Add A Goal でクリックして、最適となる選択肢が持つべき属性を表す Goal という対象を作成する。ここでは「Zeus の開発が可能」「開発コストの最小化」となる。これらの目標と選択肢は、図3.7-6 に示すように、SIBYL が自動的に意思決定課題例と関係づける。次に、この意思決定課題に対する意思決定マトリクスを要求する。図3.7-7 に意思決定マトリクスを示すが、これはSIBYL とユーザとの間の主要なインタフェースである。いちばん上に目標を示し、選択肢は、左端の列に示される。各セルの値は、そのセルの目標に関して、その選択肢の現在の評価を表す。はじめは、各セルとも unevaluated と表示される。選択肢に対して賛否両論の主張を作成するに従って、セルの値は更新される。

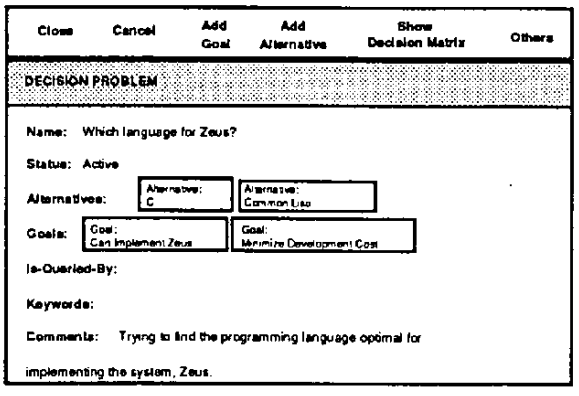


図3.7-6 意思決定課題の例

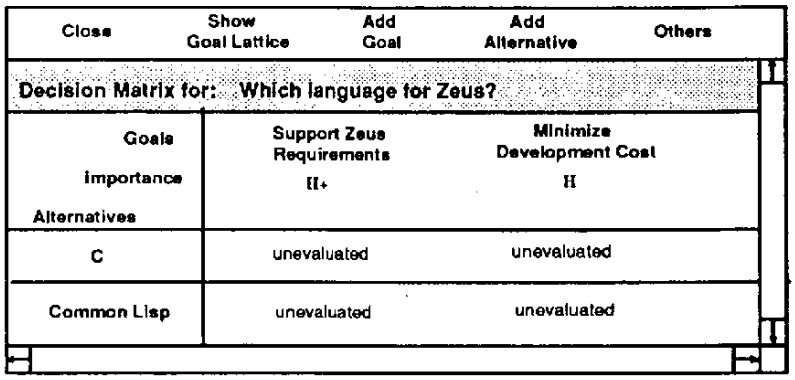


図3.7-7　意思決定マトリクス

(c)　グループ・メンバからの入力の受付

Jane は、意思決定マトリクスができたことをグループのメンバに告げて、各人の入力を要請する。グループのメンバは、目標に関する選択肢を評価したり、既存の評価に対する疑問を提示したり、新たな目標や不要な目標を指摘したりすることによって意思決定に寄与する。

(d)　目標の練り上げ

John は、Zeus のチーフ設計者で、この知識ベースに追加を行う最初の人物である。彼は、Lisp の強力な信奉者である。Zeus はオブジェクト・システムが必要であるため開発には Lisp が良いと思っている。また、Zeus のウィンドウ・システムは X ウィンドウ・システムで書かれなければならないと思っている。X が C で書かれていることは知っているものの、X のライブラリおよびツールキットの Lisp 版である CLX と CLUE があるので Lisp でうまく行くだろうと考えている。Lisp 支持を表すために、John はまず意思決定マトリクスを検討し、「Zeus の要件を支援する」という目標をいくつかのサブ目標に分けることにする。「Zeus の要件を支援する」と書かれたセルでマウスをクリックし、ポップアップ・メニューから Add A Subgoal を選択し、「オブジェクト・システムの提供」と「X ウィンドウでのインタフェース」という目標を 2 つ追加する。同じように、Zeus には電子メールが必要であると同時に自分がいづれかの選択肢を支持するような特定の要素をもっていないということがわかっているので、もう一つ「電子メールのサポート」をサブ目標として作成する。これらのサブ目標は、Zeus の開発要件として John が考えていることを表す。こうすれば、例えば、Zeus と似たシステムの要件を考える必要が出てきた場合など、これらの考えを検討したり、再利用することができる。

(e)　論拠の反論の追加

John は、Common Lisp と「オブジェクト・システムの提供」と「X ウィンドウでのインタフェース」という目標とを関係づける意思決定マトリクス内のセルでマウスをクリックし、ポップアップ・メニューから Show Argument Browser を選択して、自分の論拠を追加する。Common Lisp と「X ウィンドウでのインタフェース」という目標に関するアーギュメント・ブラウザを図3.7-8 に示す。はじめは、そ

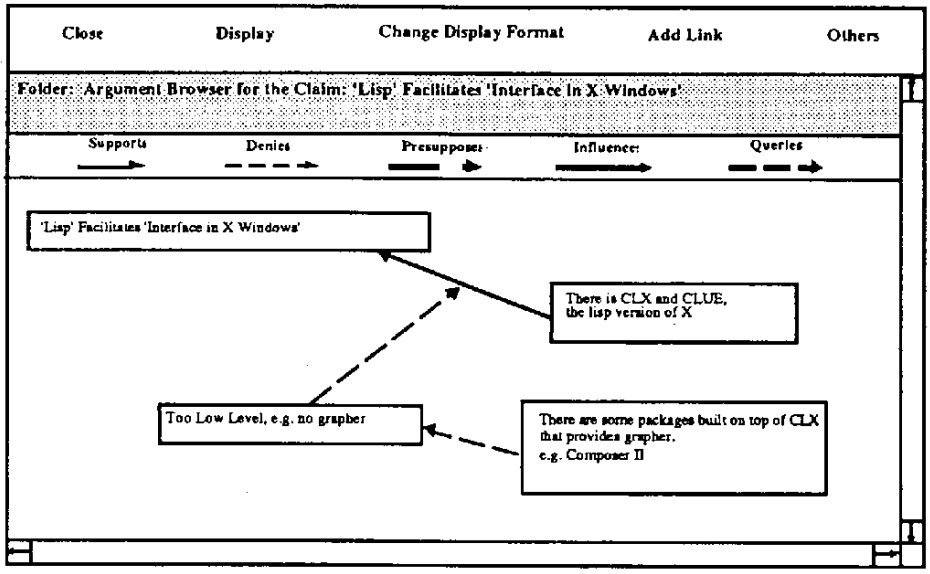


図3.7-8　アーギュメント・ブラウザ

のセルに関する選択肢が目標を達成するという主張が1つあるだけで、ブラウザのなかではいちばん上の主張である。すでに述べたように、DRLのなかでは、関係はClaim（主張）のサブクラスである。特に、選択肢Aと目標Gをリンクする Achives（達成）という関係は、Aは目標Gを達成するという主張である。この主張が妥当か否かという点については、最初はそのどちらでもない。この主張は、各選択肢について目標が作成されるごとにシステムが自動的に生成し、意思決定マトリクス内の各セルごとのアーギュメント・ブラウザには、はじめにたたき台としてこの主張が入る。

各人は、このきっかけとなる主張を支持または拒否する主張として、賛成または反対の論拠を表現する。すなわち、その主張のもっともらしさは支持または拒否の主張が追加されるに従って更新され、目標に対してその選択肢がどう働くかというものさしを表わす。SIBYLでは、ユーザは対象上でマウスをクリックすることで、その対象について実行できる全操作のメニューを出すことができる。Johnは最初の主張Achivesでマウスをクリックし、詳細（In particular）を出す。ある主張ノード上でマウスをクリックすれば、Add A Supporting Claim（賛成意見の追加）、Add A Denying Claim（反対意見の追加）、Add A Question（疑問の追加）、Specify A Presupposition（前提条件の特定）といった可能な操作メニューを表示することができる。ここで、Add A Supporting Claimを選択すると、新しいClaim（主張）のためのテンプレート・エディタが表示されて、この新しい主張はSupports（賛成）という関係で元の主張と自動的にリンクされる。JohnがAdd A Supporting Claimを選択すると、SIBYLは新しい主張のためのテンプレート・エディタを表示し、新しい主張と元の主張とをSupport（賛成）という関係でリンクする。

このようにして、何人かのユーザがそれぞれの知識を入れた後、Johnは、また意思決定マトリクス

を検討する。更新されたマトリクスを図3.7-9 に示す。意思決定マトリクスのセルのうちunresolved となっているのは、そのセルの目標については、その選択肢の評価を出すには解決すべき問題がいくつかあるということを意味する。H、M、L というのは、それぞれ High、Medium、Low を意味し、そのセルの目標についてその選択肢がどう働くかというものさしを表す。この値は、中心となる意思決定者が与えることもできるし、賛成・反対の主張のもっともらしさから導き出すこともできる。John は、自分が最後に SIBYL を使ってから追加された主張のすべてを表示するように SIBYL に要求する。SIBYL は表形式または、ネットワーク形式でこれを表示することができる。また、作成者、作成日、関係する主張などそこにあるフィールドのどれを使ってでもソートすることができる。このようにして今ある知識ベースを検討したあと、その間にうかんできた疑問に答えるか、先に進んでいる主張にある程度対応したり、新しく得た知識を追加したりすることにする。その結果を示す知識ベースを描いた意思決定グラフが図3.7-5 に示したものである。

Close	Show Goal Lattice	Add Goal	Add Alternative	Others
Decision Matrix for: Which language for Zeus?				
Goals importance Alternatives	Support Zeus Requirements H+	Minimize Development Cost H	Interface in X Windows M	Provides ObjectSystem H
C	unresolved	L	H	unresolved
Common Lisp	H	H	M	H

図3.7-9　更新された意思決定マトリクス

(3)　関連する研究との比較

最近、意思決定原理や論拠を表すという同様の目的を持った研究がいくつか出てきた。Argnoter は、なかでも早くに出てきたもののひとつであるが、立論の構造を明確にし、その構造における依存性の管理を補佐することでグループ意思決定を助けようとしている点で、SIBYL の考え方にもっとも近いものである。しかし残念ながら、その機能を理解するためにはどのような表現構造を用いるべきかが明確でないなど、Argnoter は上位レベルの記述に留まっている。SIBYL は Argnoter の背後にある概念をもっと進めて表現し、かつ、具体化する 1 つの試みとしてみることができる。また、SYNVIEW は、ネットワーク上に分散したユーザ間での議論を表現しようとするもののなかで、最も初期のもののひとつである。しかし、そのために、その表現が厳密に制限されており、インデントがテキストにあると、時には選択肢となったり、証拠関係となったりするのである。SYNVIEW は、合意の物差しを提供しようとしている点では興味深く、これはグループ意思決定の重要な調査項目である。

Potts と Bruns は、設計歴を設計の産物の交代、理論的解釈のセットとして捕捉すべきものとし、逆に、それぞれは IBIS 構造によって捕捉されると提唱した。彼らは、この構造を使ってソフトウェア設計の例を表した。SIBYL の DRL 構造は、理論的解釈を表すもう 1 つの構造とみなすことができる。

gIBIS (graphical Issue Based Information System) は、「調査方針議論のためのハイパーテキスト・ツール」である。gIBIS の目標は、設計原理、すなわち、「設計の問題点、代替解決案、これらの代替案間のトレードオフ分析、意思決定過程において行われた仮または正式の言質」の捕捉にある。gIBIS は良く知られ、その目標とするところも SIBYL とまったく同じであり、gIBIS が使う IBIS 構造は、他の多くのシステムで採用されている。SIBYL と gIBIS とは、設計や意思決定の過程で蓄積する知識を表現し、見直しや再利用を可能にするという共通の目的を持つ。SIBYL、gIBIS のどちらも、議論を通して選択肢を評価することを伴うある種の課題、すなわち、意思決定に共通の対象と関係をあらわす用語を持つ言語を提供することで、この目的を達成している。

SIBYL は gIBIS を拡張したものと考える。gIBIS も SIBYL も、構文上の型（テキスト・ノード、イメージ・ノード等）だけを持つ単純なハイパーテキスト・システムと、すべての知識がシステムによる自動処理のために形式に従って表現される完全な形の知識ベースとの中道をとっているのであるが、SIBYL は、gIBIS に比べ、知識ベース寄りである。gIBIS は主にハイパーテキスト・システムであり、そのサービスの焦点が構造の表現におかれており、何を表現しているかという構造を理解し、新たな知識を加えることが簡単にできるようにすることを目標としている。一方、SIBYL は、主として知識ベースのシステムで、セミフォーマルな表現を使うので、知識のすべてを形式化することを強制しない。SIBYL が提供するサービスは、いわば、知識ベース・システムに一般的な、依存性管理、不確実性管理、ビューポイント管理、優先度管理である。

SIBYL と gIBIS が提供使用としているサービスの違いは、互いの持つ用語の違いで説明できるかもしれない。gIBIS でいう Issue は、DRL では Decision Problem といい、同様に Position は Alternative、Argumentは Claim という、しかし、gIBIS に欠如しているという点で特に目だつのが、選択肢が評価されるべき目標や目的という概念である。gIBIS では、対象は暗黙のうちに表されるだけである。例えば、「Lispを使用する」という立場を「CLOS があるので Lisp はオブジェクト・システムを提供する」という論拠を使って支持することができる。この関係から、オブジェクト・システムの提供が目的なのだということは、おそらく類推されるだろうが、なぜそれが目的なのかとか、他の目的とどう関わっているのかは明確ではない。さらに悪い点は、中間的な目的については論拠の中でまったく触れられないことが多いということである。例えば「C には C++ がある」という論拠が「C を使用する」という立場を支持する場合を考えてみると、この場合オブジェクト・システムを持つという目的は明らかにされていないので、それについて論じることは難しい。また、「オブジェクト・システムの提供」という目標を明確に表わさないのでは、片や CLOS に言及し、他方が C++ に言及している前出の 2 つ

の論拠は、選択肢の同じ属性についての比較を行っているのだということがはっきりしない。こうした理由から、少なくともある程度、人々を合意へ導く手助けをするに当たってのgIBISの難しさが説明できる。

それゆえ、目標を明白に表現することの利点は多い。そうすることで、考えを表現し、選択肢を評価すべき対象がわかるようになる。明白に表現することは、また、その目的について議論したり、必要に応じてそれを変更したりできるということである。目標を変えようとするなら、仮に「オブジェクト・システムの提供」は「Zeusの開発が可能」という目標のサブ目標としてはもはや重要ではないとすれば、SIBYLでは、そのサブ目標を削除する（または、その重要度をゼロにする）だけでよい。そうすれば、それに関連するすべての主張が自動的に消されるが、gIBISではこうした変更を受け入れるモジュラーがない。

目標は、SIBYLのサービスすべての中で重要な役割を果たしている。目標を明白に表現することによって、その目標が変更されたとき、依存性管理機能が知識ベースを更新できるようになっている。例えば、このシステムを教育的なものにするという目標の重要度がかなり高くなるとすると、Pascalが選択肢の一つとして登場する。Is-the-Best-Alternative-Forという関係のもっともらしさは、その目標の重要度と、ある選択肢をその目標のそれぞれと結びつけているAchives関係のもっともらしさとに比例している。gIBISでは、なにが目的であるかを語ることができないので、それぞれの重要度の違いもそのままである。

その他にも、関係というものが、議論の対象となり得る上位クラスの対象であるかどうか等、SIBYLとgIBISの構造的違いはいくつかある。たとえば、gIBISでは、「AとBには賛成であるが、AがBを支持するということには賛成できない」とか、「Gを目標にすべきであるとは思わない」などということはできない。全体的にみて、DRLの構造はより表現力に富み、gIBISが人間の使い勝手の良さに主として動機を持つの対し、SIBYLは提供するサービスに動機づけられているように思われる。

3.8 ツールキット

3.8.1 LIZA

原文タイトル： LIZA: An Extensible Groupware Toolkit

翻訳タイトル： LIZA：拡張可能グループウェア・ツールキット

著書： S.J. Gibbs
(Software Technology Program, MCC, Austin, Texas, USA)

出典： CHI'89 Proceedings, May 1989, pp.29-35

本論文では、まずグループウェアの定義が行われた後、「アクティブ・オブジェクト」に基づくグループ・ツールのモデルが提示される。このモデルは、LIZAとして知られる拡張可能グループウェア・ツールキットの設計と実現に使われている。マルチ・ユーザのインタフェースを構築するシステムであるLIZAについて紹介され、その後、LIZAで構築したグループ・ツールの例について述べられている。

(1) グループウェアの定義

グループウェアと言えば、ユーザーがグループである場合のソフトウェアを指すが、ここではさらに特定して次のように定義する。

「共通のタスクについて（同時に）作業する2人以上のユーザをサポートし、共有環境にインタフェースを提供するソフトウェア・システム」

この定義においては、「共有環境」の概念が重要である。この場合共有環境とは、1人のユーザの行動がもう1人のユーザに見えることを意味する（例えば通知メカニズム等によって）。わかりやすい例が、共同執筆編集システムや、マルチ・プレーヤ・ゲームである。データベースのようなものは、多くのユーザが1つのデータベースにアクセスして1つのタスクについて作業を行うが、ユーザが他のユーザの行動を知らない（また実際にデータベース・システムはそれを保証するようになっている）ため、グループウェアではない。

本研究では、リアルタイムのグループ・ツールに焦点を当ててLIZAの開発を行った。これによってマルチ・ユーザ・インタフェース設計の問題（すなわち、グループ・オペレーション、WYSIWIS、社会的プロトコル対ソフトウェア・プロトコルなどの問題）を明らかにできることを望んでいる。

(2) LIZAツールキット

LIZAは、LAN上に接続されたハイパワーのパーソナル・ワークステーションで実行される。「セッション」とは、LIZAを使う参加者グループのことで、分散していたり（参加者がそれぞれのオフィスにいる）、集中していたり（参加者が同じ場所にいる）、この2つの組合せであったりする。会議開催コールを行って、物理的に離れた参加者間に聴覚チャネルを設定する（スピーカホンかマイク付きヘッドホンを使う）。集中型参加者は、特別な電子会議室を使うことができる。この会議室には、多数のワークステーションと、ワークステーション・ディスプレイを背面映写する大型（5インチ×7インチ）スクリーンが設置される。

① アクティブ・オブジェクト

LIZAのツールは協同作業を行う1組の「アクティブ・オブジェクト」である。アクティブ・オブジェクトとは、オブジェクト指向のプログラミング・パラダイムを拡張してオブジェクトの同時実行を

可能にしたものである。アクティブ・オブジェクトは、Hewitt のアクタ理論や、Emerald およびKnos 等、アクティブ・オブジェクトをサポートする最近の言語と密接な関係がある。

分散制御や並列処理や分散知能が必要な場合、アプリケーション内にアクティブ・オブジェクトを採用する。例えば、鳥や魚の群のような動物グループの動きをモデル化するのには、アクタ・ベースの言語が使われている。集合体の各メンバはアクタによって表され、その行動によって集合体の動きのアニメーションを駆動する。Knos は、「メッセンジャ」オブジェクトの指定に使われ、行き先を探索して電子メールを配達するのを補助している。Knos は、ネゴシエーションを行うオブジェクトの行動を指定するのにも使われている 。

LIZA のグループ・ツールによって、グループ・メンバが同時に作業できるようになる。これにはシステムの最も基本的なレベルに同時性を導入しているため、LIZA の構築にはアクティブ・オブジェクトが適しているように思われる。

② ツール構造

LIZA では、各グループ・ツールは「カーネル・オブジェクト」と呼ばれる 1 つのアクティブ・オブジェクトと、「イメージ・オブジェクト」と呼ばれる多数のアクティブ・オブジェクトからなる。イメージ・オブジェクトは、ツールのユーザ・インタフェース部分を担当し、一般にツールを使う参加者 1 人につき 1 つのイメージ・オブジェクトがある。参加者が有意な行動（ボタンを押したり、メニュを選択したり）を取ると、イメージ・オブジェクトが「イベント」の信号を送り、カーネル・オブジェクトがこれを捕まえる。カーネル・オブジェクトは、イメージ・オブジェクトから入って来るイベントに対する対応を処理、調整する。カーネル・オブジェクトは、1 つ以上のイメージ・オブジェクトに対して「ディスプレイ指令」を送り返して対応することもある。

グループ・ツールを形成するオブジェクトは、1 つのツール・クラスのインスタンスである。ツール・クラスの一般的な構造を描くのにテンプレートを使うこともできる（図3.8-1 参照）。図のテンプレートの最初の 2 つの部分には、クラスの名称とその上位クラスを示す。グループ・ツールの全てのクラスには、上位クラスとして group-tool-mixin が含まれているため、ツール・クラスは、この mixin がイメージ・オブジェクトを生成／抹消する方法（参加者がセッションに入ったり抜けたりする場合に必要）とその他の記帳機能を継承している。テンプレートの 3 番目の部分ではクラスのインスタンスが保持する変数、残りの部分ではツールの行動を指定している。述語（メソッド）は 4 タイプに分け、最初の 2 つはカーネル・オブジェクトがイメージ・オブジェクトから送られたイベントの処理と、イメージ・オブジェクトが核の送ったディスプレイ指令処理に使う述語で、この 2 つは、核ーイメージ間のインタフェースを形成する。次の 2 つは、ツールが他ツールに提供するパブリック述語と、ツール内部で使うプライベート述語である。

③ システム構造

LIZA は、多数の「クライアント」と1つの「サーバ」からなる。イメージ・オブジェクトはクライアント上で動作し、サーバはカーネル・オブジェクトを動作させる。サーバとクライアントの間の連絡は、Unix のソケット・メカニズムを介して行われる。全体的なアーキテクチャを図3.8-2 に示す。

class name	vote-tool	
superclasses	group-tool-mixin	
instance variables	list of issues, array of votes on issues	
kernel-image interface	event handling methods	display methods
	StartVote StopVote CastVote ScoreWinPaintreq	MapScoreWin UnmapScoreWin PaintScoreWin
public methods	private methods	
ReadVote	none	

図3.8-1 投票ツールのクラス

このようなアーキテクチャを選んだのには、単純さと即応性という2つの理由がある。集中型サーバでグローバルな情報を維持することで、同時制御と通知が容易になる。一方クライアントも多くのユーザ・インタフェース・オペレーション（ウィンドウ再表示やメニュー表示等）を自動的に実行できるので、即応性が失われることがない。

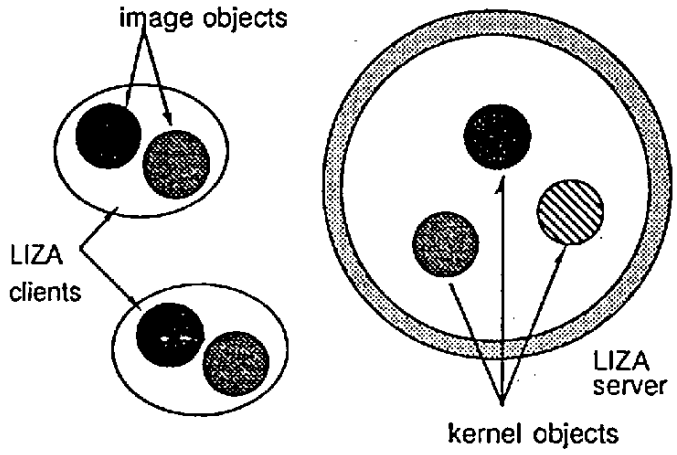


図3.8-2 LIZA のアーキテクチャ

LIZA は、サーバとクライアントに新しいツール・オブジェクトを追加することで拡張可能である。例えば現在の LIZA システムには8個のツールがあり、それぞれが既存システムに接続されたものである。もう1つの拡張方法としては、イメージ・オブジェクトからカーネル・オブジェクトへの通信にメッセージではなくイベントを使うことが挙げられる。イメージ・オブジェクトがイベントの信号を送ると、そのイベントは、関心を示したカーネル・オブジェクトの処理述語で処理されることになる。

(3) グループ・ツール

LIZA の8個のツールの内の4ツールの例と投票ツールについて説明する。

① ツールの例

図3.8-3 に、LIZA のツールの内、次の4個を示す。

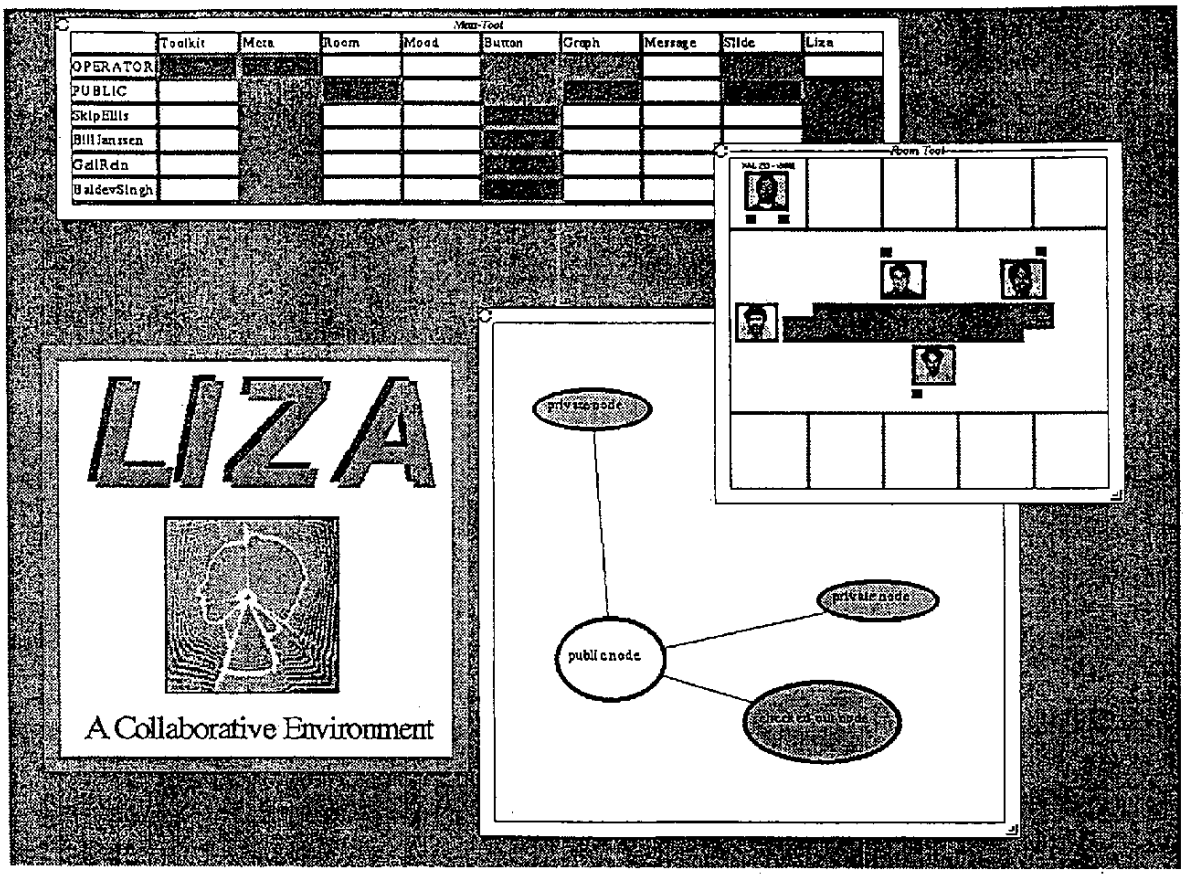


図3.8-3　LIZA のメタ・ツール、ルーム・ツール、グラフ・ツール。スライド・ツール
（上から時計回り順に）

(a)　ルーム・ツール

ルーム・ツールは、セッション参加者の位置マップを示す。参加者は、それぞれオフィスや会議室にいる。オフィスのレイアウトは物理的な位置には対応していないが、各オフィスには部屋番号や電話番号が付いている。会議室内では、参加者は会議テーブルの物理的位置に応じて配置されている。

(b)　グラフ・ツール

グラフ・ツールを使って、ユーザ・グループは共有グラフを編集することができる。このツールは、Cognoter に似ている。オペレーションとしては、生成、削除、ノードのリンクやリンク解除、ノード内容の更新やノードの移動、コピー、サイズ変更がある。ノードはそれぞれ、プライベート、パブリック又はチェック・アウトの３つの状態のいずれかになっており、オペレーションでノード状態を変更する。

(c)　スライド・ツール

スライド・ツールを使って、参加者は一連の「スライド」イメージをロードしたり複数のスライドを流し読みしたりできる。このツールは、スライド選択用ボタンのあるウィンドウと、スライドを表示する「エピソード・ウィンドウ」の2つのウィンドウからなる。参加者が特定のスライドを選ぶと、全参加者のエピソード・ウィンドウにそれが表示される。

(d) メタ・ツール

このツールは「操作盤」を表し、横の各段が参加者、縦の各列がツールに対応している。項目はマウスで選択することが可能で、特定のユーザがツールをスタートさせたりストップさせたりするのに使える。これは、一部のメンバがシステムをよく知らない場合、経験のあるユーザが手助けできるため、有益である。

② 投票ツールとこれに伴う社会的問題

ここでは、LIZAを使って投票ツールを設計する場合を考えてみる。参加者の1人が投票ツールを起動すると、「投票ウィンドウ」と呼ばれるウィンドウが全員のディスプレイに現れる。投票ウィンドウには、投票すべき問題を入力するタイプイン領域と、「投票開始」、「投票終了」の 2つのボタンがある。前者を押すと、参加者全員のディスプレイに「スコア・ウィンドウ」が現れ、後者を押すとこれがウィンドウから消える。スコア・ウィンドウには投票する問題が表示され、「はい」、「いいえ」、「どちらともいえない」、「棄権」から1つを選んでマウスで入力できるようになっている。ユーザの現在の選択が強調表示される。スコア・ウィンドウはグループの投票状況を示す棒グラフも表示する。投票開始時は全投票者が「棄権」であるが、投票が進むにつれて棒グラフが変っていく。図3.8-4に、ウィンドウの表示を示す。

この投票ツールは非常に単純なもので、LIZA内には数ページのコードしか必要としないが、グループ・ツールの設計者が、ツールの利用に影響する社会的組織的問題をいかに認識していなければならないかをよく表している。投票ツールの設計の際に見過ごされた問題には次のようなものがある。

(a) 途中結果

この投票ツールは、参加者が進行中の投票結果を見ることができるため、投票に影響を与えることがある。ツールを使用すべき背景を理解していないと、このような影響をそのままにしておくべきか、全員が投票を終わるまで合計がわからないようにする「目隠し」投票にすべきかどうかがわからない。

(b) 匿名性

スコア・ウィンドウは匿名性を保持し、自分のスコア・ウィンドウを見ているユーザが他ユーザの投票を決定することができない。しかし、他ユーザのウィンドウを盗み見したり、他の参加者によるマウスのクリック音をスコア・ウィンドウでの変更と結びつけたりして見当をつけることはできる。

(c) 投票提案

この投票ツールでは、どのユーザも投票のための議題を提起することができる。これは、それぞれ

の地位が同等のグループには適しているかもしれないが、民主主義をこれほど高度に発達させていないグループも多い。また、同等のグループの中でも、ある問題に関する投票を強制されることを好まない参加者がいることが考えられる。そのため、誰が投票議題を提案できるか、いつこれを行うかは、投票ツール設計者が考慮しなければならない問題である。

(d) 投票変更と棄権

この投票ツールでは、ユーザは投票を変更したり取下げることさえできる。このツールを、コンセンサスを得るための状況に使うのであれば、変更も有益だが、得票集計に使う場合は、変更によって結果がゆがめられることがあるので、望ましくない。

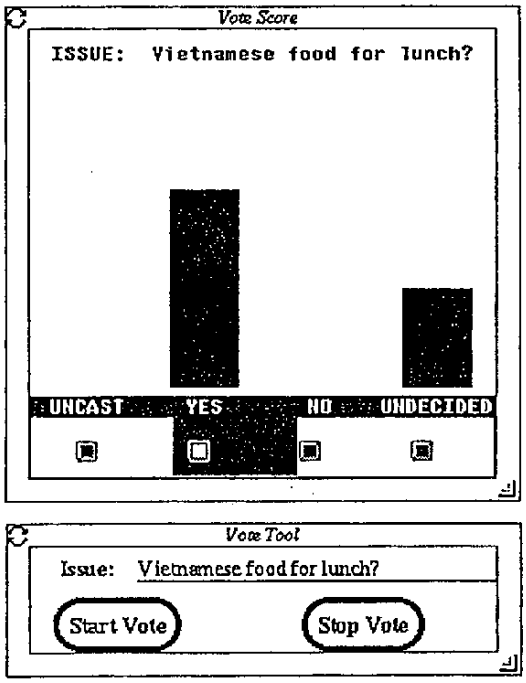


図3.8-4 投票ウィンドウとスコア・ウィンドウ

社会的要因に対する感受性では投票ツールの例はやや極端だが、一般的にグループ・ツールには、程度の差はあっても同様な問題が起こることが予想される。社会的組織的要因は、グループ・ツールの容認に重要な役割を果すため、まずLIZAのようなシステムをグループ・ツールのプロトタイプ化に使って、実験的にグループによる受入れを評価していくアプローチが考えられる。

3.8.2 Strudel

原文タイトル： Strudel - An Extensible Electronic Conversation Toolkit

翻訳タイトル： Strudel — 拡張性ある電子会話ツールキット

著書： Allan Shepherd
(Hewlett-Packard Laboratories, Palo Alto, CA, USA)
Niels Mayer
(Hewlett-Packard Laboratories, Palo Alto, CA, USA)
Allan Kuchinsky
(Hewlett-Packard Laboratories, Palo Alto, CA, USA)

出典： CSCW'90 Proceedings, October 1990, pp.93-104

Strudelは、会話と行動の管理のための支援システム構築用部品群からなるツールキットである。本

論文では、まずStrudelの概要が述べられた後、利用シナリオ例に基づいてStrudelの概念モデルが説明される。最後に、アクティブ・メッセージ・システム相互間の相互運用可能性等の問題についても手短に説明されている。

(1) Strudelの技術的アプローチの概要

Strudelは、エンド・ユーザが電子メールに基づく会話と行動項目を管理するための部品からなるツールキットを提供する。ここでは、ツールキットの部品の設計について現在のプロトタイプの例とともに説明する。このツールキットにおいては、標準化、ツールキット部品のユーザによる拡張可能性、既存のアプリケーションとの相互運用可能性、および広範囲に普及したプラットフォーム上での実行時性能の良さ等に重点が置かれている。Strudelは、既存システムとコンパチブルである。例えば、既存の電子メールのユーザ・エージェントと同様のルック・アンド・フィールを持つ。その目的は、UNIX、X Windowシステム、およびARPA Internetメール等の標準的ソフトウェアが動作する相対的に低価格なグラフィック・ワークステーションの上で、グループウェア・アプリケーションを構築可能な、小型で高速でポータブルな、C言語によってインプリメントされたプラットフォームを構築することである。さらにStrudelは、OSF/Motif UI Toolkitの提供するグラフィカル・インタフェースを利用することによって、ユーザに受け入れやすくする配慮がなされている。カスタマイジングを支援するために、Strudelでは、Winterpに基づく拡張言語が提供されている。

Strudelは、会話、会話手番、行動、行動項目、通知等に対するユーザによるカスタマイズ可能な定義を含む部品ライブラリを持つ。プレゼンタによってこれらのオブジェクトの表示、編集、探索が可能である。

既存の研究は、到着する大量の電子メールをメッセージ・フィルタ用のルールによってメッセージ・クラスに分類し、ブラウザによって表示し、さらに、「Xからのすべてのメッセージを Yへ転送せよ」など、あるクラスに属するメッセージに適用する行動を指定できるようにしたものであった。Strudelでは、この研究をさらに拡張し、ユーザによる拡張可能なメッセージ・タイプおよび会話部品の管理が可能なライブラリ機能を提供することによって、ユーザがメッセージを作成中にメッセージへ構造を追加することを可能にしている。

また、Strudelにおいては、メッセージはwinogradの「行為のための会話」におけるRequestのようなタイプ化された会話手番(conversational moves)を含めることができる。ここでは、メッセージ「タイプ」ではなく「会話手番」という用語を使うが、これは、1つのメッセージには複数の「手番」が含まれることがあり得るということと、それぞれの手番は会話の相手によって典型的に取られるであろう次の手番を示唆するからである。半構造化されたメッセージの場合のように、手番はフィールドを持つことができる。例えばRequest Meetingという手番は、日付、場所および議題のフィールドを持

つ。手番は行動を含むことがある。例えばRequest Meetingという手番は、Add to To-Do Listという行動を含み、これはその手番のフィールドをアクセスするかも知れない。ユーザはトップレベルのメニュー、ボタン、あるいは以前のメッセージに表示されたメニューや項目リストから手番を選択することによってメッセージを作成する。

メッセージが集まると、連続した会話手番の間がつながれ、会話が構成される。会話の中で、従来の電子メールのメッセージとStrudelのメッセージを自由につなげることができる。従来の電子メールのメッセージは、既存のIn-reply-toや、Subjectフィールドのエントリを利用して、可能な限り前のメッセージにトレースされる。

Strudelが従来のアプローチと異なっているのは、ユーザがダイナミックに会話手番と会話タイプ定義を発展させることができる点である。したがって、これらの定義を特定のタスクに合わせた特別のものとすることが可能である。例えば、ユーザは、Ask who is responsible for repairing a medical instrument defectという新しい会話手番タイプを作成することができる。ユーザはその後、この手番をMedical Instrument Defect Repairという会話タイプの最初の手番として追加することができる。しかし、現在のところ、新しい手番定義や会話を中央集中化されたライブラリに統合することは殆どサポートされていない。Strudelは、エンド・ユーザが、会話とタスクにおいて、次の手番を選択し作成することを支援するための単純なスクリプトのみをサポートしている。

メッセージやその他のオブジェクトによってサポートされている行動も、特定のタスクに合わせた特別のものとすることが可能である。他のツールに対するインタフェースもサポートされており、会話の中で指定された行動とタスク・オブジェクトは、他のツールによって管理されたオブジェクトへの操作を開始することができる。特に、会議室予約システムなどの汎用アプリケーション、および、ソフトウェア保守や欠陥追跡システムなどの特定領域の協調ツールへの簡単なプログラム・インタフェースを定義することができる。例えばRequest meetingメッセージは、Make room reservationという行動を持つことができる。この行動は、会議室予約アプリケーションの手番を起動することができる。

ユーザがメッセージ・システムを自分のタスクと結合できるようにするために、Strudelではユーザは行動項目(action item)を作成することができる。行動項目とは、彼らが実行したいと考える活動、およびその進行状態を記述するためにユーザが作成するメモである。例えば、ソフトウェア開発技術者は、欠陥修理の予定、その優先順位等を記入した行動項目を作成するかもしれない。さらに、Strudelユーザに外部ツールで発生したイベントについての情報を知らせるため、そのツールからの呼出しによって通知(notification)と呼ばれる特別の行動項目を作成することができる。会話の中におけるメッセージと同様に、行動項目は、半構造化されタイプを持つフォームとして表現され、タスクの中に結び付けられる。行動項目フォームは、行動とは別のものである。

Strudelの会話タイプは、その次に行われる手番のタイプを制約するものではない。したがって、会

話の任意の時点で、異なった会話タイプを自由に開始することができる。例えば、欠陥の解決を話題とする会話において受け取った Defect notification への返事として、Request meeting や Post design issue などの手番を送ることが可能である。

(2) Strudel 利用のシナリオ例

これまでプロトタイプ開発されたシナリオをサポートするために、いくつかの会話タイプがライブラリに定義された。これらのタイプには、設計問題の討論に利用される会話型の IBIS、「行為のための会話」、会議スケジューリング、ソフトウェア欠陥追跡および修理などがある。

ここでは、欠陥修復のシナリオを例として用いる。このシナリオにおいては、ソフトウェア開発システムが Strudel に製品組立てにおいて欠陥が発見されたと通知する。その開発プロセスに責任を持つソフトウェア技術者は、Strudel を利用して、ペンディングになっている通知レポートを読む。欠陥通知は、グラフィカルなフォームとして表示される。フォームの上のボタンによって、ソフトウェア技術者は、特定のタイプのメッセージを作成し、送ることができる。例えば、いく人かの開発技術者に、その欠陥の処理に責任のある人は誰であるか知っているかと問い合わせるメッセージなどが考えられる。1 人の開発技術者が、その欠陥は自分のものだと認めるメッセージを返して来る。また、この開発技術者は、メッセージ・フォームの上の別のボタンを押し、修理タスクのための最初の行動項目フォームを作成することによって、修理プロセスを開始する。後になって、この開発技術者は、報告された欠陥に関した設計上の問題点を提起するメッセージを他の技術者たちに通知することによって、関係した会話を開始する。行動項目フォームから、技術者は、ソフトウェア開発システムからの関連した欠陥報告のコピー、その欠陥に関する他のメール・メッセージ、欠陥修理の進行状況等にアクセスすることができる。

(3) Strudel の概念モデル

ここでは、Strudel の抽象的計算モデルにおける基本的なタイプについて、(2) のシナリオ例に基づいて解説する。

会話とタスクは、それぞれメッセージと行動項目の集合である。メッセージは会話手番を伝えるために利用される。例えば Request to repair a defect という手番などである。通常、ひとつの会話手番は次の会話手番を示唆し、そのメッセージを読む人によって実行される行動を提示する。例えば Request to repair a defect という手番は、次の会話手番として Agree to repair a defect という手番を示唆する。会話とタスクを開始し、展開し、終了する経路を導くために、会話タイプとタスク・タイプを定義することができる。例えば、欠陥修理の会話においては、Request to repair a defect という手番あるいは、Are you the right person to handle this defect? という質問を最初の手番として定義する。欠陥追跡のタスクは、

製品開発システムなどの外部のツールからStrudelへ欠陥の通知が報告されたとき、あるいは、ユーザがRepair Defectという行動項目を作成したときに開始される。図3.8-5にStrudelの抽象タイプの間の関係の概要と、欠陥修理の会話における手番の例を示す。

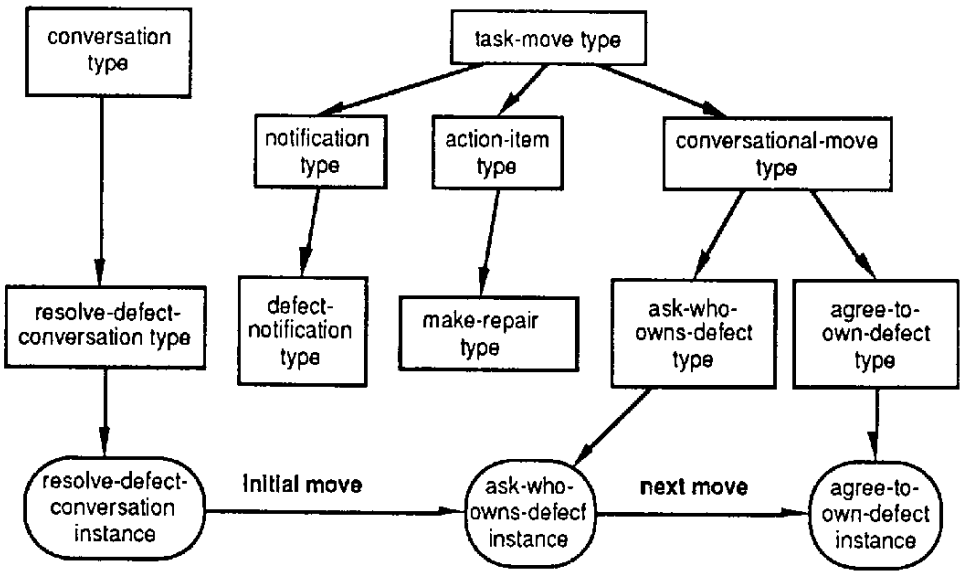


図3.8-5　会話と手番タイプの階層

conversational-move（会話手番）、action-item（行動項目）、notification（通知）等の基本タイプは、task-moveというルート・タイプのサブクラスである。これらのタイプのインスタンスは、タイトル、オプションのフィールド、およびその手番の文脈の中でユーザが適用できる行動を持っている。図3.8-6に単純化されたRepair Defect行動項目のためのMotifフォームの例を示す。図3.8-7は、この行動項目のためのタイプ定義を示す。ユーザによって行動項目のインスタンスが作成されたとき、ユーザのローカルなライブラリの中のタイプ定義は、デフォルトのフィールドと行動情報を提供する。

通知は、ユーザがインスタンスを作成するのではなく、アプリケーションからの呼出しへの応答としてインスタンスができるという意味で、擬似行動項目として取扱われる。この点を除いては、通知は行動項目と同一の性質を持つ。図3.8-8は、ユーザに欠陥を教えるためのソフトウェア開発ツールからの連絡によって作成された通知の例を示す。

特定の手番タイプに対応した行動を定義することができる。これらは、行動が提示されたとき、ボタンに対応付けされる。例えば、図3.8-6におけるInitiate Repairボタンである。手番に対して定義された行動は、例えば、行動項目や次の会話手番を作成するために利用することができる。フィールドの上で定義された行動は、例えば、カレンダやスケジュールの上である時間が空いているかどうかを確

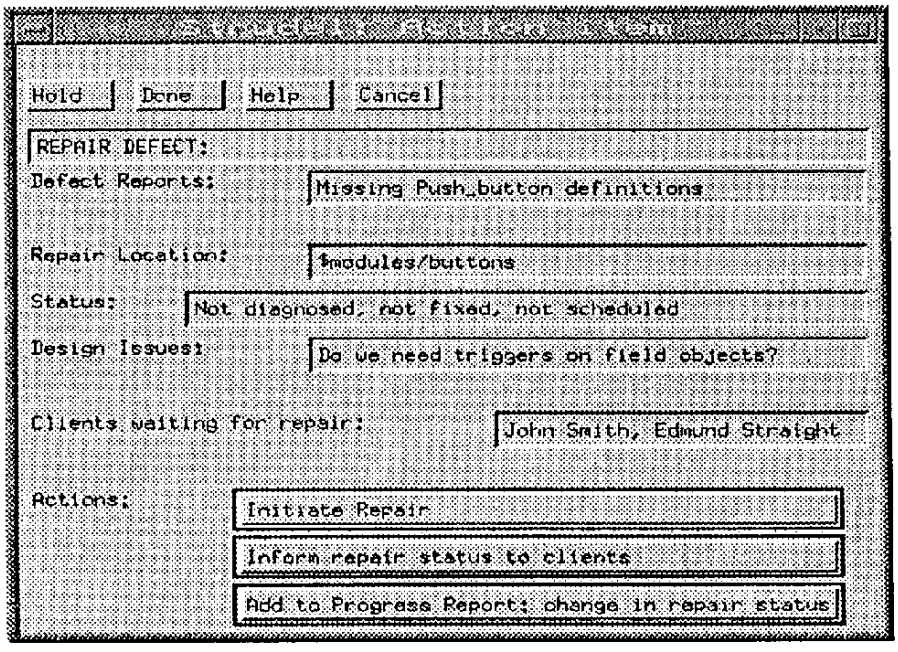


図3.8-6　「Repair Defect」の行動項目の作成

```
(register-task-move-type 'make-repair
      :title              "Repair Defect"
      :intro              "REPAIR DEFECT:"
      :field-sequence     '(review repair-location status design-issues who)
      :action-types '(    '(start-repair "Initiate Repair")
                          '(inform-clients "Inform repair status to clients")
                          '(note-in-progress-report
                                "Add to Progress Report: change in repair status")))
```

図3.8-7　「Repair Defect」の行動項目のタイプ定義

認するために利用できる。また、行動をユーザが定義し、特定のタスクのための操作をタスク・オブジェクトに施すことができる。例えば、技術上の変更指示を製品の設計履歴に記入するための行動を、生産管理システムおよび在庫管理システムへのインタフェースを呼ぶことによって、インプリメントすることができる。欠陥のあるソフトウェア・モジュールを検索する行動は、ソフトウェア保守ツールを呼ぶことによってインプリメ

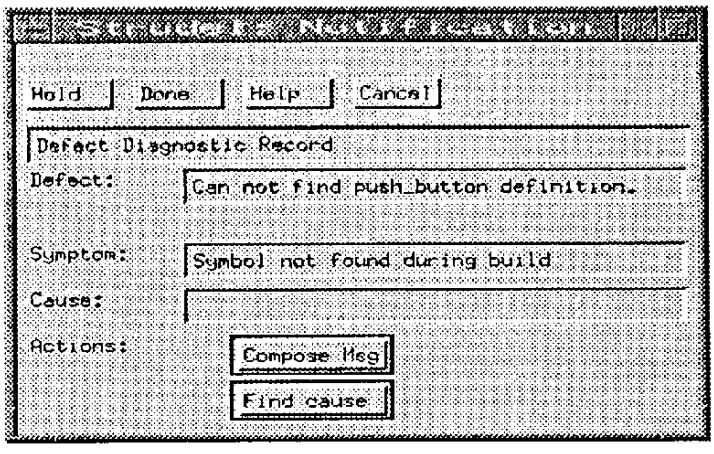


図3.8-8　受け取った通知

ントすることができる。手番が作成されているのか、あるいは、読まれているのかの違いによって、その手番のタイプ定義に指定された異なった行動を提示することができる。

会話手番は、半構造的であってよい。この点では、手番は「半構造化されたメッセージ」とよく似ている。Strudel においては、それぞれの手番がフィールドからなる構造を持つけれども、フィールドの内容は構造化されていないという意味において、手番は半構造化されたものであるといえる。ユーザは、フィールドに好きなだけの情報を書き込むことができるし、書かなくてもよい。また、フィールドの中の情報は特定のタイプである必要はない。会話手番が作成されたとき、それは電子メールのメッセージの中に挿入され、他のユーザに送られる。1つのメッセージにいくつかの手番を含めることができる。

メッセージをつなぎ合わせると、1つあるいは複数の会話となる。会話は、オープニングおよびそれに続くメッセージから構成される。オープニングは、参加者、最初のメッセージ等々を指定する。ユーザは1つの会話手番タイプを選択し、そのインスタンスを作ることによって、新しい会話を開始する。この選択は、トップレベル・メニューである Start conversation、あるいは既存の行動項目あるいは通知の文脈の中でおこなわれる。その後、ユーザはメッセージを編集し送信する。メッセージへの応答として、ユーザは次のメッセージを送り、このようにして会話が進行する。ユーザは会話をコピーしたり仲間に入ったりすることができる。

会話手番のための特定化された行動によって、ユーザは、会話の中の以前の手番およびそれにリンクされた行動項目と通知へ遡ることができる。このように探索する能力があるので、会話の中の行動をネットワーク形のグラフにして表示することができる。図3.8-9は、欠陥修理シナリオにおける手番のいくつかを示している。長方形で囲まれた手番が実行されたものである。破線のリンクは、可能ではあったが実行されなかった手番を結んでいる。

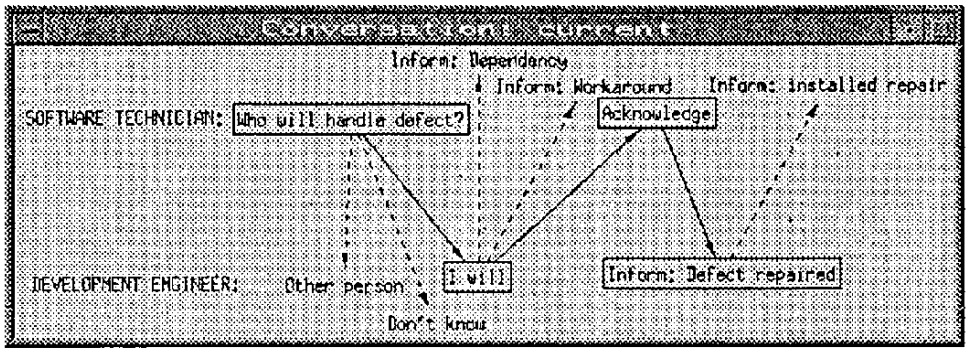


図3.8-9　会話のグラフ

タスクとは、タスクの最初の行動あるいは通知によって生成される、行動項目あるいは通知の集合である。会話も最初の会話手番から生成されるものであり、タスクは構造的には会話と類似している。

会話は特別のタスクとして扱われる。会話タイプおよびタスク・タイプは、一組の初期手番タイプを指定する。会話において宣言された初期手番タイプおよびタスク・タイプのメニューが現在、新しい会話とタスクを開始するために利用されている。

行動項目、通知、メッセージ等多様なクラスのためのブラウザを含め、タスクと会話のいくつかのデフォルトの表示方法が提供されている。現在の電子メール・ブラウザに似た、単純なメッセージ・ブラウザによって、従来の電子メールのメッセージとStrudelのメッセージを表示することができる。カレンダとTo-Do listを表示するために、ブラウザは、ユーザが行動項目とメッセージを、時間の関係とキーワード・マッチに基づいてソートし、フィルタをかけ、グループにまとめることを許している。例えば、メッセージを送信時間についてソートし、参加者によってグループ化するなどが可能である。会話ブラウザによって、ユーザは、会話の内部を探索したり、会話を永久保存したりすることができる。

(4) メッセージの作成

ユーザが会話手番を含むメッセージを読むとき、彼は、前の手番のタイプと内容に基づいて、次の会話手番を作成することができる。ユーザは、現在の手番の中に示唆されたものの中から次の手番タイプを選択する場合もあるし、または、ライブラリに定義されたタイプから選択しても良い。あるいは、タイプの付いていない手番を含むメッセージを送っても良い。また、新しいタイプを作成して使用してもよい。会話タイプは、会話の進行中に拡張することができる。ユーザは既存の手番タイプの中に、next-conv-move-typesとして、新しい手番タイプを定義することができる。同様な方法によって、タスクのタイプを定義し進化させることができる。

Strudelの現在のインタフェースにおいては、次の手番タイプの候補は、ボタンあるいはメニューを介して提示される。到着したメッセージを読んでいるとき、ユーザはボタンを押し、次の手番タイプを選択する。メッセージの送り手は、次のデフォルト手番タイプを設定することができる。これは、方法や方針を表現する、望ましい選択枝を示唆するためでもあるし、あるいは、ただ談話におけるデフォルトの焦点を設定するためでもあろう。ユーザは次の手番を編集するか、選択したタイプのデフォルトの手番を送るかのどちらかを選択する。この次の手番は、現在の手番への応答（in response to）である。これは、従来の電子メールにおけるin-reply-toによって作られる結び付きに類似している。しかし、in response toの手番は、誰に宛て送っても良い。このようにして、その時点で他の人たちを会話に入れることができる。例えば、問題の記述を受け取ったとき、ユーザは異なった人々に異なったリクエスト・メッセージを送り、問題に対する異なった解決方法の提案を要求することができる。

(5) 問題点

従来のオフィス・フォーム・システムにおいては、フィールドとボタンの上のテキスト・ラベルは固定である。Strudelにおいては、ユーザはラベルの表面のテキストおよびデフォルトのフィールド内容を容易に変更することができる。例えば、表示された挨拶文章の礼儀正しさの度合いを変更することができる。この柔軟性から多くの問題が発生する。読む人は固定したフォームの方が素速く理解することができるという利点は失われる。既存の電子メールと同じく、変更したというしるしがないので、送り手が行った小さな変更によって誤解が生じるかもしれない。

重要な問題は、会話管理システムと協調システムの間の相互運用可能性のためのフレーム・ワークの開発である。この問題に対する本研究の最初の実際的なアプローチは、従来からの「構造化されていない」電子メールおよび他のプロトタイプの会話管理、アクティブ・メッセージ、あるいは協調システムとの実験的な相互運用性を許すようStrudelを設計することであった。このため、会話手番のタイプ、フィールドの追加、および行動を互いに独立に定義した。これによってStrudelは、これらの機能の1つ以上をサポートしている他のシステムからのメッセージを実験的に解釈することが可能となった。例えば、特定のアプリケーションは半構造化メッセージだけを送るかもしれない。あるいは、Requestのような非構造化のタイプされたメッセージだけを送って来るかもしれない。または、アクティブで構造化されたメッセージを送って来るかもしれない。キーとなる問題は、（メッセージの中の）行動に対して、相互運用的な参照を提供することである。

この他、サポートすべき適切な会話「単位」を発見すること、ユーザ主体の設計と技術主体の設計との間に適切なバランスを取ること、機械によるサポートに最も適した会話クラスの発見などの問題を考慮しなければならない。

付録：参考文献一覧

(1) Ackerman, Mark S.他: Answer Garden: A Tool for Growing Organizational Memory, ACM SIGOIS Bulletin,Vol.11, No.2, pp.31-47(1990.4.25)

(2) Ahuja, S.R.他: The RAPPORT Multimedia conferencing system, COIS'88 Proceedings, pp.1-8 (1988.3.23)

(3) Ahuja, S. R.他: A Comparison of Application Sharing Mechanisms in Real-Time Desktop Conferencing Systems, ACM SIGOIS Bulletin,Vol.11, No.2, pp.238-248(1990.4.25)

(4) Aizu,Izumi （会津泉）：ユーザーの立場から視たグループウェアの問題点, ソフトウェア・ツール・シンポジウム' 9 2 予稿集, pp.25-28(1992.1.9)

(5) Aoyama,Mikio （青山幹雄）他: 大規模ソフトウェア分散並行開発の品質管理, Engineers, pp.18-24(1991.2)

(6) Aoyama,Mikio （青山幹雄）他: 通信サービス開発支援環境：ICAROS, 電子情報通信学会論文誌, pp.47-52(1991.11.21)

(7) Aoyama,Mikio （青山幹雄）：分散開発環境：新しい開発環境像を求めて, 情報処理学会論文誌,Vol.33, No.1, pp.2-13(1992.1)

(8) Applegate,Lynda M.他: A Group Decision Support System for Idea Generation and Issue Analysis in Organizational Planning, CSCW'86 Proceedings, pp.16-34(1986.12.3)

(9) Auramaeki, Esa他: A Speech-Act-Based Office Modeling Approach, ACM Transactions on Office Information Systems,Vol.6, No.2, pp.126-152(1988.4)

(10) Austin, Laurel C.他: Determinants and Patterns of Control over Technology in a Computerized Meeting Room, CSCW'90 Proceedings, pp.39-51(1990.10.7)

(11) Baecker, Ronald他: Bringing Icons to Life, CHI'91 Proceedings, pp.1-6(1991.4.27)

(12) Baecker, Ronald他: The University of Toronto Dynamic Graphics Project, CHI'91 Proceedings, pp.467-468(1991.4.27)

(13) Bair, James H.: Supporting Cooperative Work with Computers:Addressing Meeting Mania, Proceedings of IEEE COMPCON Spring 1989, pp.208-217(1989.2.27)

(14) Barfield, Lon他: The Views User-interface System, CHI'91 Proceedings, pp.415-416(1991.4.27)

(15) Bauersfeld, Penny F.他: User-Oriented Color Interface Design: Direct Manipulation of Color in Context, CHI'91 Proceedings, pp.417-418(1991.4.27)

(16) Beard, David他: A Visual Calendar For Scheduling Group Meetings, CSCW'90 Proceedings, pp.279-290(1990.10.7)

(17) Begeman,Michael他: Project NICK: Meetings Augmentation and Analysis, CSCW'86 Proceedings, pp.1-6(1986.12.3)

(18) Belew, Richard K.他: Hyper Mail: Treating Electronic Mail as Literature, ACM SIGOIS Bulletin, Vol.11, No.2, pp.48-54(1990.4.25)

(19) Bell, Bringham他: Usability Testing of a Graphical Programming Systems: Things We Missed in a Programming Walkthrough, CHI'91 Proceedings, pp.7-12(1991.4.27)

(20) Benson, Ian他: Some Social and Economic Consequences of Groupware for Flight Crew, CSCW'90 Proceedings, pp.119-129(1990.10.7)

(21) Bermann,Tamar他: Can Networks Make an Organization?, CSCW'88 Proceedings, pp.153-166 (1988.9.26)

(22) Bertino, E.他: An Object-Oriented Data Model for Distributed Office Applications, ACM SIGOIS Bulletin,Vol.11, No.2, pp.216-226(1990.4.25)

(23) Bielawski, Larry他: Intelligent System Design, John Wiley & Sons(1991)

(24) Bier, Eric A.他: Documents as User Interfaces, CHI'91 Proceedings, pp.443-444(1991.4.27)

(25) Bjerknes,Gro他: The Memoirs of Two Survivors: or the Evaluation of a Computer System for Cooperative Work, CSCW'88 Proceedings, pp.167-177(1988.9.26)

(26) Blomberg,Jeanette: The Variable Impact of Computer Technologies on the Organization of Work Activities, CSCW'86 Proceedings, pp.35-42(1986.12.3)

(27) Bly,Sara A.: A Use of Drawing Surfaces in Different Collaborative Settings, CSCW'88 Proceedings, pp.250-256(1988.9.26)

(28) Bly, Sara A.他: Commune: A Shared Drawing Surface, ACM SIGOIS Bulletin,Vol.11, No.2, pp.184-192(1990.4.25)

(29) Bochenski, Barbara: Workgroup Goals Push Groupware Boundaries, Software Magazine, pp.69-79 (1990.11)

(30) Bodker,Susanne他: Computer Support for Cooperative Design, CSCW'88 Proceedings, pp.377-394(1988.9.26)

(31) Bond, Alan H.: A Computational Model for Organizations of Cooperating Intelligent Agents, ACM SIGOIS Bulletin,Vol.11, No.2, pp.21-30(1990.4.25)

(32) Borenstein,Nathaniel S.他: Cooperative Work in the Andrew Message System, CSCW'88 Proceedings, pp.306-323(1988.9.26)

(33) Borning, Alan他: Two Approaches to Casual Interaction over Computer and Video Networks, CHI'91 Proceedings, pp.13-19(1991.4.27)

(34) Bowers, J. M.他: Studies in Computer Supported Cooperative Work: Theory,Practice and Design, North-Holland(1991)

(35) Bowers,John他: Local and Grobal Structuring of Computer Mediated Communication: Developing Linguistic Perspectives on CSCW in Cosmos, CSCW'88 Proceedings, pp.125-139(1988.9.26)

(36) Boy, Guy A.: Intelligent Assistant Systems, Academic Press(1991)

(37) Brothers, L.他: ICICLE: Groupware For Code Inspection, CSCW'90 Proceedings, pp.169-181 (1990.10.7)

(38) Bui, Tung X.: Co-op A Group Decision Support System for Cooperative Multiple Criteria Group Decision Making, Springer-Verlag(1987)

(39) Bullen, Christine V.他: Learning from User Experience with Groupware, CSCW'90 Proceedings, pp.291-302(1990.10.7)

(40) Card, Stuart K.他: The Information Visualizer,an Information Workspace, CHI'91 Proceedings, pp.181-188(1991.4.27)

(41) Cashman,Paul他: Achieving Sustanable Complexity Through Information Technology: Theory and Practice, CSCW'86 Proceedings, pp.307-317(1986.12.3)

(42) Catlin, Timothy他: InterNote:Extending a Hypermedia Framework to Support Annotative Collaboration, Hypertext'89 Proceedings, pp.365-378(1989.11)

(43) Chalfonte, Barbara他: Expressive Richness: A Comparison of Speech and Text as Media for Revision, CHI'91 Proceedings, pp.21-26(1991.4.27)

(44) Chaudhury,Abhijit他: What Can Computer Programs Do To Facilitate Negotiation Processes ?, COCS'91 Proceedings, pp.269-284(1991.11.5)

(45) Chen, Hsinchun他: A Knowledge-Based Approach to the Design of Document-Based Retrieval Systems, ACM SIGOIS Bulletin,Vol.11, No.2, pp.281-290(1990.4.25)

(46) Chien, Jen-Hsien他: RPP: A System for Prototyping User Interfaces, CHI'91 Proceedings, pp.419-420(1991.4.27)

(47) Chrysanthis, Panayiotis K.他: A Logically Distributed Approach for Structuring Office Systems, ACM SIGOIS Bulletin,Vol.11, No.2, pp.11-20(1990.4.25)

(48) Ciborra,Claudio他: Encountering Electronic Work Groups: A Transaction Costs Perspective, CSCW'88 Proceedings, pp.94-101(1988.9.26)

(49) Clement, Andrew: Cooperative Support for Computer Work: A Social Perspective on the Empowering of End Users, CSCW'90 Proceedings, pp.223-236(1990.10.7)

(50) Clemons,E.K.他: Inforamation Technology and the Changing Nature of the Financial Services Industry, COSCIS'91 Proceedings, pp.93-116(1991.8.27)

(51) Cohen, Philip R.: Computer Dialogue Laboratory SRI International, CHI'91 Proceedings, pp.469-470(1991.4.27)

(52) Colazzo,L.他: Interpretation of Human Relations in Computer Supported Communication: A Test with a Pragmatic Model, COSCIS'91 Proceedings, pp.77-92(1991.8.27)

(53) Conklin, Jeff他: gIBIS: A Hypertext Tool For Exploratory Policy Discussion, CSCW'88 Proceedings, pp.140-152(1988.9.26)

(54) Conklin,Jeff: Supporting Decision Dialog and Capturing Rationale, ソフトウェア・ツール・シンポジウム' 9 2 予稿集, pp.73-75(1992.1.9)

(55) Cook, Peter他: Project Nick: Meeting Augmentation and Analysis, ACM Transactions on Office Information Systems,Vol.5, No.2, pp.132-146(1987.4)

(56) Coutaz, Joeulle他: Applications: A Dimension space for User Interface Management Systems, CHI'91 Proceedings, pp.27-32(1991.4.27)

(57) Crowley, Terrence他: MMconf: An Infrastructure for Building Shared Multimedia Applications, CSCW'90 Proceedings, pp.329-342(1990.10.7)

(58) Crowston,Kevin他: Cognitive Science and Organizational Design: A Case Study of Computer Conferencing, CSCW'86 Proceedings, pp.43-61(1986.12.3)

(59) Cypher, Allen: Eager: Programming Repetitive Tasks by Example, CHI'91 Proceedings, pp.33-39 (1991.4.27)

(60) Cypher, Allen: Video Presentation Eager: Programming Repetitive Tasks by Example, CHI'91 Proceedings, pp.445-446(1991.4.27)

(61) Danielsen,Thore他: The Amigo Project: Advanced Group Communication Model for Computer-Based Communications Environment, CSCW'86 Proceedings, pp.115-142(1986.12.3)

(62) De Cindio,F.他: CHAOS as a Cordination Technology, CSCW'86 Proceedings, pp.325-342 (1986.12.3)

(63) Delisle,Norman M.他: Contexts -- A Partitioning Concept for Hypertext, CSCW'86 Proceedings, pp.147-152(1986.12.3)

(64) Dennis, Alan R.他: Group,Sub-group and Nominal Group Idea Generation in an Electronic Meeting Environment, IEEE HICSS, pp.573-579(1991)

(65) DePaoli, Flavio: A Model for Real-Time Co-operation, ECSCW'91 Proceedings, pp.203-217 (1991.9.24)

(66) Dewan, Prasun他: Flexible User Interface Coupling in a Collaborative System, CHI'91 Proceedings, pp.41-48(1991.4.27)

(67) Dewitz,S.Donaldson: Contracting on a Performative Network: Using Information Technology as a Legal Intermediary, COSCIS'91 Proceedings, pp.271-294(1991.8.27)

(68) Dietz, J. L.他: Speech Acts or Communicative Action ?, ECSCW'91 Proceedings, pp.235-248 (1991.9.24)

(69) Dobson,J.E.他: Determining Requirements for CSCW: the ORDIT Approach, COSCIS'91 Proceedings, pp.333-354(1991.8.27)

(70) Dyson, Esther: A Notable Order For Groupware, DATAMATION, pp.51(1990.5.1)

(71) Dzida, Wolfgang他: ERGO-Shell: A Unix-Interface for task Preparation, CHI'91 Proceedings, pp.421-422(1991.4.27)

(72) Edigo,Carmen: Video Conferencing as a Technology to Support Group Work: A Review of its Failures, CSCW'88 Proceedings, pp.13-24(1988.9.26)

(73) Ehrlich, Kate: Human Interface at Sun, CHI'91 Proceedings, pp.471-472(1991.4.27)

(74) Ellis,C.: The Socialization of Computers, COSCIS'91 Proceedings, pp.373(1991.8.27)

(75) Ellis, C.A.他: GROUPWARE: some issues and experiences, Communications of the ACM,Vol.34, No.1, pp.38-58(1991.1)

(76) Ellis, Clarence: CSCW'88 Report, ACM SIGOIS Bulletin,Vol.10, No.1, pp.2-4(1989.1)

(77) Elwart-Keys, Mary他: User Interface Requirements for Face to Face Groupware, CHI'90 Proceedings, pp.295-301(1990.4.1)

(78) Engelbart, Douglas C.: Knowledge-Domain Interoperability and an Open Hyperdocument System, CSCW'90 Proceedings, pp.143-156(1990.10.7)

(79) Engestrom,Yrjo他: Computerized Medical Records,Production Pressure and Compartmentalization in the Work Activity of Health Center Physicians, CSCW'88 Proceedings, pp.65-84(1988.9.26)

(80) Erickson, Thomas他: Designing a Desktop Information System: Observations and Issues, CHI'91 Proceedings, pp.49-54(1991.4.27)

(81) Eveland,J.D.他: Evolving Electronic Communication Networks: An Empirical Assessment, CSCW'86 Proceedings, pp.91-101(1986.12.3)

(82) Eveland,J.D.他: Work Group Structures and Computer Support: A Field Experiment, CSCW'88 Proceedings, pp.324-343(1988.9.26)

(83) Fanning,Tony他: Computer Teleconferencing: Experience at Hewlett Packard, CSCW'86 Proceedings, pp.291-306(1986.12.3)

(84) Feiner, Steven K.他: COMET: Generating Coordinated Multimedia Explanations, CHI'91 Proceedings, pp.449-450(1991.4.27)

(85) Feldman,Martha: Constraints on Communication and Electronic Messaging, CSCW'86 Proceedings, pp.73-90(1986.12.3)

(86) Fischer, Gerhard他: Interwining Query Construction and Relevance Evaluation, CHI'91 Proceedings, pp.55-62(1991.4.27)

(87) Fischer, Gerhard他: Information Access in Complex,Poorly Structured Information Spaces, CHI'91 Proceedings, pp.63-70(1991.4.27)

(88) Fish, Robert S.他: The VideoWindow System in Informal Communications, CSCW'90 Proceedings, pp.1-11(1990.10.7)

(89) Foster,Gregg他: Cognoter,Theory and Practice of a Colab-orative Tool, CSCW'86 Proceedings, pp.7-15(1986.12.3)

(90) Frishberg, Nancy他: John Cocke: A Retrospective by Friends (An Interactive Multimedia Scrapbook), CHI'91 Proceedings, pp.423-424(1991.4.27)

(91) Frontczak, Susan他: Distributes Computing and Organizational Change Enable Concurrent Engineering, ECSCW'91 Proceedings, pp.131-146(1991.9.24)

(92) Frye, Colleen: Team Technologies Target the PC Office Community, Software Magazine, pp.111-117(1991.4)

(93) Furnas, George W.: New Graphical Reasoning Models for Understanding Graphical Interfaces, CHI'91 Proceedings, pp.71-78(1991.4.27)

(94) Gaines,Brian R.: Organizational Modeling and Problem Solving Using an Object- Oriented Knowledge Representation Server and Visual Language, COCS'91 Proceedings, pp.80-94(1991.11.5)

(95) Galegher, Jolene他: Computer-Mediated Communication for Intellectual Teamwork: A Field Experiment in Group Writing, CSCW'90 Proceedings, pp.65-78(1990.10.7)

(96) Galliers,R.D.他: Effective Strategy Formulation Using Decision Conferencing and Soft Systems Methodology, COSCIS'91 Proceedings, pp.157-180(1991.8.27)

(97) Garrett,L.Nancy他: Intermedia: Issues,Strategies,and Tactics in the Design of a Hypermedia Document System, CSCW'86 Proceedings, pp.163-174(1986.12.3)

(98) Gaver, William W.: Technology Affordances, CHI'91 Proceedings, pp.79-84(1991.4.27)

(99) Gaver, William W.他: Effective Sounds in Complex Systems: The ARKola Simulation, CHI'91 Proceedings, pp.85-90(1991.4.27)

(100) Gaver, William W.: Sound Support For Collaboration, ECSCW'91 Proceedings, pp.293-308 (1991.9.24)

(101) Giannetti,A.: A Computional Pragmatic Theory for Bussiness Messages Structuring and Generation, COSCIS'91 Proceedings, pp.53-76(1991.8.27)

(102) Gibbs, S.J.: LIZA:An Extensible Groupware Toolkit, CHI'89 Proceedings, pp.29-35(1989.5)

(103) Gomoll, Kathleen M.: Apple Computer's Human Interface Group: Advanced Technology Group, CHI'91 Proceedings, pp.473-474(1991.4.27)

(104) Goodman,George他: Collaboration Research in SCL, CSCW'86 Proceedings, pp.246-251 (1986.12.3)

(105) Gorry,G.Anthony他: Computer Support for Biomedical Work Groups, CSCW'88 Proceedings, pp.39-51(1988.9.26)

(106) Green, Eileen他: Office systems development and gender: Implications for computer supported co-operative work, ECSCW'91 Proceedings, pp.33-49(1991.9.24)

(107) Greenbaum,Joan: In Search of Cooperation: A Historical Analysis of Work Organization and Management Strategies, CSCW'88 Proceedings, pp.102-114(1988.9.26)

(108) Greenberg, Saul: Sharing views and interactions with single-user applications, ACM SIGOIS Bulletin,Vol.11, No.2, pp.227-237(1990.4.25)

(109) Greenberg, Saul: Personalizable groupware: Accommodating individual roles and group differences, ECSCW'91 Proceedings, pp.17-31(1991.9.24)

(110) Greenberg,Saul (ed.): Computer-supported Cooperative Work and Groupware, Academic Press Ltd(1991)

(111) Greif,Irene他: Data Sharing in Group Work, CSCW'86 Proceedings, pp.175-183(1986.12.3)

(112) Greif, Irene: Computer-Supported Cooperative Work: A Book of Readings, Morgan Kaufmann Publishers,Inc.(1988)

(113) Grudin,Jonathan: Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces, CSCW'88 Proceedings, pp.85-93(1988.9.26)

(114) Grudin, Jonathan他: User Interface Design in Large Corporations:Coordination and Communication across Disciplines, CHI'89 Proceedings, pp.197-203(1989.5)

(115) Grudin, Jonathan: Interface, CSCW'90 Proceedings, pp.269-278(1990.10.7)

(116) Grudin, Jonathan: CSCW: The Convergence of two Development Contexts, CHI'91 Proceedings, pp.91-97(1991.4.27)

(117) Gruman, Galen: The Beginnings of Collaboration Technology, IEEE Software, pp.106-107 (1991.5)

(118) Hahn, Udo他: Teamwork Support in a Knowledge-Based Information Systems Environment, IEEE transactions on Software Engneering,Vol.17, No.5, pp.467-482(1991.5)

(119) Halonen, David他: Shared Hardware: A novel Technology for Computer Support of Face to Face Meetings, ACM SIGOIS Bulletin,Vol.11, No.2, pp.163-168(1990.4.25)

(120) Hammainen,Heikki他: Form and Room: Metaphors for Groupware, COCS'91 Proceedings, pp.95-105(1991.11.5)

(121) Hanseth,O.: Integrating Information Systems: The Importance of Contexts, COSCIS'91 Proceedings, pp.133-156(1991.8.27)

(122) Hara Yoshinori他: A Set-to-Set Linking Strategy for Hypertext Systems, ACM SIGOIS Bulletin, Vol.11, No.2, pp.131-141(1990.4.25)

(123) Harrison, William H.他: Coordinating Concurrent Development, CSCW'90 Proceedings, pp.157-168(1990.10.7)

(124) Hart, Paul他: Inter-Organization Computer Networks: Indications of Shifts in Interdependence, ACM SIGOIS Bulletin,Vol.11, No.2, pp.79-88(1990.4.25)

(125) Hart, Paul他: Computer Integration: A Co-Requirement For Effective Inter-Organization Computer Network Implementation, CSCW'90 Proceedings, pp.131-142(1990.10.7)

(126) Hartfield,B.他: Issue-Centered Design for Collaborative Work, COSCIS'91 Proceedings, pp.295-310(1991.8.27)

(127) Heath, Christian他: Disembodied Conduct: Communication through Video in a Multi-Media Office Environment, CHI'91 Proceedings, pp.99-103(1991.4.27)

(128) Heath, Christian他: Collaborative Activity and Technological Design: Task Coordination in London Underground Control Rooms, ECSCW'91 Proceedings, pp.65-80(1991.9.24)

(129) Hedberg,B.: The Role of Information Systems in Imaginary Organizations, COSCIS'91 Proceedings, pp.1-8(1991.8.27)

(130) Hellman, Riitta: User Support: illustrating computer use in collaborative work contexts, CSCW'90 Proceedings, pp.255-267(1990.10.7)

(131) Helm, Richard他: Building Visual Language Parsers, CHI'91 Proceedings, pp.105-112(1991.4.27)

(132) Hiltz,Starr Roxanne: Collaborative Learning in a Virtual Classroom: Highlights of Findings, CSCW'88 Proceedings, pp.282-290(1988.9.26)

(133) Hix, Deborah他: Human-Computer Interface Development Tools:A Methodology for Their Evaluation, Communications of the ACM,Vol.34, No.3, pp.74-87(1991.3)

(134) Hogg, John他: OTM:Applying Objects to Tasks, OOPSLA'87 Proceedings, pp.388-393(1987.10.4)

(135) Holt, Anatol W.: Organizing Computer Use in the Context of Networks, Proceedings of IEEE COMPCON Spring 1989, pp.201-207(1989.2.27)

(136) Howes, Andrew他: Predicting the Learnbility of Task-Action Mappings, CHI'91 Proceedings, pp.113-118(1991.4.27)

(137) Hughes, John他: CSCW: Discipline or Paradigm? A sociological perspective, ECSCW'91 Proceedings, pp.309-323(1991.9.24)

(138) Ishida,M. (石田雅也): 分散処理の新しいスタイル，チーム・コンピューティング登場, 日経コンピュータ, pp.89-101(1990.5.21)

(139) Ishii,Hiroshi （石井　裕）: グループウェア技術の研究動向, 情報処理,Vol.30, No.12, pp.1502-1508(1989.12)

(140) Ishii,Hiroshi （石井　裕）: TeamWorkStation: Towords a Seamless Shared Workspace, CSCW'90 Proceedings, pp.13-26(1990.10.7)

(141) Ishii,Hiroshi （石井　裕）: コンピュータを用いたグループワーク支援の研究動向, コンピュータソフトウェア,Vol.8, No.2, pp.14-26(1991.3)

(142) Ishii,Hiroshi （石井　裕）他: ClearFace: Translucent Multiuser Interface for TeamWorkStation, ECSCW'91 Proceedings, pp.163-175(1991.9.24)

(143) Ishii,Hiroshi （石井　裕）: グループウェアの現状と展望, ソフトウェア・ツール・シンポジウム’９２予稿集, pp.5-17(1992.1.9)

(144) Jarrell,Nancy F.他: Network-based Systems for Asynchronous Group Communication, CSCW'86 Proceedings, pp.184-191(1986.12.3)

(145) Jeffries, Robin他: User Interface Evaluation in the Real World: A Comparison of Four Techniques, CHI'91 Proceedings, pp.119-124(1991.4.27)

(146) Jewett, Tom他: The Work Group Manager's Role in Developing Computing Infrastructure, ACM SIGOIS Bulletin,Vol.11, No.2, pp.69-78(1990.4.25)

(147) Jirotka, Marina他: Participation frameworks for computer mediated communication, ECSCW'91 Proceedings, pp.279-291(1991.9.24)

(148) Johansen, Robert: Groupware, The Free Press(1988)(会津　泉訳: グループウェア, 日経ＢＰ社, 1990.4.24)

(149) Johnson,Bonnie M.他: Using a Computer Based Tool to Support Collaboration: A Field Experiment, CSCW'86 Proceedings, pp.343-352(1986.12.3)

(150) Johnson, Peter: Human Computer Inteaction Laboratory Queen Mary and Wertfield College University of London, CHI'91 Proceedings, pp.475-476(1991.4.27)

(151) Jong, de Peter: Structure and Action in Distributed Organizations, ACM SIGOIS Bulletin,Vol.11, No.2, pp.1-10(1990.4.25)

(152) Kador, John: New Tools To Integrate The Office, DATAMATION, pp.55-58(1991.2.15)

(153) Kaplan,Simon M.他: Supporting Collaborative Processes with ConversationBuilder, COCS'91 Proceedings, pp.69-79(1991.11.5)

(154) Karbe, B.他: Support of Cooperative Work by Electronic Circulation Folders, ACM SIGOIS Bulletin,Vol.11, No.2, pp.109-117(1990.4.25)

(155) Kawakami,J. (川上潤司): 普及の糸口を模索する日本のグループウェア, 日経コンピュータ, pp.48-65(1989.3.11)

(156) Kensing,F.他: The Language/Action Approach to Design of Computer-Support for Cooperative Work: A Preliminary Study in Work Mapping, COSCIS'91 Proceedings, pp.311-332(1991.8.27)

(157) King,R.C.: The Effects of Communication Technology and Reward Systems on Collaborative Work, COSCIS'91 Proceedings, pp.355-372(1991.8.27)

(158) Knister, Michael J.他: DistEdit: A Distributed Toolkit for Supporting Multiple Group Editors, CSCW'90 Proceedings, pp.343-355(1990.10.7)

(159) Kobsa, Alfred他: User Models in Dialog Systems, Springer-Verlag(1989)

(160) Koenemann, Juegen他: Expert Problem Solving Strategies for Program Comprehension, CHI'91 Proceedings, pp.125-130(1991.4.27)

(161) Kraemer,Kenneth他: Computer-Based Systems for Group Decision Support: Status of Use and Problems in Development, CSCW'86 Proceedings, pp.353-375(1986.12.3)

(162) Kraut,Robert他: Relationships and Tasks in Scientific Research Collaborations, CSCW'86 Proceedings, pp.229-245(1986.12.3)

(163) Kraut,Robert他: Patterns of Contact and Communication in Scientific Research Collaboration, CSCW'88 Proceedings, pp.1-12(1988.9.26)

(164) Kraut, Robert E.他: Computerization and the Quality of working life: The role of control, ACM SIGOIS Bulletin,Vol.11, No.2, pp.56-68(1990.4.25)

(165) Kreifelts, Thomas他: Experiences with the DOMINO Office Procedure System, ECSCW'91 Proceedings, pp.117-130(1991.9.24)

(166) Kurlander, David他: Editable Graphical Histories: The Video, CHI'91 Proceedings, pp.451-452 (1991.4.27)

(167) Kuutti, Kari: The concept of activity as a basic unit of analysis for CSCW research, ECSCW'91 Proceedings, pp.249-264(1991.9.24)

(168) Kyng,Morten: Designing for a Doller a Day, CSCW'88 Proceedings, pp.178-188(1988.9.26)

(169) Kyng, Morten: The System Work Group Computer Science Department Aarhus University, CHI'91 Proceedings, pp.477-478(1991.4.27)

(170) Lai, Kum-Yew他: Object Lens: A "Spreadsheet" for Cooperative Work, CSCW'88 Proceedings, pp.115-124(1988.9.26)

(171) Lai, kum-Yew他: Object Lens: Letting End-Users Create Cooperative Work Applications, CHI'91 Proceedings, pp.425-426(1991.4.27)

(172) Lakin,Fred: A Performing Medium for Working Group Graphics, CSCW'86 Proceedings, pp.255-266(1986.12.3)

(173) Lantz,Keith: An Experiment in Integrated Multimedia Conferencing, CSCW'86 Proceedings, pp.267-275(1986.12.3)

(174) Laufmann,Steven C.他: Direct End-User Access to Remote Information, COCS'91 Proceedings, pp.16-28(1991.11.5)

(175) Lauwers, J. Chris他: Collaboration Awareness in Support of Collaboration Transparency: Requirements for the Next Generation of Shared Window Systems, CHI'90 Proceedings, pp.303-311(1990.4.1)

(176) Lauwers, J. Chris他: Replicated Architectures for Shared Window Systems: A Critique, ACM SIGOIS Bulletin,Vol.11, No.2, pp.249-260(1990.4.25)

(177) Lee, Jeffrey J.: Xsketch: a multi-user sketching tool for X11, ACM SIGOIS Bulletin,Vol.11, No.2, pp.169-173(1990.4.25)

(178) Lee, Jintae他: How Can Groups Communicate When They Use Different Languages? Translating Between Partially Shared Type Hierarchies, COIS'88 Proceedings, pp.22-29(1988.3.23)

(179) Lee, Jintae他: Partially Shared Views:A Scheme for Communicating among Group that Use Different Type Hierarchies, ACM Transactions on Information Systems,Vol.8, No.1, pp.1-26 (1990.1)

(180) Lee, Jintae: SIBYL: A Tool for Managing Group Decision Rationale, CSCW'90 Proceedings, pp.79-92(1990.10.7)

(181) Leibs, Scott: The Promise and The Pitfalls, Infomationweek, pp.38-40(1991.2.11)

(182) Leland,Mary D.P.他: Quilt: a collaborative tool for cooperative writing, COIS'88 Proceedings, pp.30-37(1988.3.23)

(183) Leland,Mary D.P.他: Collaborative Document Production Using Quilt, CSCW'88 Proceedings, pp.206-215(1988.9.26)

(184) Lewis, J. Bryan他: Dialogue Structures for Virtual Worlds, CHI'91 Proceedings, pp.131-136 (1991.4.27)

(185) Linde,Charlotte: Who's In Charge Here?: Cooperative Work and Authority Negotiation in Police Helicopter Missions, CSCW'88 Proceedings, pp.52-64(1988.9.26)

(186) Lochovsky, Frederick H.他: A Micro-Organizational Model for Supporting Knowledge Migration, ACM SIGOIS Bulletin,Vol.11, No.2, pp.194-204(1990.4.25)

(187) Loevstrand, Lennart: Being Selectively Aware with the Khronika System, ECSCW'91 Proceedings, pp.265-277(1991.9.24)

(188) Lohse, Jerry: A Cognitive Model for the Perception and Understanding of Graphs, CHI'91 Proceedings, pp.137-144(1991.4.27)

(189) Losada, Marcial他: Collaborative Technology and Group Prcess Feedback: Their Impact on Interactive Sequences in Meetings, CSCW'90 Proceedings, pp.53-64(1990.10.7)

(190) Lowe,David: SYNVIEW: THe Design of a System for Cooperative Structuring of Information, CSCW'86 Proceedings, pp.376-385(1986.12.3)

(191) Lu, Iva M.他: Idea Management In a Shared Drawing Tool, ECSCW'91 Proceedings, pp.97-112 (1991.9.24)

(192) Lunati, Jean-Michel他: Spoken language interfaces: The OM system, CHI'91 Proceedings, pp.453-454(1991.4.27)

(193) Lundell, Joy他: Human Factors in Software Development: Models,Techniques,and Outcomes, CHI'91 Proceedings, pp.145-151(1991.4.27)

(194) Mackay,Wendy E.: More than Just a Communicaton System: Deversity in the Use of Electronic Mail, CSCW'88 Proceedings, pp.344-353(1988.9.26)

(195) Mackay, Wendy E.他: How Do Experienced Information Lens Users Use Rules?, CHI'89 Proceedings, pp.211-216(1989.5)

(196) Mackay, Wendy E.: Patterns of Sharing Customizable Software, CSCW'90 Proceedings, pp.209-221(1990.10.7)

(197) Mackay, Wendy E.: Triggers and Barriers to Customizing Software, CHI'91 Proceedings, pp.153-160(1991.4.27)

(198) MacKenzie, I. Scott他: A Comparison of Input Devices in Elemental Pointing and Dragging Tasks, CHI'91 Proceedings, pp.161-166(1991.4.27)

(199) Mackey, Wendy E.: Ethical issues in the use of video: Is it time to establish guidelines?, CHI'91 Proceedings, pp.403-405(1991.4.27)

(200) Mackinlay, Jock D.他: The Perspective Wall: Detail and Context Smoothly Integrated, CHI'91 Proceedings, pp.173-179(1991.4.27)

(201) Mackinlay, Jock D.他: Rapid Controlled Movement Through Virtual 3D Workspaces , CHI'91 Proceedings, pp.455-456(1991.4.27)

(202) MacLean, Allan他: Reaching through Analogy: A Design Rational Perspective on Roles of Analogy, CHI'91 Proceedings, pp.167-172(1991.4.27)

(203) Madsen, Christian M.: Using Persistent Objects to Implement an Environment for Cooperative Work, ACM SIGOIS Bulletin,Vol.10, No.4, pp.11-20(1989.12)

(204) Mahling, Dirk E.他: An Interface for the Acquisition and Display of Office Procedures, ACM SIGOIS Bulletin,Vol.11, No.2, pp.123-130(1990.4.25)

(205) Malone,Thomas W.他: Semi-structured Messages are Surprisingly Useful for Computer-Supported Coordination, CSCW'86 Proceedings, pp.102-114(1986.12.3)

(206) Malone, Thomas W. : Center for Coordination Science Massachusetts Institute of Technology, CHI'89 Proceedings, pp.145-146(1989.5)

(207) Malone, Thomas W.他: What is Coodination Theory and How Can It Help Design Cooperative Work Systems?, CSCW'90 Proceedings, pp.357-370(1990.10.7)

(208) Mantei,Marilyn: Capturing the Capture Lab Concepts:A Case Study in the Design of Computer Supported Meeting Environments, CSCW'88 Proceedings, pp.257-270(1988.9.26)

(209) Mantei, Marilyn M.他: Experiences in the Use of a Media Space, CHI'91 Proceedings, pp.203-208 (1991.4.27)

(210) Markus, M. Lynne他: Why CSCW Applications Fail: Problems in the Adoption of Interdependent Work Tools, CSCW'90 Proceedings, pp.371-380(1990.10.7)

(211) Marmolin, Hans他: An analysis of Design and Collaboration in a Distributed Environment, ECSCW'91 Proceedings, pp.147-162(1991.9.24)

(212) Martens,Clarence他: OASIS: A Programming Environment for Implementing Distributed Organizational Supprort Systems, COCS'91 Proceedings, pp.29-42(1991.11.5)

(213) Martial, Frank von: A Conversation Model for Resolving Conflicts among Distributed Office Activities, ACM SIGOIS Bulletin,Vol.11, No.2, pp.99-108(1990.4.25)

(214) Matsushita,Y. (松下　温): 図解グループウェア入門, オーム社(1991)

(215) Mattos, Nelson M.他: An Approach to Integrated Office Document Processing & Management, ACM SIGOIS Bulletin,Vol.11, No.2, pp.118-122(1990.4.25)

(216) McCarthy, John C.他: An experimental study of common ground in text-based communication, CHI'91 Proceedings, pp.209-215(1991.4.27)

(217) Minneman, Scott L.他: Managing a Trois: a study of a multi-user drawing tool in distributed design work, CHI'91 Proceedings, pp.217-224(1991.4.27)

(218) Moad, Jeff: Tools To Automate Quality Production, DATAMATION, pp.63-66(1991.4.15)

(219) Moran, Thomas P.他: The Workaday World As a Paradigm for CSCW Design, CSCW'90 Proceedings, pp.381-393(1990.10.7)

(220) Morch, Anders他: JANUS: Basic Concepts and Sample Dialog, CHI'91 Proceedings, pp.457-458 (1991.4.27)

(221) Muller, Michael J.他: Toward a Definition of Voice Documents, ACM SIGOIS Bulletin,Vol.11, No.2, pp.174-183(1990.4.25)

(222) Muller, Michael J.: PICTIV - An Exploration in Participatory Design, CHI'91 Proceedings, pp.225-231(1991.4.27)

(223) Mulligan, Robert M.他: User Interface Design in the Trenches: Some Tips on Shooting from the Hip, CHI'91 Proceedings, pp.232-236(1991.4.27)

(224) Murakami Kouichi他: Gesture Recognition using Recurrent Neural Networks, CHI'91 Proceedings, pp.237-242(1991.4.27)

(225) Myers, Brad A.: Graphical Techniques in a Spreadsheet for Specifying User Interfaces, CHI'91 Proceedings, pp.243-249(1991.4.27)

(226) Myers, Brad A.: Text Formatting by Demonstration, CHI'91 Proceedings, pp.251-256(1991.4.27)

(227) Nagano,Hironobu （長野宏宣）: 分散開発環境の基盤技術, 情報処理学会論文誌,Vol.33, No.1, pp.14-21(1992.1)

(228) Nakamura,Hideo （中村英夫）他: ソフトウェア分散開発支援システム　D2, ソフトウェア・ツール・シンポジウム' 9 2 予稿集, pp.37-46(1992.1.9)

(229) Nakamura,M. (中村　正弘): グループウェア：電子メール普及を受け模索始まるグループ・コミュニケーションの姿, 日経コンピュータ, pp.48-65(1989.7.31)

(230) Nardi, Bonnie A.他: An Ethnographic Study of Distributed Problem Solving in Spreadsheet Development, CSCW'90 Proceedings, pp.197-208(1990.10.7)

(231) Neches,Robert: Tools Help People Co-operate Only To The Extent That They Help Them Share Goals and Terminology, CSCW'86 Proceedings, pp.192-201(1986.12.3)

(232) Neuwirth, Christine M.他: Issues in the Design of Computer Support for Co-authoring and Commenting, CSCW'90 Proceedings, pp.183-195(1990.10.7)

(233) Newman,Denis: Sixth Graders and Shared Data: Designing a LAN Environment to Support Collaborative Work, CSCW'88 Proceedings, pp.291-305(1988.9.26)

(234) Newman, William M.他: PEPYS: Generating Autobiographies by Automatic Tracking, ECSCW'91 Proceedings, pp.175-188(1991.9.24)

(235) Newman-Wolfe,R.E.他: MACE: A Fine Grained Concurrent Editor, COCS'91 Proceedings, pp.240-254(1991.11.5)

(236) Nisida,Shougo （西田正吾）他: 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, ソフトウェア・ツール・シンポジウム' 9 2 予稿集, pp.53-67(1992.1.9)

(237) Nonogaki Hajime他: FRIEND21 Project: A Construction of 21st Century Human Interfaces, CHI'91 Proceedings, pp.407-414(1991.4.27)

(238) Nowaczyk,Ronald H.他: The Influence of Video in Desktop Computer Interactions, COCS'91 Proceedings, pp.106-116(1991.11.5)

(239) Nunamaker, J.F.他: Enterprise Analyzer:Supporting CASE with Electronic Meeting Systems, CASE'90 Proceedings, pp.108-109(1990.12.5)

(240) Nunamaker, J.F.他: Electronic Meeting Systems To Support Group Work, Communications of the ACM,Vol.34, No.7, pp.40-61(1991.7)

(241) Ochimizu,Kohitiroh （落水浩一郎）: ソフトウェア開発におけるグループウェアの役割, ソフトウェア・ツール・シンポジウム' 9 2 予稿集, pp.83-92(1992.1.9)

(242) Ohkubo,Masaaki他: Design and Implementation of a Shared Workspace by Integrating Individual Workspaces, ACM SIGOIS Bulletin,Vol.11, No.2, pp.142-146(1990.4.25)

(243) Opper,Susanna他: Technology for Teams : Enhancing Productivity in Networked Organizations, Van Nostrand Reinhold(1992)

(244) Orr,Julian: Narratives at Work, Story Telling as Cooperative Diagnostic Activity, CSCW'86 Proceedings, pp.62-72(1986.12.3)

(245) Palmiter, Susan他: An Evaluation of Animated Demonstrations for Learning Computer-based Tasks, CHI'91 Proceedings, pp.257-263(1991.4.27)

(246) Pankoke-Babatz, Uta: Computer Based Group Communication: the AMIGO activity model, Ellis Horwood Limited(1989)

(247) Patterson, John F.他: Rendezvous: An Architecture for Syncronous Multi-User Applications, CSCW'90 Proceedings, pp.317-328(1990.10.7)

(248) Pausch, Randy: Virtual Reality on Five Dollars a Day, CHI'91 Proceedings, pp.265-270 (1991.4.27)

(249) Pernici, Barbara: Object with Roles, ACM SIGOIS Bulletin,Vol.11, No.2, pp.205-215(1990.4.25)

(250) Pintado, Xavier他: Satellite, A Visualization and Navigation Tool for Hypermedia, ACM SIGOIS Bulletin,Vol.11, No.2, pp.271-280(1990.4.25)

(251) Pittman, James A.: Recognizing Handwriten Text, CHI'91 Proceedings, pp.271-275(1991.4.27)

(252) Plaisant, Catherine他: Scheduling ON-OFF home control devices, CHI'91 Proceedings, pp.459-460(1991.4.27)

(253) Poole,Marshall Scott他: Conflict Management and Group Decision Support Systems, CSCW'88 Proceedings, pp.227-243(1988.9.26)

(254) Press, Larry: Personal Computers as Reserch Tools, Communications of the ACM,Vol.34, No.4, pp.19-25(1991.4)

(255) Rapaport, Matthew: Computer Mediated Communications, JohnWiley&Sons,Inc.(1991)

(256) Rash, Wayne: Getting Bigger Groupware, BYTE, pp.93-96(1990.12)

(257) Reder,Stephen他: The Communicative Economy of the Workgroup: Multi-Channel Genres of Communication, CSCW'88 Proceedings, pp.354-368(1988.9.26)

(258) Reder, Stephen他: The Temporal Structure of Cooperative Activity, CSCW'90 Proceedings, pp.303-316(1990.10.7)

(259) Richards, John T.: Reserch in HCI and Usability at IBM's Interface Institute, CHI'91 Proceedings, pp.479-480(1991.4.27)

(260) Rieman, John他: An Automated Cognitive Walkthrough, CHI'91 Proceedings, pp.427-428 (1991.4.27)

(261) Rivers, Rod: Human Computer Interaction Division Logica Cambridge Ltd., UK , CHI'91 Proceedings, pp.481-482(1991.4.27)

(262) Robertson, George G.他: Cone Trees: Animated 3D Visualizations of Hierarchical Information, CHI'91 Proceedings, pp.189-194(1991.4.27)

(263) Robertson, George G.他: Information Visualization Using 3D Interactive Animation, CHI'91 Proceedings, pp.461-462(1991.4.27)

(264) Robinson, Mike他: Questioning Representations, ECSCW'91 Proceedings, pp.219-233(1991.9.24)

(265) Robinson, William N.: Negotiation Behavior During Requirement Specification, IEEE 12th ICSE, pp.268-276(1990)

(266) Rodden, Tom他: CSCW and Distributed Systems: The Problem of Control, ECSCW'91 Proceedings, pp.49-64(1991.9.24)

(267) Root,Robert W.: Design of a Multi-Media Vehicle for Social Browsing, CSCW'88 Proceedings, pp.25-38(1988.9.26)

(268) Ropa, Amanda: Computers as Communicators: Designing a Multimedia Interface that Facilitates Cultural Understanding Among Sixth Gragers, CHI'91 Proceedings, pp.429-430(1991.4.27)

(269) Rosson, Mary Beth他: A View Matcher for Reusing Smalltalk Classes, CHI'91 Proceedings, pp.277-283(1991.4.27)

(270) Rosson, Mary Beth他: Demondtrating a View Matcher for Reusing Smalltalk Classes, CHI'91 Proceedings, pp.431-432(1991.4.27)

(271) Rudnicky, Alexander I.他: Models for evaluating interaction protocols in speech recognition, CHI'91 Proceedings, pp.285-291(1991.4.27)

(272) Ruohonen,M.: Contradictions of Managerial Learning System for Strategic Information Systems Planning, COSCIS'91 Proceedings, pp.181-212(1991.8.27)

(273) Saeki,Yutaka （佐伯　胖）：ヒューマンインターフェースと認知工学, コンピュータソフトウェア,Vol.8, No.2, pp.4-13(1991.3)

(274) Sage, Andrew P.: An Overview of Group and Organizational Decision Support Systems, IEEE Control Systems, pp.29-33(1991.8)

(275) Sakashita,Yoshihiko （坂下善彦）: 分散開発環境の事例と今後の展望, 情報処理学会論文誌,Vol.33, No.1, pp.32-39(1992.1)

(276) Sakata,Shiro （阪田史郎）他: グループ通信アーキテクチャ - マルチメディア分散会議システム構築のための基本概念 -, 電子情報通信学会論文誌,Vol.OS89, No.26, pp.19-24 (1989.9.18)

(277) Samarasan, Dhanesh K.: Collaborative Modeling and Negotiation, COIS'88 Proceedings, pp.9-21 (1988.3.23)

(278) Sarin,Sunil K.他: A Process Model and System for Supporting Collaborative Work, COCS'91 Proceedings, pp.213-224(1991.11.5)

(279) Schill,Alexander他: Language and Distributed System Support for Complex Organizational Services, COCS'91 Proceedings, pp.1-15(1991.11.5)

(280) Schlack, Mark: Client/Server's Hot In New Appications, DATAMATION, pp.56-57(1991.5.15)

(281) Schmidt, Kjeld: Riding a Tiger,or Computer Supported Cooperative Work, ECSCW'91 Proceedings, pp.1-16(1991.9.24)

(282) Schooler, Eve M.他: A Packet-switched Multimedia Conferencing System, ACM Transactions on Information Systems,Vol.10, No.1, pp.12-22(1989.1)

(283) Schrage,Michael: Shared Minds : The New Technologies of Collaboration, Random House, Inc. (1990)

(284) Schultz, Jamie: A Graphical Reflection Notation used in an Intelligent Discovery World Tutoring System, CHI'91 Proceedings, pp.433-434(1991.4.27)

(285) Schutzer, Daniel: Business Decisions with Computers, Van Nostrand Reinhold(1991)

(286) Sebrechts, Marc M.他: Question asking as a Tool for Novice Computer Skill Acquisition, CHI'91 Proceedings, pp.293-299(1991.4.27)

(287) Sebrechts, Marc M.他: Hypermedia and Echocardiography: An Interface Design for Guided Discovery, CHI'91 Proceedings, pp.435-436(1991.4.27)

(288) Sen,Sandip他: A Formal Study of Distributed Meeting Scheduling : Preliminary Results,COCS'91 Proceedings, pp.55-68(1991.11.5)

(289) Shaw, Mildred L G他: Supporting Personal Networking Through Computer Networking, CHI'91 Proceedings, pp.437-438(1991.4.27)

(290) Sheffield, Jim: The Effects of Bargaining Orientation and Communication Medium on Negotiations in the Bilateral Monopoly Task:A Comparison of Decision Room and Computer Conferencing Communication Media, CHI'89 Proceedings, pp.43-48(1989.5)

(291) Shepherd, Allan他: Strudel - An Extensive Electronic Conversation Toolkit, CSCW'90 Proceedings, pp.93-104(1990.10.7)

(292) Shimojoh,Shinji （下条信二）他: 分散環境における教育支援通信システムの設計・開発, 電子情報通信学会論文誌D-1,Vol.J73-D-1, No.8(1990.8): , 情報処理学会論文誌

(293) Singer,Janice他: Children's Collaborative Use of a Computer Microworld, CSCW'88 Proceedings, pp.271-281(1988.9.26)

(294) Singley, Mark K.: Molehill: An Instructional System for Smalltalk Programming, CHI'91 Proceedings, pp.439-440(1991.4.27)

(295) Siochi, Antonio C.他: A Study of Computer-Supported User Interface Evaluation Using Maximal Repeating Pattern Analysis, CHI'91 Proceedings, pp.301-305(1991.4.27)

(296) Soares, L. F. G.他: LAN Based Real Time Audio-Data Systems, ACM SIGOIS Bulletin,Vol.11, No.2, pp.152-157(1990.4.25)

(297) Sobiesiak,Rick他: A Conceptual Modelling Approach to Authoring-in-the-Large for Hypertext Documents, COCS'91 Proceedings, pp.225-239(1991.11.5)

(298) Stage,J.: The Use of Descriptions in Analysis and Design of Information Systems, COSCIS'91 Proceedings, pp.237-260(1991.8.27)

(299) Stasko, John T.: Using Direct Manipulation to Build Algorithm Animations by Demonstration, CHI'91 Proceedings, pp.307-314(1991.4.27)

(300) Stastz,Cathleen他: Computer-Supported Cooperative Work: Examples and Issues in One Federal Agency, CSCW'86 Proceedings, pp.318-324(1986.12.3)

(301) Stefik,Mark他: WYSIWIS Revised: Early Experiences with Multi-User Interfaces, CSCW'86 Proceedings, pp.276-290(1986.12.3)

(302) Stefik, Mark他: Beyond The Chalkboard:Computer Support For Collaboration and Problem Solving In Meetings, Communications of the ACM,Vol.30, No.1, pp.32-47(1987.1)

(303) Subramanian, Ramesh: PENDS: an approach to modeling and retrieving 'pending' information, ACM SIGOIS Bulletin,Vol.11, No.2, pp.158-162(1990.4.25)

(304) Suchman,Lucy他: A Framework for Studying Research Collaboration, CSCW'86 Proceedings, pp.221-228(1986.12.3)

(305) Tan,B.C.Y.他: Impact of GDSS and Task Type on Consensus in Small Group Meetings, COSCIS'91 Proceedings, pp.33-52(1991.8.27)

(306) Tang,John C.他: A Framework for Understanding the Workspace Activity of Design Teams, CSCW'88 Proceedings, pp.244-249(1988.9.26)

(307) Tang, John C.他: Videodraw: A Video Interface for Collaborative Drawing, CHI'90 Proceedings, pp.313-320(1990.4.1)

(308) Tang, John C.他: Videowhiteboard: Video Shadows to Support Remote: Collaboration, CHI'91 Proceedings, pp.315-322(1991.4.27)

(309) Tarumi,Hiroyuki （垂水浩幸）：ソフトウエア設計用グループウェアに関する一考察, ソフトウェア工学,Vol.80-21, pp.159-166(1991.7.19)

(310) Tarumi,Hiroyuki （垂水浩幸）：グループウェアのソフトウェア開発への応用, 情報処理学会論文誌,Vol.33, No.1, pp.22-31(1992.1)

(311) Tatsukawa Kosuke: Graphical Toolkit Approach to User Interaction Description, CHI'91 Proceedings, pp.323-328(1991.4.27)

(312) Tetzlaff, Linda他: The Use of Guidelines in Interface Design, CHI'91 Proceedings, pp.329-333 (1991.4.27)

(313) Thiel, David D.: The Cue as part of a Gestural Interface, CHI'91 Proceedings, pp.463-464 (1991.4.27)

(314) Thovtrup, Henrik他: Assessing the Usability of a User Interface Standard, CHI'91 Proceedings, pp.335-341(1991.4.27)

(315) Trigg,Randall他: Supporting Collaboration in NoteCards, CSCW'86 Proceedings, pp.153-162 (1986.12.3)

(316) Trigg, Randall H.: Guided Tours and Tabletops: Tools for Communicating in a Hypertext Environment, CSCW'88 Proceedings, pp.216-226(1988.9.26)

(317) Truex,D.P.他: A Rejection of Structure as a Basis for Information Systems Development, COSCIS'91 Proceedings, pp.213-236(1991.8.27)

(318) Tseung, L. C. N.他: Implication of the Guaranteed,Reliable,Secure Broadcast Technology to Office Information Systems, ACM SIGOIS Bulletin,Vol.11, No.2, pp.147-151(1990.4.25)

(319) Ueda Hiroshi他: Impact: An Interactive Natural-Motion-Picture Dedicated Multimedia Authoring System, CHI'91 Proceedings, pp.343-350(1991.4.27)

(320) Urken, Arnord B.: Coordinating Distributed Actions via Agent Voting, ACM SIGOIS Bulletin, Vol.11, No.2, pp.136-141(1990.4.25)

(321) Viller, Stephen: The Group Facilitator: A CSCW Perspective, ECSCW'91 Proceedings, pp.81-95 (1991.9.24)

(322) Vin,Harrick M.他: Hierarchical Conferencing Architectures for Inter-Group Multimedia Collaboration, COCS'91 Proceedings, pp.43-54(1991.11.5)

(323) Ward, Douglas R.: Boosting Connectivity in a Student Generated Collaborative Database, ECSCW'91 Proceedings, pp.191-201(1991.9.24)

(324) Watabe,Kazuo （渡部和雄）他: Distributed Multiparty Desktop Conferencing System: MERMAID, CSCW'90 Proceedings, pp.27-38(1990.10.7)

(325) Watabe,Kazuo （渡部和雄）他: Distributed Desktop Conferencing System with Multiuser Multimedia Interface, IEEE Journal on Selected Areas in Communications,Vol.9, No.4, pp.531-539(1991.5)

(326) Watabe,Kazuo （渡部和雄）他: マルチメディア分散在席会議システムMERMAID, 情報処理学会論文誌,Vol.32, No.9, pp.1200-1209(1991.9)

(327) Watabe,Kazuo （渡部和雄）：コンセンサスにもとづくグループ意思決定支援方式, オペレーションズ・リサーチ,Vol.36, No.11, pp.547-551(1991.11)

(328) Watanabe,Isamu （渡部　勇）：共同作業を支援する計算機技術　- 集団合意形成の計算機支援に向けて -, オペレーションズ・リサーチ,Vol.36, No.11, pp.552-556(1991.11)

(329) Waterworth, John A.他: HCI Reserch at the Institute of System Science, CHI'91 Proceedings, pp.483-484(1991.4.27)

(330) Watson,R.T.他: An Integrative Framework for Understanding Why a GDSS Is Successful, COSCIS'91 Proceedings, pp.9-32(1991.8.27)

(331) Weigand,H.: The Linguistic Turn in Information Systems, COSCIS'91 Proceedings, pp.117-132 (1991.8.27)

(332) Weiser, Mark: The Computer for the 21st Century, SCIENTIFIC AMERICAN, pp.66-75(1991.9) (浅野正一郎: ２１世紀のコンピュータ, 日経サイエンス, pp.60-70(1991.11))

(333) Weltevreden,P.S.: The Development of EDI in the European Scheme, COSCIS'91 Proceedings, pp.261-270(1991.8.27)

(334) Wenzel, Elizabeth M.他: Localization with non-individualized virtual acoustic display cues, CHI'91 Proceedings, pp.351-359(1991.4.27)

(335) Whiteside,John他: Contextualism as a World View for the Reformation of Meeting, CSCW'88 Proceedings, pp.369-376(1988.9.26)

(336) Whittaker, Steve他: Co-ordinating Activity: An Analysis of Interaction in Computer-Supported Co-operative Work, CHI'91 Proceedings, pp.361-367(1991.4.27)

(337) Williams, Daniel: New Technologies for Coordinating Work, DATAMATION, pp.92-96 (1990.5.15)

(338) Winograd, Terry他: Understanding Computers and Cognition, Addison-Wesley Publishing Company,Inc.(1986)

(339) Winograd,Terry: A Language Perspective on the Design of Cooperative Work, CSCW'86 Proceedings, pp.203-220(1986.12.3)

(340) Winograd, Terry: Groupware:The next wave or just another advertising slogan?, Proceedings of IEEE COMPCON Spring 1989, pp.198-200(1989.2.27)

(341) Wolf, Catherine G.他: WE-MET (Window Environment-Meeting Enhancement Tools), CHI'91 Proceedings, pp.441-442(1991.4.27)

(342) Woo, Carson C.: SACT: A Tool for Automating Semi-Structured Organizational Communication, ACM SIGOIS Bulletin,Vol.11, No.2, pp.89-98(1990.4.25)

(343) Wroblewski, David A.他: DETENTE: Practical Support for Practical Action, CHI'91 Proceedings, pp.195-202(1991.4.27)

(344) Wulff, Wendie他: Animating Interfaces, CSCW'90 Proceedings, pp.241-254(1990.10.7)

(345) Yakemovic, K. C. Burgess他: Report on a Development Project Use of an Issue-Based Information System, CSCW'90 Proceedings, pp.105-118(1990.10.7)

(346) Yamada,Yoshiyasu （山田善靖）: 情報技術を用いる集団意思決定の支援, オペレーションズ・リサーチ,Vol.36, No.11, pp.531-537(1991.11)

(347) Yoder, Elise他: Collaboration in KMS,A Shared Hypermedia System, CHI'89 Proceedings, pp.37-42(1989.5)

(348) Zabback, P.他: Office Documents on a Database Kernel - Filing,Retrieval,and Archiving -, ACM SIGOIS Bulletin,Vol.11, No.2, pp.261-270(1990.4.25)

(349) Zanden, Brad Vander他: The Lapidary Graphical Interface Design Tool, CHI'91 Proceedings, pp.465-466(1991.4.27)

(350) Znati,Taieb F.他: Multi-Level Specification and Protocol Design for Distruibuted Multimedia Communication, COCS'91 Proceedings, pp.255-268(1991.11.5)

(351) パーソナル情報環境協会: グループウェア, 平成２年度 情報処理のパーソナル化に関する技術動向調査報告書, pp.89-107(1991.3)

平 成 4 年 3 月 発 行

発行所　財団法人　日本情報処理開発協会
東京都港区芝公園３丁目５番８号
機　機　振　興　会　館　内
TEL （３４３２）９３８４

印刷所　株式会社　正　文　社
東京都文京区本郷３丁目12番２号
TEL （３８１５）７２７１

03－R007