04—R004

グループワーク支援システムの
研究開発報告書

平 成 5 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4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、ワークステーションやパーソナルコンピュータの急速な普及とともに、ローカルエリアネットワークや広域ネットワーク等を介して地理的、時間的に離れている利用者をお互いに結び、ソフトウェアや情報等のリソースを交換して利用されるようになってきた。

これとともに、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にネットワークとワークステーションを活用する考え方が提案され、人間性にもとづくグループ活動と論理性にもとづく情報技術との共存を実現する技術の開発が要請されている。

このような状況のもとに、当協会では平成３年度より２年間にわたって、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情報システムの研究開発を行った。

実施に当たっては、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（主査：斎藤信男　慶應義塾大学環境情報学部　教授）を設置し、協調支援のモデル、協調作業を支援するためのコンピュータ基盤技術、分散開発技術、事例等について幅広く検討した。また、平成４年１０月には海外調査員を派遣し、国際会議での討論、大学・先進研究機関の訪問調査を実施した。

さらには、マルチメディアを用いた分散グループレビューシステムのプロトタイプを作成し、コンピュータによる共同作業支援の効果、問題点等の研究を行なった。このシステムは、ソフトウェア開発時の典型的な協同作業である開発成果物のレビューの支援をメンバーが分散したままで可能にさせるものである。

本報告書は、平成４年度に実施した事業の成果をとりまとめたもので７章で構成されている。

第１章では、コンピュータによるグループワーク支援について、また第２章では、それらへのニーズを述べている。第３章と第４章は、コンピュータによる支援モデルや技術について述べている。第５章は、ソフトウェア開発時のグループワーク支援に焦点を当て、その技術的な問題を述べている。第６章は、ソフトウェア開発におけるグループワーク支援システムの実現例、第７章は、文献調査結果を紹介している。

こうした研究開発の成果がコンピュータによるグループワーク支援に関心を持つ方々の参考になると共に広く情報処理技術発展の一助となれは幸いである。

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５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

（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	富士通株式会社　ビジネス通信事業本部 複合交換機事業部　第一ソフトウェア部第二開発課　技師
委　員	筏井 幸夫	株式会社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式会社 SI事業本部　品川システムセンタ　第１グループ　部長
委　員	小川　裕	日本電信電話株式会社　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・エム株式会社　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	日本電気株式会社　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	株式会社日立製作所　ビジネスシステム開発センタ 第二開発部　主任技師
委　員	貫井春美	株式会社東芝　システム・ソフトウェア生産技術研究所 開発第三部　ソフトウェア生産技術第二担当　主任
委　員	船川 浩伸	東京電力株式会社　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式会社　技術開発部 研究開発担当係長
委　員	向山　博	財団法人日本情報処理開発協会　開発研究室　主任研究員
事務局	土川　潤	財団法人日本情報処理開発協会　開発研究室　主任部員

目　次

はじめに

1．コンピュータによるグループワーク支援

1．1　はじめに …… 1
1．2　情報処理技術の進展とグループワーク …… 1
1．3　新職業人とグループワーク …… 3
1．4　ソフトウェア工学の先端性 …… 4
1．5　グループワークの今後の課題 …… 6

2．グループワーク支援のニーズ

2．1　グループワークの問題点 …… 7
2.1.1　問題点の整理 …… 7
2.1.2　問題点の要約 …… 10

2．2　問題点の構造 …… 11
2.2.1　問題の発生空間 …… 11
2.2.2　米国における調査例 …… 15

3．協調作業支援技術

3．1　メールの構造化技術 …… 19
3.1.1　構造化メール …… 19
3.1.2　構造化メールの事例 …… 20
3.1.3　構造化技術の課題 …… 23

3．2　ワークフロー制御技術 …… 26
3.2.1　ワークフローとプロセスプログラミング …… 26
3.2.2　ワークフロー制御 …… 27
3.2.3　ワークフローの事例 …… 28
3.2.4　ワークフロー制御の課題 …… 30

3．3　会話支援技術 …… 31
3.3.1　Speech-Acts 理論と会話理論（会話の語用論的考察） …… 31
3.3.2　会話の意味論的考察 …… 33
3.3.3　会話の構文論的考察 …… 34
3.3.4　会話支援技術の課題 …… 34

3．4　議論支援技術 …… 35
3.4.1　IBISモデルとgIBISツール …… 35
3.4.2　Pottsの設計手法表現モデル …… 37

3．5　協同文書作成支援技術 …… 39
3.5.1　協同文書作成システム …… 39
3.5.2　協同文書作成支援技術の事例 …… 40
3.5.3　協同文書作成支援技術の課題 …… 47

3．6　グループ意思決定支援技術 …… 48
3.6.1　GDSSとは …… 48
3.6.2　GDSSの実際とその問題点 …… 49
3.6.3　GDSSの課題 …… 52

3．7　作業空間の共有技術 …… 53
3.7.1　テレポインティング …… 53
3.7.2　情報共有技術 …… 53

4．グループウェア構築のためのコンピュータ基盤技術

4．1　マルチメディアネットワーク技術 …… 59
4.1.1　マルチメディアネットワークのシステム体系 …… 59
4.1.2　通信インフラ …… 60
4.1.3　高速プロトコル処理 …… 65
4.1.4　メディア入出力（圧縮）制御 …… 66
4.1.5　分散コンピューティング …… 67
4.1.6　マルチメディア文書通信 …… 69

4．2　グラフィカルユーザインタフェース …… 73
4.2.1　GUIとグループウェア …… 73

4.2.2 共有ウィンドウ …… 74
4.2.3 テレポインタ …… 78
4.2.4 動画像 …… 79
4.2.5 新しいGUI …… 80

4．3 オブジェクト指向データベース …… 83
4.3.1 オブジェクト指向データベースとは …… 83
4.3.2 OODBと協同作業支援 …… 85
4.3.3 OODBの問題点 …… 89
4.3.4 将来へ向けて …… 89

5．ソフトウェア開発におけるグループワーク支援

5．1 ソフトウェアの開発プロセス …… 91
5.1.1 ソフトウェアライフサイクルプロセスの国際標準化 …… 91
5.1.2 ソフトウェア開発工程の標準化 …… 93
5.1.3 開発プロセスの制御と管理 …… 96

5．2 開発プロセスとグループワーク支援 …… 98
5.2.1 コーディネーション理論 …… 98
5.2.2 コーディネーションプロセスとソフトウェア開発 …… 100

5．3 共通オブジェクトの参照 …… 104
5.3.1 リポジトリ …… 104
5.3.2 リポジトリマネジメント …… 108
5.3.3 リポジトリの製品 …… 109

5．4 コミュニケーション支援 …… 114
5.4.1 協調支援レベルとコミュニケーション支援 …… 114
5.4.2 コミュニケーション支援の具体例 …… 115

5．5 意思決定支援 …… 117
5.5.1 ソフトウェア開発における意思決定 …… 117
5.5.2 ソフトウェア開発における意思決定支援の適用 …… 119
5.5.3 グループレビューの支援 …… 119
5.5.4 分散開発のプロジェクト管理 …… 120

5.5.5 設計理由の記述 …… 122
5.5.6 ソフトウェア開発における意思決定支援の今後のあり方 …… 123

5．6 協調作業支援 …… 125
5.6.1 ソフトウェア開発における協調作業支援 …… 125
5.6.2 ワークフロー管理とプロセス管理 …… 126
5.6.3 なぜ、ワークフロー管理や開発プロセスが問題なのか？ …… 126
5.6.4 開発プロセス管理のアプローチ …… 127
5.6.5 ワークフロー管理のアプローチ …… 129
5.6.6 ソフトウェア開発におけるワークフロー管理の適用 …… 129
5.6.7 協調作業支援の今後の方向と課題 …… 131

5．7 新しいソフトウェア開発環境 …… 134
5.7.1 ツール統合とソフトウェアプロセス実行環境 …… 134
5.7.2 オープンなグループワーク支援環境 …… 138

6．ソフトウェア開発におけるグループワーク支援の実現例

6．1 マルチメディア分散グループレビューシステム …… 141
6.1.1 はじめに …… 141
6.1.2 ソフトウェア開発におけるレビュー作業 …… 141
6.1.3 グループレビューシステムの開発 …… 143
6.1.4 グループレビューシステムの利用 …… 146
6.1.5 おわりに …… 148

6．2 CAPSDFのグループワーク支援 …… 149
6.2.1 CASEプラットフォーム …… 149
6.2.2 グループワーク支援機能 …… 150
6.2.3 作業誘導 …… 151
6.2.4 関連情報の参照と変更通知 …… 153

6．3 SDEワークベンチ/6000 および CMVC/6000 による
分散ソフトウェア開発支援 …… 155
6.3.1 SDEワークベンチ/6000 …… 155
6.3.2 コンフィギュレーション・バージョン管理/6000（CMVC/6000） …… 157

7．グループウェアに関する調査

7．1　協調理論 …… 161
7.1.1　学際的調整理論 …… 161

7．2　協調支援環境構築のためのアーキテクチャ …… 165
7.2.1　統合型グループ支援環境（IGSE） …… 165
7.2.2　Mocca プロジェクト …… 171

7．3　設計議論モデル …… 173
7.3.1　IBISを応用した設計議論モデル …… 173

7．4　グループ意思決定支援システム …… 177
7.4.1　SAMMの利用調査 …… 177
7.4.2　EMS …… 181
7.4.3　要求仕様作成段階におけるGDSS …… 188

7．5　協調作業システムの設計支援 …… 193
7.5.1　拡張SADT …… 193

7．6　オフィスプロシージャシステム …… 199
7.6.1　Action Workflow …… 199
7.6.2　Oval …… 205
7.6.3　EuroCoOp …… 211

7．7　グループエディタ …… 217
7.7.1　Flexible Diff …… 217
7.7.2　CoAUTHOR …… 221

1．コンピュータによるグループワーク支援

1．コンピュータによるグループワーク支援

1.1　はじめに

高度情報社会を迎え、コンピュータの利用方法にもさらに進展が見られる。人間の活動を支え、社会の活性化に役立つためには、より進んだ利用技術、それを支えるソフトウェアおよびハードウェア技術が必要になる。

コンピュータによるグループワーク支援の環境やツールの問題は、最初ソフトウェア工学の一分野としての研究が始まった。その後、いろいろの分野でも、このような課題はさまざまな興味をもたれ研究されている。ある意味では、コンピュータの利用分野で最後に残された問題領域として追求されるべきものである。

ここでは、グループワーク支援という研究分野の位置付けを一般的な観点から述べてみる。

1.2　情報処理技術の進展とグループワーク

コンピュータの出現以来、大型電子そろばんという範疇からもっと別の分野のツール、環境としてコンピュータを捉えることが1970年以降行なわれてきた。図1.2-1に示すように、この進展は、人間の持つ情報処理能力をディジタルコンピュータで代行しようという試みととってもよいであろう。

その最初は、情報を蓄積し抽象化し検索するという機能で、いわゆる知識処理あるいは人工知能と呼ばれる分野である。人間の知をディジタルコンピュータで扱おうとするもので、1970年代の高速かつ会話型コンピュータシステムの出現により、それまでの人工知能の研究のほとんどをディジタルコンピュータの上で展開しようと提案したTerry Winograd[Winograd71]の論文がかなりのインパクトを持って迎えられた。この方向に沿い、MITあるいはスタンフォード大学の人工知能研究所（AI Lab）が華やかな活動を行なっていた。米国防総省もこの分野に対して大きな期待をし、研究費をつぎこんだのである。わが国では、1980年から始まった第五世代コンピュータのプロジェクトが、大きなインパクトを与えた。

次の課題は、人間の感覚や感性を重視した情報処理を追求するもので、いわゆる感性情報処理と呼ばれる分野である。これは、人間のもつ情に関連する感覚要素である視覚、聴覚、などをも十分満足させられ利用できるような情報処理のツールや環境を追求するもので、いわゆるメディアを扱うものである。これをディジタルコンピュータで扱えるようにするということで、ディジタルメディアと呼ぶ対象を考える。このような方向をいち早く打ち出したのがMITのMedia Lab[Brand87]であろう。1980年代には、これがマルチメディアというもっと一般向けする術語になって普及していった。わが国では、むしろ1990年代になってから、マルチメディアという分野が一斉に広まっていったと言える

であろう。これは、ちょうど家電業界など民生用の新製品を見つけ出さなければやっていけなくなった時期と一致したので、いろいろの試みがマルチメディアの分野で行なわれていくであろう。

最後の課題は、人間の意に関するもので、コンピュータの上では人間の協同作業を支援するツールや環境を追求するものである。人間は、どの分野でも意思決定をしていくが、特に複数の人間が意思決定に加わるのが複雑化した現代社会の問題解決では必然的であり、複数の人間で協同作業をした方が単独の人間の作業よりも有利であるのは当然である。人間の意思をディジタルコンピュータで扱う場合の対象を仮にディジタルマインドと呼ぶと、複数の人間のディジタルマインドをうまく扱えれば、協同作業の支援ツールや支援環境が実現できよう。このような分野を扱う研究のセンタとして、例えばCenter for Coordination Scienceという組織の提案がある[Malone92]。

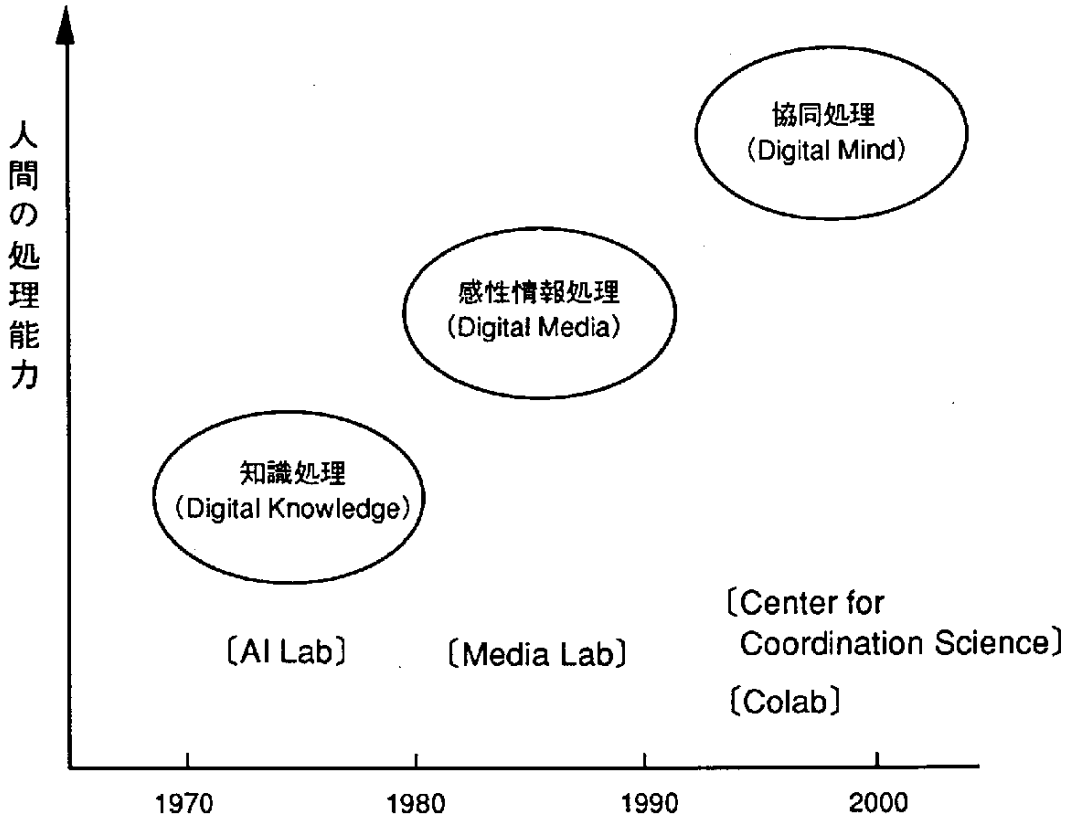


図1.2-1　人間の処理能力の追求

以上の三つの分野は、いずれも人間の本来持っている情報処理機能をディジタルコンピュータで代行したり支援したりする試みであり、簡単な問題ではない。したがって、たとえ1970年代に始まっている知識処理であっても1990年代には完全に解決されるわけではない。むしろ三つの分野が相互に関連しあい、融合化することによりまた新たな展望が開けてくるということで、ようやく人間の「知情意」が、ディジタルコンピュータで何らかの扱いをできるところまで到達したかということである。

グループワーク支援は、ディジタルマインドによる協同作業を扱う分野での課題であると言えよう。もちろん、これを支援し実現するためのツールや環境は、ディジタルメディアを扱うマルチメディア技術や知識処理の技術を融合させればより優れたものになる。人間の創造的な作業はマルチメディア的なグループワーク支援環境がなければできないということは、文献[Schrage90] に述べられている。新しいメディアの形態が、協同作業の成否を決めてしまうという位、相互の依存関係は密接なのである。グループワークということは、結局人間の知的な作業の最終局面を表していると言ってよいということで、個人の能力を最大限に引き出しながら、グループの他のメンバとの相互作用により創造的な知的作業まで昇華させていくのが、グループワーク支援の真骨頂と言えるであろう。

1.3 新職業人とグループワーク

高度な情報社会は、さまざまな影響を社会の仕組みに及ぼす。この影響を巧みに捉え、それを最大限に活かしていくのがこれからの組織や国のあり方であると言えよう。

Robert B. Reich は、米国クリントン政権の労働長官に抜擢されたハーバード大学の教授であるが、その著書[Reich91] によれば、今後の職業分類は、図1.3-1に示したような三つになると主張している。この中で、もっと重要で、経済的にも主要な働きをすると思われるのは、シンボル分析者である。米国は、この分野で圧倒的に世界をリードし、またその後継者の育成のための教育環境は米国にしか整備されていないと述べており、それを活かした活性化こそ米国の再生の唯一の方法であると説く。

このシンボル分析者として、具体的に現状のどんな職業が入るかといえば、ソフトウェアエンジニア、金融アナリスト、各種デザイナ、法律家、政策設計者などの例が挙げられている。いずれも、新しいものを実現するために、重要な意思決定をしている職業人であり、記号の操作（すなわち情報の操作）で世の中に大きな影響を与えていく専門家集団である。

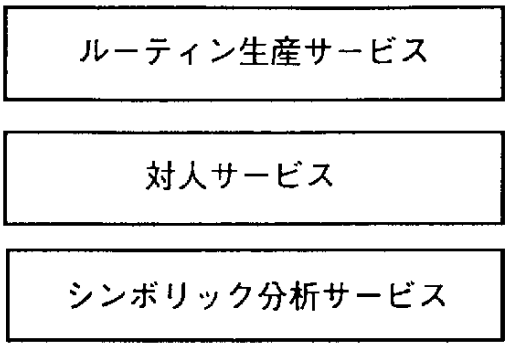


図1.3-1 新しい職業分類（R.Reich :" ザ・ワーク・オブ・ネイションズ " より）

これらの新職業人が活動する場面では、同種あるいは異種の専門家の間で協同作業が行なわれていく。複雑な問題解決をしていくためには、個人の能力ではすでに限界のあることは自明であり、どうしても協同作業が必要になる。これは、社会的な活動だけでなく、自然科学の分野でも見られることで、理論物理でも今や巨大な加速器で多数の物理学者の参加した実験結果を求めることが重要になっている。全く個人的な知的作業しか通用しなさそうな数学でも、4色問題の証明には、コンピュータを使って多数の場合について調べる必要があった。ソフトウェアの開発は、マルチディシプリン的なところがあり、プログラムを正確に組み上げる論理的な部分と、応用分野の機微に通じたり人間の感覚や気持ちを十分理解したりするという非論理的なものが必要になる。どちらかが欠けても、よいソフトウェアは実現できないであろう。これを一人のエンジニアに要求するのは酷であり、エンジニアの良いグループが良い協同作業を行なっていけばよいことになる。

人間の創造性は重要な資質であるが、これはむしろ異質な人間あるいは異分野の人間の協同作業で発揮できることが多いと文献[Schrage90]は主張している。たしかに、全く新しいことを思いつくのは、同じことを考えそうな人と相互作用を行なうよりも、異った分野、性格、能力を持った人と相互作用を行なう方が良さそうに思える。

このような考察から判断すると、高度な相互作用を支援してくれる協同作業支援ツールや支援環境は、シンボル分析者にとって非常に重要なものであることがわかる。これは、単にソフトウェアエンジニアだけの問題ではないのである。

1.4 ソフトウェア工学の先端性

グループワークを支援するグループウェアあるいはCSCWは、ソフトウェア工学の一分野として出現した。このような重要な問題が、何故ソフトウェア工学から出てきたのであろうか？これには、いろいろと議論もあろうが、結局ソフトウェア工学の先端性から出てきたことであろう。

ソフトウェア開発は、さまざまな問題が関連していることは前にも述べた。プログラムというなかなか捉えることの難しいものを作成することが課題であり、それを開発する技術体系を構築するのがソフトウェア工学であるから、簡単な話ではない。ソフトウェア工学を教育するために、Wang社が始めたWang Instituteは、必要な費用の膨大さに気がつき、結局やめてしまった。また、ソフトウェア工学の成果はなかなか上がらず、当初の目的通りいかないというのが、ソフトウェア工学を課題としている多くの研究者の常識となっている。これは、彼らが悪いのではなく、問題が難しいからであるといえる。

このようなソフトウェア工学であるから、その扱う課題や方式にはかなり先端な部分が含まれている。図1.4-1に示すように、いくつかの課題は他の工学分野、あるいはそれ以外の分野でも通じる手法であり方式である。

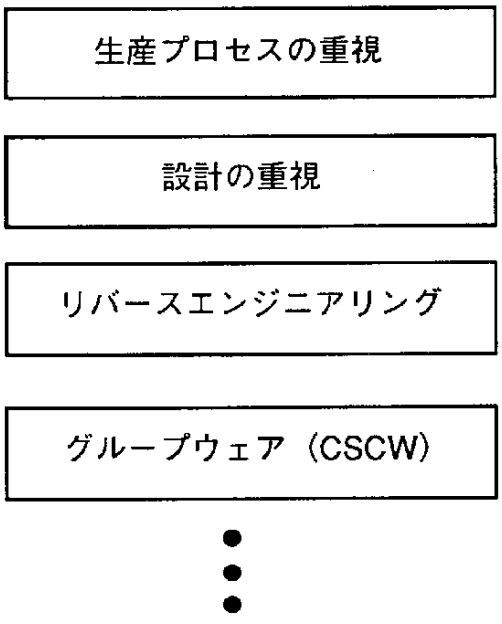


図1.4-1　ソフトウェア工学の先進性

例えば、生産プロセスを最初から考えてものを作るのは、なかなかできない。この生産プロセスには、ユーザの要求の分析から始まり、でき上がった製品の保守まで含まれている。自動車や電化製品がユーザの要求の分析をやって製品を開発するであろうか？最近では、ユーザの調査もかなりやられているようであるが、現在の自動車の形や機能は、必ずしもユーザの要求を分析した結果得られたものではない。

ソフトウェア開発では、その実装よりもその設計を重視している。つまるところ、大量生産というよりも、個々のユーザのニーズに合わせたものを作成するのであり、個別の要求や環境を反映した個々の設計を重要視するのは当然である。

ソフトウェア工学では、リバースエンジニアリングも重要な手法である。単にもの真似ではなく、製品開発されたプロセスやそこで使われている技法までたどることができれば、ソフトウェアエンジニアの苦労が財産として残っていく。

以上、いずれも他の工学分野で役立ちそうなものであり、先端的な工学の手法を示していると言える。グループウェアあるいはCSCW も、ソフトウェア工学の先端的な手法の一つであり、他の工学分野あるいはそれ以外の分野でも重要な課題となっていくものである。これを支援するツールあるいは環境は、ソフトウェア工学においてやはりうまく成果の上がらない手法の一つと見られるかもしれないが、単にCASE ツールや開発環境と考えるのではなく、多くの分野から期待されている課題でありツール開発であることを認識して取り組んでいかねばならない。

1.5 グループワークの今後の課題

グループワークは、前述したように非常に重要な課題であると言える。今後、その動向は無視できないものであろう。

一つの課題は、協同作業を支援するためには、知識処理あるいはマルチメディアの分野の技法を大いに取り入れ、その支援環境を作っていかなければならないことである。そのような統合化された協同作業環境は、今後どの分野でも重要になっていくであろう。

また、創造性向上のためには、異った分野の人間がスムーズに相互作用を実施できるように支援することである。その意味で、現在開発されているグループウェアは必ずしも簡単な問題ではない。本当に、能力、興味、性格などが異っている人間同士が、グループウェアを介して相互作用を及ぼし合い、創造性のある協同作業ができれば、その有効性はさらに増してくる。

【参考文献】

[Brand87] Brand, S.: The Media Lab:Investigating the Future at MIT, Viking (1987) (室、麻生共訳: メディアラボ, 福武書店(1988)).

[Malone92] Malone,T.W. and Crowston, K.: Toward an Interdisciplinary Theroy of Coordination, Center for Coordination Science, MIT (1992).

[Reich91] Reich, R. B.: The Work of Nations (1991) (中谷訳: ザ・ワーク・オブ・ネイションズ, プレジデント社 (1991)).

[Shrage90] Shrage, M.: Shared Minds: The Newest Technologies of Collaboration (1990) (藤田監修: マインドネットワーク, プレジデント社 (1992)).

[Winograd71] Winograd, Terry: Procedurs as a Representation forData in a Computer Program for Understanding Natural Language, MAC TR-84, Project MAC, MIT (1971.1).

2．グループワーク支援のニーズ

2．グループワーク支援のニーズ

2.1　グループワークの問題点

本節では、ソフトウェア開発に焦点を絞りこの分野におけるグループワークの問題点について述べる。

グループワーク支援システムの調査を進める上で、実際に携わった業務あるいは研究の経験の中から、ソフトウェア開発の分野におけるグループワーク上の問題点について様々な意見があげられた。

2.1.1　問題点の整理

その検討過程において抽出された項目について、これらを分類しグループワークを行う上での問題点として以下に整理する。

(1)　グループマネジメントについて

- 開発においてグループあるいは個人の役割分担や責任範囲は十分に明示されないことが多く、むしろ曖昧にして冗長性をもたせ、自発的な協調を期待する傾向がある。そのため懸案の見過ごしや重大な手戻りを引き起こす場合がある。
- グループのメンバを新たに構成する場合、グループの目的と個人の背景（スキル、相性、意識など）を的確にとらえて配置することは重要であるが、非常に難しい課題である。
- グループの目標が具体的に提示されることが少ない。
- メンバは自分の所属するグループがどの方向にどういう理由で進んでいるのかを知りたがっている。その参加意識を引き出し、どのように培うかが問題である。
- 大規模な開発を数社により協同で行う場合、末端の技術者まで目的意識を一致させるよう啓蒙することは不可能である。
- 強い仲間意識は現在のソフトウェア開発技術がカバーしきれないところを埋めてくれる。また高い信頼性と生産性に寄与するところも大である。
- 組織や役割分担は開発の進行に伴って変化するが、グループあるいは個人がその変化を必ずしも理解していない。
- 日本企業の強さである構成員同士の密度の高い情報共有あるいは価値観の共有が、フレックスタイムの導入や分散開発によって見直しを迫られている。
- リーダの資質（リーダシップ、意思決定力、コミュニケーションのうまさなど）が、プロジェクトの成功を大きく左右する。
- 以前にも増して、リーダおよび技術者の育成に投資が必要となってきている。

・　進度の遅いグループにより全体の進捗が遅れがちになることがあるが、早期の発見と適切な調整ができる管理手法が必要である。

・　品質をより良くするためには、作業分担においてグループあるいはメンバを得意な分野に割り当てたいところであるが、マンパワー不足で十分な配分はできない。

(2)　仕様・情報の伝達について

・　開発の分担によって情報（知識）の分断をいかに抑えるか。たとえば関連するモジュールを別グループで開発すれば、グループ間のコミュニケーションの増大や進捗管理が困難になる。

・　設計ドキュメントとして残らないインフォーマルなプロダクトが多く、後工程に引き継がれない。

・　仕様の作成にあたっては、多くの代替案からの仕様の決定の背景・根拠があるが、一般にそれらはドキュメントから省かれ、結果のみが書かれている。このため、他の者がその仕様書だけで意図を汲み理解することは困難であり、作成者への質問、確認作業が必要になってしまう。

・　設計者がどういう思考プロセスで設計を進めたかをモデル化などによって支援し「設計過程」の記述として記録する手法の実現が必要である。

(3)　レビュー／仕様変更について

・　工程間の仕様を的確に伝達するための効率の良いレビュー実施方法論が必要となる。また、レビューへの参加意識を高めることも重要な要素である。

・　いずれのレビューを行うためにも、会議室の確保、関連メンバのスケジュール調整が大変で、加えてインスペクタ等の有能な技術者が長時間拘束されてしまう。

・　対象システムが複雑になってくると、要求仕様に変更があった場合にその影響がどの程度に及ぶかを把握することが困難になる。また、変更の通知が必要なメンバに伝わらないということも発生する。プログラムの保守段階になると一層深刻な問題となる。

・　開発の規模が大きくなるに従って、仕様変更の指示や問題発生の連絡に遅延が生じる。

(4)　ドキュメントの記述について

・　ドキュメントの表記について標準化が行われている場合でも、細かい内容の表記については各個人に任されるため、担当者によって仕様書の記述内容に濃淡が出て深さに均一性がなくなる。

・　標準にそって作成されたドキュメントでも、形式的な記述ばかりで内容の乏しいものが混じる。

・　ソフトウェア開発においては、建築業の図面に相当するような規格はなく、ドキュメントベースでの協同作業を困難にしている。つまり表現が冗長となったり逆に表現が抜けるなどの問題が多く発生する。

・　ドキュメントの形式や用語の不統一は保守管理上に重大な影響をおよぼす。

・　ソフトウェア開発で作成されるドキュメントの電子化にあたっては、データ相互のリンク付けが重要な問題となる。加えてインフォーマルな情報のリンク付けも必要となる。

(5) 開発環境について

- 遠隔地において数社との分散開発を行ったが、電子メールですら利用ネットが異なるなど、システムやツールの違いで十分なコミュニケーションが行えない。
- 開発環境のインフラ整備が、分散開発のベースとなる。すなわち、経済性の高いネットワーク、ワークステーションの普及あるいは電子対話システムや電子メール網などの地域分散開発を支援する環境の拡充が肝要である。
- 協同オフィス環境は、グループの他のメンバの存在が意識でき連絡も随時行える。また、メンバの状態もそれとなくわかるなど、うまくゆけばグループの団結も強くなりメリットが多い。一方、多人数が同じ部屋を共有することから、雑音が多い、他人に干渉され易い、集中できないなどのデメリットも持つが、軽視せず解決してゆかねばならない。

(6) 開発方法論の適用について

- 開発プロセスの構造化は各企業において定着しつつあるが、細かく規定しすぎると手戻りの記述が困難となる。IEEEによるプロセス標準化はアクティビティレベルまで行われているが、組み合わせはユーザに委任つまりカスタマイズする必要がある。
- 上流工程に関してドメイン（業種）ごとに開発プロセスモデルが作成できれば、工程の管理が的確に行える。
- 開発プロセス（ワークモデル）によって、グループ間の相互作用を直接的に管理しようとする試みは行われているが、現場に定着しているとはいえない。
- 技術革新が激しくオープン化やダウンサイジングなどに対し、開発方法論の拡充が付いていけない。
- 計画から試験までを一貫してサポートし、オープン性のあるプラットフォーム、リポジトリを提供する「統合化CASE」ツールの整備が必要である。
- 開発方法論を普及し習熟させるのためには、多くの投資（教育体制、要員への初期投資など）を必要とする。
- 開発方法論を確立したうえで、グループ間・グループ内・個人の作業領域の切り分けを的確に指示することが必要である。
- 本来ソフトウェア開発の各工程は、本来その開始条件が整ってからスタートすべきものであるが、見切り的に不完全な状態で開始せざるをえなくなっている。同様に終了条件も不明確であり不完全のまま前工程が終了している場合もある。このように不明確な開発プロセスによって作業を進める場合、各作業者は関連する工程と緊密に連絡をとり、常に最新の状態を把握しながら作業を進めなければならない。

(7) 計画と評価について

・ ソフトウェアの品質、進捗、規模などを計るメトリックスが必要とされているが、経験と勘に頼ることが多く、また適切な解はない。

・ 客観的に把握できるプロジェクト管理データの収集方式が確立されていない。

2.1.2 問題点の要約

上記で整理された問題点はそれぞれの視点から抽出された意見であるが、これをキーワードに注目して頻度別にまとめ、表2.1-1に示す。

表2.1-1 ソフトウェア開発における協同作業の問題点

摘出頻度	問題点キーワード	補足キーワード
5	目的意識・達成目標の共有	動機付け、参加意識
4	作業境界・責任範囲の明確化	
4	設計書記述内容の均一化、正確化	
4	開発方法（プロセス）論の確立	的確な協同作業の切り分け
3	設計の背景／根拠の伝達	インフォーマルな情報の埋没
2	仲間意識・信頼関係	
2	適切な作業を行えるしくみ	
2	教育の実施・投資（期間、工数）	
2	正確で遅延のない指示・報告	
2	進捗管理と適切な対策実施	
1	レビュー実施方法論の検討	
1	レビュー作業の工数圧迫	
1	開発プロセスの不明瞭な適用	次工程への見切り発車
1	協同オフィス環境が持つデメリット	
1	開発環境のインフラ整備（ﾈｯﾄﾜｰｸ, WS, 電子ﾒｰﾙ等）	
1	統合化CASEの適用	
1	客観的なプロジェクト管理データの収集	
1	メトリックスの確立（品質・進捗・規模の測定）	
1	均一な品質達成	
1	ドキュメント形式、用語の統一	
1	意思決定と合意形成のメカニズムの解明	
1	適切な開発スケジュール	
1	開発作業の分担による情報（知識）の分断	
1	開発の進行に伴うグループの役割の動的変化	

2.2 問題点の構造

2.2.1 問題の発生空間

ソフトウェア開発時のグループワークで発生している問題がどのような要因と関係しているか、どういったときに発生し易いかを調べるために要因となる軸を考え、それらの軸からなる二次元上に問題点をマッピングし検討を行った。以下に、いくつかの例を示す。

(1) 公式度と開発工程

グループでの協同作業は、公になっているルールがあり、そのルールによって作業が拘束されながら進むものと、明確なルールは無いけれどもそのときの状況に応じて作業が進むものとがある。こうした、公式度（拘束力）と開発工程の関係を図示したのが図2.2-1である。

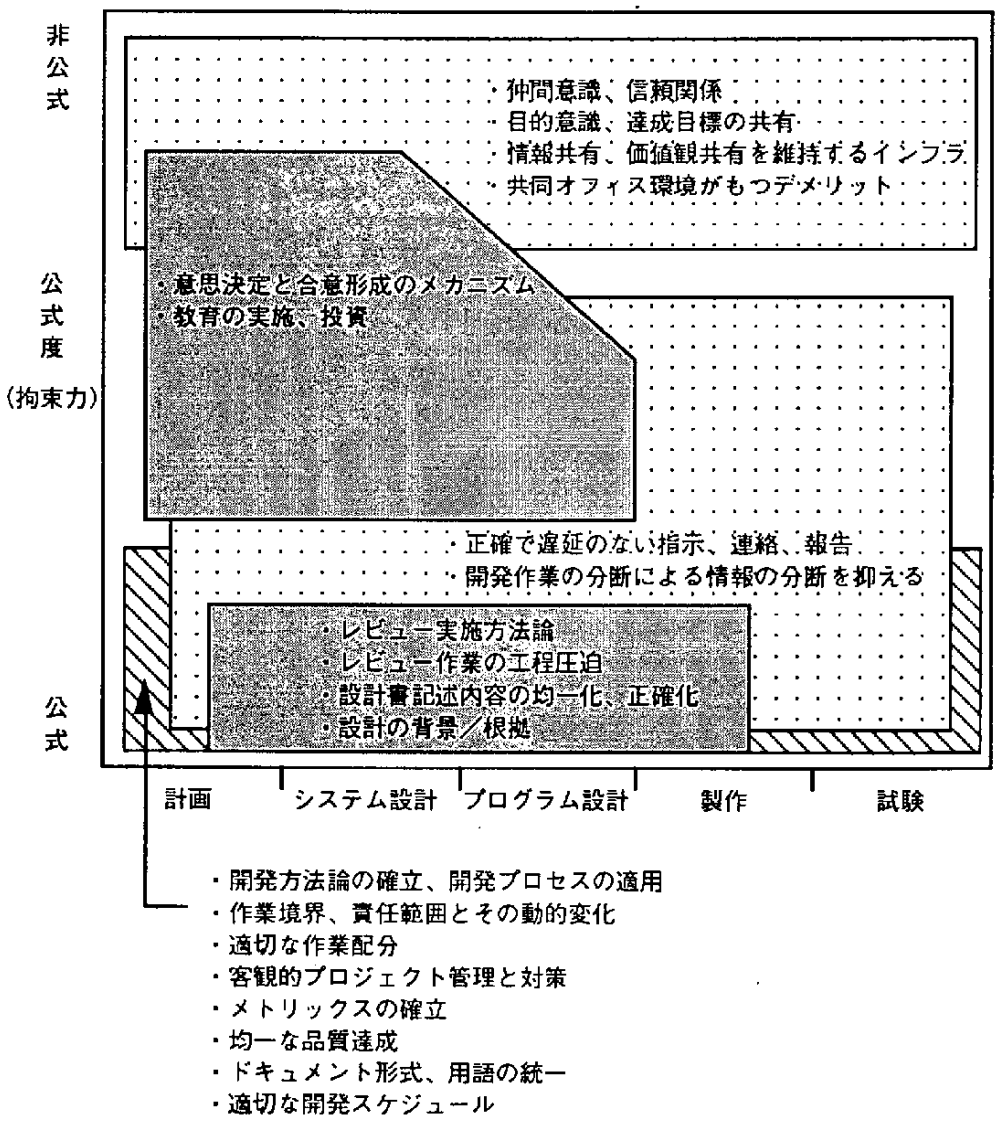


図2.2-1 公式度と開発工程図

図に示したように、仲間意識、信頼関係、目的意識といった問題については公式性が低い。一方、開発方法論・開発プロセス、作業境界、レビュー実施方法論、設計書の記述等は公式性は高い。意思決定と合意形成のメカニズム、指示・連絡・報告等は、中間に位置する。

さらに、意思決定・合意形成は、ソフトウェア開発の上流工程と関係が深く、レビュー等は、上流工程から中流工程に関係している。これら以外は、ソフトウェア開発工程全体に当てはまる。

公式性の高いものについては、構造化されたグループワーク支援の可能性の検討ができ、公式性の低いものについては、構造化されていないグループワーク支援による解決策の検討ができる。例えば、仲間意識、信頼関係といった作業の支援には、リアルタイムのビデオ画像の利用によって、親近感、共存感を持たせる方法が考えられる。また、開発方法論、開発プロセスによって引き起こされている問題については、開発プロセスをスクリプトとして記述し、実行するといった構造化された作業支援環境の構築が考えられる。

(2) 理論の成熟度と実施の難易度

協同作業の過程がどの程度理論的に解明されているか、モデル化されているか、またそうした理論の実施がどの程度容易に行えるかを考えることは、協同作業で発生している問題の解決にコンピュータを用いることができるかを検討する上で役に立つ。図2.2-2に理論の成熟度と実施の難易度の関係を示す。

仲間意識、信頼関係の形成といった問題は、人間の心理的な側面が強く理論的な解明が最も困難な分野であるといえる。一方、プロセス／プロダクトの依存関係の明確化といった問題は、解析可能であり、モデル化も可能である。しかし、手作業で詳細な関係付けや関係の整合性を維持することは、膨大な工数を必要とすることから極めて困難である。

意味的に統一のとれた仕様書の作成は、形式言語を用いることにより可能であるが、形式言語自体が難しく利用が困難である。

図の中で、理論的な解明がある程度なされているものの工数、教育等の問題で実施が困難となっている左上の部分に対するコンピュータ支援の実現可能性が高いと考えられる。

(3) 分散と組織の違い

数々の問題は、組織の違いや分散によって引き起こされていると考えることができる。そうした関係を図示したのが図2.2-3である。

図から分かるとおり、スケジュール調整といった作業は、組織あるいはグループが分散されることにより引き起こされ、形式の不統一は、組織の違いによって引き起こされる場合が多い。また、作業境界、責任範囲の不明確化は、組織の違い、分散によって引き起こされる。

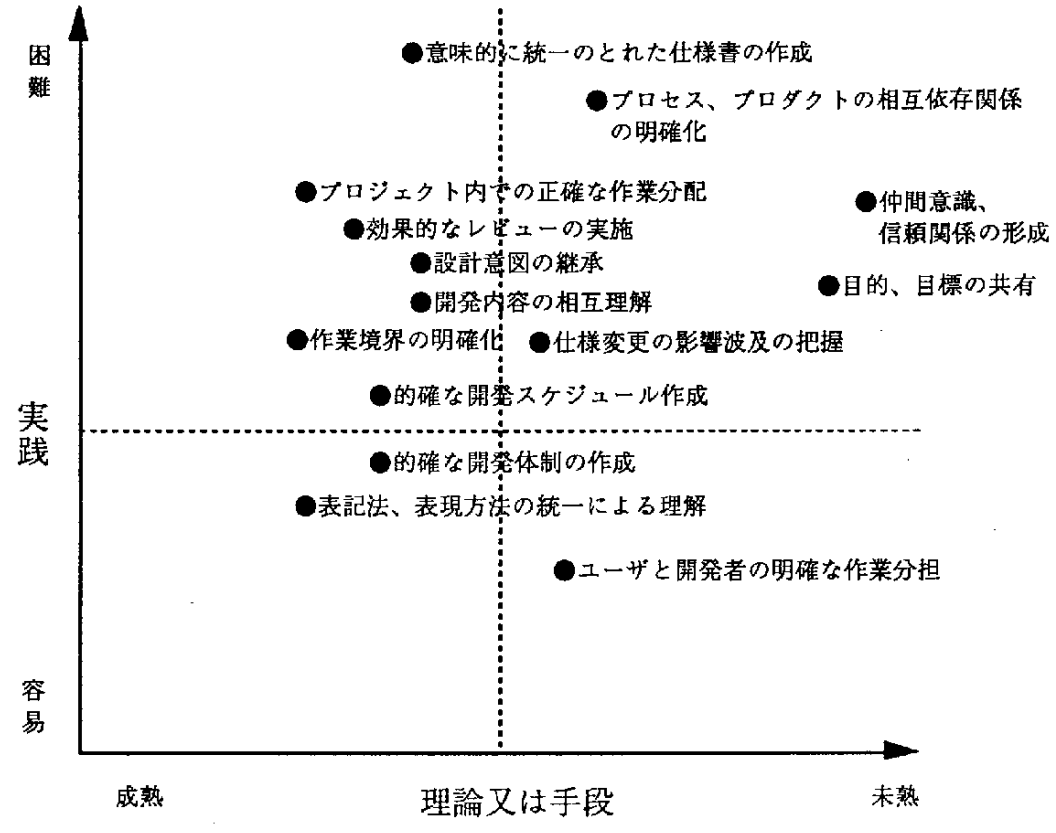


図2.2-2　理論の成熟度と実施の難易度

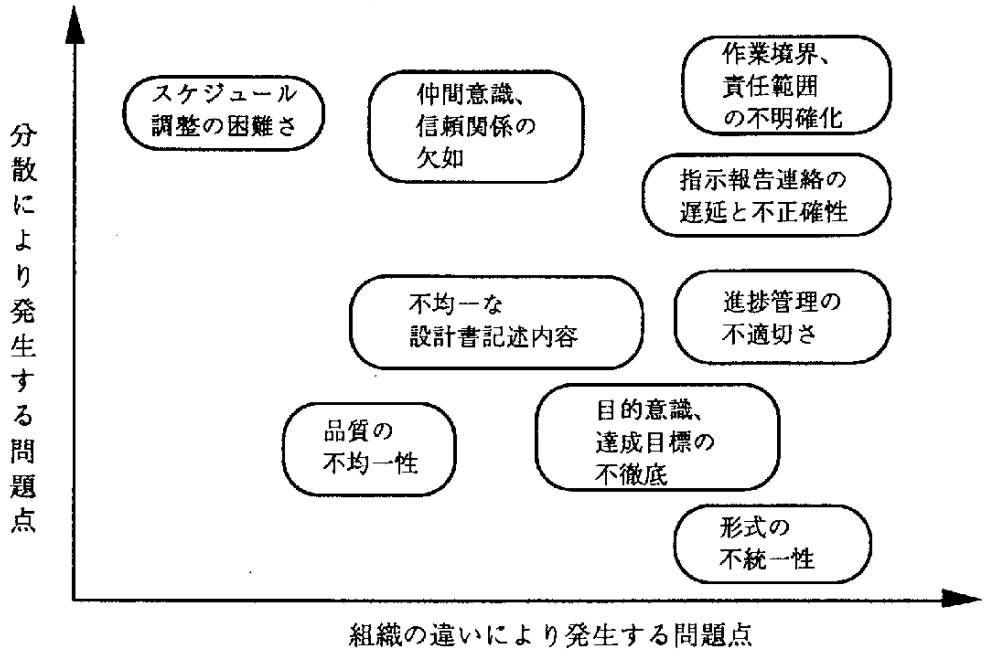


図2.2-3　分散と組織の違い

(4) 開発期間と集団の規模

集団の規模や開発期間の違いによっても問題を整理することができる。この関係を示したものが図2.2-4である。インフォーマルな情報の共有の問題は、集団の規模が小さいうちはあまり発生せず、規模が大きくなるに従って顕在化してくる。しかし、規模がさらに大きくなると、フォーマルな形で情報を共有する方法を導入するため、あまり問題となってはこない。例えば、大規模の集団で開発を行う場合には、あらかじめワークシートなどの記述方法や記述すべき内容を明確にし、情報交換する方法も検討しておくため大きな問題とはならない。仲間意識も中間規模での集団で一番問題になると考えられる。

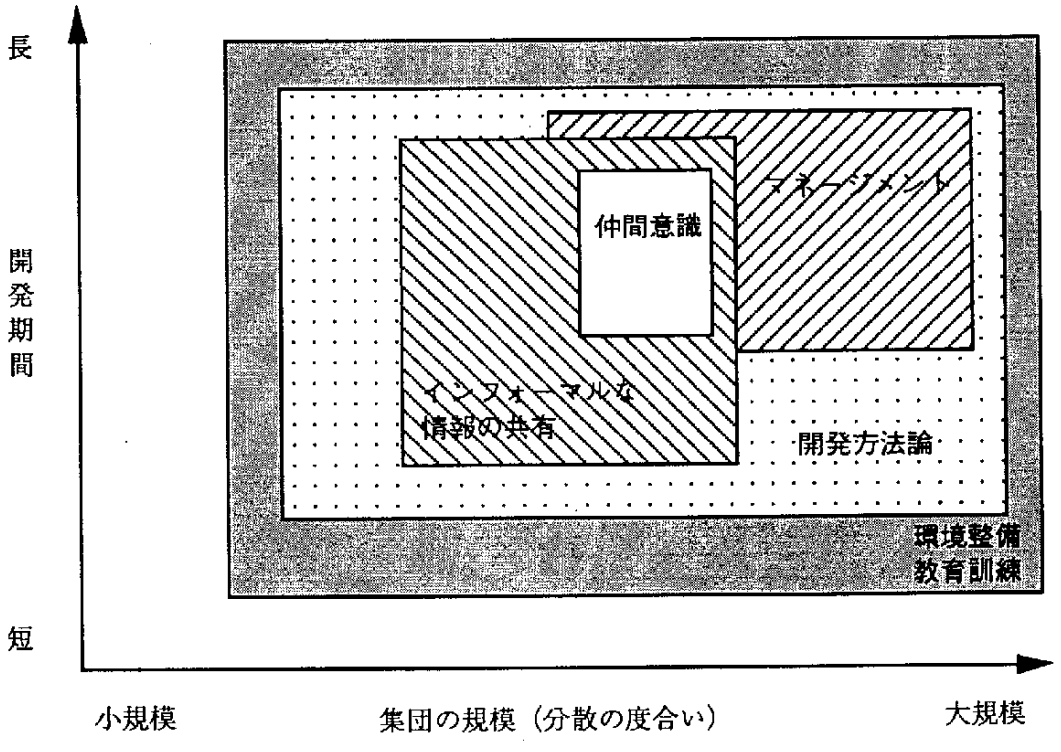


図2.2-4 開発期間と集団の規模

2.2.2 米国における調査例

米国の大規模協同研究組織であるMCC（Microelectronics and Computer Technology Corporation）のソフトウェア技術計画では、技術指向でソフトウェア生産性向上システムの開発を考えるのではなく、問題指向で考えるために、ソストウェア開発プロセスの上流工程において発生している問題の調査を実施した[Curtis88]。また、ソフトウェア開発の多くは、機械ではなく人間によって行われるため行動に中心を置いた調査を行った。図2.2-5に階層化された行動モデルを示す。

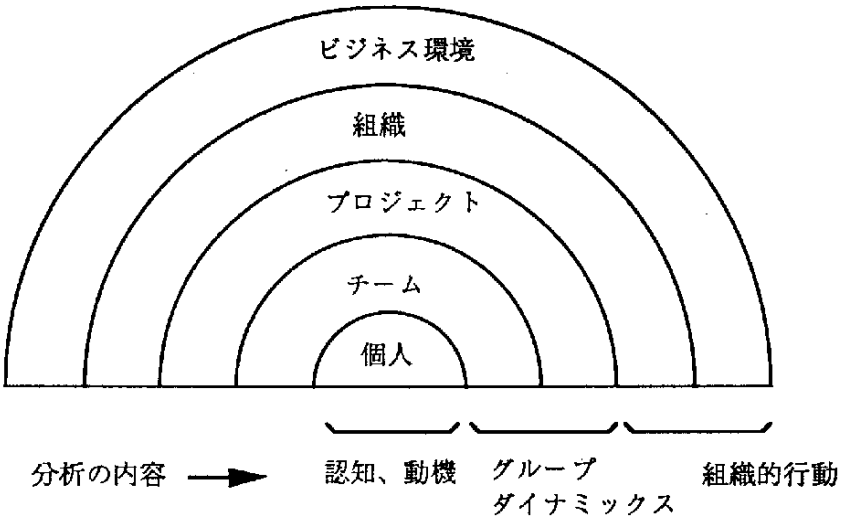


図2.2-5 階層化された行動モデル

調査は、九つの会社の17の大規模プロジェクトのスタッフに対するインタビューの形式で行われた。その結果は、次に示す三つの問題に集約されている。

- アプリケーションドメイン知識の浸透の困難さ
- 要求仕様の矛盾と変動
- コミュニケーションと調整の障害

ここでは、グループワークと関連が深いと考えられるコミュニケーションと調整の障害に関する問題を中心に先の階層に従って述べる。

(1) 個人レベル

作成されたドキュメントは、プロジェクトメンバー間でのコミュニケーションの一つの形である。しかしながら多くの回答者は、ドキュメントは、コミュニケーション媒体としては非常に弱いと指摘していた。また、ドキュメントが、プロジェクトメンバー間で必要なコミュニケーションを減少させたという証が認められなかった。こうしたことから、多くのドキュメントの形式は、開発工程全体を通して必要とされる設計情報を伝えるには不十分であることが分かった。

ソフトウェア開発で良いと考えられているコミュニケーションに関する活動の多くは、納期が迫った大規模プロジェクトのコミュニケーション支援には無力であった。

(2) チームレベル

ドキュメントは、要求仕様や設計への誤った理解を解消させる手段を提供していない。そのためチームレベルで必要とされるコミュニケーションにはほとんど役立たない。

共通のシステムモデルを作ったり、そのモデルを伝えるために多くの技法が開発されてきた。また、成功しているプロジェクトの多くは、開発するシステムについて議論するために共通の表現方法を利用していた。チームの観点で考えるとこれらの表現方法は、単なる静的なドキュメント以上に共通の言語としての価値を持っている。しかし、多くのエンジニアは、アプリケーションドメイン構造にマッチした表現形式を自由に選択することに難色を示した。

(3) プロジェクトレベル

レビューは、欠陥を発見するという目的以上にコミュニケーションの観点で効果があった。レビュー以外では、他のチームに介入することへの遠慮や調整に対する認識の欠如によってコミュニケーションがうまく行われていなかった。

プロジェクトチーム間でのコミュニケーションの欠陥は、チームあるいは組織をまたがって所属している人がいる場合解消されることがあった。こうした人の代表としてチーフシステムエンジニアがいる。彼らは、顧客の要望をチームが理解できるように翻訳することができる。こうした人は、良いコミュニケーションスキルをもつと同時に対面でのコミュニケーションをもつように心掛けていた。

システムの分割方法もコミュニケーションに大きな影響を与えていた。すなわち結合度の高いシステムの分解は、インタフェースを保持するために多くのコミュニケーションを必要としていた。しばしば、システムの分解は、論理的な観点のほかにスタッフの観点で行われていた。

(4) 組織レベル

組織は、通常大規模システム設計上の意思決定やそのレビューのためのフォーマルな手順をもっている。しかし、これらのフォーマルな手順は、その手順に含まれていないセクションからの設計上の問題を伝播するのには適していない。むしろ、組織間でのメッセージを伝えるにはインフォーマルなコミュニケーションが最も効果的である。設計上の重要な情報が組織外の人からもたらされることさえある。

大規模プロジェクトの場合、開発の各工程は異なったチームによって行われる。こうした場合、各工程間で中間成果物を引き渡す必要があるが、それらがうまく伝わらないという問題がある。後工程で設計の変更が必要になり、しかも前工程のチームが存在していないとき、問題はさらに大きくなる。

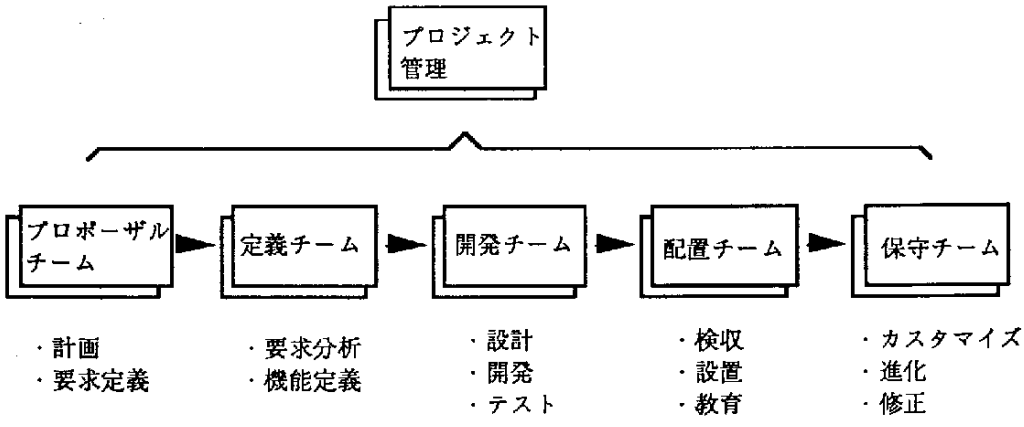


図2.2-6　開発ライフサイクルの工程とチーム

ドキュメントは、開発者との会話に代わるものではないが、後続チームでの主なコミュニケーションソースである。しかし、ドキュメントには、開発期間の関係上必要な情報がすべて記述されているわけではないという問題がある。システムの関する知識の継承は、同じスタッフをあるチームから次のチームに連続的に配置するといった方法で行われている。しかし、そうしたスタッフが辞めるとき問題が顕在化する。

(5)　ビジネス環境

アプリケーションやその環境を理解し調整するためには、顧客と開発者での絶え間無いコミュニケーションが必要になる。この顧客と開発者のコミュニケーションは、微妙な理解の相違を生じさせないために直接もつべきである。しかし、開発者とエンドユーザのコミュニケーションの距離は非常に遠い。

こうした調査結果を踏まえてMCCでは、ソフトウェア開発の上流工程を支援するモデルの研究やgIBIS等のツールの開発を行った[Belady92]。

【参考文献】

[Belady92] Laszlo A. Belady : 7 years of MCC's innovative software technolpgy program, American Programmer, pp.10-15(1992.1).

[Curtis88] Bill Curtis et al. : A field Study Of The Software Design Process For Large Systems, Comm.of ACM, Vol. 31, No. 11, pp.1268-1287(1988).

3．協調作業支援技術

3．協調作業支援技術

3.1　メールの構造化技術

3.1.1　構造化メール

電子メールはコンピュータによるコミュニケーション手段としては現状最も普及の進んでいるものといってよい。普及の理由はいくつか考えられる。ワークステーションやパーソナルコンピュータの小型化・低価格化によりオフィスの机上に一人一台といった割合で普及し、これらをネットワーク接続する技術が高度化してきたといったプラットホームの進化、それから、非同期型通信である電子メールは相手の都合に関係なくメッセージを送れることができ同報も容易であるといった利用のしやすさ、といった点が考えられる。

電話にとって代わろうとする電子メールであるが、受信側から考えると、メッセージが氾濫し不要なコミュニケーショントラフィックが増加したり、重要な情報が埋没してしまったりといった問題もある。俗に“ジャンクメール”と呼ばれるものである。

そこでメッセージを伝達するだけの通信手段としての電子メールではなく、電子メールによって運ばれるメッセージ自身をコンピュータが処理できるようなシステムの要求が強くなる。

コンピュータが電子メールのメッセージ自身を処理することとは、コンピュータがメッセージを解釈し、そのメッセージの意味に従った処理を行なうことである。例えば、同じ種類のメッセージをまとめて保存するとか、関係者に配送するとかといった処理が考えられる。こういった処理を可能にするためには、まずメッセージを解釈する必要がある。自然言語で書かれたメッセージを解釈するためには、厳格に意味を解釈しようとする言語解析的なアプローチと、メッセージをフィールドに分割しその内容により形式的にメッセージの属性を判断しようとする構造化によるアプローチがある。

Malone等はInformation Lens [Malone87] の中で Semi-Structured Message（半構造化メッセージ）の概念を提案している。半構造化メッセージの考え方は、グループワークの構造をコンピュータが処理可能な形でシステムに反映させる手段として重要である。ここでは「半構造化メッセージ」を、様々なタイプに識別できるメッセージであり、各々のタイプには複数のフィールドをもつ、と定義している。すなわちタイプの種類や各々のフィールドに関しては規定しておらず、そのような意味から厳密な構造化を緩和した「半構造化」であると言える。

半構造化メッセージの利点として以下が挙げられる。

- コンピュータによる自動処理が容易となる。
- 厳密な構造化に比べ情報交換の柔軟性がある。
- システムの拡張や種々のワークスタイルに対応が容易である。

3.1.2 構造化メールの事例

ここでは、構造化メール（正確には半構造化メールというのかもしれない）の事例を紹介する。

(1) Information Lens／Object Lens

① Information Lens [Malone87]

Information Lensは、半構造化メッセージの特徴を生かした知的情報共有システムである。半構造化メッセージの考えに基づきメッセージを構造化（Maloneはあくまでも半構造化といっている）し、ルールベースのエージェントと組合せることで、電子メールやニュースの自動フィルタリングに応用した。

Information Lensの特徴を以下に述べる（半構造化メッセージに関しては前記したのでここではふれない）。

・ テンプレートといった概念で半構造化されたメッセージを作成する。

半構造化されたメッセージはタイプを持つ。各タイプごとにさまざまなテンプレートを用意し、各タイプのフィールド値を設定させることにより半構造化されてメッセージを作成する。フィールド値は「既定値」かマウスによる「内容選択」かまたは自身で設定することにより与えられる。フィールド値は知識ちして保存され後の「内容選択」に活用される。図3.1-1はテンプレートの例である。

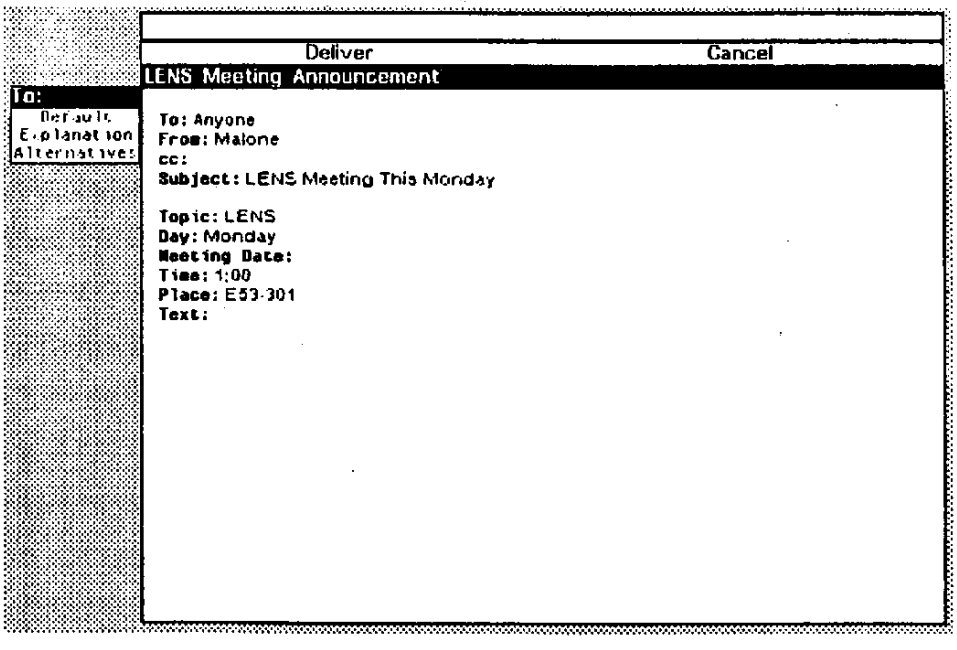


図3.1-1 半構造化メッセージのテンプレートの例

・　半構造化されたメッセージを自動処理するためのルールを指定する。

ルールはIF-THEN形式で記述する。IF部においてタイプやフィールド値の判断がされTHEN部の動作をするといった形式が一般的であるが、LispCodeというルールも用意されており、このルールが指定されるとLISP関数の呼び出しが可能となり、複雑な処理の記述も可能となる。図3.1-2はルールの例である。

・　メッセージタイプを継承することが可能である。

メッセージタイプにフィールドや特性を追加し新たなメッセージタイプが生成されることはよくある。この時テンプレートの構成が親子関係を表すようなネットワーク構造として表現され、子のテンプレートが親のテンプレートの属性を継承する機能を用意することによって、重複する情報を記述することが不要となりメッセージの作成は容易となる。また、これを応用し議論の整理への適用も考えられる。

(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3.1-2　自動処理のためのルールの例

② Object Lens[Malone88]

Object LensはInformation Lensのコンセプトを以下のように拡張したメッセージ処理システムである。

・　情報をフォーマルに表現しコンピュータによる自動処理を可能にする。

・　インフォーマルな表現も行い人間の処理を容易にする。

・　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を融合する。

これらを実現するには、システム中の情報や知識は可視的かつ可変的にユーザに提示される必要がある。Object Lensでは「半構造化オブジェクト」と「半自律エージェント」といった機能を実現している。

・　半構造化オブジェクト

メッセージ、人、会議、部品といった仕事に関連する種々のものをオブジェクトとして表現し、各オブジェクトがタイプとフィールド値をもっている。Information Lensの半構造化メッセージと同様の概念である。ただし、オブジェクト間をリンク構造で関係付けられるため、オブジェクトの関連（例えば同一の仕事をしているメンバー）を直接検索することが可能である。

・　半自律エージェント

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できる。エージェントはメールの到着や、指定した時刻、イベントにより起動し、指定されたオブジェクトの集合に対してルールを適用させる。

エージェントは自動的に動くが、ルールはユーザにより作成され、また、ユーザにオブジェクトを提示し処置を求めることがあるといった理由から、「半自律エージェント」と呼んでいる。

(2)　The Cordinator[Winograd89]

Winogradは、図3.1-3に示すビジネス社会における会話のプロトコルモデル（Conversationモデル）を提唱した。会話は9種の状態とそれらの間を流れる8種の属性を持ったメッセージから構成される遷移図として表されるというものである。

The Coordinator は、この会話モデルに基づいて電子メールにより交わされる人々の会話の状態遷移を管理するシステムである。すなわち、この会話モデルに基づき、要求・約束・拒否等のメッセージ属性（種別）を電子メール中に構造として取り入れることにより、会話の状態遷移をシステムが追跡管理できるようにしている。

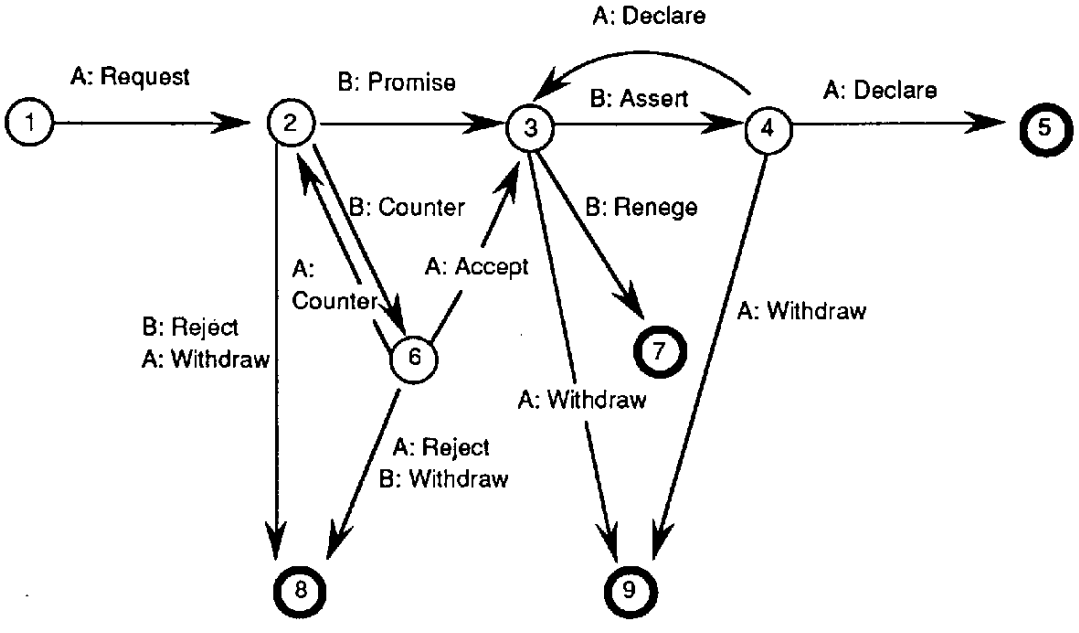


図3.1-3　会話モデル

(3) D^2／エージェントメール[貫井92]

D^2／エージェントメールは、非同期型のコミュニケーションである電子メールのバックエンドに、メールの送受信の際に人が行なっていた手作業を代行する「エージェント」を配し、メッセージの自動処理を支援するシステムである。D^2／エージェントメールでは、エージェントを以下の機能を有するものと定義している。

- 代行：ユーザの代わりに仕事を行なう機能
- 依頼：他のエージェントに仕事を依頼する機能
- 協力：相互に問い合わせを行ない協力する機能

D^2／エージェントメールのエージェントは機能は、サービスの規模、汎用性といった観点から、基本機能エージェント／複合エージェント／プロシージャといった3階層の構造から成る。

基本機能エージェントは主にメッセージ操作に関するサービス機能であり、subjectやdataの抽出やメッセージのcompare、mergeといったものが用意されている。複合エージェントは基本エージェントを組み合わせたサービス機能である。複合エージェントの代表的なサービスを以下に示す。

- filtering： subjectのキーワードや差出人、data等によりメッセージを分類する。
- routing： 関連する人にメッセージを転送する。
- executing：関連するアプリケーションを起動する。

プロシージャとは仕事の流れそのものであり、基本エージェントや複合エージェントの組み合わせで構成される。複合エージェントやプロシージャは、IF-THEN形式のスクリプトで各ユーザ単位に記述される。

D^2／エージェントメールは、UNIXメールをベースとしており、エージェントが種々の判断をするための情報をメッセージヘッダに埋め込んでいる。

図3.1-4にシステム構成を示す。

3.1.3 構造化技術の課題

メールの構造化技術について述べてきた。今後の課題を以下に列挙する。

(1) 構造化の厳密さと汎用性の両立

メールの構造化はメッセージの自動処理が目的である。厳密に構造化を行うことにより、正確な自動処理は容易となる。反面、多くの仕事のモデルに対応させるための汎用性に欠ける恐れが多い。汎用性を保ちながら、正確な処理を可能とする構造化の厳密さを実現する必要がある。

(2) 構造化の範囲

メールは大きくヘッダとボディーから成る。ヘッダは発信人や経路制御等のプロトコル上の情報であり、構造化された情報と考えられる。しかし、プロトコルで規定された以上のことを操作すること

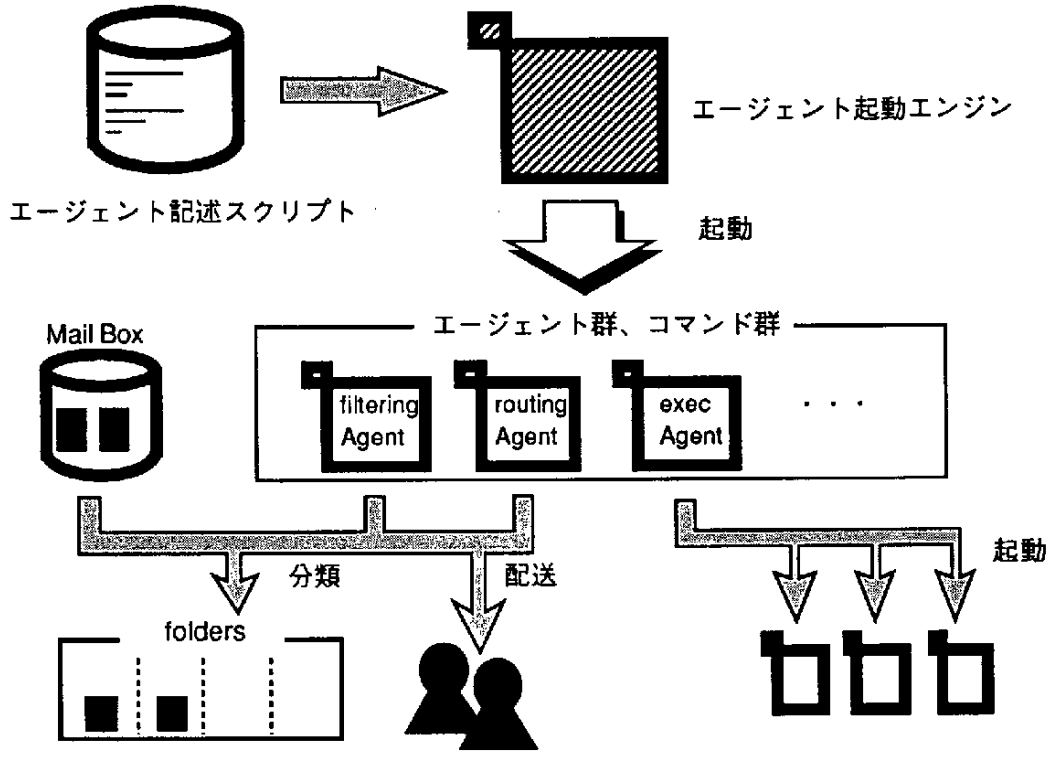


図3.1-4　D²/エージェントメールのシステム構成

はインターオペラビリティを保つ上では問題となる（リザーブ領域を操作しているシステムもある）。したがって、ヘッダがもつ情報を構造化と考え形式的な判断に適用する。メールの持つ意味を判断するためにはボディーの解釈が必要であり、そのためにはボディーの構造化を行わなければならない。

自動処理の範囲を考慮し構造化の範囲を明確にすることが課題となる。

(3)　ユーザインターフェース

前記したようにボディーの構造化を行う場合、ユーザにメールの本文を構造化したような文章で、意識して作成させるべきではない。また、メールを受信したユーザがメールを見る場合も同様である。すなわち、構造化された情報を扱うのはシステム内部だけの問題であり、それを意識させないユーザインターフェースの構築が課題である。

(3)　標準化

構造化メールの効果を発揮させるためには、少なくともEnd-To-Endでは同様の構造化メールを利用する必要がある。メールの範囲が広域化している（さらに広域化する）のは明白であり、そのような意味からもメール構造化に関する標準化が課題となろう。

【参考文献】

[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,p p.1502-1508(1989).

[Malone87] Malone,T.: Semistructured Messages Are Surprisingly Useful for Computer-Supported Coordination, ACM Transactions on Office Information Systems, Vol.5, No2, pp.115-131(1987).

[Malone88] Malone,T.: Object Lens:A "Spreadsheet" for Cooperative Work, Proc. CSCW'88, pp.115-124(1988).

[Winograd88] Winograd,T.: Where the Action Is, BYTE, pp.256-257(1988.12).

[貫井92] 貫井春美: ソフトウェア分散開発支援システム D^2, 情報処理学会グループウェア研究グループ研究報告, 92-GW-3, pp.41-48(1992).

3.2 ワークフロー制御技術

3.2.1 ワークフローとプロセスプログラミング

プロセスモデル、プロセスプログラミングといった概念は1980年代にArcadiaプロジェクトの中から発生した概念である。ここでは、「プロセスプログラミングとはソフトウェア開発・進化に内在する諸活動を明示的にかつ手順的に書き下そうとする試みであり、要求定義・設計・実現・テストのすべての開発活動を対象とする」と言っている。

Arcadiaプロジェクトの推進者であったOsterweil教授は、1987年米国カリフォルニア州モンタレイで行なわれたThe 9th. ICSE(International Conference on Software Engineering)の基調講演の中で、プロセスプログラミングといったパラダイムを追求することの意義を次のように述べている[Osterweil87]。

- プロダクトから別プロダクトを生成していく問題解決の過程を「もの」として扱うことが可能となる。
- 上記により自動化支援への手がかりとなる。
- 再利用に対して、プロセスの再利用すなわち設計上の意思決定の再利用へのアプローチとなる。

プロセスプログラミングに関する研究・開発は、

- 観測とモデル化
- プロセスプログラミング言語の設計
- プロセスプログラミング支援環境の開発
- ソフトウェアメトリックスの導入

といったアプローチをとることが多く、Osterweilもそのような研究目標を掲げていた。

このように、ソフトウェア開発過程を形式的に表現する概念として「プロセスプログラミング」が提案されている。一方では、グループによる協同作業を支援する概念として「グループウェア」や“CSCW”（Computer-Supported Cooperative Work）が提案されている。プロセスプログラミングに類似するグループウェアの要素技術として、「グループワークモデル化技術」[石井89]が挙げられる。

仕事の流れやオフィス業務の手順を明示的に定義するといったワークフローやオフィスプロシージャといった考え方は、まさにプロセスプログラミングの概念に沿っている。プロセスプログラミングは抽象化された概念であるが、ワークフローやオフィスプロシージャはDomain Specificにそれを具象化しているものであると考えられる。ここでは、ワークフローとオフィスプロシージャを総称しワークフローと呼ぶことにする。

3.2.2 ワークフロー制御

前記したようにワークフローは仕事の流れを手順的かつ明示的に記述した一種のルールである。協調作業支援から考えると、記述したルールを実際の支援システムに反映させることが必要となる。すなわち、システムがワークフローに沿った動作をすることにより、業務に密着したシステムとして仕立てられるようになる。

現状では、電子メールのようなコミュニケーションシステムにワークフロー制御を適用するといったアプローチが多い。電子メールによるメッセージの経路をワークフローにより制御するといったものである。この場合の利点は、仕事の進行をシステム側で追跡することが可能なため、従来では困難であった進行状況の問い合わせが可能となることである。

また、ワークフローに求められる最も重要なものは、仕事のモデルを変更することで他の仕事への対応が容易であるといった点である。言い換えれば、種々の仕事をモデル化できる記述能力がワークフローの生命線ともいえる。

ワークフロー制御の方法は以下の二つに大別されると考える。

(1) スーパーバイザがワークフローを解釈し仕事を配送する方式

スーパーバイザがワークフローを解釈し、該当する仕事のステップ（仕事を行うプロセスと考える）を呼び出す。「呼び出す」とは、その仕事を実行するプロセスに仕事の内容とデータを送ることである。呼び出されたプロセスは自身が行う処理を終了するとスーパーバイザに結果を返す。スーパーバイザはワークフローの次のステップのプロセスにその結果をデータとして与え仕事を指示する。もちろん作業者がスーパーバイザの存在を意識することがないようにしなければならない。

ワークフローデータは全体で一つを管理する形態となる。

(2) 仕事のステップ単位にワークフローを解釈し仕事を実行する方式

仕事の流れの各ステップおいて、インプットされてきたデータにより自身が行う処理とその結果により次に進む経路を判断する。後のステップでの処理は後のステップで解釈する。すなわち、ワークフローを個々に所有し解釈を行い、個々のワークフローから全体のワークフローを形成する。これは、オブジェクト指向の考え方にも通じる。

前者は仕事全体の把握が容易なため管理的側面からの長所が認められる。例えば、仕事の進行をシステム側で追跡することが容易であり、人的資源の配置を含めた負荷分散等にも応用できる。しかし担当作業者が全体の関連を把握することが困難であるといった短所もある。また後者は個々の方法に合わせたワークフローが形成されるといった長所はあるが、仕事の追跡が容易でなく、一か所への負荷集中を招く恐れもある。

したがって、スーパーバイザが仕事全体のワークフローをステップの関連程度に管理し、各ステップでは、個々の方法に合わせて個々にワークフローを管理するといった形式が望まれる。

3.2.3 ワークフローの事例

ここではワークフローの事例として二つのシステムを紹介する。

(1) ECF [Karbe90]

表紙に作業者の宛名が記載されており、その作業者が作業する文書が入っているフォルダを、メッセンジャサービスによりオフィスの各作業者間を順次伝達することによってオフィス業務の効率化を図る仕組みがある。これを循環フォルダ（CF : Circulation Folders）と呼んでいる。

CFの問題は、伝達時間がかかる、宛先の作業者が不在の場合の対策がない、関連する他の業務との関連が明確でないといった点である。

これらの問題を解決するため、ProMInanD（Extended Office Process Migration with Interactive Panel Displays）プロジェクトではCFの代わりとなるべく電子循環フォルダ（ECF : Electronic Circulation Folders）を開発した。

ECFは、伝達仕様・状態情報・識別子・他ECFとの関係・作業者の進捗状況から構成される説明部と、作業に必要な文書やフォームを持つワークパート・作業者が編集可能なオプションフォルダ・作業者が追加した文書を入れるためのオプションフォルダから構成される内容部から成る。

ECFは説明部に記述される伝達仕様によって作業者間を伝達される。伝達仕様を決定するパラメータとして、オフィス内の役割（オフィスロール）と実行者の役割（パフォーマロール）が定義され、これによってどの作業者がどの役割を果たすことができ、それによってどのECFのインスタンスが生成されるかといったことから伝達仕様は決定される。

ECFでは上記した説明部とオフィスロール・パフォーマロールといった二つの役割定義を変更することで他の業務の手続きを記述することが可能と考える。

(2) DOMINO [Kreifelts91]

DOMINOは、オフィス内の各個人の物品購入願いや出張・休暇願い等を管理者に提出し、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ことを目的としている。

DOMINOは、図3.2-1に示すようにオフィスにおける仕事をペトリネットの考えに基づきモデル化し、MacとSUNをEthernetで接続したネットワーク上にシステムを構築した。図3.2-2 はDOMINOシステムの構成である。ホストマシンと位置付けられたSUN上にはMediaterと呼ぶエージェントが動作している。Mediaterは、電子メールの疑似ユーザとして各ユーザとメールの交換を行なう。Mediater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、これを解釈することによって共同作業の処理の流れに沿ったメールの送信を行なう。

ECFでは、データベースは集中管理するが処理の流れの解釈は各ECFが行なうといった、どちらかというと分散型の処理形態となる。これに対しDOMINOは、処理の流れの解釈を行なうスーパーバイザ（Mediaterと呼んでいる）を配した集中型の処理形態と考えられる。

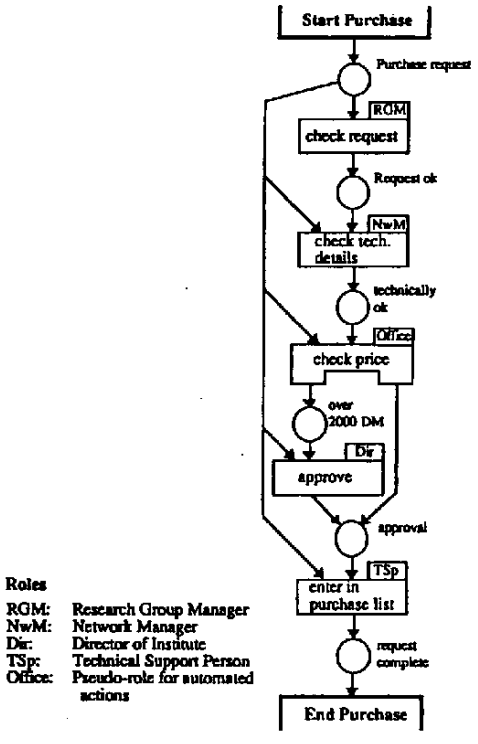


図3.2-1　DOMINOオフィスプロシージャの例

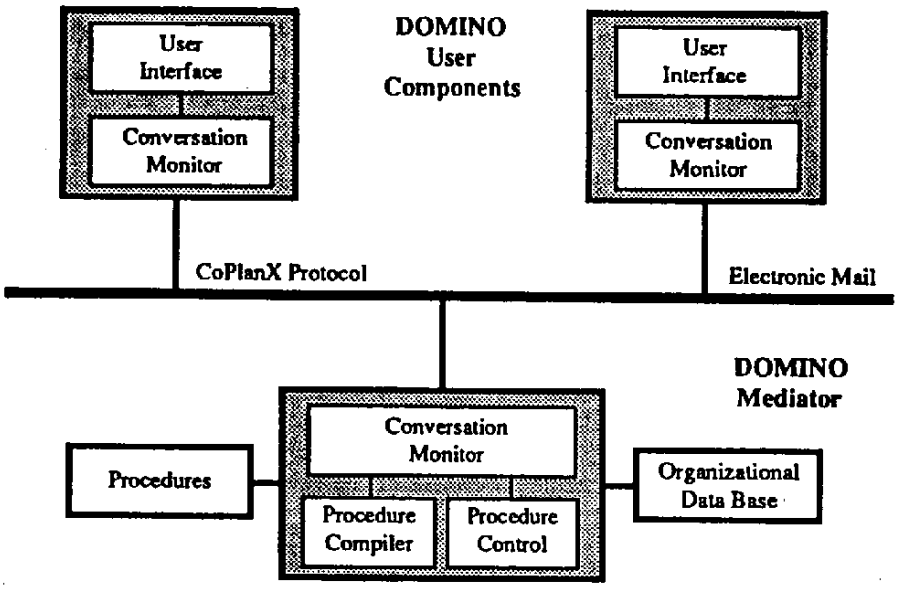


図3.2-2　DOMINOシステムの構成

3.2.4 ワークフロー制御の課題

ワークフロー制御技術の今後の課題に関して列挙する。

(1) 拡張性（記述能力）

前記したが、ワークフローの利点として重要なものは、仕事のモデルを変更することで他の仕事への対応が容易であるといった点である。これを実現するために、DomainSpecificに具象化された記述能力に加え、他の仕事に対応可能な拡張性を備えることが必要となる。ワークフローの個々を部品として捉えたソフトウェアの再利用技術の導入もひとつの鍵と考えられる。

(2) 支援システムとの融合

現状では電子メールへの組み込みが主であることは前記したが、他の支援システムへの組み込みを考える必要がある。電子メールのメッセージ制御をワークフローで行うということは、仕事における指示・報告・連絡・データ・成果物等の流れを制御することであり、仕事自身の手順を支援することではない。例えば、指示された仕事を行うためのガイドラインを出し、仕事全体としてのフェーズを合わせることを支援するような仕組みが望まれる。それには、知識ベース等の他の支援システムとの融合が必要となる。

(3) 緩やかなワークフロー管理の実現

ワークフローにより仕事の流れの追跡が可能となることは大きな利点ではあるが、管理的側面の重視に偏ったシステムでは問題がある。Plan-Do-See-Checkといった管理サイクルに基づいたワークフロー制御を考え、See-Checkばかりでなく、Plan-Doを重視した緩やかなワークフロー管理の実現が課題である。

【参考文献】

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

[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).

[Karbe90] Karbe,B.: Support Cooperative Work by Electronic Circulation Folders, ACM SIGOIS Bulletin, Vol.11, No.2, pp.109-117(1990.4).

[Kreifelts91] Kreifelts,T.: Experences with the DOMINO Office Procedure System, Proc. ECSCW'91, pp.117-130(1991).

3.3 会話支援技術

電子メールや電子ニュースが我が国でも普及してくるにつれ、これらの手段を上手に使って、地理的に離れているなどの理由で今まではできなかったような人々の間で議論が行われ、成果をあげている例が多くなってきている。しかし、一方では些細なことから誤解が誤解を生み議論が混乱してしまい、最悪の場合は互いに中傷しあうほど事態が悪化し、ついには参加者の間の信頼関係にひびがはいってしまうような場合があることを、時々耳にする。電子メールや電子ニュースなどの電子メディアが人々の協調作業に貢献できる潜在性は高いと考えられる一方、このように協調作業を妨害するような事例があるのも事実であるが、これはなぜだろうか？コンピュータをベースにした新しい情報システムが協調作業環境に与える影響は大きいと考えられる。新しいシステムが協調作業を支援するように働く場合もあるが、意図に反して協調作業を妨害する可能性は常に存在すると言えよう。

この節では協調作業環境における人の言語活動の重要性に着目し、協調作業を支援するシステムの設計において、採用すべき新しい視点を導入を提案している文献[Winograd86]に沿って、会話支援技術について紹介する。

言語理論が示唆するように言語は構文論（syntax）、意味論（semantics）、語用論（pragmatics）の観点から分析することができる。構文論では言語の構造（単語などの構成要素と句、文などの生成規則＝文法）が論じられる。意味論では構文論に沿って生成された文とその文が指示する対象や真偽が論じられる。語用論では構文論、意味論的に「正しい」文の使用に関する問題が論じられる。文を発信する人がその文を使うことで何を表現しようとし、受け手が何をメッセージとして受けとめるか、が問題となる。会話支援技術の観点では構文論、意味論のレベルはもちろん大切なのであるが、語用論のレベルが決定的に重要になる。子供が「喉が乾いた!」と言えば、「だめ、さっきジュースを飲んだばかりでしょ！」と答えるお母さんは多いのではなかろうか。

3.3.1 Speech-Acts 理論と会話理論（会話の語用論的考察）

この節では会話の語用論的考察である、Speech-Acts 理論といくつかの会話理論を紹介する。人の発言の表現する内容を、SearleのSpeech-Acts（発言動作）理論は発言が意図する人の行動の観点から次の５種類に分類している。

(1) 主張的（Assertive）

表現された命題が真であることに発言者が自らを束縛する（コミットする）。

(2) 指示的（Directive）

受け手に何かをさせようとする。これには質問（受け手が主張的発言動作を行うよう要請する）と命令（受け手が何かを実行するよう指図する）がある。

(3) コミット的（Commisive）

発言者が将来の行動にコミットする。

(4) 宣言的（Declarative）

発言動作の命題的内容と現実世界の対応付けを行おうとする。（例:結婚の宣言）

(5) 表現的（Expressive）

ある事態に対する心理的状態を表現する。（例:謝る、誉める）

この理論に対してWinogradは次のようにコメントしている。

(1) 発言の意味を発言者、受け手ともある背景の元に解釈している。同じ発言が異なる背景の元では別の意味に解釈される。

(2) 指示的発言とコミット的発言は常に将来の行動に関係している。違いは発言者の行動なのか、受け手の行動なのかにある。

(3) 発言動作は公衆での宣言により効果を発揮する。例えば、受け手に聞こえない謝罪は謝罪とはならない。

Speech-Acts理論は主として単一の文のレベルを対象としていると考えられる。通常の会話では発言が参加者の間でキャッチ・ボールのように行われる。このような会話の構造を個々の発言動作を単位として構成することにより、会話を理解しようとする会話行動理論がFloresとWinogradにより提案されている（図3.1-3参照）。

このモデルでは最初の発言者Aによる要求（Request）は発言者A、受け手Bの双方により、ある種の充足条件を持つものとして解釈される。受け手Bはその充足条件を受け入れことができると判断した時は約束（Promise）し、受け入れることができない時は拒否（Reject）する。

Winogradは基本会話行動のほかに、基本会話行動を補うものとして次の種類の会話を識別している。

(1) 明確化会話（Conversation for Clarification）

会話行動の充足条件の解釈の破綻が予想される時に行う、会話の充足条件の明確化を行うための会話。

(2) 可能性会話（Conversation for Possibilities）

会話行動を生み出すような会話。

(3) 方向付け会話（Conversation for Orientation）

将来の会話を理解する基本となる共有の背景を生成するような会話。

言語行動の意味はそれが行われる状況などによって異なる。Winogradは異なる状況として次のものを指摘している。

(1) 個人（Individuals）

言語行動の意味は人により異なるが、同じ人の言語行動の意味は比較的安定していると考えられる。

(2) 役割（Roles）

組織の中では特定の人の違いよりも、人の組織内での役割により言語行動の意味が異なると考えられる。

(3) 職業（Profession）

それぞれの職業にはその職業に独特の専門用語などがある。職業が異なる人の間では同じ言語行動が異なる意味を持つと考えられる。

(4) 位置（Locations）

時間的、空間的位置が異なると同じ言語行動が異なる意味を持つと考えられる。

3.3.2 会話の意味論的考察

語用論では言葉の使用が論じられるが、意味論では言葉の文字どおりの意味が論じられる。語用論的考察で示されたように言語は解釈される背景により意味が異なってくる。特定の背景の元では言語のある形式はある特定のものを指示する。言語のある形式のものは違いのある複数のものの一つを指示するものと考えられる。言語の指示集合、その変化、指示方法、比喩的指示（メタフォー）についてWinograd は次のように指摘している。

(1) 指示集合

言語が指示するものの集合は適用領域の標準的な集合により決定されてしまうわけではなく、俗に「ローカル・ルール」と呼ばれるものが含まれる。

(2) 指示集合の変化

指示集合は指示できないものが発生するたび常に更新される。コンピュータ・システムの失敗のよくある理由は、コンピュータ・システムに組み込まれている指示集合が設計者が設計時に観察して知りえたものに固定されていて、進化する余地が残されていないことである。

(3) 指示方法

従来の自然言語の意味論的研究は言語の形式と指示集合との写像関係を固定的にとらえているものが多いが、最近は言語の使用状況の意味論への影響に焦点をおく研究がある。

(4) 比喩的指示の使用

メタフォーをシステムで採用する場合、メタフォーとして採用した指示集合と実際に比喩的に指示されるものの集合の間の整合性に対する留意を怠るとシステムの使用者は混乱する危険性がある。

3.3.3 会話の構文論的考察

Winogradは構文を通常の言語におけるそれよりも広く考えている。必ずしも自然言語を含まない、言語的相互作用における出現頻度の高い形式の規則性を広義の構文と考えている。このように考えるとデータベースのリレーション（表）や伝票、帳票、スクリーンの入出力フォームなどは構文要素となり、それらの流れは文を構成することになる。

この意味で、例えば構造化分析手法など、従来のシステム分析手法は構文論的なレベルでの分析と言え、意味論的、語用論のレベルでの分析は行われていないことになる。意味論的、語用論的レベルの分析を経ずして設計されたシステムが協調作業環境に導入された場合、最悪の場合導入前に存在していた協調情報＝協調言語行動が妨害され、協調作業を支援する筈のシステムが実は協調作業を妨害することになってしまう危険性がある。

3.3.4 会話支援技術の課題

Winogradの言語行動視点によれば、人の協調行動は会話の語用論のレベルでの正否に大きく左右されることになる。顔の表情、声の調子、コンピュータ・システムの入出力画面なども含めた広義の言語の語用論のレベルでの分析が重要となる。協調作業環境への新しいシステムの導入は設計者の意図（の有無）に関わらず、少なからずその環境の中の人の行動パターンに大きな影響をもたらす。

電子メールや電子ニュースの出現は確かに協調作業環境に変化を与えてきたように思われる。電子メールや電子ニュースでの議論の場合、現状ではこれらのメディアは語用論の観点から不利であることを使用者は意識する必要があると思われる。

従来のシステム分析は、Winogradの視点によれば、主として構文論のレベルに留まっており、システムが導入された場合、それまでの語用論のレベルの協調情報＝協調言語行動がブロックされてしまい、その結果として人の協調行動が妨害されてしまう危険性があると言えよう（近年注目されているエンド・ユーザ・コンピューティングはEDP部門や専門のソフトウェア会社により開発されたシステムへの不満の表れと理解できるが、Winogradの視点はこのことに一つの説明を与えるものと考えられる）。

協調作業を支援するシステムを設計するには協調作業環境の言語行動の語用論のレベルまで含めた分析を行う必要性は指摘されているが、具体的にどのようにそれを行うかは今後の研究を待つ必要があると思われる。

【参考文献】

[Winograd86] Winograd, Terry: A Language/Action Perspective on the Design of Cooperative Work, CSCW'86 Proceedings, pp.203-220(1986.12.3)

3.4 議論支援技術

協調作業環境では参加者が種々の問題について議論することが多いと考えられる。議論での問題の一つは焦点が定まらず議論が発散してしまうことであろう。議論の過程を上手に記録することも課題であると思われる。途中から参加する人が議論のそれまでの過程を知ることができるであろうし、最初から参加している人も議論の過程を振り返ることができるからである。議論の過程に緩い形式を導入することにより、議論のポイントを明確にし、同時に議論の過程の記録を残すようにする研究が行われている。

この節では議論支援技術としてIBIS（Issue Based Information System）モデル [Rittel70]、IBISモデルをベースにしたツールgIBIS（graphical IBIS）[Conklin88]、IBISモデルにプロセス情報と成果物情報を付加したPottsのモデル [Potts89] を紹介する。

3.4.1 IBISモデルとgIBISツール

IBIS（Issue Based Information System）モデルは、定式化が困難な問題に対処する方策としてRittelにより考案された[Rittel70]。

定式化が困難な問題であっても、とっかかりとして、関係者は自らが感じる関心事や疑問点を「論点」（Issue）として提示することは可能であろう。論点が提示されると関係者は、論点に対して解答となるであろう考えを「見解」（Position）として提示し、さらに賛成する理由／反対する理由などを「議論」（Argument）する、という風に進めることができよう。

MCCのConklinらは設計初期段階の、非公式な形でしか存在していない設計理由などを記録・把握することを目的にして、gIBIS（Graphical IBIS）という実験システムを開発した[Conklin88]。設計の初期段階はそのプロセスが一般にあまりよく理解されておらず、支援ツールは少ない。gIBISは設計理由の内部表現にIBISモデルを、ユーザ・インタフェースにハイパーテキストを採用した実験システムである。ConklinらはgIBISの開発とほぼ平行して、MCCの研究プロジェクトで開発されるシステムの設計にgIBISを利用してもらい、いくつかの知見を得ている。

(1) IBISモデル要素の使用頻度分析

ConklinらはgIBISの約1年間の使用結果を分析し、IBISモデルの実体、関係の使用頻度の分析をしている。しかし、母集団が小さいので断定的なことはまだ結論できないであろう。

(2) IBISモデルの有効性

利用者の評価はおおむね良好である。思考の不完全さや矛盾を発見するのに役立ったり、仮定や定義が明らかになる傾向がある。一方、目標、意思決定結果など表現されにくい点もいくつかある。

(3) IBISモデルとそれを支援するツールの相乗作用

IBISモデルは実体＝関係図の表現では簡単であるが、実際のインスタンスのレベルではツールの支

援がなければ扱いきれない。ハイパーテキスト機能を持つ、高速なgIBISというツールがあってはじめてIBISモデルが効果を発揮したといえる。

(4) 未熟成段階の議論への早期適用の危険性

まだ微妙な段階にある論点に早期にIBISモデルを適用すると、思考が不明瞭になったり、混乱したりする危険性がある。

(5) 最終結論の表現が困難

論点に対して最終的にある見解が選択されることになるが、その理由がうまく表現できない。

(6) 議論の流れなどの分脈の表現が困難

ハイパーテキストシステムに共通する問題であるが、複数のノードにまたがった思考の筋の表現が困難である。

(7) メタ的な議論の表現が困難

議論すること自体の適切性に関するようなメタ・レベルの議論の表現が困難である。

(8) ブラウザ空間の配置の問題

ノード（実体）とリンク（関係）で表現されるネットワークがブラウザ領域に表示されるが、その空間配置に意味を持たせるように使用する人がいた。無意味に配置したのでは使用者はブラウズするのに困難を感じる。

(9) 時間的変化に対する問題

IBISモデルのネットワークの中には、時間が経過するにつれ、経過的意味しか持たないような部分が出てくる。現在、熱心に議論されている論点とこれらの区別が必要になってくる。

IBISモデルは比較的簡単なモデルである。簡単である故に使用者はモデル自体は容易に理解でき、また使用できるというメリットがある。強制される構造が簡単である故に、議論の過程を理解しやすいというメリットがあると考えられる。しかし一方、モデルが簡単である故に議論の流れとか、文脈とかといった点の表現・把握が困難であるというジレンマがこのモデルにはあるように思われる。表現・把握が困難な点があるという難点は、IBISモデルに単純に実体や関係を追加することでは克服できないであろう。モデルが比較的簡単であることからくるメリットを生かしながら、このモデルをどのように進化させて行けるかが今後の課題になるであろう。

単純な表現を通してその背後の構造や意味を抽出する技術が必要とされる点でＫＪ法などに共通するメカニズムの研究が一つの方向であろうと思われる。IBISモデルをオブジェクト・レベルであると考えるならば、メタ・レベルのモデルを２段階程度追求することも研究の別の方向であるかも知れない。

3.4.2 Pottsの設計手法表現モデル

Pottsは設計上の意志決定の記録や設計手法の表現にIBISモデルを応用している[Potts88, Potts89]。設計意思決定の記録に際しては、成果物を表現するために「成果物」（Artifact）という実体をIBISモデルに追加している。設計手法の表現に際しては、さらに設計過程を表現するために「ステップ」（Step）という実体を追加している。図3.4-1はPottsの設計手法表現の汎用モデル（Generic Model）である。

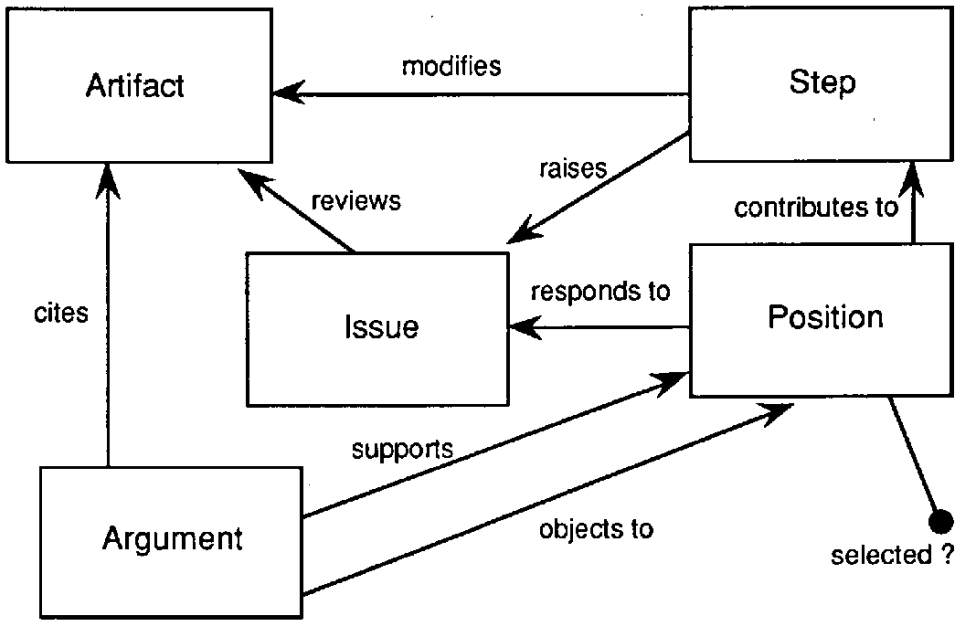


図3.4-1 Pottsの設計手法表現の汎用モデル[Potts89]

Pottsは設計手法が持つべき要件として次のものをあげている。

(1) 種々の分析に適した標準の表現法を提供すること

(2) 技術的進捗を統制するために役立つこと

(3) 標準手順とかガイダンスといった局部的ヒューリスティクスを提供すること

このうち(1)は現在のCASEツールが主として支援しているものであるが、設計者の設計活動を支援するためには(2)、(3)は設計の過程の面が重要であるとPottsは主張している。Pottsのモデルの実体「ステップ」は(2)、(3)を表現するために追加されたものであると言えよう。

Pottsはこのモデルを用いてJSD（ジャクソン法）を表現してみて次のように述べている。

(1) IBISモデルは表現法として適切である。

(2) 設計実践の場では成果物を中心にした作業パターンが多い。

(3) 成果物をベースにした表現法は粗いものになるが、これは実際的である。

(4) 複数の実体をまとめる実体をモデルに取り入れる必要があるかも知れない。

(5) 版管理のメカニズムを導入する必要があるかも知れない。

(6) 完了／未了を識別するため「論点」（Issue）や「ステップ」（Step）の完了／未了状態を表現するようにしてあり、自動通知機能や自動優先度割当機能を実現できる可能性がある。

(7) 設計意思決定の出現頻度の高いパターンにはあらかじめテンプレートを用意しておいて設計者の負担を軽くできるだろう。

設計過程表現モデルであるPottsのモデルは、IBISモデルの応用に成功した例と言えよう。Pottsのモデルは成果物を軸に表現する方法をとっており、IBISのみの場合と比較して、全体構造（大きな流れ）の把握の難点が緩和されているのがIBISモデル応用の成功の一因であろうと思われる。

【参考文献】

[Rittel70] Rittel, H. and W. Kunz: "Issues as Elements of Information System", Working Paper 131, Institut fur Grundlagender Planug I.A., University of Stuttgart(1970).

[Conklin88] Conklin, J. and M.L.Begeman: "gIBIS: A Hypertext Tool for Exploratory Policy Discussion", CSCW'88 Proceedings(1988).

[Potts88] Potts, C. and G. Bruns: "Recording the Reasons for Design Decisions", Proc. of the 10th International Conference on Software Engineering, IEEE(1988).

[Potts89] Potts, C.: "A Generic Model for Representing Design Methods", Proc. of the 11th International Conference on Software Engineering, IEEE(1989).

3.5 協同文書作成支援技術

3.5.1 協同文書作成システム

協同文書作成支援システムは、グループウェアのうちグループエディタとして分類される。また、グループウェアを分類する有効な方法として時間・空間マトリクスによる分類があるが、グループエディタの場合は、時間的な同期／非同期の差はあるものの、基本的には場所を異にする者同士の協同作業に適用されることが多い。代表的なものをこれに当てはめると以下のようになる。

【同期分散対話型】	【非同期分散対話型】
・ GROVE	・Quilt
・ Collaborative Editing System	・PREP
・ Shared Book	
・ DistEdit	

グループウェアはそれぞれのプロダクトにより特有の機能を持つが、それらは以下にあげたキーワードによりその特徴を表すことができる。

(1) 文書の形態と利用環境

- 文書形式…文章、絵図、音声などオブジェクト（文書）を構成する情報の表現形式。
- 利用環境…グループ間の文書編集を行うために用意される利用環境。

(2) ビューと共有の概念

- ビュー…文章でいえば章・節などの論理的なセグメントであり、複数のユーザで見ることのできる、文書の部分集合をさす。
- 共有状況…プライベート、共有、公開などの共有の概念によって限定される一組のユーザとビューとの関係。
- 役割…利用者の役割（例えば共著者／注釈者／読者）を規定して、文書への特権および義務を利用者に与えること。
- 注釈機能…文書の本文に対して修正理由や提案あるいはプライベートなメモなどを構造的に付加し、文書の作成過程を支援する。

(3) グループインタフェース

- グループウインドウ…ユーザ間での同期インタラクションのためのビューを表示するウインドウ。
- グループ状況の表示…共有状況において関係者全員の活動状況、例えば「今、誰が、どこをどうしているか」が、WSを通して各メンバがいながらにしてわかる表示機能。

・ テレポインタ…複数のユーザのWSに表示され、あるユーザが動かすと全てのディスプレイで同期して動くカーソル。

・ 同時実行制御…利用者Aがある文章に単語を挿入する一方で、他の利用者Bがその文章を削除するなど、同時操作による競合を解決するための方法。単純ロック、トランザクション機構、交替プロトコルなどの手法がある。

・ 通知機能…あるユーザの行った編集行為をトリガとして特定のユーザに宛て通知を行い、逐次コミュニケーションを促進する機能。

3.5.2 協同文書作成支援技術の事例

(1) GROVE

① 概要

GROVE (the GRoup Outline Viewing Editor) は、グループウェアのインプリメント上の問題を研究し利用経験を得るために開発された実験用プロトタイプである。

その内容は、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ている。図3.5-1にGROVEのグループウインドウの例をあげる。

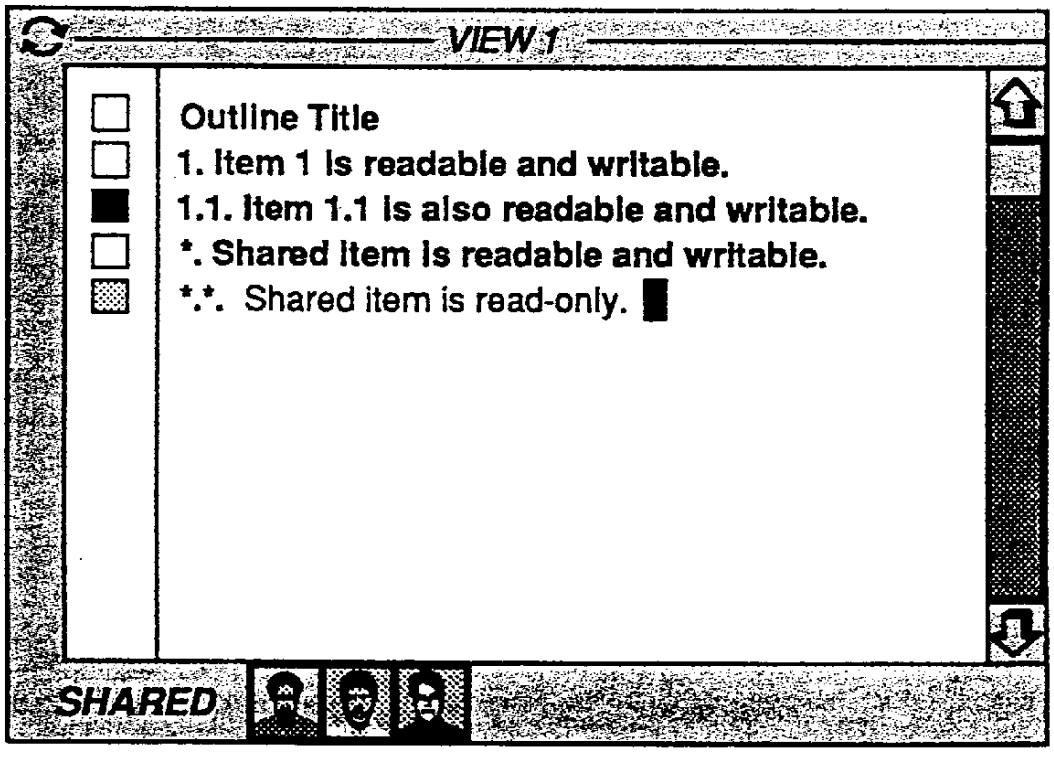


図3.5-1 GROVEのグループウインドウ

② 特徴

(a) 文書の形態と利用環境

文書はテキストタイプの文章であり、GROVEのセッションにおいて利用者は各々のWSとビットマップディスプレイを持ち、マルチウインドウによって一つ以上のテキストのビューを操作できる。

(b) ビューと共有の概念

ビューはアイテムと呼ぶ文章の単位で構成され、「プライベートビュー」「共有ビュー」「公開ビュー」という共有概念（アクセス権）により分類される。各ユーザには特に役割というものは規定されず社会的プロトコルによりインタラクションがおこなわれる。

(c) グループインタフェース

グループウインドウにはセッションに参加しているメンバの顔写真が表示され他の人とセッションがつながっていることがわかる。スクリーン上では修正あるいは参照可能な項目が指示され、一人のユーザが修正を行うと参加者全員がすぐにその修正を見ることができる。各ユーザのWSには複製された文書を置き、集中コントローラにより各ユーザの操作要求を受信しそれを順序づけて実行する分散同時実行制御がおこなわれる。

(2) Quilt

① 概要

Quiltとは「キルト（寄せ布細工）」を意味し、標準的なものあるいは斬新的な小さな文章を寄せ集め、構造化された注釈のタイプによってこれらを「縫い合わせて」文書を作成する。Quiltは、非同期なコミュニケーションにより協同執筆を支援する。

② 特徴

(a) 文書の形態と利用環境

Quiltは、ハイパーテキストを利用して文書に注釈を付ける構造化されたメカニズムを持ち、注釈にはテキストまたは音声を用いることができる。

システムは、以下に示す比較的入手の容易なソフトウェアを使用してインプリメントされている。ただし、データベースは集中制御のため、遠隔地の利用者同士は同一マシンへのアクセス権を持たねばならない。

- データベースシステム…「Orion」
- ユーザインタフェース…「Xウインドウ」「Xtツールキット」

(b) ビューと共有の概念

Quiltは「基本文書」に対し、「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」「履歴」などの注釈を作成でき、自分がコメントしたい箇所に直接記入できる（図3.5-2を参照）。

文書はスタイルとよばれる協同作業のタイプ、「排他」「共有」「編集者」の３種類が用意されているが、利用者が新たなスタイルを定義することができる。

また利用者には「共著者」「コメンテータ」「読者」といういずれかの社会的役割が与えられる。図3.5-3にスタイルと役割の定義の例を示す。

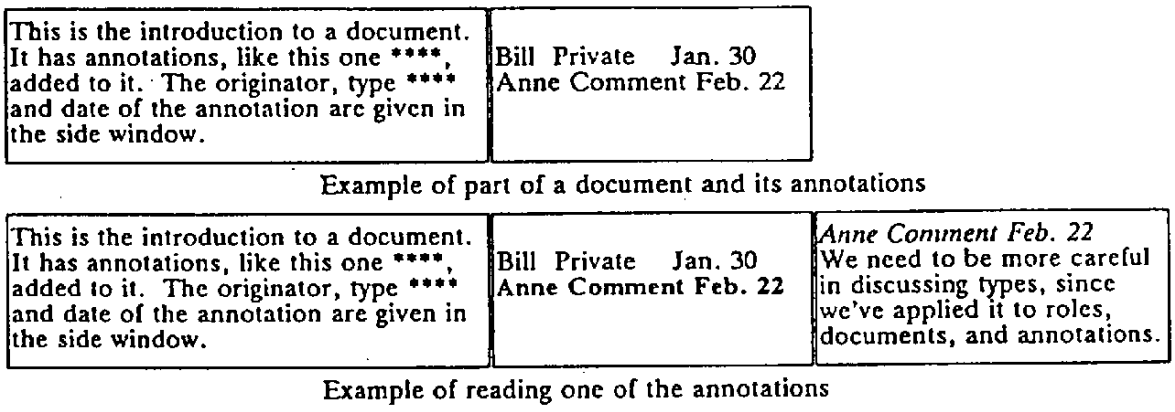


図3.5-2　基本文書と注釈の参照例

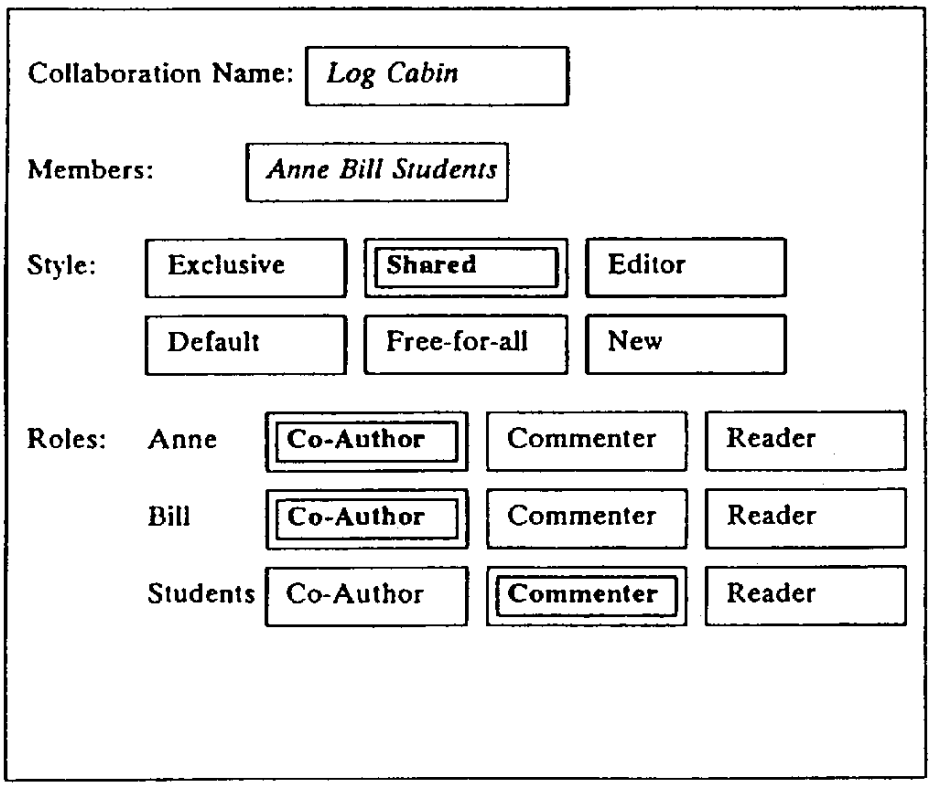


図3.5-3　協同作業のスタイルと役割の定義

さらに文書のスタイル毎に、各注釈タイプに対する社会的役割に応じた権利（アクセス権限）が定義できる。表3.5-1にその例を示す。横方向に注釈のタイプ、縦軸に役割の階層を指定している。

(c) グループインタフェース

Quiltは、非同期なコミュニケーションのもとで協同執筆を支援するため、以下の調整メカニズムを提供する。

- トリガ／通知メカニズム…共著者が交代で文書を編集する場合、例えばその行為の終了をトリガとして自動的に相手宛てにメッセージ通知を行うよう指示できる。
- アクティビティログ…本文に対する修正のたびにその理由やアイデアなどをアクティビティログに記録することにより、相手が修正結果を見るときに順序だってそこに至るまでの過程を知ることができる。

表3.5-1　協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

(3) PREP

① 概要

PREPとは協同執筆と注釈の関係を調査するために開発された「作業準備（Work in PREParation）」エディタをさす。PREPは、ネットワークを介して非同期に行われる学術的なコミュニケーションを支援するマルチユーザ環境を提供する。

② 特徴

(a) 文書の形態と利用環境

PREPは、ハイパーメディア・ツールにより、テキスト、表、ツリー図、任意のイメージを利用できる。また、Commentsとよぶハイパーテキスト・ツールによって文書への注釈機能を可能としている。

(b) ビューと共有の概念

文書は、協同作業者間で共有されるワークスペースにチャンクという情報単位で格納される。またチャンク間のリンクも可能であり、アイデアや概念のネットワークを構築することができる。

著者が他のパートナーにアクセスさせるための「草稿」という領域を定義できる。「草稿」は、チャンクを構成要素とするグリッドという表で構成され、グリッドの各カラムにはワークスペースの様々な内容を格納することができる。例えば二つのカラムに「論文の計画」と「論文の内容」を、もうひとつにはコメンテータが付け加えた「共著者の注釈」のカラムを与えることにより、共著者間で論文の草稿についてコミュニケートすることができる。

また、PREPは、利用者に対して社会的役割を明示定義しないが、協力者に基本文書の修正権を認可したり拒絶したりすることができる。

図3.2-4に、この草稿の例を示す。

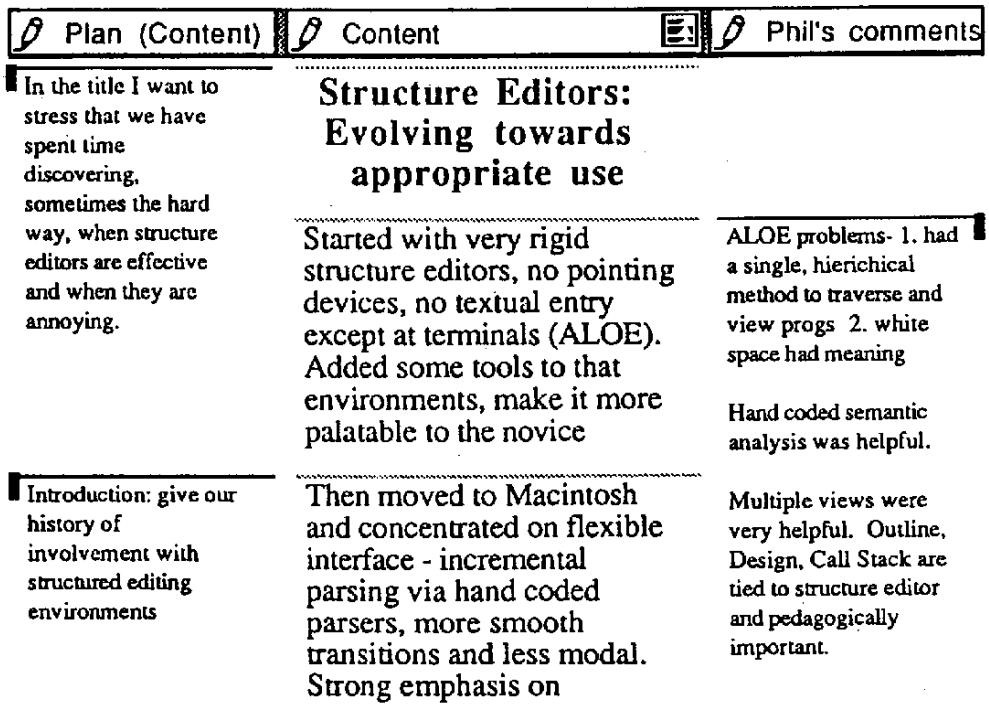


図3.2-4 三つのカラムをもつ草稿の例（PREPエディタ）

(c) グループインタフェース

(i) 計画に関するコミュニケーションの支援

執筆作業の分担を論ずる時、論文のシナリオ（計画）を作るがそれは確定的な内容ではない。それに沿って執筆を進め、言いたいことが見えてくるにつれてさらに新しい目標を設定するが、その変更は協同執筆に影響を与える。このように展開しつづける計画について共著者間のコミュニケーションを支援する。

(ii) 注釈に関するコミュニケーションの支援

PREPエディタは、利用者に注釈をつける機能を与える。注釈はテキストへの重要なノートである。注釈はコメンテータ間のコミュニケーションを促すが混乱も招く、これをどのように有効利用するかは共著者たちの問題となる。

(iii) バージョン化による訂正機能

PREPエディタは、草稿とは別個のバージョンとして既存テキストへ訂正することができる。これを利用することにより、著作者は既存資料を失う心配がなく、また訂正の背後の論拠やアイデアにもコミュニケートすることができる。

(4) DistEdit

① 概要

DistEditは、分散型環境での協同作業をサポートするグループエディタ構築のためのツールキットを提供する。つまり、DistEdit自身はエディタではなく、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。これにより同時に多種のエディタをサポートすることができ、共有文書を複数の人間が同時に編集することを実現する。

② 特徴

(a) 文書の形態と利用環境

セッションに参加できるエディタは、既存のテキスト・エディタでありそれらの種類は問わない。エディタにはバッファＩＯルーチンに若干の修正を加えるだけでグループエディタの一員として組み込むことができる。また、編集の対象は文章である。研究では、GNU EmacsとMicroEmacsをエディタとして、Ethernetを介して接続されたDEC3100ワークステーションが用いられた。

(b) ビューと共有の概念

複数のユーザが一つのテキストを共有する時、最初に起動したユーザを全ての編集機能をもつ「マスタ」と呼び、二番手以降のユーザは編集機能を持たない「オブザーバ」と呼ぶ。この場合、テキスト全体がビューであり、マスタは「編集者」として、オブザーバは「読者」としての役割を持つことになる。

図3.2-5に「マスタ」と「オブザーバ」のウインドウの例を示す。

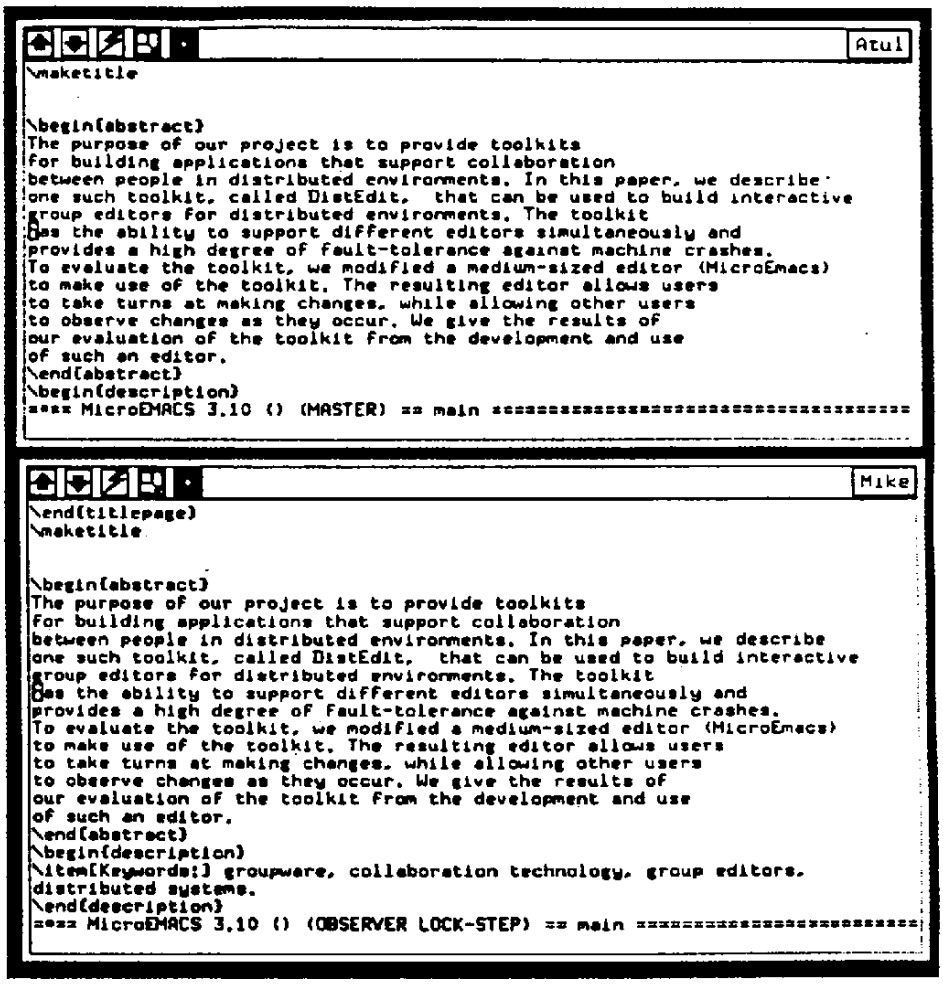


図3.2-5　DistEditにより実現されたグループ・セッション

(c)　グループインタフェース

DistEditは信頼性の高い同報通信プロトコルを使い、一つのセッションの複数のユーザは常に同じテキストの内容を見ることができる。

オブザーバはマスタの行う全ての変更とカーソルの動きを見る「ロックステップ（連動）」モードと、自分のカーソルをマスタから開放しどこでも好きなところに動かすことのできる「フリーカーソル」モードに自由に切り替えることができる。いずれにおいてもテキストの変更にかかわる編集コマンドだけは発行できない。

マスタはコマンドを用いていつでも制御を放棄することができ、オブザーバは同様にマスタになることができる。各々のウインドウの状態表示行には、自分の役割とモードが表示される。

3.5.3 協同文書作成支援技術の課題

以上に紹介した事例は、同じグループエディタの分類にありながらそれぞれに個性を強く持ち、またいろいろな示唆をわれわれに与えてくれる。ただどれもが完成された製品ではなく、同期／非同期における協同編集作業においてその参加者たちがどのような行動や感想を抱き、またその操作性やレスポンスがどうかなど、分散システム・通信・人間とコンピュータ・社会理論の視点からそのツールを用いて調査することを第一の目的として作られたプロトタイプである。これらは次のバージョンをめざしてさらに研究がすすめられているが、それぞれの特徴は必ずしも排他的でないことから、それらの利点を互いに取り入れたものに発展すればその成果は計り知れない。

協同文書作成支援技術の今後の課題としては、いろいろある中から調整の問題および文書の互換性を挙げる。

(1) 社会的局面としての調整の問題

協同作業の社会的局面に関する知識について、われわれは十分学ばなければならない。

「協同執筆を円滑に（対人間的トラブルなく）進めるために何を調整すればよいのか」という調整の問題に対して、Quiltにおいては「社会的役割」と「協同作業のタイプ」を規定することで解決しようと試みられており、GROVEにおいてはセッションに参加した人たちにお任せといったいわゆる「社会的プロトコル」により互いのインタラクションが調整されている。協同執筆の応用面においてはさまざまな組織、編集活動があるわけで、そのグループ（国民性も含め）の特殊な必要性に合うように、調整の機能をさらに研究・整理していく必要がある。

(2) 互換性

協同編集作業おいてはマルチユーザによる文書へのアクセスが基本的なテーマである。しかし、既存システムの多くは互換性のないシステムからの問題を解決していない。

ほとんどの利用者は文書とコンピューティング環境の互換性のなさに不都合を感じている。これが障壁となって、協同文書作成支援システムは研究所のシステムとしての地位にとどまっている。「互換性」の問題が解決されないかぎり、実世界で通用するグループエディタを作るための経験を積むことは非常に困難である。

【参考文献】

[Ellis91]Ellis,C.,et al.: Groupware:some issues and experiences, Communications of ACM, Vol.34, No.1, pp.38-58(1991.1).

[Leland88]Leland,M.D.P.,et al.: Collaborative document production using Quilt, CSCW'88 Proceedings, pp.206-215(1988.9).

[Neuwirth90]Neuwirth,C.M.,et al.: Issues in the design of computer support for co-authoring and commenting, CSCW'90 Proceedings, pp.183-195(1990.10).

[Knister90]Knister,M.J.,et al.: DistEdit:A distributed toolkit for supporting multiple group editors, CSCW'90 Proceedings, pp.343-355(1990.10).

3.6 グループ意思決定支援技術

3.6.1 GDSSとは

(1) GDSSの定義

グループ内で何らかの意思決定を行なう必要がある場合、大概はミーティングという形式で行なわれる。それ故に本節ではGDSSといった場合、ミーティングの支援を中心に話を進めていくことにする。

このミーティングというものに対しては、最近までコンピュータによるサポートがほとんどなされていなかった。しかしながら、最近のコンピュータ技術の発達（安価なPCやLANの設備）と、意思決定科学の分野の長年にわたる意思決定支援ソフトの開発が結びつくことにより、従来人手によらざるを得なかったミーティングをコンピュータで支援することが可能になった。この意思決定科学に基づくミーティングのためのソフトウェアをグループ意思決定支援システム（GDSS）と呼び、従来の意思決定支援からは区別している。以下によく使われるDeSanctisとGallupe('85)によるGDDSの定義を示す。

『ひとつのグループとして一緒に作業を行なっている意思決定者が、込みい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システム』

(2) GDSSの必要性

一般にグループによる意思決定（ミーティング）には個人で行なう場合に比べ、以下に示すような利点がある。

- 一人で行なうより、より多くの情報が得られる。
- 同じ情報でも違った観点から見直すことができ、同時に間違いも発見し易い。
- 他のメンバーからの刺激を受けて、より生産的になることができる。
- より優れたメンバーから学ぶことによって、生産性をあげることができる。

しかしながら、数多くの問題点も指摘されており、代表的なものをあげると、

- 参加者の数に従って発言時間が短くなってしまう。
- 発言タイミングを逸すると、意見を発表しないままで終ってしまう。
- 他の発言を注意深く聞く必要があるため、自分の意見を考えることができない。
- 他の発言を忘れてしまう。
- 他の発言を批判するのに消極的になってしまう。
- 自分の発言が批判されるのを恐れて、発言を控えてしまう。
- 他人の尻馬にのりやすい。
- 議論が発散し易く、論点が曖昧になってしまう。
- 少数の人間が支配してしまう。
- 余りに多くの意見が出されて処理しきれなくなってしまう。
- 限られた時間では外部情報にアクセスできずに不十分な情報で処理してしまう。

多くの場合、ミーティングは上記にあげた欠点にひきずられて、グループ意思決定の生産性を落している。GDDSではこれらの問題点をコンピュータを利用することにより回避し、また利点も十分に活かした上で、グループによる効率的な意思決定をサポートしようとするものである。

3.6.2 GDSSの実際とその問題点

(1) GDSS研究の現状

現在商業ベースを中心としたGDSSに関する研究は種々の支援ツールの効果を確かめるという方向に進んでおり、グループ内での相互作用や心理的社会的相互行為の研究（グループダイナミックス）は余り盛んではない。しかしながら以下に示すような例外も存在する。

- 未来研究所（Menlo Park,California）では、ミーティングをもっと効果的にする実際的な方法を探求しており、大組織における様々なコンピュータ技術の組合せによる効果を評価することに主眼を置いている。
- EDSのMachine Intelligence Centerでは、グループダイナミックスの徹底的な研究に力を注いでいる。

これに対し、アカデミックレベルでは人間の行動の理解を中心に据えて研究が行なわれている。これはテレビ会議に典型的に見られるように、行動を理解しないで作成されたグループウェアは利用されないという反省に基づいている。

(2) アリゾナ大学におけるGDDSの研究

アリゾナ大学はアメリカにおけるGDSS研究のメッカの一つであり、長年にわたる実験により、いろいろなツールを組み合わせた場合の効果の測定を行なってきている。中でもEMSはアリゾナ大学で開発されたGDDS用のソフトウェア環境であり、大規模グループミーティング用、小規模用、分散大規模用とミーティングルーム同士の遠隔会議用等に目的別に分類された施設を使って研究が行なわれている。これらの施設にはPLEXYS（アリゾナ大学が開発した同期対面式のミーティングを支援するソフトウェア）を基にした新たなソフト（UNIVERSITY OF ARIZONA GROUPSYSTEMS）が開発導入されており、その成果をもとに現在22以上の大学と12以上の企業（BELLSOUTH, GREYHOUND FINANCIAL CORP等）にEMS施設が導入されている。

基本的な構成は、参加者ひとりひとりに一台のワークステーション（ハードディスク、カラーディスプレイ、ネットワーク接続）と、リーダには２～３台のワークステーション、そして全員が見ることのできる電子黒板からなる。その他のAV機器（OHP、ホワイトボード）も利用可能である。支援ソフトウェアとしては、ミーティングにおける個々の作業ごとにツールを提供するという形をとっており、ユーザに合わせた柔軟なミーティング構造の構築ができるようになっている。提供されるツールとしては、

① セッションの計画、管理用ツール

Session Manager（SM）と呼ばれており、ミーティングの事前計画、運用、事後処理の作業を支援する。

② グループ相互作業支援ツール

実際のミーティング席上における参加者の相互やりとりを支援するツールであり、アイデア抽出支援（ブレーンストーミング、コメント付け）、アイデアの構造化、優先度付け（投票、代替手段検討、方針決定）等の作業を支援する。

③ 記録支援ツール

後々の利用のために、ミーティング中に出された意見等を構造化して記録する。

その他として、ミーティング席上での個人的な作業を支援するツールや、EMSの効果等を測定するための研究用データ収集ツールを持つ。

なお、アリゾナ大学ではミーティングを効率的にするためには、コンピュータによる支援だけでは不十分だと考えており、下記に示す要素も考慮にいれて研究を行なっている。

- グループの属性（グループサイズ、親密度、構造＝対等・階層、団結力、、）
- タスク（意思決定のために必要とされる作業、仕事の複雑さ）
- その他の要因（組織のもつ文化構造、時間的制約、、）

(3) ミネソタ大学

ミネソタ大学では新しい意思決定理論の開発とGDSSのアプリケーションの経験的な評価を主眼とした研究を行なっている。この理論面に対する強調はミネソタ大学における研究の特徴となっている。具体的な目的としては、

- グループがどのようにしてGDSSを使い、グループ間の相互作用とその結果にGDSSがどのような効果を及ぼすかの理論を構築しテストすること。
- GDSSのデザインがどのように組織のミーティングのプロセスと結果に影響を与えるかを研究しGDSSの現在の知識と社会科学に貢献すること。
- Adaptive Structuration Theory（AST）を通して、グループと組織の構造化の一般理論の開発に貢献すること。

これらの目的をベースとしてミネソタ大学で開発されたGDDS用のソフトウェア環境がSAMM（Software-Aided Meeting Management）である。SAMMはもともとC言語とUNIXの下で開発されてきたが、現在では、ハイパーカードを利用したMac版のSAMMも開発されている。GDSSの機能としてはごく一般的なものであり、問題の設定、アイデアの収集、アイデアの評価、判断支援、ミーティングのまとめの表示、レポート作成等がある。これは、ミネソタ大学のGDSSの目的が理論面の研究におかれているため、ツールの開発よりはデータ収集に重点が置かれているためだと思われる。

(4) IBM

IBMではグループ意思決定の効率化と品質向上を目的に、ミーティング支援用のソフトウェア環境であるTEAMFOCUSを提供している。TEAMFOCUSはもともとアリゾナ大学の研究の流れを汲むものであり、そのソフトウェアもアリゾナ大学からのライセンスを受けている。

基本的なハードウェア構成は、DOSベースのPC、プロジェクタ、ホワイトボード、プリンタからなり、機能としては、ブレーンストーミング、注釈付け、アイデアの構造化、論点の分析、投票機能、政策決定、議事録作成、代替手段の評価、ファイルリーダ、補助ファイル、グループ辞書等を持っている。

TEAMFOCUSの大きな特徴としては、高度な訓練を受けたファシリテータの存在が必須となっていることである。ファシリテータは各設備の開始、終了操作を行なったり、参加者に機器類の操作をガイドしたり、時には議事進行に介入して、議論の流れを制御したりする役目を負っている。

(5) GDDSを使った場合の利点、問題点

GDDSの利点としては、

- コンピュータ上で並列的作業を行なうことによって、通常の会議を越える量のアイデアの抽出・検討が行なえ、意思決定の質が向上する。
- 匿名による発言によって参加者の平等性が向上する。
- コンピュータを使うことで、余分な外乱が排除され議論に集中し易い。
- 各参加者の発言は全てコンピュータに記録され、質の高い記録を残すことができる。

等があげられる。

逆にGDDSの問題点として、

- アイデアが大量に出ることによって、逆に意見の一致が見い出し難い。
- 決定に至るまでに時間がかかる。

等がある。また上記以外にも、

- GDDSの効果をどのように測定するか。
- 現状ではGDDSのための専用設備が必要だが、どこでどう調達するか。また、施設を借りるとした場合には、いったいいくらの費用を払えばよいのか。
- ファシリテータをどこから見つけてくるか。どういった種類の訓練がファシリテータにとって必要か。

といった問題もある。

3.6.3 GDSSの課題

グループによる意思決定には、曖昧さ、常識、直感、政治的な思惑等、現在のソフトウェアが最も苦手とする分野の影響が大きいが、今後はこういったものをGDDSでどのようにサポートしていくかが大きな課題である。

さらに、一般への普及を目指すならば、ファシリテータのような高度に訓練を積んだ人間を必要としないツールの開発が必要である。また、ミーティングの結果得られたデータをどのようにして次のアクション（システム）につなげていくかといった面も今後は必要である。

最後に日本では個対個による根回しによって事前に意思決定を済ませておき、ミーティングの場では決定の確認だけを行なうといった場面もよく見られるが、今後はこういったものも含めた意思決定支援をどうしていくかといった問題もある。

【参考文献】

[Nunamaker91]Nunamaker,J.F.,et al: Electronic Meeting System to Support GroupWork, Communications of the ACM, Vol.34, No.7(1991.7).

[Kraemer86]Kraemer,K.et al.: Computer-Based Systems for Group Decision Support:Status of Use and Problems in Development, CSCW'86 Proceeding, pp.353-375(1986.12).

[松下91]松下 温: 図解グループウェア入門, オーム社(1991).

3.7 作業空間の共有技術

グループウェアにおける複数の利用者による作業空間（ワークスペース）の共有では、人間の五感に総合的かつ的確に訴えるマルチメディアの情報を共有しあう環境が土台となる。文献[阪田92]でも述べられているように、グループウェアの機能は、低位からcommunication（情報交換）、coordination（グループ調整）、creation（創造活動支援）に階層分けすることができるが、ここで述べる共有技術は、coordination の部分に対応する。ここでは、単に情報の同報通信されるだけでは情報共有とは言わない。同報された情報の同一性を保証するための制御を含んで初めて情報共有が実現される。

情報の共有には下記のように様々なレベルがあり、本節ではまず共有を支える要素技術を述べた後、情報共有のための制御技術について述べる。

3.7.1 テレポインティング

交換されたマルチメディア情報を共有しながら、プレゼンテータが他の利用者（会議などの同一協同作業の参加者）に指示棒に相当するテレポインタにより説明する機能は、作業空間の共有においては必須機能である。通常、ポイントした位置の縦横の座標値を連続的に同報する形で通信し、マルチメディア文書通信（第4.1節を参照）における図形（グラフィックス）の転送と同じ機構を利用する。冗長部分を転送しないため１０kbps程度の通信帯域があれば十分人間のペン入力の速度に追随でき、これにより高速な同報が可能となっている。

Rapport（AT&Tによる在席会議システム）などでは、ポインティングの情報を送ることができる利用者数が制限されているが、MERMAID（すでに実用化されている日本電気のマルチメディア分散在席会議システム）では全員（参加者数の制限なし）のポインティング情報が全員のディスプレイ上にリアルタイムに転送し表示できるようになっており、しかも、各利用者のポインティングのアイコンが色で識別できるようになっており、他の人達がどこを指示しているか、あるいはどこを見ているかがわかるようになっている（図3.7-1）。WYSIWIS（What You See Is What I See、下記）の実現による臨場感のある共有空間を創出している例である。

3.7.2 情報共有技術

(1) 情報共有のレベル分け

分散環境における情報の共有には、

- ビューの共有 ……利用者によるリアルタイムな情報の同時参照であり、WYSIWYG に対応する。最近はマルチウインドウのワークステーションやパソコンを使ったシステムが多いため、共有ウインドウ制御と呼ばれることもある。

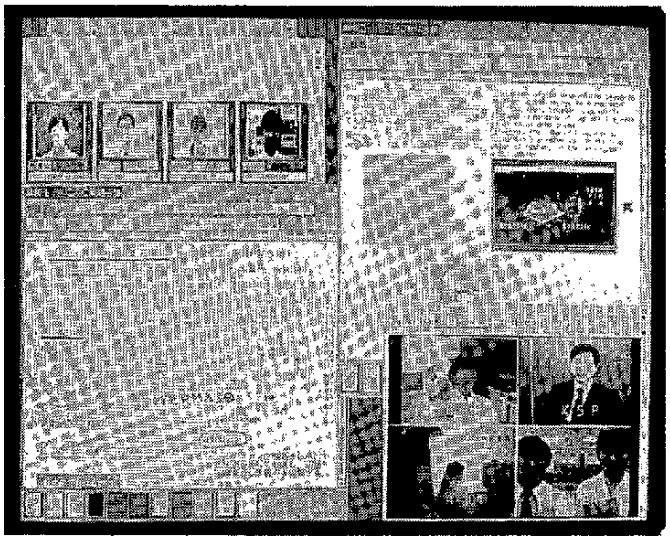


図3.7-1　MERMAID（日本電気）におけるテレポインティングの例
（全員が全員のポインタの位置を知ることができる）

- 処理結果の共有　……入力された情報の処理プロセスに関係なく、処理された同一の結果が各利用者により共有される。処理結果をファイルやデータベースなどある程度の時間間隔格納するという概念は含まない。グループウェアの基本機能として注目を集めているアプリケーションの共有、あるいはアプリケーションの分散協調制御の機能に対応する。
- 蓄積データの共有　……ファイルやデータベースなどある程度の時間間隔を置いて格納されている共有情報へのアクセス制御に対応する。

の３種類がある（図3.7-2、図3.7-3）[阪田89, 水野92]。

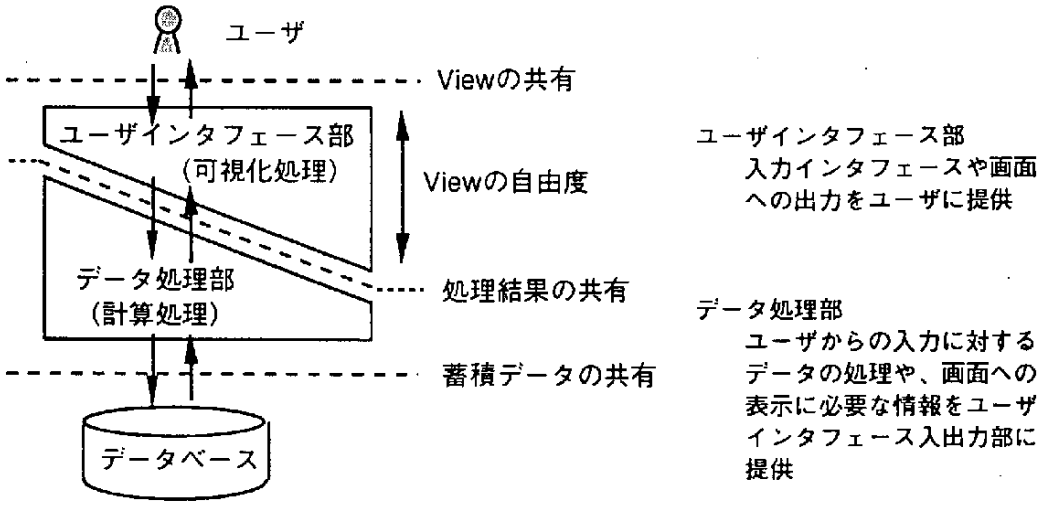


図3.7-2　グループAPの構造

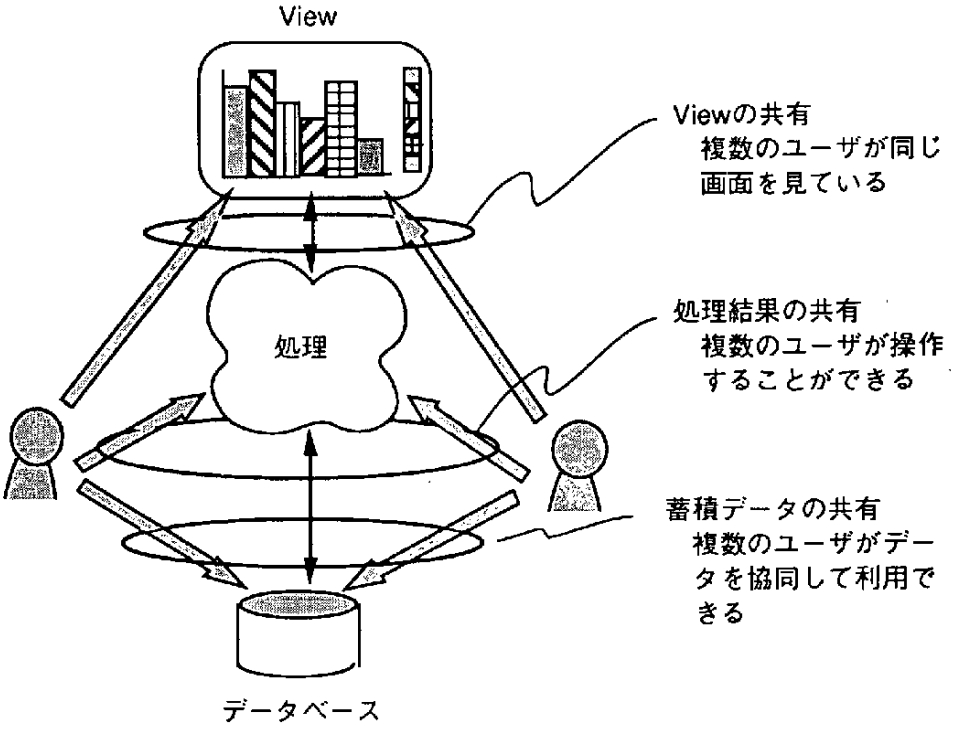


図3.7-3　グループAPにおける三つの共有

この中で蓄積データの共有については、分散データベースの分野において古くから静的なデータへのアクセス・排他制御、一貫性管理（同時更新制御）、版（バージョン）管理などの技術として研究されている。最近では、分散コンピューティング（第4.1節を参照）環境の中で、リモートプロシージャコール（RPC）などの通信手段を用い、各クライアント（アプリケーションプログラムや利用者）が同一情報を共有する分散トランザクション処理として再び研究が盛んになりつつあるが、作業空間の共有というグループウェアの環境とは直接密接な関係がないため、以下ではビューの共有と処理結果の共有のみを扱う。

(2)　同時参照のためのグループ調整・排他制御　……ビューの共有

会議システムなどのリアルタイム型のグループウェアシステムにおける発言権や共有情報へのアクセス（書込みや編集）権が代表例である。発言権については、ビューの共有というよりはグループ調整機能に相当し、システムでこの機能を支援している例はほとんどなく、利用者間での調整に任されている。WYSIWIS を実現するための、ディスプレイ画面やウインドウ上に表示された共有情報へのアクセス権制御は、参加者間で同一目的をもった協同作業の支援（CSCW）においては必須機能である。マルチウインドウに関しては、ウインドウとウインドウの間の重なりにおけるオーバレイ（上の方の

情報を優先表示）か透明（下の方の情報を優先表示）かの指示ができることが重要である。

共有情報へのアクセス権制御機能を提供していない、コンピュータ（ワークステーションやパソコンなど）ベースの会議システムなどが開発されつつあるが、このようなシステムはリアルタイム型のグループウェア（CSCW）とはいえない。単なる通信システムである[阪田91]。

また、共有情報へのアクセス権制御については、アクセス権の単位と排他制御の方法に関する自由度が重要となる。アクセス権の単位について、協同作業（会議など）におけるある時点での共有対象情報全体、あるウインドウ内の表示情報全体、特別に定義した情報の単位などが考えられる。しかし、あまり排他制御の単位を細かくすると操作が複雑になり、ヒューマンインタフェース上問題となる。排他制御の方法については、例えば上記日本電気のMERMAIDにおいては、

- 協同作業の間会議における議長に相当する集中管理者を設定し、その管理者がアクセス権の許可・移動制御を行なう
- バトン式に現在アクセス権を保持している人が次の人を指名しアクセス権を渡す
- 要求順（タイムスタンプなどによって制御する）
- 全く制御せず全員に自由にアクセスさせる（参照情報の同一性は保証しない）

等のモードを設け、利用者が自由に選択できるようにしシステムの自由度を高めている。フォーマルな会議のように参加者に上下関係が明確な場合は議長を設定したモード、技術討議のように対等な関係の参加者の場合は制御をしない形で使うことが多い、という利用評価結果も報告されている[阪田92]。

また、上記のようにWYSIWIS はCSCWにおいては必須機能ではあるが、局面によっては受信者側で自由にレイアウトしたいという要求もある。例えば、参加者一人一人の関心事が異なり、関心のある情報のみを拡大したりしたい場合である。例えばMERMAIDでは、デフォルトとしては、アクセス権保持者（プレゼンテータ）と同じ画面を他の参加者が参照できることになっているが、他の参加者が自分の操作で自由にそのレイアウトを変えたりすることができるようになっており、WYSIWIS とさらに高い自由度を兼ね備えた高度なヒューマンインタフェースを提供している。

(3) アプリケーションの分散協調（アプリケーション共有）制御 ……処理結果の共有

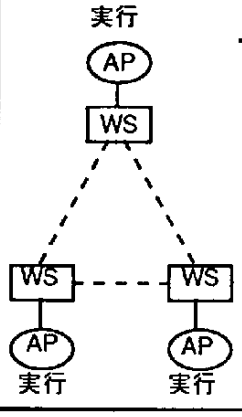
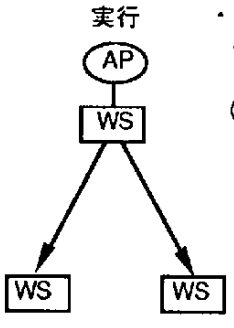
この制御は広い意味では(2) のビューの共有をも含むが、一般的にはビューの共有のみでは会議など使い方が限定される傾向にあるのに対し、処理結果の共有では上位の任意のアプリケーションが協調的に動作できるように支援することを目的とする。

アプリケーションを共有するとは、協同作業の参加者全員が実行中のアプリケーションからの出力結果をリアルタイムに得ることができ、またそのアプリケーションをリアルタイムに操作し入力を与えることができることをいう。一つの文書を協同で同時に編集したり、一つの帳票（スプレッドシート）を何人かで協同で作成したりするときに必要な機能である。アプリケーション共有に関しては、共有方式と対象アプリケーションの二つの側面から実現形態を考察する必要がある。

① 共有方式

アプリケーションの共有方式には、分散方式と集中方式がある。分散方式は、アプリケーションを各サイトで起動し、入力をすべてのサイトに分配し、実行は各サイトでローカルに行なう方式である。集中方式はアプリケーションを一つのサイトで起動し実行（出力）結果を各サイトに同報する方式である。表3.7-1に両方式の定性的な比較を示す。

表3.7-1　グループAPの分散協調制御方式の比較

	方式概要	長　所	短　所
分散方式（重複）	 ・処理指示が同報され各WS上のAPで実行 - - - - 処理指示	・処理指示のみがWS間で送受信されるので、ネットワーク・トラフィックが小さい ・実行結果を自WSのAPから受け取るので応答時間が短い	・すべてのWSが、APの分散協調が実行できる環境（ハードウェア、ソフトウェアの初期状態が同一）でなければならない ・同一の処理指示が、すべてのWSで同一の順序で処理される必要があるので同期制御が難しい
集中方式	 ・ひとつのWSのAPで実行し、その結果（データ本体）を同報 → 実行結果	・各WSに実行結果が同報されるので、APの同期制御が容易	・各WSに実行結果が同報されるのでネットワーク・トラフィックが大きい。特にイメージなどの情報量の大きなデータを利用するAPの場合、大きな遅延が生じる ・高速伝送路を必要とする

分散方式を実現している例としては、日本電気のMERMAID（図3.7-4）[大森92]、BBNのMM-Confなどがある。集中方式の例としては、最近、UNIXのX Windowを対象としたウインドウレベルでの情報の共有（ソフトウェア的には情報のコピー）を目的とした、XTV、shX、pscrawlなどが開発されており、基本ソフトウェアに近いレベルでの実現が試みられている。

② 対象アプリケーション

既存のシングルアプリケーション（スタンドアロン用に開発されたアプリケーション）を分散協調できるように拡張するアプローチと、最初からグループアプリケーション（分散協調が可能なアプリケーション）として作成するアプローチがある。

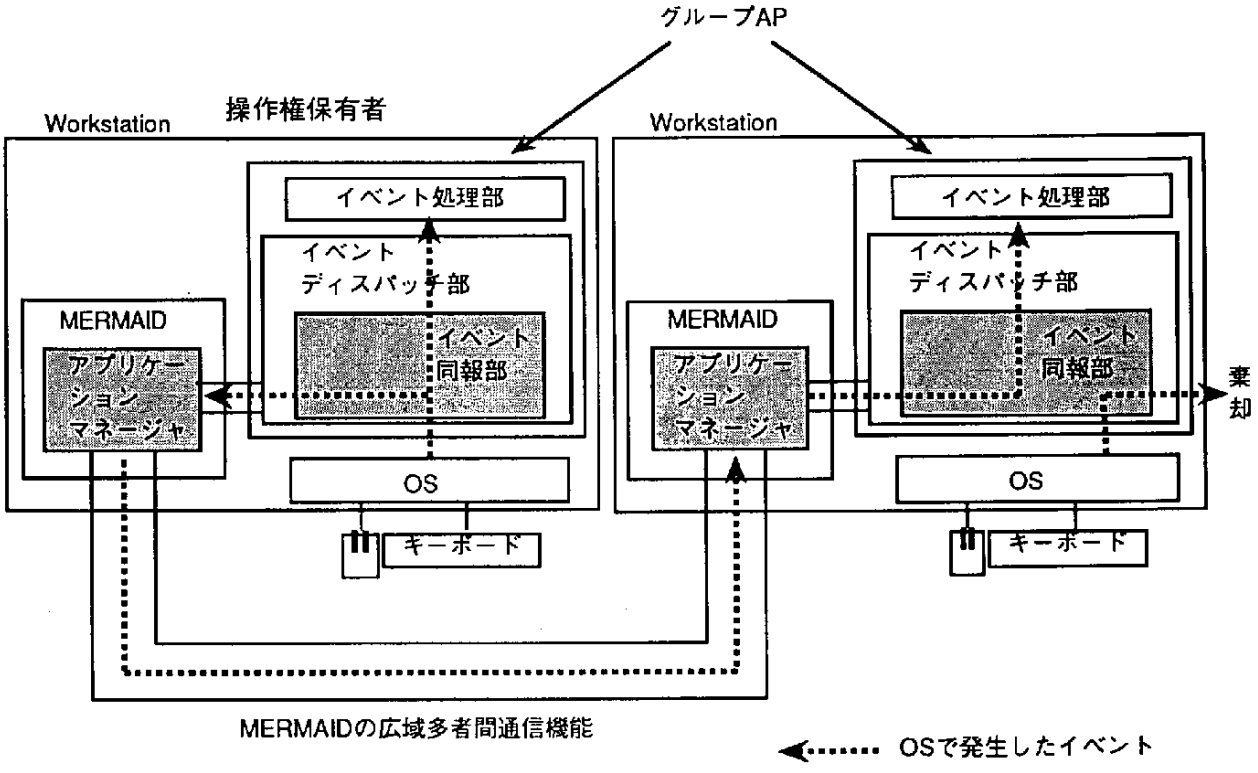


図3.7-4　分散制御によるAP共有方式（日本電気のMERMAIDの例）

シングルアプリケーションをグループアプリケーション化するためには、データの処理を行なう基本的部分には手を加えず、利用者との入出力を行なう部分の変更のみに留める必要がある。例えば、X Window上のシングルアプリケーションをグループアプリケーション化する場合、利用者からの入力（イベント）と利用者への出力（リクエスト）を処理する部分の変更やインタフェースの追加（イベントやリクエストの同報や多重化など）を行なう必要がある（図3.7-4）。

新たに最初からグループアプリケーションとして作成する場合には、様々な実現方法が考えられるが、利用者との入出力部分については、上記のシングルアプリケーションをグループアプリケーション化するときと同様の方法を適用する必要がある。

【参考文献】

[阪田92] 阪田:マルチメディアとネットワークによるグループウェアの実現技術, ソフト・リサーチ・センター (1992.5).

[阪田89] 阪田 他:グループ通信アーキテクチャ, 電子情報通信学会技報, OS89-26(1989.9).

[水野92] 水野 他:グループアプリケーションにおける機能分類の提案, 情報処理学会全大,4W-2(1992.10).

[阪田91] 阪田: ＣＳＣＷにおけるマルチメディア技術, 計測自動制御学会誌, pp.497-504(1991.6).

[大森92] 大森 他: マルチメディア分散在席会議システム（ＭＥＲＭＡＩＤ）を利用したグループアプリケーションの分散協調制御方式とその実現例, 情報処理学会研究会報告, MDP53-8(1992.1).

4．グループウェア構築のための
コンピュータ基盤技術

4．グループウェア構築のためのコンピュータ基盤技術

4.1　マルチメディアネットワーク技術

グループウェア（CSCW）の目的は、人と人とのコミュニケーションによる協調活動を支援することにある。コミュニケーションを円滑に行うためには、様々な情報メディアを効果的に組合せ（マルチメディア）、人間の五感に総合的かつ的確に訴えることにより、意思を伝達し合うことが重要となる[阪田92a] [阪田91]。

また、グループウェアは、複数の利用者（マルチユーザ）による協同作業を対象としており、このような環境では、特定のユーザ群に同一情報を通信するマルチキャスト（同報）機能が重要となる。

本節では、マルチユーザを対象としたグループウェアシステム実現の中核技術となる、マルチメディアネットワーク技術に関し、ネットワークの高速化、マルチキャスト機能を加える形で現状、将来動向、課題、今後の展望について述べる。

4.1.1　マルチメディアネットワークのシステム体系

マルチメディアネットワークに関連する技術の位置づけを明確にするため、システム全体の機能体系化を試みる。マルチメディアネットワークのシステム全体を、人間（ユーザ）を上位、インフラストラクチャ（以下インフラ）を下位とする階層構造により体系化すると、図4.1-1のようになる。アプリケーションを含めた通信機能の体系であるOSI参照モデルとの対応も図に含める。

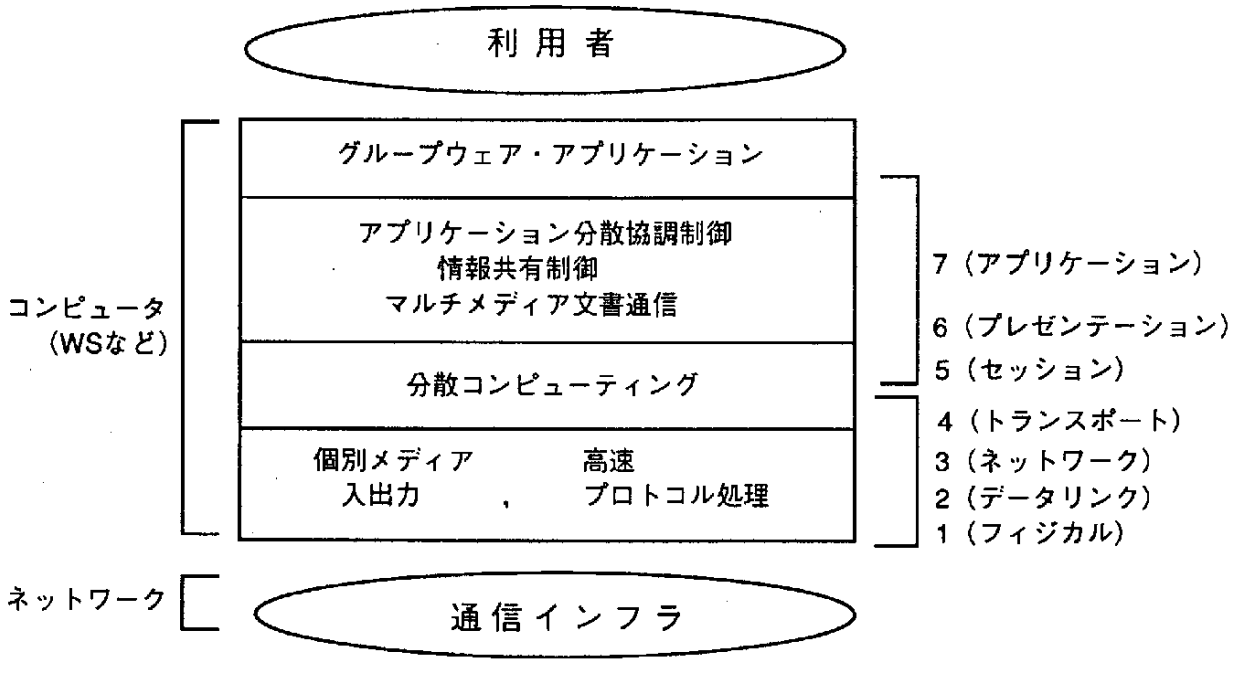


図4.1-1　グループウェアから見たマルチウェアネットワークのシステム体系

マルチメディアを考慮したプロトコルとしては、OSI参照モデルにおけるレイヤ１（フィジカルレイヤ）、レイヤ３（ネットワークレイヤ）とレイヤ４（トランスポートレイヤ）、およびレイヤ７（アプリケーションレイヤ）の三つのレベルに分けることができる（図4.1-2）。現状ではレイヤ１に関する部分は通信制御に閉じたハードウェア、レイヤ３とレイヤ４に関する部分はハードウェアとソフトウェア、レイヤ７に関する部分はコンピュータのソフトウェアにより概ね実現される。レイヤ３とレイヤ４の部分は通信とコンピュータの技術が融合する部分（C&C）で、マルチメディアに関しては最も後で手がつけられている部分であり、研究としてもまだまだ多くの課題が残されている。

以下では、図4.1-1の体系に沿って下位の部分から説明する。

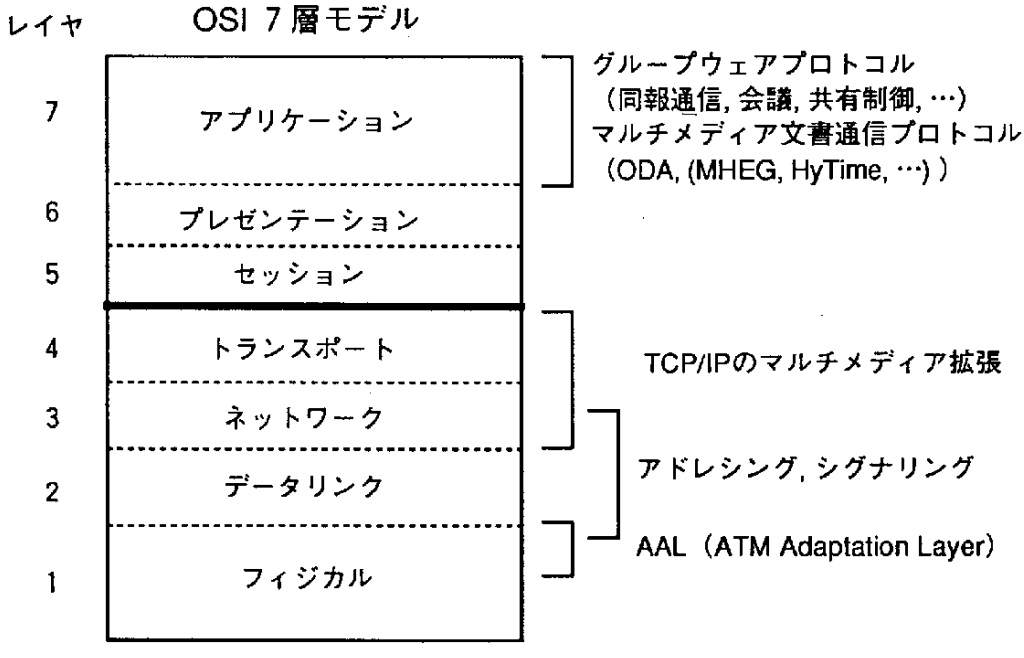


図4.1-2　マルチメディアプロトコル体系

4.1.2　通信インフラ

(1)　広域網

現在マルチメディアの形態で通信が可能なサービスとしてN（Narrow band）-ISDN（Integrated Services Digital Network）がある。しかし、N-ISDNでは最高通信速度は約1.5Mbps であり、グループウェアなどの高度なアプリケーションから見るとデータやイメージの転送速度、動画の品質の点で性能的に不十分である。通常のテレビ放送（NTSC、フル画像では216Mbps に相当）の画質を得るには10Mbps程度の帯域は必要とされており、光ファイバを利用したB-ISDNへの期待が高まっている。B-ISDNで採用されるATM（Asynchronous Transfer Mode）方式では、以下のようにメディアの統合は網内

で実現され、155Mbpsを基本とする高速通信が可能となる。HDTV（High Definition TV）に代表される高精細で臨場感のあるリアルな映像の通信も可能となる。

ATMでは、ディジタル化した文字や音声、映像などの各種情報をセルと呼ぶパケットに類似した固定長のブロック（ヘッダ部５バイト、データ部４８バイト）に挿入し、このブロックを単位として順次情報を多重化し非同期転送する（図4.1-3）。セル単位で回線型のデータ（音声や動画など）とパケット型のデータ（文字や図形など）とを区別することなく多重化し転送するため異なる伝送速度に柔軟に対応（多元トラフィック処理）でき、伝送路を効率的に利用できる。固定長のセルにすることによりプロトコル処理もハードウェア化が容易で高速通信が可能となる[電子情報通信学会91]。

B-ISDNにおけるマルチメディアを考慮した制御は、表4.1-1に示すAAL（ATM Adaptation Layer）におけるサービスクラス（A, B, C, D, X）またはAALタイプ（１，２，３，４，５）の識別により行う。

B-ISDNは、90年代半ばに開始予定の企業向けの専用線サービスを経て、2020年頃までにはFTTH（Fiber To The Home）と呼ばれる一般家庭への導入が計画されている。

表4.1-1　ATMで提供するサービスとATMアダプテーション・レイヤとの関係
（CCITT勧告1.362,1.363）

<table>
<tr><th>クラス
パラメータ</th><th>A</th><th>B</th><th>C</th><th>D</th><th>X
(SEAL)</th></tr>
<tr><td>送受信
タイミング</td><td colspan="2">必要</td><td colspan="2">不要</td><td rowspan="2">ユーザ定義</td></tr>
<tr><td>ビット・
レート</td><td>固定</td><td colspan="3">可　変</td></tr>
<tr><td>モード・
コネクション</td><td colspan="3">コネクション型</td><td>コネクションレス型</td><td>コネクション型</td></tr>
<tr><td>個人サービス</td><td>・サーキット・シミュレーション
・固定速度(CBR)符号化映像</td><td>・可変速度(VBR)符号化映像</td><td>・コネクション型データ</td><td>・コネクションレス型データ
・LAN間接続</td><td>・コネクション型データ</td></tr>
<tr><td>AALサービス</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr>
</table>

(2)　構内網（LAN）

N-ISDNのインタフェースを備えたマルチメディア対応のIVD（Integrated Voice and Data）-LANが既に製品化されているが、帯域が不十分ということもあり普及には至っていない。100Mbps以上の高速な

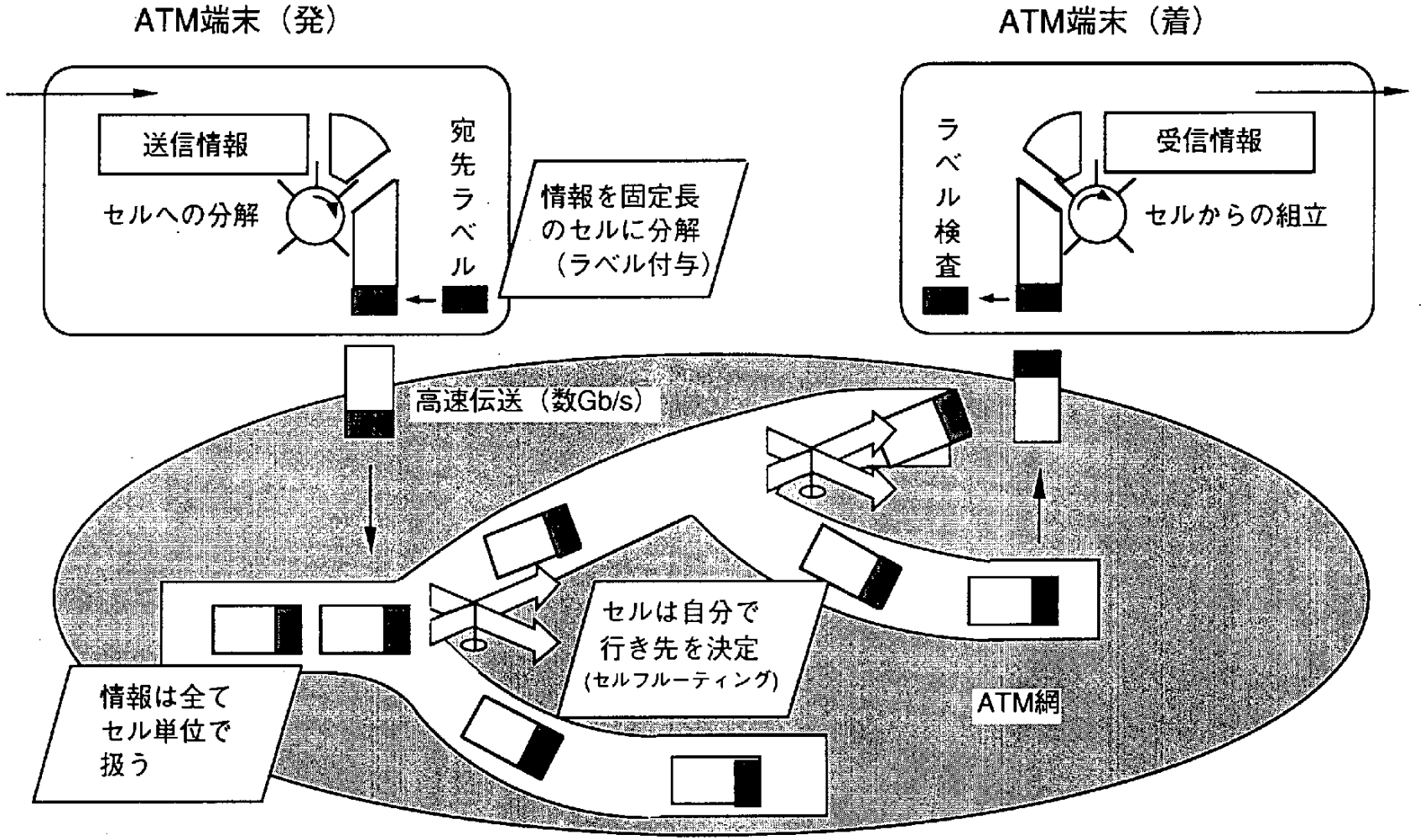
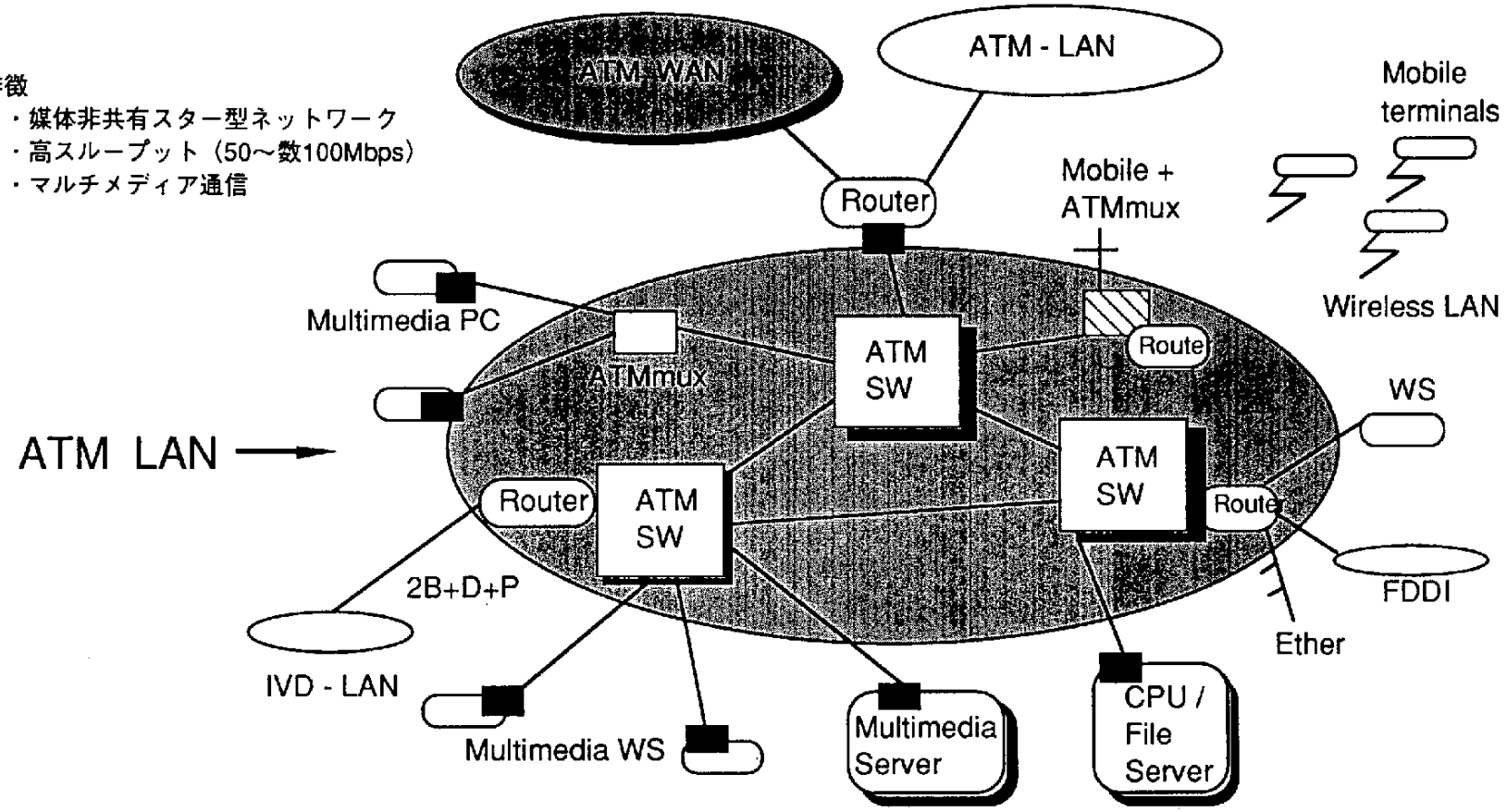

ATM端末（発）
送信情報
セルへの分解
宛先ラベル
情報を固定長のセルに分解（ラベル付与）
ATM端末（着）
ラベル検査
受信情報
セルからの組立
高速伝送（数Gb/s）
セルは自分で行き先を決定
(セルフルーティング)
情報は全てセル単位で扱う
ATM網

図4.1-3　ATMの原理

特徴
- 媒体非共有スター型ネットワーク
- 高スループット（50～数100Mbps）
- マルチメディア通信



Router：マルチメディア・マルチプロトコル対応ノードプロセッサ
■：ATMインタフェースカード

図4.1-4　ATM - LANの構成

マルチメディアLANについては、FFOL（FDDI Follow On LAN）とATM-LANの２種類のLANに関し、現在標準化の検討がなされている。特にATM-LANは、広域網としてのB-ISDNとの親和性を利点とし、コンピュータやルータのメーカも参画して、92年より急速にインタフェースの標準化が進展している。図4.1-4にATM-LANを含めたネットワーク構成のイメージを示す。

(3) 放送系との関連

同報通信に着目すると、無線による放送系を無視することはできない。これまで単方向、不特定多数（マルチキャストではなくブロードキャスト）への音声、画像の分配を主としてきた放送系と、双方向、単一／特定複数間の情報交換を可能とする通信系のインフラが、情報のマルチメディア化、コンピュータとの連携によるサービスの高度化に伴って、その境界が不鮮明になり、企業／家庭での利用など様々な形態での融合が図られていくと思われる[広帯域ISDN92]。

B-ISDNにおけるマルチキャスト機能については、発信元におけるATM交換機内のバスにおける分岐、および中継のATM交換機での受信情報のコピーとその多方路への送出により、従来のコンピュータネットワークに比べ効率的に実現できることが考えられるが、B-ISDNサービスが未だ実施されていない現在では、具体的な実現方式に関する研究段階である。

いずれにしても、マルチメディア情報を駆使した円滑でかつ効率的なグループ協同作業の支援には、B-ISDNのようなネットワークの利用が必須である。広域, 構内ともほぼ時を同じくして、90年代半ば（95年頃）にマルチメディア通信としてのインフラが整備され始めると思われる（図4.1-5）。

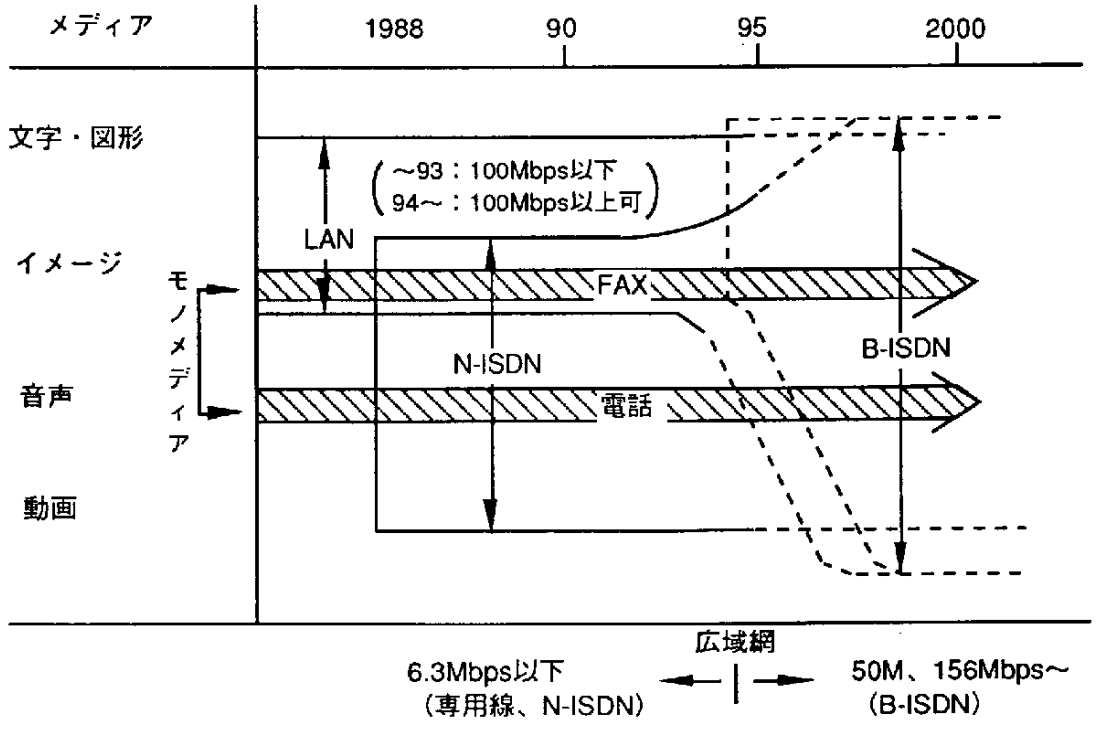


図4.1-5 情報メディアから見た通信環境の変遷

4.1.3 高速プロトコル処理

高速プロトコル処理には、プロトコルそのものを高速通信が可能なように拡張・変更していくことと、規定されたプロトコルをハードウェアないしソフトウェアにより高速に動作するよう実装するという二つの側面がある[阪田92b]。

レイヤ1では、プロトコルとして規定されたAALのサービスクラス（AALタイプ）に関し、その処理が概ねハードウェアにより実現されることから、100Mbps以上の高速性が得られる。

レイヤ3とレイヤ4については、OSIのプロトコルよりも、既にUNIXワークステーションを主体とするコンピュータネットワーク上で普及しているTCP／IPプロトコルに関して様々な角度から研究がなされている。しかし、TCP／IPでは、従来コンピュータデータしか扱ってこなかったため、動画や音声を扱えられるようにするという意味での高速性についてはそれ程考慮されていない。マルチメディア通信への要請に伴い、このレベルでは、動画と音声という大容量で時間要素をもつメディアの取り込みそのものがプロトコルの高速化の目的となっている。

- 動画と音声の取り込み・メディア間同期のための割込み優先（プリエンプション）制御、要求される帯域に応じた適応型フロー制御
- 多少のビットエラーが発生しても利用者にほとんど影響を与えない動画と音声のためのパリティチェックの省略

などの新しい機能の導入を図っている。このようなプロトコルの例として、Protocol Engine社によるXTP（eXpress Transport Protocol）、BellcoreによるTP++、MITによるNETBLT、INRIAによるTP5などが挙げられる[Dupuy92]。

マルチキャスト機能については、

- マルチコネクション制御（複数のコネクションをグループのコネクションとして同時に設定し制御する）
- グループアドレスの登録とグループアドレスを用いた自動グループ同報制御

などの機能の追加により、現在のTCP／IPの拡張が図られている。このようなプロトコルの例として、上記のXTP、Stanford大によるVMTP（Virtual Mutiple Transport Protocol）、Multi-G（スウェーデン）などで利用されているST（STream protocol）- IIなどがある[Dupuy92]。

また、プロトコルに付随するインタフェースとして、互いに密接に関連し、通信インフラ（レイヤ1）とも関係をもつ以下の項目がある。

- アドレシング……情報を交換する主体であるユーザ、端末などの単位の設定。従来の電話交換ではE.164と呼ぶアドレス体系が用いられてきたが、コンピュータ間通信ではMAC（Media Access Control）アドレスとネットワークアドレスと呼ぶそれぞれOSI参照モデルにおけるレイヤ2（データリンクレイヤ）とレイヤ3のアドレスを使っており、ISDNの利用環境ではこれらの間の整合、

変換方式が重要課題となっている。

- ルーティング……マルチキャストを含む、宛先アドレスへの情報の転送経路制御
- シグナリング……ネットワークは、端末（利用者）からの要求サービス品質（QOS（Quality Of Service）：許容伝送遅延やパケット損失率、ビットエラー率など）に応じて、回線帯域や通信バッファなどのネットワーク資源を効率的に割り当てるための調整制御を行う必要がある。シグナリングは、端末が通信を行うときに、交換機が相手端末番号（アドレス）に従って相手端末まで回線を接続していく時の手続きである。従来電話交換ではNo.7、N-ISDNではQ.931というインタフェースが用いられ、B-ISDNではQ.931の広帯域版であるQ.93Bが規定されているが、コネクションの管理との関係からTCP／IPベースのインターネットで既に普及しているSNMP（Simple Network Management Protocol）との整合が検討されている。

アドレシングとシグナリングについては、今後B-ISDN環境でのマルチメディアとマルチキャストを取り込んだ、具体的な方式の検討が待たれる。

4.1.4 メディア入出力（圧縮）制御

N-ISDN環境といえども、現行のテレビ（NTSC）の216Mbps相当のフル画像に対しては100分の1以下に圧縮する必要がある。冗長な部分が多く含まれる動画像の情報に対し動きの部分（差分）のみを取り出して冗長性を取り除くような圧縮（符号化）技術が重要となる。動画の圧縮方式については、通信（テレビ電話、テレビ会議）、蓄積（データベース、パッケージメディア）、放送（ディジタル放送・分配）の3種類があるが、それぞれCCITT、ISO、CCIRでその標準化の検討がなされている（表4.1-2）。グループウェアで重要となるのは前の2者である。

通信に関するものはリアルタイムビデオとも呼ばれ、いわゆるたれ流し的な動画で画像そのものの編集や処理は考慮していない。９０年にCCITTによりP×64kbps（Pは1～30、N-ISDNに対応する部分）チャネルの各メディアへの割付け方法に関する国際標準をH.261シリーズとして勧告した。９１年頃より、テレビ会議での利用を目的とした標準準拠の動画CODECの製品が出現し、その後在席会議システム（例：日本電気のMERMAIDなど）での利用を想定し、パソコンやワークステーションへの実装が可能な小型、低価格の動画CODECの開発が、急速に進められている[阪田92a, 安田92]。

会議を始めとするグループウェアのシステムでは、リアルタイム型だけでなく、保存・検索、編集・処理が可能な（ストレージビデオとも呼ばれる）の圧縮も重要である。蓄積型の動画に関しては、MPEG（Moving Picture Experts Group）の名で標準化が進められている。当初は表4.1-2に示すようにMPEG1, 2, 3のフェーズに分けて検討されてきたが、その後MPEG3はMPEG2に吸収されてMPEG2としての帯域の制限はなくなり、さらに移動通信や加入電話網などの低速度環境での動画伝送を目的とするMPEG4の検討が開始されている。

表4.1-2　音声を含めた符号化方式の標準化

	標準化機関	勧告/方式	ビットレート	アプリケーション
音声符号化	CCITT	G.726	16～40kbit/s	電話
		G.722	64kbit/s	TV会議, 音声会議
	ISO	MPEG1	32～192kbit/s	ディジタルストレージメディア
画像符号化	CCITT	H.261	64kbit/s～2Mbit/s	TV電話, TV会議
		H.26X	～数十+Mbit/s	ATM網での各種画像サービス
	ISO	MPEG1	1～1.5Mbit/s	ディジタルストレージメディア
		MPEG2	3～10Mbit/s → 制限なし	各種画像サービス
		(MPEG3)	～40Mbit/s	各種画像サービス(HDTV対応) MPEG2 に吸収
		MPEG4	32kbit/s 以下	移動通信, 加入電話網用低速度対応
	CCIR	Rec.723	32～45Mbit/s	ディジタルTV 1 次分配
		Rec.721	140Mbit/s	素材伝送用
		—	10～20Mbit/s ?	ディジタルTV 2 次分配
		—	—	HDTV 1 次/ 2 次分配

また、静止画に関してはJPEG（Joint Photographic Experts Group）の名で既にISOで標準化がなされているが、最近特に、米国においては、MPEGチップ／ボードの開発を待てずに、JPEGで符号化された静止画を連続的に扱うことにより動画として表現しようとする（Motion JPEGとも呼ばれる）技術開発が盛んとなっている。パソコンやワークステーションに内蔵されるMotion JPEG用チップ／ボードが米国では既に多種類が製品化され販売されている。

4.1.5　分散コンピューティング

分散コンピューティングは分散処理の基本要素として位置づけられ、表4.1-3に示すような進展を経て、マルチメディア通信やグループウェアのためのプラットフォームとして機能することが期待されている。大まかには、分散OSの時代はシステムプログラマを支援の対象としていたのに対し、現在の分散コンピューティングではアプリケーションプログラマへの支援へと対象のレベルを上げ、次の世代では分散透過性という概念をより徹底させて、シングルシステムイメージという究極の形態への発展を目指している。

分散コンピューティングの具体的な機能やインタフェースについては、UNIXに関するOSF（Open Software Foundation）とUI（Unix International）の二つのコンソーシアムで検討がなされ、それぞれ

DCE（Distributed Computing Environment）、ATLASと呼ばれ、９１年頃から急速にその基盤が構築されつつある。現在の分散コンピューティングの機能体系は図4.1-6に示す通りで、DCEもATLASも概ねこの体系と同じである。しかし、グループウェアから見た現在の分散コンピューティングには、以下のような課題がある。

- リアルタイム性の保証 ……クライアント・サーバモデルによる機能モジュール間の論理的なインタフェースについては、動画や音声の取り込みとも関連する応答性の保証、絶対時刻の扱いが未だ対象外である。
- マルチメディア情報の扱い ……現在の分散コンピューティングは、コンピュータデータしか扱いの対象としておらず、優先制御やメディア間の同期に関連するマルチメディアへの考慮がなされていない。
- 高速通信インタフェース ……現在の分散コンピューティングでは、専用線やEthernetまでの通信インタフェースしか想定しておらず、B-ISDNのような高速通信環境には対応できない。
- マルチクライアント・サービスへの対応 ……クライアント・サーバモデルでは、一つのクライアントからの要求をサーバ（群）が解決し応答するという仕組みであるが、グループウェアではクライアント同士のインタラクションが前提となり、このためのインタフェースが必要となる。

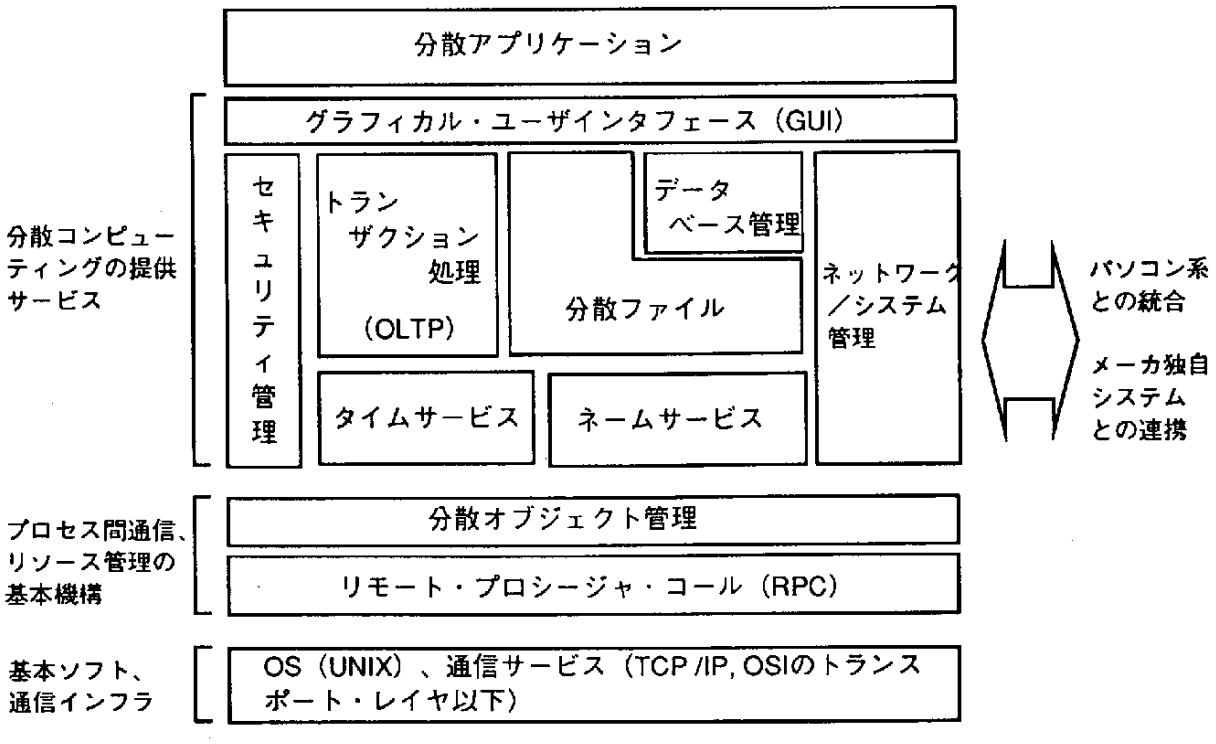


図4.1-6 分散コンピューティング・アーキテクチャ

4.1.6 マルチメディア文書通信

利用者がマルチキャスト情報を通信しながら処理するには、何らかの規約が必要となる。例えば、端末のマルチウインドウ上に表示された二次元のマルチメディア情報（各々のウインドウには、文字、図形、イメージなどのメディアで構成）、あるいは表示イメージを１頁としてその集合体として複数頁として格納した三次元のマルチメディアドキュメント（文書）を、送り手と受け手との間で、受け手で同じ位置に表示あるいは同じ形式で格納できるように、マルチメディアの情報構造を何らかの形で規約化しなければならない。なぜなら、端末に表示あるいは格納する情報は二次元あるいは三次元であるが、通信するときは送信側で一旦一次元のビット列に展開し、受信側で元の二あるいは三次元の情報に復元する必要があるからである。日本電気において、グループウェアのプラットフォームとして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では、構造化されたマルチメディア文書の交換規約として、ODAと類似のプロトコルが採用されている。

通信を前提としたマルチメディアプロトコルには、ISO（国際標準化機構）を中心に標準化を進めているマルチメディア文書通信プロトコルODA（Open/Office Document Architecture）がある。ODAでは、文字、図形、イメージという現在コンピュータで扱えるメディアのみがマルチメディアの対象範囲となっており、これらの離散系（蓄積）メディアのみで構成される構造化文書の表現形式（フォーマット）と交換手順が規約化されている。動画と音声の連続系（リアルタイム）メディアの取り込みは未だなされていない（図4.1-9）。

文書の表現形式には、図4.1-7に示すようにレイアウト構造と論理構造の２種類があり、同一文書を両方の視点から構造化できる点が特徴になっている。レイアウト構造は、ディスプレイやプリンタへの表示や出力する空間的（物理的）位置を指定する視覚的な情報である（図4.1-8）。論理構造は、目次、章、節、項、索引など文書の構成内容に関連する意味情報である。リアルタイムメディアの取り込みとそれに伴うメディア間同期のための動作記述、OSI参照モデルのレイヤ４以下のプロトコル（特に上記の拡張版）との実装を含めた整合などが当面の課題になっている。

マルチメディア情報の構造化に関しては、図4.1-9に示すようにODA以外に、HyTime（Hypermedia/Time-based Structuring Language）やMHEG（Multimedia and Hypermedia information coding Expert Group)、SMDL（Standard Music Description Language）があるが、ODA以外は、１コンピュータ内での処理すなわちスタンドアロン（通信なし）での使用を前提にした、情報の構造化あるいは情報間の関係づけの方法を規定している。

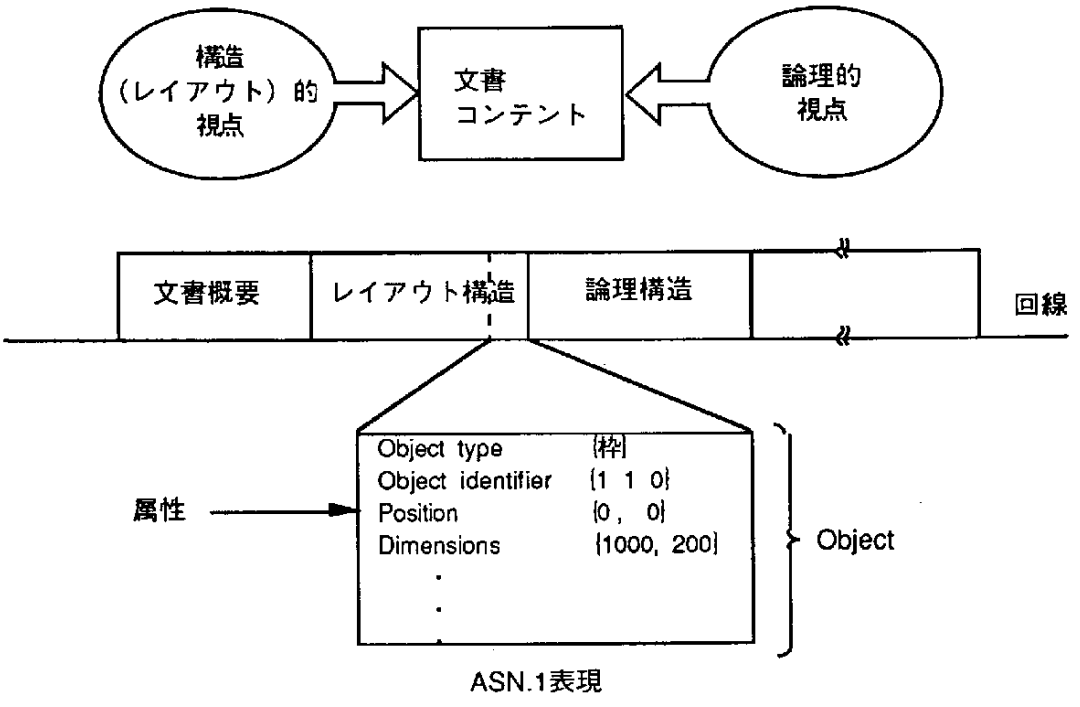


図4.1-7　文書構造とODIFデータストリーム例

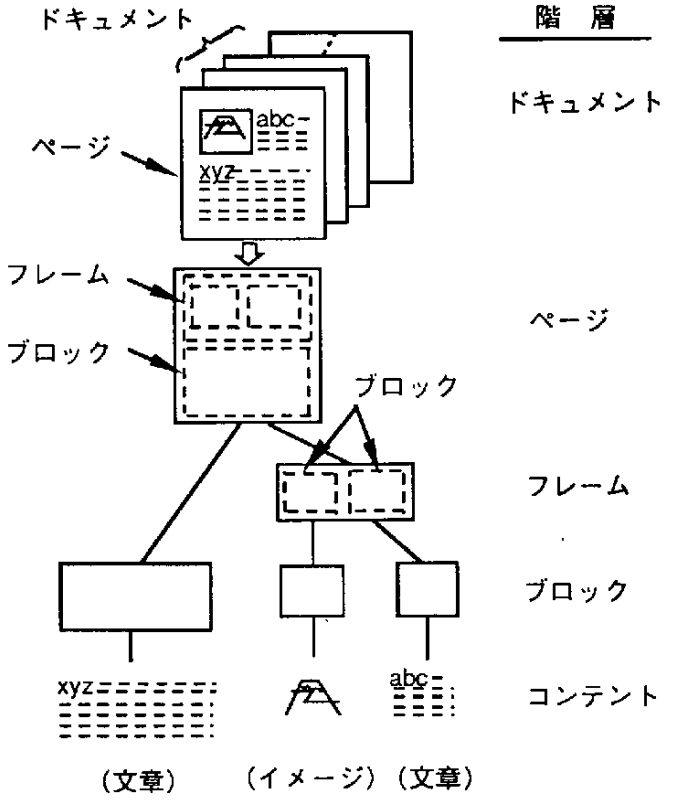


図4.1-8　ODAにおけるマルチメディア文書のレイアウト構造の例

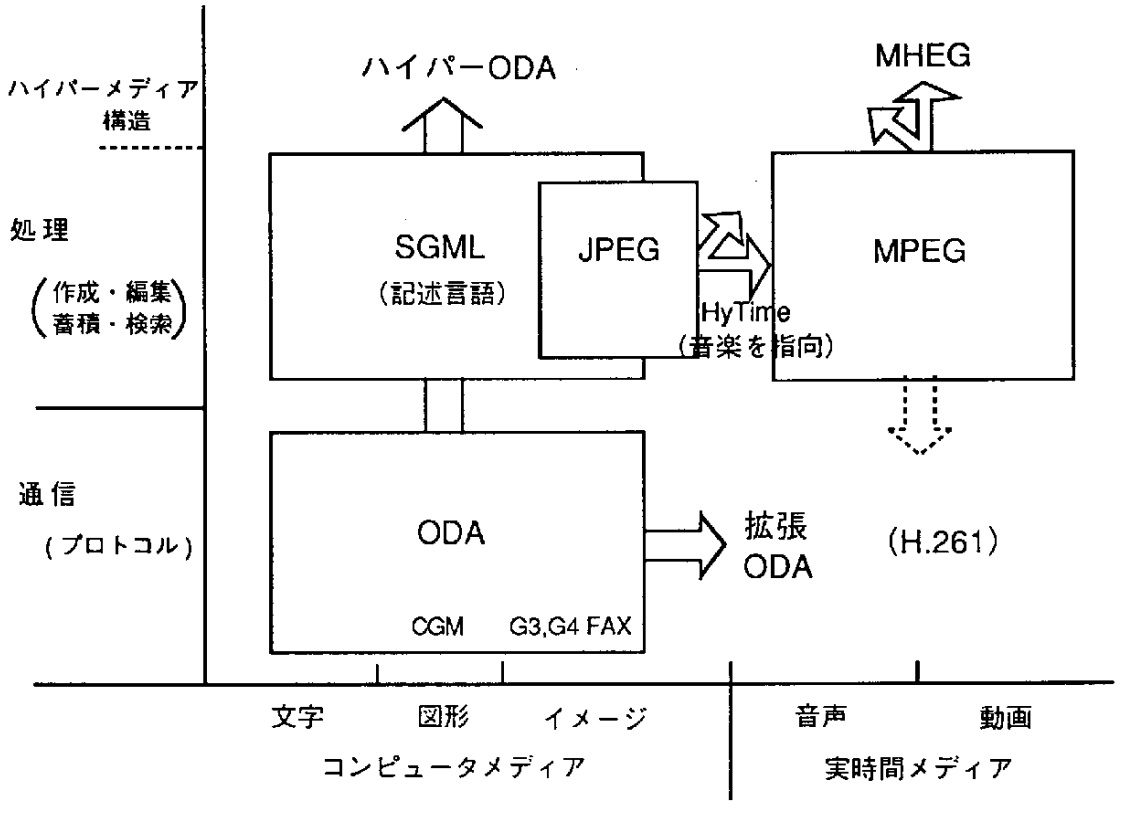


図4.1-9　マルチメディア情報表現・プロトコル標準化動向

マルチメディア文書通信プロトコルに関する標準化の課題はまさにこの点にある。すなわち、現在検討されている標準は、

- 蓄積（コンピュータ）メディアの処理のみを対象
- 蓄積（コンピュータ）メディアの通信のみを対象
- リアルタイムメディアの処理のみを対象
- リアルタイムメディアの通信のみを対象

のように、グループウェアのような通信とコンピュータの有機的結合（C&C）が重要な情報システムや応用に対し、通信とコンピュータの双方を包含したプロトコルがない、という点である。MHEGではこの反省から、ODA拡張（ODA EXtension）連携して、通信と処理を統合化したマルチメディアの標準化の推進を開始している[阪田92c]。

以上のマルチメディア通信に関連したメディアの統合、プロトコルの高速化とその処理、マルチキャスト機能についての技術動向を表4.1-3にまとめて示す[阪田92c]。

表4.1-3　分散コンピューティングの変遷

第1世代 (～80年代末)	分散OSの研究 ・ローカル（LANなどで結合）な利用環境を主な対象とした分散ファイル機能の提供
第2世代 (90年代前半)	分散コンピューティング基盤環境構築 ・クライアント・サーバ・モデル, RPC機構をベースとする ・分散処理のための機能群を整理し体系化
第3世代 (90年代半ば～)	オブジェクト指向概念の浸透による大規模な単一システム環境構築 ・ORBベースのエンドユーザレベルでの分散透過性実現 ・グループウェアとの融合

また、グループウェアにおいては、マルチメディア文書通信プロトコルの上位に、グループウェア特有の情報共有制御（共有情報の同一性維持のための排他制御）、上位のアプリケーションの分散協調制御（分散配置されたアプリケーション同士が互いにインターワークしながら協調動作を行うための支援）がCS（Computer Supported）の部分の基本機能として位置づけられるが、これらは作業空間の共有技術に対応するので、第3.7節で述べる。

【参考文献】

[阪田92a]阪田: マルチメディアとネットワークによるグループウェアの実現技術, ソフト・リサーチ・センター (1992.5).

[広帯域ISDN92]広帯域ISDN推進協議会: 広帯域ISDN推進協議会報告書(1992.5).

[電子情報通信学会91]電子情報通信学会: 広帯域ISDN特集(1991.11).

[阪田92b]阪田: グループウェアのためのマルチメディア高速多地点通信の鍵, 日経ニューメディア別冊・ディジタルテレビ, pp.115-118 (1992.11).

[Dupuy92]Dupuy, S. et al. : Protocols for High-speed Multimedia Communications Networks, Computer Communications, pp.349-358(1992.7).

[安田92]安田: マルチメディア符号化の国際標準と最新動向, 情報処理学会, アドバンストデータベース研究会(1992.12).

[阪田91]阪田: CSCWにおけるマルチメディア技術, 計測自動制御学会, pp.497-504(1991.6).

[阪田92c]阪田: マルチメディア技術, 化学と工業, pp.1842-1852(1992.10).

4.2 グラフィカルユーザインタフェース

4.2.1 GUIとグループウェア

近年、ワークステーションはもちろんのこと、ノートブック型のパソコンにおいても、ウインドウシステムをベースとするグラフィカルユーザインタフェース（以後GUIと略す）が使用可能になってきており、GUIは、人間同士のコミュニケーションをコンピュータとネットワークとを通じてサポートするグループウェアにとって、ほとんど不可欠ともいえるシステム要素である。むしろ、GUIを中心とするユーザインタフェース技術が、飛躍的に増大したコンピュータの処理能力によって発達し、同様に高速化したネットワークと結びついて、グループウェアとしてのコンピュータの利用を可能にしてきたとも言える。

歴史的に見ると、コンピュータを使ってマルチユーザで協同作業を行なうこと自体は、今まで作られてきた個々のアプリケーションシステムを考えてみれば容易にわかるように、決して新しいことではない。半導体技術の急速な発達による高機能で比較的安価なパソコンやワークステーションの普及とネットワークの高速化によって、GUIを利用しインタラクティブで柔軟性のある協同作業が、個々のユーザの作業空間を結び合わせた形で実現可能となり、グループウェアという視点が生まれてきたのであろう。これによって、個人を集めてグループができるという自然なアナロジーでコンピュータによる協同作業を一般的にとらえることができる。

また、個々のアプリケーションで対応していた協同作業に必要な機能を、より下位層においてプラットフォームとして共通に支援することも考えられるようになった。このグループウェアのプラットフォームで必要となる重要な機能として、分散環境における資源の共有とアクセスの制御、およびそれに必要な情報の同報や分配の制御がある。以下において解説する共有ウインドウ、テレポインタ、動画像についても、この二つの制御の実現方法がパフォーマンス、コスト、などに大きく影響を与える。

グループウェアは、そのユーザのコミュニケーションが、電話のように同期的（リアルタイム）に行なわれるか、メールのように非同期的に行なわれるかによって、同期型（リアルタイム型）と非同期型に分類される。本節では、特に同期型のグループウェアに注目し、そのマルチユーザ用のGUI（マルチユーザインタフェースとも呼ぶことにする）としてのいくつかの基本機能およびそれを実現するための技術について解説したい。具体的には、4.2.2および4.2.3では、WYSIWIS（What You See Is What I See）を実現するための機能としての共有ウインドウとテレポインタの機能について、そして4.2.4ではリアルタイムの動画像表示機能について、それぞれの周辺の技術にも触れながら、技術的な話題を取り上げる。

4.2.2 共有ウインドウ

共有ウインドウ機能は、リアルタイムのデスクトップ会議システムにおける共有黒板機能のように、同期型のグループウェアにおいて複数ユーザが同じ情報を同時に見ながら（ビュー（View）を共有しながら）、互いにコミュニケーションをとる場合に必要となる機能であり、多くの製品や試作システムに実装されている [Aluja88] [Stefik87] [Stefik88] [Tanigawa91] [Watabe90] [安藤92] [中山91]。

この場合、マルチユーザによって共有され、システムによってアクセス制御されるべき資源は、オペレーティングシステムの提供するウインドウ表示機能（OS/2[*1]のPM（プレゼンテーション・マネージャ[*2]）、UNIX[*3]系のXウインドウ[*4]、DOSのWindows[*5]など）である。図4.2-1に示すように、アプリケーションがGUI（マルチユーザインタフェース）を通じて共有ウインドウに表示した情報は、複数ユーザで同時に見ることができる。一方、各ユーザからの入力イベントも、GUIを通してアプリケーションに（何らかの方法でアクセス制御を経た後）渡され、共有ウインドウ上の情報を更新することになる。

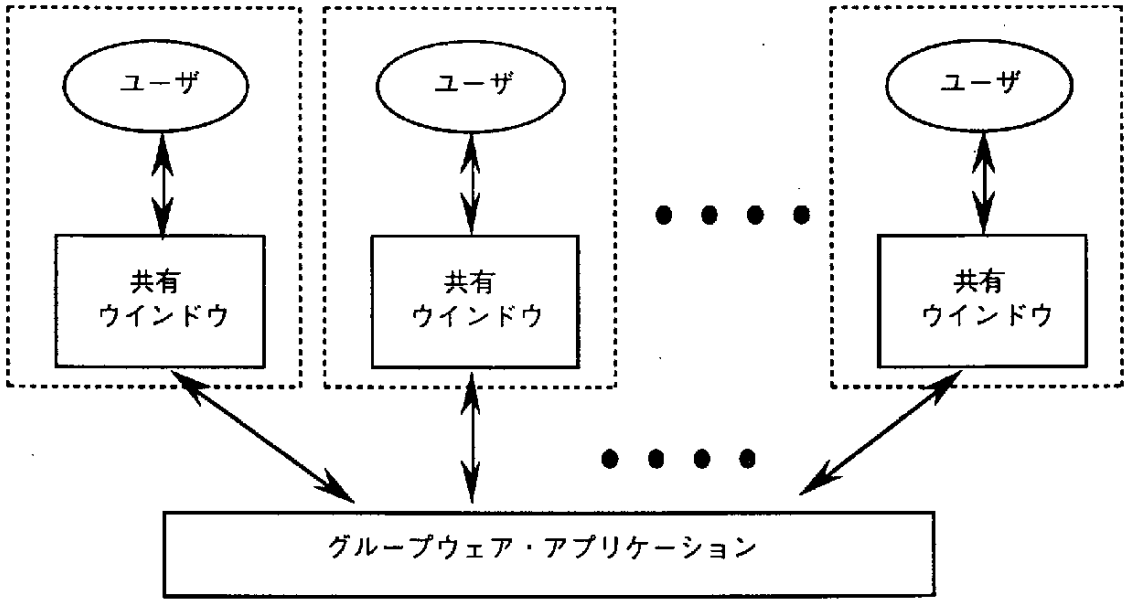


図4.2-1 グループウェアにおける共有ウインドウ機能

現在、グループウェアのプラットフォーム開発に多くの試みがなされており、将来的には、新たなプラットフォーム上でマルチユーザインタフェースのアプリケーションを開発していくことになるであろう。実際には、こうした試みは、既存のオペレーティング・システムの環境をマルチユーザ用に拡張し、今までのアプリケーション・プログラム資産の移行性を考慮しつつ行われることが多く、いかに既存の（シングルユーザインタフェースの）アプリケーションをサポートするかということは、共有ウインドウの設計における重要なポイントとなっている。

とりあえず既存のアプリケーションの議論は保留して、まず一般的に共有ウインドウを持つグループウェア・アプリケーションの実行形態を考えてみると、図4.2-2の(a)のように、アプリケーション自身がある一つのワークステーションで実行され、結果が分配される「集中型」と、(b)のように、アプリケーションはそれぞれのワークステーションで実行され、アプリケーション同士が通信し合って全体として同期をとっていく「分散型」があることがわかる。それぞれの特徴を挙げてみると以下のようになる。

【集中型】 1. アプリケーションの同期制御が容易である。
（実行結果の分配およびユーザからの入力イベントの処理が一つのワークステーションで行われる。）

2. 実行結果を通信回線を通して分配するので、結果のデータ量が多い場合（イメージデータなどの場合）にはネットワークの負荷が大きくなる。そのため、高速の通信路を利用するか、データを圧縮するなどの工夫を行なわないと応答性が悪くなることがある。

【分散型】 1. 基本的にオペレーション実行（例えば、共有ウインドウのある場所にこの大きさの円を描け、など）を指示するためのデータがやりとりされるので、ネットワークの負荷が小さい。

2. アプリケーションの実行がユーザ自身のワークステーションで行なわれるので応答性が良い。

3. 一連のオペレーションが行なわれる時、全てのワークステーションで同一の順序で実行される必要があり、一貫性（consistency）を保持するための同期制御が難しい。例えば、ほとんど同時に複数の相手からオペレーション実行の指示が出された時、何もしなければ、ネットワークの遅延等によってその複数の指示の順番がユーザごとに違い、共有ウインドウ内の情報が同一でなくなることが考えられる。

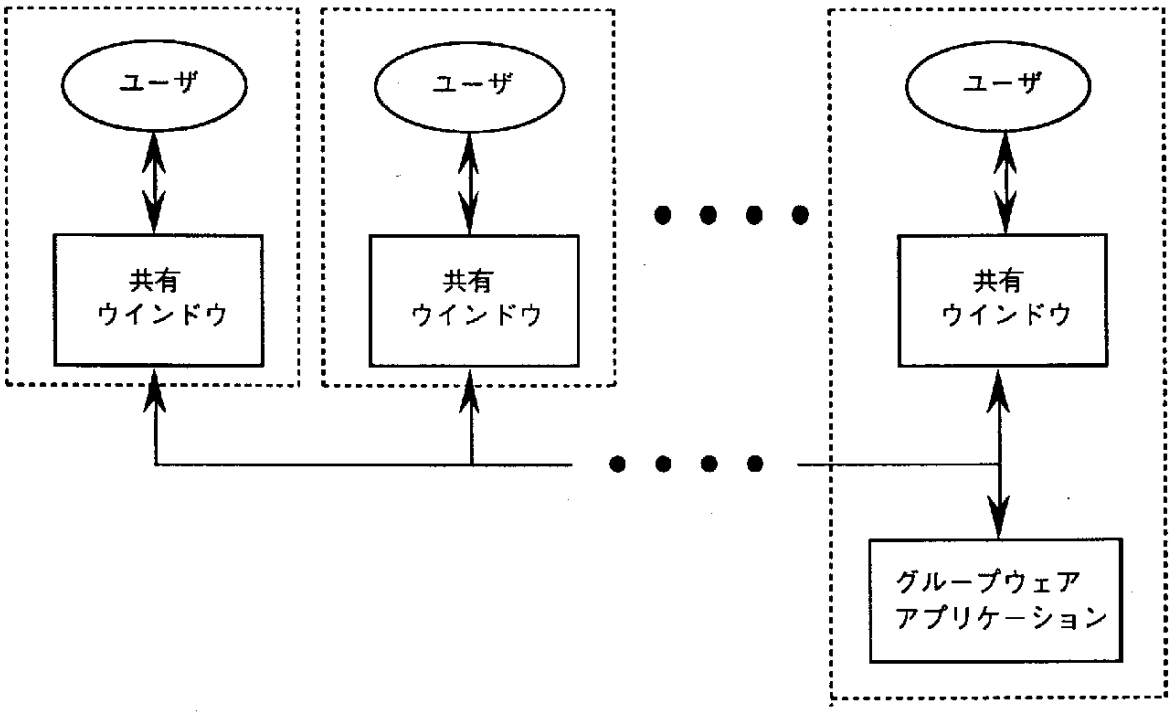


図4.2-2(a)　集中型実行形態

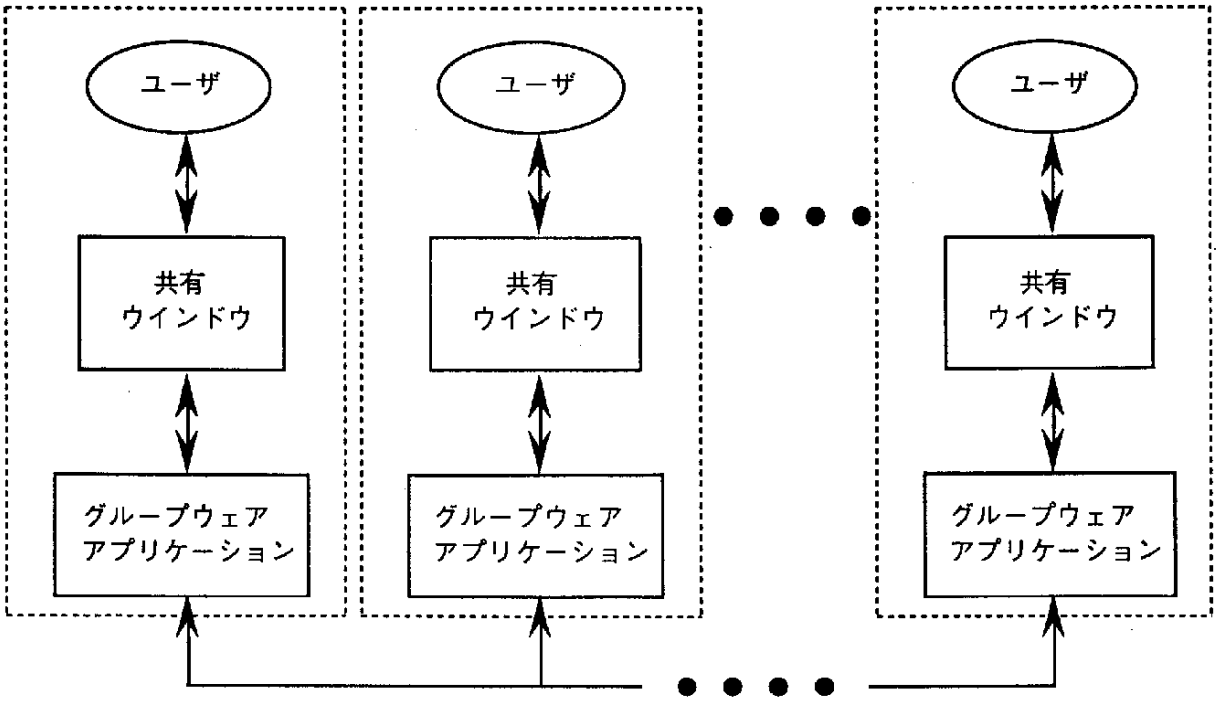


図4.2-2(b)　分散型実行形態

この二つの形態における共有ウインドウの実現方法は、利用するオペレーティング・システムに依存する。例えば、Xウインドウの場合には、集中型の形態を実現するには複数のサーバ(Xウインドウ)と通信するクライアント（アプリケーション）を作成すればよく、分散型を実現するにはクライアント間の通信メカニズムを利用することになるであろう[渡部92]。OS/2上でプレゼンテーション・マネージャと通信機能を用いて、遠隔マルチメディア会議システムを開発した例がある。このシステムの共通黒板機能の共有ウインドウは、現在は分散型として実現されており、実際には各ユーザのワークステーション上のグラフィックエディタが協調して動作することでWYSIWISを達成している。

それではつぎに、既存の（シングルユーザ）アプリケーションを「グループウェア化」し共有ウインドウで利用することについて、再び分散型と集中型の二つの形態において考えてみる。

【分散型グループウェア化】

アプリケーションはマルチユーザを考慮しておらず、それ自身で互いに協調動作することはないので、分散型として考えられる方法は、アプリケーションを各ワークステーションでそのまま動作させ、ユーザのイベント（マウス操作、キー操作など）を複数ユーザに分配する方法である。この場合、共有ウインドウ上データの一貫性を保証するため、以下の条件が必要となる。

1. 各ユーザのアプリケーション実行環境が同一であること。

   同一でない場合でも、何らかのインテリジェントなマッピングで個別に対応できることもあるかもしれないが、一般的にはユーザのイベントに対して同じ動作をすることは保証されない。

2. 各ワークステーションにおいて、アプリケーションがあるイベントに対する処理を完了したかどうか判ること。

   この条件は、アプリケーションを同期を取りながら動作させるために必要なもので、もちろん前提として複数のイベントの順序関係が保証されていなければならない。

【集中型グループウェア化】

この方法は、あるワークステーション上で動作するアプリケーションを、あたかもマルチユーザ用であるかのように見せるもので、そのアプリケーションのウインドウの内容が変化した時に全てのユーザの（共有）ウインドウにその変更を同報し、また複数のユーザからの入力イベントをアプリケーションに渡すための仕組みを作る必要がある。Xウインドウの例であれば、クライアント側で複数のサーバと通信するようライブラリを改造する方法や、複数サーバと通信する疑似的なサーバを作成する方法などがある。この集中型を設計する上での考慮点は、アプリケーションのウィンドウ内容の変更をどうやってとらえるかということである。

OS/2 PMの環境で、アプリケーションのウインドウシステムに対する処理要求をインターセプト（intercept）して、その要求を他のユーザに同報し、他のユーザからの入力イベントは、アプリケーションに対してリダイレクト（redirect）する試みを行なった。転送データ量が少ないので応答が良いシステムとなったが、API（アプリケーション・プログラム・インタフェース）によるウインドウシステムに対する処理要求をインターセプトしているため、処理要求のセマンティクスを解釈しウインドウシステム自身が持っている内部データまでシミュレートしなければならない場合もあった。APIの数も多く、またウインドウシステム自身のバージョンアップにも対応しなければならないので、この方法が全てのアプリケーションプログラムに適応できるという確証は得られなかった。

より低レベルのインタフェースを使って、ウインドウ内の変更を分配する方法としては、変更されたデータをイメージデータとして取り出して転送する方法がある。Timbuktuはこの方法をとって共有ウインドウを実現している。またIBMのPerson-to-Person/2（以後略して“P2P”とも呼ぶ）[P2P92]という製品のミラー（mirror）機能は、OS/2のアプリケーションのウインドウに表示されたデータを、必要に応じてチョークボード（chalkboard）と呼ばれる共有ウインドウにイメージとして取り込める機能である。この方法ではデータ転送量が多くなるので、高速の通信路を利用できない場合は、データの圧縮を考える必要がある。

4.2.3 テレポインタ

前述したように、同期型グループウェアの複数のユーザは、共有ウインドウ機能によって、同じ情報を同時に見ることができる。さらに、テレポインタ機能によって、他のユーザがマウスなどのポインティングデバイスを使って共有ウインドウ上で指し示している様子をリアルタイムに見ることができる。これは、遠隔地にいる相手との正確なコミュニケーションに不可欠な機能で、ほとんどすべての同期型会議システムで実装されている。テレポインタ機能の実現には、通信サービスに同報の機能がありまた分散データの一貫性を保証する仕組みがあるならば、それほど困難な技術的な問題は無いが、特にユーザビリティの観点から以下の点が論じられることが多い。

(1) テレポインタの数と制御方法

まずユーザが使えるテレポインタの数を幾つにするかという点である。各ユーザが一つずつテレポインタを持つこともできるが、ユーザが多数の時に共有ウインドウ上に多くのポインタが表示され、それぞれ独立に動き回って見にくくなる、という報告[Stefik87]もある。また、複数の共有ウインドウが存在できる時に、各ウインドウごとに別のテレポインタを用意して切り換えながら使うのか、それとも複数ウインドウで共通に使えるようにするのかということも考慮点の一つである。ユーザごとにテレポインタを持ちうる場合にはユーザの識別情報を（テレポインタ上に）どのように表現するかも考えなければならない。

文献 [Nakajima90] [中島92-3] に見られる疎結合（Loosely Coupled）方式のテレポインタは、ウインドウシステムのシステムポインタをテレポインタとしても利用するもので、自分がポインタを動かした時に、ある時間内相手のポインタが動かなければテレポインタの制御権を取り、もしこの時間内に相手も動いた場合は、自分のポインタも相手のポインタも独立に動くというものである。この方式では制御権の受け渡しが簡単であり、電話で話しながらの特に1対1のコミュニケーションではこの特長を生かすことができる。しかし自分のポインタが相手に制御されている時に、何かシステムポインタを動かして作業をしようとすると、作業はできるが（ポインタの色で相手が動かしていることはわかるが）相手のポインティング動作が見えなくなるので、特にユーザが様々な動作を行なう可能性がさらに高い複数ユーザでの使い勝手向上にはもう少し工夫が必要であろう。また、IBMのP2Pでは各ユーザがテレポインタを持つことができ、共有ウインドウ（チョークボード）上でマウスのボタンを押しながら動かすと、その様子が相手に見えるよう設計されている。

(2) ウインドウとの関係

厳密過ぎるWYSIWISの実現は、各ワークステーションユーザの自立性を阻害し、かえって使い勝手を損なうことがあるので、通常WYSIWISは共有ウインドウの中だけで実現され、ワークステーション画面上のウインドウの配置、重なり具合、ある場合には共有ウインドウの大きさや見えている範囲などは、各ユーザの自由に変えられるように設計されることが多い。この場合にはテレポインタが隠れて見えなくなってしまう等の問題もあり、いろいろな解決方法が考えられている[中島92-1]。

4.2.4 動画像

リアルタイムの動画表示機能によって、ワークステーションの同期型グループウェアはテレビ電話やテレビ会議システムの機能も合わせ持つことになる。ワークステーションのビデオカメラから取り込まれた動画は、デジタル化されて圧縮された後、通信回線上を転送され、相手側のワークステーションではこの圧縮された動画を伸張して画面上に表示する。最近ではパソコン用の動画圧縮伸張用ボードも開発され、これをビデオオーバーレイの機能と組み合わせることによって、ワークステーションのウインドウに動画を表示することができる。

例えば、文献 [安藤92] [中島92-2] に見られるシステムは、共有ウインドウ、テレポインタ機能に加え、国際標準のCCITT H.261(Px64)勧告 [マルチ91] に準拠した動画圧縮伸張機能（コーデック）をISDN基本インタフェース（2B+D）上でサポートするボード [榎92] を利用して、ISDNの128キロビット/秒の回線上でリアルタイムの動画と音声の交換を可能にしている。さらに、この試作ボードは、回線を通じて受信した動画と音声をワークステーション（のハードディスク）に圧縮したまま記録し、後で再生できる機能を持っているので、ビデオ留守録やビデオメールなどの非同期型のコミュニケーションにも利用することができる。また、IBMのP2Pのリアルタイム動画は、Action Media/2というボ

ドを使いDVI（Digital Video Interactive）の圧縮方式で実現されている。

動画表示機能は、最近は急速に下がりつつあるが一般家庭にも普及するにはまだコスト高で、必ずしもグループウェアで不可欠とは考えられないという報告 [Helms91] [Chapanis75] もあるコミュニケーションメディアであるが、おそらく使い方としては、単に相手の顔や様子を見るだけでなく、なにかジェスチャーを見せる必要があるとき（機械の操作法を教えるなど）や、静止画（および静止画の連続）ではわかりやすく見せられないとき（例えば、長い巻物の写真をとるにはビデオが有効であるそうである）に役立つ機能である。最近の新しい傾向としては、CPUの高速化と共に、特別なハードウェアを必要としないソフトウェアによるリアルタイムの動画圧縮伸張（ソフトウェアコーデック）が現れてきたことが挙げられる。このアプローチは、今後パフォーマンスと画質の向上が計られれば、さらに急速に動画表示機能のコストを下げていくことが期待される。

4.2.5 新しいGUI

最近の研究の傾向としては、ビデオを主要なメディアとして利用し [Ishii90]、視線一致の技術も含めて開発されたグループウェア [Ishii92] [Kazuoka92] [Tang91] や、三次元グラフィックシステムを利用した一つの仮想現実感（Virtual Reality）システムの例として、遠隔ユーザに対し三次元の共同作業空間を作り出す研究 [篠原92] に関する報告などに見られるように、グループウェアのGUIとしてさらに多くのヴァリエーションが現れてきた。このような様々なメディアをGUIとして利用することは、技術的には可能になってきてはいるが、実際にアプリケーションと組み合わせて業務で効果的に利用できるようになるためには、インフラの整備、さらに低いコストの実現、そしてさらに多くのテストを行なっていく必要があるであろう。

*1： IBM Corp. の商標
*2： IBM Corp. の商標
*3： Unix System Laboratories, Inc. の開発許諾製品
*4： MIT の商標
*5： マイクロソフト社の商標

【参考文献】

[Aluja88] Aluja, S.R., Ensor, J.R. and Horn, D.N.: The Rapport Multimedia Conferencing System, Proc. ACM Conf. on Office Info. Syst., pp.1-8 (1988).

[Chapanis75] Chapanis, A.: Interactive Human Communication, Scientific American 232(3), pp.36-42 (1975).

[Helms91] Helms, R.M.: Distributed Knowledge Worker (DKW): A Personal Conferencing System, IBM Canada Laboratory Technical Report TR-74.067 (1991).

[Ishii90] Ishii, H.: TeamWorkStation: Towards a Seamless Shared Workspace, Proc. CSCW '90, pp.13-26 (1990).

[Ishii92] Ishii, H. and Kobayashi, M.: ClearBoard: A Seamless Medium for Sharing Drawing and Conversation with Eye Contact, Proc. of ACM SIGCHI Conf. on Human Factors in Computer Systems, pp.525-532 (1992).

[Kazuoka92] Kazuoka, H.: Spatial Workspace Collaboration: A Sharedview Video Support System for Remote Collaboration Capability, Proc. of ACM SIGCHI Conf. on Human Factors in Computer Systems, pp.533-540 (1992).

[Nakajima90] Nakajima, A. and Fin, T.-H.: A Telepointing Tool for Distributed Meeting Systems, Proc. IEEE GLOBCOM '90, pp.76-89 (1990).

[P2P92] PS/2 Person to Person Flyer, G362-1052, IBM Corp. (1992).

[Sarin85] Sarin, S. and Grief, I.: Computer-Based Real-Time Conferencing Systems, IEEE Computer, Vol.18, No.10, pp.33-45 (1985).

[Stefik87] Stefik, M., Bobrow, D.G., Foster, G., Lanning, S. and Tatar, D.: WYSIWIS Revised: Early Experiences with Multiuser Interfaces, ACM Trans. on Office Info. Syst., Vol.5, No.2, pp.147-167 (1987).

[Stefik88] Stefik, M., Foster, G., Bobrow, D.G., Kahn, K., Lanning, S. and Suchman, L.: Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings, Commun. ACM, Vol.30, No.1, pp.1-8 (1988).

[Tanigawa91] Tanigawa, H., Arikawa, T., Masaki S. and Shimamura, K.: Personal Multimedia-Multipoint Teleconference System, Proc. IEEE INFOCOM '91, pp.1127-1134 (1991).

[Tang91] Tang, J.C. and Minneman, S.L.: VideoDraw: A Video Interface for Collaborative Drawing, ACM Trans. on Info. Syst., Vol.9, No.2, pp.170-184 (1991).

[Watabe90] Watabe, K., Sakata, S., Maeno, K., Fukuoka, H. and Ohmori, T.: Distributed Multiparty Desktop Conferencing System: MERMAID, Proc. CSCW '90, pp.27-38 (1990).

[安藤92] 安藤 史郎, 中島 周, 黒沢 隆: 遠隔マルチメディア・プレゼンテーション・システム, 情報処理学会研究報告(マルチメディア通信と分散処理), Vol.92, No.35, 92-DPS-55, pp.63-70 (1992).

[榎92] 榎 隆司: マルチメディア通信への提案, "情報化メディア懇談会" 会員誌 情報化メディア No.104, 株式会社NHK情報ネットワーク, pp.14-19 (1992.10).

[阪田92] 阪田 史郎: グループウェアの実現技術, SRCハンドブックシリーズ, 株式会社ソフト・リサーチ・センター (1992.5).

[篠原92] 篠原 克也, 平池 龍一, 中村 暢達: 仮想現実感のネットワーク化, 情報処理学会研究報告(マルチメディア通信と分散処理), Vol.92, No.35, 92-DPS-55, pp.57-62 (1992).

[中島92-1] 中島 周: 共有ウィンドウシステムにおけるテレポインティングの問題, 第45回(平成4年度後期)全国大会講演論文集(1), 社団法人情報処理学会, pp.291-292 (1992.10).

[中島92-2] 中島 周: マルチメディア会議システム,International Symposium "Computer World '92" Program & Abstracts, 財団法人関西情報センター, pp.85-92 (1992).

[中島92-3] 中島周: 共有ウィンドウのための疎結合共有ポインタ, 情報処理学会研究報告(マルチメディア通信と分散処理), Vol.92, No.91, 92-DPS-58, pp.75-82 (1992).

[中山91] 中山 良幸, 森 賢二郎, 中村 史郎, 山光 忠: 多者間電子対話システム ASSOCIA, 情報処理学会論文誌, Vol.32, No.9, pp.1190-1199 (1991).

[松下91] 松下 温編著: 図解グループウェア入門, オーム社 (1991.8).

[マルチ91] マルチメディア技術動画像, 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, 財団法人パーソナル情報環境協会, pp.65-72 (1991.3).

[渡部92] 渡部 勇,神田 陽治: グループウェア実体験, ユーザ・インターフェース大作戦 4, bit, 共立出版, Vol.24, No.4, pp.410-419 (1992).

4.3 オブジェクト指向データベース

今日データベースはコンピュータデータの効率的な蓄積とアクセスのために、種々のアプリケーションで利用されている。またデータベースモデルも、最も良く使われている関係型から、階層型、ネットワーク型等様々な形態が存在し、対象とする計算機やアプリケーションに応じて使い分けられている。しかしながら、これらのデータベースは本来ビジネス分野での利用を前提にしていたため、複雑な機能の実現よりは効率性や信頼性に重点を置いて開発されてきた。そのため、それらが扱えるデータも計算機システムが元々持っているデータ型と大差ないレベルにしか過ぎず、またモデル化能力も貧弱である。これに対し、昨今のコンピュータ技術の進展により、扱うデータも従来の文字や数字に加えて音声や画像等までに拡大してきており、それらの持つ複雑なデータ構造や処理要求は、現在入手可能なデータベースの機能を大きく越えてしまっている。一例をあげるなら、協同作業支援では利用者に対して情報の同時更新を許している。これは、トランザクションを単一の原子的な単位として扱う従来のデータベース技術とはまるで反対方向にある。また製造業に見られるような並行エンジニアリングにおいては大量のCADデータを協同で扱う必要があるが、現状のRDBのようなデータベースシステムでは、それが扱えるデータのサイズや複雑さにおいて問題が多い。現状では、次のような機能が協同作業支援のために必要である。

- 分散し且つ共用する必要のあるデータに特有のダイナミックスを扱う機能
- 異なったタイプ、異なったサイズのデータを自由に扱える機能
- 複数のデータベース、システム間でデータを扱える機能

現在上記にあげたような問題に対する有望な研究分野として、以下にあげる二つのデータベース技術が注目を浴びている。

- オブジェクト指向データベース技術
- 分散データベース技術

本節ではオブジェクト指向データベース技術に的を絞って、協同作業支援との関わりについて考えてみる。

4.3.1 オブジェクト指向データベースとは

「オブジェクト」とは実世界に存在する物と、その物に固有の操作を一体化したものと考えることができる。この考え方を使えば実世界のモデル化が容易になるため、現在では、プログラム言語、OS、データベース等をいうに及ばず、ソフトウェアの分析設計手法といった分野にまで適用されつつある。しかしながら、余りに漠然とした定義のために、使う人毎にその定義が異なっており、混乱のもととなっている（この現象はオープンシステムという言葉にも見られるものである）。

ここではまず'89年に開催されたDOOD'89のオブジェクト指向データベースに関する宣言を参考にし

ながら、オブジェクト指向データベースの定義について考えてみる。この宣言は下記のような問題点を踏まえて出されたものである。

- 共通したオブジェクト指向データベースのデータモデルがないため、何がオブジェクト指向データベースなのかといった定義ができない。
- 関係データベースが持っているような、理論的な枠組が欠如している。
- 上記にも拘らず、多くのベンダーが製品としてオブジェクト指向データベースを出荷しており、世の中に混乱を招いている。

(1) オブジェクト指向データベース宣言

この宣言にはメッセージパッシングによる自律的なデータベースの挙動といったダイナミックスに関わる面が欠けていることや、協同作業支援の観点から見た時に備えるべき条件の分類（必須条件、付加条件、選択条件）が適切でない等の批判がある。しかしながら、世界的なレベルで見た場合、入手しうる唯一の定義であり、オブジェクト指向データベースに関するある程度の概念は把握できるはずである。批判に関しては、今後の議論を待つということにしたい。

① 必須条件

- 複合オブジェクトをサポートすること

  単純なオブジェクト（整数、文字、etc）に構成子（集合構成子、レコード構成子、配列構成子、リスト構成子等）を適用することで、複雑なオブジェクトを構成できること。
- オブジェクト識別性を持つこと

  オブジェクトはその値（属性）に関係なく識別できること。これはたとえ属性値が全く同一であっても、それらを異なったオブジェクトとして識別できることを意味する（等価であることと同一であることは異なる）。
- オブジェクトはカプセル化されていること

  オブジェクトの持つデータ（属性値）とそのオブジェクトに対する操作を一体化すること。
- クラスまたは型を持つこと

  複数のオブジェクトに共通する性質をまとめて、一塊の単位として扱えること（型）。もしくは動的なオブジェクト生成機能や複数のインスタンスの保持機能を持つこと（クラス）。
- 継承機構をサポートすること

  オブジェクトに階層構造を導入し、上位オブジェクトの持つ属性や操作をそのまま下位オブジェクトに引き継げること。
- 遅延束縛（late binding）をサポートすること

  継承に関係する機能であるが、操作名とそれに対応する実操作の結合が実行時に動的に行なわ

れること。

- 計算完全性

  データ操作言語を用いて任意の計算可能な関数を表現できること。

- 拡張可能性

  新しい型を定義できる手段が用意されており、かつ、システムで予め定義されている型とユーザが定義した型の使い方に何ら区別がないこと。

- データの永続性をサポートすること

  プログラムの終了に関係なくオブジェクトが永続的に保たれること。

- 大規模データが扱えること

  二次記憶管理ができることであり、またこの管理はユーザには透明であること。

- 並行処理が制御できること

  複数ユーザからの要求を同時に受け入れ、処理が可能なこと。

- 障害回復が可能であること

  ハードウェアやソフトウェアの障害に対して、回復手段を持っていること。

- 問合せ言語をもつこと

  データベースシステム毎の固有の問い合わせ言語を持つということではなく、標準的な言語をサポートすること。

② 付加条件（詳細は説明しない）

- 多重継承をサポートすること
- 型検査と型推論ができること
- 分散データベースをサポートすること
- 長時間トランザクションもしくは入れ子のトランザクションをサポートすること
- バージョン管理ができること

4.3.2 OODBと協同作業支援

協同作業支援においてデータベースに必要とされる機能と、それに対するオブジェクト指向データベースの適用を考えてみる。

(1) マルチメディアデータの扱い

今後協同作業に限らず、全く性質の異なったデータ（マルチメディアデータ=文書、音声、画像等）をひと塊の意味を持ったデータとして扱う場合が必要になってくる。こういったデータは構造的にも複雑であり、またデータごとにその操作も全く異なっている。従来のデータベースでは、こういった複雑な構造を持つデータを扱うだけのモデル化機能が備わっておらず、またそれぞれのデータに対す

る操作も応用プログラムの責任において実現する必要があった。そのため、オーバヘッドが大きくなってしまい、処理効率も悪いという欠点を持っている。

オブジェクト指向データベースでは、複合オブジェクト機能を使ってデータベース側で性質の異なるデータを一括管理することができる。また、カプセル化機能を使って、それぞれのデータに必要な処理をデータベース側にもたせることにより、応用プログラムの構造を簡単にすることができ、同時にオーバヘッドも減ることで、効率的な処理も可能になる。

(2) 粒度

協同作業においては、ドキュメント等に見られるように非常にデータ量の多いものを一つのオブジェクトとして扱う必要がある。従来の関係データベースではこういったデータをうまく扱う方法がない。たしかに、バイナリーの巨大なデータ等を扱う技術（BLOB）もあるが、これとても後から付け加えられた機能であるし、そのデータの一部を取り出して処理する方法は確立されていない。現状では、こういったデータは all or nothing で扱うしかない。これは協同作業支援にとって、その一部のデータを取り出して処理しようとするような場合には大きな問題となる。

オブジェクト指向データベースにおいては、複合オブジェクトを使うことで、こういったデータを小さなオブジェクトの集合として扱うことができる。例えばドキュメントの場合には、全体を章、節、句等に分割し、句を最小単位のオブジェクトとして、節を句オブジェクトの複合オブジェクト、章を節オブジェクトの複合オブジェクトというふうに、順次複雑な構造を持つオブジェクトを構築していくことができる。ただし、大きなオブジェクトをどういった基準で小さなオブジェクトに分解（構造化）するかという問題は別に残る。

(3) スキーマの動的変更

協同作業においては、多種多様の目的や背景を持った人間が集まって作業を行なう。そのため、作業の進展に伴って、そこで扱われるデータ構造（データフォーマット）はしばしば変更を受ける。その度にスキーマをスタティックに変更していたのでは、そのうちにスキーマと実データの間にミスマッチングを起こす危険性がある。これを解決するためには動的かつ自律的にスキーマを変更できるような機能が必要である。

一部の商用オブジェクトデータベースには、スキーマの動的変更の機能が実装されてはいるが、まだまだ不十分であり、スキーマ情報の永続的な管理や変更管理についてはさらに研究が必要である。

(4) アクセス権の制御

協同作業においては、あるユーザがになう役割は一連の作業の中で時々刻々変化していく。これに伴い、同じデータに対するアクセス権も変化していく必要がある。通常UNIXのようなOSの場合には、書き込み、読み出し、実行権がファイル単位、ユーザ単位に与えられる。しかしながら、こういったアクセス権のみでは協同作業支援のためには不十分であり、場合によってはOSレベルのアクセス権は

まったくフリーにしてしまって、コントロールを全てアプリケーションプログラムで行なう場合もある。しかしこれはアプリケーションプログラムを通さないアクセスに対しては全く無力であり、ユーザに悪意のある場合は非常に危険な方法である。したがって、これらのアクセス制御はデータベースシステム側で行なう必要がある。一般には以下のようなアクセス制御が必要となる。

① オブジェクトのクラスに対するアクセス制御

オブジェクトのクラス（型）に応じたアクセス制御を行なう。例えば、ドキュメントオブジェクトに対しては、読み書きだけではなく、注釈付けの権利を設定するといったことが考えられる。

② 属性によるアクセス権の制御

属性といった場合、二つの場合が考えられる。一つは利用者の属性であり、もう一つはオブジェクトの持つ属性である。

- 現在対象となるユーザがどういった役割を担っているかによってアクセス権を制御する。
- データ中の特定の属性によってアクセス制御を行なう。例えば、ドキュメントを作成した人間だけに変更を許すというようなものである。

現在、こういったアクセス制御メカニズムを備えたデータベースはないが、オブジェクト指向データベースの場合は、カプセル化されたメソッドを利用することにより、これらのアクセス制御を実現することが考えられる。

(5) データ管理の階層化

いかに協同作業とはいえ、グループで共用するデータだけではなく、個人のみに意味を持つデータも存在する。オブジェクト指向データベースの継承機構を使えば、共用データには何の変更も加えずに、それに必要な変更や追加・削除を加えた個人データを無理なく実現・管理することができる。

(6) リンク（関連）のサポート

協同作業において各データがお互いの間に関連を持ち、またその関連もダイナミックに変化していく。例えば、ドキュメントの中に別のドキュメントへの参照があり、それがネットワーク状につながりあっているということなどはざらである。実際ドキュメントを先頭から最後まで通して読むのはまれであり、自分にとって必要な部分だけを拾い読みしていく場合の方が多い。現状では、こういった要求に対応するものとしてハイパーメディアが存在するが、あくまでリンク（関連）の管理が主眼であり、データの管理は通常のファイルに任せている場合が多い。このデータ管理部分をオブジェクト指向データベースでに任せることにより、柔軟性にとんだ、信頼性の高いデータ構造を構築することができる。

(7) View

協同作業の場合、ユーザはそれぞれの役割に応じて異なった視点から同じデータを眺めようとすることが多い。関係データベースの場合は論理スキーマを使うことで、こういった要求を満たすことが

可能である。逆にオブジェクト指向データベースの場合、view機能のサポートがほとんどなく、今後の大きな課題となっている。

(8) 履歴管理と版管理

複数の人間が更新するデータに対して単純な版管理をしていたのでは、特定の人間の連続したデータ更新によって、他の人間による変更版が失われてしまう。そのため、協同作業のデータに対しては文脈を考慮した（誰が作ったどのバージョンなのか等）版管理、履歴管理が必要となる。現状ではこの要求に対するデータベース側からの解決策はなく、応用プログラムの責任において実現しなければならない。オブジェクト指向データベースを使ったとしても、今のところカプセル化されたメソッドを使って実現する以外方法はなさそうであるが、本来はデータベースマネージャで実現されるべき機能である。

(9) トリガー機能

特定のオブジェクトに特定の操作が加えられた時に自動的に呼び出されるプロシージャ。多数の作業者が並行して作業する場合、しばしばデータの一貫性の保証が問題となるが、トリガー機能を使うことで自動的に一貫性をチェックする機構を備えることができ、上記の問題をある程度回避することができる。また、データが変更を受けた場合に、関連するユーザに自動的に通知を送ったりすることもできる。

一部の商用データベースシステムには、この機能が含まれているが標準として定まったものではない。データの一貫性の保証に関しては、完全性制約の一部として実現される可能性がある。

(10) 同時実行制御・長時間トランザクション

協同作業支援においては、同時実行制御と長時間トランザクションとは切っても切れない関係にある。従来のデータベースではトランザクションをシリアル化することによって見かけ上の同時実行制御を行ない、データの一貫性を保ってきた。これは一つ一つのトランザクションの処理時間が短いという暗黙の前提にたってなり立つものである。一つの作業が長時間にわたるような場合（共著システム等）に同じ方法を適用したならば、作業に極端な制限が付け加えられることになる。オブジェクト指向データベース宣言をみても、長時間トランザクションに関しては付加条件となっており、現時点では有効な解決策というものは存在しない。

ただし、協同作業環境にあっては各ユーザに誰かが使用中であることを通知するだけでも効果があるといわれている。とりあえず通知程度の簡単な機能ならば、オブジェクト指向データベースのメソッドを使っても実装できるかもしれないが、本格的な解決策は今後の研究待ちである。

4.3.3 OODBの問題点

今までみてきたように、オブジェクト指向データベースは協同作業支援用のデータベースとして、かなりの可能性を有する。しかしながら、オブジェクト指向データベースには、以下にあげるようなそれなりの問題点も存在する。

- データモデルの機能の豊富さ故に、複雑なモデルをデータベースに取り込むためには、プログラマやデザイナさらにはユーザに対して大きな負担を強いることになる。このことより、現状ではオブジェクト指向データベースの応用分野はかなり限定せざるを得ない。
- 商業的にみてもオブジェクト指向データベースはまだ一般化しているとは言い難い。ここしばらくは、協同作業支援用のデータベースは従来型のファイルか、階層型のデータベースを使っていくものと思われる。そのため、将来的にみると、その時点まで蓄積されてきたデータとの整合性が大きな問題となる。
- オブジェクト指向データベース技術は開発が始まってから日も浅く、理論的なモデルも構築されてはいない。それ故現在種々のオブジェクト指向データベースベンダが独自の方法、規格でもって製品を開発中である。かつてのSQLの標準制定に10年が必要であったように、ここしばらくは互換性のない独自規格のオブジェクト指向データベースを使わざるを得ない。

なお、1991年の終りに、オブジェクト指向データベースのベンダーが集まってODMG（Object Database Management Group）という団体を結成した。この団体の目的はオブジェクト指向データベースに対する標準問い合わせ言語を制定することと分散データに対するオブジェクト指向データベースのインタフェースを制定することにある。一部ではオブジェクト指向データベースの技術が未だに確立されていないのに、焦って標準を制定しようとする動きを疑問視向きもあるが、有力ベンダによる標準制定の動きは将来のオブジェクト間の相互運用に向かって少なくとも政治的な雰囲気を作り出すことには役だっていると思える。

4.3.4 将来へ向けて

第4.3.2項の中でもオブジェクト指向データベースを協同作業支援に利用する場合に研究しなければならない項目を述べたが、それ以外にも下記のような機能の研究開発が望まれる。

- データの意味を扱える機能：現在は演繹的データベース（deductive DB）として精力的に研究が進められている。
- データの一貫性を自律的にサポートできる機能。
- トリガーや制約をルールとして扱える機能。
- 実時間性を必要とするデータに対する、時間を扱う機能。

【参考文献】

[田中91] 田中 克己: オブジェクト指向データベースの基礎概念, 情報処理,Vol.32, No.5, pp.500-513(1991).

[宇田川91] 宇田川 佳久: オブジェクト指向データベースのCADへの応用, 情報処理,Vol. 32, No. 5, pp586-592(1991).

[Atkinson89] Atkinson,M.,et al.: The Object-Oriented Database System Manifesto, Proc.of the 1st International Conference on Deductive and Object-Oriented Databases(DOOD89), pp.40-57(1989.12).

5．ソフトウェア開発における
グループワーク支援

5．ソフトウェア開発におけるグループワーク支援

5.1　ソフトウェアの開発プロセス

ソフトウェア開発プロセスの定義は明確でない。広い視点に立つとソフトウェア開発に関与する人が、それぞれの立場で行なう作業の集合体と定義できるし、狭い視点に立つとソフトウェア開発の工程と定義できる。この場合、狭い視点でのソフトウェアとは「データ処理システムの運用に関係する計算機プログラム、手順規則及びそれらに関する文書」（JISの定義）と定義される。

広い視点でのプロセスとは、発注者と受託者、管理者と開発者等といった関与者が行なう作業の集合で「ソフトウェアライフサイクルプロセス」と呼ばれている。ISOとIEEEではこのソフトウェアライフサイクルプロセスの規格化を進めている。

狭い視点でのプロセスとは、ソフトウェアライフサイクルプロセスが作業の集合であるのに対し、時間軸による作業の流れを中心にしたプロセスで「ソフトウェア開発工程」と呼ばれている。このプロセスの標準化は主にコンピュータメーカ、コンサルタント会社を中心に実用化されており、標準手順やシステム開発方法論と呼ばれている。

グループワーク支援システムでは上記ソフトウェアライフサイクルプロセスとソフトウェア開発工程の両者を支援する必要がある。

5.1.1　ソフトウェアライフサイクルプロセスの国際標準化

ISO/IEC JTC1/SC7のWG7ではソフトウェア開発における購入者と供給者の二者間契約の客観性を明らかにして無用な混乱、誤解を防止する目的でソフトウェアライフサイクルプロセスの標準化を進めている[村上92]。この標準化の特徴はプロセスを三つの視点に分類し、視点毎の作業の枠組みを規定している点で、これにより特定の開発技法、開発手順から独立させ普遍化を図っている点である。図5.1-1にソフトウェアライフサイクルプロセスの概要を示す。作業は「プロセス」「アクティビティ」「タスク」と３レベルに階層化されている。管理の視点は管理者の作業で、他のプロセスの管理作業のプロセスを定義している。例えばプロジェクト管理作業は開発プロセスを対象にした管理プロセスである。契約の視点では購入者と供給者それぞれのプロセスが定義されており、このプロセスを相互に認識して契約行為が行なわれる。最後のエンジニアリングの視点は開発者、運用者、保守者の作業で、開発プロセスでは分析・計画からソフトウェア開発・テストまでの作業で構成されている。この他に支援プロセスとプロセスの開発、評価と改善の作業がある。支援プロセスは文書開発、ソフトウェア構成管理/品質管理、教育とソフトウェア開発環境の設定プロセスである。

ISOの他にIEEE P1074でもソフトウェアライフサイクルプロセスの標準化を進めている。IEEE標準化案の目的もISOと同様にソフトウェア開発と保守プロセスの標準化である。IEEE標準化案[IEEE92]で

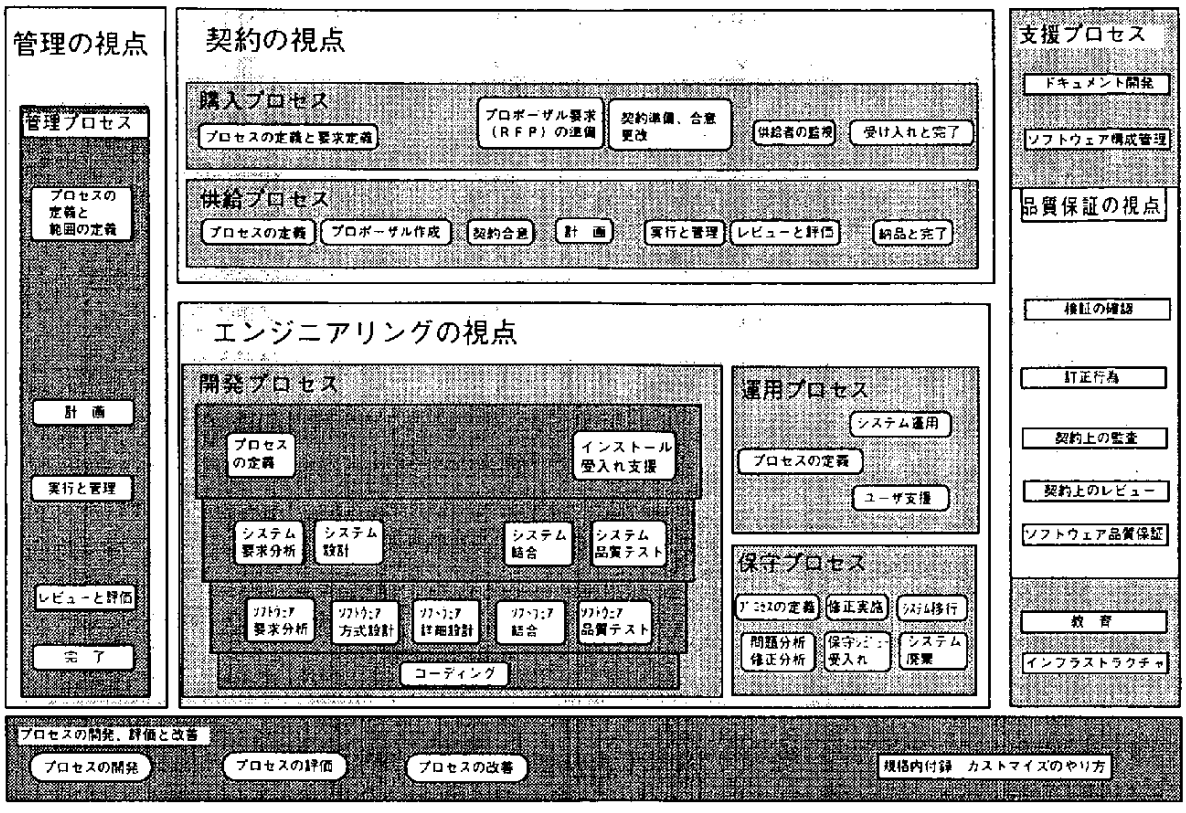


図5.1-1　ISO/IEC JTC1/SC7委員会案:ソフトウェアライフサイクルプロセスの概要

はソフトウェアライフサイクルプロセスは六つのセクションに分類した１７プロセスで構成され、さらに６５個のアクティビティに細分される。表5.1-1にプロセスの一覧を示す。各アクティビティには入出力アクティビティ名と情報及びアクティビティの内容が定義されている。例えばソフトウェアライフサイクルモデルプロセスは「ソフトウェアライフサイクルモデル候補の抽出」と「プロジェクトモデル選択」の２アクティビティがあるが、標準化案には次のように書かれている。まずソフトウェアライフサイクルモデルを外部より入力し、いくつかの候補を抽出しプロジェクトモデル選択アクティビティに送る。つぎにこの候補の中からプロジェクトの過去の経緯と方針を勘案してプロジェクトで適用するソフトウェアライフサイクルを選び、プロジェクトイニシエーションプロセスに情報を送る。

IEEEのSectionもISOと同じような視点で分類されている。ただしISOにある契約の視点からのプロセスは明示されていない。ソフトウェアライフサイクルモデルの選択、プロジェクト管理と品質管理の管理プロセス、ユーザからのニーズ確認や効果算定を行なう開発前の評価プロセス、要求定義からソフトウェア作成までの開発プロセス、運用と保守のプロセスがあり、その他に検証と確認、ソフトウェア構成管理、ドキュメント開発、教育のプロセスがある。

表5.1-1 IEEE案：ソフトウェアライフサイクルプロセス一覧

No	SECTION	PROCESSES
1	Software Life Cycle Model Process	Software Life Cycle Model
2	Project Management Processes	Project Initiation Project Monitoring and Control Software Quality Management
3	Pre-Development Processes	Concept Exploration System Allocation
4	Development Processes	Requirements Design Implementation
5	Post-Development Processes	Installation Operation and Support Maintenance Retirement
6	Integral Processes	Verification and Validation Software Configuration Management Documentation Development Training

5.1.2 ソフトウェア開発工程の標準化

ソフトウェア開発工程の標準化は主にコンピュータメーカ等を中心に開発され、標準手順やシステム開発方法論として実用化している。これは特に大規模なソフトウェアを効率良く、高い品質で開発する必要から開発されたもので、作業工程、作業内容、作業方式を共通化して分散化／分担化したプロジェクトに適用し、開発者のレベル確保、関与者間のコミュニケーションの円滑化、開発ノウハウの蓄積と再利用を図っている。

ここでの開発プロセス（工程）はソフトウェア開発作業の内容と作業進行の制御を主にしている[井上92]。標準化の内容は開発作業手順、作業内容とワークシートである。また統合型CASEがある場合、開発支援ツールや情報を統合管理をするリポジトリデータモデルが提供される。

代表的なシステム開発方法論として日立のHIPACE、富士通のSDEM90、日本電気のSTEPS II、日本IBMのADSGがある。この他James Martinの開発したIE、米国MBA社のPRIDE、アンダーセンコンサルティング社のMethod/1がある[CASE91]。これらはソフトウェアの開発工程をモデル化しており、ウォーターフォールモデルやスパイラルモデル等を採用している。最近ではオブジェクト指向モデルを用いたGE社のOMTがある。

日立のHIPACEでは計画・分析工程での情報システム計画技法HIPLANで設定した情報システム全体計画と個別システム開発計画に基づいてシステム設計、開発、運用、管理の作業を標準化している。基本的にはウォーターフォールモデルによるフェーズドアプローチを採用しているがダウンサイジングによるCSS（Client Server System）開発用としてスパイラルモデルの適用も可能にしている。図5.1-2はCSS用HIPACE作業手順の流れである。ハードウェアとソフトウェアの構成設定とGUI（画面レイアウト/遷移）の仕様を信頼性と性能等の条件などから決めて行く。

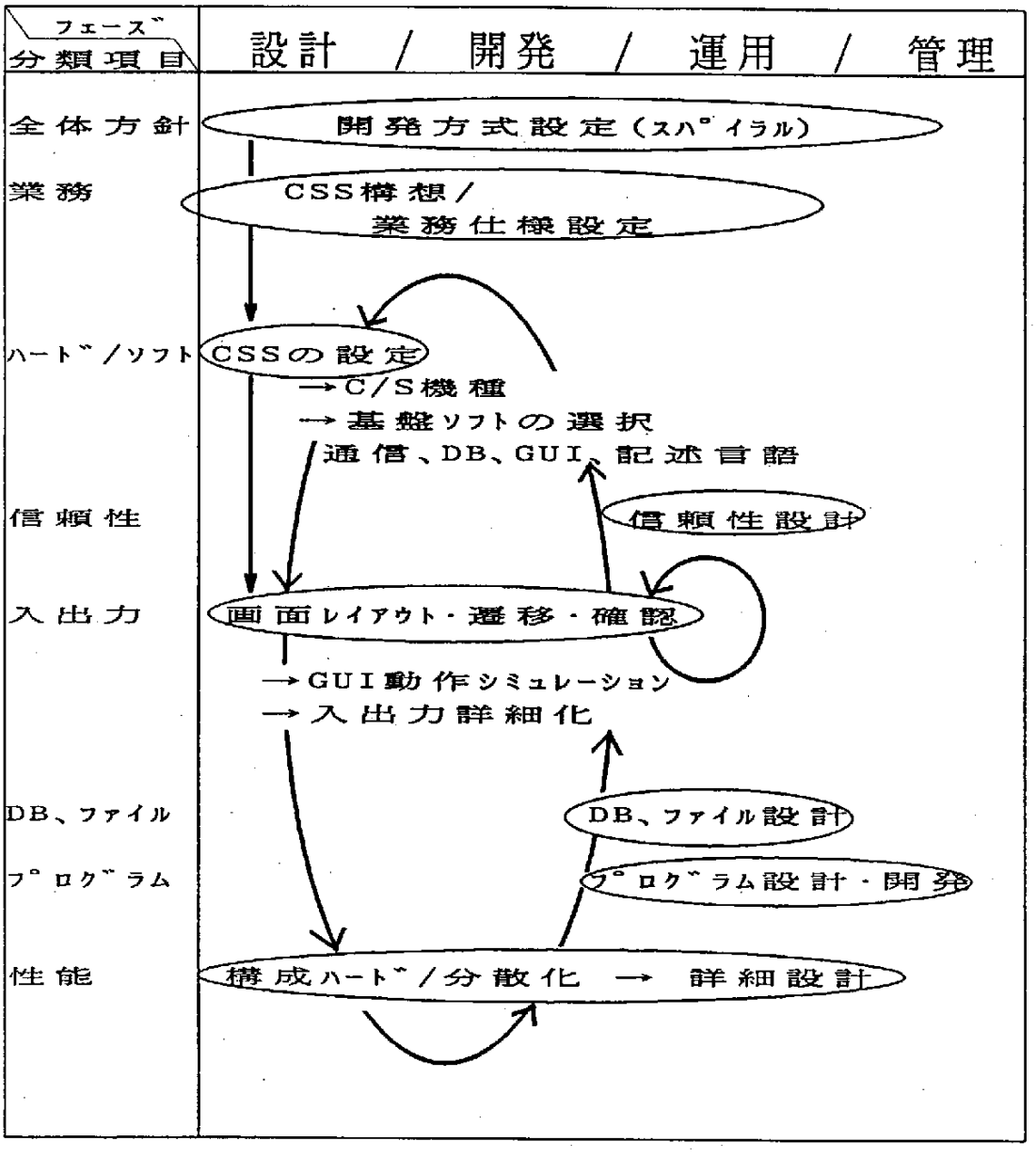


図5.1-2　CSS用HIPACEの流れ

SDEM90 作業体系

カテゴリ ＼ フェーズ		System Planning SP	System Analysis SA	User Interface design UI	System Structure design SS	Program Structure design PS	Programing PG	Program Test PT	Integration Test IT	System Test ST	Operational Test OT	Maintenance and system Evaluation ME
A	業務	全社業務構想立案	業務方式設計	業務の仕様化、変更管理、調整		[UT～IT]		システム運用の準備	[PT～ST]		仮運用と移行の決定	達成目標の評価と業務領域内の監視
B システム詳細	システム機能	情報システムの体系化	機能概要設計と実現性検討	機能の仕様化	機能の作り込みレビュー、変更管理		[SS～IT]	ユーザマニュアル作成	システムテスト仕様書作成 [SS～IT]	システム機能テスト	機能性保証	適応強化
	データ構造		データ概要設計	データの仕様化	データ仕様の実現度合レビュー、変更管理		[SS～IT]					データ資源監視
	性能		性能容量概算	性能容量詳細見積り	性能処理方式の設計	性能の作り込みレビュー、変更管理		[SS～IT]	性能テスト仕様書作成 [SS～IT]	性能テスト・チューニング	性能保証	システム性能の監視
	信頼性・セキュリティ		セキュリティ・障害対策方法の設定	セキュリティ・障害対策方法の仕様化	リカバリ・リスタート方式設計	信頼性の作り込みレビュー、変更管理		[SS～IT]		信頼性テスト	信頼性保証	障害分析・セキュリティ監視
	運用・保守	インテグレーション配置計画	システム運用・保守方法の設定	システム運用・保守性の仕様化	運用処理方式の設計	運用条件の実現度合・保守性の作り込みレビュー、変更管理	[SS～IT]	運用マニュアル作成	運用管理システム機能テスト	運用テスト仕様書作成 [IT～ST]	運用性保証	
	移行		移行方法の設定	移行の仕様化	移行処理方式の設計	移行仕様の実現度合レビュー、変更管理	[SS～PT]	移行マニュアル作成 [PS～PT]	移行アプリ機能テスト [PT～IT]	実データ移行リハーサル	移行	
C ソフトウェア	アプリケーション	パッケージの調達	アプリケーションの立案	アプリケーションの運用検討	アプリケーション構造設計	プログラム設計、統合テスト仕様書作成 [PS～PT]	モジュールの設計、コーディング、テスト	プログラムテスト	プログラム機能テスト			アプリケーションの改造
	データ				内部データ設計			統合テストマスタファイル作成	システムテストマスタファイル作成	運用テスト移行ファイル作成		
	基本ソフトウェア		基本ソフトウェア概要選定	基本ソフトウェア構成の決定	開発基本ソフトウェアの導入		運用基本ソフトウェアの導入 [PG～PT]					基本ソフトウェアの保守・バージョンアップ
	環境				モジュールテスト環境作成	プログラムテスト環境作成 [PS～PG]		統合テスト環境作成	システムテスト環境作成	運用テスト環境作成		環境定義体の保守
D ハードウェア	ホスト		ホスト機種構成の概要選定	ホスト機種構成の決定	開発ホストの導入	開発ホストの運営 [PS～IT]		運用ホストの導入 [PG～PT]	運用ホストの運営 [IT～ST]			ハードウェアの保守・グレードアップ
	ワークステーション		ワークステーション機種構成の概要選定	ワークステーション機種構成の決定	開発ワークステーションの導入	開発ワークステーションの運営 [PS～IT]		運用ワークステーションの導入 [PG～PT]	運用ワークステーションの運営 [IT～ST]			
	ネットワーク		ネットワーク構成方式の設定	ネットワーク構成方式の決定	開発ネットワークの導入	開発ネットワークの運営 [PS～IT]		運用ネットワークの導入 [PG～PT]	運用ネットワークの運営 [IT～ST]			
	設備		施設構成方式の設定	施設構成方式の決定	開発施設の導入	開発施設の運営 [PS～IT]		運用施設の導入 [PG～PT]	運用施設の運営 [IT～ST]			
E 開発支援	開発標準	開発標準の具体化 [SP～SA]	開発標準の具体化 [UI]	開発標準の具体化 [SS]	開発標準の具体化 [PS～PT]			開発標準の具体化	[IT～ST] [PT～IT]	開発標準の具体化	[UT～ST] [ST～OT]	
	技法・ツール	情報技術の調査	技法・ツールの決定	支援ツールの開発・導入	[UI～SS]			性能テストツールの開発	[PT～IT]			
	教育		教育計画立案とUIのための教育	SSのための教育	PS～PTのための教育			ITのための教育	STのための教育	UTのための教育		
	システム資源管理		システム資源管理方法の決定	システム資源管理	[UI～SS]	保守マニュアル作成、システム資源管理	システム資源管理	[PG～ST]				
	開発用品		開発用品計画立案と管理方法の決定	開発用品の手配	開発用品計画の見直し	開発用品管理と見直し	[PS～ST]					
F プロジェクト管理	プロジェクト指標管理		プロジェクト指標値の決定	プロジェクト指標管理と見直し	[UI～SS]	プロジェクト指標管理	[PS～OT]					プロジェクトの評価
	組織・要員	プロジェクト分析の組織化	開発プロジェクト体制と運営ルールの確立	組織管理と見直し	組織管理と次工程の体制見直し、決定		[SS～OT]					
	費用			開発費用管理方法の決定	開発費用の予実績管理と見直し	開発費用の予実績管理	[PS～OT]					

図5.1-3　SDEM90の作業体系

富士通のSDEM90は開発作業をカテゴリ別（空間別）工程別（時間別）に分類している（図5.1-3）。カテゴリとはISOで定義したような視点別の区分で実世界である業務(A)、実世界とコンピュータ世界とのインタフェースであるシステム仕様(B)、コンピュータ世界を構成するソフトウェア(C)とハードウェア(D)、開発資源を補給する開発支援(E)とプロジェクト管理(F)の６種類がある。SDEM90はこのカテゴリ（６種類）と工程（１１種類）のマトリックスで作業（アクティビティ）を決めている。このカテゴリ別に役割が与えられる。例えばA業務ではエンドユーザかアプリケーションエンジニアの役割で、全社業務構想立案、業務方式設計から目標達成の評価と業務動向の監視まで時系列に作業が割当てられる。このマトリックスの作業単位で契約出来るので契約の視点での標準化も実現する。またSDEM90はCソフトウェアのカテゴリに業種パッケージの利用を前提とした作業を設定している[村上92]。

5.1.3 開発プロセスの制御と管理

ソフトウェア開発のプロセスを明確にしてそれを記述し、制御と管理に応用しようとする研究が盛んに行なわれている。Osterweilはプロセスもプログラムとして記述できるという主張からプロセスプログラミングの概念を1987年に出している。Bill Curtis他はACMの論文[Curtis92]でプロセスモデルのコンセプト、目的とゴールを述べ、プロセスプログラミングの手法を紹介している。これによるとプロセス記述の目的は開発プロセスを明確にして、評価と再利用のサイクルを確立する、プロセスを制御して開発作業を誘導する、プロセスの動的把握によりプロジェクト管理の精度を上げる等である。プロセスプログラミングの手法としては手続き型言語（APPL/A, Marvel）、関数型言語（AP5, HFSP）を用いたものがある。またプロセスモデルの記述法として状態遷移図（Statemate）や拡張ペトリネット図（Funsoft）を用いたものがある。また国内では関数型言語によるPDL（Process Description Language）がある。PDLは大阪大学が開発したプロセス記述言語で、プロセスを段階的な抽象度で記述したり、具体的な作業に対しては支援ツールの指定と起動条件が設定できる。PDLでプロセスを記述する場合、まず抽象度の高いレベルでプロセスを記述する。この記述をモデルと呼び特にその記法は限定していない。つぎにこのモデルをPDLに変換して記述する。変換方式はモデルの記法毎に異なるが機械的に決まる。例えば状態遷移モデルからPDLへの変換等である。最後に支援ツールやプロダクト等の情報を付加する。またUNIX系ワークステーションでプロセスをシミュレーション実行できる[井上93]。

開発プロセスの管理ではプロセスの状況表示、プロジェクト管理データの自動収集等があり、プロジェクト実績情報の見積やプロセス改善への反映が考えられる。大阪大学のPDLをベースにした協調開発支援・管理システムとして「はこにわ」がある。「はこにわ」はグループワークを支援する目的で開発され、プロセス間で相互通信する並行プロセスモデルにより現行プロセスの状況表示と作業履歴の収集と表示を行なう。またメニューによる作業誘導と支援ツールの起動を実現している[飯田92]。

プロセスの制御に関し、プロセスプログラミングが比較的詳細、厳密な記述によるのに対し、開発作業をもう少し大きい単位で定義する考え方があり、ワークフロー管理と呼んでいる。ワークフロー管理は主に作業誘導の機能を中心に実用化されている。日立のCAPS DF、IBMのAD/Cycle、Atherton Technology社のATISでは製品化されている。

【参考文献】

[村上92]村上憲稔: ソフトウェアのライフサイクル管理, 情報処理, Vol.33, No.8, pp.912-921(1992).

[IEEE92]Standard for Developing Software Life Cycle Processes, IEEE(1992).

[CASE91]原田実他: CASEのすべて, オーム社(1991).

[Curtis92]Curtis,Bill,et al.: Process Modeling, Communication of the ACM, Vol.35, No.9, pp.75-90(1992).

[井上93]井上克郎: 最近のソフトウェアプロセスの研究動向, ソフトウェア・ツール・シンポジウム'93予稿集(1993).

[飯田92]飯田元他: 協調開発におけるプロセスモニタリング／管理システムの設計と試作, 電子情報通信学会SS91-38, pp.31-37(1992).

5.2 開発プロセスとグループワーク支援

本節では、ソフトウェア開発プロセスをCoordination Theory[Malone92]の観点から捕らえ、ソフトウェア開発におけるグループワーク支援について考察する。

第5.2.1項では、コーディネーション理論を紹介する。コーディネーション理論は、以下の4レイヤからなる協調プロセスで構成される。

(1) 調整(Coordination)

(2) グループ意思決定(Group decision-making)

(3) コミュニケーション(Communication)

(4) 共通オブジェクト認識(Perception of common objects)

第5.2.2項では、ソフトウェア開発プロセスとコーディネーション理論がどう適用できるか、考察する。

5.2.1 コーディネーション理論

コーディネーション（調整）とは、人間やシステムが物事をうまく進めていく時に、行っているプロセスであると、広く捕らえることができる[Malone92]。

"Coordination is the act of working together."

例えば、分散したサイトでソフトウェアを開発したり、また、自動車工場の自動ラインで生産がスムーズに行われている場合など、コーディネーションがうまく発揮されていると考えられる。

逆に、調整がうまく働いていないと、種々の不都合が生ずる。例えば、座席予約システムで二重に指定券を発行したり、信号管制システムの故障で引き起こされる交通渋滞など影響が大きい。

Maloneらは、アクティビティをどう調整するか、アクタ（行為者）がどう仕事をうまくすすめて行くかを示す理論としてコーディネーション理論を提唱した[Malone92]。

人々が物事をうまく運んで行くということは、組織理論、社会心理学、法律、政治科学などの分野で共通である。計算機科学では、第一の対象が人間ではないが、計算プロセスが一緒に働いていると考えればコーディネーション理論の対象となりうる。

コーディネーション理論を考えるにあたっては、Coordinationとはなにかをまず考える必要がある。コーディネーションの問題としは、資源割り当て(Resource allocation)問題をあげることができる。コンピュータネットワークにおけるタスク割り当ても調整問題の例である。

Maloneらが提唱するコーディネーション理論は、以下の

- 調整(Coordination)
- グループ意思決定(Group decision-making)
- コミュニケーション(Communication)
- 共通オブジェクト認識(Perception of common objects)

の4レイヤからなる協調プロセスで構成される。まず、上位層から順に解説する。

(1) 調整(Coordination)

ものごとをうまく進めるにあたって、調整を行う最上位のレイヤである。

Coordinationの構成要素としては、目標、アクティビティ、アクタ、これらの相互依存関係である。表5.2.-1に、これらの構成要素と関連するCoordination Processを示す。

表5.2-1 Coordinationの構成要素

Coordinationの要素	関連するCoordination Process
目標	目標の同定 （目標選択）
アクティビティ	目標のアクティビティへの対応づけ （サブゴールへの分割）
アクタ	アクティビティのアクタへの対応づけ （タスク割り当て）
相互依存関係	相互依存関係の管理 （資源割り当て、順序づけ、同期）

(2) グループ意思決定(Group decision-making)

グループで仕事を進めて行く場合に必要となる機能で、

- 代替案の提案
- 代替案の評価
- 代替案の選択

のプロセスから成る。

(3) コミュニケーション(Communication)

コーディネーション、グループ意思決定プロセスを支える機能で、

- 共通言語の確立
- 受信者の選択（ルーティング）
- メッセージの転送

などから成る。

(4) 共通オブジェクト認識(Perception of common objects)

コミュニケーションを行うためのベースであり、

- 同一物理的オブジェクトの参照
- 共有データベースの参照

などの機能を含む。

このようなコーディネーションについて、Maloneらはコーディネーション理論として、いくつかの分野に対して適用を試みている。適用分野は、計算機科学、経済、組織理論である。適用例を、表5.2-2に示す。

表5.2-2　いろいろな分野におけるCoordination Processの例

レベル	プロセス	計算機科学	経済&ＯＲ	組織理論
調整	ゴール分割	モジュラー化（ＡＩの計画）	経済の規模とスコープ	戦略的計画
	資源割当	プロセッサ・メモリ割当	市場解析	異なる組織の解析
	順序、同期	データフローペトリネットの同期	ＰＥＲＴ	
グループ意思決定	提案、評価代替案選択	ＡＩの探索	決定理論	行動決定理論
コミュニケーション	共通言語	通信標準化	標準化	言語、文化の共有の研究
	ルーティング	メッセージパッシングデータフロー解析、ネットワークネームサーバ	情報価値の解析	コミュニケーションパターン
共通オブジェクト認識	共有情報の参照、操作	共通変数共有知識、ＤＢ	共通知識の理論	状況行動理論

5.2.2　コーディネーションプロセスとソフトウェア開発

本項では、ソフトウェア開発にターゲットを当て、コーディネーションプロセスについて考察する。

(1)　調整（Coordination）

ソフトウェア開発では、Coordination機能は、ソフトウェア開発をうまく進めるためのプロジェクト管理機能に相当する。

表5.2-3に、ソフトウェア開発におけるCoordinationの構成要素（目標、アクティビティ、アクタ、これらの相互依存関係）を示す。

ソフトウェア開発における協調作業支援について、ワークフロー管理の観点から、第5.6節でさらに考察する。

表5.2-3　ソフトウェア開発におけるCoordinationの構成要素

Coordinationの要素	ソフトウェア開発におけるCoordination Process
目標（生産物）	ソフトウェア仕様の決定
アクティビティ（設計・開発作業）	サブシステムへの分割
アクタ（開発担当者）	サブシステムの開発担当の割り当て
相互依存関係	プロジェクト進捗管理、資源管理

(2)　グループ意思決定（Group decision-making）

グループでソフトウェア開発を進めて行く場合、各開発担当者間で、レビュー工程で、代替案の提案、評価、選択のプロセスが行われる。設計根拠を記録・管理するDesign Rationaleが今後、ソフトウェア開発においても使われると考えられる。

ソフトウェア開発における意思決定支援については、第5.5節で考察する。

(3)　コミュニケーション（Communication）

ソフトウェア開発におけるコーディネーションプロセス、グループ意思決定プロセスを支える機能である。コミュニケーションプロセスとソフトウェア開発での例を表5.2-4に示す。

ソフトウェア開発におけるコミュニケーションについては、第5.4節で詳細に考察する。

表5.2-4　コミュニケーションプロセスとソフトウェア開発

コミュニケーションプロセス	ソフトウェア開発での例
共通言語の確立	開発担当者間での問合せ票や問題処理票など、予め取り決めたフォーマットでやりとりを行う。
受信者の選択（ルーティング）	どの開発担当者に連絡票などの帳票類を送るべきか、システム側に登録しておくことで、自動転送する。
メッセージの転送	開発担当者間の連絡は、通常、電子メールで行う。

(4) 共通オブジェクト認識（Perception of common objects）

コミュニケーションを円滑に行うためのベースを与える機能である。ソフトウェア開発においては、設計書などのフォーマルドキュメント、設計根拠などのセミフォーマルな情報、設計担当者などのアクタなどが構成要素となる。これらは、ネットワークを介して、分散したソフトウェア開発拠点から参照されることが必要であり、分散リポジトリとして、今後実現されると考えられる。

第5.3節で、リポジトリにおけるオブジェクト管理、分散リポジトリについて詳細に考察する。

最後に、協調支援ツールとサポートするCoordination Processを表5.2-5に示す。

表5.2-5 協調支援ツールとサポートするCoordination Process

レベル	プロセス	システム例
調 整	ゴール分割	Polymer[Croft88]
	タスク割当	Coordinator[Winograd86]
	資源割当	Information Lens(sorting rules) [Malone87]
	同期	会議スケジューリングツール[Beard90]
	順序づけ	Polymer[Croft88]
グループ意思決定	代替案提案	gIBIS[Conklin88], Sibyl[Lee90] 電子的ブレーンストーミングツール[Dennis88][Stefik87]
	代替案評価	gIBIS[Conklin88], Sibyl[Lee90]
コミュニケーション	メッセージ転送	電子メール
	ルーティング	Information Lens("Anyone"rules) [Malone89] Computer conferencing[Lotus89]
共通オブジェクト認識	共通情報の参照・操作	Computer conferencing[Lotus89] Collaborative authoring tools[Ellis90] 電子会議室[Dennis88][DeSanctis87][Stefik87]

【参考文献】

[Malone92]Malone,Thomas,et al.: Towards an Interdisciplinary Theory of Coordination,MIT Technical Paper (1992.1.14).

[Croft88]Croft.,W.B.,and Lefkowitz,L.S.: Using a planner to support office work. In R.B.Allen(Ed),Proceedings of the ACM Conference on Office Information Systems,Palo Alto,CA, pp.55-62(1988).

[Winograd086]Winograd,T,and Flores,F.: Understanding computers and cognition: A new foundation for design,Norwood,NJ:Ablex(1986).

[Malone87]Malone,T.W,et al.: Semistructured messages are surprisingly useful for computer-supported coordination,ACM Tranzactions on Office Systems,Vol.5,No.2(1987.4).

[Beard90]Beard,D.,et al.: A visual Calender for Scheduling group meetings. In D.Tatar(Ed.),Proceedings of the third conference on computer-supported cooperative work,Los Angels,CA:ACM Press,pp.279-290 (1990).

[Conklin88]Conklin,J., and Begeman,M.L.: gIBIS:A Hypertext tooling for exploratory policy discussion. In D. Tatar(Ed.),Proceedings of the Second Conference on Computer-supported Cooperative Work,Portland, OR:ACM Press.,pp.140-152(1988).

[Lee90]Lee,J.: Sibyl:A Qualitative decision management system. In P.Winston(Ed.),Artificial Intelligence at MIT:Expanding Frontiers,Vol.1,Cambridge,MA:MIT Press(1990).

[Dennis88]Dennis,A.R,et al.: Information Technology to Support Electric Meetings,MIS Quarterly,12:4, pp.591-619(1988.12).

[Stefik87]Stefik,M.,et al.: Beyond the chalkboard:Computer support for collabolation and problem solving in meetings, Communication of the ACM,30(1),pp.32-47(1987).

[Lotus89]Lotus: Lotus Notes Users Guide,Cambridge,MA:LotusDevelopment Corp(1987).

[Ellis90]Ellis,C.A,et al.: Design and use of a group editor. In G.Cockton(Ed.),Enginieering for Human-Computer Interaction,pp.13-25(1987).

[DeSnctis88]DeSanctis,G., and Gallupe,B.: A foundation for the study of group decision support systems, Management Science,33(5),pp.589-609(1987).

5.3 共通オブジェクトの参照

コンピュータによる協調作業、とくにソフトウェア開発という協調作業においては、種類や大きさ、抽象度の異なるさまざまな情報が必要となり、開発の進行とともにさまざまな情報が新たに生成される。これらの情報は、開発作業の中で複数の人々によって参照・利用されるので、使用される情報を論理的に一箇所に集め、体系的にまとめて保存するしくみを設ければ、既存の情報の保存・利用にもまた新たに生成される情報の整理・利用にも有効であることは明白である。しかし単に使用される情報の集合体を形成すればよいわけではなく、「共通のオブジェクトとして認識できる集合体」であることが求められる。特にその集合体にアクセスし、情報の入出力を実行するのが人間ではなく、各種のツールである場合には、この「共通の認識」という概念は非常に重要になってくる。

CASE分野において、「リポジトリ」という考え方が出されて久しい。リポジトリは「CASEによる共通オブジェクトの認識」を第一義の存在理由としてあげることができるであろう。リポジトリにおけるオブジェクトの共通認識はいかに実現されているか、リポジトリにおけるオブジェクト管理はどのようになっているか、リポジトリ製品はどのようになっているかをまとめてみた。

5.3.1 リポジトリ

リポジトリを厳密に定義するのはむずかしい。それは、リポジトリの概念が発表されて既に数年が経過するにもかかわらず、常に「製品」主導でリポジトリが議論されてきたからであり、主要コンピュータメーカや団体が発表した特定のリポジトリについて、その機能やサービスに人々の関心が集まっているにすぎないからである。

リポジトリはどうあるべきか、リポジトリは何がどこまでサポートされるべきか、といったリポジトリを正面から取り上げた議論はあまり見られず、ましてリポジトリ仕様の標準化となると全く見通しが立っていないというのが現状であろう。そこで、ここでも市場に出された“主要”リポジトリに共通する基本的概念やリポジトリが扱う情報、リポジトリの概略の仕組み等を見ることにする。

リポジトリとは、その名の通り「情報の貯蔵庫」あるいは「ソフトウェアエンジニアリングデータベース」であるが、これだけではまだリポジトリのイメージとしては十分ではない。ソフトウェア開発には、企業経営に関わる情報からソフト開発のための仕様書にいたる広範な情報が必要である。

図5.3-1は、ソフトウェア開発のためのプロセスモデルである。「開発プロセス」という単一のプロセスだけでなく、それを取り巻いているいくつものプロセスから成っており、ソフトウェア開発はこれらのプロセスが有機的にからみあって完結する。これらのプロセスの中で扱われる情報をみれば、ソフトウェア開発に関係する情報の多様さがわかる。

まず、開発プロセスでは、システム化する業務そのものに関する情報が必要である。企業活動や組織に関する情報、業務を遂行する手順・業務範囲などの情報が要求定義という形で作成される。さら

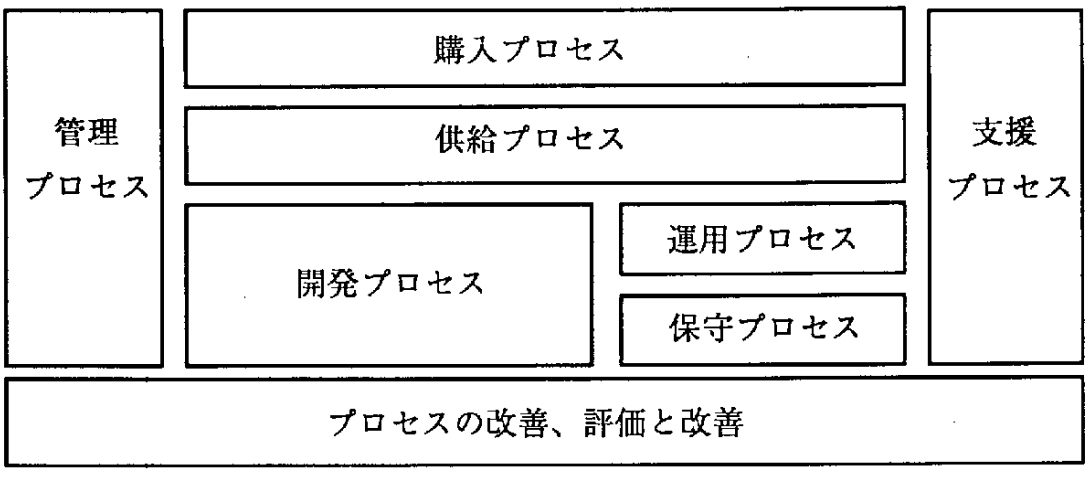


図5.3-1　ISO案：ソフトウェアライフサイクルプロセス

に開発過程で生成される情報がある。開発計画書、システム仕様書、ハードウェア仕様書、ソフトウェア仕様書、ソフトウェア方式仕様書、ソースコード、テスト仕様書等々であり、これらはさらに細分化されたドキュメントから構成されている。

管理プロセスは、いくつかの側面からソフトウェア開発の進行を監視・制御するもので、工程、原価、要員などの管理項目があり、これらが開発の進捗に合わせて変化していく。管理プロセスで生成される情報には、進捗管理表、品質管理報告書、資源管理書などがある。購入プロセス、運用プロセス、保守プロセス、支援プロセスにおいても、同様に抽象度、種類、大きさの異なる膨大な情報が関連する。これらがすべて、リポジトリの管理対象である。

では、リポジトリはデータベースとどうちがうのか、という疑問が湧いてくる。これはリポジトリの定義の問題であり、リポジトリの定義が明確になれば、リポジトリとデータベースが同じものか、異なるものかが明確になるはずである。ただ、リポジトリとデータベース、特に関係データベースとで異なっていると一般に言われている点がある。それは、管理記憶機構の重点をどこに置くかの違いである。データベースでは、情報を記憶したり、記憶してある情報を引き出すときの効率・性能などをよりよくするためにどのように情報を編成するかがもっとも重要である。一方リポジトリも情報を記憶したり、記憶してある情報を引き出して使用するのであるから、その効率・性能は重要ではあるが、それ以上に情報の「意味構造」が非常に重要である。データベースのように単に情報を貯蔵するだけでなく、リポジトリでは各々の情報の意味を明確にし、意味構造を管理記憶することが求められる。

では情報の意味構造とは何か。

情報の意味構造を記述するとは：

- まず情報そのものの性質を記述すること

  そのものの性質とは、属性のことである。その情報に付帯している性質は多数あるが、そのうちソフトウェア開発に関係する属性を記述する。

- さらに他の情報との関連・関係を記述すること

  他の情報とどういう関係にあるかを記述する。

ソフトウェア開発に関係する全ての情報（データ）について、この二つのことを記述することで、その情報がどういう情報であるのかという全体の中での位置づけが明確になる。これは別の見方をするならば、その情報の現実世界での意味と位置を記述していることになる。これが情報の意味構造を記述するということである。

(1) 情報とその情報の性質を記述する

情報の意味を明らかにするには、まず必要な限りその情報の詳細を記述しなければならない。ソフトウェア開発に関わる情報は上で見たように多種多様であり、粒度、種類とも種々雑多である。大きいものではデータベース一つそっくりが一つの情報にもなる。ファイルという情報単位もある。数値一つ一つも情報である。このような情報の性質・属性を記述するとは例えば次のようなものを記述することになる。

- 情報の名前（その情報を表すユニークな名称）
- 情報のタイプ、クラス（その情報がどういうグループに属するのかの指定）
- 情報の作成者、作成日
- 情報の変更記録
- 情報のアクセス権クラス

などなど、ソフトウェア開発で必要となる属性がすべて含まれる。

(2) 他の情報との関連を記述する

情報の意味を記述するためにもう一つ必要なのは、その情報と他の情報との関係である。関係とはある情報と他の情報との関わり方を記述するもので、汎化関係、親子関係、構成関係も関係であるとともに、「参照する」「コンパイルする」「包含する」なども関係を示す言葉であり、ほとんどの“動詞”がその関係を示す言葉となる。このとき、関係には方向性があることが分かる。例えば「参照する」という関係の場合は、ある情報が他の情報を参照するわけで、他の情報はある情報から参照されることになる。つまり関係には起端と終端がある。したがって、関係を記述するということは情報間の関係とその関係の方向性（起端と終端の指定）を記述することである。また、その二つの情報の関係が常に1:1であるとは限らない。1:n、n:1、n:nなどの関係もある。これがcardinalityの記述である。

このように、その情報の性質、その情報と他の情報との関係を記述すれば、その情報の意味すなわちその情報の位置づけが明確になるが、これを統一的に記述する手法が意味データモデルである。意味データモデルとしてはこれまでに

- Entity-Relationship model
- Functional Data Model
- Semantic Data Model

などなどが提案されており、さらに

- オブジェクトモデル

がある。これらは多かれ少なかれ情報の性質と情報と情報の関連を記述するモデルである。すなわち、リポジトリとは意味データモデルに従ってソフトウェアエンジニアリングの情報を体系化し、貯蔵・管理・利用するための情報集合体である。

ここで注意したいのは、情報の本体そのものはリポジトリの中には貯蔵されないということである。リポジトリに貯蔵されるのは、情報の性質および他の情報との関係であり、情報本体は例えばいずれかのデータベースに貯蔵されていればよいわけである。

一方、リポジトリはどのように使用されるかも見ておく必要がある。もともとリポジトリはCASEツールの出現およびその発達過程と深い関わりがある。むしろリポジトリは、ソフトウェア開発におけるCASEツールの弱点を補う救世主として登場してきたといった方がよい。1980年代からソフトウェア開発の生産性を向上させ、ソフトウェアの品質を高めるための手段としてCASEツールがもてはやされたが、ほどなくその弱点を露呈した。上流CASEツールと下流CASEツールの間の大きなギャップである。この大きなギャップを埋めるための解決策としてリポジトリが位置づけられた。この考え方はすぐに拡張され、CASE統合のための主要メカニズムとしてのリポジトリとなった。すなわち、上流・下流に関係なくCASEツールとCASEツールの間で受け渡しされる情報の貯蔵庫であり、CASEはそれ自身に最も適したVIEW（切り口）でリポジトリの中の情報を切り出して加工し（ソフトウェア開発の一工程）、それをまたリポジトリに戻す。次の工程のCASEツールは再びリポジトリの中から情報を切り出して加工するといった過程を繰り返す。このようなCASEツールとリポジトリとのコミュニケーションの集合としてソフトウェア開発がイメージされる。リポジトリはソフトウェア開発に関するすべての情報を蓄積・供給するセンターであり、CASEツールがそのまわりに近づいてはリポジトリとの間にコミュニケーションチャネルを設定し、情報の加工を行ってリポジトリに情報を戻すと、次のCASEツールがトリガされるというイメージである。

これをサポートするリポジトリは、いずれのツールにも依存しない独自の“情報の世界”を用意しており、どのツールに対しても情報を提供できるようになっているわけである。

5.3.2 リポジトリマネジメント

リポジトリは情報の集合体であり、リポジトリの情報をツールから利用できるようにするには、

- まず意味データモデルに従ってリポジトリを生成すること
- 生成したリポジトリを維持管理すること
- 情報にアクセスするためのサービスを提供すること

が求められる。これを行うのがリポジトリマネジメントである。リポジトリマネジメントの機能は情報の格納・検索・チェック・アクセス制御など、通常のデータベース管理システム（ＤＢＭＳ）に求められる機能と同様のものであるが、さらに意味データモデルの構築など、リポジトリ特有の機能も要求される。リポジトリマネジメントの主要機能を見てみる。

(1) リポジトリの論理構造生成機能

リポジトリを構成するためには、まずその論理構造、すなわち意味データモデルに従って構造を定義しなければならない。

この作業を行うには、まずリポジトリに格納する情報を全て調べ上げる。おのおのの情報の属性を列挙し、情報と情報の間の関連を明確にし、その情報がソフトウェア開発でどのような位置付けになるかを十分に分析する。さらにツールはそれら情報をどのように取り出し、どのように加工し、どういう形で次工程のツールに渡すかを明確にする。このような作業の結果明確になった情報の論理構造を、リポジトリに記述していく。すなわち、リポジトリ管理にはまず情報の論理構造を記述する機能が求められる。この場合、情報の属性や情報間の関係については、開発環境のリポジトリとして必要なものはすべて記述できることが必須である。

(2) 具体的な情報（インスタンス）の生成・削除機能

論理構造ができたリポジトリに、具体的な情報（インスタンス）を格納したり、削除したりするリポジトリ管理機能である。この機能は同時に、論理構造に合致しない情報の格納の排除、情報へのアクセス権限のチェックなど情報の生成・削除・更新に伴う機能も含まれる。

(3) 情報の一貫性管理

一貫性管理も、構築されたリポジトリの管理に必須の機能である。ある情報に変更が生じた場合、その情報と関連のある情報はすべて、データモデルの関係を保持するように変更するものである。この機能により、リポジトリ内の情報に自己矛盾を生じさせないようにすることができる。

(4) ツールの組み込み機能

リポジトリはソフトウェア開発のための各種支援ツールと情報の受け渡しを行う。ある工程のツールは、自分のVIEWに従ってリポジトリから情報を取り出し、加工して再びリポジトリに格納する。新しいツールをソフトウェア開発に使用したい場合には、そのツールが“alian”ではなく有効なツールであることをリポジトリに認識させる必要があり、それを行う機能が必要である。さらに新しいツー

ルのVIEWを設定する機能も必要である。

(5) 分散リポジトリサポート機能

ハードウェアの小型化、通信システムの充実などから、物理的に分散したハードウェア上にリポジトリを展開することが十分考えられる。この場合、リポジトリ内の情報にアクセスするツールには物理的詳細を意識させず、論理的には単一のリポジトリとして提供することが、開発環境の柔軟性の面からいって必要である。このような物理的詳細の隠ぺい化がリポジトリマネジメントの機能として求められる。

5.3.3 リポジトリの製品

現在リポジトリという名を使っている製品は50以上あると言われる。ほとんどがCASEベンダが自社のツールのために開発したリポジトリであろう。このうち、よく知られているリポジトリ二つについて、リポジトリマネジメントに必要な機能の実現方法を見てみる。

(1) PCTE（Portable Common Tool Environment）

PCTEは、1983年にヨーロッパの主要産業グループが参画して開始したESPRIT（European Strategic Programme for Research in Information Technology）計画の中の一つのプロジェクトであり、ヨーロッパの主要コンピュータメーカと大学との協力で推進している。PCTEプロジェクトは、アメリカのDoD（国防総省）が1970年代に出したStoneman文書に刺激されて開始したという経緯があり、ソフトウェア開発のための開発環境をつくること、開発環境においてはツール共通インタフェースを提供することが主要な目的である。このため、サービスやメカニズムを提供するためのライブラリのインタフェースが規定されている。例えばC言語バインディングの場合、約150の基本メカニズムと約110のユーザインタフェースの計約260個のファンクションから成っている。このようなファンクションは、データ管理、ツール実行・ツール間コミュニケーション、環境管理、ユーザインタフェース管理の4つのグループに分けることができる。

このうち、データ管理を行うのがOMS（Object Management System）で、環境が使用するデータを定義し、定義したデータに対して入出力・変更等の操作を行う。管理対象となるデータの集合体はObject Baseと呼ばれる。すなわち、Object BaseがPCTEのツールが使用するデータのリポジトリである。Object Baseを構成する意味データモデルはERモデルを拡張したモデルである。

Object Baseの中の各データ（インスタンス）は「オブジェクト」とよばれ、オブジェクト間の関係は双方向の「リンク」のペアで表現される。PCTEにおけるオブジェクトの粒度は、UNIXにおけるファイル、周辺機器、プロセス間メッセージキュー（パイプ）、プロセスなどである。ユーザが定義したデータ、例えばプロジェクトマイルストーン、仕様変更要求などもオブジェクトとなり得るが、これらのデータを扱うためにはユーザがツール（プログラム）を用意しなければならない。

Object BaseはERモデルに従って構成されたデータ集合体であるが、その中心となるのが強い「型ルール」である。Object Baseの構成要素はすべてこの型ルールに従わなければならない。そのため、構成要素であるオブジェクト、リンク、およびオブジェクトとリンクに付随した属性にはすべて何らかの型が定義される。Object BaseのERモデルが他のERモデルと異なるところは、オブジェクトの型に階層構造をもたせており、子のオブジェクト型は、親のオブジェクト型の属性をすべて継承することである。オブジェクト型の階層構造の頂点にあるのが、既に定義されている型“Object”であり、型“Object”には、子の型として“File型”、“Pipe型”、“message queue”、“device”が既に定義されている。ユーザが自分で子の型を定義することもできる。

このような強い型ルールに従って、Object Baseの論理構造を定義するのが、SDS（Schema Definition Set）である。SDSを使って、個々のオブジェクト、リンク、属性それぞれの型の定義や、どの属性がどのオブジェクト型・リンク型に対応するのか、どのリンク型はどのオブジェクト型に対応するかなどの対応関係の定義を行い、Object Baseの論理構造を設定する。Object Baseには、全体の論理構造を定義するスキーマは存在せず、むしろ個々のSDSの集合として全体スキーマがあるといえる。したがって、個々のSDSは互いに整合がとれ自己矛盾をしないものになっている。ツールがObject Baseの中の必要なデータにアクセスする場合には、SDSをいくつかまとめて構成するWorking Schemaを通して行う。SDSの定義、修正、削除などに必要なファンクションも別に用意されている。

OMSには、型ルールに従ってオブジェクトのインスタンスを生成・削除するファンクションも用意されている。ソースコードのようにOMSにはその内容がつかめないファイルをもオブジェクトとして生成することができ、ユーザは粒度の粗いオブジェクトも管理できる。

PCTEにおけるデータの一貫性管理は、リンク属性のなかにそのしかけがある。カテゴリ属性と安定性属性がそれである。例えば二つのオブジェクトの間に安定性リンクがはられた場合には、そのリンクが削除されない限り、オブジェクトは削除できないようになっている。

PCTEでは、プロセスという型のオブジェクトが用意されている。プロセスとは「プログラムの実行」のことで、ツールの起動もこれにあたる。したがって、ツールが関与するプロセスを生成することがツールを起動することにあたる。したがって新しいツールを組み入れるには、そのツールのオブジェクトを生成することであり、そのツールのVIEWは、SDSを定義することである。

PCTEでは、分散リポジトリをサポートしており、OSIのトランスポート層をもとにしてつくられている。すなわち、Object Baseはワークステーションに動的にマウントできる論理的なディスク（Volume）に分かれており、一度マウントが完了すると、Object BaseのオブジェクトはVolumeの境界を超えてリンクすることができる。したがって、どのオブジェクトがどのワークステーション上にあるかといった物理的な配置をユーザは気にする必要はなく、分散環境に関するすべての制御はPCTEがおこなうようになっている。

(2) AD/Cycle（Application Development/Cycle）：IBM

リポジトリという概念をいち早く発表したIBMが、1989年9月に公表した総合的なシステム開発支援環境のフレームワークがAD/Cycleである。この中では、ソフトウェア開発を行うときのトータル戦略、プロセスモデル、ワーク管理、データモデル、ユーザインタフェース、知識ベースによる開発支援、開発支援ツールなどと並んで、リポジトリ管理システム（リポジトリマネジャ）が大きな位置を占めている。

リポジトリマネジャは大きく分けると、ユーザに対して二つの機能を提供している。一つは、「データ指定サービス」で、リポジトリ内に保存し利用するデータはどのようなデータなのかをユーザ自身に定義させるための機能である。二つ目は、「実行時サービス」で、リポジトリ内に保存されているデータを実際に利用する場合に必要なデータ入出力を行う機能である。

① データ指定サービス

ユーザが行うソフトウェア開発には、上で述べたように非常に広範なデータが関係する。企業経営に関わるデータから、ソフトウェア開発の過程で生ずる仕様書の類までさまざまである。そのように多種で多量のデータをリポジトリという統一的な枠組みの中に体系化するために、それらデータはどのようなデータであるのか、どのようにそのデータを使いたいのかを、AD/Cycleというシステムに対して指定してやらなければならない。この指定は、ユーザがリポジトリマネジャの画面と対話しながら行う。データは一般に多様な側面をもっているので、データを定義するといってもそれほど簡単ではない。そこでAD/Cycleでは、三つの側面からデータを定義させている。それは、概念視点、論理視点、記憶視点の三つである。

概念視点とは、上の言葉で言えば意味データモデルを構築することである。AD/Cycleでは、意味データモデルとして、ERモデルとその拡張概念を採用している。したがって概念視点では、保存したいデータの属性およびデータ間の関係を定義する。さらにAD/Cycleでは、ERモデルの拡張概念として「オブジェクト」なるものを導入している。AD/Cycleのオブジェクトとは、データの集合体のことである。概念視点の主要部分は、このように意味データモデルを構築することであるが、AD/Cycleは総合的システム開発支援環境であるので、データを定義すると同時に他にも定義しておかねばならないものがある。それはデータの安全性に関する種々の制約条件であり、これをAD/Cycleでは「ポリシー」と呼んでいる。概念視点で定義するポリシーは、アクセス権限の指定（安全保護ポリシー）、データ値の許容範囲（保全ポリシー）、データ作成する方式の指定（指定方式ポリシー）、データ状態の変化に基づく処置（トリガーポリシー）の四つである。これらのポリシー（制約条件）は、リポジトリのデータに対して外部から読み出し・書き込み要求があったときに、その正当性をチェックするために使用される。

論理視点とは、AD/Cycleに組み込むツールの機能仕様を指定するための視点である。ツールはリポ

ジトリの中に保存されている全てのデータが必要なわけではなく、そのツール固有の処理に合ったデータが必要なわけである。そこでリポジトリの中のデータとツールとを結び付け、どのデータにはどのツールがどのようにアクセスするのかを指定するのが、論理視点である。指定作業は、ユーザがリポジトリ管理システムの提供する「テンプレート」という画面を介して行う。テンプレートで指定するのは、必要とするデータの属性、ツール間でやり取りするパラメータ、ツールがユーザとの対話に使用するパネルの記述、ツールの格納場所等である。テンプレートは、同じようなデータに同じようなアクセスの仕方をするツール間で共用することもでき、ツールアクセスの効率を挙げることができる。

ツールに対しても、概念視点と同じようにポリシー（制約条件）を付加することができる。ポリシーの種類は、概念視点でのポリシーと同じである。

記憶視点は、実際のリポジトリを構成するときの物理的な記憶装置を指定するものである。AD/Cycleでは、関係データベースをリポジトリの物理的な装置として使用しているので、一度概念モデルが構築されると、それを関係データベースにマッピングするだけであり、この機能はリポジトリ管理システムの機能である。

② 実行時サービス

実行時サービスは、リポジトリを利用するのに必要なサービスを提供するものである。実行時サービスには三つのサービスがある。データサービス、ユーザサービス、システムサービスである。データサービスは、リポジトリの中に保存してあるデータにツールを介してアクセスし、データそれ自身、データの属性、データ間の関係に対して読みだし・書き込みを行うものである。ユーザサービスは、論理視点のテンプレートに記述した対話用パネルを実際の対話パネルに自動的に変換してくれるものである。この場合、メッセージ管理やログサービスなど、人間との対話に必要な情報も付加することができる。システムサービスは、リポジトリ機能部品のオープン／クローズを行うほか、リポジトリを全体として管理するための情報（リポジトリのバージョン、トレース機能、時間サービス等々）を行う。

【参考文献】

[村上92]村上憲稔: ソフトウェアのライフサイクル管理,情報処理, Vol.33, No.8, pp.912-921(1992).

[ECMA90]ECMA: Portable Common Tool Environment, Standartd ECMA-149(1990).

[Thomas89]Thomas,I.: PCTE Interfaces:Supporting Tools in Software-Engineering Environment, IEEE Software, IEEE, pp.15-23(1989).

[Sagawa90]Sagawa,J.M.: Repository Manager Technology, IBM System Journal, IBM, pp.209-227(1990).

[IBM90]リポジトリ管理プログラム／ＭＶＳ概説書, N:GC26-4608, IBM(1990).

[Paine93]Paine,K.: The Repository:What Now?, CASE TRENDS, Software Productivity Group, Inc., Vol.4, No.9, pp.53-55(1993).

[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報告, 80-5(1991).

[片山90]片山卓也: ソフトウェア開発のためのオブジェクトベース, 情報処理学会研究報告,7 9-DBS-8-3 (1990).

[徳田90]徳田雄洋: 次世代データベースシステム宣言に向けてーソフトウェア・オブジェクトベース構築の立場からーオブジェクトベースはソフトウェア開発過程を強く支援し得るか?, 情報処理学会研究報告, 79-DBS-8-5(1990).

[Hull87]Hull,R.et.al.: Semantic Database Modeling:Survey, Applications, and Research Issues, ACM Computer Surveys, Vol.19, No.3(1987).

5.4 コミュニケーション支援

本節では、コーディネーション理論のコミュニケーション層について、ソフトウェア開発に適用した場合について考察する。

ソフトウェア開発におけるコーディネーションプロセス、グループ意思決定プロセスを支える機能で、

- 電子メールによる開発担当者間の連絡
- 開発担当者間での問合せ票や問題処理票など、予め取り決めたフォーマットでやりとりの支援
- どの開発担当者に連絡票などの帳票類を送るべきか、システム側が自動転送する機能

について、Information Lens[Malone87][Lai88]を例に考察する。

5.4.1 協調支援レベルとコミュニケーション支援

ソフトウェア開発におけるコーディネーションプロセス、グループ意思決定プロセスを支える機能で、下位のPerception of Common Objects層を使う。

すなわち、コミュニケーション層の機能は、ソフトウェア開発におけるプロジェクト管理機能、グループによるソフトウェア設計上の代替案の提案／評価／選択を行う上で必要となるコミュニケーションを支援する。ここで、分散環境下で共有する各種オブジェクトの論理的な配置については、コミュニケーション層の下位に位置するPerception of Common Objects層の機能を利用することになる。

(1) 共通言語の確立

分散環境でソフトウェア開発を行う場合、開発担当者間では、予め取り決めたフォーマットの問合せ票や問題処理票をやりとりして仕事を進めることになる。そのための共通のフォーマットが共通言語ということになる。お互いに共通の言葉がわかっていないと、うまくコミュニケーションが確立できない。

(2) メッセージの転送

開発担当者間の連絡は、通常、電子メールで行われる。電子メールには、それ自体、送信先、発信先のネットワーク上で認識可能なアドレスメールがつけられているので、ネットワーク上でメッセージの転送が可能となっている。

なお、どのサイトにどのユーザが存在するのかといった、ネットワークディレクトリの機能は、コミュニケーション層の下位層であるPerception of Common Objects層の機能である。

(3) 受信者の選択

ある情報を送ろうとした場合、正しい受け取り手がネットワークのどこに存在するか、一般にはわからない。どの開発担当者に連絡票などの帳票類を送るべきか、システム側に登録しておくことで、自動転送することができる。

5.4.2 コミュニケーション支援の具体例

本項では、コミュニケーション層の機能の実現例として、Information Lens[Malone87]、Object Lens [Lai88]を紹介する。

(1) 半構造化電子メール

Information Lensが提供するコミュニケーション支援機能について、Coordination Processの観点から紹介する。

半構造化電子メールとは、さまざまなタイプを識別できるメッセージであって、電子メールにさまざまなフィールドを追加したものである。

- コンピュータによる自動的な処理が可能
- 構造が厳密でないので、自由度がある
- 徐々に拡張してゆくことが可能

など、メリットがある。半構造化電子メールは、

- メッセージの送信者はメッセージを作成する際、テンプレートを使うことにより、メニュー選択形式で情報を選択可能である
- メールの受信者は、送信者が構造化して送って来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格納するようなルールを指定することができる
- 送信者は、通常の宛て先（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（"Anyone"）に送ることができこのメッセージに関心をもつと思われる人に自動的に送信される
- 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、受け取れるようにルールを選択できる

などの機能がある。

(2) 共通オブジェクト認識

Information Lens[Malone87]を拡張して開発されたObject Lens[Lai88]では、コンピュータによる自動処理を支援するため、分散環境下で、オブジェクトをモデル化している。このオブジェクトは、ちょうどコーディネーション理論における共通オブジェクト認識層の機能に相当すると考えられる。

Object Lensでは、半構造化オブジェクト、半自律エージェントの二つの概念が導入されている。

① 半構造化オブジェクト

Object Lensが扱うオブジェクトは、フィールドとフィールド値の集合から成る。例えば、PERSONのオブジェクトは、NAME、Phone、Address、JobTitleなどのフィールドを持つ。TASKのオブジェクトは、Requestor、Performer、Description、Deadlineなどのフィールドを持つ。これらは、Information Lensと同様に、テンプレートを使って、任意のフィールトにデータを入力することができる。フィールド内の情報は、自由形式であるという意味で、半構造化ということができる。

② 半自律的エージェント

Object Lensでは、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ユーザが作成することができる。エージェントは、メールの到着、指定したフォルダ内の新しいオブジェクトの作成などのイベントによって駆動され、指定したルールが実行される。半自律的というのは、適用されるルールをユーザが作成／変更ができるという意味である。

Object Lensでは、このように、共通オブジェクトを規定することにより、コミュニケーション層以上でのCoordination Processの機能の遂行を支援することができる。

【参考文献】

[Malone87]Malone,T.W.,et al: Semistructured messages are surprisingly useful for computer-supported coordination, ACM Transactions on OfficeSystems,Vol.5,No.2(1987.4).

[Lai88]Lai,K.,and Malone,T.W.: Object Lens:A "Spreadsheet" for Cooperative Word,CSCW'88 Proceedings, pp.115-124(1988).

5.5 意思決定支援

5.5.1 ソフトウェア開発における意思決定

ソフトウェアの開発プロセスは、個人、あるいは、グループによる絶えざる意思決定のプロセスとも言える。このような意思決定プロセスの問題点は、図5.5-1に示すような視点から見ることができよう。

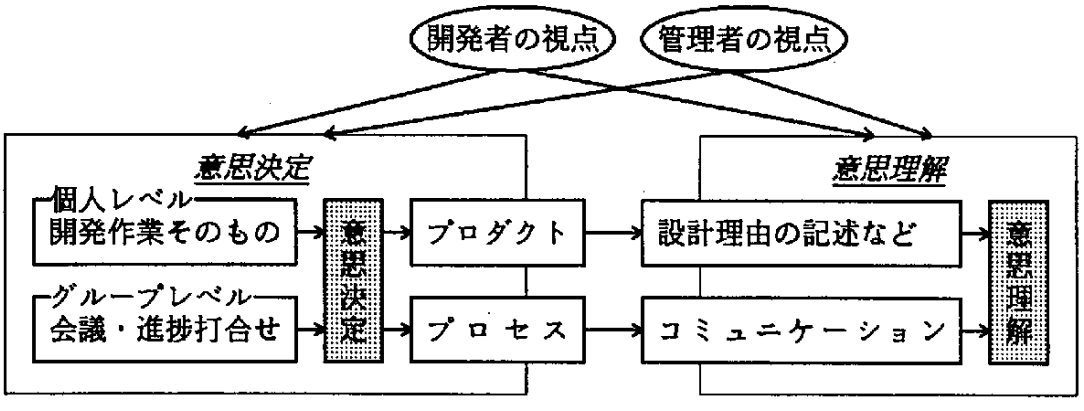


図5.5-1 ソフトウェア開発における意思決定

(1) 開発者の視点

ソフトウェア開発者の視点では、要求分析や設計など個々のプロセスにおける意思決定である。例えば、図5.5-2に示すように、設計工程における最終目標は、いくつかの設計案の中から最適な設計案を決定することである。一方、機能追加や保守プロセスでは、既に決定された設計者の意思（意図）やその決定に至る理由を知る必要がある。しかし、このような意思決定のプロセスや決定の理由は設計ドキュメントなどには陽に記述されることは稀であったし、ソフトウェア工学の中でもこれまであまり議論されなかった。しかし、実務においては、このような設計の決定事項とその理由や検討経過は「検討書」として「インフォーマル」な形ではあるが、重要なドキュメントとして保存されていた。最近、このようなソフトウェア開発における意思決定の理由、いわゆる「設計理由(Design Rationale)」の記述が重要であるとの認識が高まっている[垂水92]。

(2) プロジェクト管理者の視点

ソフトウェア開発プロジェクトの進行は、時々刻々変化する「リアクティブシステム」あるいは「リアルタイムシステム」であると考えられる。したがって、プロジェクトの管理も、タイムリな意思決定が重要である。

また、多くのリアクティブシステムと同様、定常的なプロセスと例外的なプロセスがある。例えば、

日常の進捗管理や品質管理では、プロジェクトで決められた作業標準や標準開発プロセスに従って進められる意思決定のプロセスとも言える。一方、あるグループにおける著しい進捗の遅れの発生や顧客における障害の発生に対する対処は、問題の状況に応じた意思決定が求められ、現状では、管理者の資質や経験に依存している。

また、近年普及している「分散開発」では、分散した開発拠点が自律的に行動する傾向があるため、プロジェクト全体の迅速な意思決定や問題の対策が困難となる[青山92]。

一方、グループウェアの分野では、会議を中心としたグループによる意思決定支援「GDSS(Group Decision Support Systems)」が広く研究、開発されてきた[Kraemer88]。しかし、これらの支援ツールの多くはビジネス一般における意思決定支援を主眼としており、ソフトウェア開発などの特定の分野に固有の問題を支援しているものではない、いわば「汎用会議システム」である。このような背景から、マルチメディア在席会議システムをソフトウェア開発における会議へ適用する事例も見られる[阪田92]。しかし、これらの会議システムがソフトウェア開発における意思決定プロセスを広く支援するまでには至っていない。

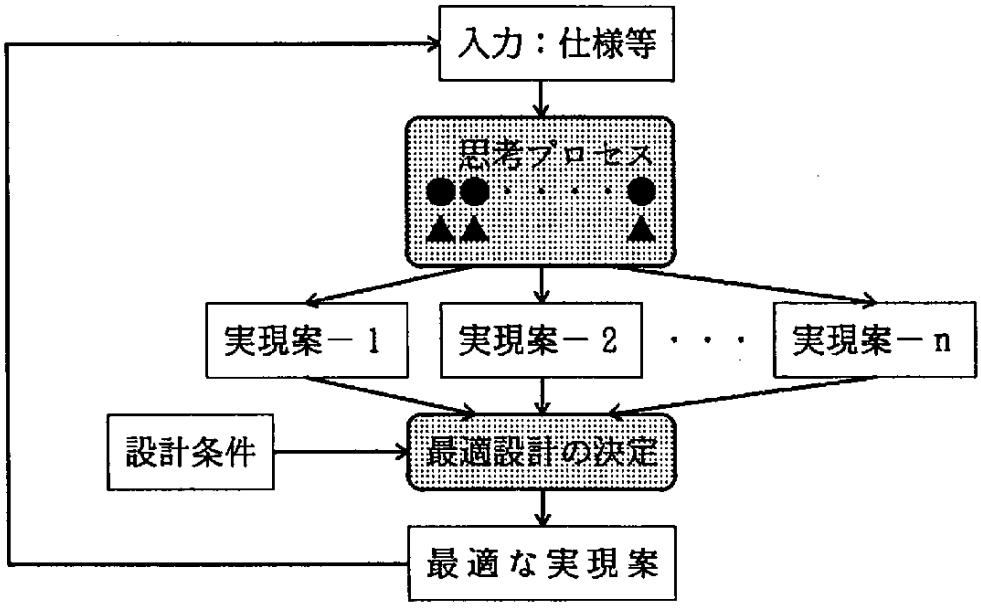


図 5.5-2　ソフトウェア設計における意思決定のプロセス

5.5.2 ソフトウェア開発における意思決定支援の適用

ソフトウェア開発における意思決定支援の特徴的な課題として、次の二つがある。

(1) 意思決定のプロセスを支援する

(2) 決定された意思を読み取る

以下では、(1) の例としてグループレビューの支援と分散開発のプロジェクト管理支援システムを取り上げる。また、(2) の例として設計理由の記述を紹介する。

5.5.3 グループレビューの支援

グループレビューは、ソフトウェアの品質を保証するために、グループの協調作業を積極的に利用しているプロセスである。個人の誤りや一人では発見できない問題が、グループレビューにより発見されている。米国、Bellcore(Bell Communications Research)のBrothersらは、グループレビューを支援する環境 ICICLE(Intelligent Code Inspection environment in a C Language Environment：アイシクル) を開発した[Brothers90]。ICICLEでは、レビューの準備、レビューミーティングの開催、レビュー報告書の作成から成るレビューの三つのプロセスを支援する。特に、グループウェアの観点からは、レビューミーティングの支援が興味深い。

図 5.5-3はICICLEのシステムアーキテクチャを示す。UNIXの上のX-ウィンドウ環境上で動作する。共用ウィンドウは、コードウィンドウ、コメントウィンドウ、プロポーザルウィンドウの三つのサブウィンドウから成る。例えば、コードウィンドウは、プログラムの読み手(Reader)がプログラムを追っていく画面と同一の画面を同期しながら他の参加者に表示する。また、コメントとの対応付けなどもできる。共通ウィンドウと個別ウィンドウを提供する仕組みもあり、会議支援システムの一アプリケーションと見ることもできる。

従来、参加者各自がリストをかき回して、プログラムを追っていたレビュー作業が、各自のWSの画面にプログラムを連動して表示することにりより極めて協調的に行える。また、レビューミーティングでのコメントの提出や討議、レビュー報告書の作成なども効率よく行える。また、ICICLEの試行を観察した結果、参加者の役割りに変化が認められた。議長(Moderator) の役割が相対的に低下するとともに、書記(Scribe)が単なる書記から意思決定に係わるようになった。

ICICLEの特徴は、レビューという特定のプロセスでのグループワークを支援することにある。Brothersらは、このようなグループウェアの開発と適用が比較的容易に実現できた理由として、レビューのプロセスが、もともと明確であったからであると指摘している。

ICICLEは、上記のシステムの後に、分散開発における利用を狙った二つのバージョンが開発されている[Brothers92]。一つは、ISDN回線により分散グループレビューを支援するTRICICLE(Tele-Remote ICICLE：トライシクル) である。さらに、インフォーマルなレビューを支援するために、簡単にレビ

ューの開催や参加が可能となるようなLAN上での通信プロトコルライブラリHydraを統合したシステムも開発している。このシステムでは、LAN上でのパケット化した音声(Packetized Voice)も利用して、音声と画面によるグループレビューを可能としている。

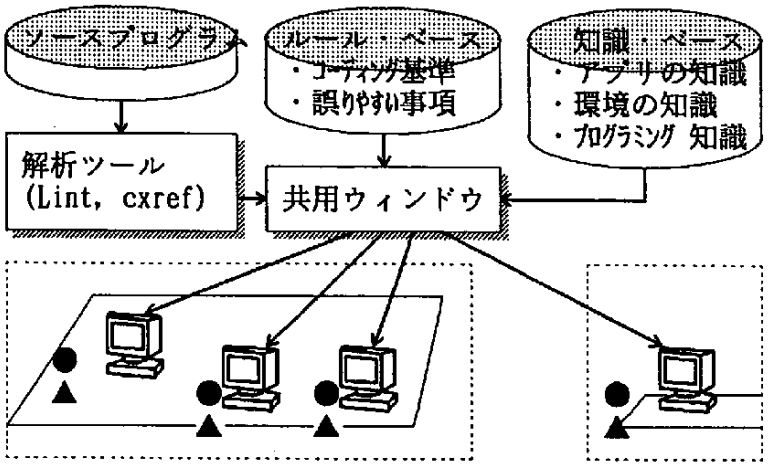


図5.5-3 ICICLEのアーキテクチャ

5.5.4 分散開発のプロジェクト管理

分散組織開発のプロジェクト管理のためには、プロジェクトの計画作成から進捗、品質などの正確な管理データを収集し、正しい意思決定を支援する必要がある。また、このようなデータに基づき、開発プロセスの改善にフィードバックできることが望ましい。しかし、組織の分散により、データの収集やフィードバックが問題となる。筆者らは、分散開発の試験プロセスを管理するために、一人一台のWSから自動的にデータを収集し、プロジェクトの進捗と品質を管理するための支援環境TOPICS (Test Outcome and Process Information Control System) を開発し、1988年から実際の大規模ソフトウェアの分散開発に適用してきた[青山90]。TOPICSのアーキテクチャを図5.5-4に示す。各開発者が作業日報として、各自のWSに登録した計画と実績のデータベースをネットワークを介して収集するネットワークシステムである。TOPICSの適用により、次のような効果が得られた。

(1) 一人一人、グループ、プロジェクト全体の各階層でデータの収集と進捗、品質の管理が行えるため、開発組織の各階層で適切な意思決定ができる。

(2) リアルタイムなデータ収集により、正確でかつ迅速な意思決定ができる。

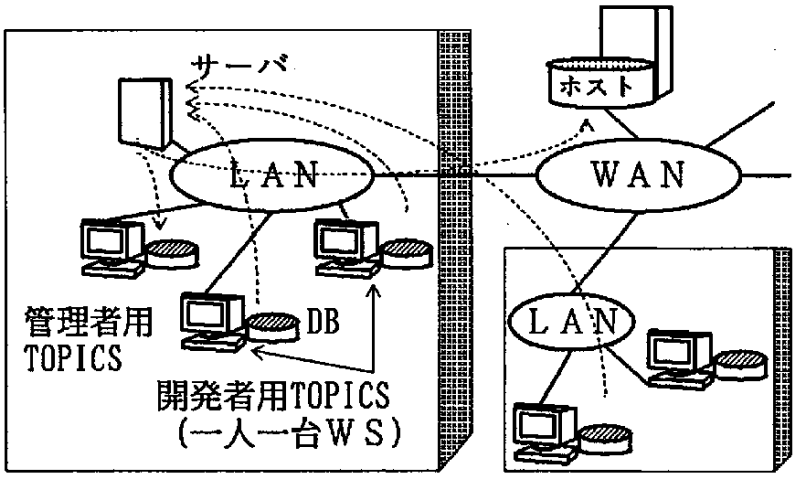


図 5.5-4　TOPICSのアーキテクチャ

図 5.5-5は、TOPICSによる問題処理票の解決プロセスを示す。問題処理票は保守における重要な情報である。これを、問題の発見者、解析者、処置者などが電子メールを介して送受することにより、分散開発における問題処理のプロセスが円滑になるとともに処置のプロセスが追跡可能となる。さらに、問題一件ごとに、その処置状況を管理者や開発者のWS上に表示し、進捗定例や問題対策会議などにおいて、処置方針の決定などに利用されている。

仲谷らは、ソフトウェアの問題解決プロセスには、次のような二つの特徴があることを指摘している[仲谷92]。

(1)　問題の発見者、解決策の決定者、処置者が別になる。

(2)　問題解決の最終意思決定者は、知識や技術だけでなく、処置に対する権限による。

TOPICSの開発においても、このような特徴が考慮されているが、ソフトウェア開発における意思決定支援においても、定常的な状態における意思決定と、問題解決における意思決定とのプロセスの違いや特徴を考慮する必要がある。

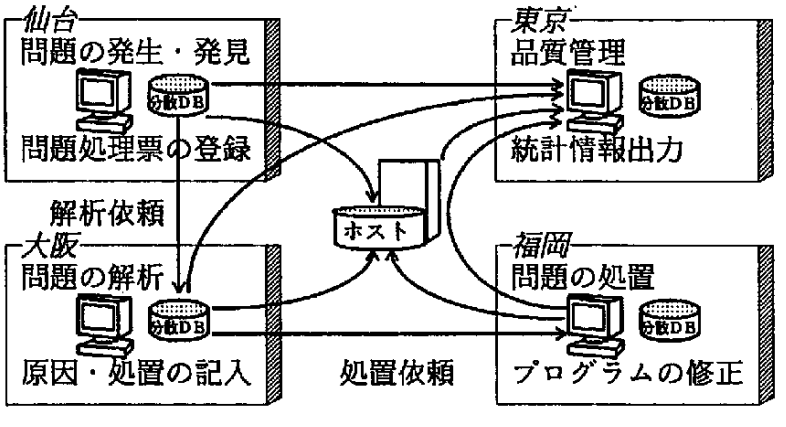


図 5.5-5　問題処理票の処置プロセス

5.5.5 設計理由の記述

設計理由(Design Rationale)とは、設計における決定や判断の理由「なぜ、そうしたのか？」である。最近、ソフトウェア工学の分野でも設計理由を明確に記録する重要性が指摘されてきた。特に、分散開発においては、設計者（特に上流工程の設計者）間や上流と下流とを分散して開発する場合の設計者間での設計情報の共有が極めて困難であるため、設計理由の記録が望まれている。また、保守工数が増大の一途にある現状では、設計理由は、保守の生産性と品質向上のためにも有用である。

グループウェアの概念に基づき、設計理由を記録するアプローチとして次の二つの例がよく知られている。

- IBIS(Issue-Based Information System)アプローチ
- 知識指向アプローチ(SIBYL)

(1) IBISの事例

① IBISのモデル

IBISは1970年代、ドイツにおいて、H. Rittel が建築設計や都市計画などの問題解決のために、複数の人が討議する方法として提案した。ハイパーテキストモデルに基づき、議論の構造化を狙いとする。米国、MCC のConklin らは、このモデルをソフトウェア開発に応用し、その支援環境を開発した[Conklin88]。図 5.5-6にIBISの議論のモデル（動的挙動）とその論理構造のモデル（静的構造）を示す。図中、問題(Issue) とは解くべき課題であり、見解(Position)は問題の解決案であり、意見(Argument)は見解に対する賛否の意見である。

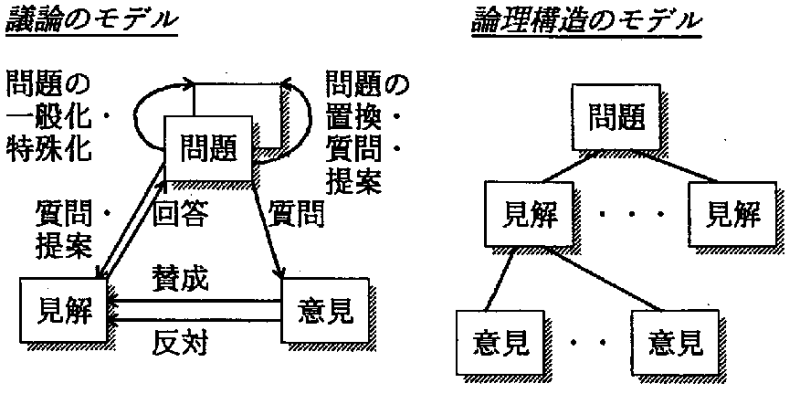


図 5.5-6 IBISのモデル

② IBISの試行評価

Conklin らはWSのグラフィカル・ユーザ・インタフェース(GUI) 上にIBISによるブレーンストーミングを支援する環境gIBIS(graphical IBIS) を開発し、評価した結果を２回にわたり報告している[Conklin88, Yakemovic90]。特に、Yakemovic らは、実際のソフトウェア開発で18ヶ月以上にわたり

IBISを試行し、次のような成果を得た[Yakemovic90]。

- 問題の理解が容易となり、問題点の早期発見が行えるため、開発コストの削減に寄与する。
- 議論済みの問題と議論未熟な問題を明確に区別できるので、議論の全体像を把握できる。
- 従来と異なる視点でレビューができる。
- 会議の進行（アジェンダ）を支援できるため、会議が本論から逸脱することが防止でき、会議の効率向上に寄与する。
- 分散した部門間でのコミュニケーションを改善できる。

一方、議論の匿名性により、誰の意見かわからない、などの意見も報告されている。

(2) SIBYL の事例

SIBYL は、MIT のLee らにより開発された[Lee90]。セミフォーマルな言語DRL(Decision Representation Language) により、議論を構造的な知識として知識ベースに蓄積し、GUI を介した支援環境により意思決定を支援する。図 5.5-7にSIBYL の議論のモデルを示す。

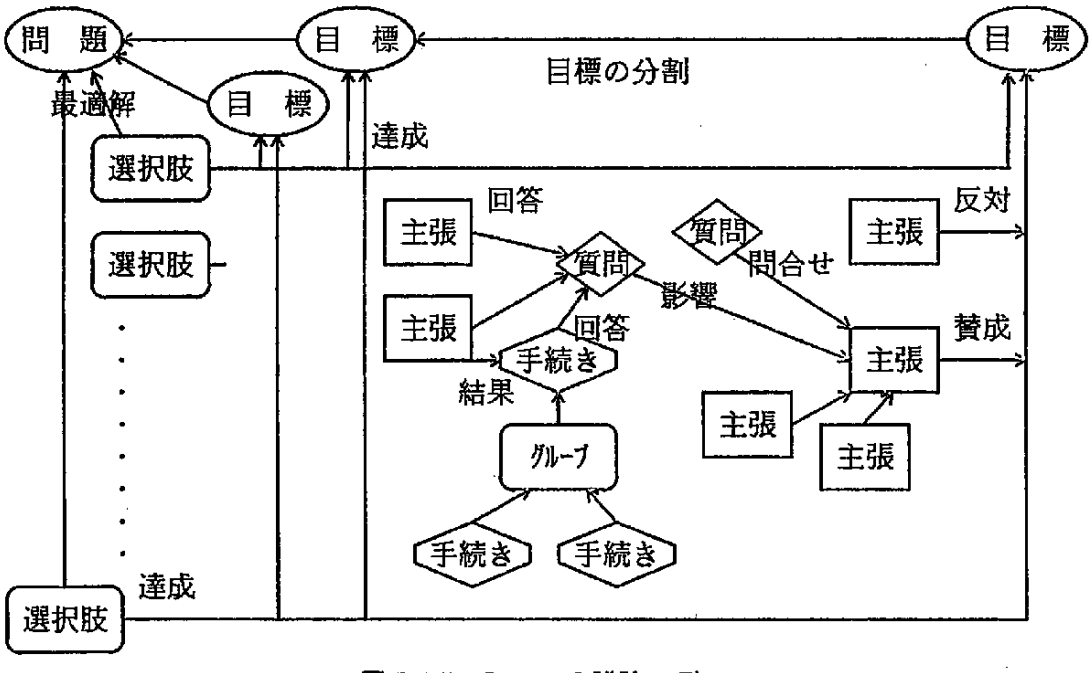


図 5.5-7 SIBYL の議論モデル

5.5.6 ソフトウェア開発における意思決定支援の今後のあり方

ソフトウェア開発は絶えざる意思決定のプロセスである。今後、ソフトウェア開発における意思決定支援を行うために、次の二つの方向で研究、開発を推進する必要があろう。

(1) 意思決定プロセスの支援

ソフトウェア開発の分散化や開発期間の短縮に伴い、ますます迅速で的確な意思決定支援が求められる。ソフトウェア開発においても、多くの意思決定は会議を通してなされることから、Colab に始ま

る多くのグループ会議支援システムは、ソフトウェア開発における意思決定支援を支援すると期待される。しかし、これらの意思決定支援システムは、必ずしも、ソフトウェア開発における意思決定の支援を目的にしているとは言えない。今後、グループレビューなどソフトウェア開発プロセスの理解やリスク管理技術の研究など、ソフトウェア開発における意思決定プロセスの研究を深め、ソフトウェアにおいて遭遇する意思決定の問題に直接応える必要があろう。

(2) 意思理解の支援

ソフトウェア開発における保守コストの増大に伴い、既存ソフトウェアのリエンジニアリングに対する要求が高まっている。リエンジニアリングのためには、単にプログラムレベルに止まらず、設計や要求分析段階での決定理由を知る必要がある。したがって、「設計理由の記述」とともに、設計理由の「リバースエンジニアリング」や「ソフトウェアの理解」を支援する技術が必要になろう。

【参考文献】

[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 (1990.6).

[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(1992.1).

[Brothers90] Brothers, L., et al.: ICICLE: Groupware for Code Inspection, Proc. ACM CSCW '90, pp. 169-181 (1990.10).

[Brothers92] Brothers, L.: Multimedia Groupware for Code Inspection, Proc. IEEE ICC '92, pp.1076-1081 (1992.6).

[Conklin88] Conklin, E.J., and Begeman, M. L.: gIBIS: A Hypertext Tool for Exploratory Policy Discussion, Proc. ACM CSCW '88, pp. 140-152 (1988.9).

[Kraemer88] Kraemer, K. L. and King, J. L.: Computer-Based Systems for Cooperative Work and Group Decision Making, ACM Computing Surveys, Vol. 20, No. 2, pp. 115-1146 (1988.6).

[Lee90] Lee, J.: SIBYL: A Tool for Managing Group Decision Rationale, Proc. ACM CSCW '90, pp. 79-92 (1990.10).

[仲谷92] 仲谷美江他: トラブルコミュニケーションモデルに基づいたソフトウェア開発プロジェクト支援システム, 電子情報通信学会ヒューマンコミュニケーション研究会, No. HC92-26, pp. 47-53 (1992.7).

[阪田92] 阪田史郎: グループウェアの実現技術, ソフト・リサーチ・センター (1992).

[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 (1992.1).

[Yakemovic90] Yakemovic, K. C. B., and Conklin, E. J.: Report on a Development Project Use of an Issue-Based Information System, Proc. ACM CSCW '90, pp. 105-118 (1990.10).

5.6 協調作業支援

本節では、ワークフロー管理や開発プロセス管理などの新しい協調作業支援方法とその環境について述べる。

5.6.1 ソフトウェア開発における協調作業支援

ソフトウェア開発おける協調作業を支援するには、図5.6-1に示すような開発者を取りまく次の三つの要素が協調する仕組みを構築する必要がある [青山91]。

(1) HHC (Human-Human Collaboration)

人と人やグループ間のコミュニケーションや活動を協調するための方法。

(2) HMC (Human-Machine Collaboration)

複数の人がワークステーションやネットワーク上に分散したリソースを相互に、かつ有効に使うための協調的なインタフェースの提供。

(3) HIC (Human-Information Collaboration)

人が情報を探し、参照し、理解し、共有するための情報のモデル化やアクセスインタフェースの提供。

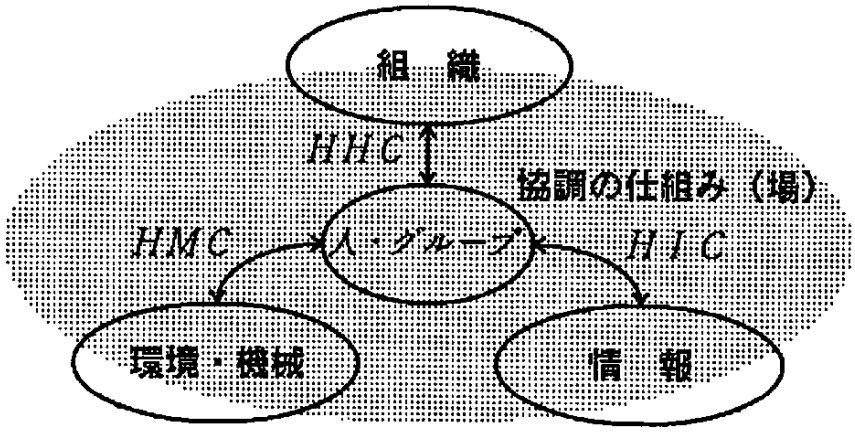


図 5.6-1 協調作業支援における三つのインターワーク

このような協調を支援するための種々の方法が提案されている。本節では、特に、ソフトウェア開発における人と人との協調(HHC) を支援する方法として最近注目されている「ワークフロー」管理とソフトウェア工学の研究から生まれた「開発プロセス支援」とを関係づけて議論する。

5.6.2 ワークフロー管理とプロセス管理

ソフトウェア開発の協調作業を支援する方法として、図5.6-2に示す二つのアプローチが提案されている。ソフトウェア工学の立場からは、「ソフトウェア開発プロセス」の概念に基づき、そのモデル化や管理、開発環境による支援方法などが提案されている[Osterweil87, Curtis92]。一方、CSCWやグループウェアの立場からは、「ワークフロー管理」として捉えられている。実際には、ソフトウェア開発においては、両者の目的は同一であり、ソフトウェアの開発「プロセス」を、その動的側面、すなわち挙動も含め、いかにモデル化し、設計し、管理するかが課題となる。

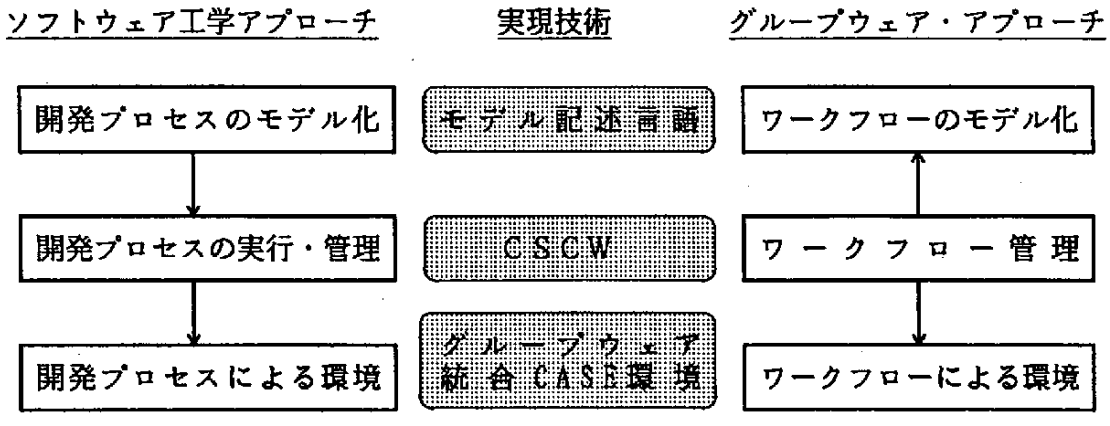


図5.6-2 ソフトウェア開発におけるワークフロー管理へのアプローチ

5.6.3 なぜ、ワークフロー管理や開発プロセスが問題なのか？

ソフトウェア開発における開発プロセスやワークフロー管理の発展過程を図5.6-3に示す。この二つのアプローチは、次のような点から今後ますます重要になると思われる。いずれも、その根底に「作業のプロセス」を重視する思想がある点に注目すべきである。

(1) 開発プロセス成熟度(Maturity)モデルの普及

米国Carnegie Mellon大学のHumphreyらにより提案された「プロセス成熟度(Process Maturity)」モデルが、米国を中心として普及しつつある[Humphrey89]。開発プロセスの「良さ」を5段階の尺度で評価するモデルである。開発プロセスの改善や国防省のソフトウェア開発契約の条件として用いられているため、米国産業界では、極めて重要視されている。プロセス成熟度モデルを用いて、開発プロセスの分析や改善が精力的に行われている。今後、ソフトウェア開発を改善するための基本的なアプローチとして広く認識されている。

(2) ISO 9000-3

「ISO 9000」は、ハードウェアの製造におけるプロダクト重視からプロセス重視へ転換する強力な誘因となっている。ISO 9000のソフトウェア版である「ISO 9000-3」も、ソフトウェア開発における同様なインパクトを秘めている [飯塚92]。今後、ISO 9000の普及に伴い、ソフトウェア開発における「プロセス」概念の普及が促進されるであろう。

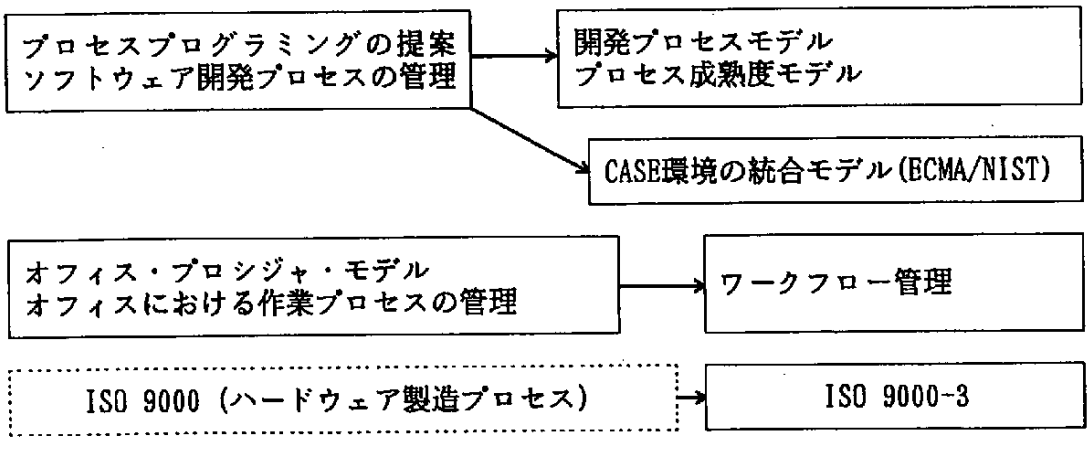


図 5.6-3 ソフトウェア開発におけるプロセス概念の発展

5.6.4 開発プロセス管理のアプローチ

ソフトウェア工学の研究から生まれた「プロセスプログラミング」[Osterweil87] の概念は、「開発プロセスモデル」として一般化され、個人やグループによる作業のプロセスをモデル化し管理する点で、ワークフロー管理と同一アプローチと言える。

(1) 開発プロセスモデルの狙い

開発プロセスのモデル化には、次のような狙いがある。

① プロセスの「リエンジニアリング(Re-engineering)」[Hammer90]

現在の開発プロセスモデルの分析を通して、その問題点を発見し、改善することにより開発の生産性と品質の向上、開発期間の短縮を図る。

② 協調（Coordination）と管理

開発に従事する人や物の間の協調を図り、より円滑な開発と管理を支援する。

③ プロセスに基づく統合環境(Process-driven Software Development Environment)

開発プロセスに基づき開発作業のガイドやツールの起動を統合し、より開発の実体に即した支援を行う。

(2) 開発プロセスのモデル化方法

開発プロセスとソフトウェアプロセスとの類似性から、開発プロセスにも情報システムのモデル化において重要な特性である「機能」、「構造」、「挙動」といった異なる側面があると考えられている [佐伯92]。このような異なる側面をモデル化するために、表 5.6-1 に示す様々なモデル化の手法が提案されている。しかし、情報処理システムと同様、複雑な開発プロセスの全体を単一のモデルで表現することは無理があり、表 5.6-2に例を示すように複数の視点を備えた統合的なモデル化が必要である[Aoyama90]。

表 5.6-1 プロセスモデルのアプローチ

アプローチ		プロセスモデルの例[Curtis92]
プロセスプログラミング	手続き型	APPL/A(Ada Process Programming Language based on Aspen)
	関数型	HFSP(Hierarchical and Functional Software Process)
	制約指向	GRAPPLE
機能のモデル化		DFD (Data Flow Diagram)
構造のモデル化		ERD (Entity-Relationship Model)
挙動のモデル化		ペトリネットモデル(RADDLE, OOPM)
		State Chart モデル

表 5.6-2 複数の特性をモデル化する例

プロセスモデル例	機能（手続き）	構　造	挙　動
APPL/A [Sutton90]	Ada	クラス・インスタンス・インヘリタンス	イベント／トリガ
OOPM [Aoyama90]	YAC II （木構造チャート）	Coad-Yourdon OOA 手法	ペトリネット

5.6.5 ワークフロー管理のアプローチ

「ワークフロー」の概念は、グループウェアの研究からプロセスに着目したアプローチである。これは、オフィス・プロシジャ・モデル、すなわち、オフィス作業のプロセスモデルから発展した概念である。初期のグループウェアではコミュニケーション支援が中心であったが、最近のグループウェアでは、このワークフローの管理を支援するものが開発されている [川上92]。表 5.6-3にワークフロー管理システムの例を示す。

表 5.6-3 ワークフロー管理システムの例

アプローチ	システムの例	適用領域
プロセスベースの統合開発環境	統合CASE環境[Mi92 他] ECMA/NIST の統合環境モデル	ソフトウェア開発専用
タスクモデルによる分散型ワークフロー管理	EuroCoOpシステム[Hennessy92]	オフィス作業一般
会話モデル(Speech Acts Theory)構造的な会話モデルに基づく作業フローの調整と管理	Coordinator [Medina-Mola92] AMIGO [Danielsen86] ECF [Karbe90]	

5.6.6 ソフトウェア開発におけるワークフロー管理の適用

表 5.6-3から、ソフトウェア開発におけるワークフロー管理を適用する方法として、次のような事例が報告されている。

(1) プロセスベースの統合開発環境

プロセスベースの統合開発環境は、開発プロセスやプロダクトを各開発者のワークステーションに表示し、各工程の作業に適したツールを自動的に起動したり、各生成物が求める条件を満足しているか判定したりすることにより、統一的な作業環境を提供する。例えば、Mi らは、CASEツールを統合するフレームワークとして開発プロセスの利用を提案している[Mi92]。 図 5.6-4は、Miらの提案する、開発プロセスによるCASEツール統合環境のグラフィカル・ユーザ・インタフェースを示す。

(2) タスクモデルによるワークフロー管理：EuroCoOp システム

ESPRITプロジェクトの一環として、分散した開発活動間の協調を支援するEuroCoOpシステムが開発されている[Hennesy92] 。図 5.6-5にシステムのアーキテクチャを示す。グループ協調活動支援を行う各種グループウェアのツールを統合する基盤環境をツールキットとして提供する。この中で、「半構造化(Semi-structured) 作業」をタスクと呼び、このタスクに基づき協調作業支援を行う「アクティビティ・コーディネーション・ツールキット」が開発されている。このモデルが生まれた背景には、従来

の「構造化(Structured)作業（プロセス）」による協調作業支援環境が柔軟性に乏しく、実際のソフトウェア開発プロセスの支援には適さないとする考えがある。

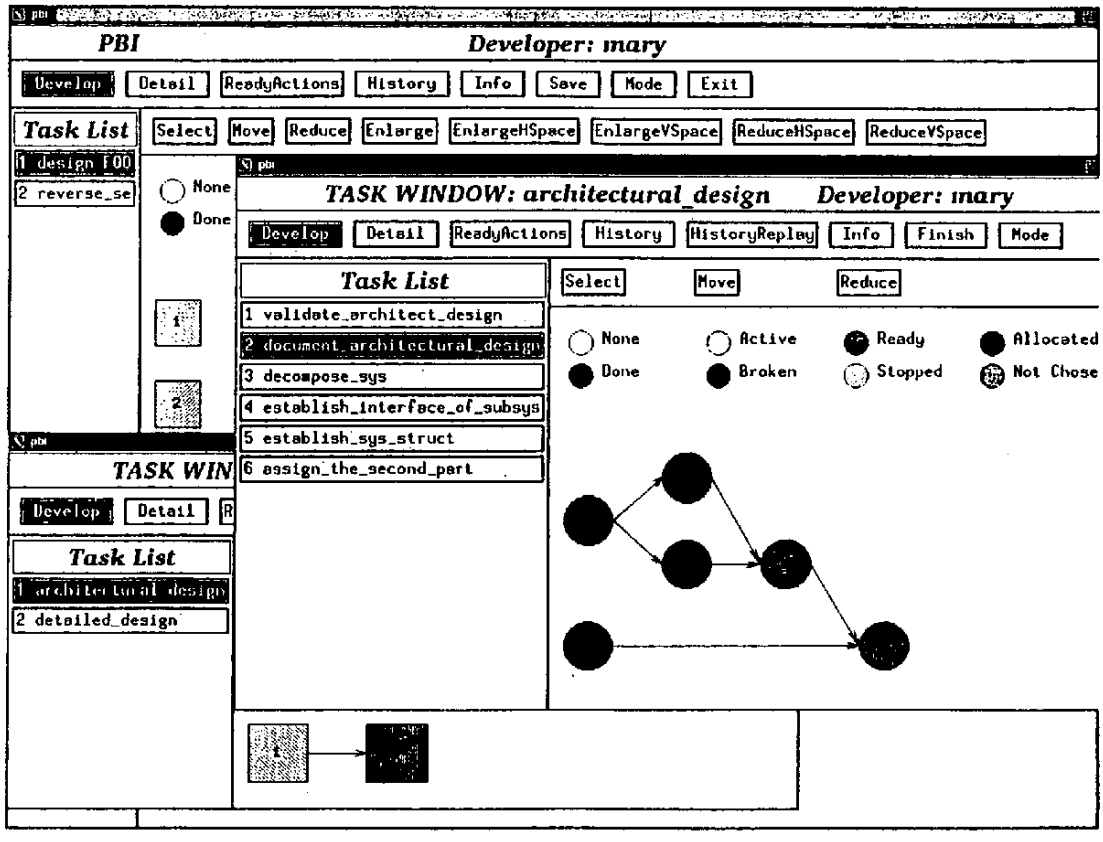


図 5.6-4　プロセスベースの統合開発環境

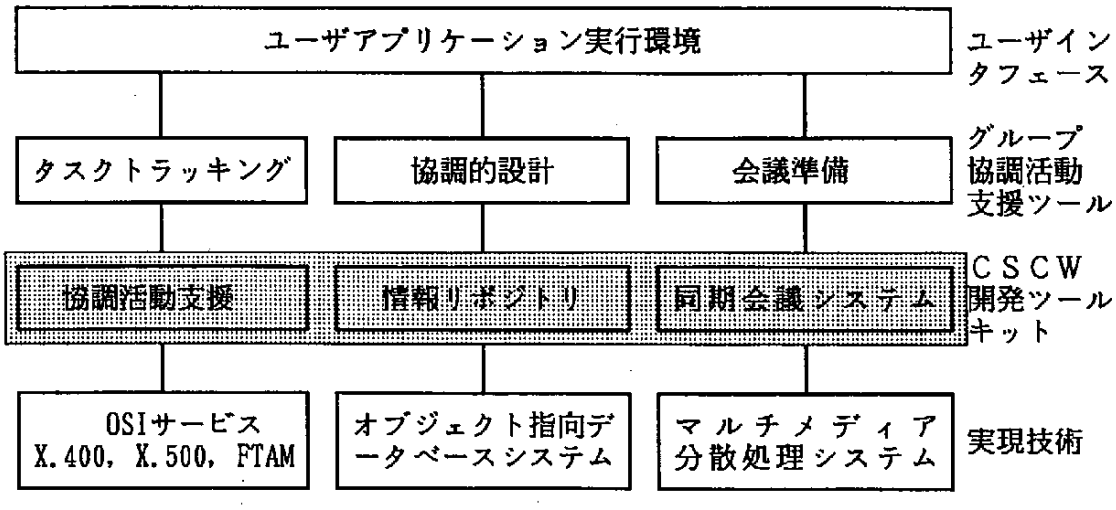


図 5.6-5　EuroCoOpシステムのアーキテクチャ

図 5.6-6は、EACT(EuroCoOp Activity Coordination Toolkit)のアーキテクチャを示す。各開発者の作業の協調(HHC) や作業の管理は、ユーザインタフェース層を介して、アクティビティモニタにより行われる。作業 (タスク) の内容 (目的、作業内容、担当者など) はアクティビティストアに定義されている。必要であれば、メッセージハンドリング層を介して、各作業者間のコミュニケーションが図られる。

このように、ワークフロー管理は、開発プロセスに基づく管理システムと極めて類似している。しかし、このタスクモデルが提起しているように、プロセスやワークフローの「構造化の度合い」をどうするかが重要な課題であり、対象とする作業内容や特性に依存する。ソフトウェア開発においても、中流工程のルーティン化された作業は構造的なプロセスと言えようが、上流工程の要求分析などは極めて非構造的なプロセスと考えられる。

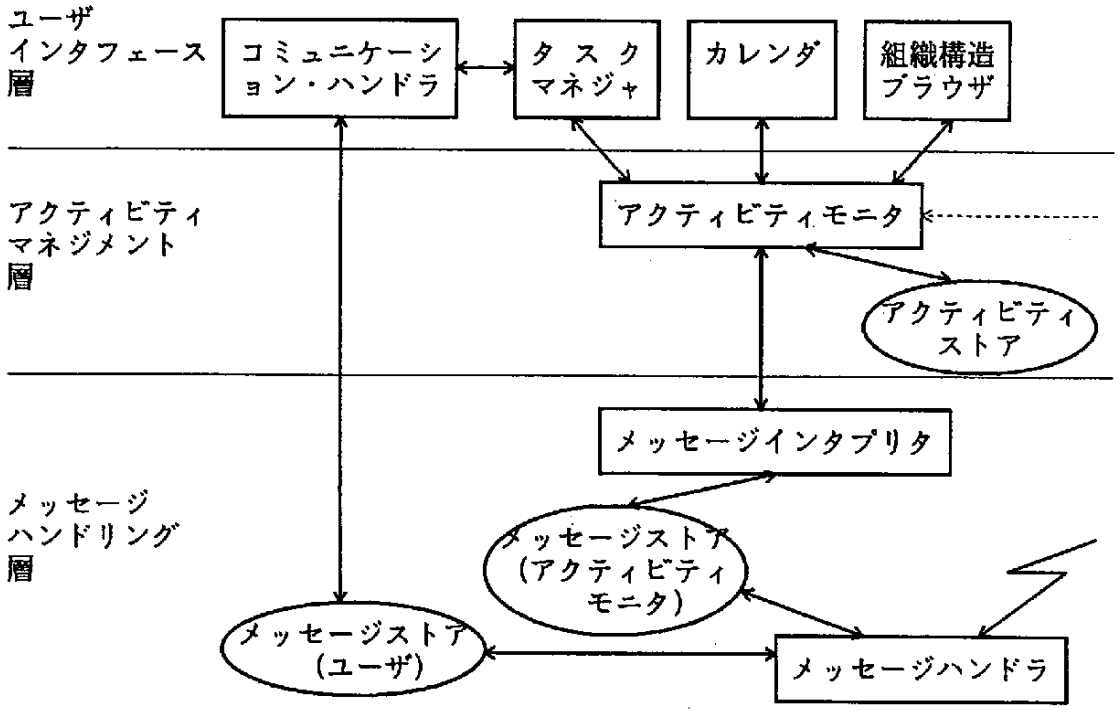


図 5.6-6　EuroCoOpシステムのアクティビティ・コーディネーション・ツールキット

5.6.7　協調作業支援の今後の方向と課題

開発プロセスやワークフロー管理を中心とする協調作業支援は、ソフトウェア開発において、次のような点で、今後ますます重要な役割を果たすと期待される。

(1)　プロセスの果たす役割の増大

ソフトウェア開発において「プロセス」の果たす役割が重視されていることから、プロセス管理やワークフロー管理は、今後、ますます重要なアプローチになると期待される。特に、ソフトウェアの

分散開発や並行開発が進むにつれて、ソフトウェア開発におけるプロセスやワークフローの管理はますます複雑で困難な課題となっている[Aoyama87, 青山92]。

(2) 作業の効率化から価値創造へ

開発プロセスやワークフロー管理に対する従来のアプローチは、主に、構造的なプロセスを中心として、作業の効率化に主眼があった。しかし、ソフトウェア開発においても付加価値の追求や顧客満足度の向上が重要な課題となっていることから、今後は、上流工程を中心とする非構造的なプロセスの支援が重要な課題となろう。しかし、このようなプロセスそのものが「オープン」であり、かつ、その実行主体である人にも大きな差があることから、どこまで構造化し、どのように管理すべきかが明確でない[Hewitt86]。また、ワークフロー管理の技術をソフトウェア開発作業へ適用するためには、一般のオフィス作業とソフトウェア開発作業との相違を理解する必要があろう。そのためにも、今後、ソフトウェア開発プロセスの理解を深める必要がある。

ワークフロー管理や開発プロセス支援環境は、単に、ワークフローやプロセスの管理だけではなく、その背後にあるプロセスそのものや企業組織、さらには企業カルチャーをも変革する力を秘めている。一方、このような作業のやり方に関わるグループウェアは、その利用者の意識や作業の進め方、ひいては、企業組織や文化が変わらなければ、かえって、生産性や協調性を低下させる危険がある。また、同一企業内であっても、同じグループウェアがどの部門でも有効であるとは限らない。このようなグループウェアの開発や導入にあたっては、開発プロセスやワークフローそのものを分析し、理解を深めることが成功への鍵である。場合によっては、仕事のやり方を変えるだけで、グループウェアを導入する必要がないこともあるかもしれない。一方、現在のワークフロー管理システムや開発プロセス支援環境には、まだまだ、改善の余地があるように思われる。今後、この分野の研究、開発の推進を期待したい。

【参考文献】

[Aoyama87] M. Aoyama: Concurrent Development of Software Systems: A New Paradigm of Software Development, ACM Software Engineering Notes, Vol. 12, No. 3, pp. 20-24 (1987.7).

[Aoyama90] M. Aoyama: Distributed Concurrent Development of Software Systems: An Object-Oriented Process Model, Proc. IEEE COMPSAC '90, pp. 330-337 (1990.11).

[青山91] 青山幹雄: 分散開発とグループウェア, 情報処理振興事業協会技術センター技術発表会論文集, pp. 152-153 (1991.10).

[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 (1992.1).

[Curtis92] Curtis, B., et al.: Process Modeling, Comm. ACM, Vol. 35, No. 9, pp. 75-90 (1992.9).

[Danielsen86] Danielsen, T., et al.: The Amigo Project: Advanced Group Communication Model for Computer-Based Communications Environment, Proc. ACM CSCW '86, pp. 115-142 (1986.12).

[Hammer90] Hammer, M.: Reengineering Work: Don't Automate, Obliterate, Harvard Business Review, No. 4, pp. 104-112 (1990.7-8).

[Hennessy92] Hennessy, P., et al.: Distributed Work Management: Activity Coordination within the EuroCoOp Project, GMD-FIT News (AC Special), No. 10, pp. 1-20 (1992.6).

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

[Humphrey89] Humphrey, W. S.: Managing the Software Process, Addison-Wesley (1989).

[飯塚92] 飯塚悦功: ソフトウェアの品質管理と品質保証, 情報処理, Vol. 33, No. 8, pp. 922-933 (1992.8).

[Karbe90] Karbe, N., et al.: Support of Cooperative Work by Electronic Circulation Folders, ACM SIGOIS Bulletin, Vol. 11, No. 2, pp. 109-117 (1990.4).

[川上92] 川上潤司: ワーク・フロー管理が仕事を変える、組織を変える, 日経コンピュータ, pp. 56-75 (1992.9.21).

[Medina-Mola92] Medina-Mola, R. et al.: The Action Workflow Approach to Workflow Management Technology, Proc. ACM CSCW '92, pp. 281-288 (1992.11).

[Mi92] Mi, P., and Scacchi, W.: Process Integration in CASE Environments, IEEE Software, Vol. 9, No. 2, pp. 45-53 (1992.3).

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

[佐伯92] 佐伯元司: ソフトウェアプロセスとネット理論, ネット指向ソフトウェア設計技術に関するチュートリアル講演論文集, 電子情報通信学会ソフトウェアと離散事象研究会, pp.12-2 (1992.4).

5.7 新しいソフトウェア開発環境

5.7.1 ツール統合とソフトウェアプロセス実行環境

ソフトウェア開発を効率良く進めるためには、各種ツールがバラバラに存在するではなく、体系的にまとまっていたほうが使いやすく、また効果も大きい。最近では、体系的にまとめられたツール群のことを開発環境と呼ぶようになってきている。こうした問題を最初に明確にしたものとしてDoD（米国防総省）が、1980年に発表したAPSE（Ada Programming Support Environment）と呼ばれるAdaのためのプログラミング支援環境がある。APSEは、KAPSE、MAPSE、APSEの三層の機能構造からなっている。KAPSEは、核支援機能でありOSに対応している。MAPSEは、プログラミングに必要な最小限度のツールの集まりであり、APSEは、Ada向きのツールの集まりを示している。こうした米国での動きに刺激されて、ヨーロッパにおいてPCTE（Portable Common Tools Environment）と呼ばれる開発環境が開発されてきている。PCTEは、当初EC（欧州共同体）のEspritプログラムで開発が進められ、現在はECMA（ヨーロッパコンピュータ製造業者協会）により普及が進められている。一方、ヨーロッパの産業界を中心としたEurekaプログラムでは、ESF（Eureka Software Factory）と呼ばれるソフトウェア開発環境の構築のためのプロジェクトを1986年にスタートさせている。このプロジェクトは10年計画であり、コミュニケーションを中心としたツール統合環境の構築を目指している。ここでは、新しいツール統合環境としてPCTEとESFを解説し、さらにこれらの基盤の上に開発されているソフトウェアプロセス実行環境について述べる。

(1) PCTEとALF

PCTEでは、ツール統合環境として参照モデル、オブジェクト管理システム、インテグレーション機構、メッセージサービスのためのインタフェースを定めている。PCTEの中心は、オブジェクト管理システム（リポジトリ）であり、このリポジトリを介して各種ツールが統合されることになる。

このPCTEにソフトウェアプロセスモデリングの機能を付け加えたものがEspritプログラムで開発されたALF（Accueil de Logiciel Futur）と呼ばれるシステムである。

ALFでは、ソフトウェアプロセスモデルのことをMASP（Models of Assisted Software Processes）と呼んでいる。これは、ALFが、ソフトウェアプロセスの実行と言うよりはプロセスのガイドや支援に重点を置いているためである。ソフトウェアプロセスは、オブジェクトモデル、操作タイプ、条件式、ルール、順序、制約条件によってモデル化される。図5.7-1にALFに基づいたIPSEのアーキテクチャを示す。

最下層は、PCTEのオブジェクト管理システム（OMS）であり、その上に情報システム、案内システム、ユーザインタフェースが置かれる。

① 情報システム

情報システムは、プロセスモデルを定義したり、実行する上で必要な次の機能をもっている。

・ ソフトウェアプロセスモデル生成したり解釈する上で必要な情報の蓄積と検索

・ PCTEのOMS内でのMASPの意味的一貫性の保持

・ イベントの発生を検知して対応するアクションを発火させるメカニズム

② 案内システム

案内システムは、ソフトウェア開発者によってなされた作業を監視し、必要とあらば適切なアクションをとる。

案内システム内にあるMASPの部分には、MASPを記述するためのエディタ、意味チェッカ、MASPモデルをプロジェクト等の特性に合わせて具現化するツールが含まれている。MASPで記述されたプロセスモデルはMASPインタプリタによって解釈実行される。MASPインタプリタは、プロセスモデルに従って必要な行動をとるのみならず開発作業者の質問に答えたり、ガイドを行うことができる。例えば、editツールを実行することができるかとか、なぜeditツールを実行することができないかという問い合わせに対して説明を行う。また、次に何を行うことができるかとか、次にどのツールを実行すべきかといった問い合わせに対してガイドを与える。

③ ユーザインタフェース管理システム

ユーザインタフェース管理システムには、プレゼンテーションマネジャ、会話コントロール、アプリケーションインタフェースが含まれている。

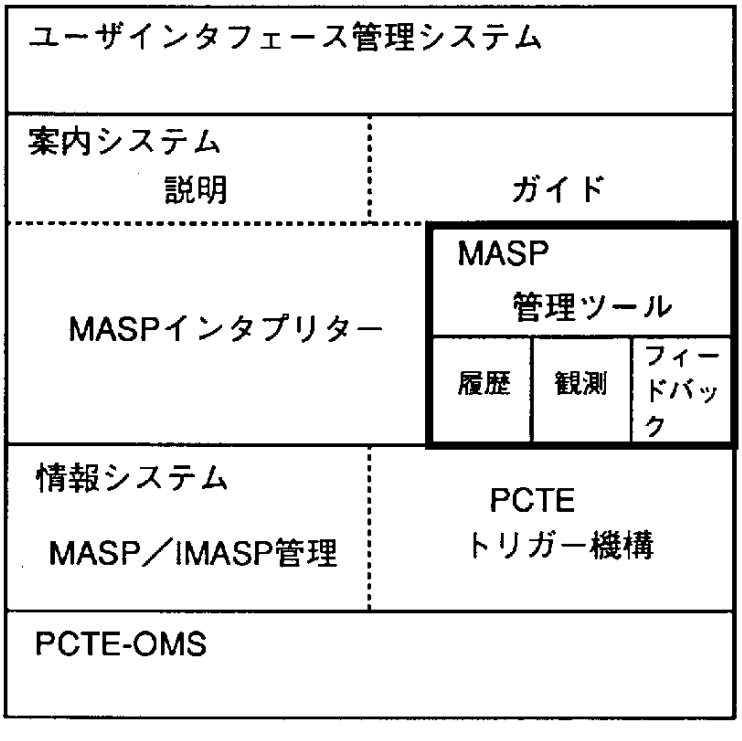


図5.7-1 ALFに基づいたIPSEのアーキテクチャ

(2) ESF

ESF（Eureka Software Factory）は、CASE統合環境開発のためのプロジェクトであり、5ヶ国13の機関が参加している。開発には毎年200人年以上の工数がかけられている。図5.7-2にESF環境参照アーキテクチャを示す。

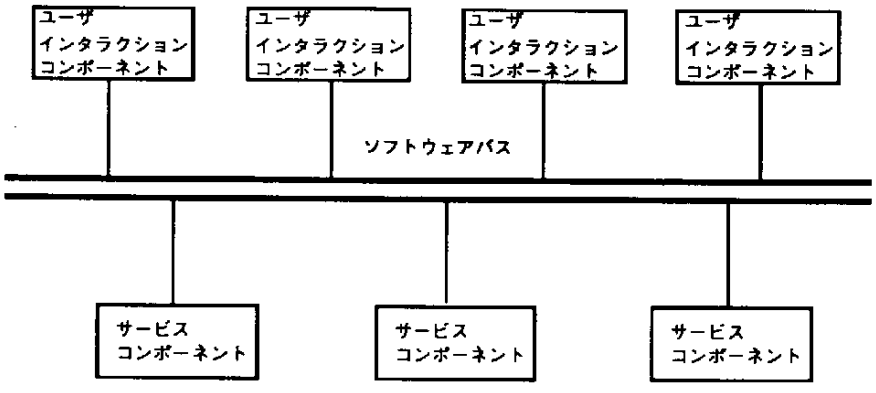


図5.7-2　ESF環境参照アーキテクチャ

ESF環境は、ユーザインタラクションコンポーネントとサービスコンポーネントからなっている。中間に位置するソフトウェアバスは、コンポーネントの分散や異種性を吸収しコンポーネントの内部実行環境に制約を加えない機構になっている。

ユーザインタラクションコンポーネントとサービスコンポーネントの分離は、SmalltalkのMVC（Model-View-Control）の概念からきている。ユーザインタラクションコンポーネントはSmalltalkでいうview-controlに該当し、サービスコンポーネントは、modelに該当する。

ESFでのユーザインタラクションはワークコンテキストと呼ばれる概念で行われる。図5.7-3にユーザインタラクション環境を示す。ワークコンテキストは、インタラクションオブジェクトが集まったもので、インタラクションオブジェクトは、情報とツールを隠蔽している。開発要員がある役割（ロール）に割り当てられると、プロセスプログラムが、その作業員に対してワークコンテキストの実行を可能にさせる。また、作業員が実行を完了するとワークコンテキストを削除する。

図5.7-4にユーザインタラクションの実現例を示す。このツールはAgendaと呼ばれるもので仕事を選択したり、仕事を生成して他の要員に依頼することができる。左上のウインドウには、実行可能な仕事が重要度と緊急度を軸とした二次元空間に配置されている。この中の一つを選ぶとその仕事の内容が右上のウインドウに表示される。右中のウインドウには、インタションオブジェクトが表示される。右下は仕事の完了状態を指示すボタンが表示される。

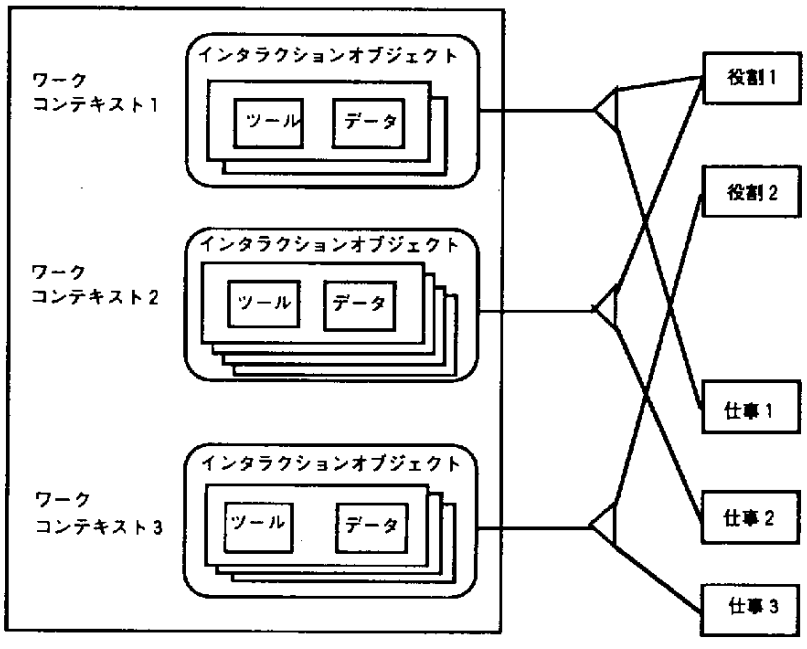


図5.7-3　ユーザインタラクション環境

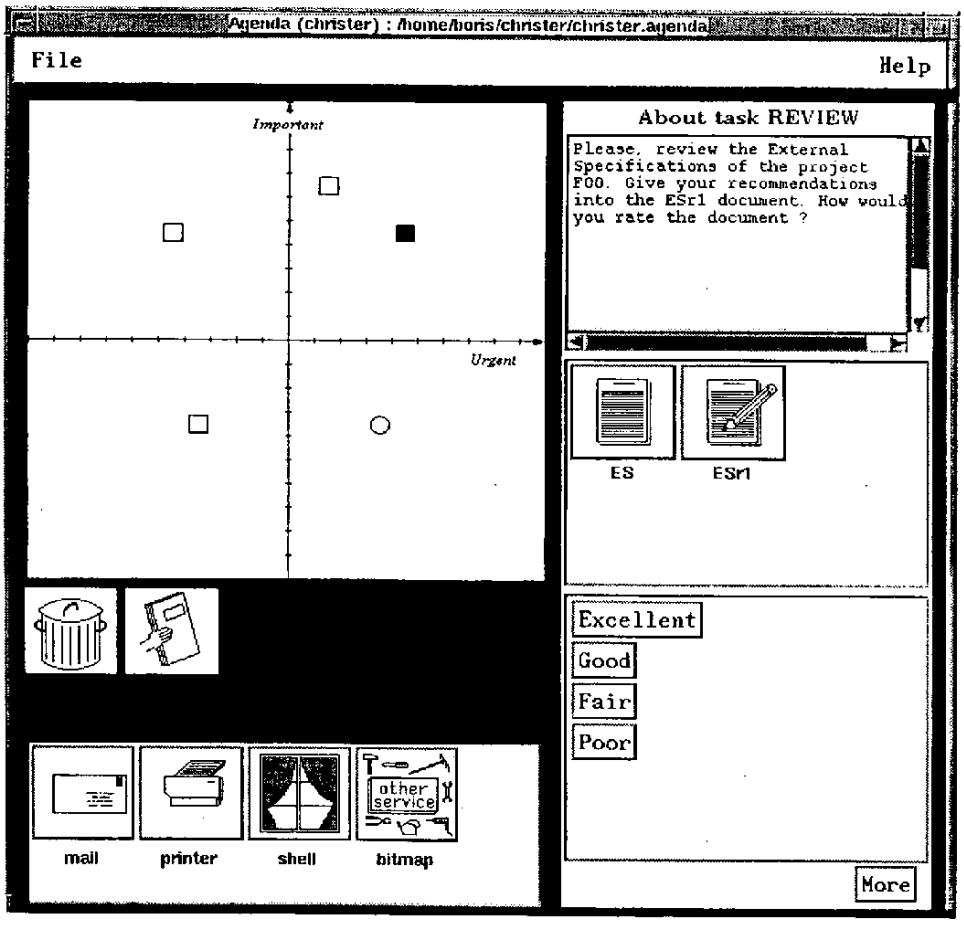


図5.7-4　ユーザインタラクションの実現例（Agendaツール）

5.7.2　オープンなグループワーク支援環境

先に見てきたように、現状のソフトウェア開発環境はツール統合の上にソフトウェアプロセス実行環境を構築するといった方向で研究開発が進められている。ここで考えられているソストウェアプロセスは、例えばすべてのモジュールのコンパイルが正常に終了したならばリンケージを行うといった程度のものでソフトウェア開発時に発生する様々な協調作業の問題を扱っていない。しかし、ソフトウェア開発工程の上流においては、様々なメンバの共同作業が必要とされていることから、今後の開発環境はこうした協調作業の支援をも組み込む必要がある。

協調作業を支援するツールとして、Information Lens、DOMINO、Coordinator等様々なものが開発されている。しかし、これらはいずれも個別に開発されてきているため互いに孤立したものとなっている。

一方、現実の協調作業は、これらのツールが単独に支援している内容よりはるかに複雑なものであることから、各アプリケーションはインタオペラビリティを保って意味的統合がされる必要がある。

こうした観点からEC（欧州共同体）のCOST14計画に支援されたMOCCAプロジェクトでは、CSCWの統合環境の研究開発を進めている。

MOCCAでは、既存のグループウェアやそこで用いられているモデルを分析し、図5.7-5 に示すCSCW統合化のためのアーキテクチャを開発した。

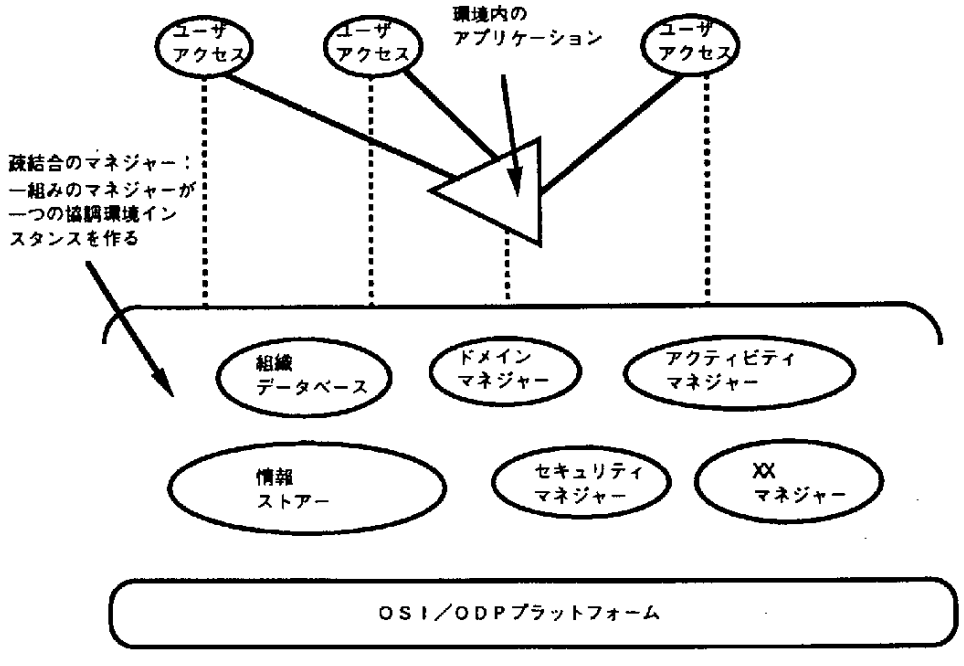


図5.7-5　MOCCAアーキテクチャ

MOCCAは、ISOで標準化されているOSI（開放型システム相互接続）や標準化が進められているODP（開放型分散処理）をコミュニケーション基盤としている。

この上に現在のところ次の五つのマネジャを考え、各種アプリケーションが関連をもって実行できるようにしている。

① ドメインマネジャ

ドメインマネジャは、各種オブジェクトをドメイン単位にまとめ見通しのよいものにするためのものである。ドメインとは、ポリシーあるいは共同作業の単位をいう。

② アクティビティマネジャ

アクティビティマネジャは、アプリケーションで支援されるアクティビティを登録する。例えば、アクティビティの目的、状態、参加者、リソースなどがアクティビティマネジャに登録される。

③ 情報ストア

情報ストアは、環境内に存在するオブジェクトに対する共通的な蓄積サービスを提供する。

④ 組織データベース

組織データベースは、組織に関する情報を蓄積する。例えば、組織で使われるリソース、従業員、役割、組織間の関連、組織の手順やポリシーが蓄積される。

⑤ セキュリティマネジャ

セキュリティマネジャは、システム内のリソースの利用やアクセスを管理する。

この他にも、例えばMERMAID等が用いているアプリケーションの複製を管理するマネジャの必要性についても検討している。

こうした研究と先に示したCASE環境を統合化させていくと図5.7-6 に示すような統合化環境が考えられる。現在のところCSCWを研究しているグループとCASEを研究しているグループでは、あまり交流が行われておらず、またOSIやODPのグループとも交流がない。今後は、各コミュニティのメンバーが集まり、全体的なフレームワークを作り、その中で調和の取れた研究開発が進むことが望まれる。

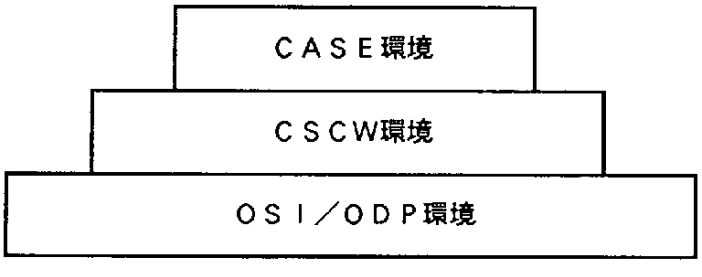


図5.7-6　CSCWとCASE環境の統合

【参考文献】

[斎藤87] 斎藤信男: ソフトウェア開発環境,情報処理,Vol.28,No.7,pp.887-897(1987).

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

[Canals91] Gerome Canals et al. : Working in an ALF-based IPSE, Toulouse'91, pp.549-560(1991).

[Fernstrom92] Christer Fernstrom et al. : Software Factory Principles, Architecture, and Experiments, IEEE Software, pp.36-44 (1992.3).

[Fernstrom91] Christer Fernstrom et al. : Integration Needs in Process Enacted Environment, 1st International Conference on the Software Process, pp.142-158 (1991).

[Navarro92] Leandro Navarro et al. : CSCW requires Open Systems, GMD-FIT-News, AC Special No.15, pp.1-20 (1992.11).

[Mocca92] The Mocca Group : Mocca : An Environment for CSCW Applications, ACM SIGOIS Bulletin , pp.21-23 (1992.4).

[Linington92] P. F. Linington : Introduction to the Open Distributed Basic Reference Model, Open Distributed Processing, North-Holland, pp.3-13 (1992).

6．ソフトウェア開発における
グループワーク支援の実現例

6．ソフトウェア開発におけるグループワーク支援の実現例

6.1 マルチメディア分散グループレビューシステム

6.1.1 はじめに

近年、ワークステーションの普及やLAN等のネットワークの進展によりソフトウェア開発環境は大きく変化してきている。例えば、ワークステーションの普及に伴いCASE等のツールが開発され、開発工数の軽減、設計品質の向上がもたらされている。従来のツールの多くは、開発メンバーが個別に行う作業を支援するものであったが、開発時には調整、意思決定、連絡等非常に多くの協同作業が存在している。したがって、こうしたグループでの協同作業を支援できるならば大きな効果が期待できる。こうした背景の基に、ソフトウェア開発時のグループ作業で特に重要なレビュー作業を取り上げ、その支援システムを開発した。システムは、分散開発環境でのレビュー作業の効率化を目指したもので、レビューしようとするドキュメントの表示ウインドウとコメント、カウンタコメントを記録するカードスタックを持っている。また、ドキュメント中の指摘箇所とコメントはリンク構造をもち、文字単位で関連付けられるようになっている。コメント、カウンタコメントはテキストの他に音声および動画を用いて行うことができる。本節では、開発の経緯、システムの概要、利用方法等について述べる。

6.1.2 ソフトウェア開発におけるレビュー作業

ソフトウェアの開発は、抽象的な課題をコンピュータで実行できる最終成果物（実行可能なプログラム）にする過程であり、規模が大きなものになると開発期間が長くなり、また参加メンバーも多数になる。その結果、中間成果物に誤りが含まれたり、メンバー間で情報がうまく伝わらない等の問題が発生している。こうした問題の発生を避けたり、解決策を検討するために開発の途中で様々なレビューが行われる。レビューの種類としては、管理レビュー、技術レビュー、インスペクション、ウォークスルー、監査等があり、様々な形態、手順で実施される。

例えば、代表的なレビューであるウォークスルーは、次のように行われる[IEEE89]。

① 目的

ウォークスルーの目的は、ソフトウェアの構成要素を評価することである。ウォークスルーは、プログラムコードの検査手法として長く用いられてきたが、概念設計、詳細設計等のソフトウェア構成要素にも適用できる。その目的は、欠陥、欠落、矛盾を発見しソフトウェアの品質を向上させたり、代替方法を検討することである。この他に、ウォークスルーは、技術の交換、参加者の教育といった面も持っている。

② 構成要員と役割

ウォークスルー会議は、調整者、書記、作成者、レビュー担当者から構成される。調整者は、会議の進行管理等を行い、書記はウォークスルー会議で指摘されたすべてのコメントを記録する。作成者はソフトウェア構成要素を作成した人であり、レビュー担当者は、コメント等を作成する人である。これらの役割は、メンバー内で共有されることがある。例えば、調整者は作成者が兼ねる時がある。

③ 用意するもの

ウォークスルーを行うに当っては最小限、ウォークスルーの目的を書いたドキュメント、レビュー対象物、作成時に使用した標準、仕様書を用意する。

④ 開始条件

ウォークスルーは、一般に品質保証計画やソフトウェア開発計画に基づいて実施される。また、ソフトウェア構成要素の作成完了がウォークスルーのトリガーとなる。

⑤ 手順

まず、調整者が、チームの結成、スケジュールの作成、必要なドキュメントの事前配付を行う。つぎに作成者によって概要の説明が行われる。この後、各レビュー担当者は、各自でドキュメントのレビューを行い、ウォークスルー会議に備える。ウォークスルー会議では、まず全体的な説明がなされ、つぎに作成者がドキュメントを逐一読み上げる。各レビュー担当者は、その内容に対して質問を行ったり、検討すべき事項を指摘する。書記は、会議で取り挙げられたコメントや決定を記録しレポートとしてまとめる。

⑥ 終了条件

ウォークスルーは、十分に詳細な検討に基づいて、すべての欠陥や欠落の指摘、効率に関する議論、向上のための助言がなされ、それらがレポートとしてまとめられたとき終了する。

⑦ 成果物

最終的に出される成果物は、ウォークスルーレポートである。

こうしたレビューによる効果は、既に多くの報告により明らかにされている。しかし、実施するためには、会議室の予約、レビュー・メンバーの選定、スケジュール調整、ドキュメントの事前配付、レビュー実施、レビュー結果の作成等煩雑な仕事が多く、また多くの工数を要することから必ずしも十分に実施されていないのが実情である。また、最近では次のような問題点も顕在化してきている。

- レビューできる能力をもった人材が少なく特定の人に作業が集中する。
- オフィス空間の事情から会議室の予約が困難になってきている。
- オフィス等の関係で参加メンバーが分散しており一堂に会するのが困難になってきた。

・ 要員等の不足から外注が多くなりスケジュール調整が困難になってきている。
・ ドキュメントが電子媒体で作成されているのにレビューは依然紙媒体で行われている。
・ 開発規模の大型化に伴いドキュメントの量が多くなってきた。

6.1.3 グループレビューシステムの開発

(1) システム基本方針

レビューは、メンバーが会議室に集まって行うのが一般的であるが、先に述べたようにメンバーがなかなか一堂に集まれないことから、メンバーが分散したままの状態で行えることを基本にした。また、現状のネットワーク環境で、対面のリアルタイム会議環境を構築することは伝送速度、容量、コスト等の面で困難なため、電子メールを用いた非同期でのレビュー作業が行えるようにした。

ユーザインタフェースの設計に当たっては、操作方法について特別な予備知識がなくても利用できるようクリック操作で各種機能が実行できるようにした。さらに、各種機能は階層プルダウンメニューによってパラメータが選択できるようにし操作性の向上を計った。

(2) コメント付けの方法

非同期でレビューを行う場合、ドキュメント作成者とレビューメンバーの間でリアルタイムの会話ができないことから、ドキュメント中のどこに対する指摘であるかを明確に示す必要がある。一般にドキュメントとコメントの対応を示す方法として表6.1-1の方法がある。このシステムでは、見やすさ等の観点から分離型の方法をとることにした。また、ドキュメント中の指摘箇所をマーカでマークする感覚で文字の白黒反転ができるようにし、さらにコメントとの自動リンク付けを計った。

表6.1-1 コメント付け方法の比較

	方法	利点	欠点	例
カラム型	原文の横にコメントカラムを設ける	原文と同じ行にコメントできる	原文章が細切れになる	PREP
分離型	原文とは別の用紙にコメントを書く	原文が維持される	原文との対応が取りにくい	For Comment ICICLE
オーバレイ型	原文の上にコメントを上書きする	指摘箇所が特定できる	文字が見にくくなる	MarkUP
貼付型	原文の上にコメントを張り付ける	コメントを重ねられる	一度に両者を見れない	MacDraw

(3)　マルチメディアのコメント

最近のワークステーションは、音声や動画等のメディアをテキスト等の従来のメディアと同様に扱うことができるようになってきている。

非同期のレビューシステムとした場合、従来のレビュー会議でみられたリアルタイムの会話ができないため、テキストによるコメント付けに加えて音声や動画によるコメント付けを行えるようにした。

(4)　システムの機能とユーザインタフェース

機能としては、レビュードキュメントと関連を持たせたコメント付け、カウンタコメント付け、コメント結果に基づく品質の自動表示等がある。ユーザインタフェースの基本である画面は、マルチウインドウで構成され、オリジナルドキュメント表示領域、コメント入出力領域、カウンタコメント入出力領域からなっている。コメントとカウンタコメントは対で１枚のカードを形成し、スタックに保存される。図6.1-1に表示例を示す。

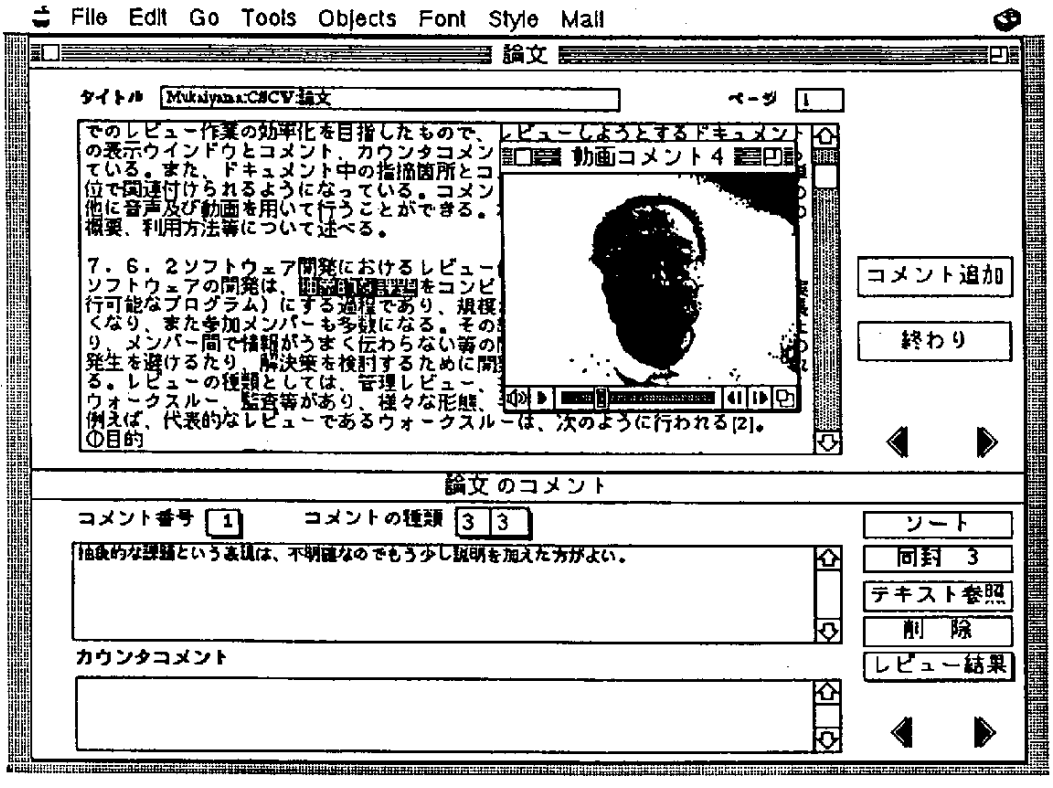


図6.1-1　画面表示例

①　コメント付けのための操作

まず、オリジナルドキュメント表示領域内にあるドキュメントをスクロールさせながら検査していく。コメントを付ける箇所を発見したならば、その箇所をマウスでドローする。ドローとは、指摘箇所の先頭にカーソルを移動させマウスのボタンを押下したまま、指摘箇所の最後までカーソルを動か

すことである。こうすることにより指摘箇所の文字が白黒反転する。指摘は、文字単位で行うことができる。このようにして、指摘箇所を示した後コメント追加ボタンを押下するとコメント入出力領域に新しいコメントカードが現われ、コメントが入力できる状態になる。ドキュメント全体に対するコメントの場合は、直接コメント追加ボタンを押下する。この場合、確認のメッセージがシステムから出される。

図6.1-1に示すようにコメントカードには、コメント入力ウインドウの他に次のボタンがある。

・　コメントの種類

コメントの種類は、階層プルダウンメニューになっており、コメントの種類を選択できる。このコメントの種類は、通常レビュー基準としてあらかじめ決まっているものである。後に説明する品質は、この基準によって計算される。

・　ソート

ドキュメントにコメントを付けていくとき、ドキュメントの先頭から次々とコメントを付けて行くだけでなく前に戻ってコメントを付けたい場合などがある。このソートは、そうして付けられたコメントをドキュメントの出現順に並び変えたりするためのものである。また、全体に対するコメントもドキュメント検査中に随時書かれる場合がある。こうした場合でも最後にソートすることにより全体に対するコメントをまとめることができる。

・　同封

図を用いたコメント、音声によるコメント、動画によるコメントは同封として受け渡すことができる。同封ボタンをクリックすると、どのメディアを同封／参照するかのメニューが表示され、次の階層のメニューとして作成／参照／削除が表示される。これらは、必要に応じて選択する。

・　テキスト参照

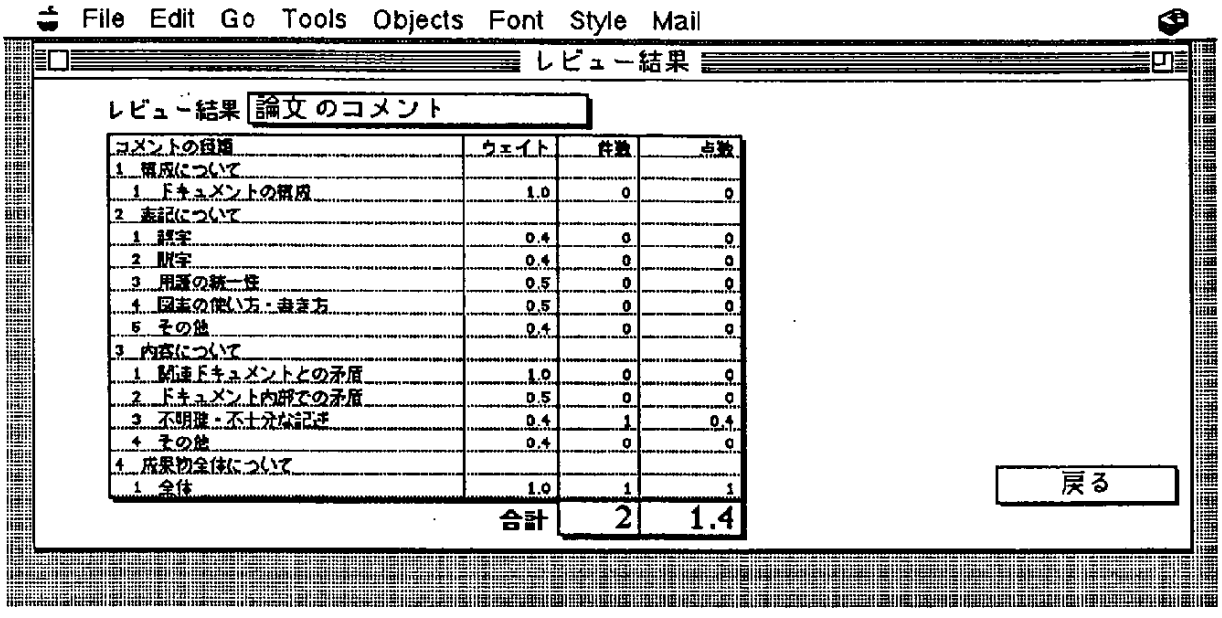
通常、オリジナルドキュメントの指摘箇所とコメントは、自動的にリンク付けされているため、コメントを次々とめくると、それに連動してオリジナルドキュメントが自動的にスクロールし、該当指摘箇所が白黒反転して表示される。この状態でオリジナルドキュメントのスクロール等を行うと現在のコメントに対応した指摘箇所が分からなくなる。そうした場合にこのテキスト参照のボタンを利用する。このボタンが押下されると常に現在表示中のコメントに対応した指摘箇所が表示される。

・　削除

既に付けたコメントを削除するときに利用する。

・　レビュー結果

このボタンを押下すると、コメントの種類ごとの件数が計算され、コメントの種類に応じた品質が計算される。図6.1-2にレビュー結果の表示例を示す。このボタンは、レビュー中いつでも押下でき、その時点でのレビュー結果を見ることができる。



コメントの種類	ウェイト	件数	点数
1 構成について			
1 ドキュメントの構成	1.0	0	0
2 表記について			
1 誤字	0.4	0	0
2 脱字	0.4	0	0
3 用語の統一性	0.5	0	0
4 図表の使い方・書き方	0.5	0	0
5 その他	0.4	0	0
3 内容について			
1 関連ドキュメントとの矛盾	1.0	0	0
2 ドキュメント内部での矛盾	0.5	0	0
3 不明瞭・不十分な記述	0.4	1	0.4
4 その他	0.4	0	0
4 成果物全体について			
1 全体	1.0	1	1
	合計	2	1.4

図6.1-2　レビュー結果

② カウンタコメント付けの操作

カウンタコメント入出力領域は、コメントに対してドキュメント作成者が応答するとき使用する。例えば、指摘されたコメントの内容な明確でないときや、指摘に対してどのように修正するか等を記述する。

このカウンタコメント入出力領域は、コメント入出力領域と対で作成される。

③ システムの構成

システムは、Macintoshの上に構築されており、HyperCardを利用している。また、コメントスタックの転送には、既存の電子メールを利用している。

6.1.4 グループレビューシステムの利用

このシステムは、レビューメンバーが遠隔地に分散している環境で非同期に作業が行えることを前提に開発を行ったが、他のアプリケーションと組み合わせて利用することにより図6.1-3に示すように全てのグループ作業形態で利用でき、実験を進めている。

(1) 対面同期形態

メンバーがレビューのための会議室に集まり、直接対面状態でレビューを行う形態である。

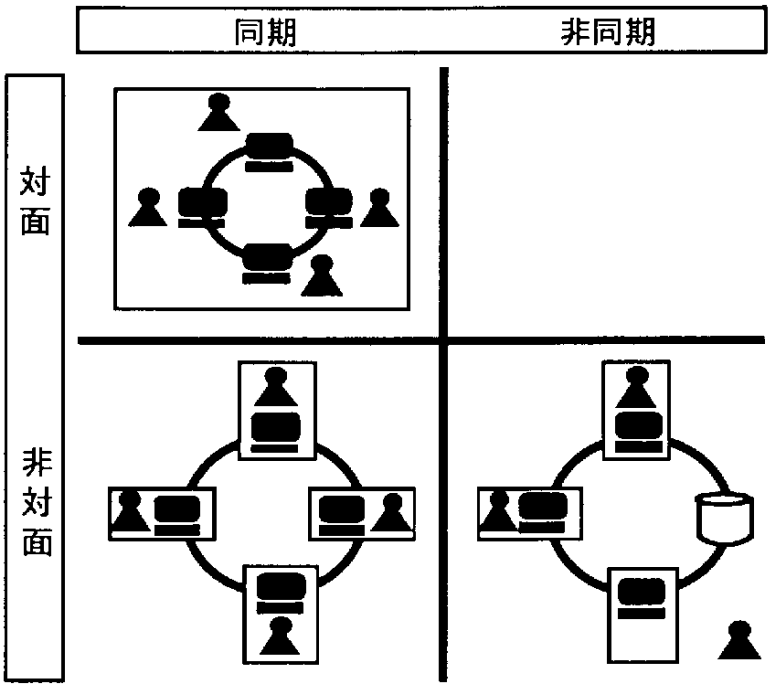


図6.1-3　利用形態

実施に当たっては、Timbuktuなどの画面共有システムと大画面プロジェクタを組み合わせて電子会議システムを構築し、その上でこのシステムを稼働させる。画面共有システムとは、あるワークステーションに表示されている画面を自画面内に取り込むことのできるシステムである。レビューメンバーは各自のワークステーションでレビューを行い、それを電子会議室に持ち込む。

(2)　非対面同期形態

メンバーが遠隔地に分散している状態で、ネットワークを用いて会議を行う形態である。この場合も画面共有システムと組み合わせて利用する。各レビュー担当者は、自分のワークスペースに居ながらレビュー会議が行える。メンバーが遠隔地に離れており、他のメンバーの意図が見えないことからフロアーパッシングの機能が必要になる。非対面の場合、共有画面のみで作業を行うことは困難であり、電話を利用して会議の進行を行っている。電話により、システムにフロアーパッシングの機能がなくても衝突などは起きない。

(3)　非対面非同期形態

メンバーが分散している状態で、電子メールを用いてレビューを進める形態である。

対面のレビュー会議では、テキストなどの静的な媒体に加えて表情、動作、声を用いて意図を伝えることができるが、この形態ではそうしたことができないため、録音あるいは録画して授受すること

になる。

(4) 混合形態

何人かが会議室に集まり、残りのメンバーが会議室以外の場所に分散した状態で行う形態である。

6.1.5 おわりに

分散ソフトウェア開発環境下におけるレビュー作業の支援を目的に、分散レビューシステムの開発を進めてきた。

このシステムはプロトタイプであり、現場での利用を行いながら有効性の確認、システムの改善改良作業を行っている。現実の利用では、レビューの他にちょっとした検討を頼む場合等に多く使われ効果を上げている。音声や動画などのマルチメディアの利用については、興味があるものの作成の操作が面倒であったり、CCDカメラ等の装置がすべてのワークステーションに備え付けられているわけでない等の理由によりあまり使われていない。これらマルチメディアは、単に聴いたり、見たりすることができるといった使い方の他にもっと有効な使い方を今後検討すべきである。

レビュー作業は、欠陥、欠落、違反等の指摘のみでなく、その指摘が次のバージョンのドキュメントにどのように反映されているかの確認も重要となる。今後は、レビューシステムと連係した形でドキュメントの変化が確認できるシステムの構築を目指したい。

【参考文献】

[Brothers90] L. Brothers et al. : ICICLE : Groupware For Code Inspection, CSCW'90, pp.169-181 (1990).

[IEEE89] IEEE Standard for Software Reviews and Audits, IEEE Std 1028-1988 (1989).

[JIPDEC92] グループワーク支援システムの研究開発報告書, 日本情報処理開発協会 (1992,1993).

[Neuwirth90] Cristine M. Neuwirth : Issues in the Design of Computer Support for Co-authoring and Commenting, CSCW'90, pp.183-195 (1990).

[Neuwirth92] Cristine M. Neuwirth : Flexxible Diff-ing In A Collaborative Writing System, CSCW'92, pp.147-154 (1992).

[Timbuktu90] Timbuktu User's Guide, Farallon (1990).

6.2 CAPSDFのグループワーク支援

CAPSDF（Computer-aided Application System Development Framework）はソフトウェア開発方法論/技法と統合型CASEからなる日立アプリケーション開発支援体系である。CAPSDFはクライアントサーバ型（CSS型）分散開発環境を実現しており、CASEプラットフォームとリポジトリを中心にしてグループワークを支援している。

6.2.1 CASEプラットフォーム

図6.2-1に統合型CASEの構成を示す。CASEプラットフォームはNISTとECMAのリファレンスモデル（トースタモデル：図6.3-1を参照）のコンセプトに基づき、ツール統合、ユーザインタフェース統合、データ統合、プロセス統合のメカニズムを提供している。ツール統合は支援ツール間のイベント情報の通信により、支援ツールから別の支援ツールの起動を行う。ユーザインタフェース統合は国際標準であるOSF/Motifで統一している。

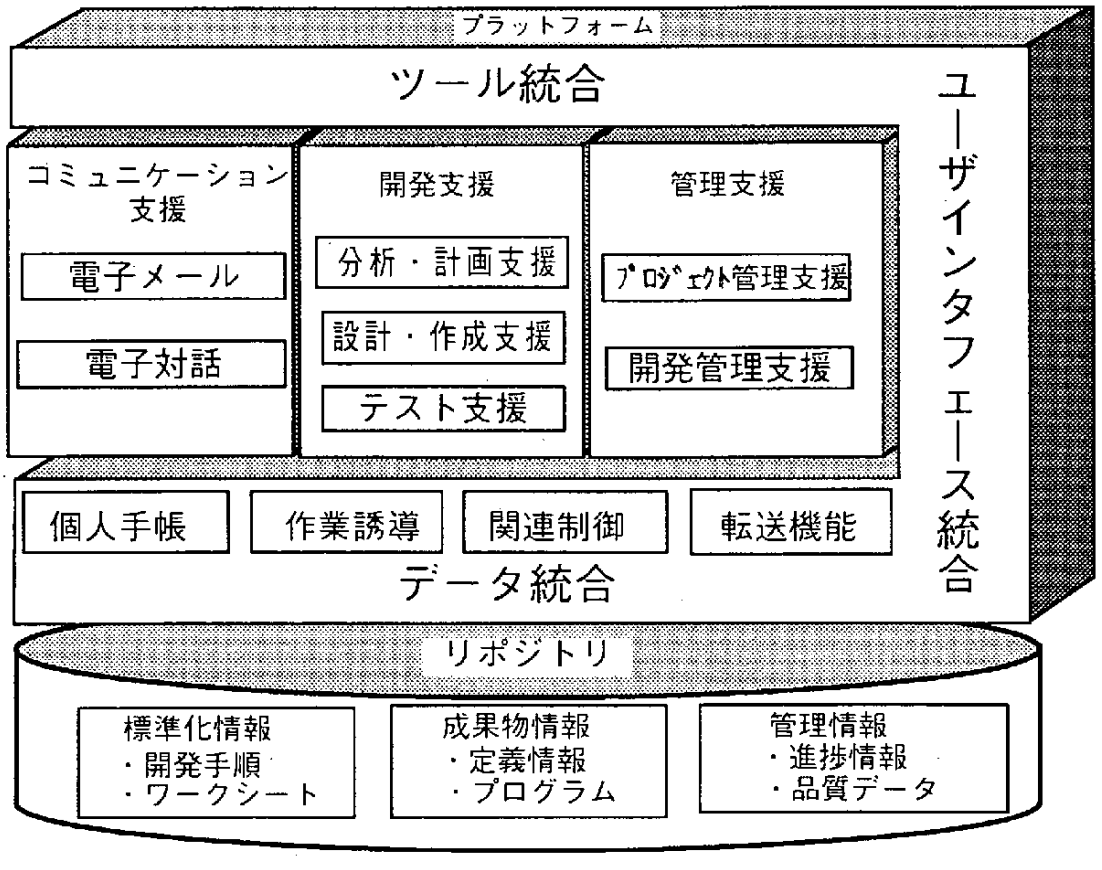


図6.2-1 統合型CASEの構成

データ統合では情報資源倉庫（リポジトリ）により情報の一元管理を実現している。情報の整合性を維持すると共に、関連制御機能により変更通知の送受信、変更状況の監視ができる。リポジトリには作業誘導で用いる開発手順／ワークシートや、システム開発で用いるデータ項目辞書／プログラム部品などの標準化情報、定義情報／プログラムなどの成果物情報、プロジェクト管理で用いる管理情報が管理される。リポジトリではこれらの情報を一括して仕様書という単位で管理している。特に設計ドキュメントも管理されるので、例えば支援ツールとワードプロセッサツールとの連動、ソースプログラムと設計ドキュメントとの相互変更通知ができる[田村92] [築城92] [末宗92]。

6.2.2 グループワーク支援機能

CAPSDFは、大規模な情報システムを分散開発環境で開発する方式を体系化している。分散開発の意味を広義に考えると、地域的な分散開発、複数の会社参画による協同開発、多様なスキルの要員による開発、工程別の分担開発等がある。このような複合開発形態のグループワークを支援する機能が要求される。またプロジェクトの役割（管理者、開発者、プログラマ、標準化担当者など）別の機能も必要である。

開発作業の流れと関連するグループワーク支援機能を図6.2-2で説明する。開発者はログイン後個人

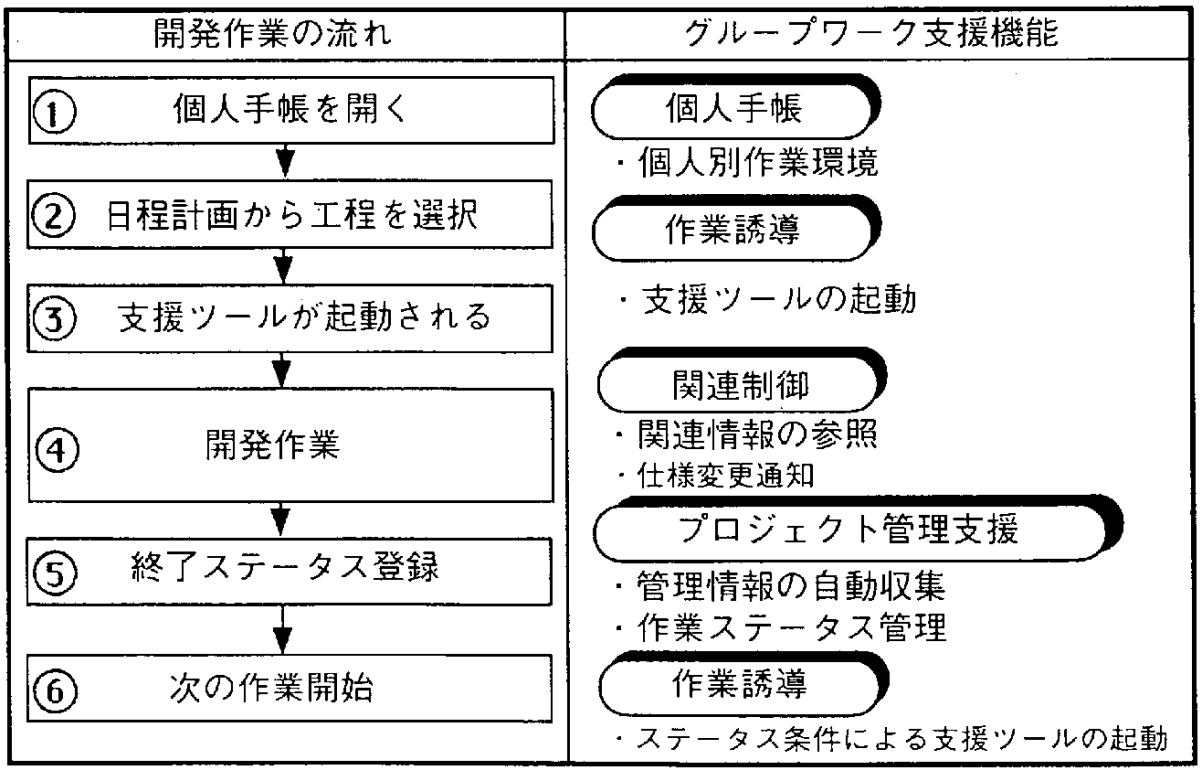


図6.2-2 開発作業の流れ

手帳を開く（①）。個人手帳には役割別個人別に作業環境が設定しており、日程計画、管理作業、仕様書一覧、電子対話等開発に必要な作業や情報が入っている。プロジェクトの役割別に標準作業環境を設定し、つぎに個人のスキルに合わせた開発環境（サービスメニューと成果物一覧）を設定する。例えば初心者では細かい作業単位でのレビューが必要だが、熟練者では大きな作業単位のレビューで充分な場合がある。個人手帳の中から日程計画を選択し、日程計画表を表示する。日程計画表には作業予定と実績がアローダイヤグラムで表示[JIPDEC91]されている。日程計画表のアローを選択（②）すると、対応する支援ツールが起動される（③）。支援ツールを用いて開発作業を行う（④）。情報システムの開発とは、ユーザからの要求をコンピュータシステムでの実行形式に変換する作業であり、抽象情報の具体化と詳細化の一連の手続きの作業である。

前の工程での成果物を転記、参照、詳細化しながら開発を進める。開発者は参照したい仕様書をマルチウインドウに表示しながら開発作業を行う。また作業中にひらめいたアイデアや残しておきたい設計根拠等は、作業を中断することなく設計メモに記録する。設計メモもリポジトリで管理される。仕様書の変更により他の仕様書に影響を与える時、関連仕様書に変更通知する。該当関連仕様書の担当者は、作業開始時変更メッセージを見て対応する。開発作業が終了すると、開発者はステータス登録する（⑤）。ステータスはプロジェクト毎に仕様書単位で定義しており、例えば「作成中」「作成完」「承認完」等である。このステータスの条件により次の作業が誘導される。プロジェクト管理者は開発者の作業状況をステータス情報で把握する。また作業終了時、支援ツールは管理情報をリポジトリに登録するので、ステータスだけではなく定量的な作業状況の把握ができる。

6.2.3 作業誘導

作業誘導では、設定した作業手順に従って支援ツールを起動し、開発者の作業をガイドする。作業誘導の実現法としてスクリプト実行方式を用いている。これは作業手順定義情報よりスクリプトを自動生成し、スクリプト実行により作業誘導する方法で、カスタマイズが容易にできる利点がある。作業手順定義では作業の階層と順序、仕様書、ステータスパターン、作業誘導条件等を定義する。支援ツールは仕様書の種類から一意的に決定される。仕様書がドキュメントの場合、支援ツールは汎用のワードプロセッサツールになる[吉川92]。

図6.2-3は作業誘導の例である。開発者が日程表から作業Bを選択（①）すると、該当するレコード仕様定義ツールが起動される（②）。レコード仕様定義ツールは「未着手」「作成中」「作成完」のステータスパターンがあり、作業者が「作成完」のステータスを選択（③）すると、スクリプトが次の支援ツールであるコピー句生成ツールを起動する。またレコード仕様定義ツールはレコード数、データ項目数等の管理情報をリポジトリに登録する。

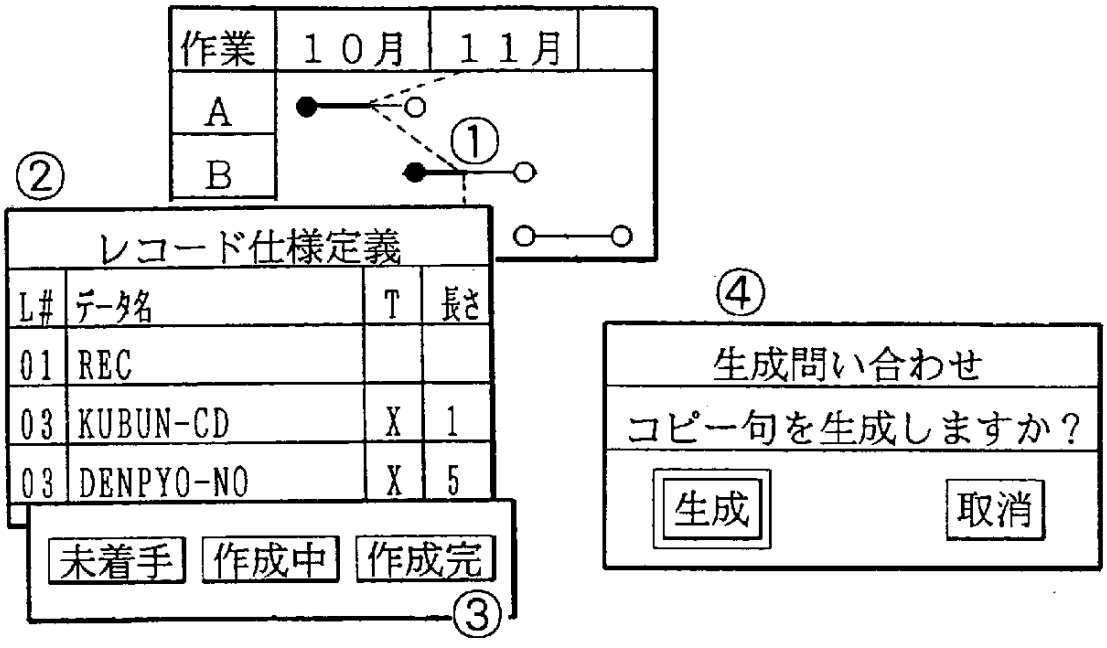


図6.2-3　作業誘導

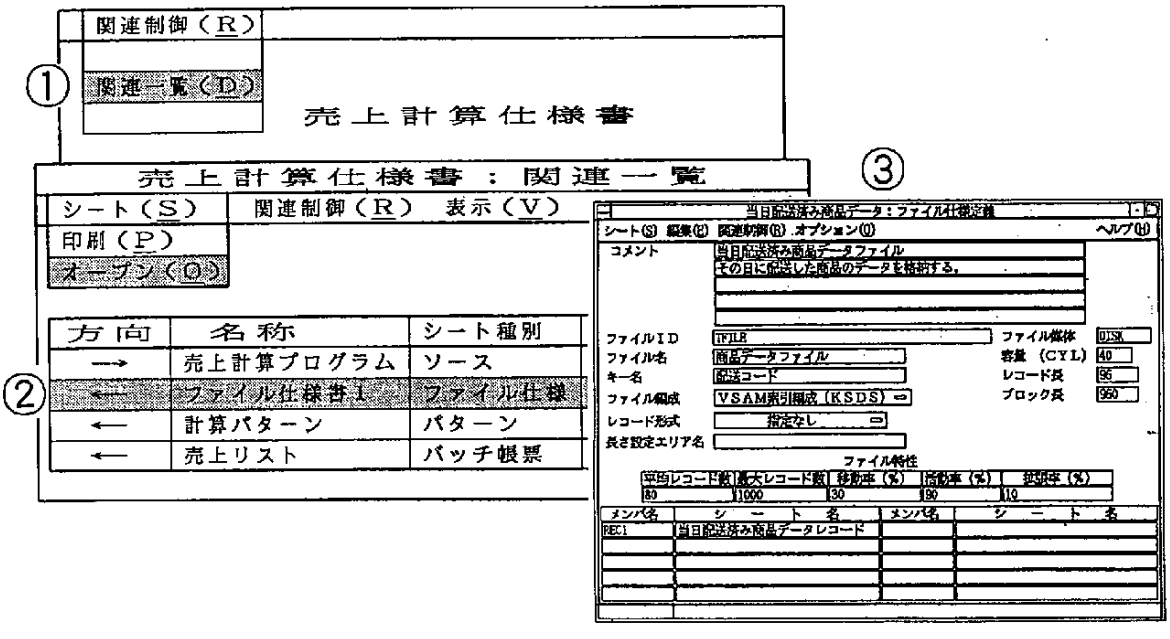


図6.2-4　関連情報の参照

6.2.4 関連情報の参照と変更通知

リポジトリでは仕様書は他の仕様書と関連を張って管理されている。関連種別として「参照」「生成」「変換」等がある。この関連により関連仕様書の参照、変更波及調査や変更通知を行う。関連付けは支援ツールにより自動的に行う方法と、CASEプラットフォームの登録機能により管理者が事前に関連付けをする方法、開発者が開発作業中に仕様書を選択して関連付けをする方法がある。

図6.2-4はプログラム仕様書作成中に関連するファイル仕様書を参照するため呼び出す例である。まず「売上計算仕様書」プログラム仕様書作成画面から関連一覧を選択する（①）。関連一覧が表示されたので関連仕様書の中から「ファイル仕様書1」を選択する（②）。関連一覧にある方向は仕様書間の方向で、「プログラム仕様書はファイル仕様書から作られる」という意味である。最後にファイル仕様書（③）が表示される。

変更通知の例を図6.2-5に示す。この例ではファイル仕様書を変更した作業者（発信者）が売上計算仕様書の担当者（受信者）に変更メッセージを送っている。発信者は作業終了時変更通知発信を指定し、メッセージ「利率カラムを追加・・・」を送信する（①）。リポジトリには関連情報が登録(②)しているので発信者は相手先を指定する必要はない。受信者は作業開始時変更メッセージの到着を知り、該当する仕様書に対する変更メッセージの詳細を読む（③）。つぎに変更通知に対する確認をする（④）。「対策要」を選択（⑤）したので、管理者は受信者が「対策は必要だが、まだ対策していない」状況であると認識する。このようにして管理者は仕様変更の状況が把握する。

以上CAPSDFのグループワーク支援機能を述べたが、この他にも分散開発拠点からのプロジェクト管理情報の収集機能、連絡票や作業報告書の管理支援機能等がある。

またコミュニケーション支援ツールの電子対話機能を用いた電子レビュー機能がある。DFDの画面を遠隔地のWS間で同期表示して業務仕様のレビューを実施したり、ソースプログラムとテスト結果を表示したリモートデバッグができる。

【参考文献】

[田村92] 田村和敏他: CSS統合開発環境(1)-概要-, 情報処理学会第４５回全国大会論文集, 情報処理学会, pp.5-339, 5-340(1992).

[築城92] 築城和康他: CSS統合開発環境(2)-リポジトリ-, 情報処理学会第４５回全国大会論文集, 情報処理学会,pp.5-341, 5-342(1992).

[末宗92] 末宗英利他: CSS統合開発環境(3)-プラットフォーム-, 情報処理学会第４５回全国大会論文集, 情報処理学会, pp.5-343, 5-344(1992).

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

[吉川92] 吉川彰弘他: CSS統合開発環境(4)-作業誘導-, 情報処理学会第４５回全国大会論文集, 情報処理学会, pp.5-345, 5-346(1992).

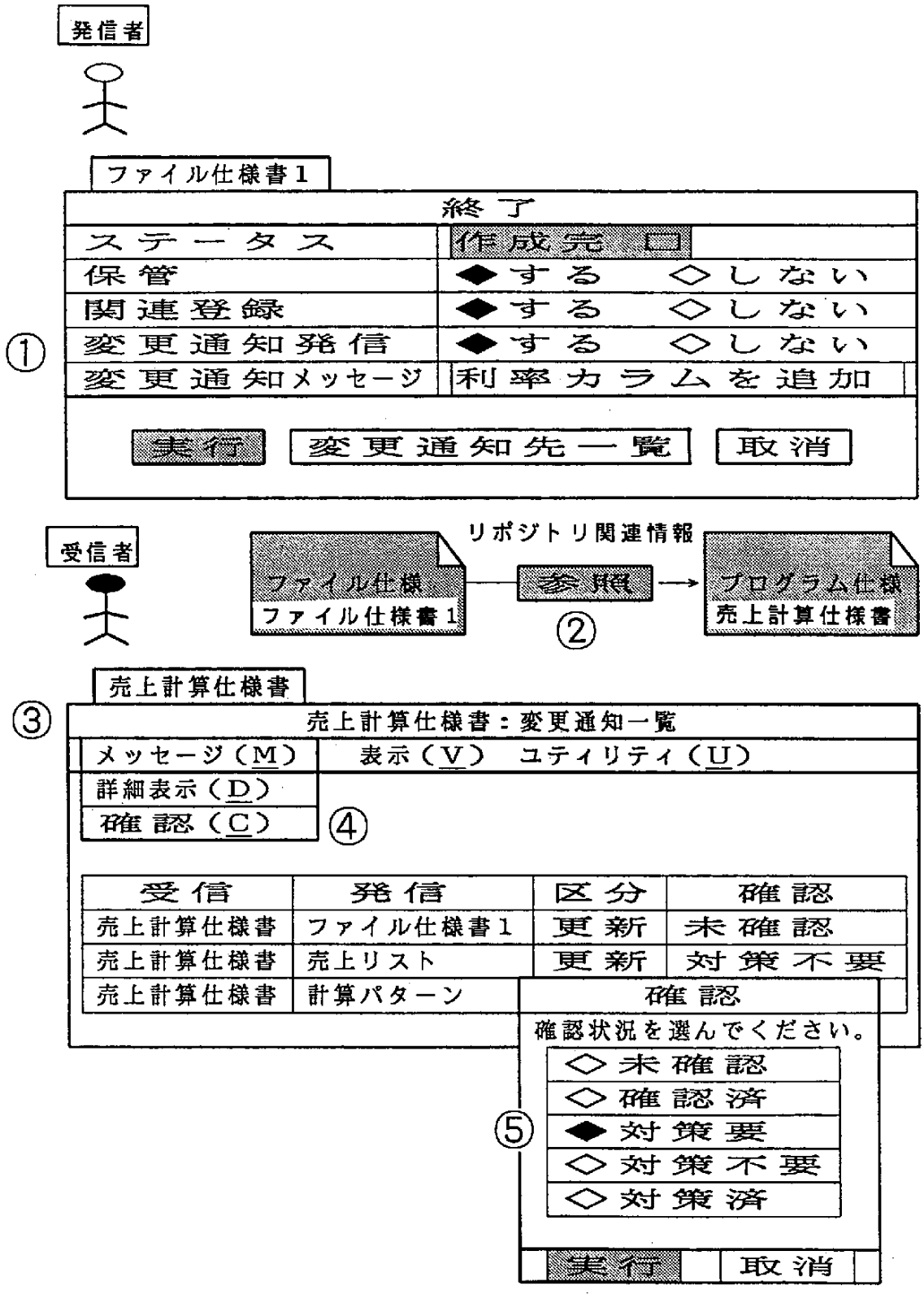

発信者
ファイル仕様書1
終了
ステータス
作成完
保管
する
しない
関連登録
する
しない
①
変更通知発信
する
しない
変更通知メッセージ
利率カラムを追加
実行
変更通知先一覧
取消
受信者
リポジトリ関連情報
ファイル仕様
ファイル仕様書1
参照
②
プログラム仕様
売上計算仕様書
③
売上計算仕様書
売上計算仕様書：変更通知一覧
メッセージ（M）
表示（V）
ユティリティ（U）
詳細表示（D）
確認（C）
④
受信
発信
区分
確認
売上計算仕様書
ファイル仕様書1
更新
未確認
売上計算仕様書
売上リスト
更新
対策不要
売上計算仕様書
計算パターン
確認
確認状況を選んでください。
未確認
確認済
⑤
対策要
対策不要
対策済
実行
取消

図6.2-5　変更通知

6.3 SDEワークベンチ/6000 および CMVC/6000 による分散ソフトウェア開発支援

ここでは、IBM[*1] RISCシステム/6000[*2]用のソフトウェア開発環境としての「SDEワークベンチ/6000」およびクロス・ライフ・サイクル・ツールである「コンフィギュレーション・バージョン管理/6000（CMVC/6000）」について紹介する。これらは、IBMのテクノロジー・フェア'92（6月にIBM箱崎事業所で開催）において、分散ソフトウェア開発技術として展示されていたものである[RISC92] [CASE92] [OPEN92]。

また、IBMのCASE関連の情報としては、昨年11月に新聞発表された、日本生命保険向け分散アプリケーションソフト開発支援システム「ゆとりのシステム」がある。このシステムは、CASEツール「HLL/WB VGA版」を採用したもので、利用者は、LANで接続されたノートパソコンを使い、ホスト抜きでソフトの設計、製作、テストができるものである[日刊工業92] [日経産業92] [電波92]。

6.3.1 SDEワークベンチ/6000

SDEワークベンチ/6000は、ソフトウェア・エンジニアリング環境を構築するための基本的な構築ブロックを持っており、ワークベンチのビジュアルおよびコントロール統合機能によって、新しいツールおよび既存のツールを比較的容易に、一貫性のあるユーザ・インタフェースを付加し、またカスタマイズして統合することができる。SDEワークベンチ/6000は、ヒューレット・パッカード（HP）からライセンスを取得したSoftBench[*3]フレームワーク・テクノロジーをベースにした、ビジュアル統合サービスおよびツール・コントロール・サービス、そしてメッセージ・サービスを提供している。また、これらの統合サービスと要素の定義には、ECMA[*4]のリファレンス・モデル（図6.3-1）が採用されている。

ユーザは、ビジュアル統合によってOSF/Motif[*5]ベースの一貫性のある外観、動作、起動操作方法を持った共通ユーザ・インタフェースをツールに付加することができ、コントロール統合によって、ツール間のコミュニケーションがメッセージ経由で行うことができる。これらの統合サービスの本質は、ブロードキャスト・メッセージ・サーバー（BMS）および実行マネージャーである（図6.3-2）。二つのツールの連携動作の例としては、ユーザがコンパイル・エラーのメッセージを選択すると、エディタが自動的に起動され、エラーのあった行にカーソルを自動的にセットしてくれる。

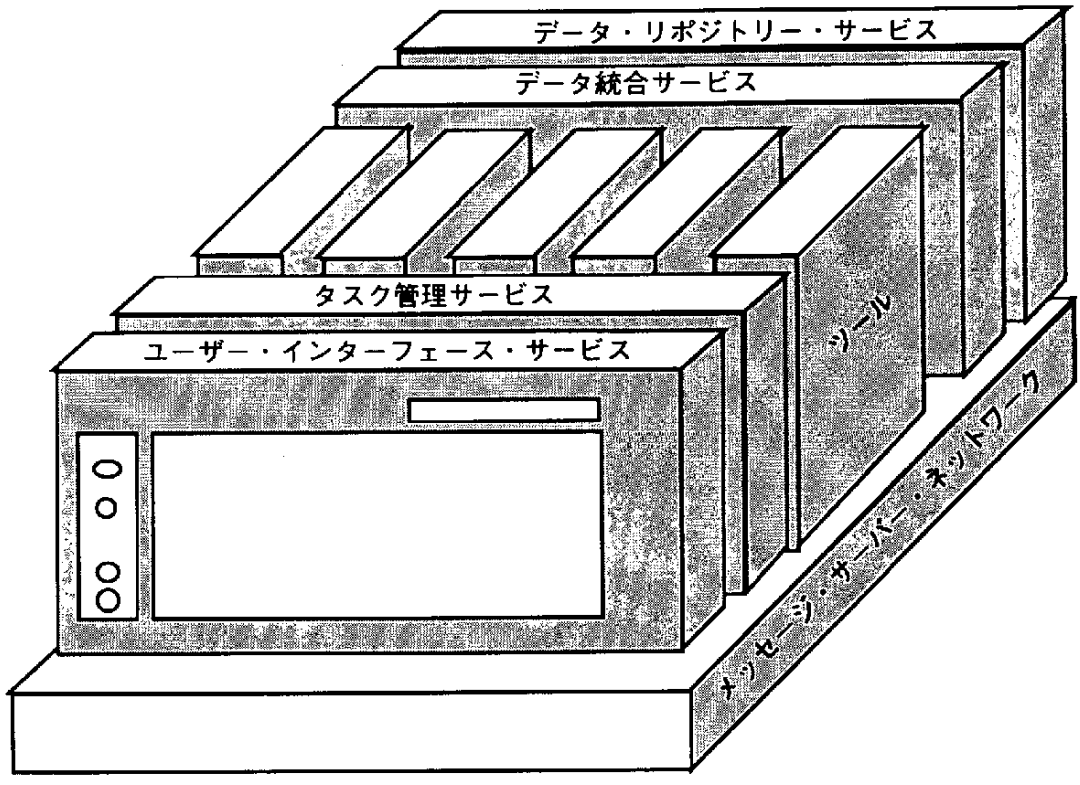


図6.3-1　ECMA リファレンス・モデル

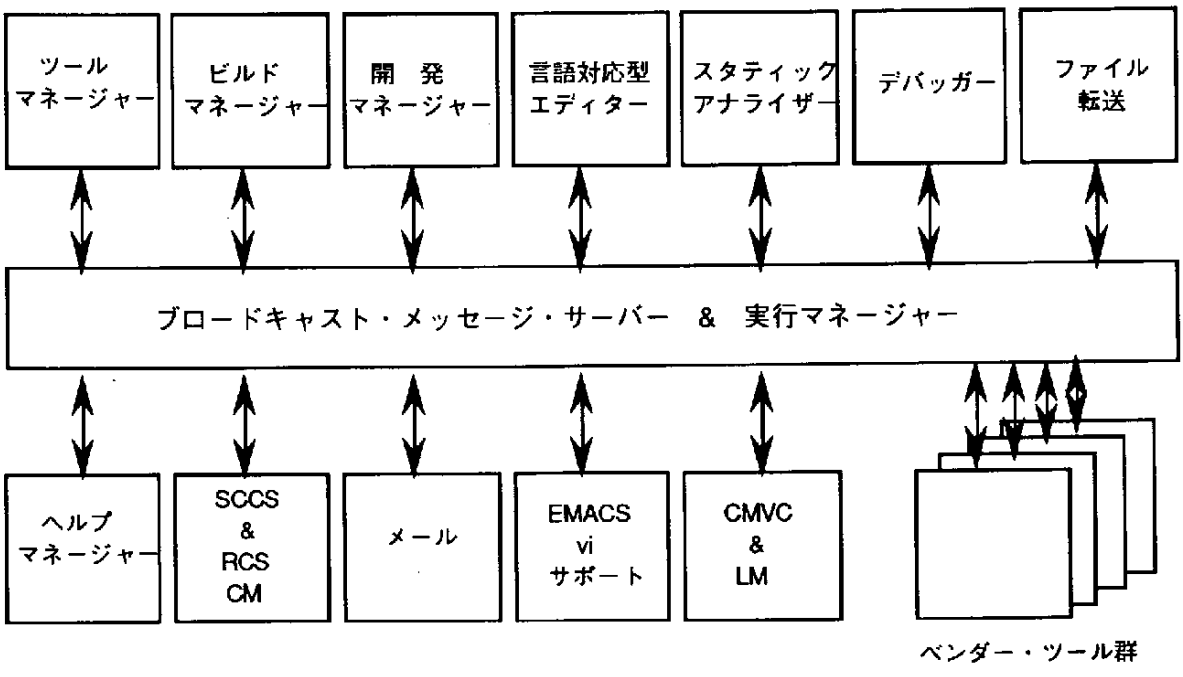


図6.3-2　SDEワークベンチ/6000 の構成要素

6.3.2 コンフィギュレーション・バージョン管理/6000（CMVC/6000）

CMVC/6000は、分散開発環境下におけるすべての開発成果物（ソース・コード、バイナリ・ファイル、ドキュメントなど）を、中央サーバ上で集中管理するもので、成果物への共有アクセスをサポートし、データのコンポーネントごとに柔軟なアクセス制御方法を提供する。CMVC/6000の変更管理機能によって、一時点でファイルを編集できる開発者は一人に制限され、複数のアプリケーションと実行環境において、あるモジュールの変更が及ぼす影響を追跡するメカニズムが提供される。また、通知機能によって、ある開発者によるエラー報告が、関係者に自動的に電子メールによって配布されるなどのプロジェクト・コミュニケーションがサポートされる（図6.3-3）。

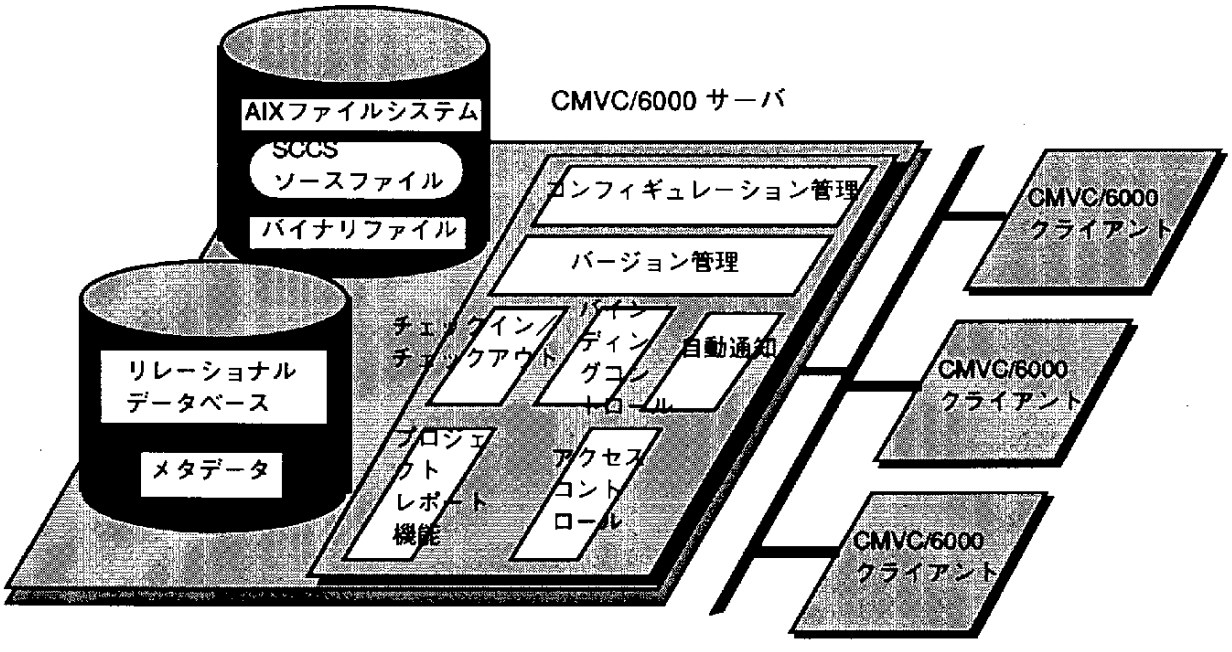


図6.3-3 CMVC/6000 概念図

*1： IBM は IBM Corp. の商標
*2： RISCシステム/6000 は IBM Corp. の商標
*3： SoftBench は Hewlett-Packard （米国）の商標
*4： ECMA は European Computer Manufacturors Association の商標
*5： OSF/Motif は Open Software Foundation, Inc. の商標

【参考文献】

[CASE92] IBM AIX CASE ソフトウェア開発ソリューション, N:G518-3762-00, N:Z218-3930-00(プレゼンテーション・ガイド), IBM Corp. (1992).

[OPEN92] IBMテクノロジー・フェア'92レポート, IBM OPEN Vol.1, N:GE18-9978-00, IBM Corp., pp.31-34 (1992.10).

[RISC92] RISC System/6000 POWER Frame5, N:G580-0010-00, IBM Corp. (1992).

[SDE92] IBM AIX SDE WorkBench/6000 User's Guide and Reference, SC09-1453-00, IBM Corp. (1992).

[日刊工業92] 日刊工業新聞 (1992.11.20).

[日経産業92] 日経産業新聞 (1992.11.20).

[電波92] 電波新聞 (1992.11.20).

7．グループウェアに関する調査

7．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国際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国際会議でも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国際会議が開催され、その後はCSCW会議と同様に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選びその概要を紹介する。

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。

なお、グループウェアという用語は、1978年にJonson-Lenzによって考案されたと言われ、CSCWのCSの部分、すなわちシステムとして具現化されたものを指す場合が多い。

- グループウェア全般（協調理論）
- 協調支援環境構築のためのアーキテクチャ
- 設計議論システム
- グループ意思決定支援システム
- 協調作業システムの設計支援
- オフィスプロシージャシステム
- グループエディタ

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、グループの協調作業において重要なウエイトを占める「調整」作業を構造化し、さまざまなグループウェアにあてはめた論文を紹介する。

② 協調支援環境構築のためのアーキテクチャ

ここでは、既存のグループウェアやアプリケーション・システムを統合し、グループ作業に対し広範な支援を提供できる環境を構築するためのアーキテクチャが対象となる。ここで紹介するアーキテクチャは、IGSE および Mocca である。

③ 設計議論システム

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を取り入れてプロセス情報を表現する汎用モデルを考案した論文を紹介する。

④ グループ意思決定支援システム（GDSS）

ここでは、グループウェアを研究する上で重要な要素となるグループ意思決定支援を支援するシステムに関する論文を紹介する。ミネソタ大学、アリゾナ大学、およびジョージ・メイソン大学において開発されたシステムを紹介する。

⑤ 協調作業システムの設計支援

ここでは、言語・行動理論をシステム分析の方法論に応用し、協同作業支援システムの要求仕様定義にあてはめた研究論文を紹介する。

⑥ 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、Action Workflow 、Oval および EuroCoOp である。

⑦ グループエディタ

ここでは、文書の協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、Flexible Diff および CoAUTHOR である。

7.1 協調理論

7.1.1 学際的調整理論

原文タイトル： Toward an Interdisciplinary Theory of Coordination

翻訳タイトル： 調整の学際的理論に向けて

著　者： Thomas W. Malone
Kevin Crowston
(Center for Coordination Science, Massachusetts Institute of Technology, Mass., USA)

出　典： MIT Technology Paper, January 1992

本稿では、調整の学際的研究に焦点を当てた「調整理論」と呼ばれる新しい研究分野が取り上げられている。この分野の研究は、コンピュータ・サイエンスや組織理論、オペレーションズ・リサーチ、生物学等の学問的分野の調整に関する考え方を利用、拡大したものである。

まず、この調整理論が定義され、つぎに、調整を分析するための枠組み（モデル化）が提案されている。本稿で示す枠組みでは、目標を達成するため独立した活動を実行するアクタの観点から調整を分析される。この観点からさまざまなプロセスが分析され、学問分野をまたがる共通点が特定される。分析したプロセスには、目標分解、資源配分、同期化、グループ意思決定、コミュニケーション、共通対象の認識等がある。

さらに、この調整理論を (1) 情報技術が人間組織と市場に与える影響の理解、(2) 協調作業ツールの設計、(3) 分散型並行処理コンピュータ・システムの設計、という三つの領域で応用した例を説明している。

最後に、この新しい分野で今後研究すべき項目が概観されている。

(1)　調整理論とは

①　「調整」の定義

「調整」という用語は既にさまざまに定義されているが、本稿では、まず次のように単純に定義されている。すなわち、「調整とは、共に作業する行為である」。これを調整の「広い」定義として、より限定的な「狭い」定義が後に示される。

②　「調整理論」の定義

本稿では、「調整理論」は活動の調整方法、すなわち、どうしたらアクタが共に作業できるかに関する原則と定義されている。さまざまな分野で、このような一般的な理論の開発に役立ったり、恩恵を受けることのできる理論や概念、研究結果はたくさんある。例えば、人間の活動調整方法に関する

問題は、組織理論、社会学、社会心理学、人類学、言語学、法学、政治学の中心的課題である。経済学や管理科学でも、合理的な資源配分に焦点を当てて、人間の作業調整方法を分析している。コンピュータ・サイエンスは人間を中心としたものではないが、異なるコンピュータによるプロセスには「協同作業」が必要であり、コンピュータ・プロセス間のある種のインタラクションは人間のインタラクションに似ている。

このように調整理論を上記のさまざまな学問分野に適用し、全体的な目標がどのように行動に分割されるか、行動がどのようにグループや個人のアクタに割当てられるか、異なる活動にどのように資源を配分するか、全体の目標を達成するため異なるアクタが情報を共有するにはどうしたらいいか等の観点から各学問分野を分析して、調整理論の汎用性を立証している。

(2) 調整理論の枠組み

① 調整の構成要素

前述の調整に関する広い定義から、調整を構成する要素として、「目標（ゴール）」、「活動（アクティビティ）」、「アクタ」、「相互依存」の四つが導き出される。これらの構成要素は全て、調整という点から状況を分析するために必要である。

調整を構成要素に分解することで、調整のもう一つの定義が浮び上がる。調整の広い定義に対して、「調整とは、活動間の相互依存を管理する行為である」という狭い定義付けを行うことができる。

② 基本的な調整プロセス

調整理論の中心となるのは、調整に含まれる基本的なプロセスの理解である。調整の四つの構成要素は、調整を決定又は管理するプロセスに関連する。

なお、この他の相互依存として、次の二つが考えられる。

- 包括的相互依存（例：二つのタスクの関係が、互いと重なるものである場合）
- 特定の相互依存（例：一つの企業の異なる部署の人間が同じ顧客を扱う場合）

③ 調整の基礎となるプロセス

さまざまな調整プロセスをより正確に特徴づけるためには、これらを階層をなす基礎的プロセスとしてとらえ、それぞれが下のレベルに依存すると考えることが有益である。すなわち、「調整プロセス」は、グループが意思決定を受入れることを必要とし、「グループによる意思決定」は、グループのメンバが、達成すべき目標や、代替案、代替案の評価、決定について何らかの方法でコミュニケートすることが必要である。さらに「コミュニケーション」には、送り手から受け手に対して、両者が理解可能な言語による何らかの形のメッセージが送られなければならない。最後に、この共通の言語とメッセージの送信は、究極的には、コンピュータ・データベースの共有状況や共有情報の物理的オブジェクトのような「共通のオブジェクトをアクタたちが認識する」能力にかかってくる。以上の

「調整」、「グループ意思決定」、「コミュニケーション」、「共通オブジェクトの認識」という四つの層は、ネットワークのプロトコル層のような、他のシステムの概念レベルに似ている。

層の順序は、各層が下の層のプロセスを利用するという考え方を反映している。例えば、グループ意思決定では、メンバが何らかの方法でコミュニケートできなければならない。しかし、ネットワーク・プロトコルとは異なり、ある層が上の層のプロセスを利用する場合もある。例えば、グループがコミュニケーションの共通言語の拡張のために意思決定プロセスを利用する場合や、グループが意思決定活動をアクタに割当てるため、調整プロセスを利用する場合等である。

④ 調整の階層構造の検証

調整の階層構造をさまざまなシステムに適用できることを検証するために、Malone と Smith の分析したタスク割当てプロセスを例にとって検討した。

この結果、調整理論のメカニズムを示すさまざまなモデルは、人間組織やコンピュータ・システムにあてはめた場合、多くの重要な側面が省略されるものの非常に広範囲な現象を明らかにできることが判明した。例えば、これらのモデルは、人間組織の設計に関する過去の理論や、人間組織やコンピュータ・システムの組織形態の歴史的変化と一致している。また、コンピュータ・システムの分散型スケジューリング・メカニズムの設計代替案の分析にも役立ち、新しい情報技術の組織への導入に関連する構造的変化を分析する方法も示唆している。

調整理論のコンセプトは、異なる学問分野のコンセプトや研究結果間の類似性の特定に役立つ。この類似性によって、学問分野の境界を越えてアイデアをやり取りする方法や、より深い分析が可能な部分を示唆できる。

(3) 調整理論の応用例

ここでは次の三つの分野に調整に関するコンセプトを応用した例について説明する。

① 情報技術が人間組織と市場に与える影響の理解

この例では、まず調整自体がコストと利益を持つ一つの活動であるという単純な観察から始め、情報技術がある種の調整コストを大きく削減することになる可能性を検討している。

この検討の結果、次の影響が示唆された。

まず、調整の「単位コスト」の全体的な削減によって、内部取引から市場取引への移行が促進されるというものである。これによって、縦の統合が減り、企業サイズは縮小することになる。

また、情報技術による調整コストの削減が意思決定の集中化に与える影響を分析した結果、情報技術は利用の仕方によって集中化にも分散化にもつながるという結論が得られた。情報技術と集中化に対する二つの傾向は、この問題に関する経験的研究と一致するものである。

さらに、組織の内部構造がより調整集約的になることも予想された。

② 協調作業ツールの設計

この例では、調整に関する考え方が、新しいシステムの提案やシステムの分類、利用方法に関する分析にどう役立ってきたかについて検討されている。

他の学問分野の調整に関するアイデアを利用して協調作業ツールを開発した研究例としては、(i)スピーチ・アクトに関する言語学と哲学のアイデアの利用、(ii)ブラックボードやアドホクラシに関する人工知能や組織理論のアイデアの利用、(iii)意思決定に関する哲学と修辞学のアイデアの利用、(iv)並行処理に関するコンピュータ・サイエンスのアイデアの利用、などの例が挙げられる。

協調作業ツールを設計する際には、このように調整理論を利用してそのシステム評価を行うことができる。

③ 分散並行型コンピュータ・システムの設計

この例では、調整の問題に対して、学際的方法により、他の学問分野やシステムの考え方を使って分散あるいは並行型コンピュータ・システムの設計や分析を行った例について検討されている。

特に、(i)設計アイデア源としての、社会的および生物学的システムとの類似性と、(ii)代替設計を分析するための定量的ツールの例が検討されている。

7.2 協調支援環境構築のためのアーキテクチャ

7.2.1 統合型グループ支援環境（IGSE）

原文タイトル： The Design of an Integrated Group Support Environment

翻訳タイトル： 統合型グループ支援環境の設計

著　者： Minder Chen
(Department of Decision Sciences, School of Business Administration, George Mason University, USA)
Yihwa Irene Liou
(Information & Quantitative Sciences Dept., Robert G. Merrick School of Business, University of Baltimore, USA)

出　典： Proceedings of the 24th. Annual Hawaii International Conference on System Sciences, Vol.IV, 1991, pp.333-342

本稿では、グループ作業に対し広範な支援を提供する統合型グループ支援環境（IGSE：Integrated Group Support Environment）のアーキテクチャと、その核となるグループウェア・ジェネレータの設計について紹介されている。

(1) 既存のグループ支援システムの特徴

IGSE はグループ支援システム（GSS：Group Support System）の概念的枠組みを基に開発されるものであるため、まず Information Lens、Colab、The Coordinator 等の既存の GSS についてレビューが行われている。

その結果、既存のグループ支援システムは、次のような特徴をもっている（あるいはもっていなければならない）ことが判明した。

① コンピュータ支援グループ手法を導入することによって、グループ・プロセスを支援すること。

② グループ情報を共有するためのグループ・リポジトリを開発すること。

③ 半構造化された情報の交換を通してグループ対話を促進すること。

(2) グループ支援システムの枠組み

つぎに、これらのGSS の技術的・行動学的観点をまとめるために、技術、グループ、タスク、プロセス、組織、環境の六つの要素からなるGSS の概念的枠組みが提案される。GSS の設計・使用に大きく関係するこれらの要素の属性について、以下に説明する。

① 技術

技術は、コンピュータ・システムと通信システムを統合したものである。技術の一例としては、通

信チャネルを介して伝送可能な媒体（例えば、テキスト、音声、画像、映像）を扱うことなどが挙げられる。GSS は、さまざまなレベルのコンピュータ支援（例えば、情報交換、意思決定モデリング、エキスパート・システム）を提供できる。GSS 技術を分類するために使われてきた属性としては、この他にメッセージの構造化等がある。

② グループ

グループは、グループ・プロセスに関与する個人の集まりである。グループのサイズと異質性は、タスクの複雑化に従い増大する傾向にある。グループは、そのライフ・サイクルでさまざまな段階を経る。

③ タスク

タスクは、グループが行う活動である。グループ・タスクの分類例としては、プランニング・タスク、創造的タスク、知的タスク、意思決定タスク、認知的衝突タスク、混合動機タスク、競争/闘争/競合タスク、パフォーマンス/精神運動性タスクの八つに分ける方法がある。これらの各カテゴリに対する支援は、変化することがありうる。グループ・タスクの他の特性としては、時宜性、複雑性、政治的影響、資源要件などがある。

④ プロセス

グループ・プロセスは、いくつかの要因による特徴づけが可能な一連のグループ活動である。ブレーンストーミング、ノミナル・グループ、デルファイ手法などのグループ手法は、グループ・プロセスの支援に使用できる。また、中心的なグループ活動は、コミュニケーション、意思決定、調整である。グループ・プロセスは、グループのライフ・サイクルを通してそのタスクを実行するために、グループが協調技術をどのように使用できるかを決定する場合がある。

⑤ 組織

GSS を採用する組織は、その組織の文化と構造、技術移転機構、既存のコンピュータと通信のインフラストラクチャ、業務の性質といった要因によって特徴づけることができる。これらの要因は、GSS の採用・導入に直接的・間接的に影響する。

⑥ 環境

世界的な競争、労働人口の調査、コンピュータと通信の規格、政府の規制、登場しつつある協調技術は、チーム/情報重視型組織へ移行しつつある企業の勢いを決定する可能性のある外的要因である。

(3) IGSE（統合型グループ支援環境）のアーキテクチャ

既存のほとんどのGSSは、各種のグループ・ツールを統合する機構を備えていない。したがって、ほとんどのグループ・ツールは、既存のアプリケーション・システムと統合しない。グループ・ツール同士を完全に統合することができないことさえある。

このような欠点を避けるため、IGSEは、次のような特性を持つことを前提とした。

- 対面形式の会議と非対面形式の会議の両方を支援する。
- 同期的な対話と非同期的な対話の両方を許容する。
- テキスト、グラフィック、画像、アニメーション、音声、映像を含む豊富な媒体をサポートする。
- システム機能を他のアプリケーションから呼び出すことができるように、アプリケーションプログラミングインタフェース（API）を備える。
- 特定の領域を支援するために環境を変更できるように、カスタマイズ機能を提供する。
- グループ・タスクの内容とプロセスを適切に構造化する。

これらの特性を持ったIGSEのアーキテクチャは、次の六つのサブシステムから構成される。

① グループウェア・ジェネレータ

グループウェア・ジェネレータは、IGSEの核となるシステムであり、(a)グループ・リポジトリの構造の定義、(b)リポジトリ内の情報にアクセスするGSSツール群の作成、(c)領域モデルに従ったグループ・プロセスの指定、をユーザが実行できるように設計される。ユーザはグループウェア・ジェネレータを使って、特定領域用にIGSEをカスタマイズすることができる。これは、IGSEの最も重要な特徴の一つである。

② グループ・リポジトリ

ER属性のメタ・メタ・モデルに基づくグループ・リポジトリは、SQL方式のデータベース・サーバの上で開発される。グループ・リポジトリは、グループがそのライフサイクルで必要とする情報の全てを格納し、グループウェア・ジェネレータが生成するグループウェア・ツール間のデータ統合サービスを提供する。

③ ユーザインタフェース・マネージャ

マイクロソフト社のWindow 3.0を使って、ユーザインタフェースを管理する。このインタフェースは、IBM社のSAAのユーザ・インタフェース規格に準拠する。IGSEはWindow下で動的データ交換（DDE）機構を利用して、既存のアプリケーションとの統合を促進する。

④ ツール・サーバ

グループ・ツール群は、ツール・サーバによって登録・管理される。これらのツールは、グループウェア・ジェネレータによって生成することができる。

⑤ グループ・プロセス・マネージャ

グループ・プロセス・マネージャ（GPM）を使うと、IGSEユーザは各種のアプリケーション領域についてプロセス・モデルを指定できる。プロセス・モデルの仕様では、一連の個人／グループ活動、これらの活動への参加者の役割、プロセスで使われるツールの呼出し条件と順番が定義される。GPM

はプロセス・モデル仕様を使って、グループ活動の調整、ツールの利用の管理、グループに対するポリシーの促進を実施する。

⑥ コミュニケーション・エージェント

IGSEのコミュニケーション層は、LAN ManagerやNovell社のNetWareなどのLANオペレーティング・システムをベースにしている。ネットワーク・オペレーティング・システムは、ネットワーク資源の共有、リモート・プロシージャ・コールの実行、蓄積交換メッセージ処理の管理などのサービスや、他のネットワークへの接続サービスを提供する。

(4) GSSジェネレータの設計

IGSEの核となるグループウェア・ジェネレータは、多種多様なアプリケーション領域を支援するグループウェア・ツールを生成しカスタマイズするための機構を提供する。グループウェア・ジェネレータを使うと、グループ・エンジニアは特定領域の環境で、いくつかの構造をGSS言語の形で定義することができる。GSS言語の処理可能性が高いほど、コンピュータはGSS言語を処理しやすくなる。GSS言語の表現力が高いほど、人間はGSS言語を使ってコミュニケートしやすくなる。

GSS言語によってユーザが指定できる構成体には次の6種類のものがある。

① リポジトリ

グループ・リポジトリは、グループ・プロセスが行われるコンテキストを維持するための機構を提供する。コンテキスト情報は、グループ履歴、タスク構造、進行中の会話の発展のパターンなどから構成される。これらのコンテキストにより、グループ・ツール間の結合と、アプリケーション・システムとの結合が実現される。コンテキストはまた、グループ・プロセス内の個人が実行する活動の意味づけも行う。グループ・リポジトリの構造は、一群のオブジェクト・タイプ、関係タイプ、属性タイプから構成され、これらのタイプにより、拡張されたER属性モデルを基づいて領域の知識が記述される。

② メッセージ

IGSEでのメッセージ構造は、どのツールとも独立に定義できる。メッセージ構造は、グループ・リポジトリのビューとして定義される。個々のグループウェア・ツールの構造と振る舞いは、それらのツールが受け取ったり処理したりできるメッセージに基づいて指定できる。ツール内でのメッセージ処理の振る舞いは、方法として定義される。

③ エージェント

グループ・プロセスに参加する可能性のあるエージェントには、人間、グループ・ツール、自律型知的システムなどがある。能力と、参加エージェントとコミュニケートするためのプロトコルは、グループ・リポジトリに指定しなければならない。各エージェントの能力に関する知識を使うと、シス

テムはエージェントにいつ相談するべきかを決定できる。

④ 対話

グループ・ツールの使用とグループ・ツール間の調整をコントロールするに際しては、エージェント間の対話のパターンと制約を指定しなければならない。対話パターンの例としては、サーバ対クライアント、ピア・トゥ・ピア、マスタ対スレーブ、ブロードキャスト、マルチキャストなどがある。

⑤ プロセス

全体的なグループ・プロセスの構造には、グループ・プロセスで実行される活動、グループ・メンバ間の対話パターン、これらの活動を支援するために呼び出されるツールなどがある。グループ・プロセス・マネージャ（GPM）はこれらの知識を使って、グループ・プロセス全体の管理を助ける。

⑥ 構造と目標

IGSE におけるグループの組織構造と目標、ならびにグループの相互関係を定義して、組織内の既存のアプリケーションに GSS を組み込むことができるようにしなければならない。

グループ・ツールの定義が済めば、グループ・ツールをツール・サーバに入力できる。ツール・サーバはツール仕様を解釈して、特定領域用のグループ・ツールを生成する。特定のアプリケーション領域でのグループ作業を支援するために、いくつかのグループ・ツールを生成できる。インフォメーション・リソース・ディクショナリ・システム（IRDS)規格に基づく3層のリポジトリ・システムが、組織メモリ、グループ・メモリ、個人メモリを支援するために定義される。このリポジトリ・システムは、各種のグループ・ツールの統合に使用できる。IRDS は組織メモリについて使われる。グループ・メモリは、組織メモリのサブセットから構成でき、個人メモリは、グループ・メモリと組織メモリの両方から構成できる。IRDS は組織の情報資源を統合する機構になる可能性があるため、IRDS 規格に準拠してグループ・リポジトリを構築すれば、IGSE を全社的な情報システムへ発展させたり、全社的な情報システムと統合させたりすることができる。

(5) IGSEのインプリメント

IGSE はまだ設計段階であり、ハイパーテキスト・システムを使ってプロトタイプが開発されている。プロトタイプ・システムの開発に使われているハイパーテキスト・システムはToolBook で、マイクロソフト社の Window 3.0 下で動く。ハイパーテキスト・システムを使うと、テキスト、グラフィック、画像、映像を容易に IGSE に取り込むことができる。ToolBook は、ウィンドウ方式のアプリケーションを構築するための高度な機能を各種備えている。ToolBook はまた、より複雑なシステムを構築するために、OpenScript と呼ばれるスクリプト言語を備えている。この OpenScript からは、動的リンク・ライブラリ（DLL）の機能を呼び出すことができる。外部データベースや通信システムなどの他のア

プリケーションとのインタフェースは、Windows DLL を使って開発できる。また、マイクロソフト社の Window の DDE（動的データ交換）を使うと、ユーザは他の Windows アプリケーションと通信したり、アプリケーション間で情報を共有することができる。

本プロジェクトの最初のフェーズでは、次のようなグループ・ツールが開発され配付された。

① ウィンドウ方式の構造化ブレーンストーミング・ツール

創造的な思考ができるように、ユーザを支援する。

② 複数基準の意思決定ツール

いくつかの重みつき基準をベースに、グループが代案を評価できるようにする。

③ 投票ツール

投票結果を分析し、グラフの形で示す。

④ 電子アンケート・ツール

デルファイ方法の支援に使用できる。

⑤ ハイパーテキスト方式のグループ用語集とグループ・リポジトリ。

⑥ IGSE のプロトタイプ。

(6) IGSEの対象アプリケーション領域

IGSE によって支援される対象アプリケーション領域が例示され、これらの領域に対するプロセス・モデルが示される。ここでは、ジョイント・アプリケーション・デザイン、戦略的プランニングと政策決定、衝突の解決、複数基準意思決定（MCDM）、複数のエキスパートからの知識獲得、遠隔学習の 6 種類の領域が選ばれている。

7.2.2 Mocca プロジェクト

原文タイトル： Mocca: An Environment for CSCW Applications

翻訳タイトル： Mocca：CSCWアプリケーションのための環境

著　者： Wolfgang Prinz
(GMD, Germany)
Tom Rodden
John Mariani
(Lancaster University, UK)
Torbjorn Naslund
(Linkoping University, Sweden)
Elsa Bignoli
(Milan University, Italy)
Steve Benford
(Nottingham University, UK)
Carl Brown
(SICS, Sweden)
Leandro Navarro
(UPC, Spain)

出　典： ACM SIGOIS Bulletin, Vol.13, No.1, April 1992, pp.21-23

本稿では、各種CSCWアプリケーションを一緒に機能させることができる環境の開発を研究しているMoccaプロジェクトが紹介されている。本プロジェクトには、いくつかのヨーロッパの研究機関が参加し、それぞれ協同作業というものを異なった見地から支援するCSCWアプリケーションの開発に関わっている。

(1) Mocca の研究アプローチ

CSCWアプリケーションには、各CSCWシステムが互いに調和を取って機能することができることを保証するものがあるが、CSCW環境の役割は、そうしたさまざまなCSCWアプリケーション間の相互操作性を提供することである。特に、コンピュータ・ベースの環境で異なるアクティビティを確立・実行するための一連の支援サービスを提供することが必要である。

組織的関係の中でアクティビティを管理する環境の開発は、協同作業技術に対しても重要な問題となる。こうした環境の開発における中心テーマは、可変性（tailorability）の概念である。すなわち、これらの環境は支援する実際の事例に合わせて変更・調整できるようにするということを想定している。

(2) Mocca のモデル

本プロジェクトは、現在、CSCWシステムの中心にある知識が、その環境の中で表現できるように

するモデル群を作り上げて、移植しているところである。これらのモデルが直接CSCWシステムをサポートし、いくつもの異なるCSCWシステム間の相互動作ができるようになることを目標としている。現在、プロジェクトはいくつかのモデルに分かれて開発中である。

また、本プロジェクトの一部として、ユーザが探求することのできる「部屋」の概念に基づいた実世界のメタファが進行中である。このメタファの目標は、グループの相互動作を簡単にMocca環境に移行させることができ、Mocca環境が提供するいくつもの異なるモデルによってサポートされるようにすることである。部屋の概念は、Mocca環境のなかで各モデルの役割を検討するために簡単に使える機構を提供する。

これらの異なるモデルはオブジェクト指向によって表現されている。各モデル毎に基本概念を識別し、これらの概念をとらえたオブジェクトのセットがどこで使えるかを定義する。これらのオブジェクトが、将来のCSCWシステム開発にとって、一種の「協同ツールボックス」を形成することをねらっている。

現在分析中のモデルとしては、次の5種類のものがある。

① 相互アクティビティ・モデル

本モデルの目的は、アクティビティと役割の表現に対して共通の機構を見つけるというより、異なるアクティビティと役割間の依存関係を表現するようにするというものである。

② 情報モデル

本モデルは、異なるCSCWシステムで使われた情報を外部的に表現して、システム間で共有しようというものである。モデルは、情報オブジェクト、情報オブジェクト間の関係（構成物、依存関係など）、情報オブジェクトへのアクセスという見方で表される。

③ 組織モデル

Moccaプロジェクトの開発の動機の中心は、CSCWシステムの成功にとって組織的なアプローチが重要であるという認識である。組織モデルの目標は、組織的資源・方針・規則の共有を明確にすることである。モデルは、組織オブジェクト（資源、プロジェクト、人、役割など）、組織関係、および規則のセットから構成される。

④ コミュニケーション・モデル

本モデルは、コミュニケータが交換する情報オブジェクト、コミュニケーションが発生する状況といった概念を使ってコミュニケーションを表わすことを目標とする。

⑤ ユーザ熟練モデル

本モデルの目的は、その環境や他のシステムで利用できるようにユーザの熟練度を表すことである。このモデルは、その組織によって与えられた「ユーザ責任」、ユーザ個々の能力を表す「ユーザ能力」という概念で表される。

7.3 設計議論モデル

7.3.1 IBISを応用した設計議論モデル

原文タイトル：	A Generic Model for Representing Design Methods
翻訳タイトル：	設計方法を表現するための汎用モデル
著　者：	Colin Potts (MCC, Software Technology Program, Austin, Texas, USA)
出　典：	Proceedings of the 11th. International Conference on Software Engineering, IEEE, 1989, pp. 217-226

本稿では、プロセス情報を表現するパラダイムについて述べられている。まず、汎用モデルを提示し、特定の設計手法についてこのモデルがどうカスタマイズされるかを示し、カスタマイズされたモデルを典型的なプロジェクトでどのように例示できるかという順に説明されている。

(1) 汎用モデルのスキーマ

ここでは、汎用スキーマを記述するために次のようなER属性データ・モデルが採用されている。

① エンティティ・クラス

基本モデルはできるだけ汎用性のあるものでなければならないという目標のため、基本クラスは非常に一般的である。設計状況の情報は、「成果物（artifact）」に格納される。改善と改訂は、「設計ステップ（step）」を実行することによって達成され、設計決定の原因となる推論は、「問題点(issue)」、「解決案（position）」、「議論（argument）」によって表現される。モデルで使われるエンティティ・タイプは、この五つしかない。特定手法用のオブジェクトはすべて、これら五つのエンティティ・タイプを特殊化したものである。

(a) 成果物

成果物は、特定の手法用の「文書（document）」である。すべての成果物が、テキスト形式である必要はない。また、品質保証目的の場合は、「文書」として認識可能である必要もない。

(b) 設計ステップ

設計ステップは、成果物を作成／修正する。例えば、データ・フロー図のプロセスを別のデータ・フロー図に分解することは、設計ステップである。設計ステップが作成する新しい成果物は、その成果物の元になる成果物と手法に特有な関係をもつ。プロセスの操作的仕様（operational specification）に存在するバグを訂正することも、設計ステップである。元の成果物を変更すると、新しいバージョンが作成される。

(c) 設計推論要素（問題点、解決案、議論）

設計の推論と論理的根拠を「問題点」、「解決案」、「議論」で表現するという考え方は、Rittelの研究によるIBIS（Issue-based Information Systems）に基づいている。

「問題点」は重要な焦点に関しての質問を出し、「解決案」は「問題点」に対する候補応答である。また、「議論」は「解決案」を支援したり「解決案」に反対したりする意見である。決定は、「解決案」の「選択」属性が真になったとき行われる。「問題点」は、それに応答する選択された「解決案」を持つ場合にだけ解決される。

② エンティティ・タイプ間の関係

五つのエンティティ・タイプの間には、次の８種類の二項関係がある。

(a) 修正

「ステップ」は「成果物」を修正する（modify）。最初の場合には新しい「成果物」が作成され、改訂（revision）の場合には既存の「成果物」の新バージョンが作成される。

(b) 問題提起

「ステップ」は「問題点」を提起する（raise）。「ステップ」を実行すると、自動的に「問題点」が提起されることがある。

(c) 見直し

「問題点」は「成果物」を見直す（review）。「問題点」は、「成果物」の特性を見直すために提起されることがある。

(d) 応答

「解決案」は「問題点」に応答する（respond）。

(e) 支持

「議論」は「解決案」を支持する（support）。

(f) 反対

「議論」は「解決案」に反対する（object to）。二つの「解決案」が「問題点」に応答し、そのうちの一方を支援する「議論」が他方に反対するというケースがしばしば起こる。

(g) 言及

「議論」は「成果物」に言及する（cite）。「成果物」は「議論」に証拠を提供する。

(h) 寄与

「解決案」は「ステップ」に寄与する（contribute to）。「ステップ」が実行されるのは、一群のコミットメント（選択された「解決案」）が行われたためである。

③ 階層化による汎用スキーマのカスタマイズ

汎用スキーマのカスタマイズは、エンティティ・クラスのカスタマイズによって達成される。設計

手法のエンティティ・クラスは、「成果物」クラスについて一つ、「問題点」クラスについて一つといった具合に、一群のクラス階層を形成する。各クラスは、その先祖クラスの関係を継承する。

(2) JSDの分析

ここでは、汎用モデルはJSDについてどのようにカスタマイズされるかが述べられている。まず、Jacksonの考案した手法であるJSDの概要が紹介される。つぎに、汎用モデルのプリミティブを使うと、表現手法、プロジェクトの戦略的順序付け、JSDのヒューリスティックスをどのように表現できるかについて説明される。この説明では例として、Jacksonが示したエレベータ・システムを使っている。汎用モデルのアプリケーションを説明してから、それが同じ例についてどのように特殊化されるかが示されている。

(3) パラダイムの妥当性の考察

五つの基本的エンティティ・クラスだけを使って汎用スキーマを設計することの妥当性を以下のように考察する。

① 成果物について

成果物は、設計情報の粗粒子的な表現である。例えば、JSDエンティティ-アクション・リストは事実上は、次のような主張を合わせたものである。

- さまざまな名前付きオブジェクトはエンティティである。
- その他はアクションである。
- アクションの一部は特定のエンティティによって実行されたり被害を受けたりする。

これらの主張の各々は、設計記述の「原子」として扱うことができる。例えば、アクションの削除は、それはアクションであるという主張と、どのエンティティがそれに含まれているかを明言する主張も削除する。成果物は、部分的に冗長なことがある。複数の人間のプロジェクトでは、冗長性は不可避であり、一部の初期設計活動／要求分析では冗長性が奨励されることさえある。さらに、成果物の生成と修正は、ほとんどの設計手法の主なフェーズに対応する。

② 設計ステップについて

ステップは、成果物を変更する原始的な操作である。しかしながら、導出（derivation）と改訂（revision）を区別するほうが良いと考えられる。導出では新しい成果物が生成されるのに対し、改訂では新しい成果物はそれに先行する成果物に取って代わる。基本的なデータ・モデルがバージョンを支援する場合には、改訂と導出を別個にモデル化する方が実現可能性が高いと考えられる。

③ IBIS モデルについて

設計の論理的根拠の表現と手法のヒューリスティックスの適用とに目を向けると、IBIS モデルは表

現の点で妥当である。IBISは、その解決のための十分に定義された判断基準がない問題の調査を支援するために開発された。多くのソフトウェア設計は、このような問題から構成されている。しかし、十分に定義された判断基準がある単純な推論についても、適切に特殊化すればIBISは有用である。

④ 今後強化すべき機能について

ステップが、成果物の変形とみなされるという特徴により、汎用モデルは現在のところは小規模のプロジェクトに制限されている。設計者やチームの数、自主性、責任は、対処されない。大規模なプロジェクトでは、ある時点で一元化しなければならない並行的な開発ストリームをモデル化することが必要になる。そのためには、「役割（role）」か「エージェント（agent）」エンティティ・タイプが必要になる。

また、汎用スキーマで使われるデータ・モデルは、一般化と特殊化で強化された二項ER属性モデルであるが、集合体を取り込むことも必要になると考えられる。エンティティ・インスタンスが別のエンティティ・インスタンスを含むことは、現在のところできない。

必要になるかもしれない他の強化機能としては、バージョン付けがある。バージョンを並列の異形（variant）も含めて、データ・モデル自体で支援するというアプローチが考えられる。

現在のところは、ステップの開始と終了の決定は設計者に任されている。ただし、一部の手法の一部のステップは、成果物の形式的な分析によって決定可能な条件によってトリガーされる。多くのステップは、前条件と後条件のペアによって指定可能であり、結果の成果物の分析によりステップの完了を決定できる。したがって、一部の状況では、「問題点」のタイプが提起される条件と、ステップを達成できる合法的な変形を形式的に指定することが可能となる。

7.4 グループ意思決定支援システム

7.4.1 SAMMの利用調査

原文タイトル： Using Computing to Improve the Quality Team Process: Some Initial Observations from the IRS-Minnesota Project

翻訳タイトル： 品質改善チームのプロセスの改善にコンピュータを使う：IRS - ミネソタ・プロジェクトの初期調査結果

著　者： Gerardine DeSanctis
Marshall Scott Poole
(University of Minnesota, Minneapolis, MN, USA)
Howard Lewis
George Desharnais
(Internal Revenue Service, New York, NY, USA)

出　典： Proceedings of the 24th. Annual Hawaii International Conference on System Sciences, 1991, pp.750-757

本稿では、小グループ会議におけるGDSSの利用に関するIRS（米国国税局）とミネソタ大学の共同調査研究の一環として行われた、IRSマンハッタン地区オフィスでの品質改善チームによるGDSS利用の範囲とタイプについての調査結果の一部について報告されている。

(1) 調査の背景

① IRS／NTEU共同品質改善プロセス

IRS／NTEUの共同品質改善プロセスは、IRSの内外の業務の成果の品質を改善するために、NTEUの組合員とIRSの管理者が積極的に協力するプロセスである。このプロセスは、小グループによる問題解決アプローチを適用して、積極的な組織変革を実現する。8～10人のボランティア（労働組合員と管理職の両方を含む）から構成された、多様な職務を含むチームが新たに編成され、6ヵ月から2年の期間に渡って、作業者のモラル、人事政策、帳票処理、コンピュータ・システム管理などの組織関連問題の一つに取り組む。チームは数週間ごとに約2時間の会議をもち、そこで品質・効率・作業環境の改善方法を見出し実施する。意思決定は、コンセンサスによって行われる。各チームにはリーダの他に外部のファシリテータも一人いて、会議の最中にグループに対しフィードバックや支援を行う。

② グループの意思決定支援システム（SAMM）

IRSでは、グループの意思決定を支援するシステムとしてSAMM（Software Aided Meeting Management）システムが選ばれた。

SAMMは、小規模の対面形式の会議用に設計されたマルチユーザシステムで、議題の設定、メモ取

り、問題の定式化、アイデアの生成、解決策の案出・評価、プランニング、交渉などの活動を支援する。SAMM を構成するメニュー方式のモジュール群は、IRS 品質改善プロセスで使われるようなグループ問題解決手法の実施を促進する。積極的な参加とコンセンサスをベースにした意思決定を推奨するために、アイデアの無記名記録・表示、数字による意見投票、系統的なアイデア縮小・優先順位付けなどの機能を提供する。

③ 調査の概要

IRS マンハッタン地区の共同品質改善プロセスとミネソタ大学は、1989 年 1 月に研究プロジェクトを開始した。この研究には、既存の 5 チームと新たに編成された 5 チームが参加している。新たに編成された 5 チームは、チーム発足時から SAMM を利用可能な状態にある。既存の 5 チームのうちの 2 チームに対しては、品質改善プロセスへの参加の最後の数ヵ月の期間に SAMM が導入され、残りの 3 チームに対してはその存続期間の中程で SAMM が導入された。どのチームも、研究へ参加を開始してからは、SAMM を利用した定期会議に出席している。この10 チームのほかに、マンハッタン地区の他の品質改善チームや、GDSS 施設の利用に関心をもつスタッフ・チームに対しても、SAMM は開放された。

本稿で報告される分析はすべて、1989 年 9 月から1990 年 3 月までの期間に収集されたデータに基づいている。分析はコンピュータ・システム利用ログ、ビデオに録画したサンプル会議の観察、この技術に対する品質改善チームの満足度を調査するために会議後に実施されたアンケートを基にしている。

(2) 調査結果

① システムの利用頻度

1990 年 3 月現在でのシステム・ログ記録が示すところによれば、IRS に設置されて以来、SAMM は 225 会議で使用され、会議の所要時間は合計 600 時間、参加者は合計 250 人であった。今回の分析が対象にした 7 ヵ月の期間におけるシステム利用は、平均 20.6 会議／月であった。このうち、約 7.8 会議／月 (38%) は、本調査に参加した10 の品質改善チームによって開催された。3.4 会議／月 (17%) は、SAMM を 2 会議以上に使い、会議ログ・ファイルの記録を許可したスタッフ・チームによって開催された。7 ヵ月の期間中に開催された残る 45% の会議は、ログ・ファイルを記録しなかった。この中には、デモンストレーションや、1 回限りのスタッフ・チーム／品質改善チームの会議がある。

調査の結果、SAMM の導入の時機が、チームの会議で SAMM が利用される可能性に大きく影響することをが明らかになった。品質改善プロセスの作業の最終段階で SAMM を導入されたチームは、会議でSAMM を利用する傾向が低く、利用した場合には、基本的なシステム機能の利用にとどまる傾向にあった。これに対し、作業の初期段階や中間段階で SAMM を導入されたチームは、SAMM を利用する傾向がずっと高く、しかも上級のシステム機能を利用する傾向が見られた。

新しい品質改善チームと同様に、スタッフ・チームも、基本的なシステム機能の利用に加えて、上級の機能をしばしば利用した。スタッフ・チームには新しいチームは少なかったが（つまり、メンバはSAMMの利用を始めるずっと前から、一緒に働いていた）、SAMMを利用した彼らの会議にはほとんどの場合、特別のSAMMファシリテータが参加した。この点が、品質改善チームと違っていた。今回のデータから推測される点は、新たに編成されたチームほど上級のシステム機能をあえて使う傾向にあることと、編成後かなり経つ安定した作業グループでSAMMが利用される場合には、上級のシステム機能の利用を促進するための特別のファシリテータが必要になることである。

② SAMMに対する満足度（利用の効果）

SAMMの支援を受けた会議の終了直後に実施されたシステム応答アンケートの結果は、(i)チームは非常に快適に、楽しんでSAMMを利用したこと、(ii)SAMMが品質改善会議に対し「適切な支援」を提供したのでチームが満足したこと、(iii)しかし、その一方で、チームは、より効率良くSAMMを利用するための特別の訓練の必要性を感じていること、を示している。これらの三つの尺度の平均スコアは、それぞれ（7段階評価で）5.50、5.29、2.64だった。

12のサンプル会議に立ち会ったオブザーバの評価はこれよりも少し低かったが、三つの尺度に対し同じパターンを示した。平均スコアは、それぞれ4.25、4.25、2.80だった。

③ 利用の提案

チームでのSAMMの利用をうまく提案できたのは誰かという点について12のサンプル会議を詳細に分析した結果、権威をもつ人間（グループ・リーダとファシリテータ）とチーム・メンバのどちらも、この点に関してはうまくいったことが判明した。分析の結果、会議は次の三つのタイプに分類できるように思われた。(i)チーム・メンバによる利用提案とリーダ／ファシリテータによる利用提案のバランスがとれている会議、(ii)ほとんどの利用提案をチーム・メンバが出した会議、(iii)ほとんどの利用提案をリーダ／ファシリテータが出した会議、の三つのタイプである。ほとんどの会議が、第一のタイプに分類された。利用を提案する可能性は、どの段階でチームにSAMMが導入されたかとは無関係だった。すなわち、三つのタイプのどれについても、SAMMの利用開始時期は初期段階、中間段階、最終段階が混在していた。

12の会議について平均した場合、メンバ提案による利用のほうがリーダ／ファシリテータ提案による利用より多かった。SAMMはチーム・メンバによる運用を目的に設計されていることを考えると、ほとんどのSAMM利用がグループのメンバによって提案されたという事実は望ましいことと思われる。

④ 利用の目的

12のサンプル会議でSAMMがどのように利用されたかについては、全体的に見ると、チームは主にタスクの目的にSAMMを利用したことが分かった。すなわち、まず、会議で何を達成したいかを見

出してから、支援を受けるためにSAMMを利用した。次に利用頻度が高かったのが、プロセス利用である。タスクを達成することとチームに対し会議プロセスを提供することが主なSAMM利用だったことは、望ましい発見である。すなわち、全般的に見て、SAMMは玩具として使われたり、混乱の元になったり、個人が「周囲に注意を払わずに」自分の作業をするために使われたりすることはなかった。娯楽／調査タイプと混乱タイプの利用は頻度が低く、少数の会議に集中していた。

権力行使目的の利用は比較的多く、割合均一に分散していた。表面的に見れば、権力行使目的の利用が多いのは憂慮すべきだといえるかもしれない。SAMMは、一個人に不当な影響力を与えるためではなく、グループにおける民主主義と参加の促進を目的に設計されたからである。しかし、他人へ影響を与えることを目的としたSAMMの利用を詳細に検討すると、ほとんどのケースでは肯定的な利用だったことが分かる。すなわち、特定のタスク活動へチームを引っ張っていくためや、脱線したチームを元に戻すためや、チームを議論から活動へ誘導するために、一個人によってSAMMが使われたのである。

12のサンプル会議の中で、SAMMを親睦の目的に利用した会議は一つだけだった。SAMMそれ自体は親睦のためのツールとしては設計されていないが、からかったり、冗談をいったり、愛情を交換したりする行為は、チームの一体化に役立つ社会感情的な「接着剤」になる。SAMMにはこの目的に使用可能なコメント機能があるが、これらが使われることはめったになかった。コメント機能はたいていは、親睦の目的ではなく権力行使目的に使われた。

また、前述の三つの会議タイプと利用の回数／タイプとの関係について次の表に示す。

会議タイプ	1会議当たりの平均利用回数
1	4.5（中程度）
2	9.5（多い）
3	2.5（少ない）

主にメンバの提案によってSAMMを利用したチームが、SAMMの利用回数が平均して最も多かった（第2タイプ）。メンバの提案による利用とリーダ／ファシリテータの提案による利用がバランスしていたチームは、中程度の利用回数だった（第1タイプ）。主にリーダ／ファシリテータの提案で利用したチームは、利用回数が最も少なかった（第3タイプ）。したがって、SAMMの利用提案者と、SAMMの利用回数との間には関係があるように見える。

⑤ 利用の感想

全体的に見ると、12のサンプル会議の最中に表明されたSAMMに関する意見は、肯定的なものと否定的なものがほぼ同数だった。次の三つのチーム群と、SAMMの利用に関する意見のタイプとの間には、多少の関係があるように見える。まず群1は、SAMM利用回数が中程度だったチームから構成される群で、メンバによるSAMM利用提案とリーダ／ファシリテータによる利用提案の回数のバラ

ンスがとれているが、この群については、SAMM利用に関する意見は肯定的なものより否定的なものが多いという傾向が見られた。興味深い点は、オブザーバは、SAMM利用の快適さと楽しみの尺度について、この群にかなり高い点を付けたことである。また群2は、SAMM利用回数が多いチームから構成される群で、利用提案はほとんどメンバが出しているが、この群のSAMMに関する意見は、肯定的なものが多いか、あるいは肯定的と否定的のバランスがとれているという傾向にあった。この群に属するチームは、会議の最中にSAMMから「適切な支援」を受けているとして、オブザーバから最高の点を得た。群3は、SAMMの利用回数が少なく、リーダ/ファシリテータだけが利用を提案した群だが、SAMM利用に関する肯定的な意見はこの群が最も多かった。

7.4.2 EMS

原文タイトル： Electronic meeting support: the GroupSystems concept

翻訳タイトル： 電子会議支援：「グループシステム」コンセプト

著　者： Joseph S. Valacich
(Decision and Information systems Dept., Scool of Business, Indiana University, IN, USA)
Alan R. Dennis
J. F. Nunamaker Jr.
(Arizona University, AZ, USA)

出　典： International Journal of Man Machine Studies, 1991, pp.133-154

本稿では、アリゾナ大学において開発された「グループシステム」と呼ばれる電子会議システム（EMS: Electronic Meeting System）の技術と実験評価について紹介されている。まず、「グループシステム」の開発の動機が説明され、さらにどのようにその設備とソフトと手順の統合が会議を改善しているかが説明される。つぎに、アリゾナ大学がこれまでに実施した大学内での実験や、学外の現場での実施結果の評価がまとめられている。最後に、これらの技術開発と利用と評価から得た経験に基づいてEMS開発担当者や使用者に対する勧告が示されている。

(1) アリゾナ大学の「グループシステム」の開発アプローチ

本システムを開発するに当たっては、プロセスのロスとゲインの考えに基づいて考察するアプローチを採用した。すなわち、会議の実際的生産性を、生産性を増す作用をもつ要素（すなわちプロセス・ゲイン）と生産性を抑制する（すなわちプロセス・ロスを与える）要因の結果として見て、グループの内部相互作用の機能不全（すなわちプロセス・ロス）を除去し、グループがその作業潜在能力

を達成または越える（すなわちプロセス・ゲイン）することを目指す。

以下にプロセス・ロスの要素を示す。

① プロダクション・ブロッキング

言葉による交信中に一時に集団の中の一人だけしか発言できないという問題であり、つぎの３種類の影響が考えられる。

・ 減衰ブロッキング

着想が発生したときに発言することを妨げられたグループのメンバは、時間が経過するとこれを忘れてしまうか、またはその関連性や独創性が弱く感じられるようになるためこれを抑制してしまう。

・ 集中ブロッキング

着想の発言を待っていると、そのグループのメンバは新しい着想を創作するよりも最初の着想を忘れないようにすることに注意を集中してしまう。

・ 注意ブロッキング

他のメンバの発言を聞くことにより、新しい着想を作り出すことが妨げられる。

② 発言時間の不均等

グループ中における言語通信可能時間の配分を意味し、集団が大きくなるに従って各参加者の割り当て時間は短くなるという問題である。

③ 評価不安

着想に対してグループが共有する否定的評価に対する不安のために、個人が着想や意見を引っ込めてしまうという問題である。

④ ただ乗り

自分自身の貢献はなくて、グループのメンバがそのタスクを達成するため他のメンバに依存してしまうという問題である。複数の人が共通の目的に向かって努力を合わせるときは、一人一人が責任を持つときより努力が少ないことが報告されている。また自分たちの努力が成功のために必要性がより少ないと感じるときにただ乗りする傾向があり、これは大きいグループほど増加するようになる。

⑤ 認知の惰性

現時点における主題からはずれないで１本の思考の経路にそって討議を進めようとする問題である。

⑥ ソーシャライジング

グループがタスクに関連しない機能不全な行動をとるという問題である。ある程度のソーシャライジングは機能の効率化のために必要であるが、多すぎると会合の成果をさまたげる。

⑦ 支配

グループの一部のメンバが非能率的な方法でグループのための時間に不当な影響を与えるか、または独占するという問題である。

⑧ 記憶誤り

メンバが他のメンバの意見を充分に聞かないか、忘れてしまうという問題である。

⑨ 不完全分析

利用できる情報を使用しないで、他のメンバのもつ仮定条件に対抗しようとするという問題である。

(2) 「グループシステム」の概要

① 設備

アリゾナ大学は、複数のセットのEMS設備を使用している。すべての設備は、部屋の前方に焦点を集められるように参加者のテーブルが設けられている。各参加者は、各々のテーブルにはめこまれたカラーグラフィックディスプレイのワークステーションを使用する。各ワークステーションはネットワークで接続されている。最低限1個の大型ビデオスクリーンが使用され、その他の視聴覚支援（例えばホワイトボードや投影装置）も設けられている。会議室の隣には電子制御室があり、また即時にハードコピーを作るためのレーザープリンタとコピーマシンがある。

アリゾナにある最初の設備は、1985年に使用を開始した。主会議室には16人が着席できる大きいU型テーブルがあった。1987年につくられた二番目の設備は、アリゾナ・コラボラティヴ・マネジメントルームと呼ばれ、24のワークステーションが同心2列に配列されている。

これらの設備では、ネットワークに接続したワークステーションを使用しているため、グループの全員が同時に情報を提供できる設計のグループ作業ツールの設計が可能になり、さらに一貫してプロセスに関連する会議構成を強制することができた。この平行性と会議構成によって生産性プロッキングや、発言時間の不均衡や、支配や、ただ乗り等の欠陥は少なくなる。中央にある大型のグループ用スクリーンは「グループ記憶」として使用でき、プロセス構成の配分と記憶誤りのロスを少なくする。

② ソフトウェア

「グループシステム」のツールは、次の3種類のグループ行動のうちの一つを支援するように設計されている。

(a) 着想の発生

着想の発生は、多くの場合グループ行動（例えば意思決定や交渉企画や情報システム開発等）の最初の段階であり、ここで種々のオプションや代案が表面化する。この段階ではグループは主として創造性を重視し、着想の発生に専念し、着想の評価は後の段階に回す。着想発生用のツールとしては次のものがある。

・ 電子ブレーンストーミング（EBS）

参加者は全グループ内に無作為に配分されている個別のファイルに含まれる多数の個別の討議の中に意見を入れることができる。EBSのファイルの無作為配分によって、多数の個別の討議の展開が促

進され、そのため認知惰性ロスが克服される。ただし、意見に対するフィードバックの受け入れは困難となる。

・　トピック・コメンタ（TC）

各々の意見はタスクに特定の構造によって会議参加者から集められ、TC は 1 枚ずつ固有の名前を持つ 1 組の索引カードに似た働きをする。参加者は矢印キーを使用してカードタイトルのリスト上を反転表示バーを移動させてカードを選択する。カードを選択すると、参加者は機能キーを押して意見を入力するか、または別の機能キーを押してその主題に関連するすべてのそれまでのグループの意見を読むことができる。TC は意見が無作為に配分されず特定の主題に集中されているので、EBS より着想発生のための構造は多い。

(b)　着想の合成

着想合成の目的は、着想発生ツールで討議された着想や提案、代案を確認／合成／様式化／確定することである。参加者は多重ウインドウ表示スクリーンを使用して、着想発生中に展開した意見とタスクに特定の合成ツールの機能を読み取る。着想合成用のツールとしては次のものがある。

・　アイデア・オーガナイザ（IO）

参加者は各自独自の着想カタログ表を作ってグループに提出する。IO の画面は 3 個のウインドウからなる。1 番上のウインドウには、1 行の着想種別が表示されている。中央の IO ウインドウには、着想種別に関連する EBS 意見を切張りすることができる。意見の切張りは、二つ以上の種別に入れることができる。1 番下のウインドウは読取りのみが可能であり、EBSを使用して展開した先行意見が表示されている。使用者には先行 EBS 意見を一読し、着想種別を考案するように指示する。

(c)　優先度付与

優先度付与用のツールとしては次のものがある。

・　選択投票ツール

選択投票ツールは、種々の優先度付与法を提供している（例えば複数選択や 10 点評価や順番付け）。それぞれ投票を集計し、その結果を討議するために利用する。

・　代案評価（AE）

グループは指定された評価一覧表の中の各項目に対し、1 から 10 までの段階に一組の代案を評価する。評価基準は同じ重要性と考えることもあり、異なる重み付けをすることもある。

③　手順

手順とは、ソフトウェアや設備を使用する方法である。ソフトウェアや設備がプロセス・ロスを克服するためにどれだけ巧妙に設計されていても、正しく使用しないかぎり有効ではない。「グループシステム」手順における重要な要因を以下に示す。

(a) 会合の計画管理

会合計画管理の主目的は、合理的な会議議題を確定し、会議中におけるこの議題の取り上げを促進し、さらに会議後の活動の組織化を助けることである。この活動を支援するために、セッション・マネジャ（SM）という「グループシステム・ツール」を使用する。SMは重要な計画情報を見逃すことがないようにして会議前の活動を支援する。議題を確立することによって、グループは会議中により一層タスクに注意を集中することになる（ソーシャライジング対策）。会議中は、すべてのツールはSM経由で稼働するので、SMが議題を追って議事を進める働きがある。議題の確立によってより信頼できる流れが使用されるので、会議中の連携問題は改善される。SMはまた、グループ記憶の一部として会議の情報を永久に記録するので、会議後の活動の助けともなる。

(b) 会議の推進

「グループシステム」使用における会議のファシリテータ（推進者）の役割は多様である。ファシリテータは装置利用の技術支援を与え、集団ダイナミック支援を与え、会議議題計画を助け、進行中の組織設定を助け、システムの使用と訓練と保全の基準を設定して、組織の継続を確保する。会議中、ファシリテータは評価の不安やただ乗り、支配、ソーシャライジング、認知の惰性等のプロセス・ロスを克服する。

(3) 「グループシステム」の利用の評価

① 研究所内での研究

ここでは、着想発生と意思決定の2種類の一般的タスクを選択してさまざまな観点から実験を行った。主な実験結果の概要を以下に示す。

(a) 着想発生の研究

・ 着想発生グループのグループ作業における匿名の影響

匿名で意見をよせるグループメンバは、名前で識別された意見をよせるメンバより、より多くの意見を生み出し、より批判的探求的であり、人の提案した構想をより補足する傾向がある。このことから、匿名方式は、不安の程度の少ないタスクで実験する学生のグループにおいてさえ、評価不安を少なくすることが推定された。

・ 匿名とグループメンバの物理的近接性がグループ・プロセスに与える影響

グループメンバが匿名でさらに離れているときに、もっとも多くの意見が出たが、同室で作業する場合は満足度が最高であった。観察されたシステムの効率は、グループが匿名条件で作業するとき最高度となる報告となった。これらの結果は、匿名状態ではやはり評価不安が少なくなることを暗示している。

・　グループ着想発生能力に与える匿名性と評価的雰囲気の効果

匿名状態でしかも批判的な作為的意見のある状態で作業するグループがもっとも多く着想と総意見数を記録した。個別に識別されて着想を出し、作意的な賛成の意見を入力したグループは、もっとも満足し、最高の効果の感覚を持ったが、着想数と意見数は最低であった。匿名状態と批判的状態で性能が高くなる理由は、評価不安が少なく、分析がより完璧となるためと考えられる（着想がより頻繁に批判されるため）。しかしこのグループでは満足度は最低であり、そのため匿名性や批判もマイナスの副作用があるといえる。

・　グループの大きさと匿名性の組合せのグループ着想発生生産性に与える影響

発生した着想の数と品質の点で、大型グループの方が小型グループより有意に高い生産性を示した。すべての条件のもとで、グループのメンバは平均して約同数の意見を出した。このことは従来の非コンピュータ支援グループ研究ではグループ・プロセス・ロス（例えば生産性ブロッキングや均一な発言時間）が増す条件でも、ロスが少なくなっていることを暗示している。ただし大型グループで匿名の場合は、小型グループで名前を識別される場合より、批判的な意見が多くあり、匿名は個性をなくし、行動も抑制が除かれることを暗示している。小規模で名前を確認されるグループは、もっとも満足度が高く、他の実験条件のグループのメンバより自己能率評価は高いが、この技術の有用性についての感じ方はすべての条件のメンバの間でほとんど差はない。

・　小（3人）中（9人）大（18人）の3種類のグループによる比較

グループの生産性とメンバの満足度は、グループの大きさが大きいほど大きくなった。すなわち大型グループは中型グループより着想数は多く、満足度も大きかった。このことは、従来はグループの大きさとともに増加していた多数のプロセス・ロスのマイナス効果をEBSは打ち消すことができることを示している（すなわち生産性ブロッキングや発言時間や評価不安等）。

(b)　意思決定の研究

・　グループが実質的に作業手順を展開することを許されている場合と、あらかじめ規定された議題に従う場合とのコンピュータ支援グループの行う決定の品質と代案の数とメンバの満足度と合意度を比較した。議題に従うグループは、合意に達する見通しが低い。両グループの間には、他に統計的な差はないが、議題を追うグループはより多くの代案を作ることが明らかになった。

・　「グループシステム」のソフトウェアの特性がグループの意思決定性能に与える影響を調査した。ある実験環境としてEBSとIOと投票ツールを採用し、もうひとつの環境としてグループメンバ全員が同時にすべてのグループ情報を共有し、IOや投票ツールと類似の機能を組み入れたEBSのバージョンを使用した。実験の結果は、EBSのもとのままのバージョンはよりユニークな代案を発生し、EBSの第二のバージョン（多目的ツール）はより品質の優れた決定を発生した。このようにEBSが認知の惰性を克服するために使用した技術は、確かにより多くの着想をもたらしたが、これら着想の討議か

らの意思決定はより困難であった。

② 実地研究

ここでは、「グループシステム」の実際の組織グループでの利用の現地調査結果について、以下の事例が紹介されている。

- ブールブラウン社によるコンピュータ支援企画会議（社長以下30人の副社長と、機能別生産別地域別部門の最高責任者が参加）
- サウスウエスト・ガス社のコンピュータ支援戦略企画検討会議（全管理職を代表する29人の部長が参加）
- グレイハウンド・フィナンシャル・コーポレーションの競争分析会議（全社部門の部長30人が参加）
- ヒューズ航空機会社のブレーンストーミング（現在の性能評価システムの弱点の分類と優先順位付け）
- IBMの製造工場における戦略企画や工程問題や機能部門のデータ処理ニーズ等の企画や問題解決のための会議

(4) EMS開発担当者や使用者に対する勧告

① 設備

- EMS設備は種々の大きさのグループに対応する必要がある。
- 種々のプロセス・ロスに強力に対抗するため、各参加者はそれぞれ独自のワークステーションをもつ必要がある。
- ワークステーションは独自のハードディスクを持ち、（サーバではなく）各ワークステーション毎にEMSシステムソフトをインストールすることが必要である。
- 全参加者がグループメモリを直視できる位置に共有スクリーンを設けなければならない。
- 共有スクリーンは、コンピュータの作る情報表示以外に、種々のメディア（スライドや光学ディスクやビデオテープ等）をサポートする必要がある。

② ソフトウェア

- 「グループシステム」はツールキット方式を採用しているが、この方式は最大のフレキシビリティを提供し、広範囲なグループをサポートすることができる。すなわちEBS等の着想発生ツールは、交渉を行なうグループが使用すれば代案を作ることができ、システム開発者が使用すれば一連のシステム要件を作成することができ、また戦略企画者は決定的成功要因のリストを作ることができる。
- 一方、グループがグループリーダの意図に従ってシステムを使用するように、会議の特定の時点で参加者に提供される機能の種類と数を制限することが重要である。

・　情報の所有権の問題として、いったん参加者がグループの中で情報を共有すると、それはグループメモリの一部となる。「グループシステム」では、ファシリテータはグループの権限によって、公共情報を変更または削除することができる。このことは特に競合する派閥のある大型グループでは重要である。しかし小型でより同質的なグループではこのような正式な情報管理は重要ではないかもしれない。

・　ツールは容易に進化成長できるように設計すべきである。

③　会議の推進と手順

・　ファシリテータは会議過程と成果の改善のために重要である。もしファシリテータがいないときは、グループのだれかがその役割を果たすべきである。

・　「グループシステム」会議の重要な部分は、会議前の企画である。グループ・リーダにファシリテータと会って会議の目標と目的を明確化するように正式に要求すれば、会議成功の手段が決定でき、適切なグループ活動を選択することができる。

7.4.3　要求仕様作成段階におけるGDSS

原文タイトル：　An Integrated Environment for Requirements Engineering

翻訳タイトル：　要求エンジニアリングのための統合環境

著　者：　James D. Palmer
N. Ann Fields
(School of Information Technology and Engineering,
George Mason University, VA., USA)

出　典：　IEEE Software, May 1992, pp.80-85

本稿では、ソフトウェア開発の品質改善にとって最も重要と考えられる要求エンジニアリングのための統合環境が紹介されている。まず、この統合環境開発のための要件が述べられ、つぎに開発されたアーキテクチャが述べられる。さらに開発された統合環境を利用した実験とその評価について説明されている。

(1)　統合環境の要件

本プロジェクトのコンピュータ支援による協同作業環境では、大規模かつ複雑なアプリケーションのためのシステム・レベルおよびソフトウェア・レベルの要求の開発と分析を支援することを目標としている。概念的ひらめきから計画、プロジェクトの詳細定義に至るまで、要求開発のすべての観点

を取り込み、調整する。

この環境を利用するグループの事例研究では、大規模かつ複雑なシステムのシステム・レベルの要求を決定するユーザグループに対して、要求エンジニアリングを支援することができることが示された。また、ユーザと要求エンジニアリング・チームとの間で「要求の明確化」、「分類」、「分析」、「追跡可能性」、「検証」、「設計」等の対話および反復動作も支援することができる。

さらに、これらの動作はテスト計画段階へと進めることができる。このオープン・アーキテクチャが意味するところは、目的に応じて商用CASEツールを統合することができるということであり、また、たとえ要求開発支援用のツールであっても、それを開発のライフ・サイクル上の別のフェーズにも応用できる。

ライフ・サイクルのなかで、要求エンジニアリングは主要な、継続的な動作で、その最も重要な機能はユーザが認識しているニーズに合うシステムを定義することである。ユーザのシステム構想が正しく反映されることを保証するには、要求の明確化および評価において、ユーザの参加が望ましくかつ必要である。そこで、正確・完全かつあいまいなところのない情報を明確化するために、次の二つの機能が必要であることが分かった。

- 個人またはグループから有益な情報を入手すること
- そうして得た情報を適切なメディア形態で提示すること

このため本環境では、小グループの対話のテクニックを組み込み、グループの対話を支援するツールが必要となる。設計者が入手した情報を適切なメディア形態で提示することができるよう、テキスト、グラフィクス、音声、ビデオなどの情報の獲得と操作を支援するのがマルチメディア環境である。このマルチメディア・ベースの環境は、即座にプロトタイプ化を行い、その要求の形態だけでなく「内容」を考慮に入れる。概念とプロトタイプの提示を提供し、正式な詳細定義へと導く。ここでのCASEツールは、ユーザが不正確・曖昧・不完全な要求などの問題を取り扱うことができるようにするものである。

(2) 統合環境のアーキテクチャ

本環境は、Apple Macintosh fx に 8M バイト RAM 、160M バイト・ディスク装置、アナログ・ビデオ I/O 、オーディオ I/O 、静止デジタル・カメラ、その他の周辺装置という構成で稼働する。開発したソフトウェアは、SuperCard、HyperCard、Smalltalkを含むオブジェクト指向プログラミング環境を使用する。また、不正確・曖昧・矛盾する要求を取り扱うためのさまざまなCASEツールを設計、開発、採用した。

本環境は、システム・レベルの要求の分析、評価、テスト計画の立案、品質保証、追跡をライフ・サイクル全体にわたって行うための、要求の分類、索引付け、クラスタ化といった基本的な作業を支

援する。複数のユーザと設計者間の会話を支援し、従来どおりのテキスト形式やグラフィカルな情報資源だけでなく、音声、ビデオ、アニメーションなどの非標準的な資源についてもサポートする。

本環境の基本構成要素は、本プロジェクトで開発されたCASEツール群で、次のようなものがある。

- COSP：オブジェクト管理
- Lexscan：語彙の走査・分析
- Knowledge-Based Requirements System（KBRS）：知識ベースの問題解決ツール
- Costtool：コスト評価ツール

以下にこれらの各構成要素について説明する。

① COSP

オブジェクト管理を行うCOSPは、情報の「獲得」、「組織化」、「同期化」、「提示」を行う。

獲得機能は、ユーザ・設計者間の対話に重点を置く。本環境を利用すると、空間的・時間的に分散しているグループから、また、すべてのタイプのメディアから情報を得ることができ、結合ダイナミック・リンクを通してあらゆるオブジェクト・セットを他のオブジェクト・セットと関係づけることができる。

組織化機能は、特定の要求機能を支援するオブジェクト間のダイナミックな結合リンクを促進する。同期化機能は、要求のプロトタイプ化を随時行うもので、また、提示機能はいくつもの表示形態をサポートしており、ユーザや設計者に合わせて選ぶことができる。

本環境における情報はすべてオブジェクトに変換され、オブジェクト管理システムが管理する。

② Lexscan

このツールは、自然言語文の文法を分析する。ここでは、索引付けとクラスタ化テクニックを適用して要求を自動的に分類する。ここでの目的は、N個の要求セットをM個の要求グループにまとめ、MはNよりかなり小さくなるようにすることである。

Lexscanは二段階式クラスタ化アルゴリズムを使って、M個の要求文のからなるセットの中を切り分けたり、その相似点に従ってクラスタ化する。二段階式集団化アルゴリズムは、同じような要求とそうでないものをうまく区別して、同じ要求をまとめるのに必要な情報を提供する。

③ KBRS

要求が分類されると、知識ベースの要求システム（KBRS）がそれらを分析し、要求集団の中または要求集団相互間で、相反するもの、不完全なもの、首尾一貫しないもの、不明確なものを見つけ出す。

KBRSは、知識ベースのCASEツールで、不正確であいまいな語の使用のチェック、品質判定基準用語の間の矛盾の検知を行い、その問題の解消・解決のためにユーザに問題を提示する。それらの問題が解決されると、要求エンジニアがあとで追跡できるよう各文にタグをつけ、別のCASEツールの助けを借りて判定基準の基関数を割り当てる。また、エンジニアは、テストツールを割り当て、これらの

判定基準の基関数を利用する要求文の評価を行うためのテスト計画を立てる。

④ Costtool

この効果判定ツールは、新しいCASEツールの導入の影響によって変わってくる開発グループの平均生産性、グループの経験レベル、プロジェクトの精通度を測定する。オブジェクト指向設計向きに開発されているため、Costtool は入力としてクラスとメソッドを利用する。

⑤ オブジェクト・ライブラリ

以上の分析結果は検証済み要求文のセットである。エンジニアは、これらの文をライブラリに入れる。本環境では分類・クラスタ化アルゴリズムを利用しているため、要求仕様を記述し、それらを機能とキーワードに従って記録しておくことができ、そのキーを使って取り出すこともできる。このため要求仕様の再利用が可能である。

⑥ プロトタイプ化

いったん要求文を分析し、可能な限りの欠点を修正する（または、後で修正するよう注を付ける）と、それに合わせて設定されるシステムのプロトタイプ化が直ちに開始される。プロトタイプ化ツールには、さまざまな組織化設計ツール、アニメーション・ツール、画面開発ツールがある。その後、これらのプロトタイプはユーザおよび開発者にそれぞれが求める形式で示される。

(3) CSCW 用に設計された統合環境

CSCW 用に設計された統合環境では、システムは黒板式制御構造によって管理され、ミーティングのプロセスを管理するサブシステムと情報および方法を制御するサブシステムを持つ。

ミーティング管理部では、ミーティングのファシリテータと通常のミーティング管理を支援する。このミーティング管理部を通して、ファシリテータは提示の制御やネットワーク構成の制御、パブリック画面、検討項目、方法の評価などを行うよういくつかの構成要素に指示を出す。ミーティングの流れをとらえるためのツール群やミーティングの記録を保存するサブシステムもある。

一方、情報／方法管理部では、マルチ属性ユーティリティを含むいくつかの共通分析ツールと、領域固有の分析ツールを提供する。現在、二つの領域（転送システムとシステム要求）に対するツールがあり、必要に応じて他の領域分析ツールを取り入れるよう構成することができる。例えば、CSCW 環境を利用してシステム要求を生成する場合、情報/方法管理部からLexscan、KBRS、Costtoolにアクセスして、ユーザがその領域の問題を理解し、正しく、明確な要求仕様を生成できるようにする。

(4) CSCW 用の統合環境を利用した実験

① 実験環境

五つの意思決定グループ、それぞれに 4 ～ 9 人が参加して、全体で32人が参加する形のCSCW 環境

を使って実験を行った。うち3グループは、19人の哲学博士過程の学生と3人の教授からなる。各グループとも3時間のミーティングで、今後の2～5年間でジョージ・メイソン大学で行う要求エンジニアリング研究のための研究課題に優先順位をつけるというものである。

第4のグループは、6人の上級行政官からなり、3時間で米国における原子力の将来に関する研究の要求を開発するというものである。このグループはグループ自身で検討項目を設定したが、他のグループのように決定問題は与えられなかった。

第5のグループは、5人の修士過程の学生で、2時間で学内の駐車問題の解決策の優先順位一覧を作成した。

② 実験の評価

意思決定支援環境の効果に関するデータを集めるため、ミーティングの様子をビデオにとり、ミーティング後参加者全員に対してアンケートをとった。

アンケートは、一連の質問文について賛成かどうかを5段階スケールを使って尋ねるものである。各質問文は、意思決定過程、これ以前のミーティングに参加しての満足度、使用したコンピュータ支援ツールの適切性と満足度を評価するものである。

アンケートの最初の部分は12の質問文からなり、グループでアイディアを出してゴールに到達するためのコンピュータ支援による意思決定の価値、到達した結論に対する参加者の参加度や信頼度、その過程と結果に対する満足度に関するものである。ここでの回答では、56～90%の参加者が意思決定支援環境を好ましいと回答している。

アンケートの次の部分では、使用した特定のコンピュータ支援ツールについて評価させた。ここでの4つの質問文に対して、81～97%の参加者が好ましいと回答している。

アンケートの最後の部分では、問題点は何か、なぜ決定がされたか、どのような決定かを参加者が理解するにあたりミーティングは有効であるか、また、各自の懸念をミーティングで表現できたかどうかについて尋ねた。質問文は3群からなり、以前のミーティングにおける経験（5問）を評価するもの、今回のミーティングにおける経験（5問）を評価するもの、以前のミーティングと今回のミーティングをどのように比較するか（4問）を参加者に尋ねた。これらの質問文に対する回答では、63～91%の参加者が意思決定支援環境を好ましいとしており、以前のミーティングのほうがよいというものは、35～58%であった。

この最初の評価では参加者の数が少なく、分かったことの解釈も限られるが、これらの結果は、本環境で使用した意思決定プロセスとコンピュータ支援ツールが役に立ち、かつ満足できるものであるということが意思決定者に理解されたということを明らかに示している。

7.5 協調作業システムの設計支援

7.5.1 拡張SADT

原文タイトル：	Augmenting SADT To Develop Computer Support for Cooperative Work
翻訳タイトル：	CSCWのためのSADTの拡張
著　者：	David A. Marca (Digital Equipment Corporation, Software Development Technology Group, NH, USA)
出　典：	Proceedings of the 13th. International Conference on Software Engineering, May 1991, pp.1-10

本稿では、システム分析の方法論であるSADT（Structured Analysis And Design Technique）を、オフィスにおける協調作業にあてはめた研究について紹介されている。まず、従来のSADTの技法が概説され、つぎに、WinogradとFloresの提案した言語・行動フレームワークをシステム分析に応用するアプローチが紹介される。さらに、このフレームワークを応用した場合のシステム分析の新しい実践状況や、SADTで使用される図形とモデルの新しい解釈が説明された後、この拡張SADTを協調プログラムの仕様決定に利用した例が述べられる。最後に拡張SADTを利用した結果をシステム・アナリストとユーザのそれぞれの立場からまとめている。

(1) SADTの技法

SADTは、システム分析のための言語であるとともにそのプロセスであり、システムとその動作を記述するために開発されたものである。SADTの言語は、ボックスと矢印から構成されており、システム・アナリストは、それを用いてシステムの活動、情報、そして物理的な構成要素をダイアグラムとして記述することができる。例えば、ボックスは組立てという活動を表現し、矢印は、部品、ツール、およびアセンブリ命令を表す。

SADTの言語とプロセスは、一般システム理論に基礎を置いた方法論である。したがって、SADTのグラフィックな言語は、協調、リアルタイムの相互作用、およびフィードバックなどのシステム特性を記述する表現能力がある。SADTを利用して、コンピュータ支援による製造、リアルタイム・システム、銀行業務、意思決定、プロセス改良、技術計画、ソフトウェア工学、企業統合などを記述することができる。

(2) システム分析のための言語・行動フレームワーク

システム・アナリストは伝統的にオフィスの情報の流れをモデル化することを追及してきた。しか

し、このアプローチによってこの種の仕事の要求仕様を作成しようとすると、非常な困難を経験することとなる。その理由は、グループのメンバの仕事の非常に重要な部分が会話であることである。誰と話すか、何を言うか、それらのディスカッションのビジネスの上の理由などが重要なのである。人々がどのように協力して仕事をするかを理解できるような別のフレームワークを最初に検討した方が、システム分析が簡単になる。

WinogradとFloresが提案したパラダイムは、仕事中の人々の会話を研究することを示唆している。このパラダイムを利用してオフィスを見ると、仕事とは基本的には社会的な性質を持っており、単に「情報」だけではなく「会話」が人々の仕事にとって本質的であることが示唆される。

この洞察はシステム分析の一般的な実践では以前は欠けていたのであるが、これをもって、人々が多様で複雑で互いに関係しあった会話を交わしている場所であるオフィスを記述するために、SADTの方法論を拡充することができる。

(3) システム分析の実践の新しい解釈

現在のシステム分析の実践は、グループのインタラクションについて適切な理解を得るためには不適切なものである可能性がある。本研究において、方法論の不適切性について特に次の三つの側面が発見された。

① ユーザの仕事を実体験によって理解する

伝統的なシステム分析は、ユーザを「研究」することに重点を置く。伝統的な面接技法は、人々に自分の仕事について語ることを奨励するけれども、システム・アナリストの目の前で、あるいはアナリストと一緒に実際に仕事をやってみることは奨めていなかった。グループのインタラクションのニュアンスを理解するためには、システム・アナリストは、彼らと一緒に実際に仕事をやってみることによって、ユーザを理解する必要がある。

② グループの規範と共有された作業方法を発見する

伝統的な面接技法では、特定の仕事のやりかたについて個人に質問することに重点がおかれていた。この質問スタイルの性格からして、ある一人の作業者の仕事が他の人の仕事にどのように影響するか、また他社の仕事からどのような影響を受けているかを発見するのは困難であった。面接を受ける人は、自分の仕事について各々個人的な解釈を持っている。

しかし、もっと多くの情報を与えてくれ、詳細なのは、仕事についてのグループとしての解釈である。この解釈を引き出し、彼らの規範と共有された作業方法を発見するには、チームの全体を同時に、しかも彼らが仕事をしているときにインタビューすることが必要である。グループ面接のセッションでは、セッションの進行につれて、システム・アナリストにもユーザにも、仕事についてのよりよい理解を得る機会が与えられる。

③　要求仕様を確認するためのラピッド・プロトタイピング

伝統的なシステム分析では、ソフトウェアの要求仕様と設計を文書で定義することを強調している。仕様書を書き、レビューし、修正するというサイクルが繰り返される。このような仕様定義と伝統的なレビューの反復プロセスは、システム開発の非常に初期の段階では有用である。これによって、開発したいシステムの最初のバージョンを構築するためのベースラインが作成されるからである。残念なことに、通常これらの仕様がシステムに対する唯一の要求として見られている。伝統的に、実際の作業場所においてソフトウェアへの要求仕様の調査が行われることはほとんど皆無であった。

本プロジェクトにおいては、最初の要求仕様はいくつかの非常に単純な再設計プロジェクトを管理するためのプロトタイプの開発につながった。この経験から、ユーザ・チームを支援するための初期のソフトウェア要求仕様の多くは、人々がそのシステムを日常的に利用する様子を観察することによって確認された。このアプローチは、再設計プロジェクトという特定のコンテキストにおいて、グループがチームとして仕事をするための要求仕様を検証するために役だった。また、システム開発に対する段階的アプローチは、不適切なシステム要求仕様の発見と修正を簡単にした。

(4)　SADTグラフィックスとモデルの新しい解釈

まず、伝統的なSADTグラフィックスとモデルを前述した新しいシステム分析技法とを比較した。伝統的なSADTグラフィックスでは、各シンボルは次の意味を持っている。

- ボックス：タスクを表現する。
- 入力矢印／出力矢印：利用される情報と統合された情報を表現する。
- 制御矢印：トリガやルール、ガイドラインを表現する。
- メカニズム矢印：タスクを、誰が何を使ってどのように行うかを表現する。

本プロジェクトの最初のモデル化作業から、SADTグラフィックスとモデルについてのこの従来の解釈が言語・行動のフレームワークに正確には対応していないことが明らかとなった。従来の解釈によるモデルは、情報の流れについて正確であることに価値を置いているが、この種の正確性はグループ内のインタラクションのダイナミクスを理解するにはあまり役に立たない。

協同作業の状況を記述するため、SADTアクティビティ・モデリング言語のグラフィックスについて「意味」を正しく変更することが考えられた。SADTのボックス、入力矢印、出力矢印、制御矢印、およびメカニズム矢印は、「会話としての仕事」の観点から見る時、次のような新しい意味を持つ。

- ボックス：行動を表現する。
- 入力矢印／出力矢印：要求と約束を表現する。
- 制御矢印：要求と約束の行われるコンテキストを表現する。
- メカニズム矢印：人、グループ等を表現する。

このように拡充されたSADTグラフィックスを用いて、新しいモデルが試みられた。しかし、この新しいモデルの検討を始めるとまもなく、仕事をタスクと情報として表現するのではなく、会話として表現しようとするならば、SADTモデルの意味そのものも変更しなければならないことが明らかになった。

協同作業状況のためのモデリング・フレームワークを作成するため、SADTモデルの意味を次のように変更した。「モデル」は、協同して作業する人々を示す。特に彼らが何を発言するか（会話）、および何を行うか（行動）を記述する。「目的」は、これらの会話および行動に関する質問である。質問の一般的な形式は「特定のビジネス・コンテキストが与えられたとき、人々は何を言い、何を行うか」というものである。これらの質問への回答は、グループのためのソフトウェア・システムを設計するために必要な理解を提供する。「観点」とは、会話を解釈する上での見地（適切な人間あるいはグループ）を提供する。オフィスを単一の観点から記述することは、解釈に整合性を与え、質問の集合への整合性のある回答が得られる。「精密性」は、目的に関係した質問のすべてに答えるためにモデルが備えなければならない詳細さのレベルを決定する。したがって、精密性は会話の粒度レベルを決定する。それ以下のレベルでは、より多くの質問をしても追加の価値が得られないようなレベルである。「境界」は、会話と行動が行われる範囲を定義する。会話とは他の人々あるいはグループと行うものであるから、境界はグループの間の「組織インタフェース」を示す。したがって、境界は開発されるソフトウェア・システムが究極的にはどのようにオフィスに適合するかを決定する役に立つ。

このようなSADTモデルの意味の再定義によって、伝統的なSADT方法論の拡張は完成した。

(5)　拡張SADTによる協調プログラムの仕様決定

拡張SADTはコンピュータによる協同作業支援システムの要求仕様定義に十分適している。協調プログラムについての過去の研究と先進的開発プロジェクトにおける理論とシステムのすべてには、二つの欠点がある。最初の欠点は、言語・行動の観点を中心にした長命な要求仕様定義方法論が存在していないことである。従来の研究では、広範囲で利用されているシステム分析方法論を「会話としての仕事」の観点と整合性の取れたかたちで適用することに成功していない。SADTの利用は、この最初の欠点を解決しようとするものである。

第二の欠点は、協調プログラムの主要な要素を同定し分離するアーキテクチャの欠如である。主要な要素とは、(i) コンテキストと会話、(ii) 会話に含まれる情報、および (iii) 会話情報がいかにして保持されるか、である。この第二の欠点は、複数の互いに関連したモデルというパラダイムから作成される仕様定義アーキテクチャによって解決される。このパラダイムでは、小さなモデルの集合を示唆する。一つ一つのモデルがそれぞれ一つの要素に焦点を当て、それぞれがある意味のある方法で他のモデルと関係している。このパラダイムを適用することによって、SADTでは新しい仕様定義アーキテク

チャを考案した。

このアーキテクチャでは、協同作業の主要要素を次の三つのモデルに表現する。(i)「協調モデル」は、会話とそれが行われるコンテキスト（状況）を記述する。(ii)「会話モデル」は、会話の情報内容を定義する。(iii)「ネゴシエーション・モデル」は、ある種の会話がネゴシエーションのプロセスによって保持される過程を記述する。

さらに、この仕様定義アーキテクチャは、それぞれのモデルを特定のレベルと関連づける。アーキテクチャの低位レベルのモデルは、高位レベルのモデルに依存しなければならない。例えば、会話モデルは、特定の約束（コミットメント）の情報的内容を記述するが、そのネゴシエーションのプロセスが具体的にどのようにネゴシエーション・モデルにおいて詳細に記述されるについては関知しない。協調モデルは、会話モデルに記述された任意の特定の会話に対する状況的コンテキストを含んでいる（このコンテキストは、会話の間に人々がどのような情報を共有しなければならないかに強い影響を与える）。

(6) 拡張SADTの使用経験

本プロジェクトの期間中、システム・アナリストとユーザは協同で拡張SADTを使用した。その使用結果の概要を以下に示す。

① システム・アナリストの使用経験

システム・アナリストは、拡張SADTを使用することによって、新しい方法でのシステム分析を経験した。これらの経験は協調プログラムについて次のような新しい洞察と、この種のソフトウェア・システムの仕様を定義するより良い方法へ導いた。

(a) ユーザの仕事を経験することの重要性

アナリストがユーザの仕事を直接経験するとき、協同作業グループの要求を引き出すことは容易になる。この方法によってシステム・アナリストは、ユーザと密接な関係を持つようになる。この密接さは、仕様定義および仕様定義プロセスに重要な、そして積極的なインパクトを与えた。

(b) レビュー・サイクルの圧縮

特別の会議室で行った協同著作セッションによって、SADTの作成者と読者間のサイクルが大きく圧縮された。例えば、システム・アナリストが再設計プロセスのモデルをユーザ（そのプロセスに従事中）の目の前で作成したとき、彼らはモデルの正当性について即座に検証し、モデルの不正確なところや欠点について指摘した。

(c) 統合された仕様

仕事を「会話としての仕事」と解釈するフレームワークは、従来のSADT方法論を拡張するための適切な視点を提供する。拡張したSADT方法論を適用することによって、協調的作業場所のSADT記述が

ソフトウェア設計に直接的に結び付くこととなった。

(d) 仕様の保守性

ネゴシエーション・モデルを会話タイプと結び付けることによって、保守性と拡張性に優れたモデルのネットワークが得られた。例えば、ネゴシエーション・モデルを変更すると、それにリンクされたすべての会話タイプも同時に影響を受ける。このスキーマは、オブジェクト指向プログラミングにおける同様の原理と直接の対応が付く。

(e) 視点が重要

有効な仕様を作成するには、「視点」をうまく利用することが必要である。特に、会話モデルは、その会話タイプが単一の視点から定義されたときに有効である。特定の会話が約束（コミットメント）を与えるものであるか、単に事実の表現を提供しているものであるかが、単一の視点から整合性を持って解釈されたときに、明確さが得られる。

② ユーザの使用経験

ユーザはシステム・アナリストに、拡張SADT方法論の利用価値についてのフィードバックを与えた。例えば、ユーザは、協同著作セッションの間に、自分たちの作業方法をシステム・アナリストと共有することは自分たち自身の再設計プロジェクトの効率性を高めたと述べた。また、彼らは、この仕様定義の結果、非常に価値のあるシステムが開発され、正確な報告書の作成が可能となり、彼らは製品再設計のためのすべての内部的チェックをパスした最初のグループとなったと述べた。

7.6 オフィスプロシージャシステム

7.6.1 Action Workflow

原文タイトル： The Action Workflow Approach to Workflow Management Technology

翻訳タイトル： ワークフロー管理技術へのAction Workflow のアプローチ

著　者： Raul Medina-Mora
Terry Winograd
Rodrigo Flores
Fernando Flores
(Action Technologies, Inc., CA, USA)

出　典： CSCW '92 Proceedings, November 1992, pp.281-288

本稿では、ワークフロー管理技術へのActionWorkflow のアプローチについて紹介されている。これは組織の中での作業の支援を目的とした設計方法論およびそれに付随したコンピュータ・ソフトウェアである。このアプローチは言語・行動としてのコミュニケーション活動の理論に基づいており、ネットワーク化されたコンピュータのユーザ間の協調のために開発された。まず、そのアプローチが説明され、つぎにその適用例が紹介され、さらにそれに基づくワークフロー管理システムのアーキテクチャが示されている。

(1) ActionWorkflow のアプローチ

過去10 年間にわたるAction Technologies 社における研究では、作業構造を言語行動として見る理論に基づいて、組織のコミュニケーションと活動を支援するためのコンピュータ・ソフトウェアを開発してきた。これらの研究から、原子的な活動「ループ」の同定と構成に基づく、ワークフローの新しい特性というアイデアを得た。そのループの中では、実行者（パフォーマ）が活動を完遂し、（組織内部あるいは外部の）顧客を満足させる。あらゆる組織の中での全体的なワークフローは、これらの活動ワークフローのループが組合わされたものである。いくつかのループは高度に再帰的（構造化された方法で何度も繰り返される）であり、他のものは状況に応じてユニークである。ワークフロー管理技術についての研究から、行動ワークフロー分析がワークフローを改善するために組織の中の行動の構造を再設計し、コンピュータによる支援を提供する上で有効であることが明らかとなった。

① 従来のアプローチ

ワークフロー管理への従来のアプローチのほとんどは、帳票や格納されたイメージなど情報オブジェクトのあるクラスから始め、それらのオブジェクトに対して行われるべき行動のシーケンスとしてワークフローを定義している。主要な組織構造は、ユーザの間における情報オブジェクトの「ルーティング」、およびそのルーティングにおいて行われるべき自動的な処理の仕様定義である。

伝統的な作業管理は、高度に定型化された書類処理には好適である。しかし、高度な教育を受けたオフィスワーカが定型的な仕事と自発的リーダシップを組合せ、仕事の質と顧客の満足への責任をとるようなオフィスワークの現実をサポートするには不適切である。

② ActionWorkflow におけるアプローチ

本アプローチも帳票を作成し管理する機能を含んでいるが、これらは行動とワークフローのループから構成されるビジネス・プロセス構造に組み込まれている。これによって、完成と満足を求める個人の仕事の結果を直接的に取り扱うための基盤が提供される。

行動ワークフロー・ループの中の行動の基本的なシーケンスは次のとおりである。常に顧客と実行者（パフォーマ）が同定され、ループは、顧客を満足するために完遂することを実行者が同意する特定の行動を扱う。ループは次の四つのフェーズにおいて進行する。

(a) 提案

表明されたある種の満足条件に従って、特定の行動を完遂することを顧客が要求する（あるいは実行者が提案する）。

(b) 同意

実行者と顧客の両者が、満足条件について合意する。満足条件には、次のステップを取るべき時間が含まれる。ネゴシエーションにおいては、この合意は部分的にしか陽表的になっておらず、仮定や標準的な慣例の共有されたバックグラウンドに依存する。

(c) 実行

実行者が顧客に対して行動が完了したことを宣言する。

(d) 満足

顧客が実行者に対して完了された仕事が満足すべきものであることを宣言する。

どのフェーズにおいても、明確化、条件についてのさらなるネゴシエーション、参加者による約束（コミットメント）の変更など、追加の行動があり得る。この構造は、満足条件を充たすため個人が行う行動ではなく、人々がそれを通じて協調するところの言語行動によって定義される。本アプローチのキーとなる相違点は、タスク構造から協調構造へのシフトである。従来のワークフロー・アプローチにおいては、協調行動は、一つのタスク、あるいは、タスクの間の情報の流れとして捉えられていたが、本アプローチでは、タスクはループ内において表現される要求と約束として定義される。

この単純なワークフロー・ループ構造は、一般的であり、普遍的である。行動ワークフロー・ループは、化学反応における原子のようなものであり、これらのループを組み合わせることによって、組織の中の複雑な現象のすべてが生成されるのである。

(2) ActionWorkflow の適用例

ここでは、特定の職務への候補者のレビューの管理のために開発されたアプリケーションの例を用いて、ビジネス・プロセス分析と支援を説明する。このプロセスは、採用広告、受付、履歴書の評価など、互いに関連した数十のワークフローに基づく職員採用という大きなビジネス・プロセスの一部分である。

このプロセスは、次の四つのループを持つ。

- 管理候補者のレビュー（人事担当取締役：人事部長）
- インタビュー・スケジュール（人事担当取締役：人事部長）
- 評価表の提出（人事部長：面接担当者）
- 候補者の評価（人事部長：人事担当取締役）

それぞれのループは、繰返しワークフローであり、顧客が左に、実行者が右に示されている。「管理候補者のレビュー」プロセスが中心となり、他の三つはこのメインプロセスと起動／依存関係を持つ。

候補者レビューのプロセスは、人事担当取締役が人事部長に特定の候補者のレビューの管理を依頼するときに開始される。人事部長は、オンライン帳票に面接担当者、候補者の求める職務、必要な技能等の情報を書き入れることによってこのプロセスを開始する。

このアプリケーションは、ワークフロー管理システムのLotus Notes版を使用して開発された。したがって、帳票はNotes フォームを設計するための標準機能を利用して定義された。レビューのプロセスの構造は、システム・アナリストと参加者によって定義され、ワークフロー管理サーバによって保持される定義データベースに格納されている。このサーバは、プロセスのすべてのワークフローのインスタンスを作成し、「インタビュー・スケジュール」というワークフローを自動的に開始する。

「インタビュー・スケジュール」というワークフローは、メインのワークフローの第2フェーズ（同意）に対応する。インタビューがスケジュールされると、部長は取締役によって要求された仕事を遂行することに同意する。同意フェーズにこのスケジューリングを含むことによって、特定の完了時間を約束することができる。

レビュー・プロセスが合意に達すると、「実行」フェーズが開始され、選択された面接担当者のひとりひとりに対して一つの「評価表の提出」というワークフローが自動的に開始される。ここでも、参加者のそれぞれに対して帳票が定義され、ワークフローの中で行動を行うために利用される。

特定の日にインタビューがスケジュールされると、評価報告書の提出のためのすべてのワークフローが開始され、特定の日付までに完了するよう、選定された面接担当者に指示が出される。

各面接担当者は、ワークフロー・データベースを利用して、どのワークフローが進行中であるかを知ることができる。もし面接担当者が合意された完了日の翌日までに評価報告書を提出しなかったならば、システムは、報告書を提出するように要求する「フォローアップ」催促を送るように定義され

ている。

ワークフロー構造の定義には、それぞれのフェーズにおいて、それぞれのワークフローの顧客、実行者、およびオブザーバが利用する帳票の定義が含まれている。面接担当者が面接のための文書にアクセスするとき、それは完成すべき評価表の形式であらわれる。面接担当者は「評価表の提出」というワークフローの実行者であるからである。他の参加者も、彼らが実行することのできる行動に対応した帳票を見ることになる。彼らの役割に応じてフィールドに書き込み可能であったり、編集不可になっていたりする。

面接ワークフローがすべて完了すると、システムは自動的にメインワークフローが完了したことを宣言し、第4フェーズへ進む。ここでは、人事担当取締役がプロセスへの満足（あるいは不満足）を宣言する。システムは取締役に、候補者の最終的な評価のためのワークフローについて行動を起こすようプロンプトするメール・メッセージを送る。

(3) ActionWorkflow のアーキテクチャ

全体的なアーキテクチャは、一つまたは複数のクライアント・アプリケーション（ワークフロー・エネーブルド・アプリケーションと呼ばれる）、およびそれらがワークフロー管理サーバとインタラクトし、それからサービスを受け取ることを可能にする構造と部品から構成されている。以下にワークフロー管理システムの主要な構成要素について概説する。

① ワークフロー・エネーブルド・アプリケーション

ワークフロー・エネーブルド・アプリケーションには次の三つのタイプがある。

(a) ワークフロー開始アプリケーション

このアプリケーションの例は、受注処理ワークフローを開始するように修正された既存のオーダエントリ・アプリケーションである。受注処理ワークフローに制御が渡され、新しい注文の確認、顧客要求の定義、製造部門への連絡など一連の行動が起動されたとき、オーダエントリ・アプリケーションのタスクは完了する。このレベルの統合は、既存システムの修正をほとんど必要としない。

(b) ワークフロー参加アプリケーション

上記のオーダエントリの例では、参加アプリケーションは、受注処理プロセスの詳細を実行するアプリケーションである。オーダエントリ・アプリケーションは、最初に、例えば信用状況をチェックするワークフローを開始する。クレジット担当部長に電子メールを送り、部長はそれに対してイエスあるいはノウをチェックして回答する。既存の電子メール帳票にこれらのボタンを追加すること、および、イエスあるいはノウの回答をワークフロープロセッサに理解できるように定義することが、アプリケーションをワークフロー可能とするのに必要なステップである。

(c) ワークフロー管理アプリケーション

ワークフロー管理アプリケーションは、作業を実行するのに必要なオペレーショナルな観点と行動に加えて、管理に必要な観点と行動を提供する。上記のオーダエントリの例では、受注処理サイクルタイムや失敗の原因等を追跡するために、ワークフロー管理がビルトインされたアプリケーションを利用することができる。

② STFプロセッサ

STFプロセッサは、アプリケーションのデータ・フォーマットとワークフロー言語インタプリタの標準トランザクション・フォーマット（Standard Transaction Format）との間の変換を行う。STFプロセッサは、アプリケーションが利用するインタフェースからワークフロー管理サーバを分離し、異なったプロトコルと技術を統合するためのレイヤを提供する。適切なSTFプロセッサを提供することによって、任意の既存のデータベース、メッセージング、あるいはネットワーク・システムをワークフロー管理ネットワークと統合することができる。

STFプロセッサには次の三つのタイプがある。

(a) メッセージ・ベース型

メッセージベースのアプリケーションは、メッセージの授受によってワークフロー管理システムとインタラクトする。STFプロセッサは、アプリケーションからのメッセージを受け取り、ワークフロー管理サーバと直接インタラクトする。同様に、アプリケーションへ送り返すメッセージを作成する。

(b) データベース・ベース型

クライアントのアプリケーションは、外部データベースへレコードを書き込み、修正する。この外部データベースが、特定のデータベース・プラットフォームのために開発されたSTFプロセッサによってコンカレントな形でアクセスされる。

(c) プロセス・ベース型

プロセス間通信STFインタフェースにおいて、クライアント・アプリケーションは、プロセス間のサービス要求（例えばリモート・プロシージャ・コール）を発行することによって、サーバからのサービスを受け取る。

③ ワークフロー管理サーバ

ワークフロー管理サーバは、格納されたワークフロー構造定義とトランザクション履歴を利用して、活動を解釈し開始する。ワークフロー管理サーバは互いにインタラクションを持つ次の部分から構成される。

(a) 定義データベース

このデータベースには、組織のワークフローを記述する。定義の中核はループタイプと活動名称、および対応した帳票の集合である。定義データベースは、また、異なったループを結び付けるリンク

関係と、エージェント・プロセッサが自動的に処理する行動を指定する。

リンク関係は、ワークフローが進行するにつれて、各参加者が取るべき適切な「次の行動」あるいはオートメーションのための「次の行動」を生成するために利用される。

(b) トランザクション・データベース

このデータベースは、完了したワークフロー・ループ、および進行中のワークフローの履歴を保存する。トランザクションの実行、および状況報告、概況報告を提供するためにアクセスされる。

(c) ワークフロー言語インタプリタ

本インタプリタはワークフロー宣言、ワークフロー活動、ワークフロー管理サービス要求など、ワークフロー言語の構成概念の形式でSTFプロセッサからのサービス要求を受け付ける。ワークフローの状態を求め、あらかじめ指定した判断基準（ワークフローの現在の状態と行動を行う人間の役割など）に基いて次の行動を計算することをワークフロー・プロセッサに指示する。その計算結果とワークフロー定義の論理に基づいて行動を実行し、報告を作成する。

(d) ワークフロー・プロセッサ

ワークフロー・プロセッサは、トランザクション・データベースの中にトランザクション・レコードを作成し、管理する。トランザクション・データベースは、ワークフローの現在の状態と履歴を、部分ループと関連した完了時刻について組織して記録する。

(e) エージェント・プロセッサ

エージェント・プロセッサは、定義の中に指定されたイベントのキューおよびワークフロー活動を開始するべき時刻を保持する。エージェントのコードは、ワークフロー定義言語で書かれており、ワークフロー・タイプおよびその開始の引き金を引く行動に基づいて開始される。ワークフロー構造の内部（活動を行い新しいワークフローを開始する）および他の機能（レポートの印刷、電子メッセージの送付、他のアプリケーションの実行など）の両方で活動を行うことができる。

④ 設計ワークステーション

設計ワークステーションは、定義を作成、修正、保持するための別個のアプリケーションである。本プロジェクトでは、高レベルでのワークフロー・マップのためのグラフィカルなノーテーションを開発し、これらのマップを作成し操作するためのインタラクティブで構造化されたドローイング・ツールを実装した。これらを用いて、ワークフロー管理システム開発とともに、あるいはそれとは別個に、ビジネス・プロセスの再設計を行うことができる。

7.6.2 Oval

原文タイトル： Experiments with Oval: A Radically Tailorable Tool for Cooperative Work

翻訳タイトル： Oval の実験：協同作業のための徹底的に柔軟なツール

著　者： Thomas W. Malone
Kum-Yew Lai
Christopher Fry
(Center for Coordination Science, Massachusetts Institute of Technology, Mass., USA)

出　典： CSCW '92 Proceedings, November 1992, pp.289-296

本稿では、Oval という名の協同作業のための「徹底的に柔軟な（radically tailorable）」ツールについて紹介されている。まず、Oval 開発のための要件が説明された後、その構成要素と利用方法について述べられている。つぎに、Oval の柔軟性を検証するために、既存の協同作業支援用アプリケーション（gIBIS 、Coordinator 、Notes 、Information Lens）の持つ機能をOval 上で実現させる実験とその評価について述べられている。

(1) Oval 開発のための要件

ユーザ自身をソフトウェア設計者にしてしまうエンド・ユーザ・プログラミング環境においては、ユーザに「徹底的に柔軟な」ツールを与えることが有効と考えられる。これらのツールによってエンド・ユーザは、実用システムを発展的に修正しながら、広範囲の異なったアプリケーションを開発することができるからである。「徹底的に柔軟な」システムが「通常の」柔軟なシステムと異なっているのは、ユーザが広範囲の非常に異なったアプリケーションを開発することのできる点である。例えば、同じブランクのスプレッドシートから出発して、ユーザは個人用の予算から、販売予測、さらには企業財務までの範囲のアプリケーションを開発することができる。

徹底的に柔軟なシステムは、また、エンド・ユーザが、なんらかのプログラミング言語で命令を与えるのではなく、実用アプリケーション（ブランクのスプレッドシートなど）を段階的に修正するという点において、従来のプログラミング言語と異なっている。このように、徹底的に柔軟なシステムは、アプリケーションを利用することとそれを設計することとの間の「認知的距離」を縮める。

徹底的に柔軟なツールを設計する上での重要な問題の一つは、「適切な」抽象レベルにおいてビルディング・ブロックを設定することである。つまり、ビルディング・ブロックは、何か有用なことを実現するには大変な労力が必要なほど低レベルであってはならないし、あるいは、ユーザのニーズが変わったときに大幅な修正が必要なほど高レベルであってもならない。

(2) Oval の構成要素

Oval の名称は、オブジェクト（object）、ビュー（view）、エージェント（agent）、リンク（link）という四つの重要なビルディング・ブロックの頭文字を取ったものである。これらのビルディングブロックについて以下に説明する。

① オブジェクト

半構造化された「オブジェクト」は、人々、タスク、メッセージ、会議など、世の中の物を表現する。それぞれのオブジェクトには、フィールドとフィールド値の集合、およびそれに対して実行され得る行動の集合が含まれる。オブジェクトのタイプは、階層的になっており、下位に行くほど特定化されたタイプとなり、下位のオブジェクト・タイプは上位のオブジェクト・タイプのフィールド、行動、その他の性質を継承する。オブジェクトは、ユーザは自分の希望するだけの情報をそれぞれのフィールドに記入でき、しかもあるフィールドの情報は特定のタイプである必要がない（つまり、自由なテキスト、他のオブジェクトへのリンク、あるいはテキストとリンクの組合せでもよい）という意味において、半構造化されている。ユーザは、テンプレート・ベースのインタフェースという非常に自然な形式のインタフェースを介してこれらのオブジェクトを見たり操作したりする。

② ビュー

ユーザがカスタマイズできる「ビュー」は、オブジェクトの集合を要約したり、ユーザが個別のオブジェクトを編集することを可能にする。例えば、オブジェクトの集まりからフィールドを選択して、テーブル形式で表示することができる。あるいは、リンクを選択してオブジェクト間の関係をネットワーク形式で表示することができる。フィールドの一つに日付を含むオブジェクトを要約するには、カレンダ形式の表示が利用できる。適当な表示形式（テーブル、ネットワーク、カレンダ）を利用して、任意のオブジェクトの任意のフィールドのオブジェクトの集合を表示することが可能である。

③ エージェント

ルールベースの「エージェント」によって、ユーザが直接的に注意を払わなくても、タスクが実行される。エージェントは、新しいメールの到着、フォルダの中における新しいオブジェクトの出現、あるいは事前に指定した時刻の到来などのイベントによって起動される。エージェントが起動されたならば、あるオブジェクトの集合に対して、ある一組のルールが適用される。ルールには、それが適用されるオブジェクトの記述と、それらのオブジェクトに実行されるべき行動の記述が含まれている。行動には、移動、メールの発送、オブジェクトの削除などの一般的な行動と、ファイルのローディング、メッセージへの応答など、特定のオブジェクトに対応した行動がある。

④ リンク

「リンク」は、オブジェクト間の関係を表現する。例えば、ユーザはリンクを利用して、メッセージとそれへの回答の関係、人々とその管理者の関係、複雑な製品の部品の間の関係などを表現するこ

とができる。ユーザは、クリックすることによって、これらのハイパーテキスト・リンクをフォローすることができる。リンクによって表現された知識は、ルールで利用することができ、また表示を作成するために利用することができる。

(3) Oval の利用方法

① ユーザによる修正

システムに対してユーザ・レベルで行える主要な修正には、次のものがある。

- 新しいオブジェクト・タイプの定義
- 既存のオブジェクトタイプへのフィールド追加
- オブジェクトおよびオブジェクトの集合に対するビューの選択（あらかじめ指定された表示フォーマットの集合からの選択）
- 与えられたビューに対するパラメータの指定（どのフィールドを表示するかなど）
- 新しいエージェントとルールの作成
- 新しいリンクの挿入

② 情報の共有

Oval において人々が情報を保存し、共有する主な方法には次の三つがある。

まず、ユーザは任意のオブジェクト集合をファイルに保存し、後で自分が（あるいは別の人が）再度ロードすることができる。例えば、特定のオブジェクトに（直接または間接に）リンクされたすべてのオブジェクトを自動的に集めてファイルに保存することができる。

また、ユーザはメッセージの中で任意のオブジェクト集合を互いにメールでやりとりすることができる。ユーザが以前に保存したファイルをロードしたり、オブジェクトを含むメッセージを受け取るとき、そのオブジェクトのアイデンティティは保存される（既存のリンクはそのオブジェクトの最新版をポイントし、以前のバージョンは「前のバージョン」として格納される）。

さらに、リモート・データベースに格納されたオブジェクトを「ライブ」なかたちで共有できるバージョンも用意されている。ただし、この機能なしでもエージェントを利用して、「共有」オブジェクトを自動的にメールしたりソートし、ライブ共有の利点を多くを提供することが可能である。

(4) Oval を利用した協同作業アプリケーションの作成実験

Oval は徹底的に柔軟であるという仮説を検証するため、このシステムを利用してこれまでに記述が公開されている多様な協同作業アプリケーションをインプリメントする実験がなされた。ここでは、各アプリケーションの機能をどのように実現したか、およびオリジナル・アプリケーションのどの機能はインプリメントしなかったかを以下に簡単に述べる。

① gIBIS（討論支援）

gIBISは、グループが意思決定の際に考慮すべき定性的な要素を検討することを支援するツールである。gIBISにおける政策分析の要素は、イシュー（問題点）、ポジション（解決案）、アーギュメント（議論）という三種類のノードをもつネットワークとして表現される。それぞれの問題には、それに対応するいくつかの立場が存在する。そして、それぞれの立場には、それを「支持」あるいは「反対」するいくつかの議論が存在する。

(a) 実現方法

まずgIBISで利用する三種類のオブジェクト（問題点、解決案、議論）を定義する。例えば、"Argument（議論）"という新しいタイプのオブジェクトを定義するために、最初に"Thing"という基本オブジェクト・タイプから新しいサブタイプを生成した。この新しいArgumentオブジェクトは、あらかじめThingオブジェクトから継承した"Name"、"Keywords"、"Text"というフィールドを持っている。これに、Argumentオブジェクトに必要なフィールド（"Supports"、"Objects to"、"Entered by"など）を追加する。最後に、タイプとリンクの追加によって、このオブジェクト（および他のオブジェクト）のフィールドに値を記入する。

オリジナルのgIBISシステムにおいては、ユーザは、議論ネットワークの中のノードの間の関係をグラフィック的に見ることができた。Ovalでもこれを行うためには、"Network"という表示フォーマットを選択することによって、そのノードからのリンクをネットワーク形式で表示させることができる。

(b) 除外したオリジナル・システムの機能

オリジナルのgIBISシステムの機能のうち、一つだけを除外した。オリジナルシステムには、それぞれのタイプについて一次的リンクと二次的リンクの区別がなされていた。Ovalでこれを実現しようとすれば、単にフィールド定義を増やせば良いが、システムを複雑にしないためにこれは行わないこととした。

gIBISの機能のうち、インプリメントするために新しいシステム・レベルでのプログラミングを必要とする機能は次の二つである。(i) 集合体ノード（一群のノードを自動的に圧縮して一つの集合体ノードとしてしまう機能）、(ii) ノードをアウトライン・フォーマットで表示するノード・インデックス（相互の関係はインデンテーションによって示す）である。これらの機能はアプリケーションの主要点からは本質的ではないと思われるため除外されている。ただし、これらはOvalの全体的なパラダイムとは整合性を持つものである。集合体ノードは、フォルダへの新しい二つの行動（集合体ノードを作成する行動と、それらを分解する行動）を追加すれば実現可能である。アウトライン表示フォーマットは、他の多くのアプリケーションにおいても再利用可能であるため、オブジェクト集合のための現在の表示フォーマット（テーブル、ネットワーク、カレンダ、マトリックス）に追加することは有用であると考えられる。

② Coordinator（会話の構造化とタスク追跡）

Coordinator は、電子メール・ベースのシステムであり、会話の構造化とタスク追跡を支援する。

(a) 実現方法

まず一連のメッセージを自動的に"Conversation"フォルダにグループ化するため、システム・レベルにおいて三種類の修正を行った。(i) ユーザが新しいメッセージを作成するとき、新しい会話フォルダが自動的に作成される。このフォルダは新しいメッセージを含み、その新メッセージの会話フィールドもこのフォルダにリンクされる。(ii) ユーザが特定のタイプのメッセージに回答するとき、回答のためのメッセージ・タイプ（リプライ・タイプ）の選択枝を提示される。(iii) ユーザがリプライ・タイプを指定した後、回答メッセージの中のConversation フィールドが、自動的にもとのメッセージで指定されたConversationフォルダにリンクされ、回答がConversationフォルダに挿入される。

Coordinatorで利用されているのと同じ会話シーケンス（行動のための会話など）を提供するため、15 の新しいメッセージ・タイプを追加し、それらのリプライ・タイプを定義した。例えば、Request（要求）というメッセージ・タイプは、Counteroffer（対案）、Promise（約束）、Decline（拒否）などのリプライタイプをもつ。

また、Coordinatorでの様々な要約表示を提供するため、23 のフォルダを作成した（"Open Matter（未決着の問題）"や"My promises（わたしの約束）"など）。また、メッセージをこれらのフォルダから出し入れするための14 のエージェントと47 のルールを作成した。

(b) 除外したオリジナル・システムの機能

ここではCoordinatorシステムとは異なり、権限委譲のための支援機能を表立っては提供していない。その代わり、ある要求を受け取った人は、もともとの要求へのリンクをもつ新しい要求を送信することによって「マニュアルで」それを他の人に委譲することができる。

また、Coordinatorのいくつかの細かい機能についてはサポートしていない。例えば、会話の中のコピーされるだけの人はその状態を変えることができないという制限を設けてはいない。さらに、Coordinatorでは自動的に処理された細かいことのいくつかをユーザが「マニュアルで」実行しなければならない場合がある。例えば、もしだれかが要求への返事を受け取る以前にそれをキャンセルすることを望むならば、マニュアルでキャンセル・メッセージをアドレスし、それをオリジナルの会話に挿入しなければならない。

③ Notes（半構造化された情報共有）

Lotus Notes は、Oval に最も似通ったものである。重要な特徴としては、(i) データベース中のドキュメントは、半構造化されたテンプレートであること（他のドキュメントへの「ホットリンク」がオプションで付いている）、(ii) データベースの中のドキュメントを、ユーザ定義可能なさまざまなビューに従ってフィルタリングし、要約することができる点である。

(a) 実現方法

Notes の中のテンプレートは、Oval におけるオブジェクト・タイプと同値である。Notesの中のデータベースは、Oval のフォルダと同値である。Notes の中のビューは、Oval でのオブジェクトの集合のビューと同値である。Notes のビューは、現在のところOval では存在していない、ある種類の「アウトライン」表示フォーマットのバリエーションであるが、サポートされているテーブル形式の表示に似ている。

(b) 除外したオリジナル・システムの機能

Notes は、文書の「ライブシェアリング」に近似した、データベース複製アルゴリズムに基づいている。また、商用製品として、Notes はアクセス制御、「リッチテキスト」フィールド、計算フィールドなど数多くの機能を含んでいる。これらの機能は現在のところはOval では実現していないが、Oval の全体的なユーザ・インタフェース・パラダイムと矛盾するものではなく、他の多くのアプリケーションのためにも有用な機能と考えられる。

④ Information Lens（知的なメール分類）

Information Lens は、到着する電子メッセージを自動的にフィルタリングし、ソーティングすることによってユーザを支援する。また半構造化されたメッセージ・テンプレート（オプション）によって、ユーザがメッセージを作成することを支援する。

(a) 実現方法

Oval で Information Lens をインプリメントすることは、非常に簡単である。例えば、ユーザは新しいメッセージ・タイプを、オブジェクト・タイプ階層の中の Message のサブタイプとして定義し、メッセージをフィルタしソートするためのルールを定義することができる。

(b) 除外したオリジナル・システムの機能

Information Lens は、"Anyone" 宛のメッセージ・ストリームの中から特定のユーザにとって関心ありそうなメッセージを選択するエージェントを持っていた。このようなリモート・エージェントを実現するには、システム・レベルのプログラミングを追加する必要があり、Oval ではまだこれを行っていない。しかし、これはOval の全体的フレームワークと矛盾するものではなく、他のアプリケーションにとっても有用と考えられる。

(5) 実験結果の評価

これらの実験から以下の点が明かとなった。

① アーキテクチャの汎用性

本システムの基本要素（オブジェクト、ビュー、エージェント、リンク）は、情報管理と協同作業アプリケーションへのユーザ・インタフェースのために有用な機能を提供している。前述したオリジ

ナル・システムのすべては、ある形式の半構造化されたオブジェクトと、本システムのビューと同様の要約表示形式を持っていた。また、大半のシステムは、本システムのエージェントやリンクが提供する機能のうちのいくつかを備えていた。しかし、Notesを例外として、システムは特定化されたオブジェクト（メッセージや議論ノード等）やある種の表示フォーマット（テーブル形式等）だけを処理できるように開発されていた。半構造化オブジェクト、カスタマイズ可能なビュー、ルール型のエージェント、およびリンクという抽象レベルまで一般化することによって、Oval ではアプリケーションの範囲を大きく広げることができるし、相対的に複雑さはあまり増加させないですむのである。

② アプリケーションの統合

徹底的な柔軟性は統合を促進する。徹底的に柔軟なシステムが提供する用途の広さに加えて、同一のシステムの中で異なった多様なアプリケーションを開発できるという重要な利点がある。異なったアプリケーションの統合が非常に容易になるのである。Oval のような統合されたシステムによれば、行動項目とそれに対応するディスカッションの間にリンクをはることは、非常に容易になる。

③ 残された問題

本アプローチの潜在的可能性が完全に実現されるまでには、多くの困難な問題が残っている。例えば、共有分散ネットワークにおいて、リンクされた多数のオブジェクトを管理しようとすると、多くの困難な問題が発生する。この問題の特に重要な側面の一つは、部分的に共有されたオブジェクト・タイプ定義の発展をどのように管理するかである。また、本アプローチがアプリケーション開発のために必要なプログラミング労力を省くことに有効であったとしても、コンピュータにやらせるべき新しい有用な仕事をイメージすることは、簡単な仕事ではない。

7.6.3 EuroCoOp

原文タイトル：	Distributed Work Management: Activity Coordination within the EuroCoOp Project
翻訳タイトル：	分散型作業管理：EuroCoOpプロジェクトにおける活動調整
著　者：	Pippa Hennessy (X-Tel Services Ltd., University Park, Nottingham, UK) Thomas Kreifelts (GMD mbH, Augustin, Germany) Ute Ehrlich (TA Triumph-Adler AG,Nurnberg, Germany)
出　典：	GMD-FIT News (AC Special), No.10, June 1992, pp. 1-20

本稿では、非同期型コミュニケーションに基づく分散型協調作業を支援するための強力で効果的なシステム開発を目的とするEuroCoOpプロジェクトについて説明されている。ESPRITの資金援助を受けた本プロジェクトでは、多くの既存システムから構成要素を統合し、包括的な協調問題を含む大規模な技術応用研究を基に新しい協調ツールを開発するというアプローチがとられている。

(1) 既存の活動調整支援システムの問題点

最近のCSCW研究の一つに、協調作業の実施をモデル化するための「言語」を開発し、これによる表現を使ったシステムで、グループ作業を媒介させるというものがある。このようなアプローチによるさまざまなプロジェクト（DOMINOプロジェクト、Coordinator等）のシステムを利用した経験から、次のような問題点が明らかになった。

- 予め定義された手順と構造の硬直性が、応用範囲を限定する。
- システム内で適切に対処できない例外が多い。
- システムに指図されるのをいやがる人が多い（特に自分の判断が状況解決により適している場合）。
- 公式の構造化コミュニケーションと関連する非公式なコミュニケーションの間の統合が少ない（あるいは全くない）。
- タスク／会話／活動がすべて孤立して表現されるため、参加者に全体的なコンテキストが与えられない。

これらの問題点は、協調作業支援の手順と構造の硬直性が原因となっていると考えられる。これを避けるためには、モデル化のパラダイムに柔軟性を持たせ、構成可能にする必要がある。

(2) 活動調整支援の要件

本プロジェクトのシステムの中心となるのは、EuroCoOp活動調整モデル（EACM）とEuroCoOp活動調整ツールキット（EACT）である。EACMは、協調作業の実行を定義・支援する手段を提供する。一方EACTはEACMに基づくツールセットであり、ユーザのタスク調整を援助する。

EACMとEACTの設計に当たっては、上記の問題点を解決するために次のような要件を設定した。

① コミュニケーションの包括的な協調機能

コミュニケーションは、情報のやり取りと共有と定義できる。これには、メッセージを使った情報の転送と、文書等へのアクセスを認めることでの情報の共有という二つの異なるプロセスが含まれる。

② タスクのスケジューリングの支援

タスク・スケジューリングは、タスク実行の時間的順序の決定と定義できる。これは、期限、タスクの予想持続時間、必要な他タスクからのインプット、資源や人の利用可能性等、多くの要素に左右

される。

③　役割の責務の割当の支援

役割の責務の割当は、適当な人又はグループを見つけて、その人が役割を果すことができるか否かを判別する点で、資源配分に似ている。この機能は、役割に対する適性を決定できるように人の能力を特定し、利用可能性がわかるように現在の責務を記録することで、支援できる。

④　資源配分の支援

資源配分は、タスクの実行に適切な資源を利用できるようにすることである。これには、資源の特定、利用可能性の決定、それらの位置決定、関連タスクへの配分が含まれる。この機能を支援するには、資源とその利用可能性、位置の表現が必要となる。

⑤　進行トラッキングの支援

進行トラッキングには、活動を計画通り進行させるためのタスクの実行監視が含まれる。この機能を支援するには、タスク実行の予想進行度と実際の進行度の表現が必要である。この二つの表現を比較することで、活動が計画通りに進んでいるかどうかを決定できる。

さらに本プロジェクトでは、上記機能を支援するための一般的な要件に加えて、活動調整モデルとツールキットのために特に考慮すべき次の要件が設定された。

⑥　EACMは次のものを提供しなければならない。

- 様々な細分レベルで、修正可能なタスク仕様を定義するメカニズム。プロジェクトの概観から個々のタスクの詳細まで、ユーザが仕様を異なる視点で見る事ができる。
- 高度に構造化されたタスク、半構造化されたタスク、構造化されていないタスクの定義の支援。
- 時間の概念（絶対的な時間と重要な事象に関連する時間（タスクの開始や終了等）の表現を可能にする）。
- タスクの発生する全体的なコンテキスト（タスクを孤立させない）。

⑦　EACTには次の特徴がなければならない。

- インストールや利用が比較的簡単で、個々のユーザの好みに適合させることができる。
- 活動を定義、修正する手段がある。
- 協調作業を行う人々の間で、情報を交換するメカニズムがある。
- ポータブル・コンピュータを利用していて常にシステムと接続しているとは限らない移動作業者を支援できる。
- 公式のタスク関連のコミュニケーションばかりでなく、協調作業をする人々の間の非公式なコミュニケーションの手段がある。

(3) EuroCoOp活動調整モデル（EACM）の概要

EACMはタスク指向モデルであり、協調作業組織内の参加者が実行するタスクが中心的な問題となる。EACMの取り組む活動調整には、次のような側面がある。

- 実行すべきタスクの指定
- タスク実行の責務
- タスク実行の時間的制約
- タスク実行に必要な資源
- タスク間の従属関係

このため、EACMではタスクとタスク・コンテキストというコンセプトを採用している。

① タスク

タスクの主体は、責任者、タイミング制約、開始と終了の条件等、実際のタスク実行を説明する管理情報を表す。

EACMにおけるタスク・オブジェクトは、ある実際のタスクの計画された実行と実際の実行を記述する一連のディテールを表す。このディテールは、タスクの予想進行度を記述するのに用いられ、実際の進行度と比較することができる。タスク実行の管理（計画と監視）は、EACMが支援し、タスクの実行自体は、統合されたEuroCoOpシステムで利用可能な他のツールで支援できる。

② タスク・コンテキスト

タスク・コンテキストは、記述、サブタスク、タスクの参加者、タスクに関するメッセージ、タスク実行に必要な資源等、タスクの実行に関連する情報を表現する。

EACMにおけるタスク・コンテキスト・オブジェクトは、タスクの実行に関連する一連の情報を表す。この情報は、タスクの実行に係わる者が管理・利用する。そのため、ある参加者がある文書をタスクの他関係者に有益と判断した場合、その参加者はタスク・コンテキストの中でその文書を参照させることができる。

EACMでは、タスクのないコンテキスト、未拘束のコンテキストも可能である。これらは、人と資源をグループ分けするために用いられ、コミュニケーション機能は、グループの掲示板のように使われる。これらは、その目的で設定することもできるし、タスクが完了した後に留保しておくこともできる。

③ タスクの階層

タスクはそれぞれ関連するコンテキストを持たなければならない。これは、スーパーコンテキスト、すなわち、それがサブタスクとなるコンテキストに組込まれる。コンテキストは関連タスクを持つことができ、一つ以上のサブタスクを持つこともできる。あるコンテキストのサブタスクは、コンテキストに関連するタスクの微細な点を表す。こうして、タスクの階層を定義することができる。さらに、

条件属性を使ってタスク間の従属性を指定することができる。

(4) EuroCoOp活動調整ツールキット（EACT）の概要

EACTは、協調作業に係わる人々を支援するための基盤としてEACMを利用する。システムのユーザに情報その他支援を提供するために作業プロセスのモデルを利用している。

EACTのアーキテクチャは、ユーザ・インタフェース層、アクティビティ管理層、メッセージ・ハンドリング層の三つの階層からなる。

① ユーザ・インタフェース層

最上部のユーザ・インタフェース層は、タスクの参加者がシステムとインタラクトするためのユーザ・インタフェース・ツールからなる。このツールには、次の4種類があり、ユーザは好きなようにこれらのツールを切換えて、必要な場合はその間で情報をコピーすることができる。情報は、外部ツールとの間でやり取りすることもできる。またユーザは、アプリケーションのデフォルト値を決定して、ツールとのインタラクションを構成することができる。

(a) タスク・マネージャ

ユーザは、本ツールを使って、タスクを見たり、リストにしたり、ブラウジングしたり、作成、編集、操作ができる。このツールはまた、優先順位や注釈等の個人的なタスク属性も処理し、グラフィック・タスク・エディタの適合化のような個人的な機器構成を管理する。

(b) カレンダ

本ツールは、タスクに関連するか否かに拘らず、時間関連の情報を管理する。カレンダ・インタフェースの基盤は、個人的なカレンダ・ツール（共有カレンダではない）で、活動モニタからタスクに関する情報を得て、これを他のエントリと共にユーザに示すものである。データの提示にはさまざまなフォーマットが利用できる。

(c) オーガニゼーショナル・ブラウザ

本ツールでは、タスクの実行のため組織的コンテキストを掴めるように、組織内の人物、役割、資源に関する情報を見たり、利用したり、修正したりすることができる。この情報は、オーガニゼーショナル知識ベースに記憶される。

(d) コミュニケーション・ハンドラ

本ツールの主な目的は、タスク実行の中断を最低限にして、ユーザがメッセージの授受をできるようにすることである。そのため、このインタフェースでは、構成可能性が中心的な機能の一つとなっている。もう一つの重要な機能は、複数のルーチン機能の自動化能力である。例えば、あるフォルダが一定のタスク関連メッセージを入れるよう指定された場合、そのフォルダへの適切なメッセージの割当は、コミュニケーション・ハンドラが行う。

② アクティビティ管理層

中間のアクティビティ管理層では、活動モニタがユーザ・インタフェース層のインタラクションを媒介し、活動ストアを管理する。

③ メッセージ・ハンドリング層

一番下のメッセージ・ハンドリング層は、参加者と他参加者又は活動モニタ間のコミュニケーションを担当する。メッセージ・ハンドラは、実際にはX.400のMessage Transfer Agent（MTA）で、他のメッセージ・ハンドラからX.400メッセージを受信したり送信したりするためのチャネルの役目を果す。

(5) プロジェクトの現状と今後の予定

現時点では、EACTの基盤となる活動管理層とメッセージ・ハンドリング層が完成し、ユーザ・インタフェース・ツールの初期プロトタイプが開発された段階である。今後は、これらプロトタイプの評価にエンドユーザを参加させ、インタフェース・ツールの実現に対するインプットを行うことである。今後の検討を必要とする分野は、ツール間のインタラクションであり、これは、ユーザに対して整合性のある完全な支援を行うものでなければならない。

7.7 グループエディタ

7.7.1 Flexible Diff

原文タイトル： Flexible Diff-ing in a Collaborative Writing System

翻訳タイトル： 協同著作システムにおける柔軟な差分検出

著　者： Christine M. Neuwirth
Ravinder Chandhok
David S. Kaufer
Paul Erion
James Morris
Dale Miller
(Carnegie Mellon University, Pittsburgh, PA, USA)

出　典： CSCW '92 Proceedings, November 1992, pp.147-154

協同著作における重要な活動に、テキストの変更についてのコミュニケーションがある。本稿では、テキストのバージョン間の差分を報告する flexible diff と呼ばれるシステムについて紹介されている。まず、本システム開発の要件となった、柔軟な差分検出システムを設計するときの課題について述べられ、つぎに開発されたシステムのアーキテクチャについて述べられる。最後に、関連する研究と今後の課題について簡単に述べられている。

(1) 柔軟な差分検出システムを設計するときの課題

差分検出プログラムを設計する際の技術的な問題は、オリジナル・テキストとリビジョン（改訂版）をどのように取上げ、ふたつのバージョンの間の変化を記述する編集スクリプト（差分報告）を、時間的／スペース的に効率的な方法で作成するかである。時間的／スペース的効率性の観点からは、最適な差分報告は、オリジナル・テキストを改訂版に変更するために必要な最小の編集変更をリストアップすることである。しかし、本研究での焦点はこのような技術的課題ではなく、以下に述べるようなヒューマン・インタフェース設計の問題、およびそれを解決するためのアプローチである。これらの問題を検討した結果、時間的／スペース的に最適な差分報告は、ヒューマン・ファクタの観点からは必ずしも最適ではないことが明らかになった。

① どのような変更を報告すべきか

Nachbar は、すべての変更を報告することはある種のタスクにおいては不適切であると主張し、spiff（spiffy diff）という名の柔軟な差分検出システムを開発した。例えば、ユーザは、浮動小数点の数値に関して、ユーザ設定可能なしきい値以下の差を無視することを spiff に指示することができる。

この研究に基づいて本研究では、協同著作タスクに関して、柔軟な差分発見アルゴリズムを開発す

る必要のある次の三つの側面を同定した。

(a) 著作グループ内での役割の相違

著作グループは、通常、いくつかの異なった社会的役割を果たす複数の人々から構成されている。協同著作者は、自分が信頼する協同著作メンバの行った修正のすべてをチェックすることには関心を持たないが、信頼度の低い編集者の行った変更は、そのすべてをチェックしたいと望む。

(b) 協同著作グループのニーズの時間的な変化

変化を検知したいという協同著作グループのニーズは、プロジェクトのライフタイムの移行によって変化する。プロジェクトの最初の段階では、協同著作者たちは下書きを大幅に変更しているかも知れないので、詳細レベルでの変更を見ることはあまり有用ではない。むしろ、各自の提案したアイデアがどの部分に反映されているかという面に関心があると考えられる。一方プロジェクトの後の段階では、例えば、レビュアのコメントに答えるときに、協同著作者が行った変更のそれぞれに注意を払うことが有用になる。

(c) 協同著作グループのタイプの相違

グループによって、変化を検知するため方法が非常に異なる。ある種のタスク（例えば契約書の作成にあたる弁護士）に対しては、著作者は前のバージョン以来行われたすべての変更を知りたいと望む。例えば、もし読み手が、法律的な意味を持つ可能性のある変更のすべてを発見することを任務としている弁護士であるならば、変更のすべてを報告することに意味がある。

しかし、別のタイプの著作グループにおいては、著作者たちは前回からの変更点のすべてを見せられても、本来のタスクには関係のない情報に注意を向けさせる結果に終わるかもしれない。例えば、読み手が協同著作者である場合には、その目的は新しい原稿を意味中心に読むことであり、同時に、論文の中にまだ残っている問題をどのように表現すれば良いのかに関心がある。このような読み手は、すべての変更に関心があるのではなく、いくつかの変更点（例えば削除された文やパラグラフ）だけに関心がある。苦労して書いた文章やパラグラフを破壊してしまうことは重大な問題だからである。

本研究では、多くの読み手にとって、意味を取ること、残された問題の発見、および差分の検知は注意の切り替えを必要とするタスクであると仮定した。変更点の報告は、読み手にその変更点に注意の焦点を向けさせる傾向がある。しかし、このことはもっと適切なタスクから読み手の注意をそむけさせる可能性もある。このフレームワークから、「どのような変更を報告すべきか」という問題に対して、ネガティブな発散を減らし、ポジティブな発散を増やすような変更点だけを報告すべきという回答が得られる。これは読み手とタスクに依存することであるから、協同著作のための差分プログラムは柔軟で、読み手がどのような変更は無視するべきかを指定できるようでなければならない。

spiffと同様に、flexible diff は、ホワイトスペース（スペース、タブ、改行）での変更を無視して良

いかどうか、あるいはそれを他の文字と同様に扱うべきかをユーザが指定できるようにすることによって柔軟性を実現している。その目的は、表面的で意味的には無意味な指摘が差分報告に入ることを避けることである。

さらに、テキストについては、相違点をどのように測定するかという問題がある。この問題を扱うため、flexible diff は変化を測定するヒューリスティックスを導入している。ユーザは、指定した粒度以下の相違点は無視するようにプログラムに指示することができる。この粒度は、単語、フレーズ、文、またはパラグラフを選択することができる。

差分は次のようなヒューリスティックスによって測定される。もし、二つの単位の間で、それらの部分があるパーセントで一致していたならば、それらの間の相違点は無視される。この値を100%より低く指定したならば、読み手の差分検知負担は軽減される。例えば、苦労して書いた文を失わないことに関心がある協同著作者は、粒度を文の単位に設定し、しきい値を70%程度に設定すればよい。

「どのような変更を報告すべきか」という問題のもう一つの側面は、報告すべきイベントのタイプに関わる。特に、「移動」は、ある資料を削除し、別の部分に挿入することであると定義することができるが、読み手は移動が「移動」として明確に示されることを望んでいる。そこで本システムでは、同じテキストの挿入と削除を探し出し、それを移動操作として認識するための単純なテクニックを導入した。ユーザは「移動の最小長」というパラメータを指定することができる。このパラメータは、移動と判定されるための挿入／削除の最小長を制御する。これにより、"the"などの短いストリングの繰り返し出現は移動と判定しないようになる。

② 報告される変更を、どのように指摘すべきか

テキストの変更点をどのように指摘するべきかという問題には、変化の数、密度、複雑さ等いくつかの要因がある。変化の大きさが、あるしきい値以上の場合に、著作者たちはそれぞれの変更点が個別に指摘されるよりも、テキストのある領域全体が修正されたと指摘されることを望む。その方が読みやすいからである。

この観察に基づいて、flexible diff では、変化がどのように指摘されるかをユーザが制御できるようにするためのいくつかのパラメータが定義されている。最初のパラメータは、指摘の「粗さ」である。このパラメータでは、文字、単語、フレーズ、文、あるいはパラグラフのいずれかを指定できる。

また、ユーザは、置き換え（挿入／削除のペア）がどの程度正確に指摘されるかを制御することができる。次の三つのパラメータ、「共通性を探索するための最大距離」「差異の最大パーセント」および「コンカチネーションの最大距離」を指定できる。最初の二つのパラメータは、差異検出プロセスのトップダウンかつ再帰的な適用を制御する。例えば、プロセスは、単語レベルで適用される前に、フレーズレベルで適用される。もし、二つのフレーズが長さにおいて非常に異なっており（共通性を探索するための最大距離）、非常に多くの相違点を持っていれば（差異の最大パーセント）、フレー

ズレベルでの置き換えが報告される。その理由は、この場合、共通性を検査することはあまり意味がないと考えられるからである。第3のパラメータ（コンカチネーションの最大距離）は、ボトムアップのコンカチネーション・プロセスを制御する。相違点検出プロセスの進行によって、相違点のリストが作成されるにつれて、コンカチネーションの最大距離以内の置き換えが検査される。その理由は、隣接した、もしくは相対的に近い置き換えをコンカチネートした方が報告が読みやすくなるからである。

ユーザはこのようなパラメータを利用して、最初のうちは変更点について粗い記述を選択し（その方が早い）、その後特定の変更について選択的に詳細を見る（その方が正確）という使い分けが可能となる。

③ 変更報告のユーザ・インタフェースは如何にあるべきか

flexible diff のインタフェースは、協同著作プロセスを研究し、支援するために開発された著作環境である PREP Editor の中に埋め込まれている。PREP Editor は、テキストに対応したカラムと、多数のアノテーションと関連テキストを「一目で」見ることを可能にする著作環境である。テキストの比較結果は、新しいカラムにその報告を作成し、相違点は元のカラムにリンクされているので、容易な横並びの評価を行うことが可能である。例えば、一つの画面に、オリジナルの原稿、その改訂版、比較レポート、および評価の四つのカラムを表示させることができる。評価カラムは、変更点のいくつかについて説明したり、あるいは助言を求めるため、比較レポートに対して協同著作者が作成したアノテーションである。

このインタフェースは二つの重要な特徴を持っている。第一に、テキスト中の相違点に対して水平方向に整列されていることである。水平方向に書くことになれている読み手は、読む場合でも垂直方向よりも水平方向の方が速い。したがって、オリジナルと修正版を横並びに並べ、しかも変更点を水平方向に整列させることは、何が変更されたかを見るというタスクの認知的処理からの要求に合致しているのである。

このインタフェースの第二の重要な特徴は、変更に対してユーザがアノテーションを施し、変更についての説明を行ったり、協同著作者に質問することのできる機能である。レビュアは議論したり説明したい変更点について協同著作者の注意を喚起するため、変更点についてアノテーションを付けることができる。同様に協同著作者は、レビュアに変更の説明を求めることができる。この機能は、説明のない変更に対する著作者のフラストレーションを大幅に軽減する。

(2) flexible diff の動作概要

flexible diff では、まずユーザが指定した設定に基づき、入力ストリームの語彙的な分析を行う。この時点でホワイトスペースが検出され、通常は無視される。つぎに、その結果のストリームが差分プ

ロセスに入力され、編集スクリプトが生成される。このように、入力ストリームの実際の文字をシーケンス比較から分離することにより、異なった粒度に基づいてレポートを作成する上で多大の柔軟性が得られる。

差分検知アルゴリズムは、MyersのO(ND)最長共通サブシーケンス／最短編集スクリプト・アルゴリズムに基づいている。flexible diffにおいては、基本的なMyersのアルゴリズムが、比較報告の粒度（文字、単語、フレーズ、文、パラグラフ）を利用した階層的分解戦略とともに適用される。

より高次のレベルでの差分のリストを作成した後、いくつかの要因を検討して、次のレベルで対応するストリングを再帰的に差分計算するかどうかを決定する。ストリングの間の共通性が大きいか小さいかの判断は、「共通性を探索するための最大距離」、「差異の最大パーセント」という二つのパラメータによって制御される。

最後に、編集スクリプトが作成されるとき、「コンカチネーションの最大距離」というパラメータによって制御される編集スクリプトのコンカチネーションが行われる。このコンカチネーションによって、同じ場所における編集オペレーションが一つの大きなオペレーションに集約される。

Myersのアルゴリズムは、差分が少ないとき（シーケンスが近似）に特に性能が良く、その結果、ユーザが粒度を著作プロセスの段階に合わせて調整するとき、高速になることが判明した。

7.7.2 CoAUTHOR

原文タイトル： CoAUTHOR - A Hypermedia Group Authoring Environment

翻訳タイトル： CoAUTHOR - ハイパーメディア文書の協同執筆環境

著　者： Udo Hahn
Matthias Jarke
Stefan Eherer
(Fakultat fur Mathematik und Informatik, Universitat Passau, Germany)
Klaus Kreplin
(Triumph-Adler AG, Germany)

出　典： Studies in Computer Supported Cooperative Work, J.M. Bowers and S.D. Benford (Editors), Elsevier Science Publishers B.V. (North-Holland), 1991, pp.79-100

本稿では、複数の著者が協力してハイパーメディア文書を作成するための実時間環境であるCoAUTHORシステムについて紹介されている。まず、ハイパーメディア文書作成のモデルについて説明され、つぎに協同著作のグループ的側面について検討され、さらに、CoAUTHORプロトタイプの実装の基礎として利用される技術的コミュニケーションと調整のためのツールについて述べられている。

(1) 本研究のアプローチ

ハイパーメディア文書の作成レベルと協同著作活動のグループ・レベルとの関係を表7.7-1 に示す。

表7.7-1 ハイパーメディア文書作成のレベルとグループのレベル

グループのレベル / 文書のレベル	個別作成	アノテーション（注釈付与）	構成（とりまとめ）
アイデア処理	アイデアの生成	アイデア対するアノテーション	アイデアの結合・まとめ
文書設計	概要の生成	概要に対するアノテーション	概要のまとめ
文書生成	トピックの担当割り当て 各テキストの生成	テキストに対するアノテーション	上位レベルでのテキストの組立、テキストの改訂、テキストのまとめ

ハイパーメディア文書を作成するプロセスは、次の三段階のモデルに基づく。最初のフェーズ（アイデア処理）は、文書がカバーしなければならない主題（主要なトピックス）を決定する。概念的な洗練化と内容ブロックの変更が継続的に行われ、文書に実際に取り込まれるべき知識の仕様が段階的に決まっていく。この最初のフェーズは、オーサリング・プロセスの概念的レベル（テキストにおいて何が実際にコミュニケートされるべきかという問題）に集中される。第２のフェーズ（文書設計）においては、正式な文書構造が設定され、アイデア処理のフェーズからの概念的項目と対応が付けられなければならない。この段階では、アイデアをハイパーメディア文書の中に位置付けるというかたちで、アイデアの物理的な実現化が固定される（テキストの中でアイデアをどのようにコミュニケートするかが示される）。第３のフェーズ（文書生成）では、アイデアは適切なハイパーメディアのチャンク（かたまり）によって実現される。協同著作チームのメンバは、文書概要の中に埋め込むべき自然語テキスト、公式、グラフィックス、図版等を提供し最終的なハイパーメディア文書を作成する。

また、協同著作のグループ・モデルも三つの主要階層に分割できる。グループの作業は、著作というタスクのそれぞれのレベルにおいて、まずアイデア、概要についての提案、およびテキストの特定の部分等の個人の成果を作成するところから始まる。これらのフェーズから生まれるそれぞれのオブジェクトは、つぎにグループの他のメンバからのアノテーションを受け付け、適切性の評価、批判的コメント、代替案の付加等が行われる。これら個別の成果物についてのディスカッションの結果、グループのメンバ全体が賛成するアイデア、概要、テキスト部分が最終的に選択される。さらに、この

ようにして承認された文書の基本的構成要素が組み合わされ、文書作成の概念的、形式的、および生産レベルで利用されるべき、より大きな単位となる。これは本質的には、個別の断片的オブジェクトをとりまとめ、構成するというタスクである。ひとつひとつのアイデアを関係付け、構造化されたアイデアの集合（テキスト全体の内容）を形成し、概要についての提案を包括的なテキスト計画というかたちで構造化し、テキストの断片を集めて整合性の取れたハイパーメディア文書にするというタスクである。このフェーズにおいては、協同著作プロセスのさまざまなレベルで、不完全な仕様、不整合性、複数の排他的な結論が発生する可能性が高い。したがって、交渉に基づく調整の戦略によって、概念的な矛盾を解決し、代替案を選択／適応することが必要である。

本システムでは、ハイパーメディア環境における協同著作活動を、次の三種類の異なった知識ベースの作成、連続的修正、および保守として考える。

- アイデア知識ベース（IdeaKB）：ハイパーテキスト文書の概念的内容を保持する。
- ハイパーメディア文書ベース（HyperDocB）：文書概要の形式的な特徴を格納する。
- ハイパーメディア・オブジェクト・ベース（HyperObjB）：ハイパーメディア・オブジェクトそのもの（テキスト、図版、イメージ等）を格納する。

IdeaKB、HyperDocB、HyperObjBは、同一のハイパーメディア知識ベース（HyperKB）に対する異なったビューであると見ることができる。このHyperKBは、主題に関する多くの異なった表現レベルの間に成立する関係、例えば、アイデアのある集合とそれに対応するテキスト表現の、文書グラフの中の特定のロケーションへの対応関係を保存するものである。

(2) ハイパーメディア文書作成のモデル

ハイパーメディア文書を作成するために、CoAUTHORシステムが提供する主要なオペレータ(操作)について説明する。ハイパーメディア文書作成の各段階（アイデア処理、文書設計、文書生成）でもっとも特徴的なオペレータを以下に示す。

① アイデア処理

アイデアの集合とそれらの間に成立する関係を生成し、かつ補充するための概念的オペレータとして以下のものがある。

- IdeaKBの中に基本的アイデアオブジェクトを初期作成するオペレータ。
- IdeaKBの中の概念の階層化あるいは集合化をはかるために、アイデアオブジェクトとそれを特殊化／一般化する別のアイデアオブジェクトとを関係付けるオペレータ。
- アイデアオブジェクトに属性を付与するオペレータ。
- 二つのアイデアオブジェクトの因果関係／論理的依存関係を定義するオペレータ。

② 文書設計

アウトライン生成の間に操作される基本的オブジェクトは、トピック・チャンクである。これは、文書の中の特定のトピックを表現し、目次の項目に似ている。トピック・チャンクを扱うオペレータとしては以下のものがある。

- トピック・チャンクを新規作成するオペレータ。
- あるトピック・チャンクに含まれるサブチャンクを定義するオペレータ。
- ハイパーメディア文書にアクセスしたときに、トピック・チャンクが現われる順番を指定するオペレータ。
- トピック・チャンクをIdeaKBの中のアイデアオブジェクトと関係づけるオペレータ。

文書設計フェーズの後では、HyperDocBの基本的構造の指定が完了している。さらに、HyperDocBの中のオブジェクトは、アイデアとトピック・チャンクの間のアウトライン・リンクを介して、IdeaKBの中のオブジェクトと適切に関係付けられている。

③ 文書生成

文書生成フェーズでは、正式な文書構造に関係したアイデアを実際のハイパーメディア・テキストへ変換する（HyperObjBとのリンク付けを行う）。このフェーズでは、(i) 誰がトピックに関する実際のテキスト作成に全責任を負うかを決定（トピックの担当割り当て）し、(ii) 決定した担当者がアイデア仕様に基づいてトピックを現実のものとする、という2段階からなる。このフェーズで使用されるオペレータとしては以下のものがある。

- トピックに担当者を割り当てるオペレータ。
- トピックとHyperObjBにある既存のオブジェクトとを関係付けるオペレータ。

担当著者はトピックを現実のものとする際に、HyperObjB 中の既存のテキスト、写真、図表、音声録音、映画等のオブジェクトを利用したり、それを編集したり、あるいは新規に作成したりすることができる。

(3) 協同著作のグループ的側面

協同著作に関したグループ特有の側面は、基本的には、協同著作チームが個人の成果に対して与えるアノテーション、および、単一のアイデア／アウトライン仕様から整合性の取れたハイパーメディア文書を構成するプロセスと、それらの管理に起因するものである。これらのタスクのためには、主に次の三つのタイプのオペレータを考えることができる。

- すでに入力された主題に批評を加え、代替案や仕様のギャップを示すためのコメンタリ・オペレータ。
- 提供された仕様の一貫性、完全性、整合性をチェックするためのメンテナンス・オペレータ。

・　意見の不一致や仕様のギャップなどを扱うアライメント・オペレータ。

これらについて以下に説明する。

① アノテーション

アノテーションは、アイデア処理、文書設計、文書生成という文書作成プロセスのあらゆるレベルで発生する。アノテーションは、概念オペレータ（IdeaKBのレベル）、チャンキング・オペレータ（HyperDocBレベル）、テキスト・オペレータ（HyperObjBレベル）で作成された構造体に付けられる。アノテーションは、批判的コメント、賛成のコメント、あるいは代替案を提示することによって、既存のノードとエッジを増大させるのである。このフェーズで使用されるコメンタリ・オペレータとしては、以下のものがある。

・　IdeaKB、HyperDocB、HyperObjB の中の既存のオブジェクトに賛成／反対のラベルを貼るオペレータ。

・　オブジェクト全体ではなくそれらの特定の属性に対して賛成／反対のラベルを貼るオペレータ。

・　異なった知識ベースの中のオブジェクトをリンクする関係にに対して賛成／反対のラベルを貼るオペレータ。

・　二つのオブジェクトの対を代替案として関係づけるオペレータ。

② アイデア／アウトライン／テキストの組み合わせ

グループはアノテーションに基づいて個別の成果を総合し、グループが確認したアイデア／アウトライン／テキストの断片を組み合わせる。最終的なハイパーメディア文書の中での主題、その組み立ておよび実現についてのグループとしての共通認識に到達するためには、それぞれの参加者からの異なった考え方の交換、議論、および再配置が必要となる。

不確定性と潜在的な不整合の可能性を解決するために、著作チームはアノテーションに基づいて、どのオブジェクトを文書に取り込み、どのオブジェクトを取り除くかを決めなければならない。このためには、まず最初に、ネゴシエーションすべき問題を決定し、つぎにそのそれぞれについてアライメント・プロセスを開始しなければならない。整合性のない議論の多い問題に焦点を当てるためには、次の三つの主要なメンテナンス・オペレータが提供される。

・　知識ベースの中で、賛成と反対の両方を指定されたすべてのオブジェクトおよび関係を抽出するオペレータ。

・　知識ベースの中で、代替関係を指定されたオブジェクトを抽出するオペレータ。

・　知識ベースの中で、他のオブジェクトにリンクされていない孤立したオブジェクトを抽出するオペレータ。

これらのメンテナンス・オペレータによって議論の余地あるオブジェクトを発見した後、仕様のギャップを埋め、代替案を適切に選択し、矛盾を解決するためのネゴシエーション手順が必要である。

そこで、意見を異にする集団、例えば矛盾するアイデアの場合には、賛成者と反対者の間でのネゴシエーションが開始される。これらのネゴシエーションは、会議でのコミュニケーション・チャネルを利用してインフォーマルに行われる場合もあるし、コメンタリ・オペレータを利用してフォーマルに行う場合もある。本システムでは、議論の的になっている問題の重要性についてのグループの判断を求める投票手順も提供している。これは、特別のアライメント・オペレータを利用して、著作チームのメンバによる重要性評価値を求めるものである。

未解決な問題に関する一連のグループ討議の後、著作グループは構成フェーズが適切に終了できることを確認しなければならない。このため、知識ベースの中で未解決となっているオブジェクトを抽出するアライメント・オペレータが用意されている。

最後に、アイデア／トピック／テキストの各々について、ネゴシエーションなしに確認されたか、ネゴシエーションの結果確認されたもののみが採用され、最終的なハイパーメディア文書の一部分として構成される。そしてこれ以外のものは除外される。確認されたアイデア／トピックスと、それらのテキストにおける実現との対応をきちんと把握しておくためのメンテナンス・オペレータが用意されている。それらの一つは、IdeaKBの中のすべての確認されたアイデアがHyperDocBの中のオブジェクトにリンクされたアウトラインであるかどうかをチェックする。また、担当者が割り当てられたトピックスが既にHyperObjBの中の部分的テキストに対応付けられているかかどうかをテストするオペレータもある。

要約すれば、グループワークの支援の立場からすれば、CoAUTHORの開発は、可能性としては整合性のない個別の成果を組み合わせ、アイデア、アウトライン、テキスト総合のレベルにおいて、整合性をもって統合されたグループ文書とするための手順に焦点を当てている。種々のメンテナンス・オペレータに基づいて、整合性のない仕様の状態（一貫性の欠如、不完全、または代替的な問題）は、賛成意見、反対意見の交換によって発見されネゴシエーションが行われる。一方、競合する代替案の間の最終的な選択の意思決定は、アライメント・オペレータの定量的評価点に基づいて行われる。

(4) グループコミュニケーションと協同作業支援ツール

CoAUTHORのインプリメントは次の三段階で進められた。まず予備的な実証研究においては、いくつかの現実のアプリケーション（在庫管理、設計、協同著作等）が、ラピッド・プロトタイピングされた。これと並行して、前のプロジェクトで開発したツールを利用して、より本格的なインプリメントのためのフルスケールのプロトタイプが開発された。このプロトタイプはUNIX環境の中で開発されている。このプロトタイプからいくつかの改良版を開発した経験に基づき、プロジェクトの第2フェーズにおいて、最終的な実用向けシステムをMULTIWORKSマルチメディア・ワークステーションの上に開発する予定である。

このうちの最初のプロトタイプは、クライアント・サーバ・タイプのアーキテクチャを提供する。サーバは、次の三種類から構成されている。

① コンセプトベース（ConceptBase）

アイデア、文書設計、およびそれらの間の関係についての知識（IdeaKBおよびHyperDocB に格納される知識）を獲得し保守するための知識ベース管理サーバ。

② MULTOS

型を持つマルチメディア・チャンクを組織し、格納し、検索する（HyperObjBを管理する）ためのマルチメディア・データベース・サーバ。

③ 会議机（Conference Desk）

協同著作の短期的コミュニケーション管理のためのリアルタイムでの会議装置。

また、本システムには次の二種類のクライアントがある。

① インタラクション・ツールボックス

マルチメディア・オブジェクトと知識表現構造のグラフィカルな操作のためのハイパーメディア・エディタとブラウザをサポートする。

② グループ・ツールボックス

投票、構造化された討論、著作役割の管理、外部プログラムへのアクセス等のための機能を提供する。

——— 禁 無 断 転 載 ———

平 成 5 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　械　振　興　会　館　内
TEL　(03)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　(03)3815-7271

04−R004