グループワーク支援システムの研究開発報告書

CSCW

財団法人日本情報処理開発協会

財団法人
図書
日本情報処理開発協会

はじめに

近年、ハードウェア技術の進歩による価格の低廉化やソフトウェアの進歩により、ワークステーション（WS）やパーソナルコンピュータ（PC）の大幅な普及がもたらされてきた。また、ローカルエリアネットワーク（LAN）や広域ネットワーク（WAN）等の通信技術の開発は、地理的、時間的に離れているWSやPCの保有者をお互いに結び付け、リソースの交換、共有を可能にさせている。

一方、コンピュータの利用方法を見ると依然として計算、作表といった個人作業の肩代わりあるいは作業支援といった使われ方が圧倒的多数である。しかし、オフィスや生産現場等の社会では、これらデータを中心とした処理のほかに、伝達・調整・合意といったグループの協調作業や会議・レビュー・協同執筆等の協同作業が頻繁に行われ、そのウェイトは非常に高いものになっている。また、近年における組織のグローバル化、地域分散化においては、こうした作業の効率化が極めて重要な課題となっている。したがって、先に述べたPCやネットワークの技術を用いて、組織やグループの協調そのものを支援するコンピュータ利用技術を開発することができるならば大幅な生産性の向上が期待できる。

こうした認識のもとに、当協会では平成３年度より２年間にわたって、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「グループワーク支援システムの研究開発」事業を実施した。

事業の実施に当たっては、グループワークを支援するシステムについて学会論文を中心に広く調査するとともに、各界の代表者からなるワーキンググループ（主査：斎藤信男　慶應義塾大学環境情報学部　教授）を設置し、ソフトウェア開発時におけるグループワーク支援の問題を中心に取り上げ幅広く検討した。また、平成３年９月と平成４年10月には海外調査員を派遣し、国際会議の出席、大学・先進研究機関の訪問調査を実施した。

さらに、グループワーク支援システムの構成方法、可能性、問題点等を明らかにするために実験環境を構築し、実験を進めるとともに、一部についてはプロトタイプシステムの開発を行い各種検討を行った。

本報告書は、平成３年度報告書と平成４年度報告書を合本し編集を加えたもので８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、分散開発とグループウェアについて述べている。第７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８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ともに広く情報処理技術発展の一助となれば幸いである。

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」メンバー名簿

（敬称略）

主　査	斎藤 信男	慶應義塾大学　環境情報学部　教授
委　員	青山 幹雄	富士通（株）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）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）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）NTTソフトウェア研究所 ソフトウェア開発技術研究部　主幹研究員
委　員	黒沢　隆	日本アイ・ビー・エム（株）東京基礎研究所 対話アプリケーション担当　課長
委　員	阪田 史郎	日本電気（株）C&Cシステム研究所 ネットワーク研究部　部長代理
委　員	津田 道夫	（株）日立製作所　ビジネスシステム開発センタ 第二開発部　主任技師
委　員	中村 英夫 （平成3年度）	（株）東芝　システム・ソフトウェア生産技術研究所 システム・ソフトウェア開発推進部　課長
委　員	貫井春美 （平成4年度）	（株）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）技術開発本部　システム研究所 制御研究室　主席研究員
委　員	三木 亮二	新日鉄情報通信システム（株）技術開発部 研究開発担当係長
委　員	向山　博	（財）日本情報処理開発協会　開発研究室　主任研究員
執筆協力	落水浩一郎	静岡大学　工学部　情報知識工学科　教授
執筆協力	松本 吉弘	京都大学　工学部　情報工学教室　教授
事務局	土川　潤	（財）日本情報処理開発協会　開発研究室　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．分散開発とグループワーク支援

6．1　分散開発環境の出現と発展 …… 141
6.1.1　分散開発とは …… 141
6.1.2　分散開発の出現の背景 …… 141
6.1.3　分散開発環境の発展 …… 142
6.1.4　分散開発の利点と問題点 …… 144
6.1.5　分散開発の分類 …… 146
6.1.6　分散開発を支援するソフトウェア工学技術 …… 147
6.1.7　分散開発を推進するために …… 149

6．2　グループの理解とグループウェア …… 152
6.2.1　グループ理解の重要性 …… 152
6.2.2　グループワーク支援システム研究の中心課題 …… 153
6.2.3　グループの特徴 …… 153
6.2.4　グループ作業での問題 …… 154
6.2.5　グループワーク支援システムに対する期待 …… 155

6．3　分散開発のプロセスモデル …… 156
6.3.1　プロセスモデルの概念 …… 156
6.3.2　分散開発とプロセスモデル …… 158
6.3.3　プロセスモデルとグループウェア …… 161

6．4　分散開発のプロダクトモデル …… 163
6.4.1　分散開発環境の定義 …… 163
6.4.2　プロダクトモデルとは …… 163
6.4.3　プロダクトの分類 …… 163
6.4.4　プロダクトから見た分散開発の現状と問題点 …… 164
6.4.5　分散開発のプロダクトモデリング …… 166
6.4.6　Hypertext を使ったシステムの実例 …… 171

6．5　分散開発とプロジェクト管理 …… 174
6.5.1　分散開発におけるプロジェクト管理の特長 …… 174
6.5.2　プロジェクト管理標準手順 …… 175
6.5.3　プロジェクト管理の内容 …… 177
6.5.4　分散開発プロジェクト体制 …… 179
6.5.5　プロジェクト管理の機械化 …… 180

6．6　CASE 統合環境におけるグループワーク支援 …… 182
6.6.1　CASE 統合環境とグループワークの接点 …… 182
6.6.2　CASE 統合環境とグループワークのサポート状況 …… 186

6．7　グループワーク支援システムの適用分野 …… 193
6.7.1　グループワーク支援の背景 …… 193
6.7.2　情報の広場としてのグループウェア …… 196
6.7.3　システム開発の現場におけるグループウェアの利用 …… 196
6.7.4　今後の課題 …… 199

6．8　ネットワーク環境における分散開発の将来 …… 202
6.8.1　コンピュータネットワーク技術の最近の進展 …… 202
6.8.2　アナログネットワーク技術の進展 …… 203
6.8.3　新しいコミュニケーションの形態 …… 204

7．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

7．1　D^{2}による分散開発支援 …… 207
7.1.1　ソフトウェア分散開発モデル …… 208
7.1.2　ソフトウェア分散開発支援システムのモデル …… 210
7.1.3　エージェント指向電子メール …… 211
7.1.4　リアルタイム電子会議システム …… 214
7.1.5　ユーザ認証システム …… 214

7．2　MERMAIDによる分散開発支援 …… 217
7.2.1　はじめに …… 217
7.2.2　MERMAIDシステムの概要 …… 217
7.2.3　実験構成と利用評価 …… 221
7.2.4　MERMAIDシステムの利用例 …… 225
7.2.5　MERMAIDシステムの応用分野 …… 226
7.2.6　おわりに …… 227

7．3　KyotoDBによる分散開発支援 …… 228
7.3.1　はじめに …… 228
7.3.2　協調活動モデル KCAM …… 229
7.3.3　協調活動支援環境 KAE のアーキテクチャ・モデル …… 230
7.3.4　協調活動スキーマ記述言語 KCASL …… 236
7.3.5　KyotoDB における協調活動支援 …… 237

7.3.6 KAEの実装 …… 242
7.3.7 おわりに …… 245

7．4 協調支援環境 Velaによる分散開発支援 …… 247
7.4.1 ソフトウェア開発における協調作業に関する問題点 …… 247
7.4.2 ソフトウェア開発環境の近未来像 …… 248
7.4.3 Vela 開発の背景と目標 …… 249
7.4.4 Navigation 支援に関する研究の現状 …… 251
7.4.5 Communication 支援に関する研究の現状 …… 253
7.4.6 おわりに …… 256

7．5 RTP と Person to Person／2 による分散開発支援 …… 258
7.5.1 リアルタイム・プレゼンテーション・システム（RTP） …… 258
7.5.2 Person to Person／2 …… 262
7.5.3 分散開発環境での適用可能性 …… 263

7．6 マルチメディア分散グループレビューシステム …… 266
7.6.1 はじめに …… 266
7.6.2 ソフトウェア開発におけるレビュー作業 …… 266
7.6.3 グループレビューシステムの開発 …… 268
7.6.4 グループレビューシステムの利用 …… 271
7.6.5 おわりに …… 273

7．7 CAPSDFのグループワーク支援 …… 274
7.7.1 CASEプラットフォーム …… 274
7.7.2 グループワーク支援機能 …… 275
7.7.3 作業誘導 …… 276
7.7.4 関連情報の参照と変更通知 …… 278

7．8 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 …… 280
7.8.1 SDEワークベンチ/6000 …… 280
7.8.2 コンフィギュレーション・バージョン管理/6000（CMVC/6000） …… 282

8．グループウェアに関する調査

8．1　グループウェア全般 …… 289
8.1.1　学際的調整理論 …… 289
8.1.2　グループウェア …… 293

8．2　ワークステーション／ビデオ会議 …… 303
8.2.1　ビューの共有 …… 303
8.2.2　Colab …… 307
8.2.3　MMconf …… 315
8.2.4　Rapport …… 323
8.2.5　Commune …… 331
8.2.6　CAVECAT …… 335
8.2.7　CRUISER …… 341
8.2.8　VideoWhiteboard …… 348

8．3　グループ意思決定支援システム …… 353
8.3.1　GDSS 全般 …… 353
8.3.2　SAMMの利用調査 …… 364
8.3.3　EMS …… 369
8.3.4　要求仕様作成段階におけるGDSS …… 377

8．4　オフィスプロシージャシステム …… 382
8.4.1　ECF …… 382
8.4.2　DOMINO …… 389
8.4.3　Action Workflow …… 391
8.4.4　EuroCoOp …… 397

8．5　メッセージ構造化システム …… 402
8.5.1　Information Lens …… 402
8.5.2　Object Lens …… 411
8.5.3　Oval …… 418

8．6　グループエディタ …… 425

8.6.1　Quilt …… 425

8.6.2　PREP …… 432

8.6.3　Flexible Diff …… 438

8.6.4　CoAUTHOR …… 443

8.6.5　DistEdit …… 449

8．7　設計議論システム …… 457

8.7.1　gIBIS …… 457

8.7.2　IBISを応用した設計議論モデル …… 466

8.7.3　SIBYL …… 470

8.7.4　Project Nick …… 476

8．8　協調作業システムの設計支援 …… 482

8.8.1　拡張SADT …… 482

8．9　ツールキット …… 488

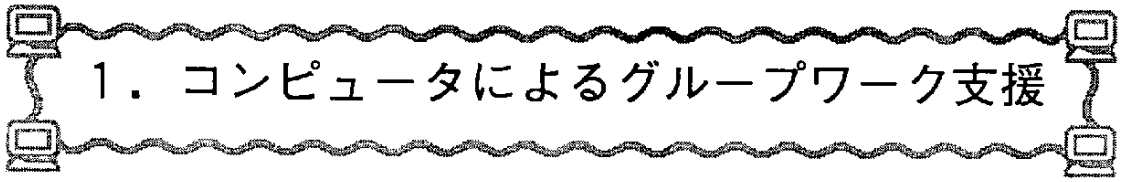
8.9.1　LIZA …… 488

8.9.2　Strudel …… 493

8．10　協調支援環境構築のためのアーキテクチャ …… 500

8.9.1　統合型グループ支援環境（IGSE） …… 500

8.9.2　Mocca プロジェクト …… 506



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]。

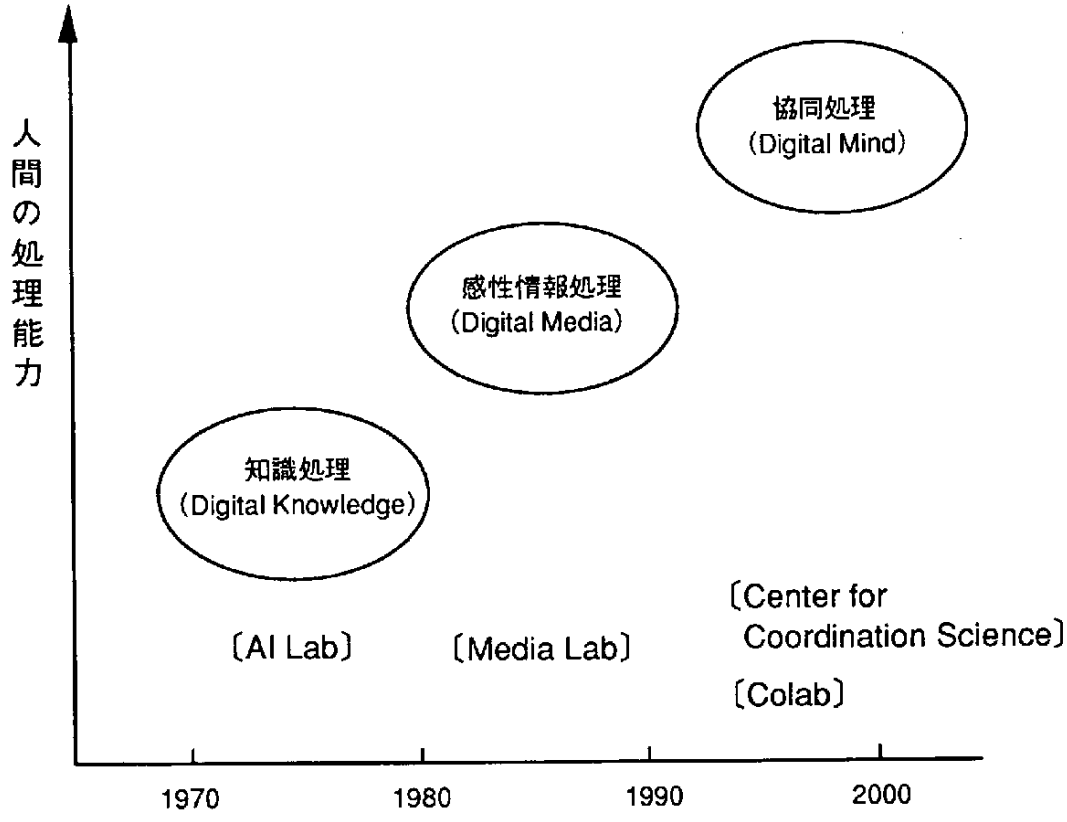


図1.2-1　人間の処理能力の追求

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。

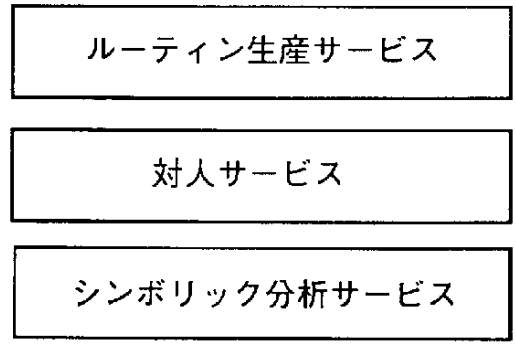


図1.3-1 新しい職業分類（R.Reich :" ザ・ワーク・オブ・ネイションズ " より）

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。

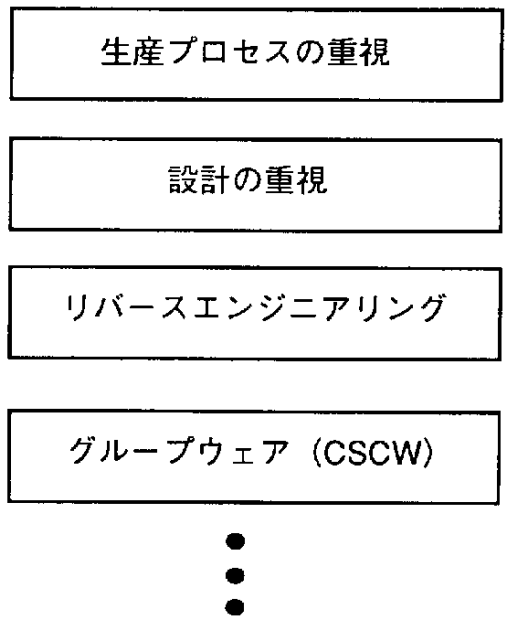


図1.4-1　ソフトウェア工学の先進性

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】

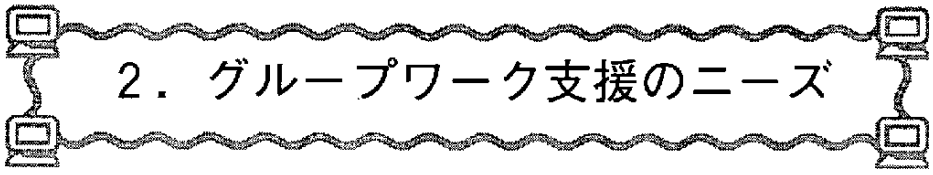
[Brand87] Brand, S.: The Media Lab:Investigating the Future at MIT, Viking (1987) (室、麻生共訳: メディアラボ, 福武書店(1988)).

[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。

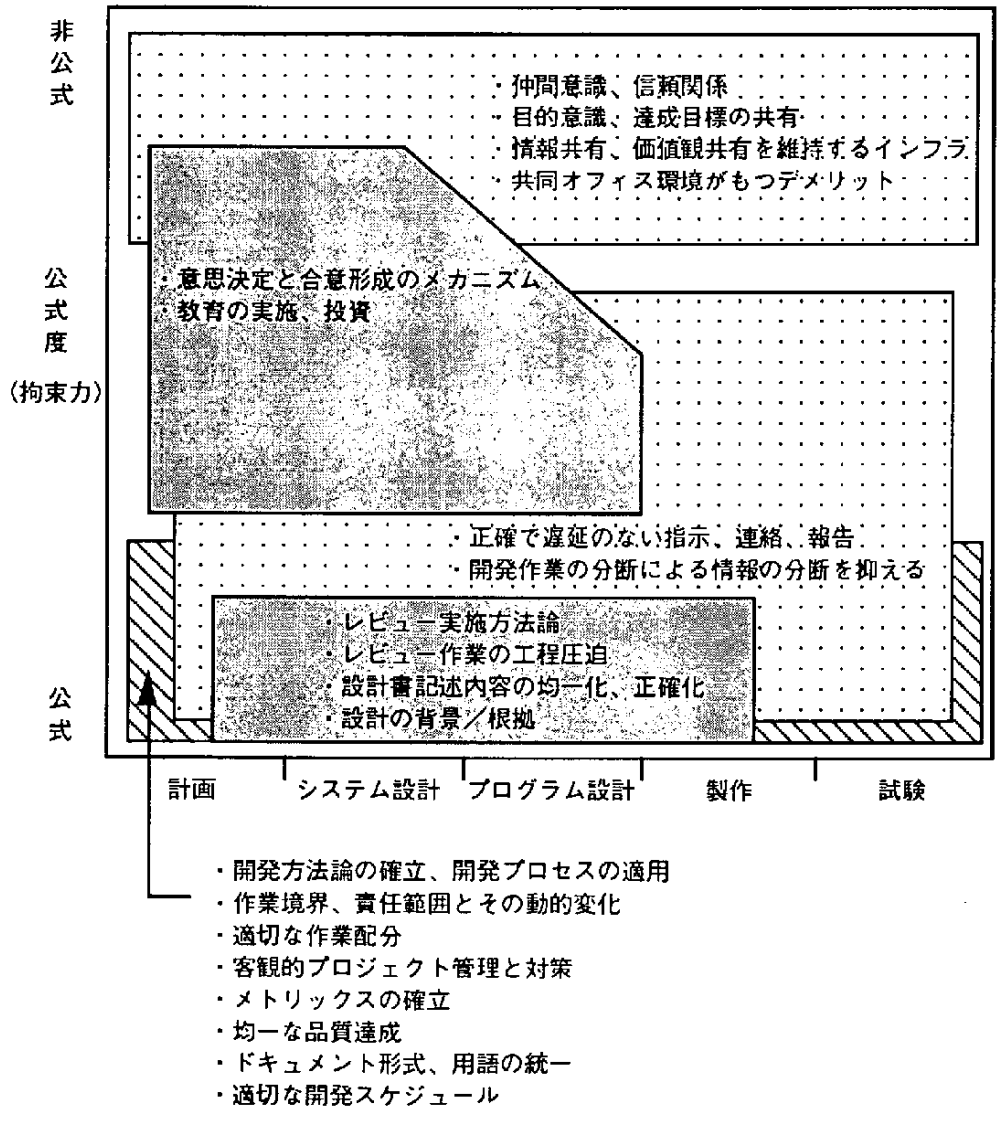


図2.2-1 公式度と開発工程図

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。

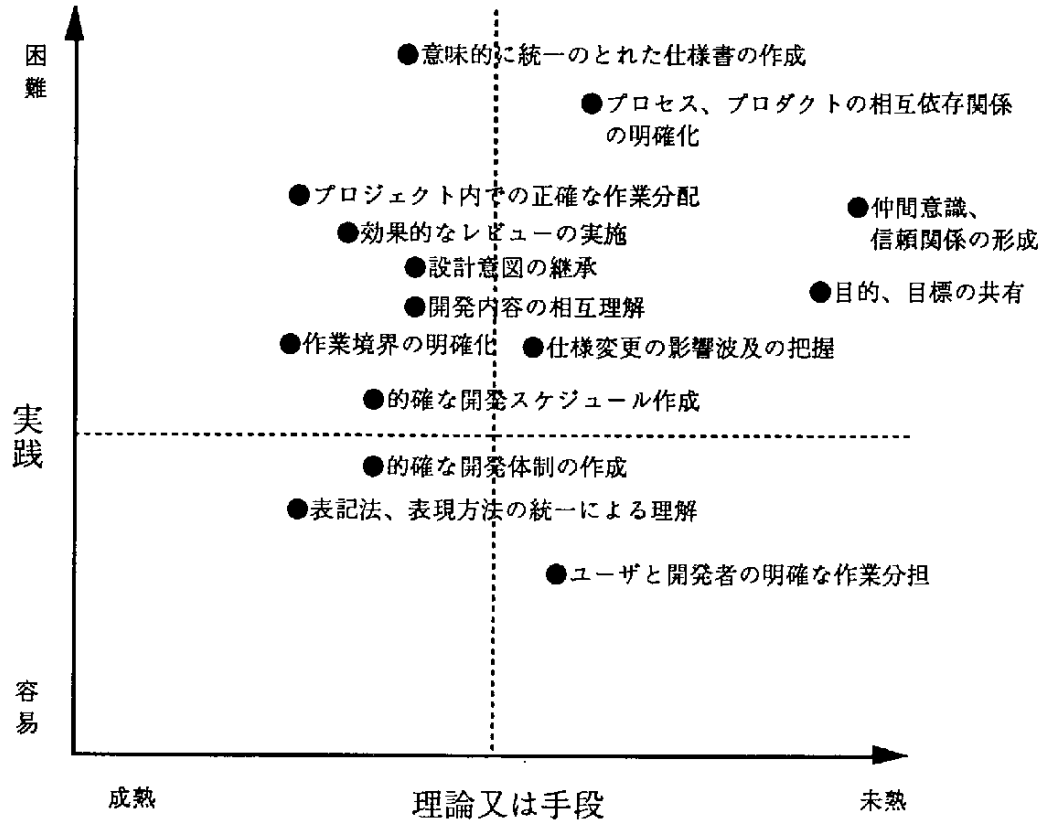


図2.2-2　理論の成熟度と実施の難易度

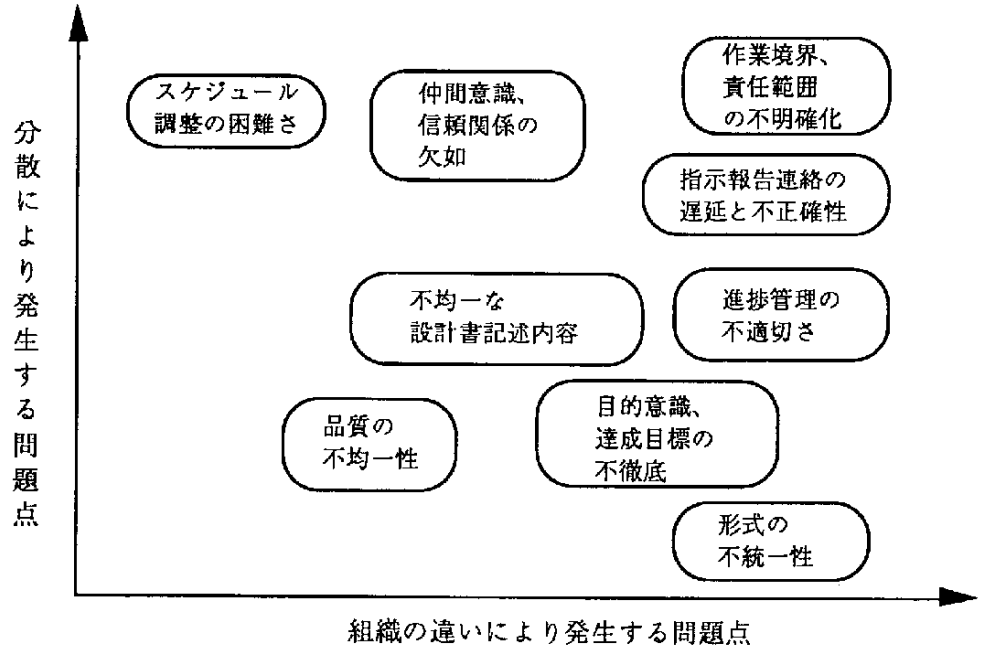


図2.2-3　分散と組織の違い

(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。

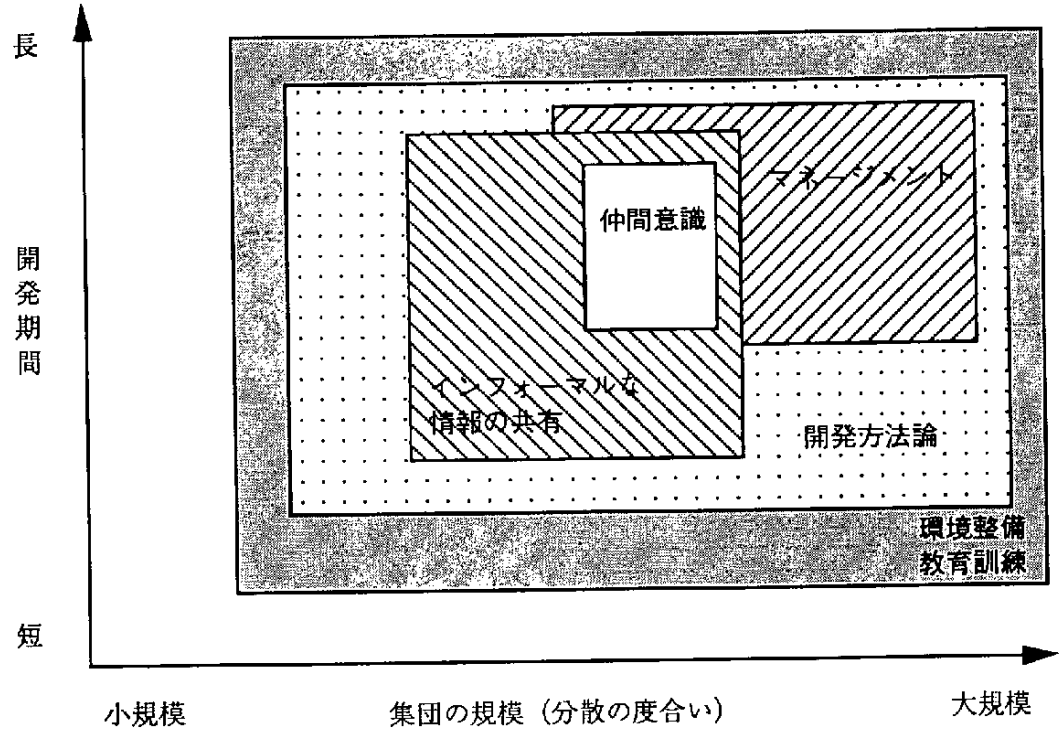


図2.2-4 開発期間と集団の規模

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。

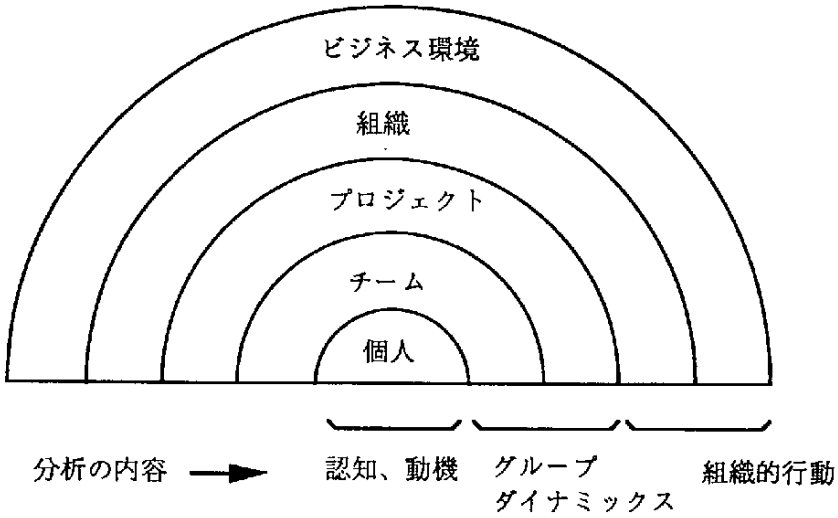


図2.2-5 階層化された行動モデル

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。

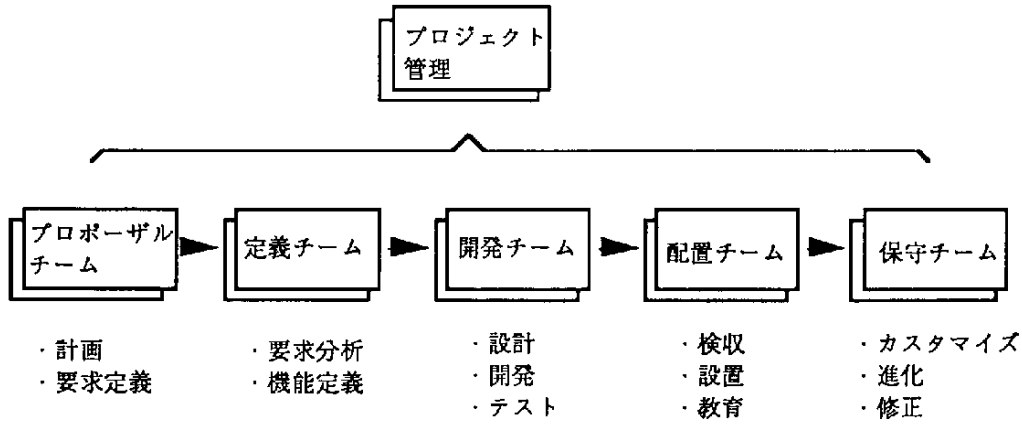


図2.2-6　開発ライフサイクルの工程とチーム

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。

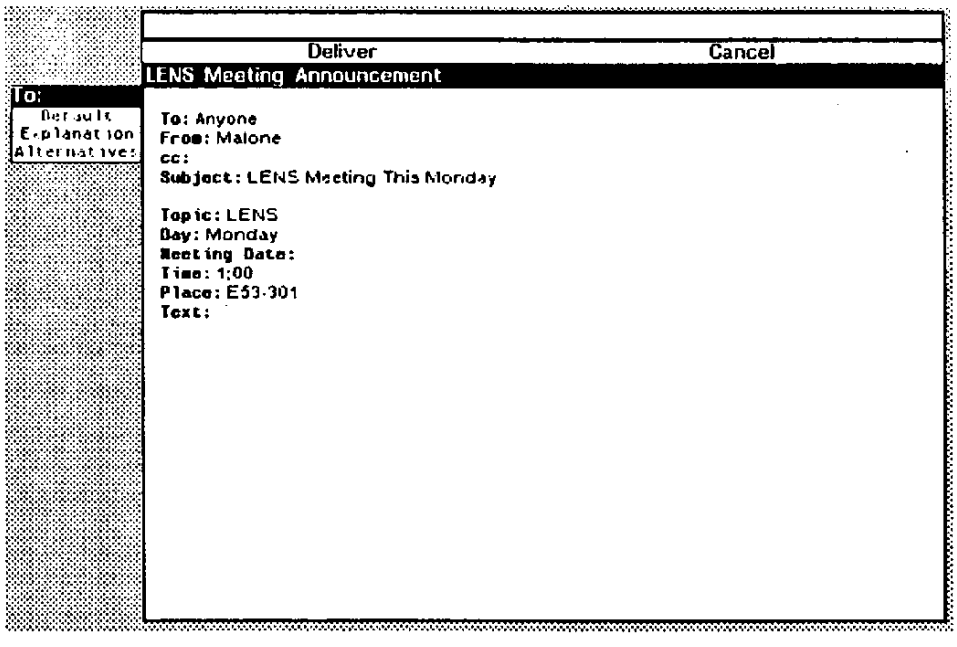


図3.1-1 半構造化メッセージのテンプレートの例

・　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。

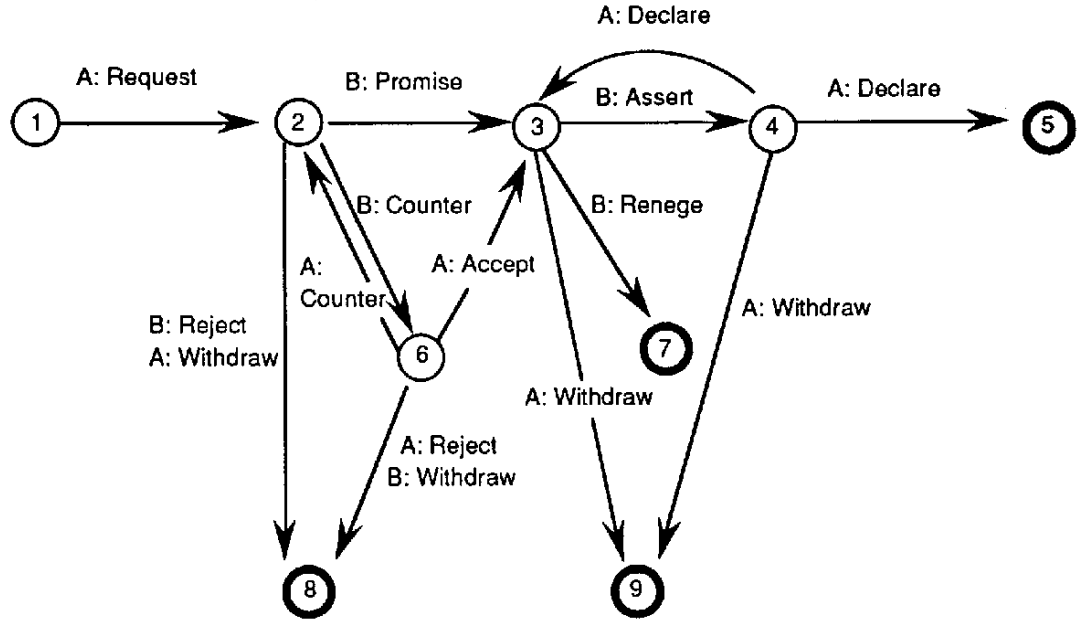


図3.1-3　会話モデル

(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

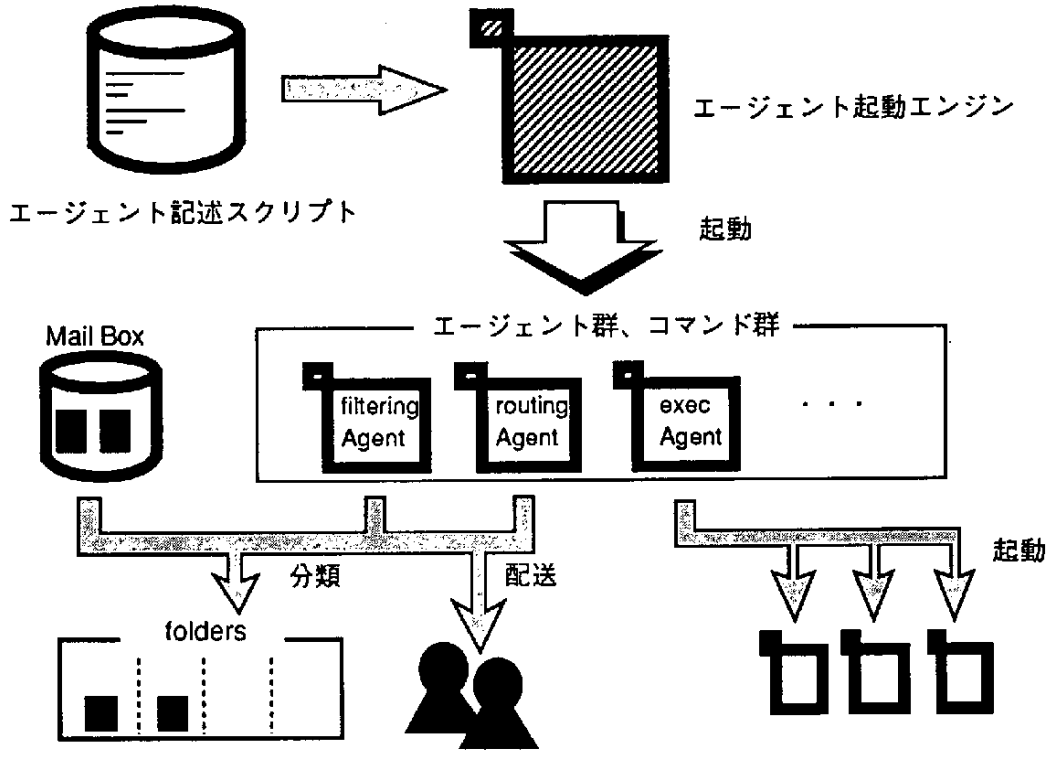


図3.1-4　D²/エージェントメールのシステム構成

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。

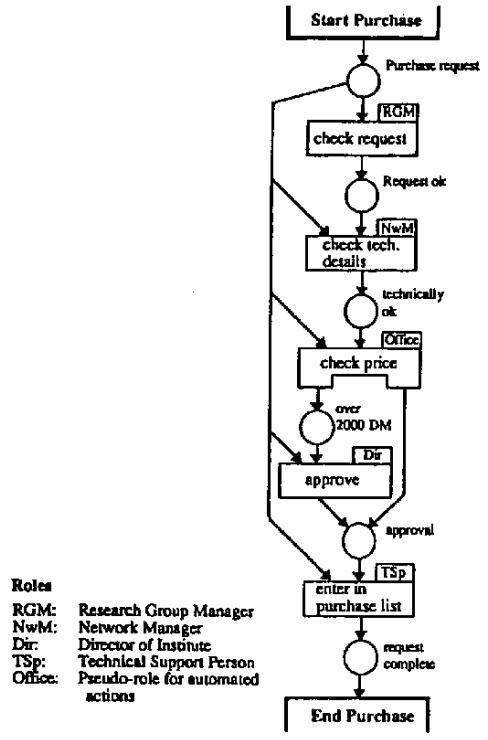


図3.2-1　DOMINOオフィスプロシージャの例

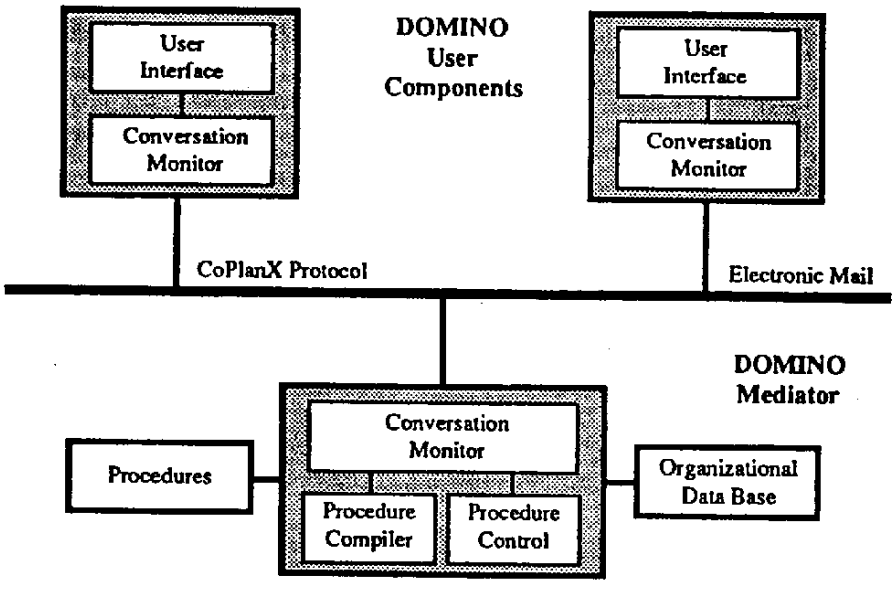


図3.2-2　DOMINOシステムの構成

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。

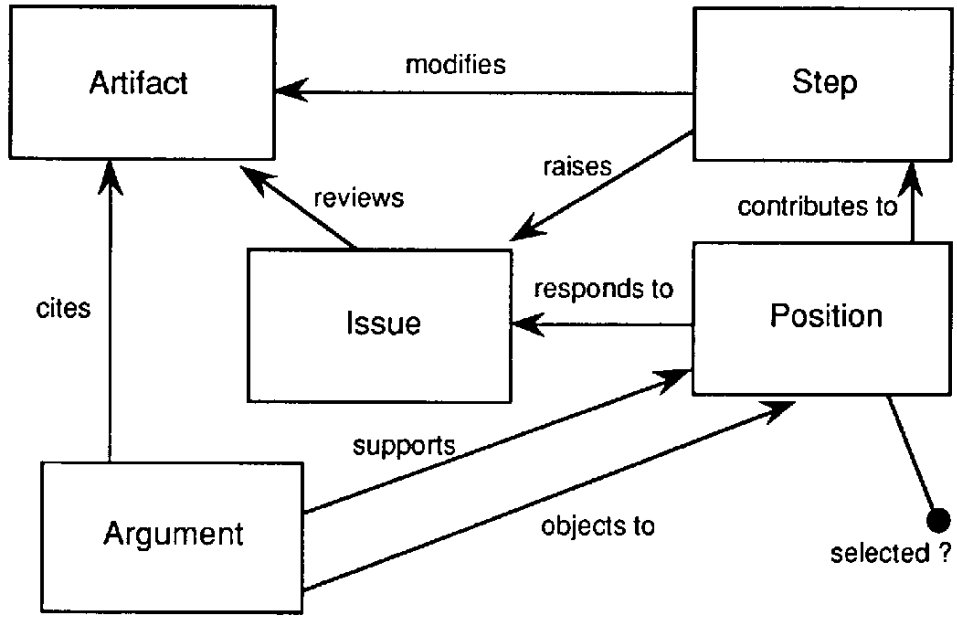


図3.4-1　Pottsの設計手法表現の汎用モデル[Potts89]

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】
- GROVE
- Collaborative Editing System
- Shared Book
- DistEdit

【非同期分散対話型】
- Quilt
- PREP

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。

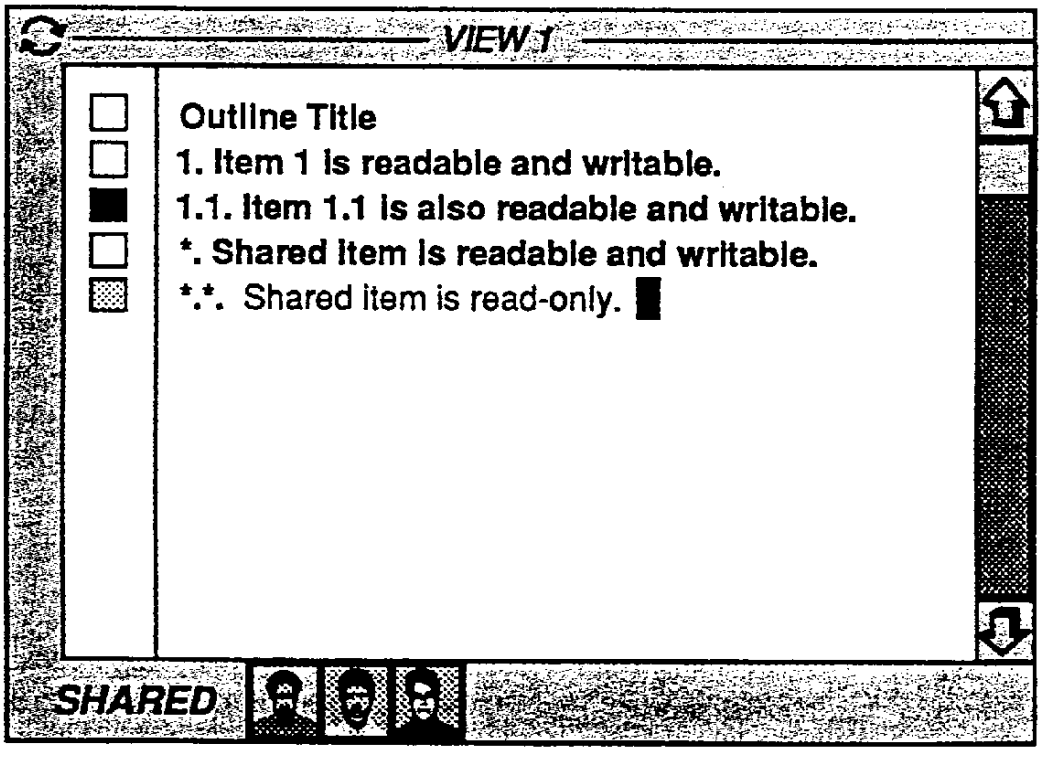


図3.5-1 GROVEのグループウインドウ

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。

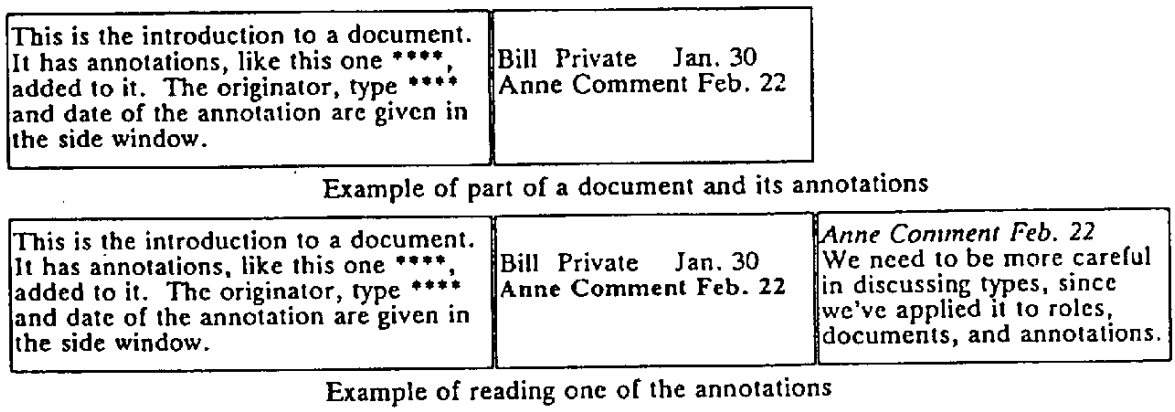


図3.5-2　基本文書と注釈の参照例

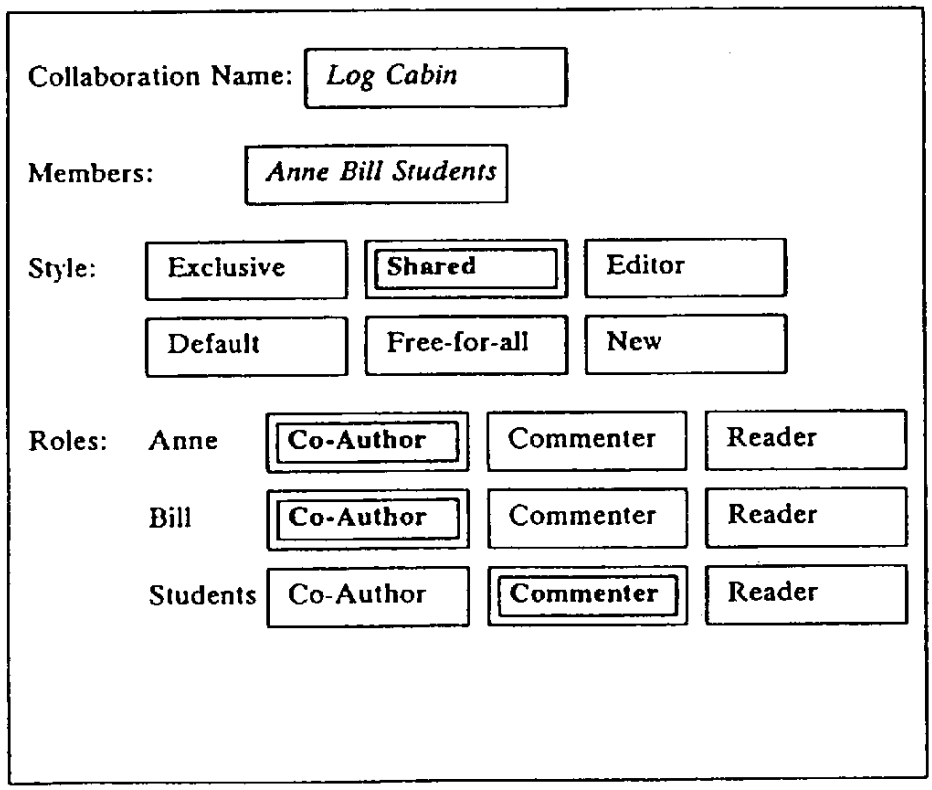


図3.5-3　協同作業のスタイルと役割の定義

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。

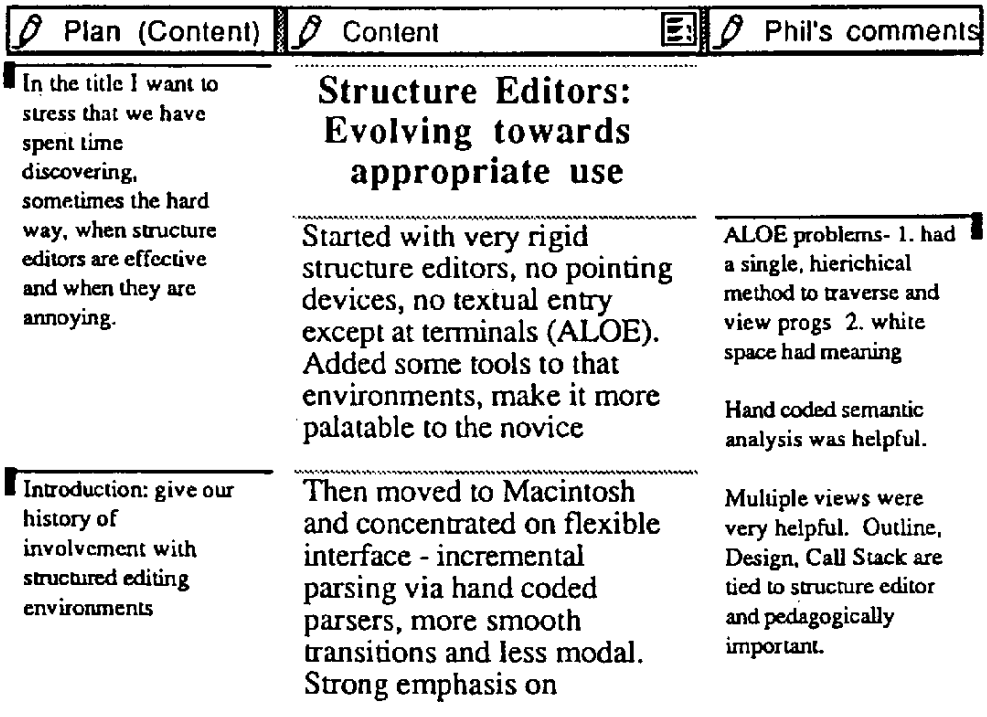


図3.2-4 三つのカラムをもつ草稿の例（PREPエディタ）

(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。

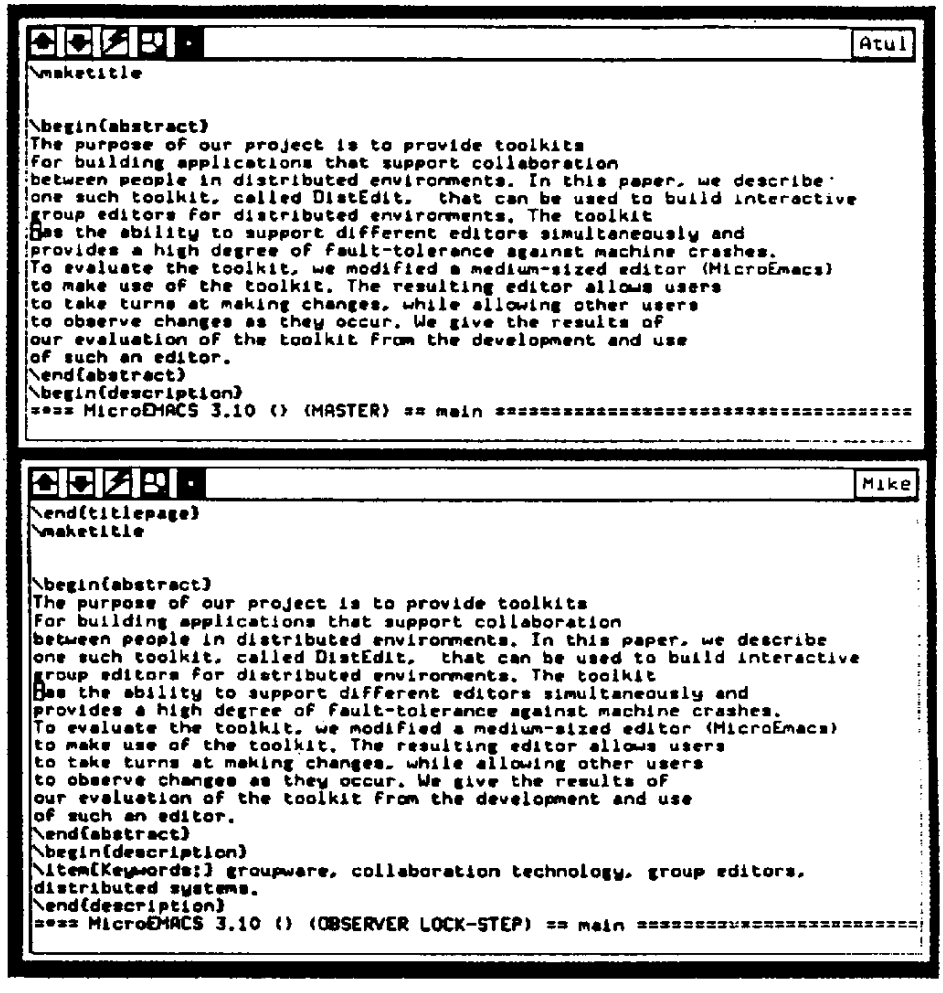


図3.2-5　DistEditにより実現されたグループ・セッション

(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。

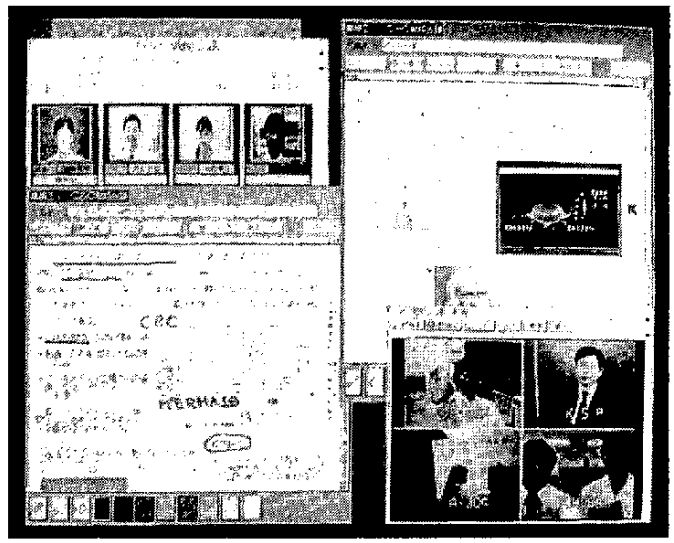


図3.7-1　MERMAID（日本電気）におけるテレポインティングの例
（全員が全員のポインタの位置を知ることができる）

- 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]。

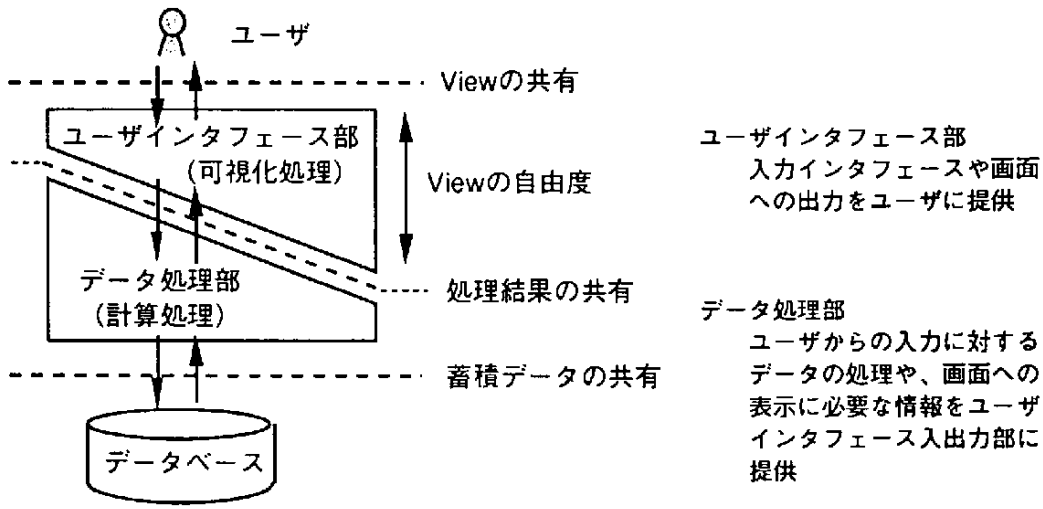


図3.7-2　グループAPの構造

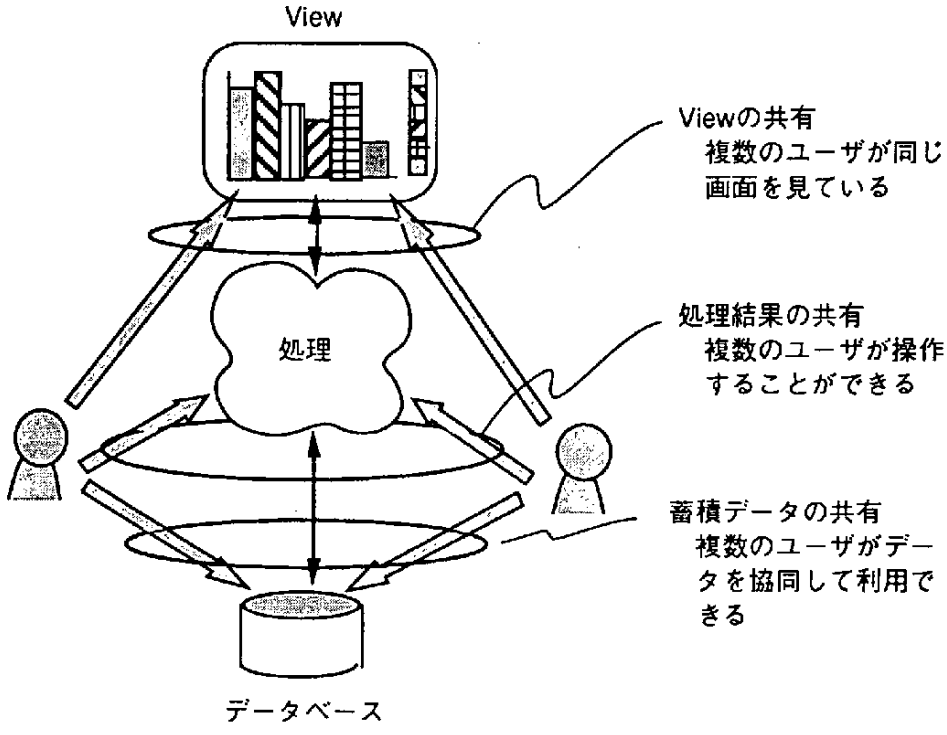


図3.7-3　グループAPにおける三つの共有

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

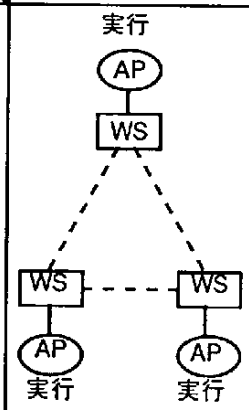
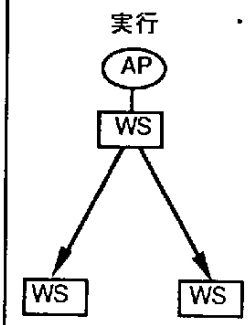
この制御は広い意味では(2) のビューの共有をも含むが、一般的にはビューの共有のみでは会議など使い方が限定される傾向にあるのに対し、処理結果の共有では上位の任意のアプリケーションが協調的に動作できるように支援することを目的とする。

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。

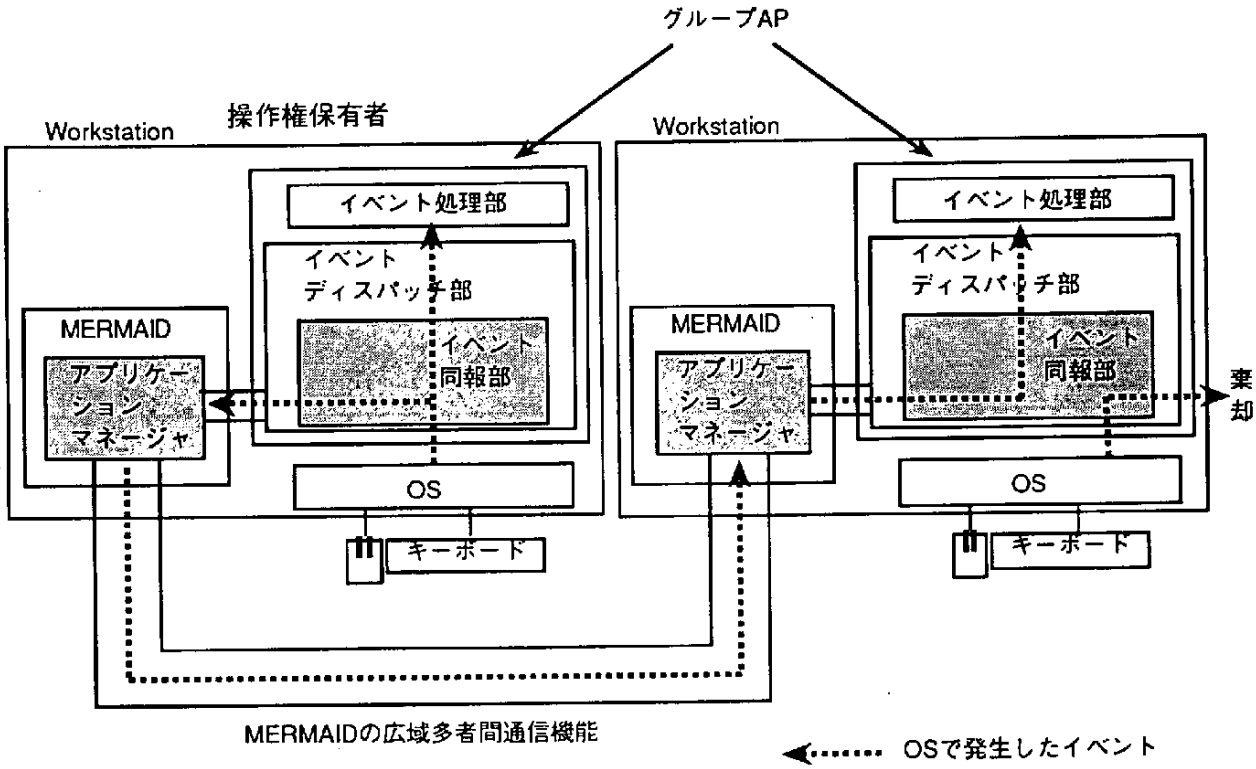


図3.7-4　分散制御によるAP共有方式（日本電気のMERMAIDの例）

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。

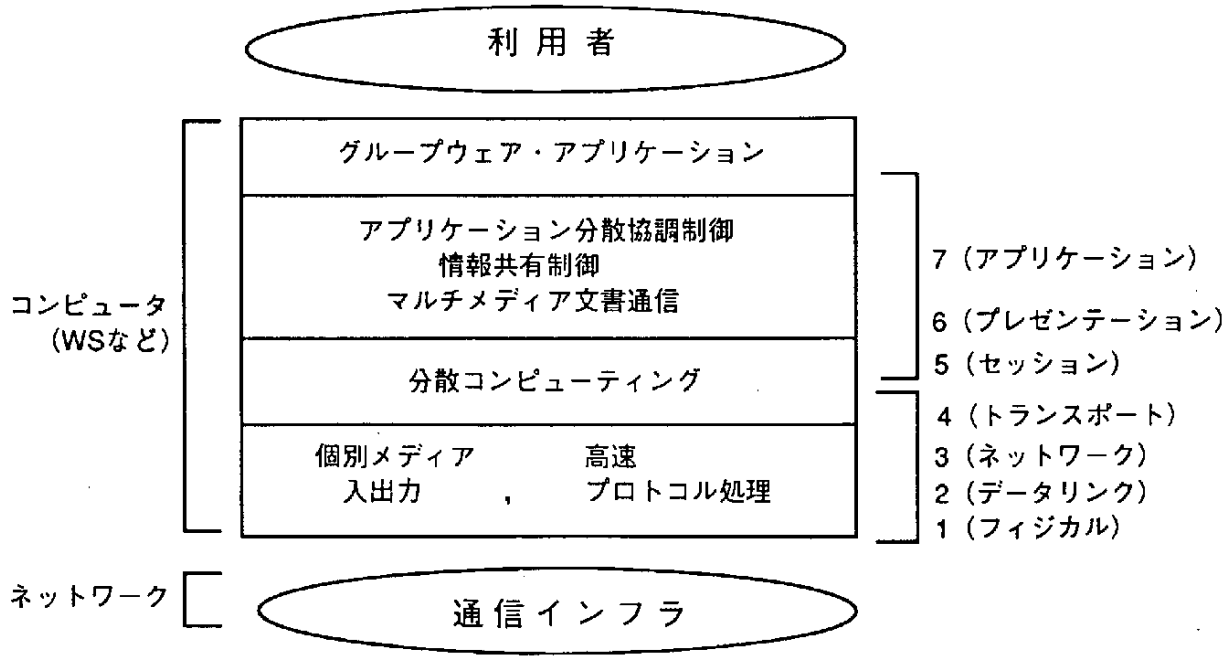


図4.1-1　グループウェアから見たマルチウェアネットワークのシステム体系

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。

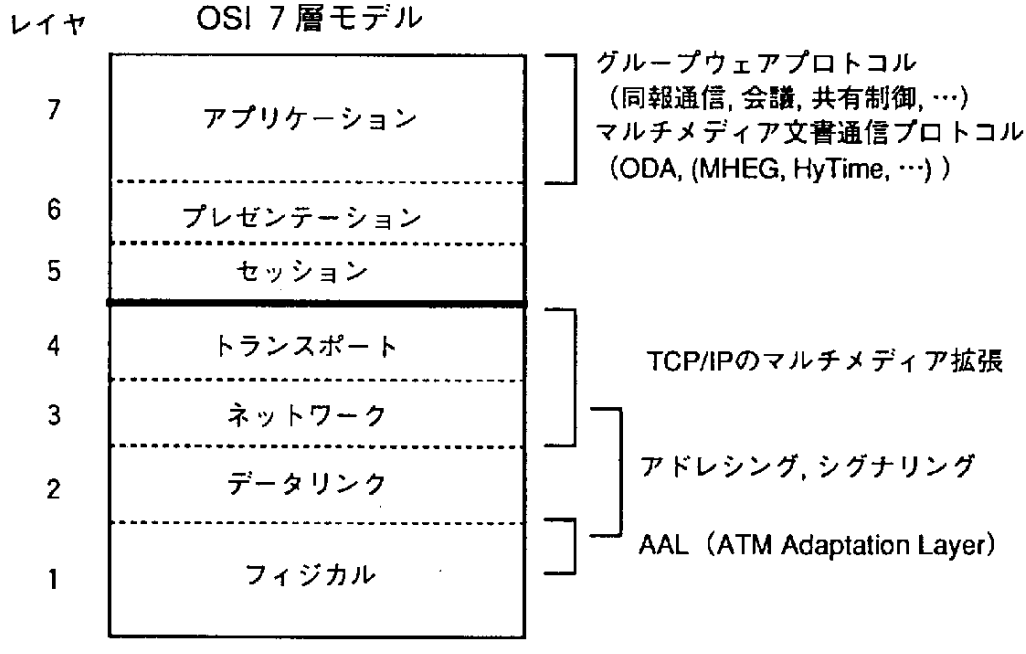


図4.1-2　マルチメディアプロトコル体系

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><td>クラス
パラメータ</td><td>A</td><td>B</td><td>C</td><td>D</td><td>X
(SEAL)</td></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

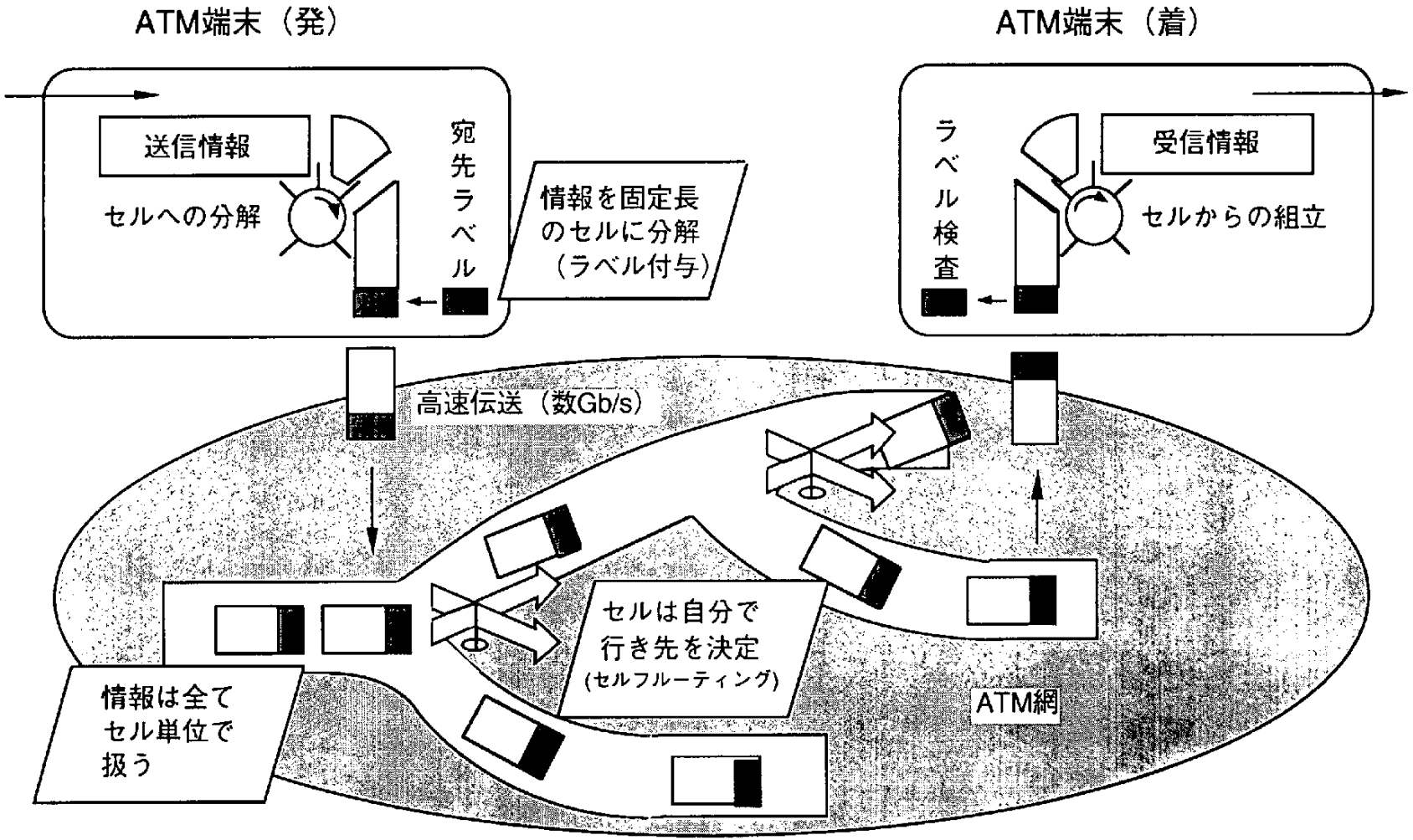

ATM端末（発）
送信情報
セルへの分解
宛先ラベル
情報を固定長のセルに分解（ラベル付与）
ATM端末（着）
ラベル検査
受信情報
セルからの組立
高速伝送（数Gb/s）
セルは自分で行き先を決定
(セルフルーティング)
情報は全てセル単位で扱う
ATM網

図4.1-3　ATMの原理

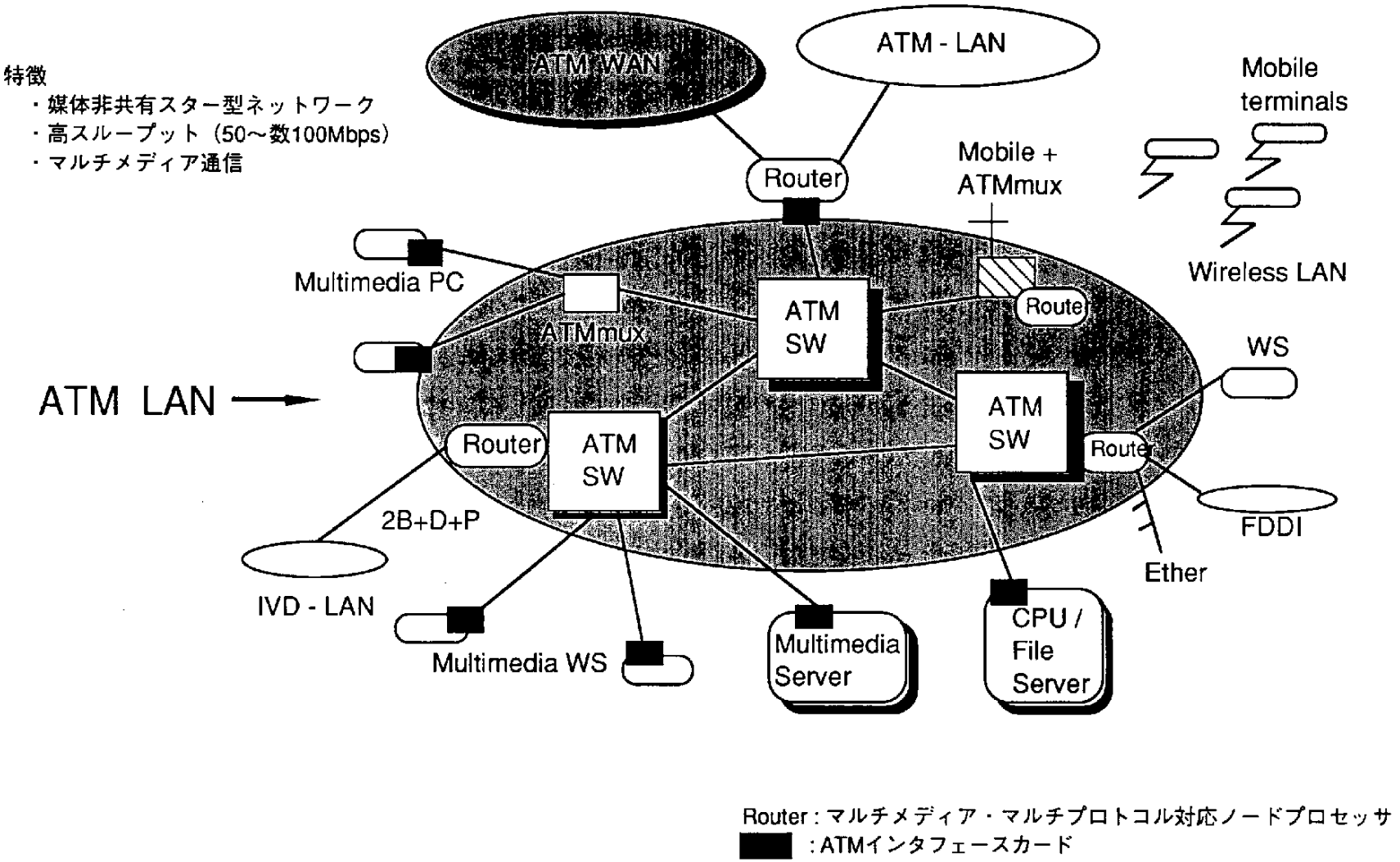


図4.1-4　ATM - LANの構成

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）。

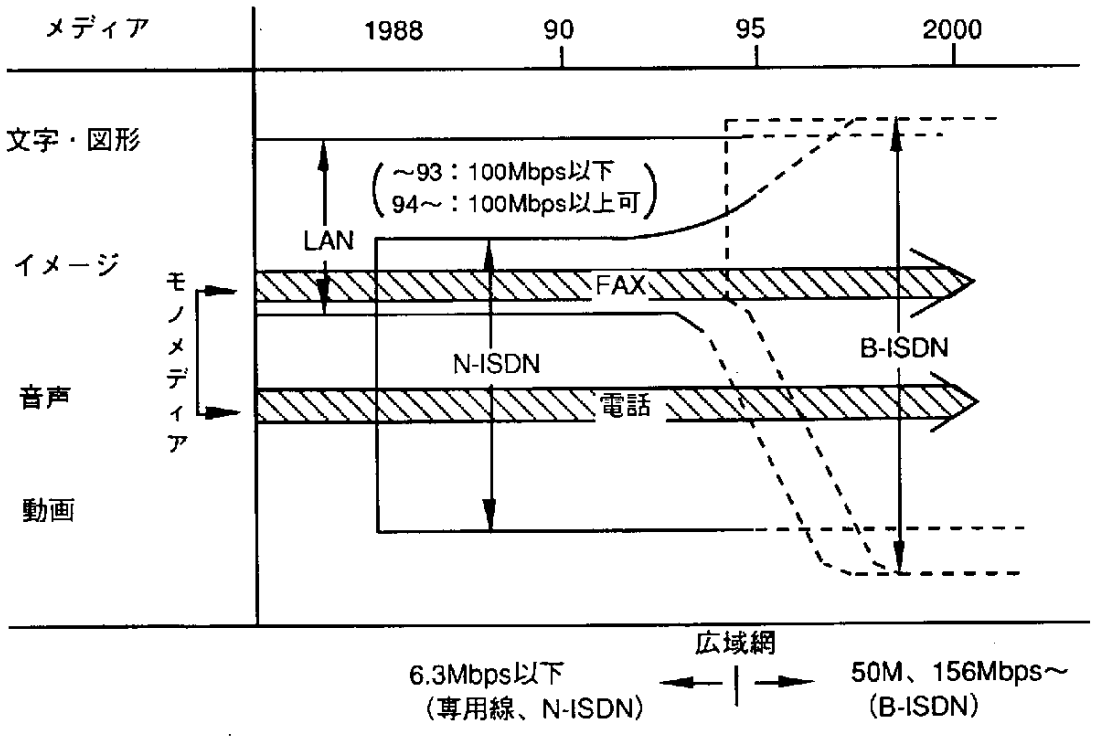


図4.1-5 情報メディアから見た通信環境の変遷

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。

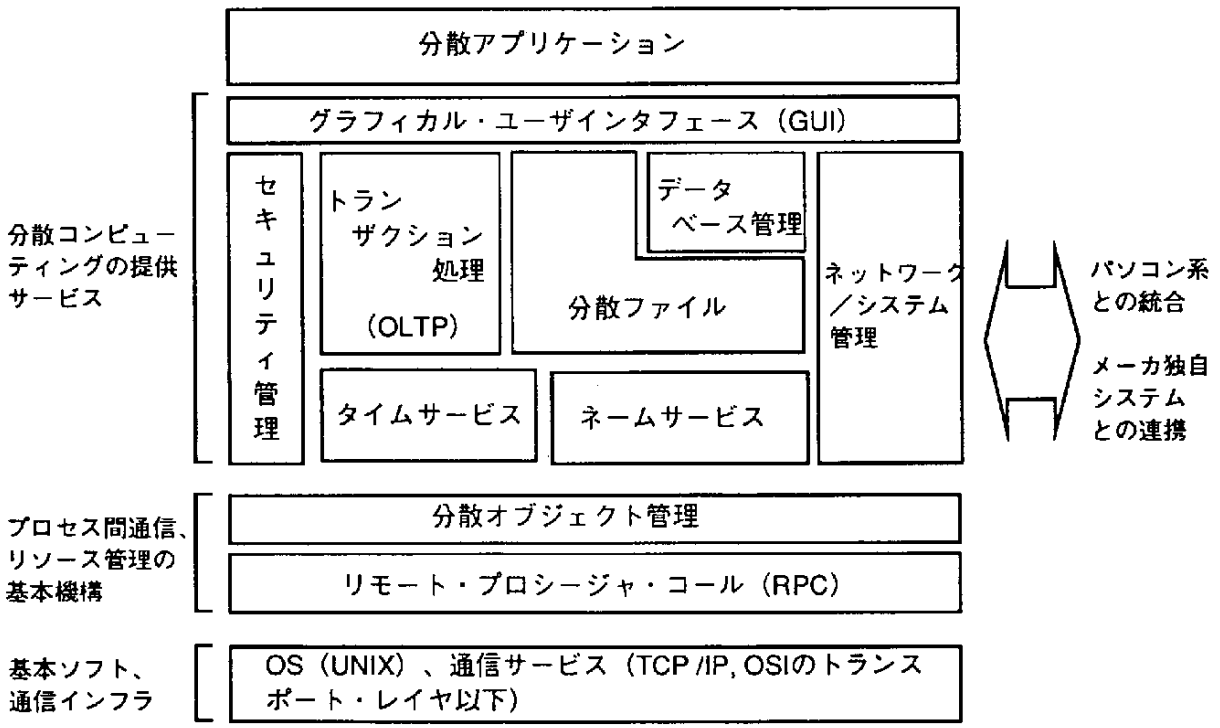


図4.1-6 分散コンピューティング・アーキテクチャ

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。

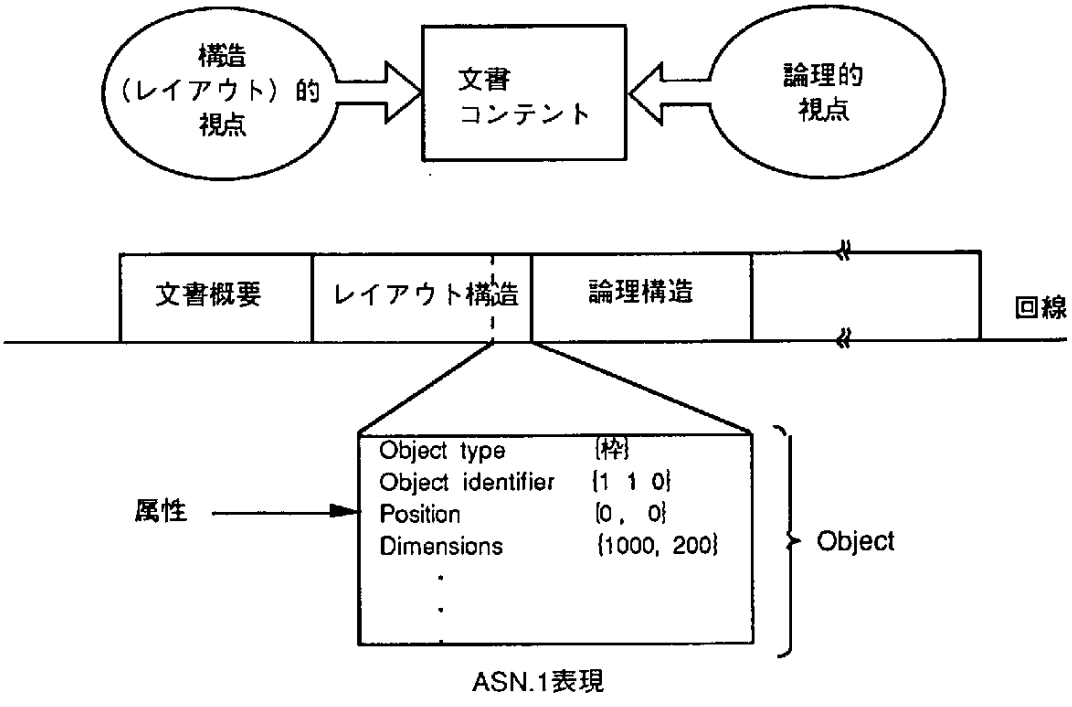


図4.1-7　文書構造とODIFデータストリーム例

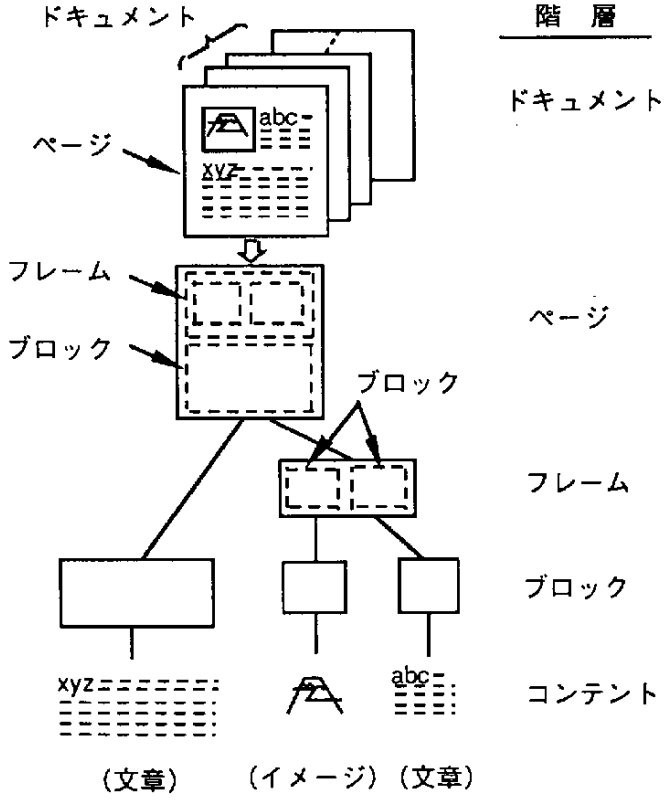


図4.1-8　ODAにおけるマルチメディア文書のレイアウト構造の例

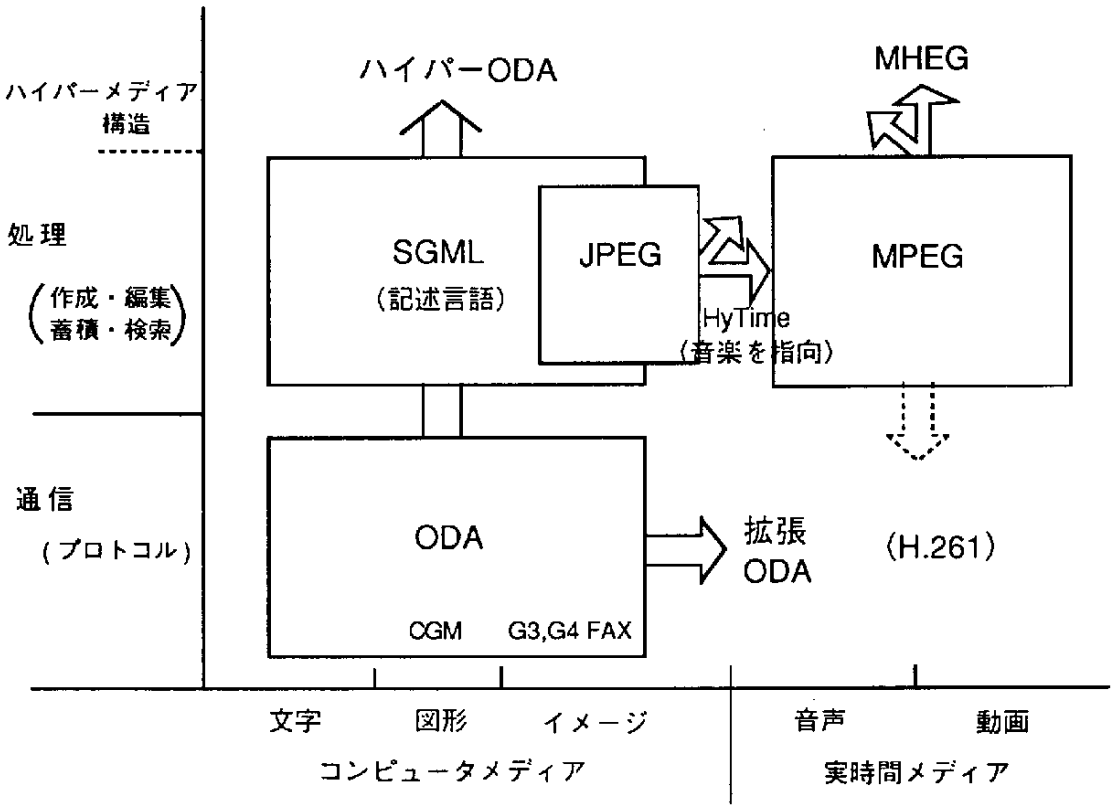


図4.1-9　マルチメディア情報表現・プロトコル標準化動向

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。

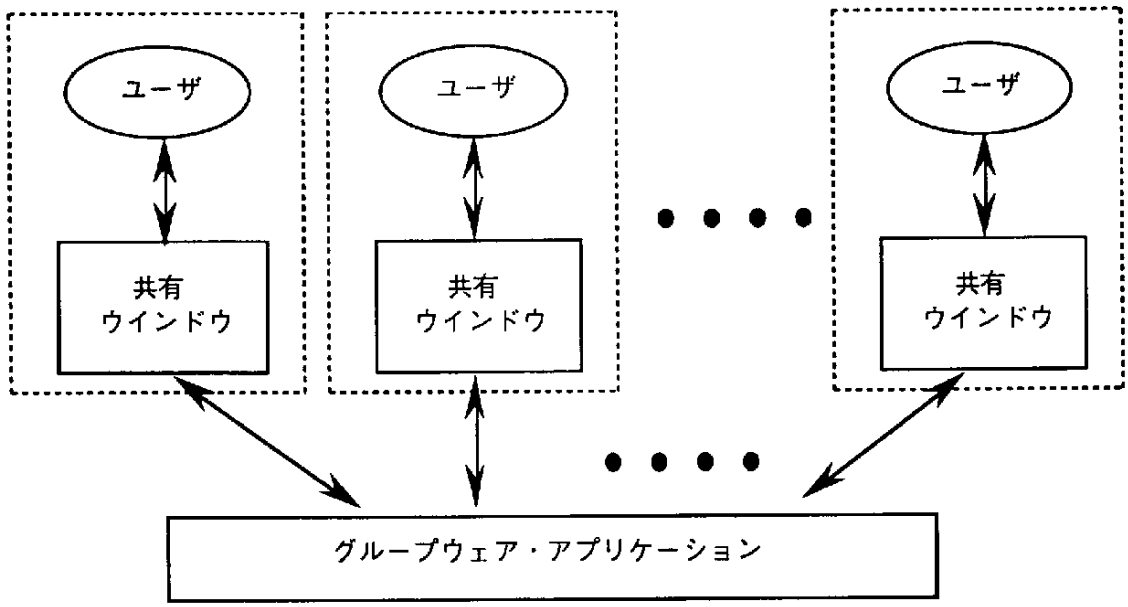


図4.2-1　グループウェアにおける共有ウインドウ機能

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。

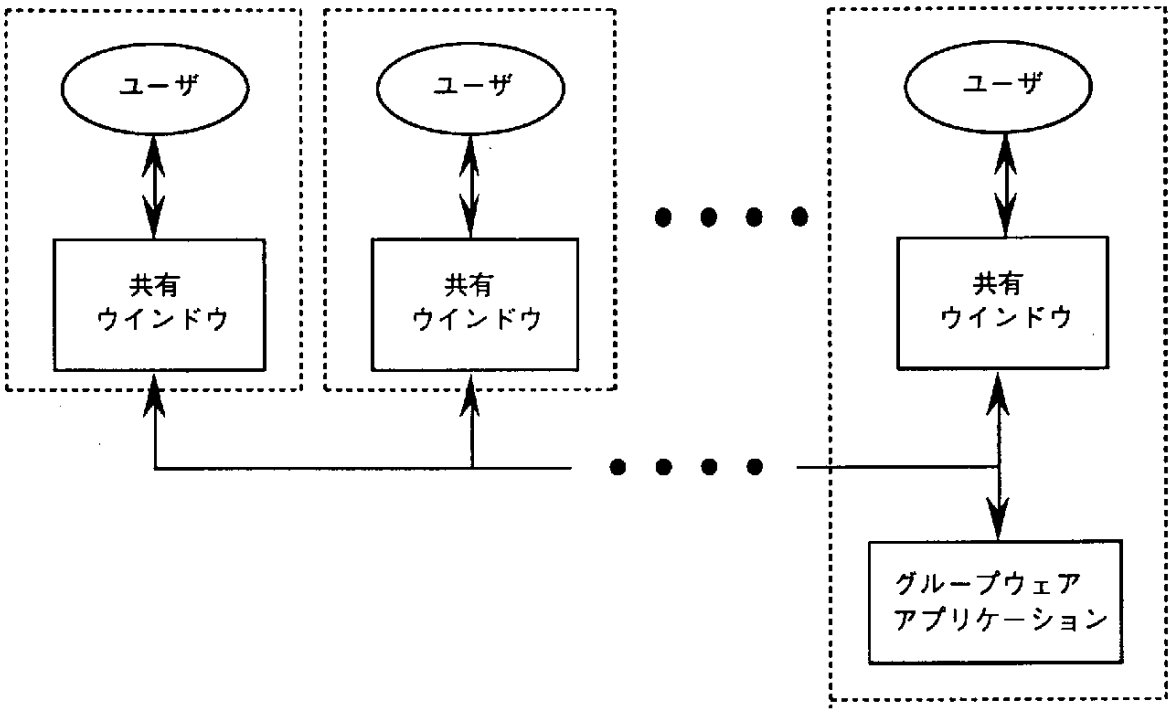


図4.2-2(a)　集中型実行形態

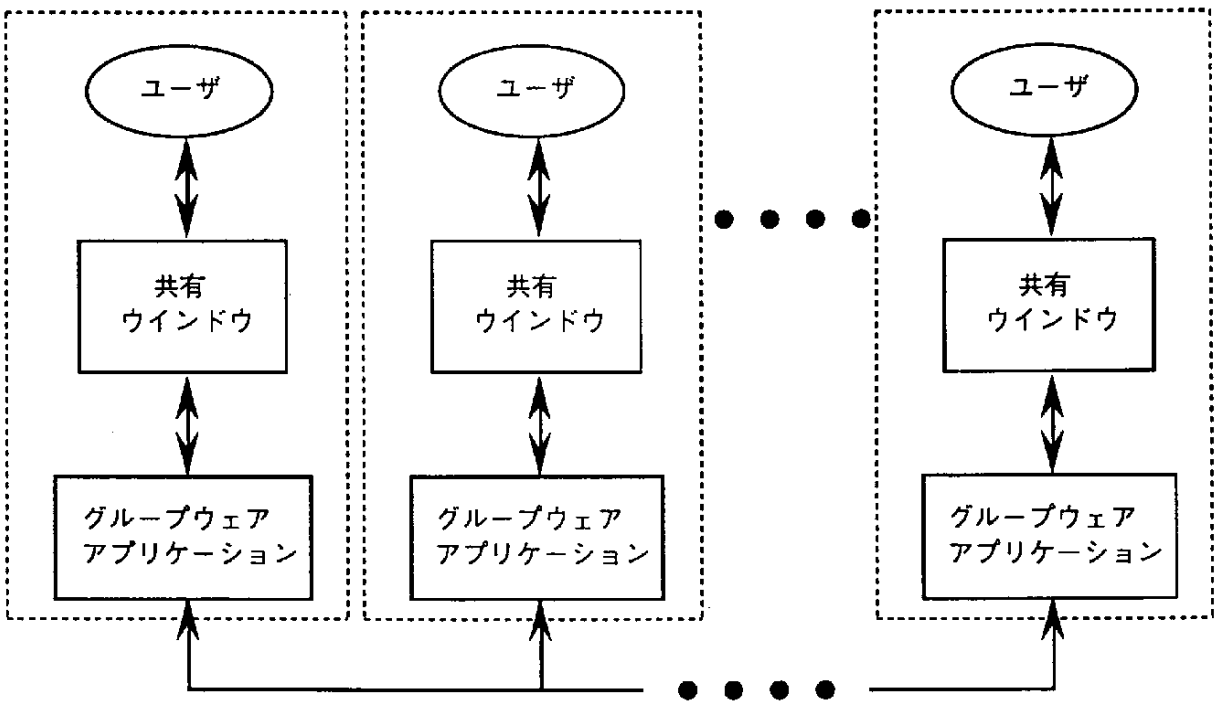


図4.2-2(b)　分散型実行形態

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

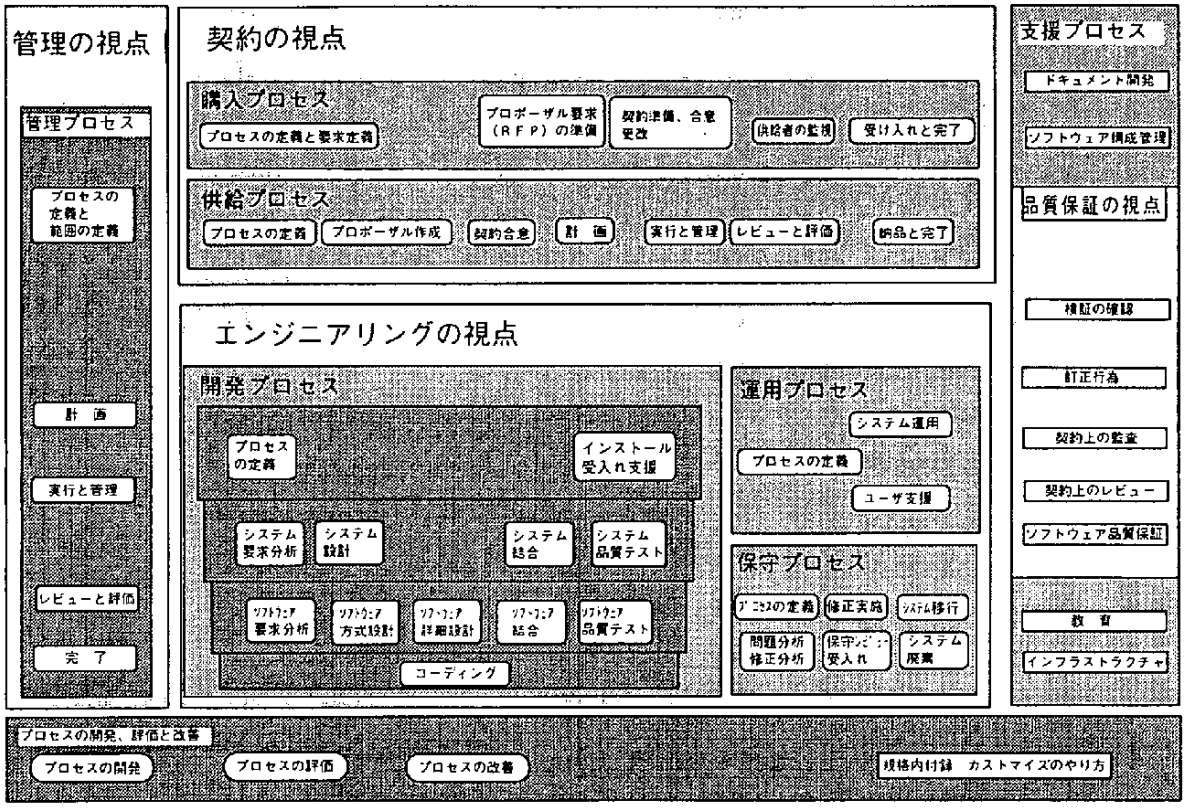


図5.1-1 ISO/IEC JTC1/SC7委員会案:ソフトウェアライフサイクルプロセスの概要

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。

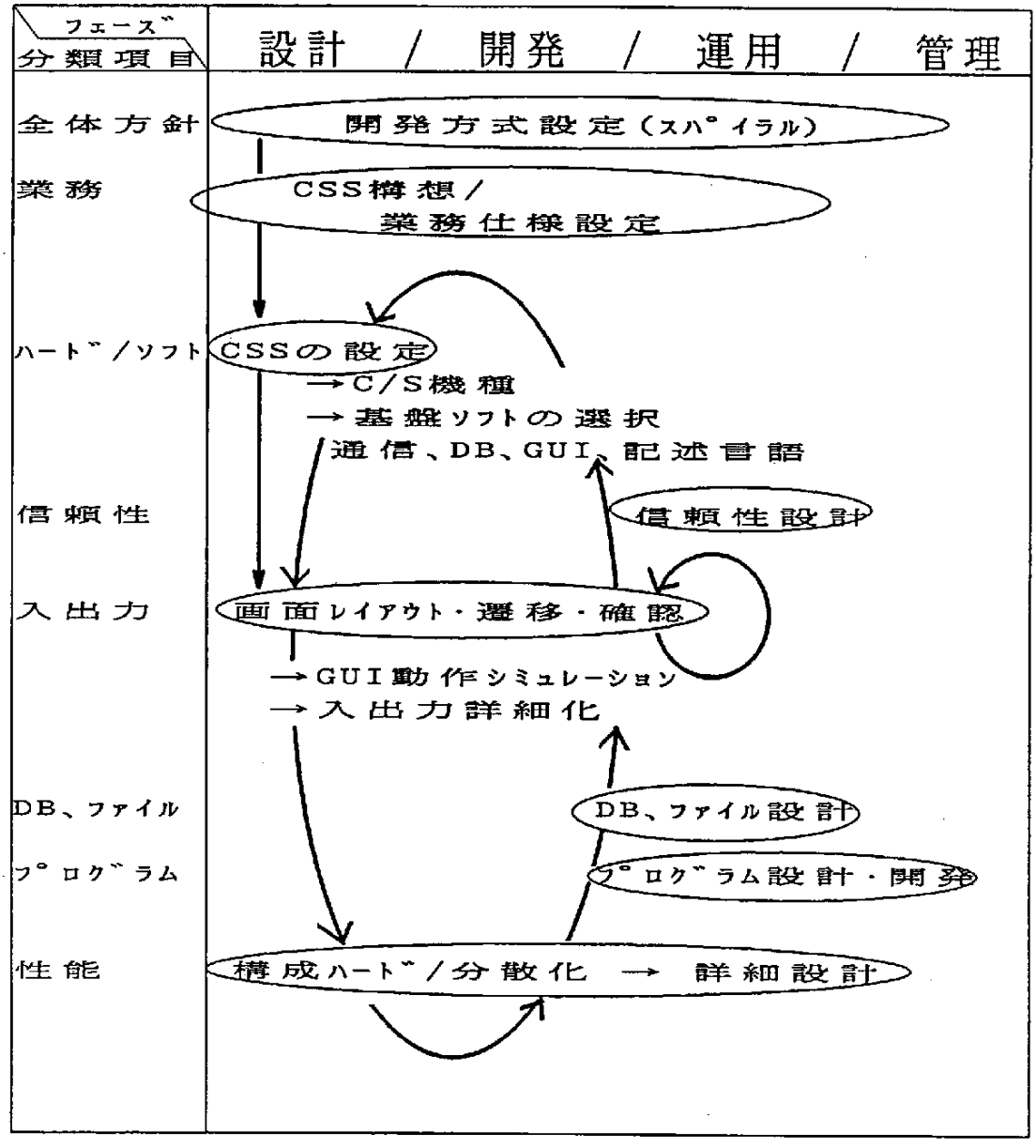


図5.1-2 CSS用HIPACEの流れ

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	Progreming 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動向の監視
B	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記述書計画	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	アプリケーション	パッケージの調査	アプリケーションの立案	アプリケーションの運用検討	アプリケーション概要設計	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	ホスト		処理機能構成の概要選定	処理機能構成の決定	開発処理の導入	開発処理の運営	[PS～IT]	運用処理の導入 [PG～PT]	運用処理の運営	[IT～ST]		
	ワークステーション		ワークステーション機能構成の概要選定	ワークステーション機能構成の決定	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]		ハードウェアの保守・グレードアップ
	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	[OT～ME] [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II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	OT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～ME]				
	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

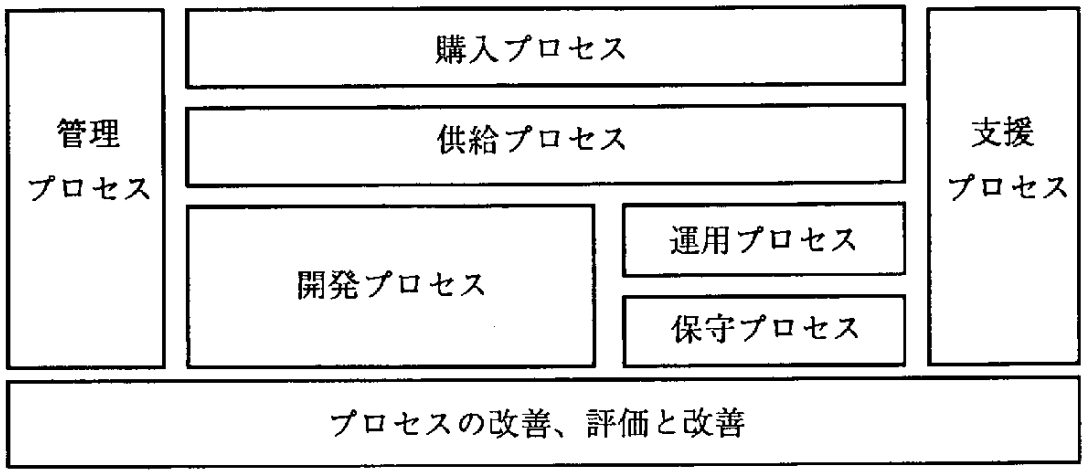


図5.3-1　ISO案：ソフトウェアライフサイクルプロセス

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、という疑問が涌いてくる。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－ソフトウェア・オブジェクトベース構築の立場から－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。

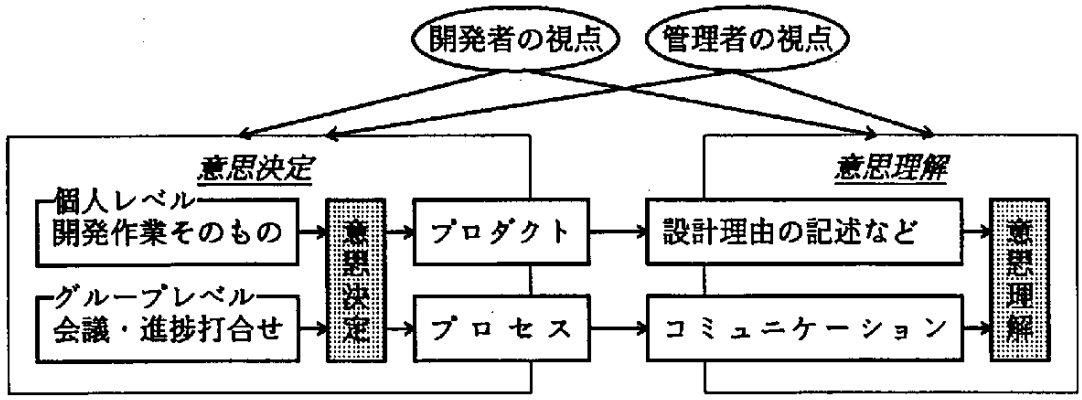


図5.5-1 ソフトウェア開発における意思決定

(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。

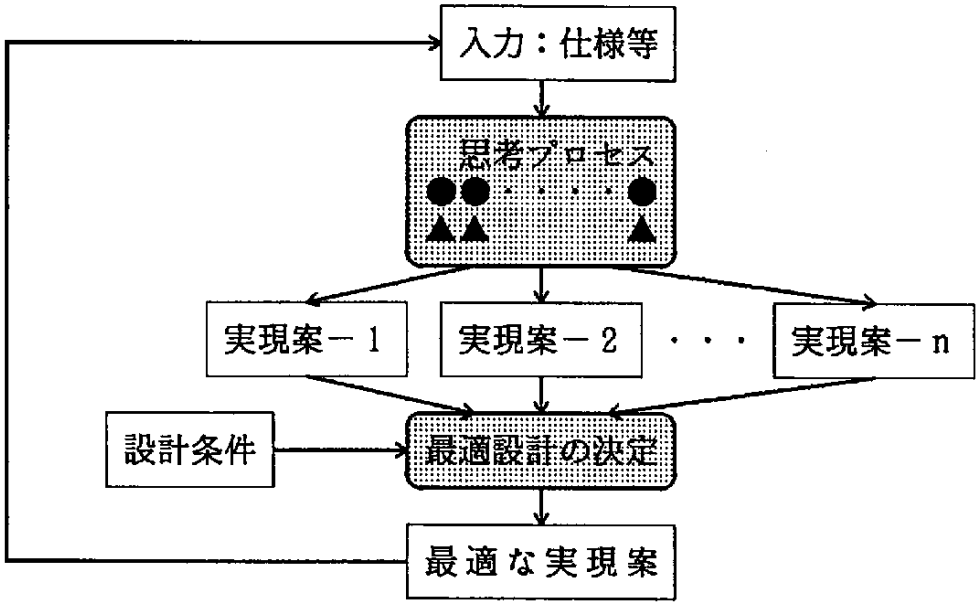


図 5.5-2　ソフトウェア設計における意思決定のプロセス

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。

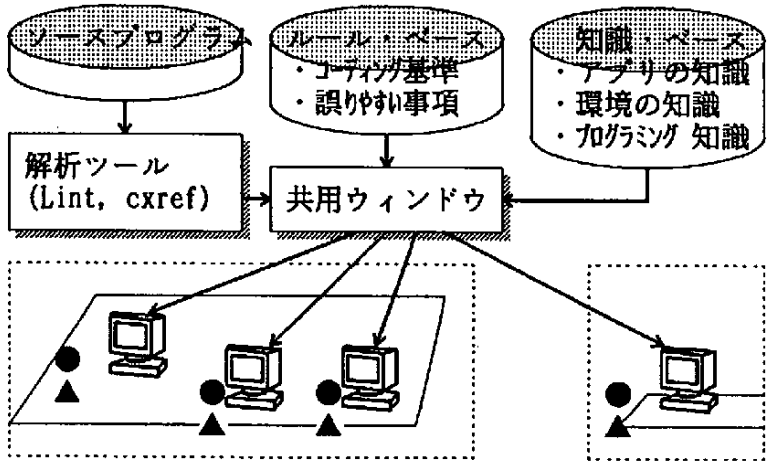


図 5.5-3 ICICLEのアーキテクチャ

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。

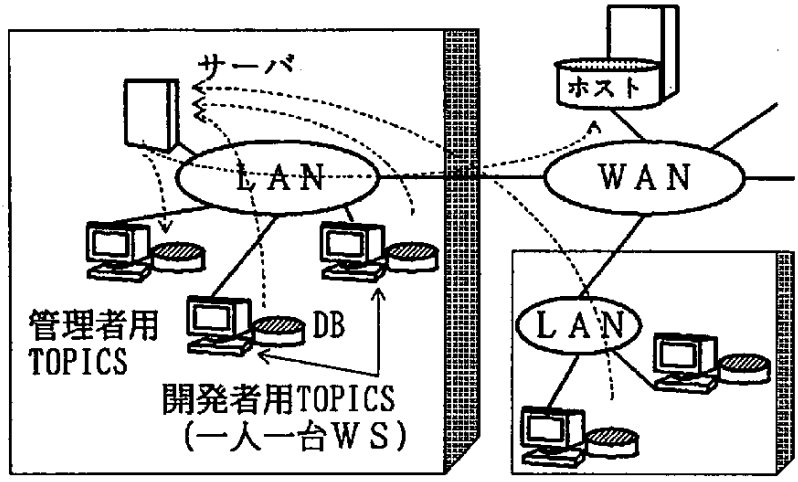


図 5.5-4　TOPICSのアーキテクチャ

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。

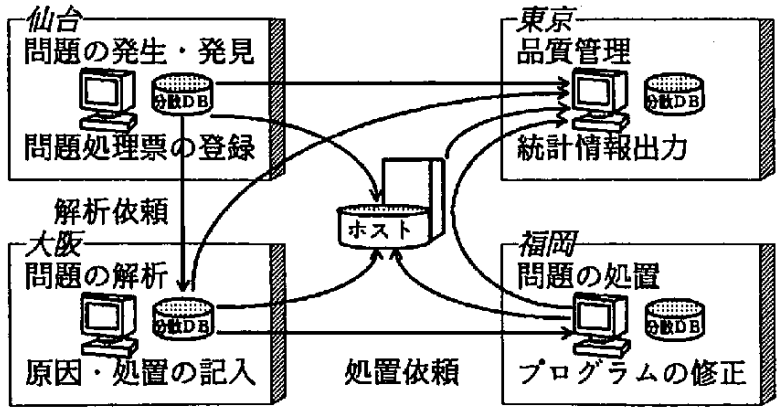


図 5.5-5　問題処理票の処置プロセス

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。

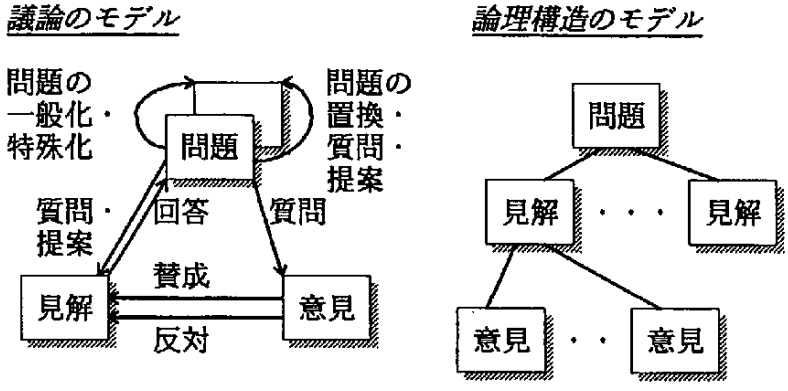


図 5.5-6 IBISのモデル

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。

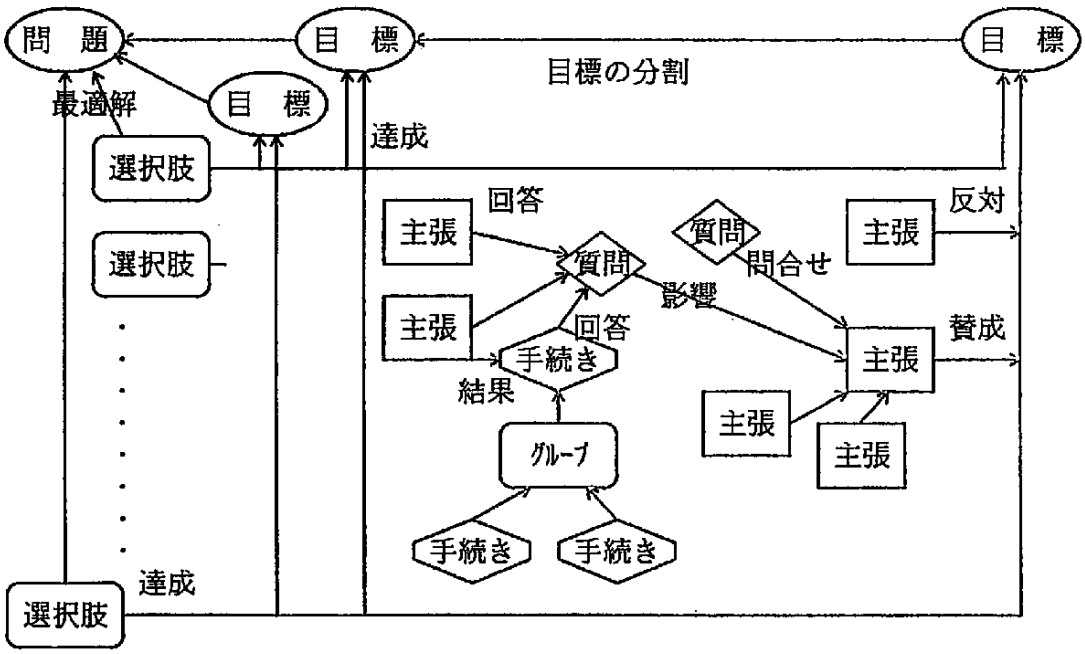


図 5.5-7 SIBYL の議論モデル

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。

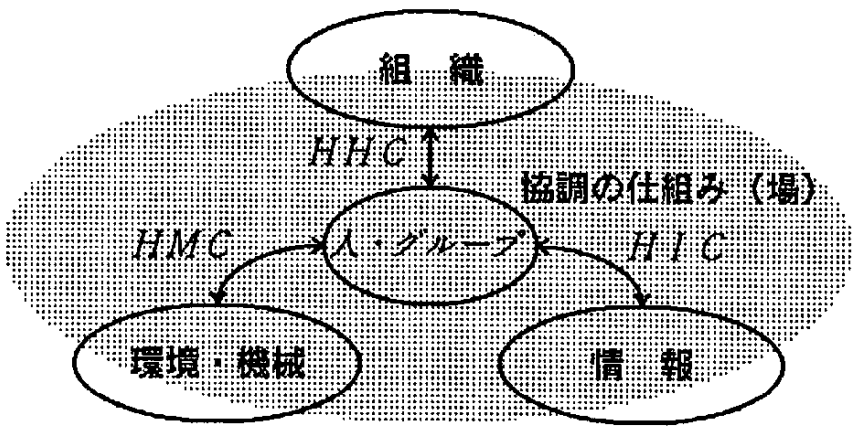


図 5.6-1 協調作業支援における三つのインターワーク

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。

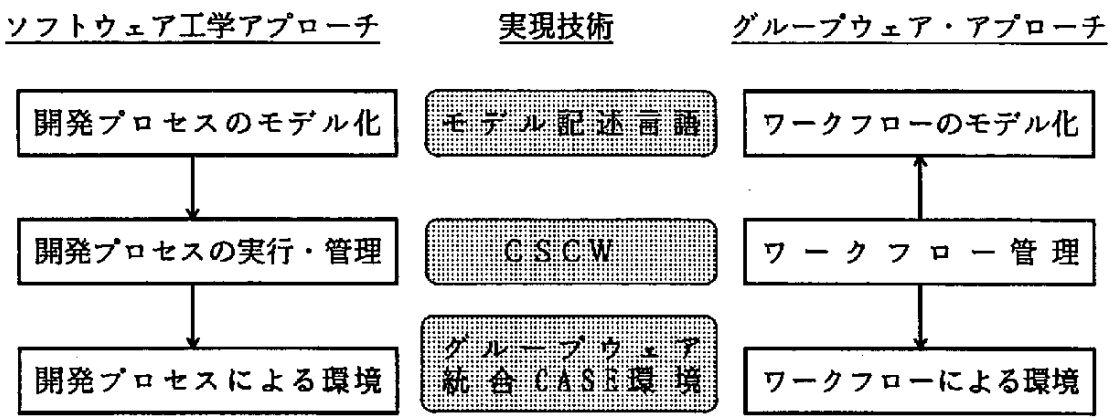


図5.6-2　ソフトウェア開発におけるワークフロー管理へのアプローチ

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。

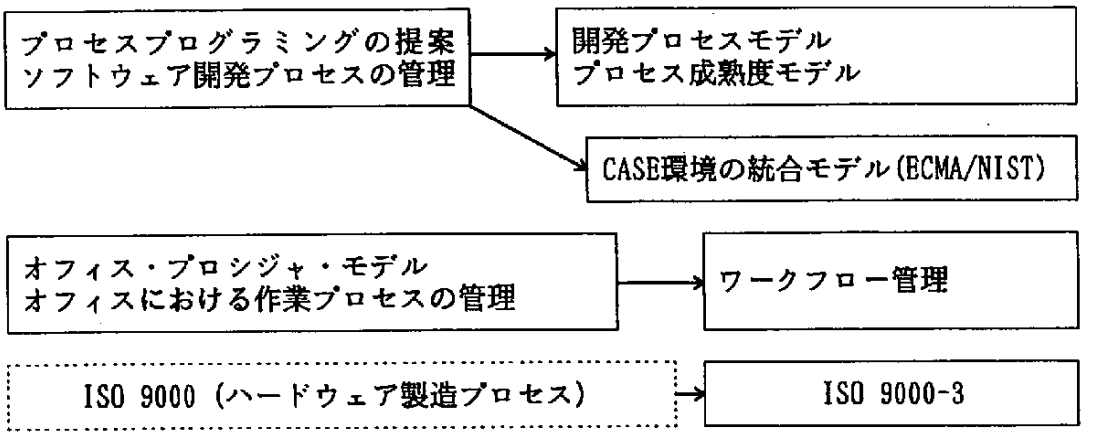


図 5.6-3 ソフトウェア開発におけるプロセス概念の発展

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。

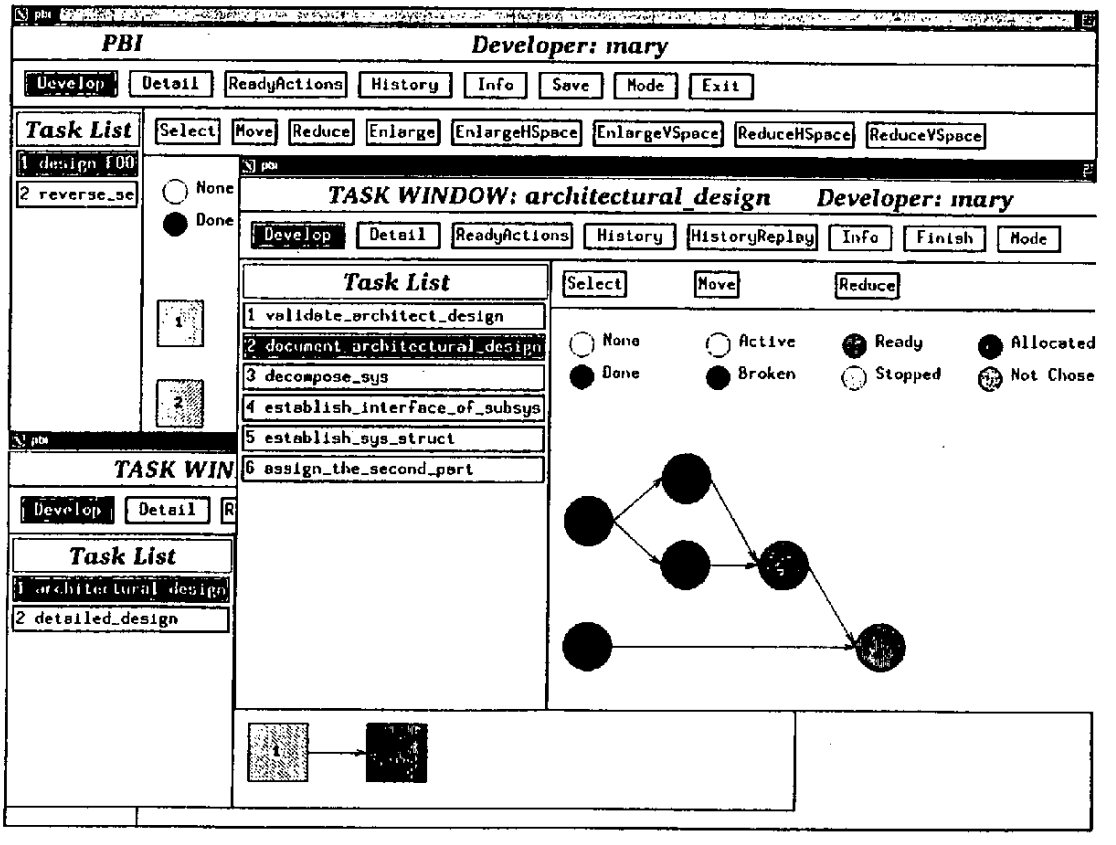


図 5.6-4　プロセスベースの統合開発環境

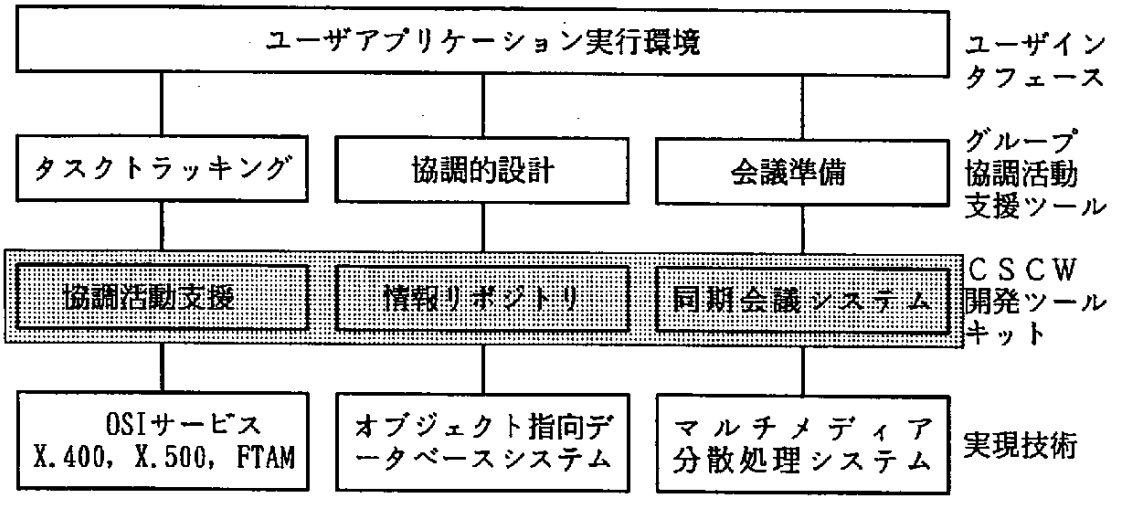


図 5.6-5　EuroCoOpシステムのアーキテクチャ

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。

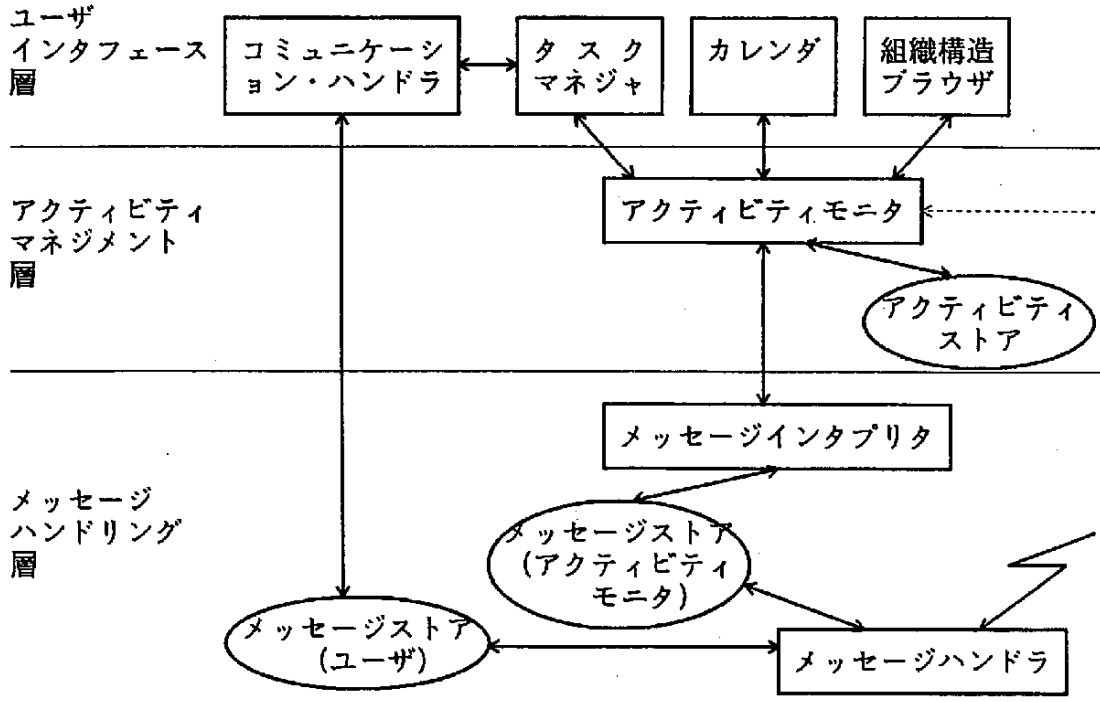


図 5.6-6　EuroCoOpシステムのアクティビティ・コーディネーション・ツールキット

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。

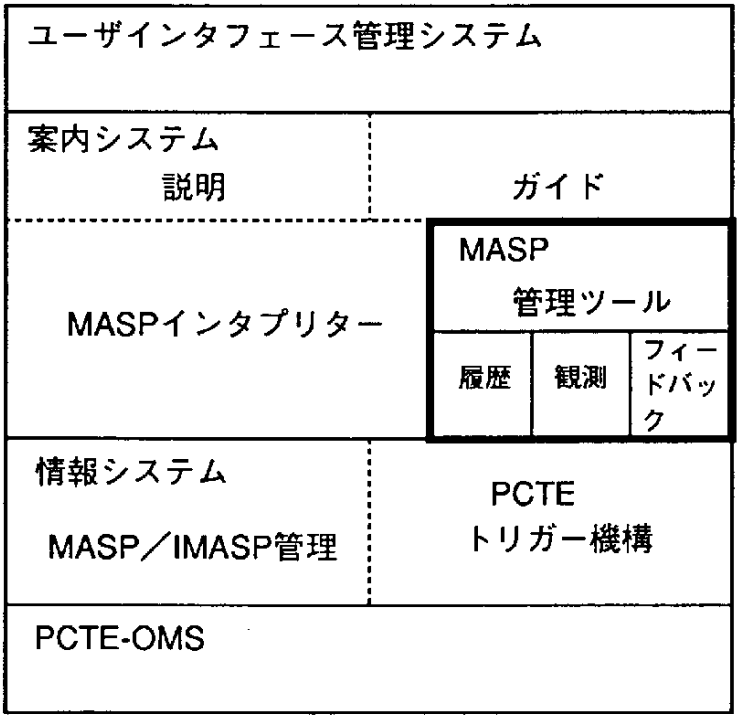


図5.7-1 ALFに基づいたIPSEのアーキテクチャ

(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。

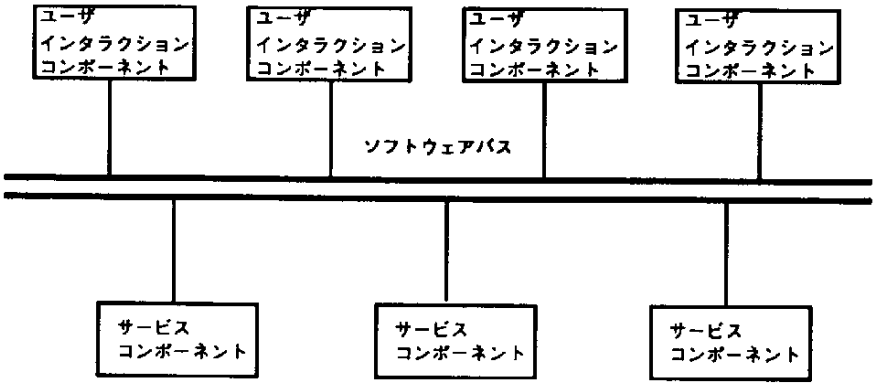


図5.7-2　ESF環境参照アーキテクチャ

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。

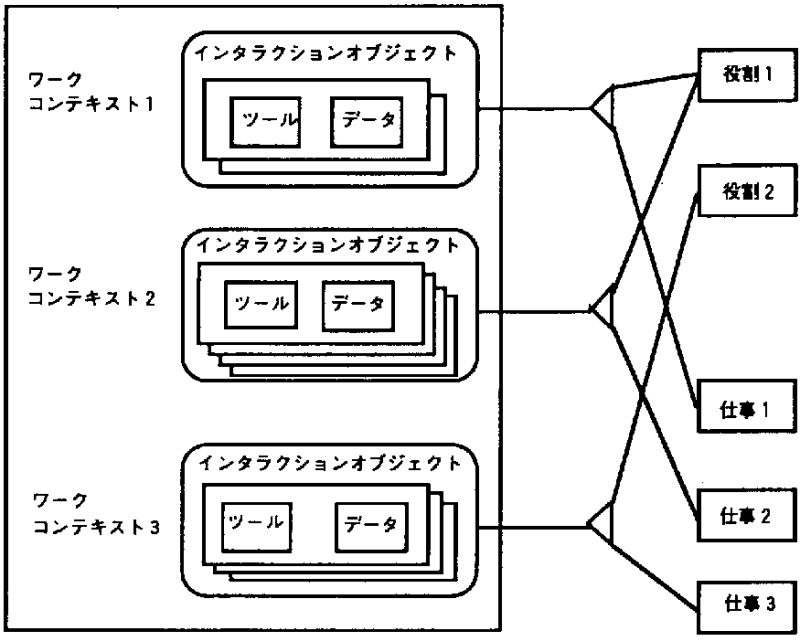


図5.7-3　ユーザインタラクション環境

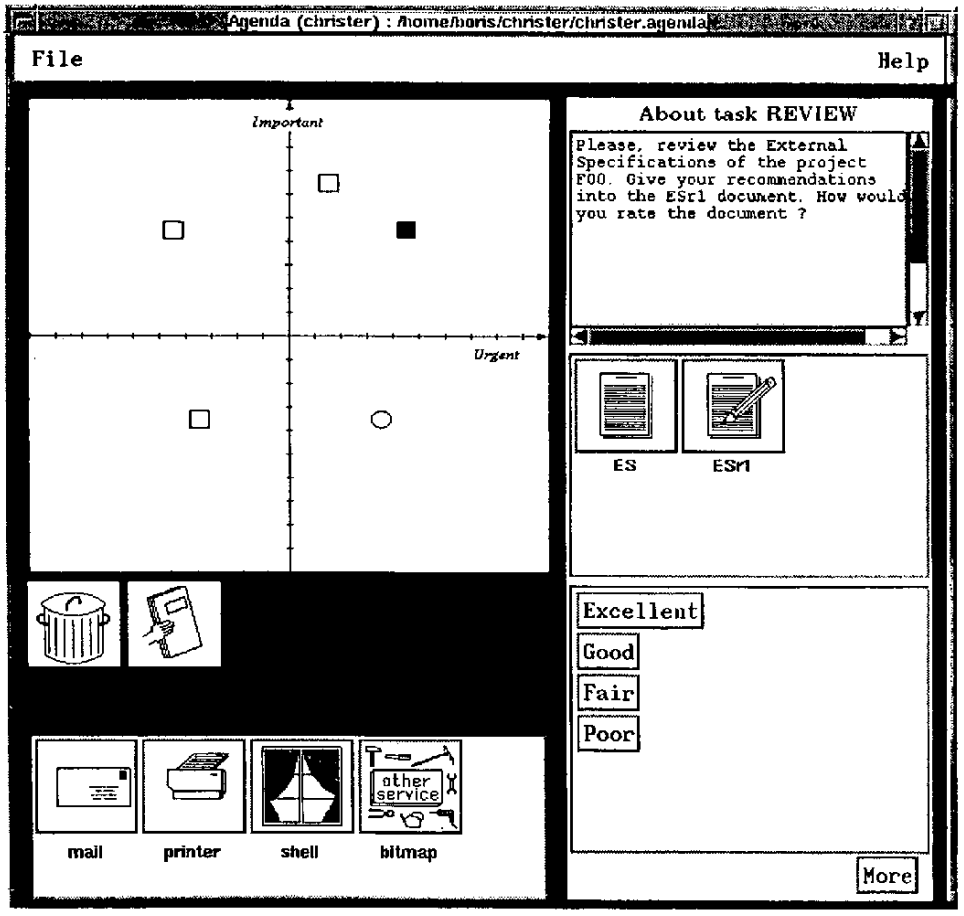


図5.7-4　ユーザインタラクションの実現例（Agendaツール）

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。

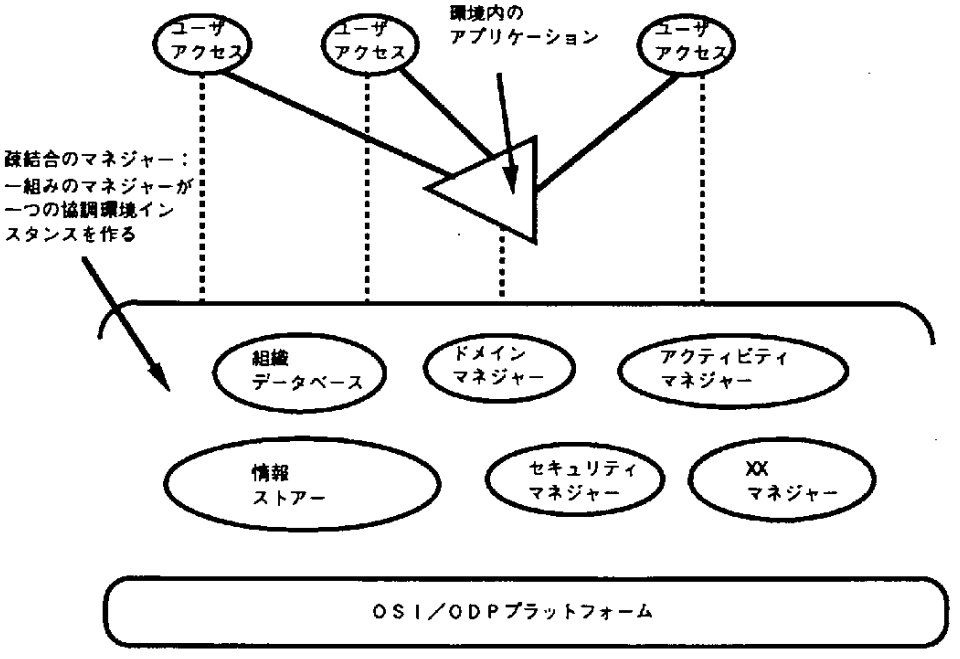


図5.7-5　MOCCAアーキテクチャ

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。

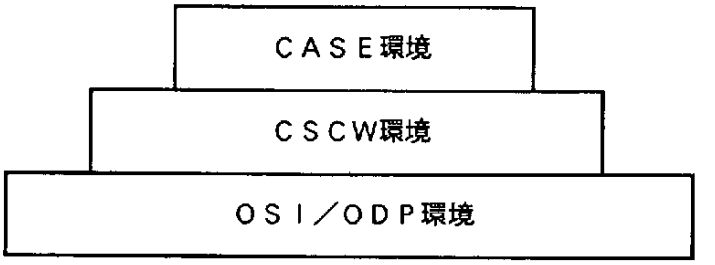


図5.7-6 CSCWとCASE環境の統合

【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．分散開発とグループワーク支援

6．分散開発とグループワーク支援

6.1　分散開発の出現と発展

6.1.1　分散開発とは

『分散開発』とは、通常、図6.1-1 に示すような二つの意味で用いられている。ひとつは、従来の集中ホストマシンに代わる、ワークステーション（WS）とネットワークから成る分散処理環境による開発であり、以下、『分散処理開発』と呼ぶ。一方、開発組織そのものが、地理的に離れた複数の開発拠点に分散する開発形態がある。これを、『分散組織開発』と呼ぶ。両者に共通の場合に、分散開発と呼ぶ。実際には、両方の分散形態が相互に推進力となり、分散処理環境上で分散組織による開発へ移行している [青山92]。

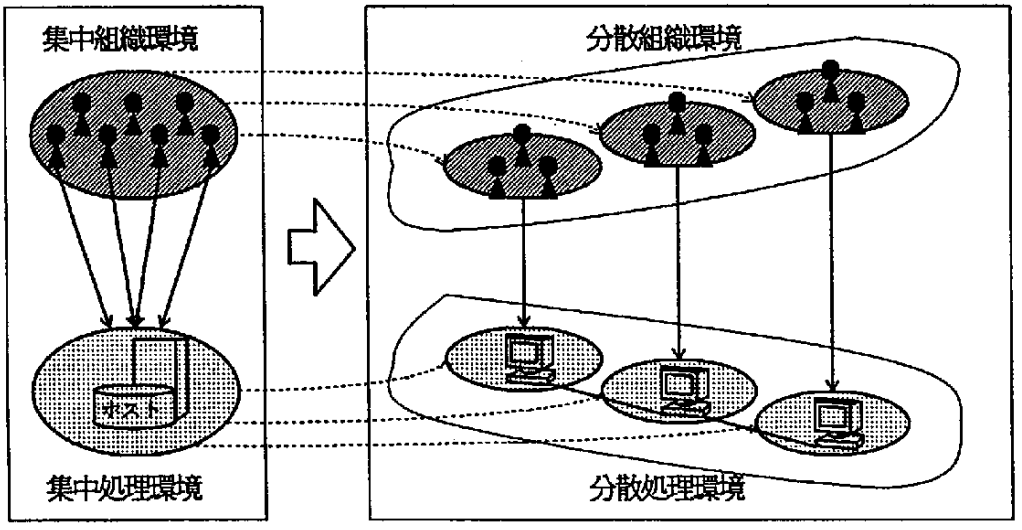


図6.1-1　分散開発環境

6.1.2　分散開発の出現の背景

ソフトウェア開発環境が分散化する背景には、次のような要因が指摘されている。

(1)　技術的背景

①　情報処理技術の変化

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

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

② 開発環境に対する要求の変化

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

(2) 社会的背景

① ソフトウェア開発を取り巻く環境の変化

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

② ワークスタイルの変化

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

6.1.3 分散開発環境の発展

分散開発環境は、技術面で、分散処理の発展に大きく依存してきた。ハードウェア面ではWSとネットワーク技術であり、ソフトウェア面では分散OSなどの基盤ソフトウェアの発展である。図6.1-2は、従来のTSSによる集中開発環境から分散開発環境への移行を示す。

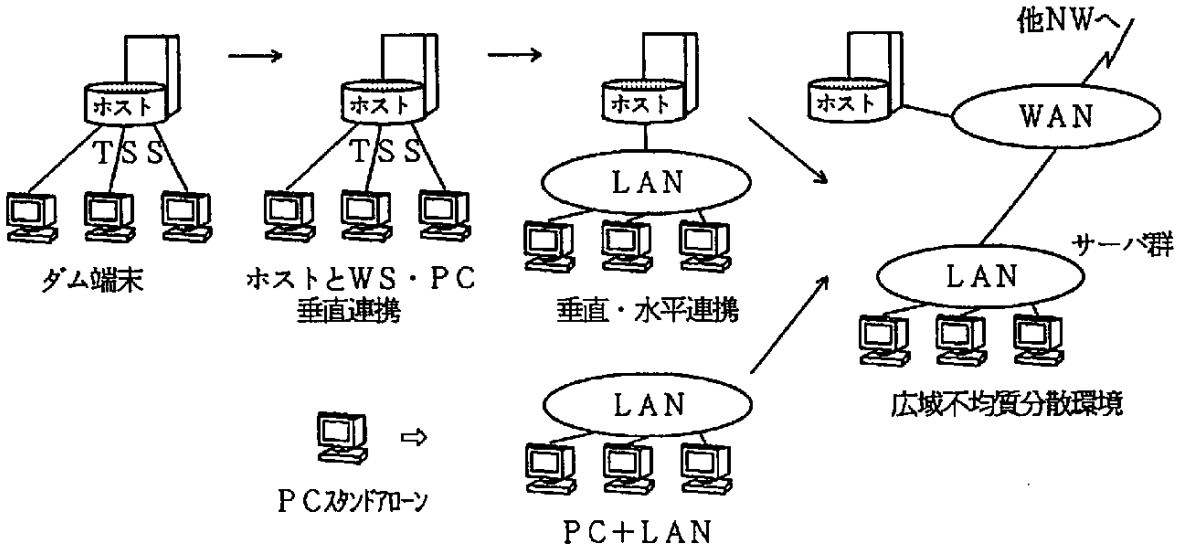


図6.1-2　分散開発環境への推移

(1)　ワークステーションの発展

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

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

(2)　ネットワークの発展

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

(3) 分散開発環境の発展

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

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

6.1.4 分散開発の利点と問題点

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

(1) 分散開発の利点

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

① 人に優しい環境

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

② オープンな環境

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

(a) 互換性 (Compatibility)

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

(b) 可搬性 (Portability)

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

(c) スケーラビリティ (Scalability)

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

(d) 相互運用性 (Interoperability)

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

③ リライアブルな環境

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

(2) 分散開発環境の問題点

分散開発環境では、開発作業の主体である人とWS、さらには成果物であるドキュメントや種々の情報が分散する。このため、開発者やグループ間での協調と情報の収集、交換などの作業オーバーヘッドと作業品質の低下が問題となる。Adamsらはこれを次の五つの問題に要約している [Adams89]。

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

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

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

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

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

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

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

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

なかった新しい課題が提起されている。

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

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

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

6.1.5 分散開発の分類

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

(1) 分散処理開発の分類

① ネットワーク形態による分類

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

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

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

② 機能と負荷による分散

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

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

③ 均質分散環境と不均質分散環境

(a) 均質分散：同一アーキテクチャのシステムで構成される分散開発環境。

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

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

(2) 分散組織開発の分類

組織分散における開発分担の観点から、次の二つに分類される。

① 組織機能分散

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

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

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

② 組織負荷分散

各拠点で、同一工程や同一アプリケーションを分担して開発する。

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

6.1.6 分散開発を支援するソフトウェア工学技術

ソフトウェア工学の観点から分散開発を支援するアプローチが研究、開発されている。特に、ソフトウェア開発活動を構成する主要素は開発プロセスとプロダクトであることから、分散開発の支援は次の二つのモデルで分類できる。

- プロセスモデル
- プロダクトモデル

(1) プロセスモデル

ソフトウェアの開発過程を形式的に表現する、『プロセスプログラミング』が提案されている [Osterweil87]。一方、グループによる共同作業を支援する枠組みとして『CSCW (Computer-Supported Cooperative Work)』[Greif88] や『グループウェア』[Johansen88, 石井89, 松下91] が提案されている。この二つのアプローチは、図6.1-3 に示すように、分散開発を支援する上で、プロセス面から考える共通の視点を提供している。

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

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

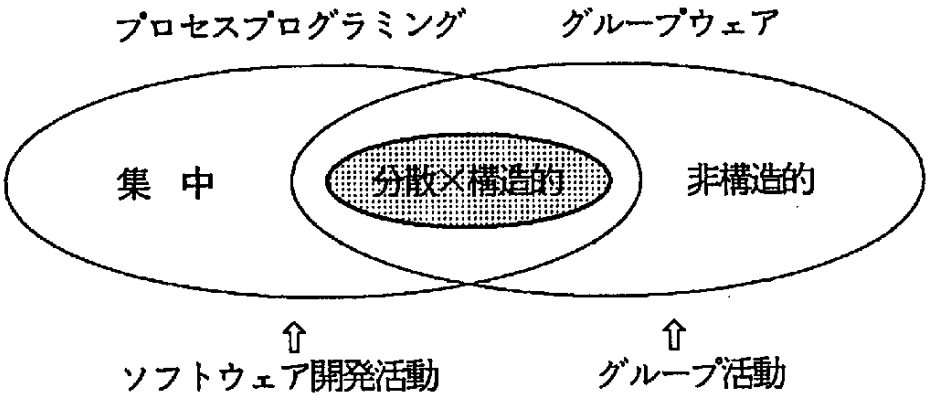


図6.1-3 プロセスプログラミングとグループウェア

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

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

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

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

(2) プロダクトモデル

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

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

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

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

6.1.7 分散開発を推進するために

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

【参考文献】

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[亀田91] 亀田靖浩: 大規模ソフトウェア開発の現状と課題, 電子情報通信学会誌, Vol. 74, No. 5, pp. 457-460 (1991).

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

[前川90] 前川　守（編）：ワークステーション, 丸善, (1990).

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

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

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

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

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

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

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

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

[村井91] 村井　純他: 大規模分散環境ＷＩＤＥのアーキテクチャ, 情報処理学会マルチメディア通信と

分散処理研究会, No. 51-4 (1991).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[垂水92] 垂水浩幸: グループウェアのソフトウェア開発への応用,情報処理, Vol. 33, No. 1, pp. 22-31(1992).

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

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

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

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

6.2 グループの理解とグループウェア

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

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

6.2.1 グループ理解の重要性

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

グループウェアに限らず、何かを支援するソフトウェアは支援対象の深い理解に基づくものでなければならない。会計システムは会計についての理解に基づくものであるべきことは当然であろう。応用分野知識の重要性が認識されつつある。グループウェアでの応用分野知識とはすなわち、グループに関する知識ということになる。また、システム開発の成否はそのシステムを導入する組織に関する知識の量にかかっている。その意味でもグループウェアではグループに関する知識が重要である。

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

6.2.2 グループワーク支援システム研究の中心課題

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

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

6.2.3 グループの特徴

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

(1) グループはソフト・システムである

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

(2) グループはダイナミックなシステムである

① グループの課題は変化する。

② グループ・メンバーの役割はダイナミックに定まる。

③ グループ・メンバーの動機は時間とともに変化する。

④ グループはその外部環境の変化に影響を受ける。

(3) グループ・メンバーは多様である

① グループ・メンバーの動機はメンバー毎に異なる。

② グループ・メンバーの能力や得意分野は異なる。

③ グループ・メンバーの経験は異なる。

(4) グループは成長する

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

(5) グループの運営はインフォーマルである

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

6.2.4 グループ作業での問題

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

(1) グループの課題が定まらない、あるいは不適切である

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

(2) グループ・メンバーの課題・役割が不明確である

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

(3) グループの状態が把握できない

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

(4) グループ・メンバーの意志や動機とグループ課題の整合性が不明になる

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

(5) コミュニケーションがよくない

コミュニケーションすべき事柄が認識されていないことが多い。グループ内コミュニケーションはかなり原始的なものの積み上げが大切である。少しの疑問、確認、提案、コメントなどが自由に流れ

る仕組みが必要である。組織レベルではコミュニケーション・チャネルや内容がフォーマルに決定されており、強制されることが多いがグループ・レベルでは意外と落とし穴になっている。

(6) リーダーが不在である

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

(7) グループ・プロセスが不明である

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

(8) グループ内のコンフリクトの管理ができない

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

6.2.5 グループワーク支援システムに対する期待

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

- グループの課題 (とその細分化したもの) とメンバー作業の関係の表示
- グループ内コミュニケーションの円滑化
- グループ内問題点リストの提示とトラッキング支援
- グループ・プロセスのガイドと記録
- グループ進捗のとりまとめと表示

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

6.3 分散開発のプロセスモデル

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

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

6.3.1 プロセスモデルの概念

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

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

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

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

ソフトウェアの開発のプロセスの典型的な例を示す（図6.3-1）。

(1) 基本検討工程（BI）

業務分析を行い、システムの要求仕様を固め、開発計画を作成する。

(2) 基本設計工程（BD）

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

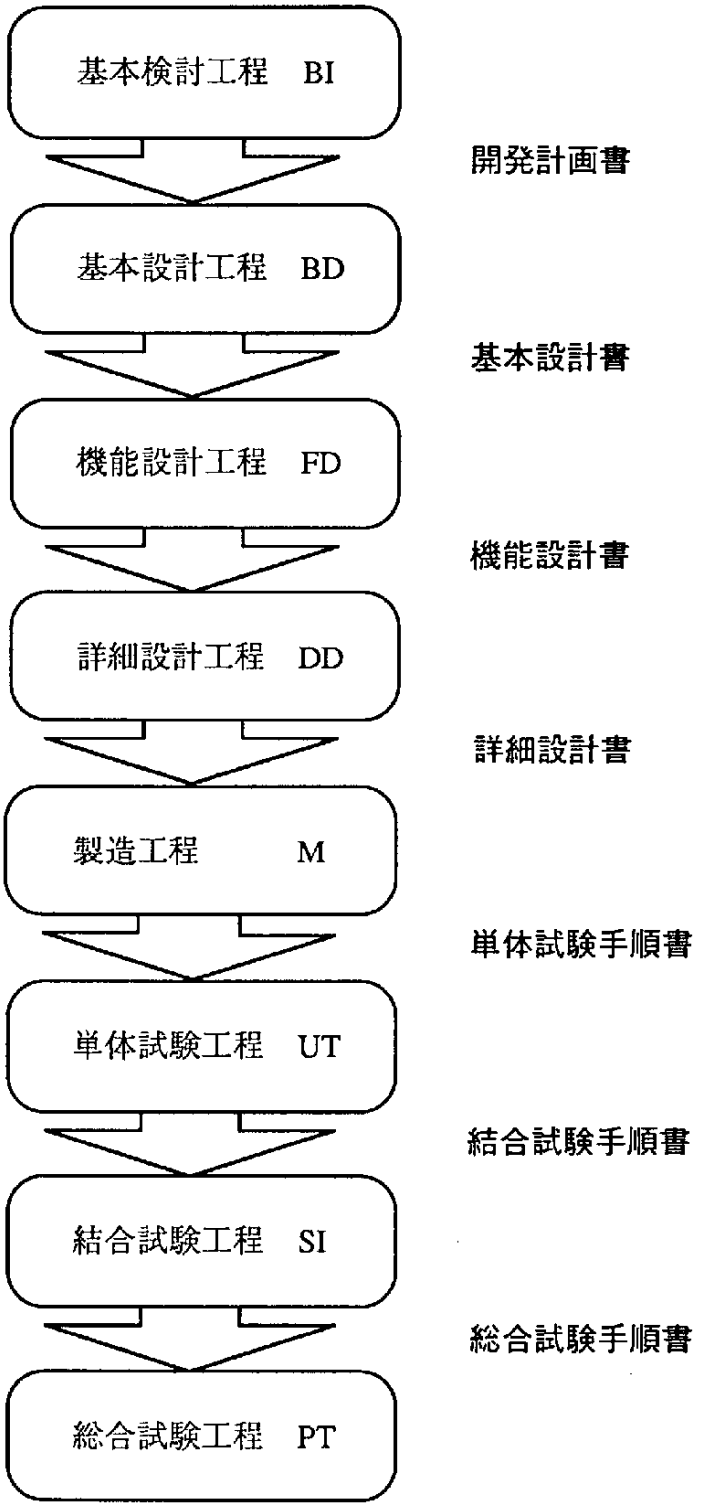


図6.3-1　ソフトウェア開発プロセスの典型例

(3) 機能設計工程（FD）

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

(4) 詳細設計工程（DD）

各機能ブロックを実現するプログラムの仕様を決定し、詳細設計書を作成する。

(5) 製造工程（M）

プログラムの論理設計、コーディング、コンパイル、机上チェックを行う。

(6) 単体試験工程（UT）

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

(7) 結合試験工程（SI）

各ソフトウェアモジュールを結合し、所定の機能を満足することを確認する。

(8) 総合試験工程（PT）

ハードウェア、ソフトウェアを含めた総合的な最終確認を行う。

6.3.2 分散開発とプロセスモデル

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

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

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

などの課題を解決しなければならない。

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

(1) 工程分離・専担化

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

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

① 基本検討工程担当（BI）

開発要望元との窓口となり、具体的な要求仕様を検討、確定する。

② 基本設計担当（BD）

確定した要求仕様から基本設計を行う。

③ 設計（機能・詳細）、製造担当（FD－DD－M－UT）

プログラム設計、製造を担当する。

④ 試験工程担当（SI－PT）

要求仕様どおり、機能するか確認する。

このような各工程担当間で、プロダクトが順次引き渡されて行く。これにより、製品開発のスループット（単位期間あたりの製品供給数）を増大させることが可能となる（図6.3-2）。

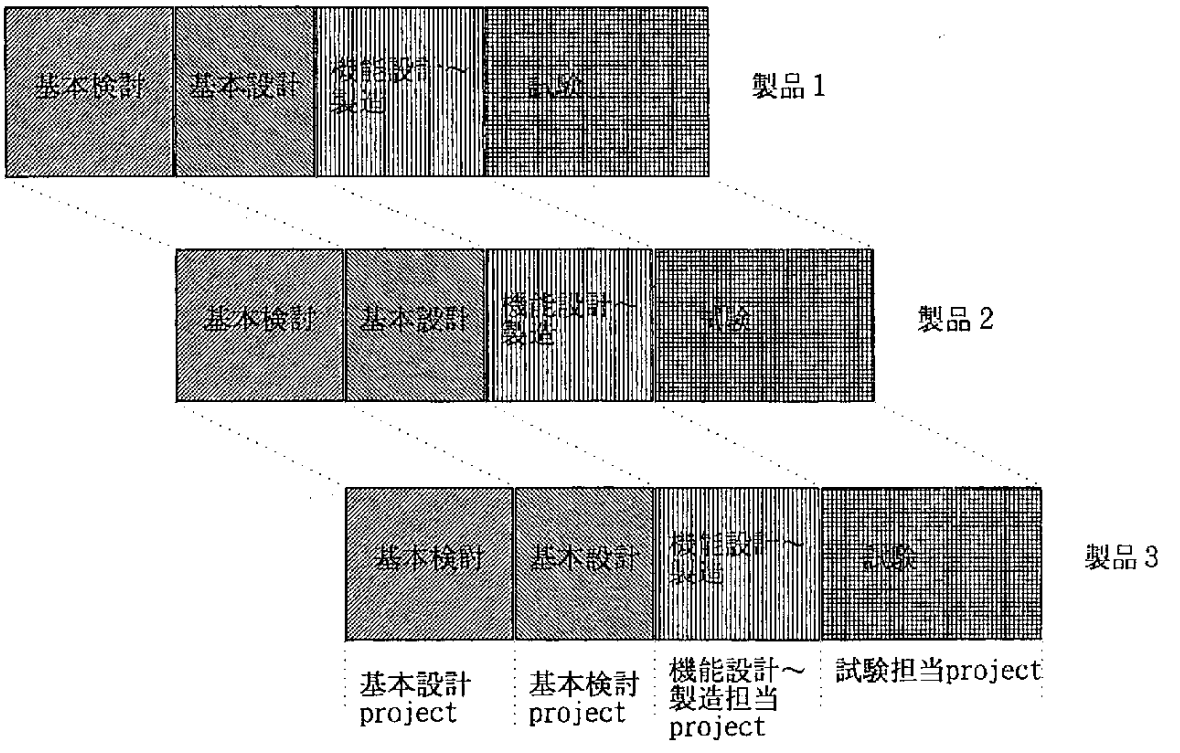


図6.3-2　工程専任による開発プロセス例

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

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

(2) サブシステム単位の平行開発

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

この場合には、サブシステムを担当するチームが地域に分散するため、

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

② 各チームで独自に設計するため、同様なモジュールを重複して開発してしまう

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

などの問題が出てくる。

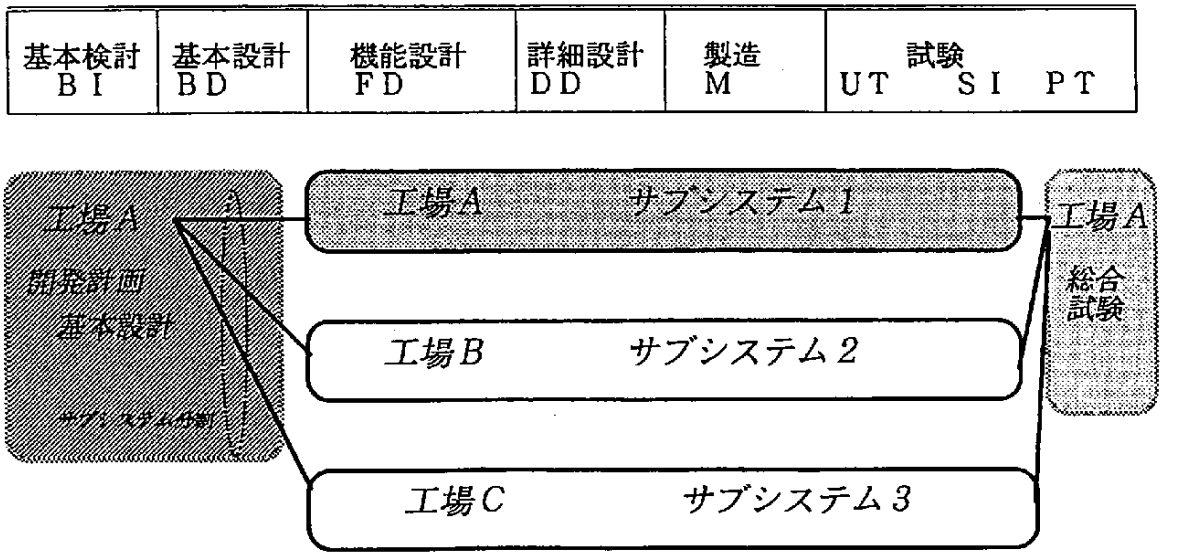


図6.3-3 サブシステム平行開発の例

(3) 専担化とシステム分割の組み合わせ

(1)、(2)を組み合わせることにより、各地域チーム（工場）の負荷を平順化する試みも行われている（図6.3-4）。

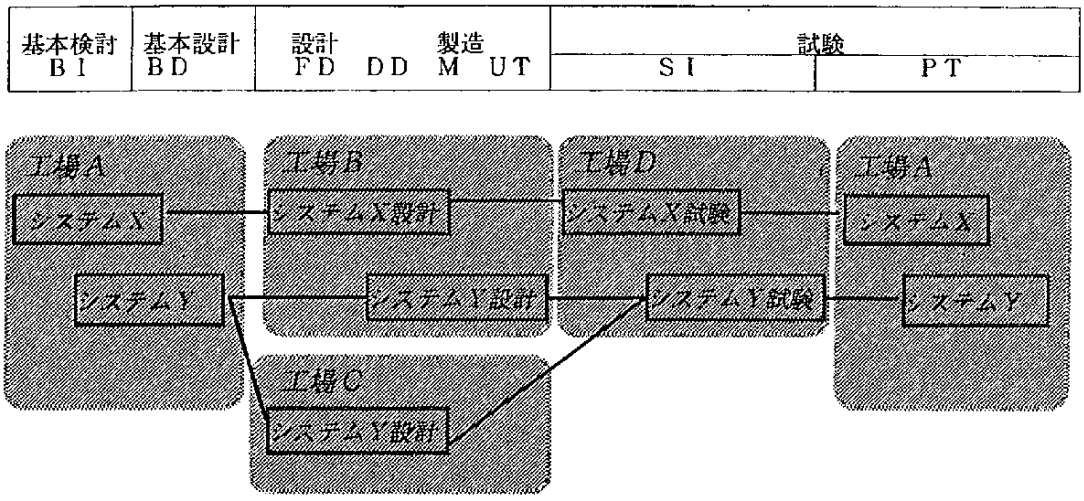


図6.3-4　サブシステム単位の分散及び工程分離の組み合わせ例

6.3.3　プロセスモデルとグループウェア

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

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

① 地域分散したチーム間のコミュニケーションの支援

② プロジェクトの進捗の把握

③ 共通のデータベースの構築

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

(1) 地域分散したチーム間のコミュニケーションの支援

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

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

(2) プロジェクトの進捗の把握

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

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

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

プロジェクト管理については、チームメンバと実際に会って進捗状況を把握できないため、十分に行う必要があろう。「6.5 分散開発とプロジェクト管理」を参照。

(3) 共通作業データのデータベースの構築

地方に分散して作業をすすめる場合、システムとして必要な設計情報、各工程の成果物を、開発メンバがオンラインでアクセスできる共通作業データのデータベースとして用意する必要がある。これは、一箇所でソフトウェア開発する場合と同じ環境をつくることを意味しており、分散開発の“距離”の制約を解決するものである。「6.6 CASE統合環境におけるグループワーク支援」を参照。

(4) 今後の課題

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

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

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

【参考文献】

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

6.4 分散開発のプロダクトモデル

6.4.1 分散開発環境の定義

この節で扱う分散開発環境の定義は、

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

6.4.2 プロダクトモデルとは

(1) プロダクトモデルの定義

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

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

(2) プロダクトモデルの必要性

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

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

6.4.3 プロダクトの分類

(1) 共用プロダクトと個人用プロダクト

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

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

(2) フォーマルデータとインフォーマルデータ

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

6.4.4 プロダクトから見た分散開発の現状と問題点

(1) 現状

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

(2) ライフサイクルモデルに基づいた問題点

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

(3) コンピュータ技術に関する問題点

① 統一的なデータの記述能力

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

② 高速通信技術

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

③ アクセス制御

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

④ ツールインターフェース

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

⑤ セキュリティ

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

⑥ バージョン管理

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

(4) その他

① 文化の壁

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

② ネーミングルール

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

③ インフォーマルな情報の取扱

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

④ デザイン理由・変更理由の取扱

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

⑤ プロジェクトマネージメント

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

6.4.5 分散開発のプロダクトモデリング

(1) モデリングのための指針

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

① オブジェクト指向概念

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

② 情報伝達のための半構造化メール

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

③ ハイパー構造

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

④ メタデータ概念

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

(2) プロダクトモデリング例

上記の指針を基に、開発環境のモデリングを考えてみる。

① プロダクトとプロジェクトの分離

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

離する（クラス概念の導入）。

② メタ化

分離したプロジェクトとプロダクトを図6.4-1に示すようにメタ構造化する。

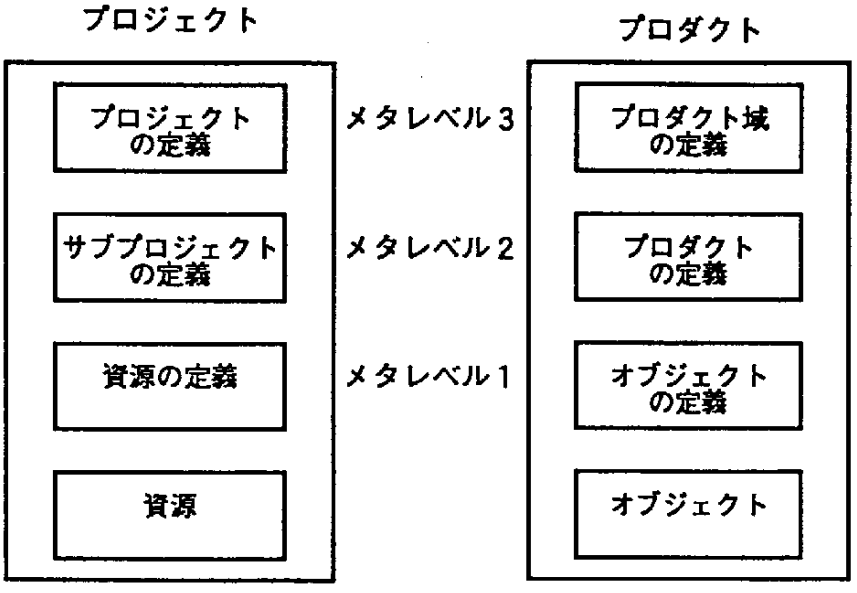


図6.4-1 メタ構造

各メタレベルに含まれるデータは以下に述べるようなものが考えられる。

(a) プロジェクトの定義（メタレベル３）

開発対象プロジェクトを定義する。定義される内容としては、

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

(b) サブプロジェクトの定義（メタレベル２）

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

- 必要な資源（人、金等）

・ 関連する部門

・ サブプロジェクトの責任者

・ 資源間の関係（人員構成、組織構成等）

・ サブプロジェクトのスケジュール 等

(c) 資源の定義（メタレベル１）

当該環境において利用可能な資源を定義する。定義される内容としては、

・ 導入されているソフトウェアやハードウェアの情報

・ 個人的スケジュール

・ プロジェクトメンバの情報 等

(d) 資源

・ ソフトウェア、ハードウェア

・ 人、金 等

(e) プロダクト域の定義（メタレベル３）

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

・ 全プロダクトの責任者

・ プロダクト間の関連

(f) プロダクトの定義（メタレベル２）

各プロダクトをを定義する。定義される内容としては、

・ プロダクトの名称

・ プロダクトを構成するオブジェクトの関係（ビューの定義）

・ プロダクトの責任者（所有者）

・ プロダクトに対するアクセス権 等

(g) オブジェクトの定義（メタレベル１）

プロダクトを構成する個々のオブジェクトを定義する。定義される内容としては、

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

・ オブジェクトの所有者

・ オブジェクトに対するアクセス権

・ オブジェクトのタイプ（テキスト、図形、イメージ等）

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

(h) オブジェクト

・ テキスト、図形、イメージ 等

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

するためのツールが単純化され、異なる環境での扱いが楽になる。

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

③　プロジェクトとプロダクトの関係付け

最後にプロジェクトとプロダクトを関係付けるために、お互いの間にリンクを張る(ハイパー機構)。今回のモデルでは、図6.4-2に示すように、同一レベル間で各々必要とされる物同士にリンクを張り、プロジェクトとプロダクトを関連付ける。リンクの表現は、上記②で定義した各メタレベルにおける属性として新たに付け加える。

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

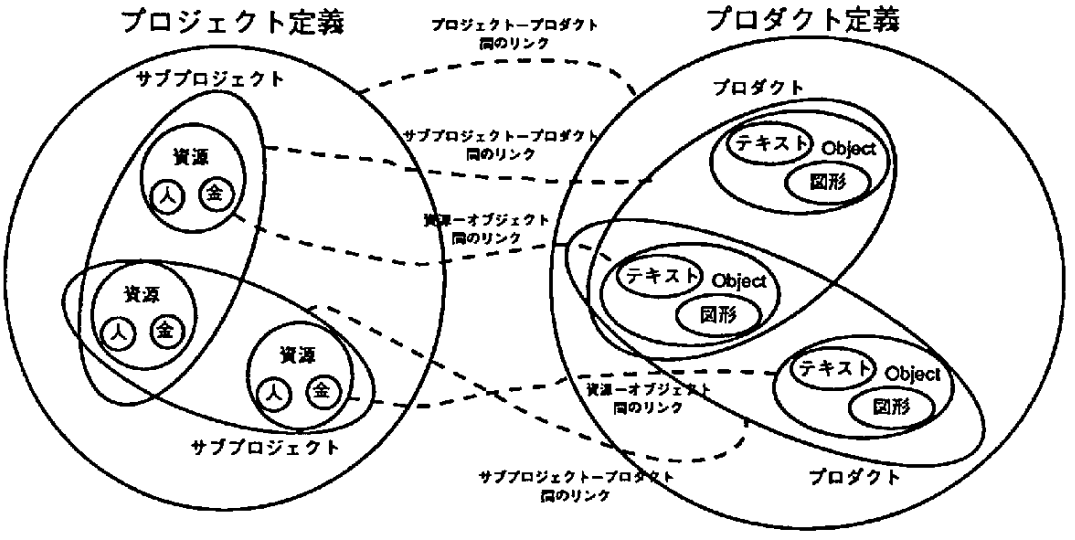


図6.4-2　プロジェクトとプロダクトの関連

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

④ メッセージの流れ

最後に、上記モデルを使ったコミュニケーションを考えてみる。図6.4-3に見られるように、コミュニケーションのために必要なメッセージは自分のメタ定義を参照しながら、半構造化メッセージの形で、関連する上位／下位のメタならびにリンクを張られたデータに送られる。メッセージを受け取った側はメッセージの内容を解析して、必要なアクションを起動させると同時に、必要ならば自分のメタ定義を通して、関連する上位／下位のメタにメッセージを送ることができる。これによって何らかの変更がデータに加えられた場合、自動的に関連する全てのデータにメッセージを送る機構（ノティファイ機構）を実現することができる。

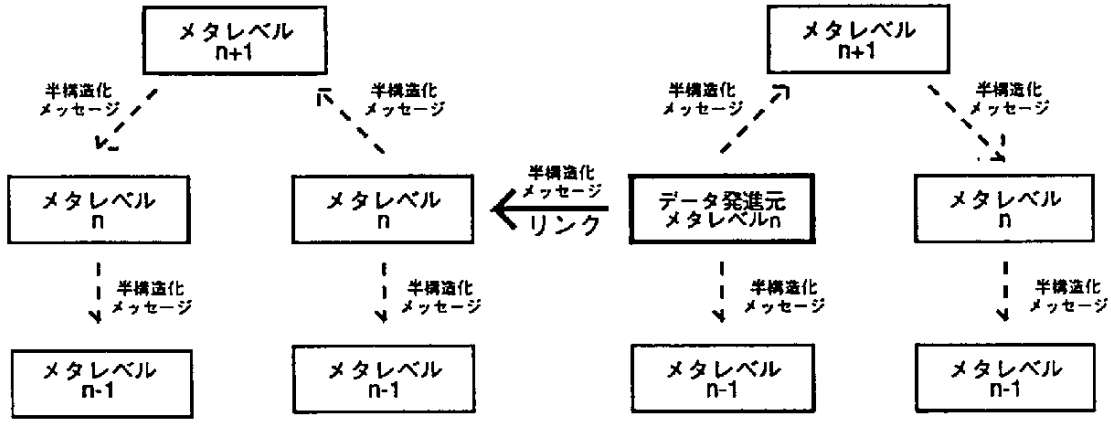


図6.4-3 モデル上のメッセージの流れ

(3) 今後の課題

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

6.4.6 Hypertextを使ったシステムの実例

(1) 概要

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

(2) Hypertextによるドキュメント管理

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

(3) 管理対象ドキュメント

DIFで採用したドキュメンテーション方法は、南カリフォルニア大学で開発されたSystem Factory Documentation Methodを利用している。これはシステム開発プロセスをアクティビティに分解し、それぞれのアクティビティに対してドキュメント作成が行われる。DIFにおいては以下の八つのドキュメントが対象とされている。

① 要求仕様書(Requirements specification)
② 機能仕様書(Functional specification)
③ 構造仕様書(Architectural specification)
④ 詳細設計仕様書(Detailed-design specification)
⑤ ソースコード(Sourcel-code document)
⑥ テスト仕様および品質保証書(Testing and quality-assurance document)
⑦ ユーザマニュアル(User manual)
⑧ 保守マニュアル(System-maintenance guide)

(4) DIFの構造

① 操作モード

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

と情報構造レベルとよばれる二段階のレベルでDIFの操作が可能である。

② フォームとテンプレート

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

③ アクセス管理

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

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

④ 情報レベル

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

⑤ 情報構造レベル

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

(a) リンク

リンクはテンプレート間の接続のために用いられる。

（例）C言語ソースコード ～ 実行モジュール

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

(b) キーワード

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

(c) フォームとコンポジション

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

⑥ ドキュメントの一貫性制御

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

⑦ ファイル構造

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

(5) まとめ

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

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

【参考文献】

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

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

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

[原田91] 原田 実 : CASEの全て, オーム社 (1991).

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

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

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

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

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

6.5 分散開発とプロジェクト管理

6.5.1 分散開発におけるプロジェクト管理の特長

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

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

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

(1) プロジェクト管理標準手順

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

(2) プロジェクト管理項目

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

(3) プロジェクト体制

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

(4) プロジェクト管理の機械化

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

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

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

6.5.2 プロジェクト管理標準手順

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

プロジェクト管理手順は図6.5-1に示すように計画段階と実施段階があり、それぞれ次に示す作業を行う。

(1) プロジェクト計画段階

① 開発作業の洗い出し……基本構想書に基づいて最終成果物を決定し、それに対応した開発作業を決める。成果物と開発作業を階層構造に詳細化して体系化した技法としてWBS(Work Breakdown Structure)がある。

② 開発手順の標準化……詳細化した作業単位をベースにして開発手順を決める。分散開発では並行開発作業も考慮する場合もある。

③ プロジェクト組織……の計画　工程別、機能別のプロジェクト体制を設定する。プロジェクト計画に必要な職能(リーダ、分析者、設計者、プログラマ等)別に開発作業を対応させプロジェクト組織を計画する。

④ 開発環境の設定……開発方法論、開発技法に加え分散開発環境を設定する。地域分散開発を実現するワークステーション型CASEを搭載したワークベンチの活用等を検討する。

⑤ プロジェクト資源計画…開発工数、開発環境等の資源見積と配分計画を立てる。

⑥ プロジェクトスケ……ジュールの設定　大日程計画書を作成する。中日程計画書から下の詳細スケジュールはそれぞれの工程レベルのイニシェーション作業で設定していく。

⑦ プロジェクト管理……基準の設定　管理項目を決め、管理情報の収集タイミングを決める。また管理基準を設定しプロジェクトの状況判断に用いる。管理基準として進捗や品質の基準の他、障害管理基準、レビュー基準、連絡票伝達等の運用基準がある。また分散開発の立場から開発者に対する委託基準を決める。最終納入物件を規定するだけでなく、開発作業状況報告の内容、チェック方法も規定する。

(2) プロジェクト実施段階

① 管理情報の収集・分析…プロジェクト管理手順に従い、開発作業のマイルストン毎の管理情報を収集し、分析する。

② 原因究明……分析された情報をもとに問題点の原因を究明し対策方針を立てる。

③ 対策……対策作業方法を決め、作業の優先度、プロジェクト資源からの制約等を考慮し作業指示をする。

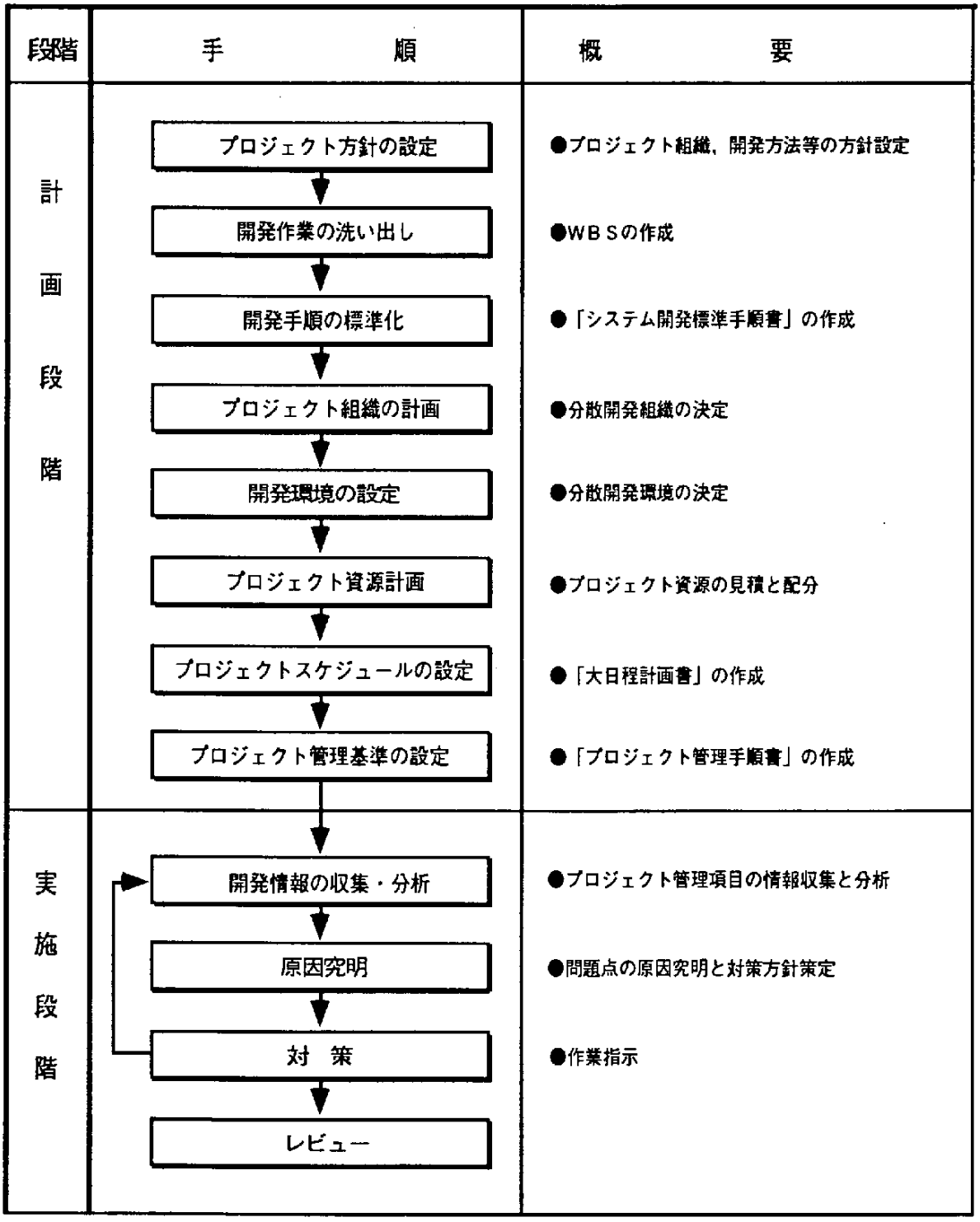


図6.5-1　管理手順の概要

④ レビュー……レビュー計画により着実なレビューを実施し、適切かつタイムリな評価と制御を図る。

6.5.3 プロジェクト管理の内容

プロジェクト実施段階におけるプロジェクト管理項目として次の項目がある。

(1) 進捗管理

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

(2) 品質管理

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

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

(3) 開発規模管理

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

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

(4) 発注管理(一括発注管理)

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

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

(5) 案件管理

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

各案件には担当者を定め、スケジュール表に載せ管理する。

(6) 仕様変更管理

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

要がある。

(7) 連絡票管理

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

(8) 障害管理

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

(9) 開発環境管理

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

(10) 性能管理

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

(11) バージョン管理

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

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

(12) 要員管理

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

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

(13)　ドキュメント管理

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

6.5.4　分散開発プロジェクト体制

分散開発プロジェクトを発足させる時、次の点に留意して体制を作る必要がある。

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

これらの留意点のほかに、開発するシステム技術的特徴と課題、開発規模、開発方針及び分担開発グループの開発能力等によりプロジェクト体制を考える。図6.5-2にプロジェクト体制例を示す。

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

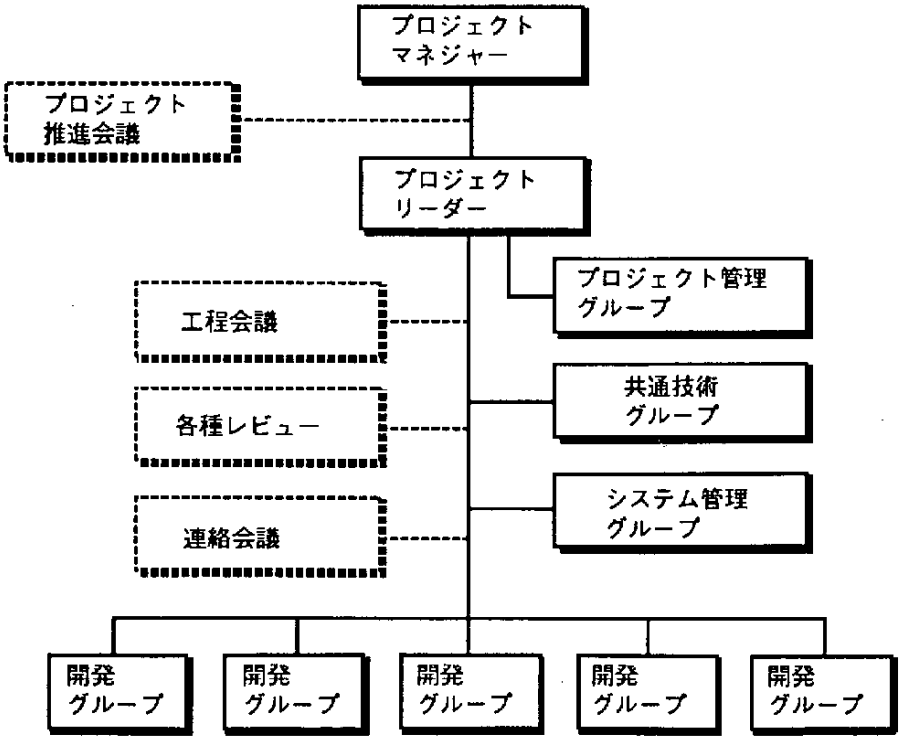


図6.5-2　開発体制の例

各グループの作業を次に示す。

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

6.5.5 プロジェクト管理の機械化

システム開発の分散化に対して管理は集中化しなければならない。ネットワークシステムを利用したプロジェクト管理の機械化を図る必要がある。図6.5-3は分散開発ネットワークシステムの例であるが、分散開発拠点のファイルに蓄積されたプロジェクト管理情報はネットワークシステムを伝送され中央側開発センタにあるリポジトリに収集される。分散拠点側では各サブシステムの開発の他、電子メールによる連絡票や障害票の送付、電子会議による設計レビュー等を行い、必要に応じて開発センタ、他分散開発拠点の設計情報を検索する。

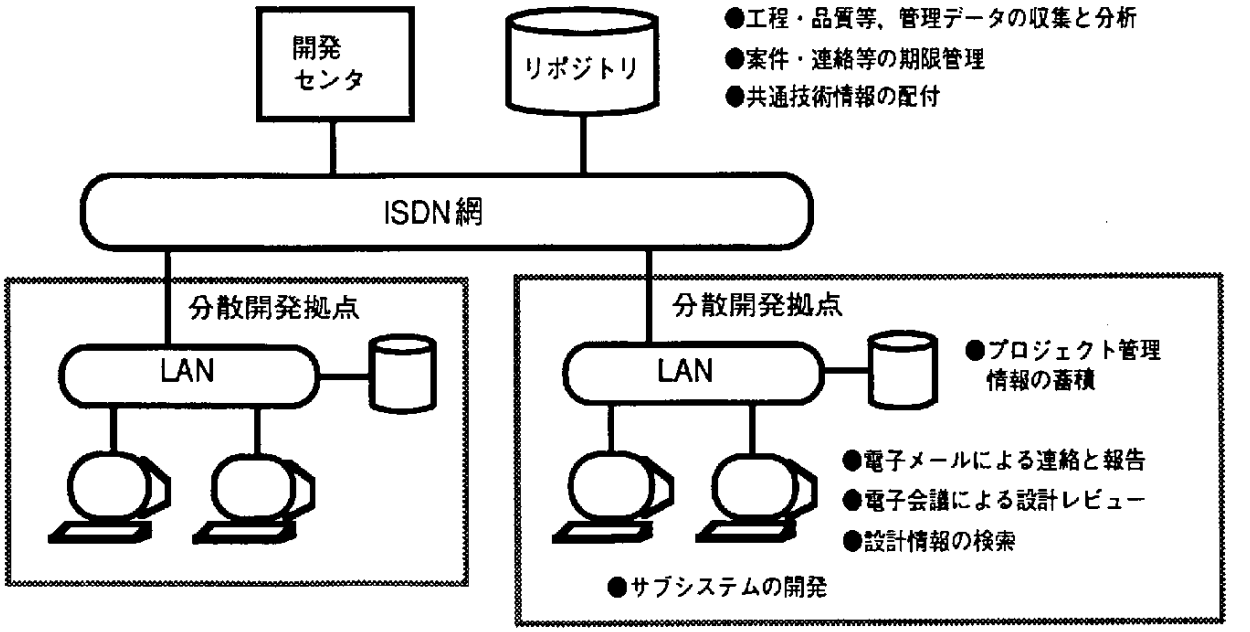


図6.5-3 分散開発ネットワークシステム

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

SEWB／PJMS は工程管理表を階層管理しており、個人から開発グループ、プロジェクト全体までレベルに応じた進捗管理が行える。下位のレベルの管理情報は上位のレベルの管理情報に自動集計する。分散開発拠点での管理情報は自動集計され図6.5-4に示すような大日程計画表に状況が反映される。プロジェクトリーダは大日程計画表の進捗カーブから工程遅延の状況を把握し対策を立てる。

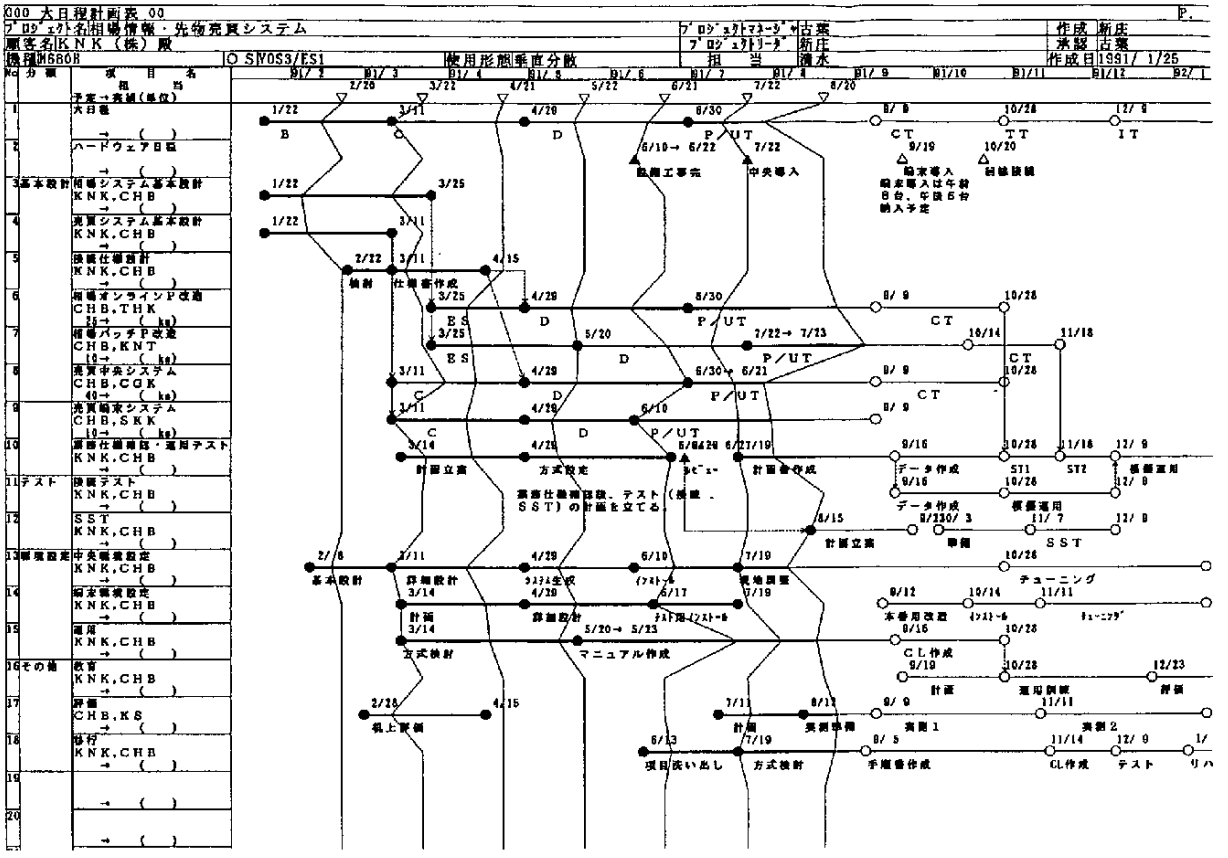


図6.5-4　SEWB／PJMS 大日程計画表の例

【参考文献】

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

6.6 CASE統合環境におけるグループワーク支援

6.6.1 CASE統合環境とグループワークの接点

(1) CASE統合環境の概略・現状

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

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

CASE統合環境がもつべき主要な機構を粒度を無視して列挙してみると、

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

・　成果物の動きを構成管理としてまとめる機構

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

① ユーザインタフェースの統合（日本語処理も含む）

② ツール制御の統合

③ ツールデータの統合

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

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

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

・ IBM AD／Cycle (Application Development／Cycle)

・ CIS ATIS (A Tool Integration Standard)

・ ISO IRDS (Information Resource Dictionary System)

・ ECMA PCTE (Portable Common Tool Environment)

などが挙げられる。

(2) CASE統合環境におけるグループワークの形態

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

まず、「ソフトウェア開発におけるグループワーク」とはどのようなアクティビティを指すのか、それを明確にしておく必要がある。が、これまでのところこれに対して明確な定義が与えられているとは言い難く、漠然としたイメージがあるにすぎない。そこで、作業仮説としてソフトウェア開発において起こる様々なアクティビティのうち、グループで作業するが故に必要となるアクティビティを列挙し、それらを整理してここでの「ソフトウェア開発におけるグループワーク」と定義することにする。それは表6.6-1のようになる。

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

表6.6-1に見るように、グループワークをCASE統合環境がサポートするためには、現在のリポジトリの持つ機能に加えて各種イベントをトリガとして稼動するメール機能、さらには共通画面制御、ポインタ制御等のリアルタイム制御機能が要求される。これを概念的に示すと図6.6-1のようになろう。

表6.6-1　ソフトウェア開発におけるグループワーク

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

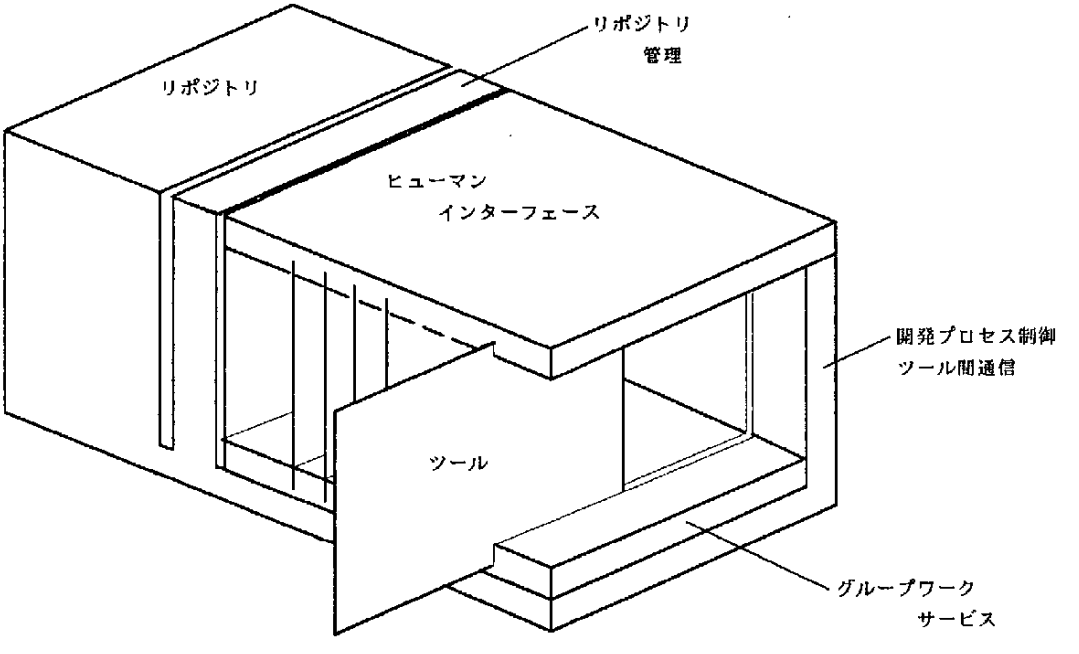


図6.6-1　グループワークを取り込んだCASE統合環境

6.6.2 CASE統合環境とグループワークのサポート状況

上記のように、現在CASE統合環境として仕様が公表されたり、インプルメントが行われたりしているシステムがいくつか存在する。この節では、それらのシステムついて、表6.6-1の三つの区分、すなわち「通知」「対話」「参照」について、具体的にどのレベルでどの程度サポートされているかを概観してみる。CASE統合環境それ自体はCASEツールを統合するためのフレームワークであるので、グループワークをサポートする個々のツールに言及している仕様はほとんどなく、したがってここでもソフトウェア開発におけるグループワークサポートのための「仕掛け」を主として見ていくことにする。

(1) AD／Cycle

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

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

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

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

AD／Cycle アーキテクチャの主要構成要素は、ツールの統合に必要な三つの要素（上記）をいずれも含んでおり、共通ユーザインタフェース、ワークステーションサービス、ワーク管理、インフォメーションモデル、ツール、リポジトリサービス、ライブラリサービスなどが挙げられる。統合されているツールのユーザインタフェースは、いずれも CUA (COMMON USER ACCESS) の規則・ガイドラインに沿った設計になっており、ツール間で互いに整合がとれたものになっている。またワークステーション上の表示にはプレゼンテーションマネジャが均一な表示フォーマットを提供している。ワーク管理は、ユーザの開発プロセスに合わせて作業をガイドする機能で、ユーザの方法論にあわせてツールの起動・終了が自動的に制御される。インフォメーションモデルでは、ユーザのソフトウェア開発に関連しているすべてのデータ、その属性およびデータ間の関連を定義することができ、それらの情報をツールが使うことになる。リポジトリサービスは、それらのデータを一元的管理するものであり、複数のユーザ・ツールに対して、データ定義、データ管理、データ共有、アクセス制御などを提供している。AD／Cycle アーキテクチャでは、統合ツールも重要な構成要素であり、企業のデータモデル構築用のツール、分析ツールなど上流工程支援のツールや、サードベンダ提供のツールも含めた下流工程支援のためのツールが整備されている。

このように広範なサービスを提供している AD／Cycle を、グループワークサポートという面からながめてみる。ソフトウェア開発におけるグループワークのうち、「通知」と「参照」の二つのサポートが実現されていることがわかる。まず、AD／Cycle の構成要素であるワーク管理がそれである。ソフトウェアの品質がコスト、スケジュール、生産性に根元的にかかわっており、品質を向上させるキーは開発プロセスの中にあるとの立場から、インフォメーションモデルを定義して生成物を明確にするとともに、その作業を実施するプロセスとそこで使用する手段を明らかにしていく。そのために個々のステップとその中で行う作業を定義し、つぎにそれらを総合した全体プロセスを定義して、ワークフローストラクチャを構成する。こうして定義されたプロセスは非常に複雑であり、中間生成物、最終生成物、チームのメンバ数などなどあらゆる要素がはいっている。これをプロセスモデルから読みとり、それを解釈し、人間系に表示し、開発作業のナビゲーションをするのがワーク管理サービスである。ただ、開発組織によってそのようなコンピュータによるナビゲーションを強く求める場合と、弱いナビゲーションの方が望ましい場合とがある。AD／Cycle では、そのあたりに柔軟性をもたせた運用ができるようになっている。

グループワークのうち「参照」もまた AD／Cycle においてサポートされている。開発作業社がツールを介してリポジトリ管理下にある任意のデータを参照する機能である。AD／Cycle でこの仕掛けを提供しているのがリポジトリマネジャである。リポジトリマネジャの機能は「仕様」と「ランタイムサービス」の二つに大きく分かれる。仕様では、データに対して三つのビューを定義している。ツール間共通の概念・意味を定義するビュー、採用しているデータベース依存部分を吸収するビュー、ツ

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

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

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

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

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

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

を提案している。またCASE 統合環境のために、ツール制御のためのインタフェースを定義しているが、特徴的なのはそれを実現するためにオブジェクト指向モデルを採用している点であり、TYPE 階層（「通常の」オブジェクト指向のクラス階層）の形態で提供されている。ATIS は機能面から見ると、「モデル」という名の六つの独立した機能の階層構造を採っており、そのおのおのにインタフェースが提供されている。

- オブジェクトモデル

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

・ バージョン管理モデル

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

・ コンフィギュレーションモデル

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

・ ワークフロー制御モデル

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

・ ツール登録モデル

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

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

・ IRDS モデル

ANS X3.138 の IRDS 用インタフェース

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

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

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

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

1970年台に米国防総省(DoD)が作成したSTONEMAN文書は、既にプログラミング用支援環境の必要性を認識しており、基本的なツールのインタフェースを規定している。この文書から大きな影響を受けてスタートしたのがECのPCTEプロジェクトである。PCTEは、ハードウェアに依存せず、ツールベンダなどが採用できる標準のツールインタフェースを規定し、ツールのポータビリティを確保することをねらいとした標準インタフェースであり、PCTE全体としてツール統合環境の枠組みを提供している。標準ツールインタフェース設定に当たっての基本的考え方は、次の五つである。

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

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

- データ管理

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

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

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

  ツールベンダがツールを開発するためのインタフェースを提供している。

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

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

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

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

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

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

【参考文献】

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

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

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

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

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

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

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

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

6.7 グループワーク支援システムの適用分野

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

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

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

6.7.1 グループワーク支援の背景

(1) 分散開発と地方分散

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

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

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

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

・　上流工程の要員として首都圏に逆Ｕターンし長期滞在を余儀なくされる。

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

(2)　開発作業における情報量の増加

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

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

まさに、開発担当者はグループのメンバとして様々なコミュニケーションを交わしながら、品質を維持するためにそれぞれに関わる情報の渦の中に身をおいて作業をしているわけであり、いかに「情報を咀嚼する」かが生産性を決定づける大きな要因となっている。グループウェアはその道具として、部品の仕様検索や担当者同士のコミュニケーション等の手助けをする機能を提供することができる。

(3)　技術革新の土壌

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

(a)　ワークステーション(パソコン)の普及

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

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

(b)　ネットワーク技術の発達

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

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

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

(c) マルチメディアへのニーズ

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

- イメージを処理するCPU処理性能の飛躍的な向上。
- CD-ROM、光ディスクなどマルチメディア対応の大容量データ蓄積装置の開発。
- マウス、タブレット、イメージ・スキャナー、ビデオカメラ、高品質プリンタなどのイメージ入出力装置の技術進歩。
- 画像圧縮技術の研究と標準化の発展。
- ISDN(サービス統合デジタル網)によるマルチメディア通信。

(d) オープンシステムの進展

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

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

6.7.2 情報の広場としてのグループウェア

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

- 情　報　　：　文字、グラフ、絵、写真、ビデオ、音声、音楽　等
- 機　器　　：　キーボード、タブレット、イメージスキャナ、ビデオカメラ、マイク、高品質ディスプレイ、ＣＤ－ＲＯＭ、光ディスク　等
- システム　：　電子メール、ボイスメール、電子掲示板、電子会議室（共用スクリーン、共用ウインドウ）、遠隔会議（電話会議、テレビ会議、ワークステーション会議）、非同期会議支援、会議準備支援、協同文書執筆、メッセージ構造化、共用ファイル、ハイパーテキスト、オブジェクト指向データベース　等
- ネットワーク：　LAN、WAN、FDDI、ISDN　等

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

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

(1) 同室会議支援：　GDSS、Colab、Project Nick
(2) 遠隔会議支援：　Cognoter、CRUISER、MERMAID、Rapport
(3) 電子メール、電子掲示板：　CC:Mail、GroupTalk
(4) 協同文書執筆支援（グループエディタ）：　Quilt、PREP
(5) 非同期討論支援（グループメモリシステム）：　gIBIS
(6) メッセージ構造化支援（情報フィルタリング）：　InformationLens、ObjectLens

6.7.3 システム開発の現場におけるグループウェアの利用

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

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

(1) 情景1：「進捗会議の日に」

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

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

【効　果】

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

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

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

(2) 情景2：「原因不明の障害を追う」

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

ンタフェースに疑いが出てきた。さっそく、サーバを開発した丙社の担当者Cに連絡を取り、同様の資料をFAXで送信し・・・。

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

【効　果】

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

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

(3) 情景3：「変更依頼書を管理する」

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

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

【効 果】

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

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

(4) 情景4：「開発基準を学習する」

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

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

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

・・・

【効 果】

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

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

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

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

6.7.4 今後の課題

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

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

(1) 電子メールにみる冗長情報

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

(2) セキュリティ機能

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

(3) プロジェクト管理ツール

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

(4) 上流工程の支援ツール

グループウェアは、分散システム開発の製造工程に多くの適用分野を見ることができる。しかし、6.7.1 において述べたように地方分散を本当の意味で実現させるためには、分析・設計などの上流工程を分散環境において支援するグループウェアのしくみを発展・充実させることが重要である。

(5) ソフトウェア開発のグループ特性

分散システム開発の体制においては、グループウェアが適用できる分野は幅広く考えられる。しかし、それに従事する人々のグループは、同一企業内とは限らず、むしろ発注する側とされる側、管理する側とされる側というケースが多く、請負い形態の立場が明確である。一般的に、発注側がグループウェアのツール環境を用意し受け手側がそれに応じた体制を整えることになろう。ここで大事な事は、グループウェアが「グループを管理する道具」ではないことをお互いに認識することである。グループウェアの目的は、グループ間の協調作業を支援することであり、発注側はこの点によく留意し

て運用すべきであり、受け手側も協力的に参加することが必要である。その時にグループウェアの機能を最大に引き出すことができる。当初は導入の効果というものが見えにくい部分もあろうが、充分な説明と理解なくしては、グループウェアも掛け声だけに終わってしまうことになろう。

(6) グループウェアがもたらす作業形態の変革

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

【参考文献】

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

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

[末松91] 末松千尋（著）：オープンシステム入門, ダイヤモンド社 (1991).

6.8 ネットワーク環境における分散開発の将来

6.8.1 コンピュータネットワーク技術の最近の進展

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

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

最近のネットワーク技術で特に注目に値する技術には、次のようなものがある。

(1) ネットワークの高速化

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

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

(2) ネットワークの柔軟化

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

(3) ネットワークの多様化

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

(4) 統合的なネットワーク環境の実現可能性

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

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

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

(5) 分散環境基本ソフトウェアの進展

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

(6) 標準化の進展

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

6.8.2 アナログネットワーク技術の進展

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

(1) CATV系の発達

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

(2) 放送衛星の進展

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

(3) TV電話

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

(4) TV会議系

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

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

6.8.3 新しいコミュニケーションの形態

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

(1) ディジタル系とアナログ系の混合したコミュニケーション

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

(2) 高速な広域ネットワークの実現

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

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

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

(3) マルチメディア環境とネットワーク

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

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

(4) 人工現実感とグループウェア

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

7．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

7．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

7.1　D²による分散開発支援

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

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

(1)　社会情勢による開発拠点の分散化

(2)　ワークステーション等の分散開発支援技術の進歩

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

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

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

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

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

7.1.1 ソフトウェア分散開発モデル

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

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

すなわち、(1)はじめにでも述べたように、まず拠点の分散が要求となるし、どの様に分散させるかが問題となる。(2)開発の開始から終了までの時間に沿って工程と作業を拠点毎に分散させるかが問題となる。(3)各工程で開発する成果物は何か、それはどこに保されるのか、その情報の共有化はどこまで許すのか最終的にどうまとめるのかなどが問題となる。(4)分散化された開発者達や組織、関係者がいつどこでどの様に情報を交換し、どの様な指示・報告がなされるべきかが問題となる。図7.1-1に制御システムの典型的な分散形態のモデルを示す。

つぎに、開発工程と情報の分散モデルについて図7.1-2にその例を示す。縦軸に工程をとり、横軸に分散された拠点を取る。交差した箇所にその工程での成果物とその共有範囲を記してある。なお、この図では、クライアントとの関連と保守工程については省略してある。また共有情報としてはこれらの成果物のほかに打ち合わせ記録やスケジュール、関連資料などが必要である。

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

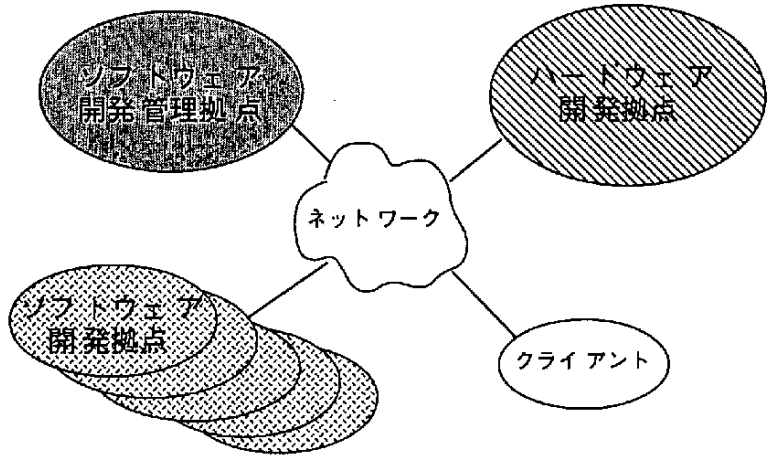


図7.1-1 開発拠点の分散モデル例

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

図7.1-2 開発工程の分散モデル及びその成果物の共有モデル

(1) 開発拠点とその役割分担の設定

(2) 開発作業項目と手順の設定

(3) 各作業段階でのインタフェースの設定

(4) 拠点間でのフローの同期点の設定

(5) 情報の保管場所の設定

(6) データ／制御フローの記述とチェック

(7) 機密性のチェックとフローの修正

(8) 支援ツールの決定と利用方式の決定

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

7.1.2 ソフトウェア分散開発支援システムのモデル

前項では、分散開発のモデルについて述べた。そこで示したようにプロジェクト毎に分散開発のモデルは異なってくる。もちろん、メタ方法論としての分散開発のマニュアル的なものを考えることもできる。したがって、支援システムとしてはこれらの変化の影響が少ないモデルに基づいていることが望ましい。また、分散開発の実験の結果からグループウェアの支援系の満たす必要のある要件として、①支援サービスの連続性、②情報の利用容易性、を挙げている [貫井90]。これらを満たすモデルとして、図7.1-3に示すような階層型支援機能モデルを考えている。

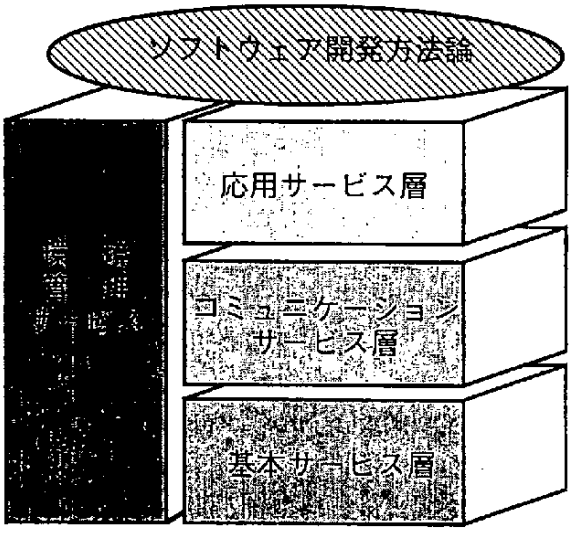


図7.1-3 階層型支援機能モデル

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

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

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

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

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

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

7.1.3 エージェント指向電子メール

ソフトウェア分散開発支援システムの第二層の例として、電子メールを中心としたコミュニケーションシステムについて述べる。コミュニケーションシステムには同期型（リアルタイム型）と非同期型とがあるが、ここではこれらを融合したシステムとしてとらえている。電子メールにおけるエージェントとは、業務に電子メールを用いる場合に発生する様々な手作業を支援する機能として考えられている。例えば、Andrew Message System や Object Lens などに典型的なものが見られる。そこでのエージェントはどちらかといえばオブジェクト的なものでメールにより起動されるサービス機能である[松尾91]。分散協調的な問題解決を実現しようとする試みも数多くあるが、進んでいるのはスケジューリングや負荷分散についてのものが多い。D^2 システムでは、ソフトウェア分散開発に必要な支援として汎用の単機能なエージェントとそれを組み合わせるエージェント記述スクリプトを用意している。汎用の単機能エージェントとしては、図7.1-4に示すように

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

・　配送（ルーティング）：自動的にメッセージを関係者に回覧、同報、発信し配送する機能。

・　起動（エグゼキューティング）：メッセージの受信・発信等のイベントにより指定したアプリケーションを実行する機能。

が用意されている。

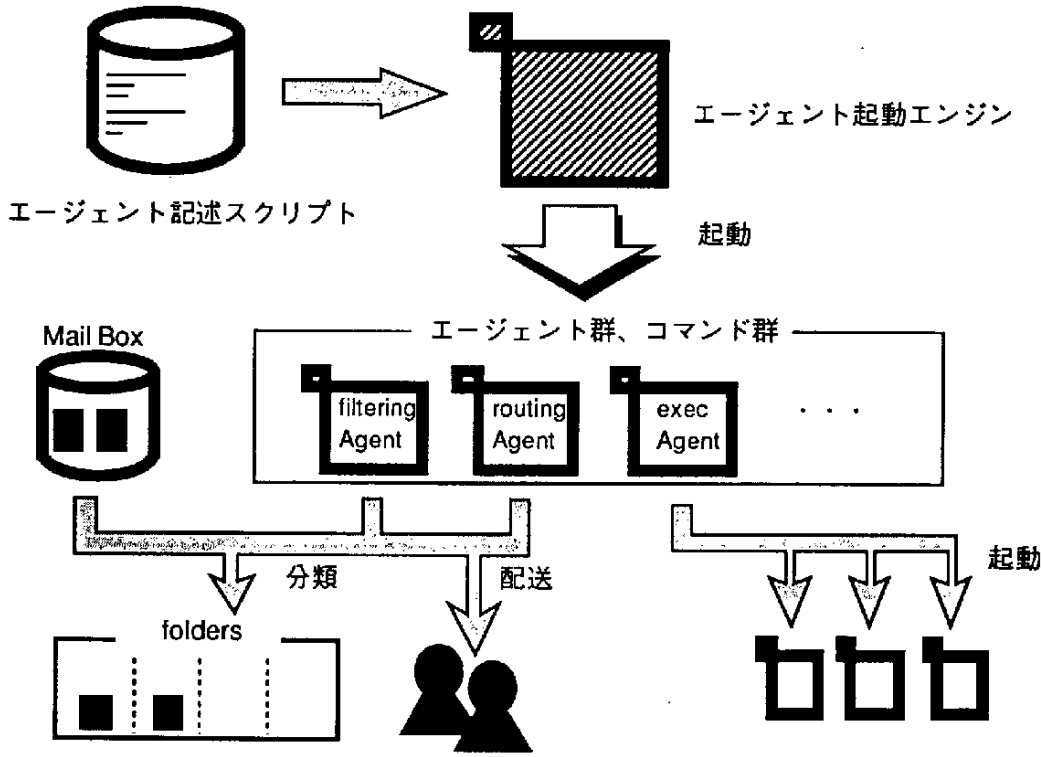


図7.1-4　エージェント指向電子メールシステム

これらのエージェントとその使用方法を記述したエージェントスクリプトを用いることにより、例えば図7.1-5に示すようなソフトウェアの成果物収集支援システムを容易に構築できる。図7.1-5において、プログラムは分散した各拠点にいる各担当により作成され、その後リーダのところに集められmakeすることになっている。このとき、ファイル転送などの方法を用いると転送先のディレクトリ構成を把握し意識して転送を行わねばならない。また、リーダが進捗を把握するのもそれなりの手間がかかる。

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

さらに、プロジェクト毎にスケジューリングエージェントを含んだ開発支援エージェントライブラ

リを関係者に送付することにより、開発に必要な支援機能と管理体制を確立することができる。この場合、様々な設計変更やトラブルに対応した開発プロセスやスケジュールの変更が、各メンバのエージェント環境に一様に行われることが必要である。

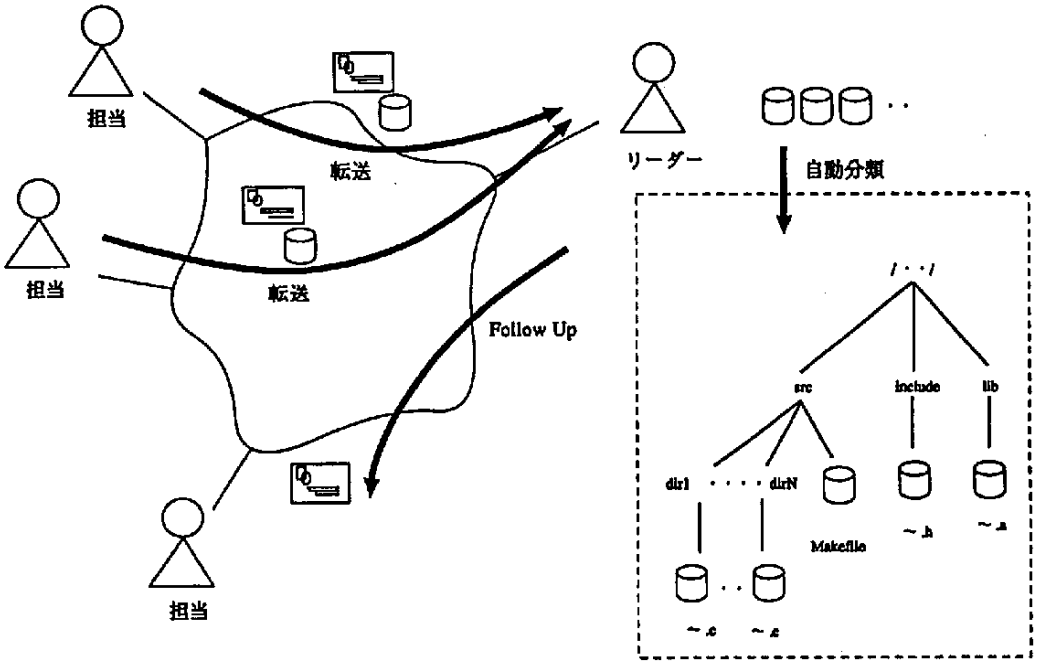


図7.1-5 エージェントによるソフトウェア成果物収集支援システム

図7.1-6 同期型コミュニケーションシステムの表示画面例

7.1.4 リアルタイム電子会議システム

図7.1-6は同期型コミュニケーションを行うためのリアルタイム会議システムを組み合わせた場合の表示画面の例である [太田92]。参加者全員に共通に見えるホワイトボードと特定のメンバのみに共通するネゴシエーションボードとからなる。ソフトウェア分散開発の面においては、コードレビューなど直接対象となるソフトウェアの実体をチェックしながら会議を進めるような場合に特に有効であると考えられる。レビューの結果はメールで送られ分類エージェントにより格納保管される。なお、会議システムの各ボード自体もエージェントである。今のところ、エージェントと呼ぶにふさわしい分散協調メカニズムはインプリメントしていないが、実用的なものとしては会議予約システムなどを考えている。

7.1.5 ユーザ認証システム

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

(1) 認証サーバによる認証

(2) 相互認証による相手の確認

(3) 暗号化によるプライバシの保護

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

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

ムはつぎの２点に特徴がある。

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

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

図7.1-7に本システムの認証モデルを示す。

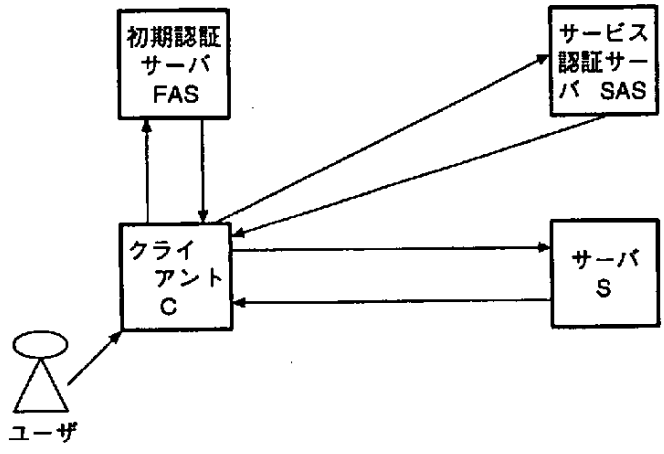


図7.1-7 D^2 Authentication systemの認証モデル

初期認証手順を図7.1-8に示すが、それは以下の手順で行われる。

(1) ユーザがユーザ名をＣに入力する。

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

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

(4) Ｃはユーザにパスワードなどを要求しユーザ情報と照合する。

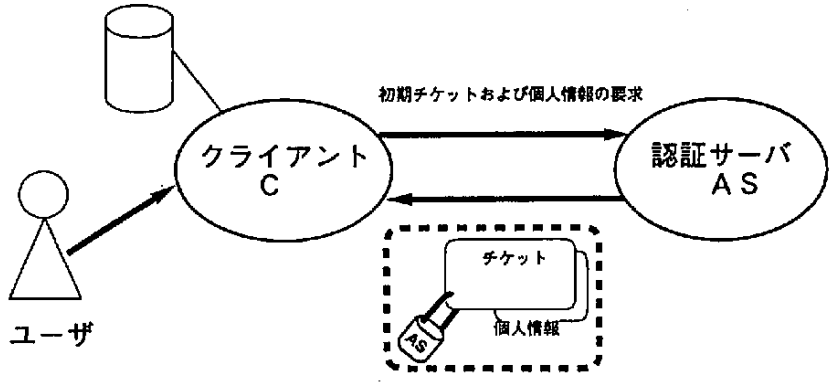


図7.1-8 初期認証手順

一方、サービス時の認証は以下の手順で行われる（図7.1-9参照）。

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

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

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

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

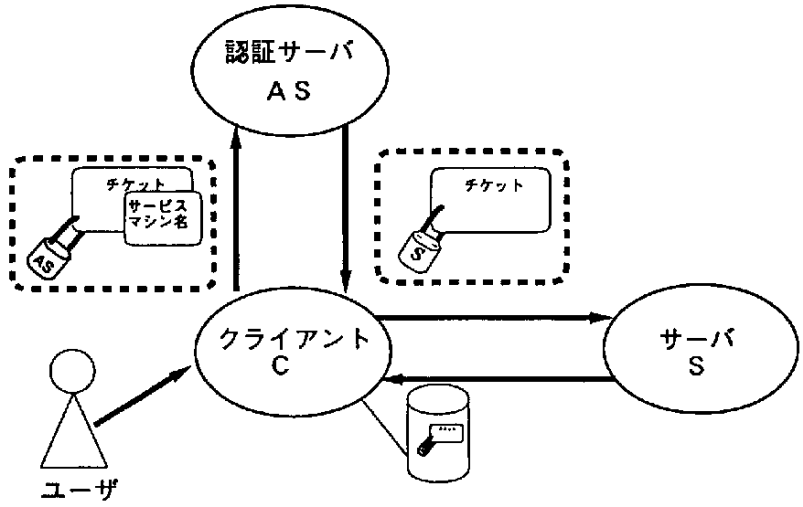


図7.1-9 サービス時の認証手順

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

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

(1) 分散開発の形態のモデルとその考え方

(2) 分散開発支援システムの階層的支援機能モデルとその考え方

(3) エージェント指向電子メールとその応用例

(4) リアルタイム電子会議システム

(5) 分散開発の基本的セキュリティ管理としてのユーザ認証システム

に関する概要を述べた。

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

【参考文献】

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

[貫井90] 貫井春美,栗原美佐,三原幸博：グループウェア支援機能の実験的考察,ソフトウェア工学研究会71-8,pp.57-64(1990).

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

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

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

7.2　MERMAIDによる分散開発支援

7.2.1　はじめに

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

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

7.2.2　MERMAIDシステムの概要

(1)　概要

MERMAIDシステムは、WSを使って、遠隔地にいる人達がテキスト（文字）、図形、イメージ、手書き、音声、動画を含むマルチメディア情報を広域網を介してリアルタイムで交換しながら、効率的に会議をはじめとした協同作業を行うことを支援する在席会議システムである [Watabe90,Watabe91,渡

部91]。このシステムはグループ通信アーキテクチャ (GCA) [阪田89] を基礎として新規に構築したものであり、現在まで約１年半にわたり分散環境での実用実験を行っている。さらに、本システムを基盤として、情報処理システムによりグループを支援できる可能性の大きい部分である文書協同作成、ソフトウェア協同作成、グループ意思決定を含むグループ協同作業を支援することを目標としている。

(2) 利用者用装置の構成

各利用者のWSは図7.2-1のような構成となる。WS は NEC EWS4800（モデル220または260）を利用している。ディスプレイ（解像度:1280×1024）にはマルチメディア文書や動画が表示される。マウスは利用者がメニューを選択したり、画面に表示された文書を指す（ポインティング）ために利用する。タブレットは手書き文字・図形を画面の文書中に入力する場合に利用する。イメージスキャナは文書を読み込んで他の参加者に送るために利用する。ビデオカメラは参加者の様子を撮るために使われる。ワイヤレスマイクとワイヤレスヘッドホンは音声のやりとりのために利用する。他に動画を通信する際の動画情報の圧縮・伸張のための動画コーデックが必要である。なお、キーボードはテキスト（文字）の入力、編集のため以外にはほとんど使う必要がない。

図7.2-1 MERMAID利用の様子

(3) ソフトウェア構成

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

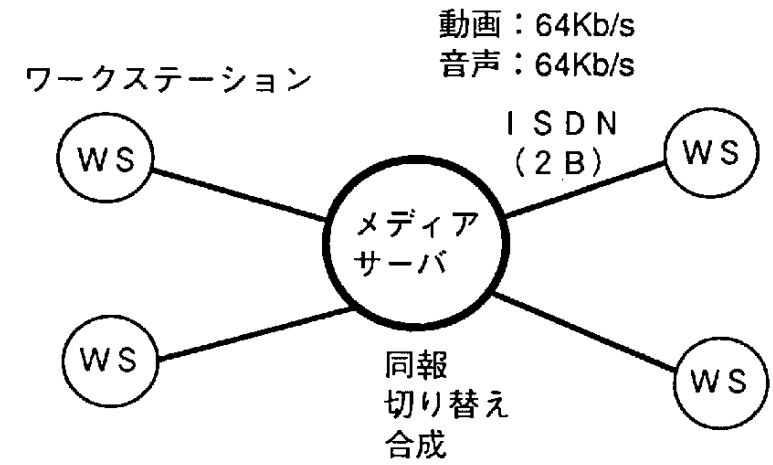
用者がこのシステムに親近感を持ち、操作も容易になった。

(4) MERMAID システムの通信環境

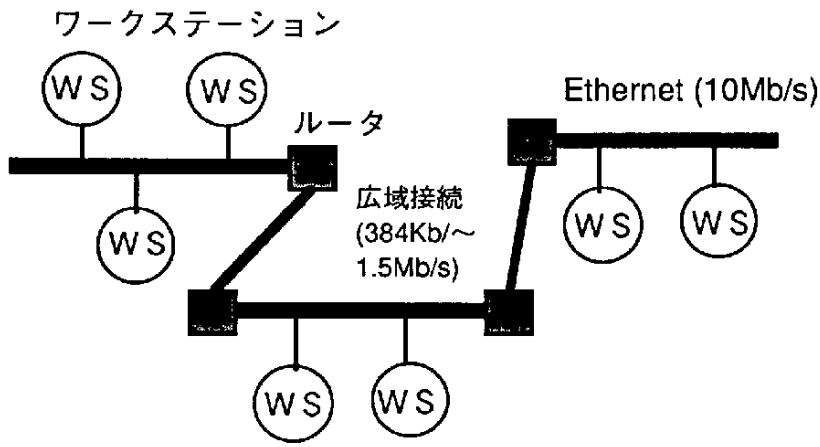
MERMAID システムでは動画、音声、データ（制御情報および文書情報）の交換のために現在は ISDN (2B+D) とブリッジで広域に接続された Ethernet を使用している。

(a) 動画、音声の通信

図7.2-2(A) に MERMAID システムにおける動画、音声の通信環境を示す。動画と音声の通信には ISDN のBチャネル (64Kb/s) を利用している。２本の B チャネルのうち、１本を動画通信、他の１本を音声通信に利用している。動画はビデオカメラから入力され、ビデオコーデックにより64Kb/s に圧縮され、通信路に送られる。音声はやはり64Kb/s に符号化され、通信路に入る。会議に参加しているクライアント（利用者の WS）からの動画はメディアサーバに一度集められて選択、合成され、全クラ



（A）動画・音声通信



（B）データ通信

図7.2-2 MERMAIDの通信環境

イアントに同報される。全員の音声はメディアサーバでミックスされ、全クライアントに同報される。

(b) データの通信

図7.2-2(B) に MERMAID システムにおけるデータの通信環境を示す。制御情報や文書情報の通信にはブリッジで広域接続された Ethernet (Branch4680) を利用している。速度はローカル（LAN 内）で 10Mb/s、リモート（LAN 間）で384Kb/s～1.5Mb/s である。通信プロトコルは低位は TCP/IP を採用している。

(5) MERMAID システムの特徴

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

(a) 広域で多者間の在席会議を支援

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

(b) 動画と音声を活用した臨場感の創出

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

(c) マルチメディア文書のリアルタイム編集、交換

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

(d) 柔軟な共有画面操作権の移行制御

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

(e) 親しみ易いヒューマンインタフェース

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

7.2.3　実験構成と利用評価

次のように利用実験を行い、評価した。

・　利用実験期間

1990年4月より1991年10月

・　接続地点

日本電気（株）の本社（東京）、中央研究所（川崎）、関西研究所（大阪）、筑波研究所（筑波）、神奈川サイエンスパーク（川崎）、北米研究所（アメリカ・ニュージャージー州プリンストン）の6箇所（図7.2-3）、計約２０台のWSを設置、会議の９０％は２地点間の接続、１０％は３地点接続、４地点以上はほとんどない。

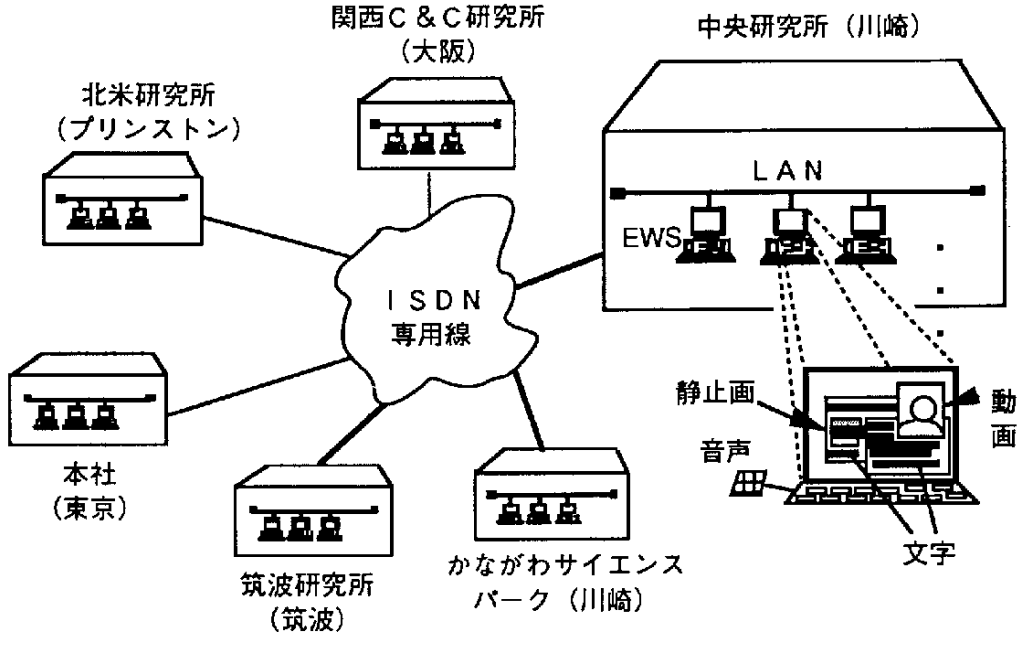


図7.2-3　現在のMERMAIDの接続地点

・　会議内容

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

・　利用頻度

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

・　利用人数

２～１５人、平均４人

・ 利用ＷＳ数

２～６台、平均３台（１地点で２台利用することも多い）

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

(1) 個人の自由の尊重について

(a) ウィンドウのフリーレイアウトについて

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

(b) 文書フリーアクセス、個人資料の処理について

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

(c) 個人作業の同時実行について

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

(2) 会議を越える会議について

(a) サブグルーピング、他の人の呼び出しについて

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

(b) 会議中の資料検索について

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

れる。

(c) 文書編集について

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

(3) グループ作業と個人作業の連携について

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

(4) 多様な会議への対応について

(a) 共有画面操作権の移行モードについて

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

(b) 操作権について

操作権は会議で一つあればよいか、黒板ごとに一つ必要かは議論の分れるところである。後者は利用者インタフェースが煩わしいと判断し、現在は会議ごとに一つの操作権を設けている。

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

(c) 定例会議と突発的会議の割合

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

(5) 分散会議の支援について

(a) 音声について

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

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

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

(b) 動画について

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

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

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

(c) 利用人数とデータ伝送遅延について

現在までのところ、一つの会議では WS４台以下で利用することが多い。WS の台数が多少増えてもデータ伝送遅延による不都合はさほど感じない。ただし、WS間で、文書の表示、スクロールのタイミングが２～３秒ずれることがある。

(d) 手書き（テレライティング）とキー入力について

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

(e) テレポインティング

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

(6) その他

(a) イメージ入力について

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

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

(b) 大画面ディスプレイについて

初めは1台のWSを一人が利用するという環境を想定して設計した。しかし、実際にはWSは十分な台数なく、WSをオフィスのコーナーに置き、1台の前に数人が座って会議を行うことも多い。この場合は、WSの画面を拡大して投影する装置があると利用しやすくなる。実際、40インチのディスプレイを利用してTV会議のように会議を行ったことがある。出力は皆に見えるが、マウスやタブレットは交代で利用しなければならず、操作が困難であるという問題が生じた。入力装置を全員が持つようにするにはハードウェアの改造も必要となる。

(c) 透明デジタイザについて

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

7.2.4 MERMAIDシステムの利用例

本節では、代表的な利用例を紹介する。現在、中央研究所（川崎）と筑波研究所（筑波）の研究員が協力して、筑波研究所内のスーパーコンピュータ(SX-2)を使って分子構造の解析システムを構築している（図7.2-4）。彼らはMERMAIDシステムの共有画面にソフトウェアの仕様案やモジュール分割案、分担案、進捗状況などを表示して検討したり、動画ウィンドウにプログラム実行結果を表示してチェックしている。スーパーコンピュータのカラーディスプレイは筑波研究所にしかないため、プログラムの実行結果（分子構造のグラフィックス）を見るために本来は中央研究所の研究員も筑波研究所に行かなければならないが、スーパーコンピュータの表示をMERMAIDシステムにより動画として中央研究所に送り、それをWSの画面に表示することにより、遠隔地でも結果を見ることができる。

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

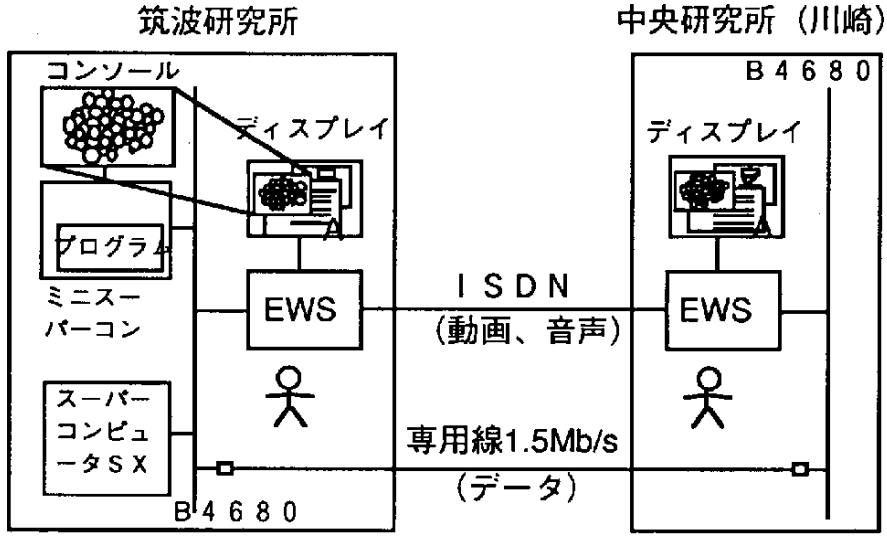


図7.2-4　MERMAIDシステムの分散ソフトウェア開発への利用例

7.2.5　MERMAID システムの応用分野

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

(a)　分散ソフトウェア協同開発支援

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

(b)　グループ意思決定支援

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

(c)　遠隔教育支援

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

(d)　マルチメディア文書協同編集支援

一つの文書に複数の人が同時にコメントを付けたり、それについて話し合ったりしながら、その文書をリアルタイムで編集し、配布することを支援する。

(e)　在宅勤務、サテライトオフィス勤務支援

通勤時間が長くなる傾向や女性の社会進出への対応として、在宅勤務やサテライトオフィスが注目され、実験されており [石割90]、これらの勤務形態を取る人が今後増加すると思われる。これらの人達と本社などのオフィスを結び、臨場感を持った打ち合わせや報告を行うことを支援する。

(f) 情報検索システムとの連動

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

(g) 他のアプリケーションと連動した業務支援

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

7.2.6 おわりに

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

【参考文献】

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

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

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

[石割90] 石割寿郎: 志木サテライトオフィス実験に関する一考察, 電気・情報関連学会連合大会講演論文集, 5-120～5-125 (1990).

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

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

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

[Watabe91] Watabe, K., et al. : Distributed Desktop Conferencing System with Multiuser Multimedia Interface, IEEE J. Selected Areas in Commun., Vol.9, No.4, pp.531-539 (1991).

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

7.3 KyotoDB による分散開発支援

7.3.1 はじめに

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

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

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

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

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

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

また、ソフトウェア開発グループ（チーム）のリーダーや、小人数での高度なソフトウェア開発における各作業者は、複数の問題を並行して処理することを要求されることが多い。このような場合、どの問題に対して誰とどのように情報交換すればよいのかを適切に判断することが重要である。しかし、同時に扱っている問題が多くなってくると、問題の把握と判断、およびその解決のためのコミュニケーションに割かれる労力が無視できないものになり、迅速な問題解決が困難になってくる。例えば、従来の電子メールを利用したソフトウェア開発を考えてみる。電子メールにより時間的障壁はある程度解消するものの、大量の電子メールの整理、内容の把握および処理に割かれる労力は小さくない。このような問題をグループワークにおけるコミュニケーション・オーバーヘッドと呼ぶ。

一般に、以上のようなグループ・コミュニケーションとグループワーク（協調活動）における諸問

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

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

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

7.3.2 協調活動モデル KCAM

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

KCAM では、ある特定の一つの目的のために行われる協調活動をアクティビティと呼ぶ。アクティビティの粒度は細かいものから大きなものまで種々のものが考えられるが、KCAM では比較的細かい粒度のものだけを対象とする。一つのアクティビティは以下の要素から構成される。

- 二つ以上の自律的に動作するエージェント
- エージェント間で交換されるメッセージ

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

さらに、KCAM エージェントが果たすことのできる機能を抽象化したものをファンクション、KCAM エージェントの入力イベントに対する振舞いを規定するものをルールと呼ぶ。ファンクション

がどう実行されるのかは、KCAMの対象外とする。

ルールの基本要素は動作部と条件部の二つである。動作部は、KCAMエージェントの動作をファンクションやその順序列の形で記述したものである。動作部は、無条件に実行される場合と入力イベントに対して実行される場合がある。後者の入力イベントの条件を記述する部分が条件部であり、入力イベントの種別とその内容に関する式を記述する。動作部 のみ あるいは 条件部 - 動作部の組をルール節と呼ぶ。

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

7.3.3 協調活動支援環境KAEのアーキテクチャ・モデル

(1) 概要

実世界の協調活動をKCAMを用いてモデル化する場合、協調活動の参加者である人間そのものを一つのKCAMエージェントとして記述し取り扱うことは非常に難しい。なぜなら、実世界では一人一人の参加者が同時に複数のアクティビティに参加し、それぞれ異なった役割を演じている場合があるからである。

例えば、全く別の開発プロジェクトAおよびBに同時に参加している作業者Wを考える。Wは、Aにおいては電子メールの発信を、Bでは電子メールの受信を行っているとする。プロジェクトAとBの間に全く関連がないとすれば、作業者Wを一つのKCAMエージェントとしてモデル化することは、情報隠蔽という観点からも適切ではない。明らかに、プロジェクトAにおける役割W_AとBにおける役割W_Bをそれぞれ別のKCAMエージェントとし、プロジェクトAとBを別のアクティビティと考えてモデル化することが、この場合には適切である。

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

- 内部エージェント

  アクティビティ毎の役割を演じるエージェント

- 外部エージェント

  外界のユーザまたは人工システム（外部存在）に1対1に対応するエージェント

KCAMエージェントは自律的に動作するエージェントであったが、これを分割した内部エージェン

トと外部エージェントも、それぞれ自律的に動作する。

アクティビティ、内部エージェント、外部エージェントとKCAMエージェントの関連を図7.3-1に示す。この図の例では、三つの外部エージェント U, V, W が存在し、それぞれ外界のユーザまたは人工システム（外部存在）に1対1に対応している。また、相互に関連のない二つのアクティビティ A と B が稼働し、KCAMメッセージがそれぞれの内部エージェント間で交換されている。さらに、各外部エージェントはアクティビティ A と B におけるそれぞれの役割を演じる内部エージェントと接続している。すなわち、アクティビティ A では、外部エージェント U, V, W に対応する外部存在がそれぞれ外部エージェント U, V, W を介し、内部エージェント U_A, V_A, W_A としてアクティビティ A における役割を演じている。また、アクティビティ B では、外部エージェント V, W に対応する外部存在が V, W を介し、内部エージェント V_B, W_B としてそれぞれの役割を演じている。したがって、外部エージェント V, W に対応する外部存在は、2つのアクティビティに別の内部エージェントを介して同時に参加し、それぞれのアクティビティにおける役割を演じていることになる。

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

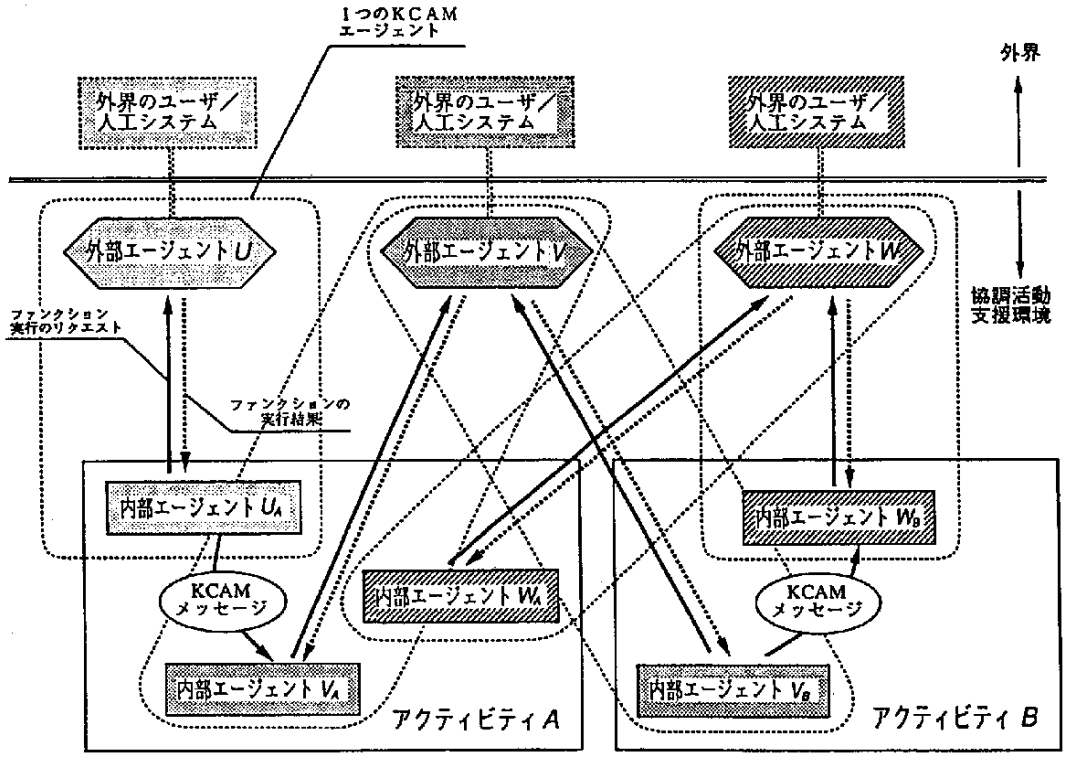


図7.3-1　アクティビティ、内部エージェント、外部エージェントの関連

(2) 内部エージェントおよびファンクションの分類

内部エージェントは、KCAMエージェントから特定の一つのアクティビティにおける役割だけを抽出したものである。外部エージェントは複数のKCAMエージェントに対応するが、内部エージェントはKCAMエージェントと1対1に対応するので、アクティビティにおいて内部エージェントを識別するためにKCAMアドレスを用いることができる。これは、図7.3-1において、点線で囲まれた1組の内部エージェントと外部エージェントが、一つのKCAMエージェントに相当することからも明らかである。

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

内部エージェントは、接続している外部エージェントがどのような外部存在に対応しているかによって、以下の三種類に分類することができる。

- Role Agent (RA)

  人間または非決定的な外部存在に対応する内部エージェント

- Service Agent (SA)

  決定性のある人工システムに対応する内部エージェント

- Independent Agent (IA)

  対応する外部存在のない内部エージェント

KCAMのレベルでは、ファンクションはKCAMエージェントの機能を抽象化したものであり、ファンクションが具体的にどう実行されるかなどの詳細はモデルの範囲外としていた。これに対し、アーキテクチャ・モデルのレベルでは、KCAMエージェントを内部エージェントと外部エージェントの二つに分割した。したがって、ルールは内部エージェントにより解釈実行され、ルール記述に現れたファンクションは内部エージェントから外部エージェントへのリクエストという形で実行される（図7.3-1）。アーキテクチャ・モデルでは、ファンクション実行の主体が何であるかという観点に基づいてファンクションを詳細化し、以下の三種類に分類する。

- Role Function (RF)

  人間または非決定的な外部存在に実行が委任されるファンクション

- Service Function (SF)

  外界の人工システムに実行が依頼されるファンクション

・ Independent Function (IF)

内部エージェント 単独で実行が完了するファンクション

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

各内部エージェントが、どのファンクションをそのルール記述で用いることができるかを表7.3-1に示す。これにより、RA はRF, SF, IF の全てを、SA は SF と IF を、IA は IF のみをそれぞれルール記述の中で用いることができるということがわかる。

表7.3-1 内部エージェントが用いることのできるファンクション

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

ファンクションはまた、アクティビティに特定の機能を果たすものと、一般的な機能を果たすものの二つに分類することができる。前者を Activity-Specific Function 、後者を General Function と呼ぶ。この分類は RF, SF, IF の分類とは直交するものである。KCAM メッセージの送信、受信とフェーズの遷移は、典型的な General Function である。

(3) 外部エージェント

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

(4) 協調活動スキーマ, パラメータ, インスタンス

これまで「アクティビティ」という用語を、協調活動そのものを指すものとして用いてきた。アーキテクチャ・モデルではさらに、協調活動スキーマ, インスタンス, パラメータ という概念を導入する。

協調活動スキーマ は個々の協調活動の枠組を定義するものであり、表7.3-2に示す要素からなる。さらに、一意な識別子（**協調活動スキーマ名**）を持つ。

協調活動パラメータは協調活動スキーマを稼働させるための情報であり、表7.3-3に示す項目がある。既に述べたように、アクティビティの稼働中は内部エージェントと外部エージェントの対応関係は変更されないので、協調活動パラメータの RA および SA のインスタンスの数と、それぞれが接続する UA および ESA の識別子はアクティビティの終了まで変更されない。

表7.3-2　協調活動スキーマの構成要素

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

表7.3-3　協調活動パラメータの項目

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

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

(5)　管理エージェント

Internal Management Agent（IMA）は、一つの協調活動インスタンスに一つだけ含まれる。IMAはまた、自分の管理する協調活動インスタンスに含まれる内部エージェントの表を持っている。この表の主な項目は内部エージェントのエージェント型、KCAMアドレスおよび最新のフェーズである。IMAがKCAMメッセージの配送を行う際には、受信者属性のKCAMアドレスを解釈し、この表を用いて対応する内部エージェントを特定してKCAMメッセージを分配する。さらに、内部エージェントがどのフェーズにあるかをモニタし、大域ルールを実行する。

Global Activity Management Agent（GAMA）は、KAE内に一つだけ存在する。IMAと同様に、GAMAはKAE内の全ての協調活動インスタンスの表を持ち、後述する協調活動インスタンスの連係において重要な役割を果たす。また、GAMAは協調活動スキーマのライブラリを管理しており、外部から協調活動スキーマ名とパラメータを与えられると、新しい協調活動インスタンスを生成する。

(6)　複数の協調活動インスタンスの連係

モデル化の対象になる協調活動が複雑なものであっても、協調活動全体を複数のアクティビティに分割することにより、KCAMによるモデル化が行える程度まで粒度を細かくすることが可能になる場

合が考えられる。このような場合、分割してモデル化した個々のアクティビティを連係させることにより、最終的に協調活動全体のモデル化が行える。

例えば、グループによるソフトウェア開発において、作業者 V は決定事項に対し、まずグループのサブリーダー L_1 の承認を得て、次にリーダー L_2 の承認を得なければならないという規則があるとする。これを電子メールを用いて行う協調活動のモデル化を考えてみる。全体を一つのアクティビティとしてモデル化することも可能ではある。しかし、承認を得る手順 P が、相手がサブリーダーかリーダーかに関係なく同じだとすると、作業者 V は以下の二つの手順を直列に実行することにより最終的な承認を得ることができる。

① サブリーダー L_1 の承認を得るための手順 P

② （①が成功した場合）リーダー L_2 の承認を得るための手順 P

すなわち、承認を得る手順 P を一つのアクティビティとし、このアクティビティを二つ直列に実行する。承認を得る必要がある相手の人数が多くなる場合を考えると、このようなアクティビティの分割とその連係によってモデル化が容易になることは明らかである。

KCAM では、複数のアクティビティ間の相互干渉はありえないとしていた。アーキテクチャ・モデルでも、稼働中の協調活動インスタンス相互の干渉はありえない。ただし、アクティビティの分割と連係を可能にするため、協調活動インスタンスの起動と終了においてのみ、二つの協調活動インスタンスが同期できるものとする。これは大域ルールにおいて、協調活動インスタンスの起動と終了待ちを司る大域ファンクションである *fork* と *wait* を用いることにより実現される。

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

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

fork, wait は、ともに IMA で解釈され、IMA から GAMA へのリクエストにより実行される。図7.3-2に、*fork* と *wait* による協調活動インスタンスの連係のパターンを示す。

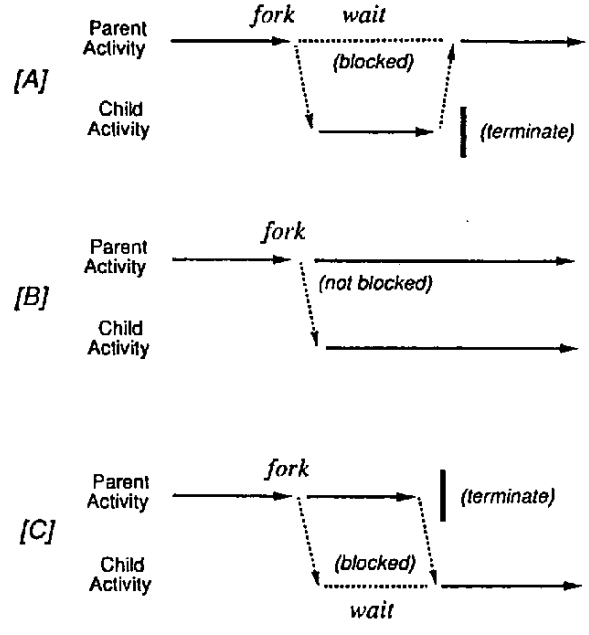


図7.3-2 協調活動インスタンスの連係のパターン

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

7.3.4 協調活動スキーマ記述言語 KCASL

協調活動スキーマ記述言語 KCASL は、協調活動スキーマを実行可能な形で記述するための言語である。KCASL は、第7.3.6 項で述べる UNIX 上での実装を前提に、C言語をベースにして設計されている。KCASL では表7.3-2に示されたルール節や、その他の協調活動スキーマの要素を記述するための文法が定義される。

KCASL の設計上の要点を以下に列挙する。

(1) メッセージ型 と KCAM メッセージ の取扱い

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

(2) select 文によるルール節の記述

ルール節は、RuleDescription 節に記述される。C言語は並列に条件を評価する構文を持たないため、select 文を追加してこれを記述する。KCAM では入力イベントを KCAM メッセージの受信と 時間イベントの二種類としていた。KCASL では、Role Agent のルール 記述の場合のみ、Role Function の実行を入力イベントとして条件部に記述できる。これにより、複数の Role Function の実行を一つのフェーズで待つようなルール記述が行える。

(3) 変数

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

(4) ファンクション の宣言と呼び出し

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

(5) 制御構造

Independent Function の中や、select 文の各後続部では、C言語の任意の制御構造が使用できる。た

だし、後続部から外部へ脱出したり、他の後続部や Indepent Function の中に分岐するような制御構造の記述は許されない。

7.3.5 KyotoDB における協調活動支援

(1) KyotoDB の概要

KyotoDB は、ソフトウェア開発のプロセスと生産物（プロダクト）を管理するソフトウェア・エンジニアリングデータベース（KyotoDB カーネル）と その開発支援環境全体の総称である。KyotoDB の目的とするところは以下の三つである[鯵坂92]。

(a) 生産物とその要素相互の意味的関係の管理

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

(b) 生産物とプロセスの統合管理

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

(c) 協調作業の支援

これが KAE の機能に対応する。

KyotoDB では、作業者一人の担当する作業単位を Unit Workload（UW）と呼ぶ[Matsumoto90]。また、UW によって作成される生産物を 目的生産物 と呼び、目的生産物を作成するために参照される生産物を 参照生産物 と呼ぶ。一つのソフトウェア開発プロジェクトは、参照関係に従って構成された UW の半順序集合（UW ネットワーク、UWNet）という形で扱われる。UWNet はプロジェクトの開始時に作成されるか、必要な場合にはプロジェクトの進行に伴って変更、作成あるいは分割されていく。プロジェクト全体の再利用は、記録されていた UWNet を元に必要な部分にのみ変更を加えることで行える。

(2) KyotoDB カーネル

KyotoDB カーネル は、オブジェクト指向データモデルに基づいており、以下の四つのクラスのオブジェクトが複合構造をなしている[鯵坂92]。これを図7.3-3に示す。

(a) Project Object

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

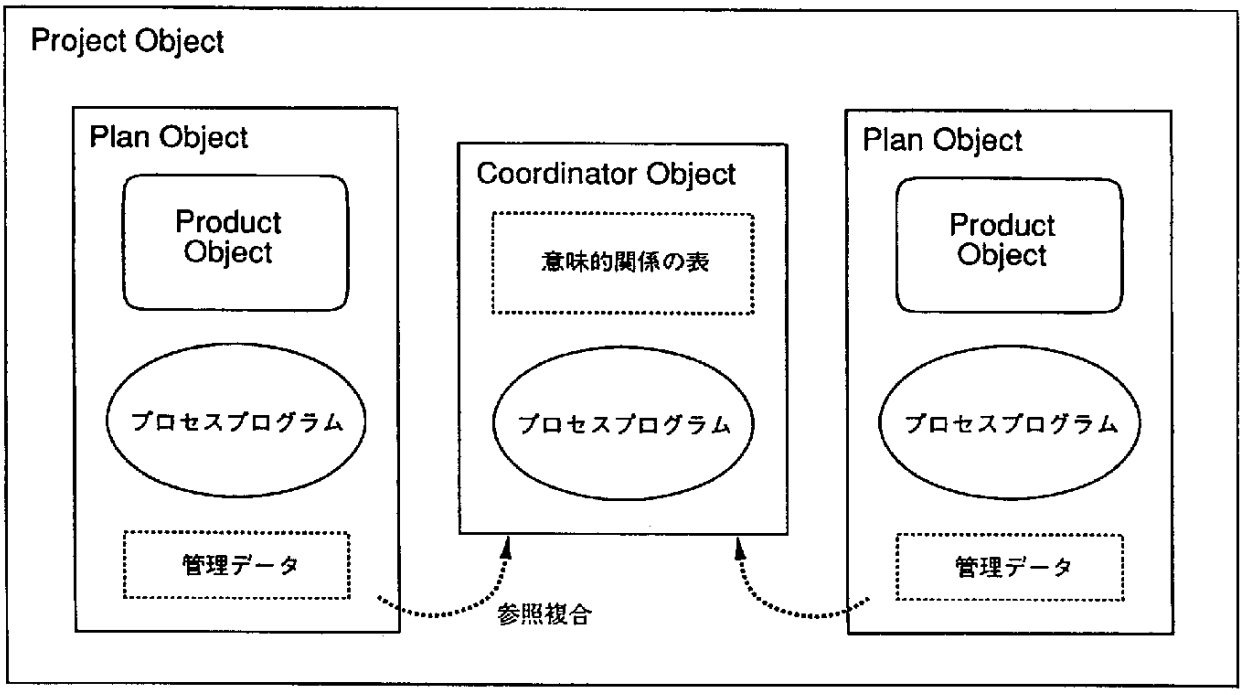


図7.3-3　KyotoDB カーネルにおけるオブジェクト

(b)　Plan Object

一つの UW のプロセスプログラムと、その管理データを格納するオブジェクト

(c)　Product Object

一つの UW の目的生産物を格納するオブジェクト

(d)　Coordinator Object

二つの生産物の相互の意味的関係を管理し、生産物の変更による再作業の発生の可能性(波及効果)を検出して、その解決のためのプロセスプログラムを格納するオブジェクト

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

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

(3) 作業者1名のプロセスプログラムの実行

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

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

以上のプロセスプログラム（以下 "Plan1" と表記）を、KAE において実働化することを試みる。まず、Plan1 は KyotoDB カーネルと UW の担当者という二つの KCAM エージェントによるアクティビティであると考える。また、操作列や O-Stream は形式化されているので、KCAM メッセージとして取り扱える。したがって、KCAM によるモデル化と協調活動スキーマ記述を行えば KAE 上で稼働させることができる。以下、必要なエージェント型、メッセージ型およびファンクションについて述べる。

まず、Plan1 の協調活動スキーマにおけるエージェント型は以下の二種類であると考えられる。

- PLAN

  UW での KyotoDB カーネルの役割を演じる SA

- WORKER

  UW の担当者の役割を演じる RA

主なファンクション には以下のものがある。

(a) 単位操作( )

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

(b) コミット( )

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

(c) 作業完了( )

作業完了に対応する WORKER の RF。前回のコミットから単位操作が行われていない場合だけ実行

が許される。理想的には、KyotoDB カーネルが UW の完了を検査する必要があるが、ここでは作業者の判断に任されるとしている。

(d) 目的生産物更新( )

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

また、主な KCAM メッセージは以下のとおり。

(a) productMsg

目的生産物を最初に WORKER に送る

(b) commitMsg

コミット( ) の際の操作列を送る

(c) doneMsg

作業完了( ) を PLAN に通知する

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

(4) 協調活動支援

現在の KyotoDB カーネルでは、任意の一つの UW は特定の 1 名の作業者にだけ割り当てられる。すなわち、複数の作業者に共有される UW（目的生産物）は存在しない。したがって、目的生産物と参照生産物の関係の一貫性が損なわれた場合にのみ、複数の作業者の協調活動が行われる。このような関係の一貫性の破れ（波及効果）は、ある UW における作業によって目的生産物が変更された時に、それを参照している UW が既に存在し、その間に関係が定義されている場合に発生しうる。

この節では、第7.3.5項の(3)で述べた作業者 1 名のプロセスプログラム "Plan1" に対し、コミットによる波及効果の解決のための機能を追加することを考える。すなわち、第7.3.3項の(6)で述べた協調活動インスタンスの連係を利用して、もう一つの協調活動インスタンス（子アクティビティ）を起動し、波及効果の解決を任せる。その際、波及効果の内容に応じた協調活動スキーマ名を決定し、パラメータを波及解析によって求める。協調活動スキーマ名は、波及効果の緊急度に基づいて決定できる。すなわち、緊急度の高い場合はリアルタイム型を、低い場合は非同期型のスキーマを選ぶことになる。協調活動パラメータは、波及効果の及ぶ他の作業者が誰であるかを示す。

また、子アクティビティとの連係のパターンは、波及効果の重要度に応じて変化する。すなわち、

重要度が高い場合には、子アクティビティの結果によってはコミットが失敗することがありうる。一方、波及効果の重要度が低い場合は、子アクティビティの結果に関わらずコミットは常に成功する。これらは、図7.3-2に示したパターンAとBにそれぞれ対応する。

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

(a) 波及効果解析( )

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

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

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

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

(a) noProblemMsg

問題となるような波及効果がなかった

(b) forkMsg

子アクティビティの協調活動スキーマ名 など

(c) solvedMsg

子アクティビティによる解決が成功した

(d) notSolvedMsg

子アクティビティによる解決が失敗した

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

図7.3-4に、Plan2 の協調活動スキーマ のグラフによる表現を示す。

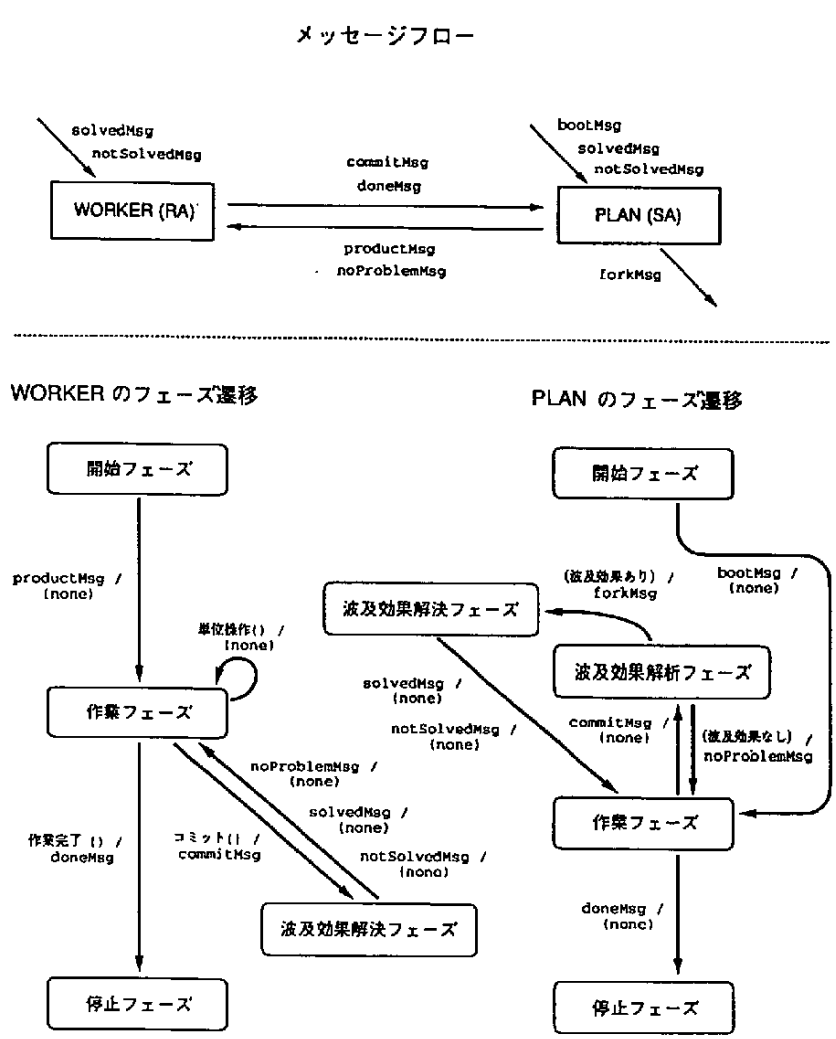


図7.3-4　Plan2 の協調活動スキーマ のグラフによる表現

7.3.6　KAE の実装

(1)　アーキテクチャ・モデル の UNIX モデルへの変換

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

(a)　内部/外部/管理エージェントの実現

アーキテクチャ・モデルでは、これらのエージェントは全て自律的に並行動作する。よって、UNIX では各々をプロセスとして実現する。ただし、UNIX は種々のネットワークサービスを提供するが、プロセスの制御に関しては完全な位置透過性を持たない。すなわち、ネットワークを介して他のマシンの上のプロセスを制御する場合と、ローカルマシンのプロセスを制御する場合では、制御の方法が異なっている。したがって、KAE は単一のマシンの中で閉じているものとする。また、外部エージェ

ントのうちUAは、外界のユーザ1名に対し一つ存在する。したがって、ユーザのワークステーションで稼働しているXウィンドウシステムのサーバ（Xサーバ）をUAと見なす。

(b) KCAMメッセージの送受信

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

(c) Service Function, Role Functionの実行

SFとRFはそれぞれ、内部エージェントから外部エージェントへ実行が依頼または委任される。SFは外部エージェントから外界の人工システムへのリクエストにより実行される。KAEでは、UNIX上の他の受動的システムのみを人工システムとして取り扱う。RFは、前述したようにユーザの自発的な選択や作業に相当する。KAEでは、ユーザがRFを実行するための手段として、ツールを導入する。RFの粒度には種々のものがあるため、ツールには全体が一つのRFに相当するものや、ツールのコマンドがそれぞれRFに相当するものなどがあると考えられる。つまり、あるフェーズにおいてルール節に現れているRFは、ユーザによるツールの実行やツールのコマンドの選択肢に対応する。ツールは内部エージェントから起動され、内部エージェントと双方向に通信する。ここでもソケットによるプロセス間通信を用いる。

(2) KCASLの実装方針

第7.3.4項で述べたように、KCASLはC言語をベースに設計されている。したがって、拡張された文法要素を変換し、C言語のみで記述できる形式に帰着させることにより、UNIX上のC言語の言語処理系とライブラリがそのまま利用できる。このような変換を行う処理系をKCASLプリプロセッサと呼ぶ。この項では、KCASLプリプロセッサによる変換の方針と、変換されたプログラムを実働化するために必要なライブラリの機能について説明する。

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

しない形式に変換するためのパッケージである。KCASL によるメッセージ型の定義は、そのまま XDR の定義に変換される。

ルール節では、複数の入力イベントを並列に待つ構文である select 文が拡張されている。入力イベントには、KCAM メッセージの受信、RF の実行および時間イベントの三種類である。KCAM メッセージの受信と RF の実行は、それぞれ IMA と接続した外部エージェントからのソケットを介してのデータ受信に対応する。したがって、この二つは UNIX のシステムコール select() により並列に待つことができる。時間イベントは、UNIX のタイマ機構を利用し割り込みにより検出する。select() の実行中も割り込みは有効なので、三種類の入力イベントを並列に待つことができる。

KCASL による記述から最終的に UNIX の実行ファイルを生成し、協調活動スキーマを実働化する過程を図7.3-5に示す。現時点では、図中の KCASL プリプロセッサを除く部分について実装が完了している。また KCASL による変換方針も確立している。

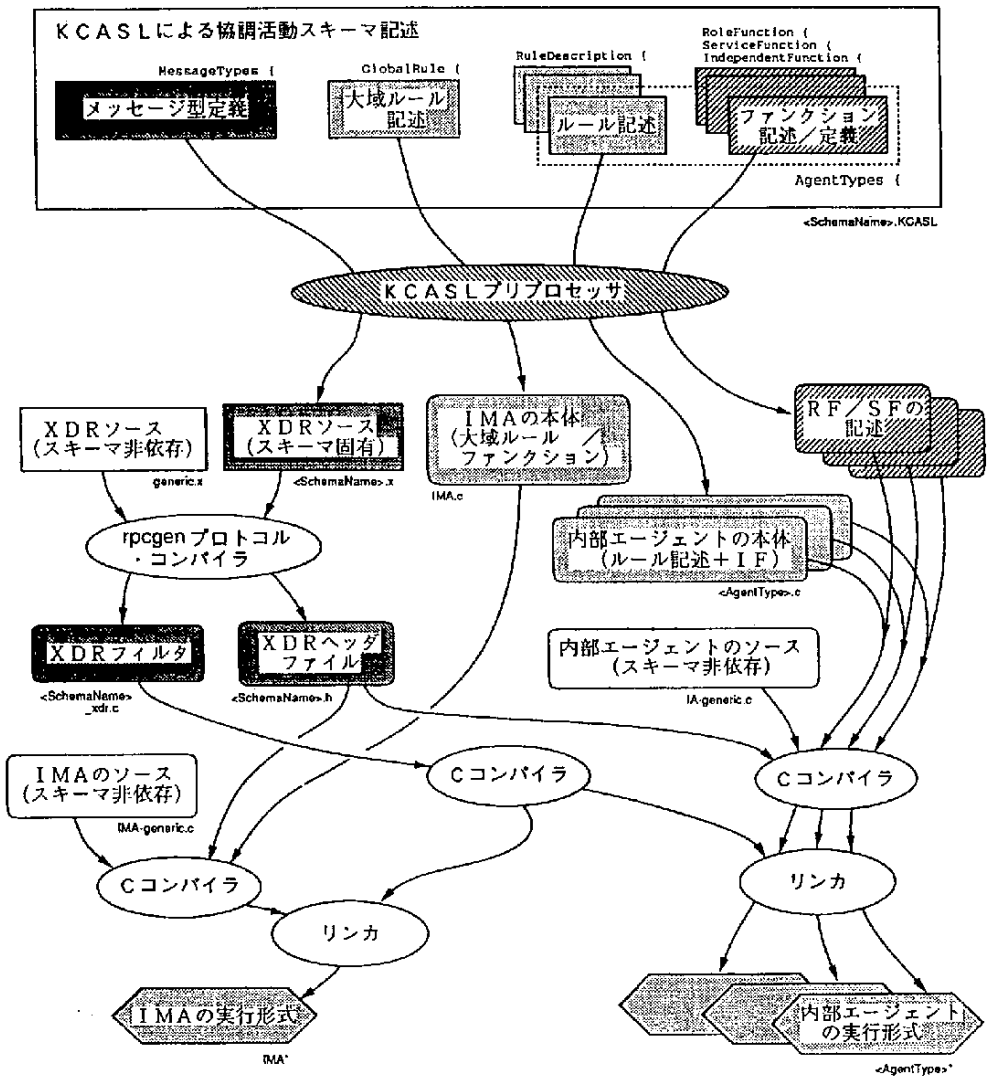


図7.3-5　協調活動スキーマの実働化の過程

(3) ツールの実現と制御

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

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

7.3.7 おわりに

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

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

将来の課題としては、以下のものが挙げられる。

(a) KCAM の拡張

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

また、大域ルールは通常のルールに対するメタレベルのルールであると考えられる。現在は、大域ルールは疑似コードのレベルでしか記述できていないが、このようなリフレクションの概念を導入することにより、厳密な記述が可能になると考えられる。

(b) モデル化と スキーマ 作成の方法論の確立

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

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

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

(c) 分散化およびオープン化

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

(d) UWの作成方法論の確立

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

【参考文献】

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

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

[垂水92] 垂水 浩幸: 分散開発環境 : グループウェアのソフトウェア開発への応用,情報処理, Vol 33, No.1, pp. 22-31 (1992).

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

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

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

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

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

7.4 協調支援環境 Vela による分散開発支援

7.4.1 ソフトウェア開発における協調作業に関する問題点

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

(1) データベース技術が支援する成果物の管理機構

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

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

等の存在が必要であり、それらの要素技術が組合わせられ、適合されて、

(1) 各自の持つ不完全で部分的な知識の統合

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

(3) 各作業単位毎に、必要な視野やコンテキストを設定すること

等に関する有効な支援が実現されなければならない[落水92-1]。

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

(1) マクロなレベルでは、距離的に離れた場所に存在する開発チーム間のコミュニケーションの支援が十分でなく、また、工程対応で分割されたソフトウェア開発プロセスが分散開発に適していない等の問題がある。

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

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

(1) 人間は忘れやすい生き物である（時空間分散に弱い）

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

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

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

7.4.2 ソフトウェア開発環境の近未来像

ソフトウェア開発環境の近未来像として筆者は図7.4-1に示すようなサービスが必要であり可能であると考えている[落水91-1]。

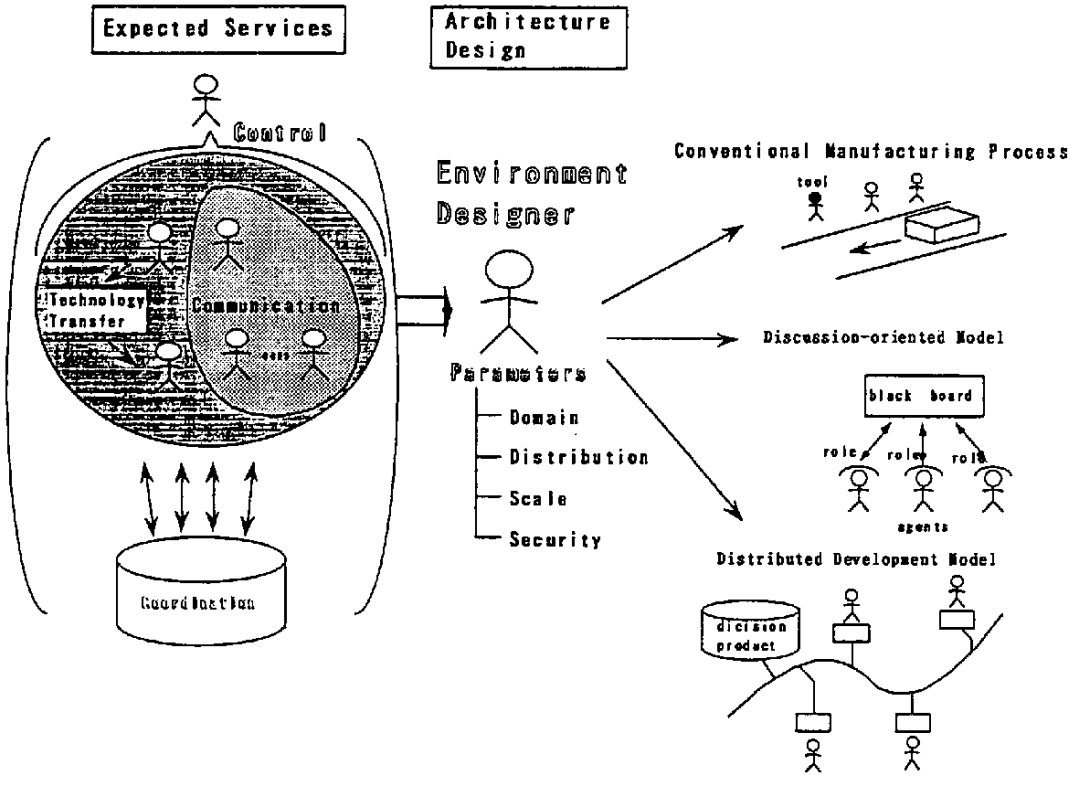


図7.4-1 ソフトウェア開発環境の近未来像

図7.4-1において、環境利用者へのサービス機能としては、

(1) Coordination支援（ソフトウェアオブジェクトベース）

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

(2) Navigation支援（熟練者の作業プロセスの再利用）

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

(3) Communication 支援（情報交換・伝達の支援）

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

(4) Control 支援（集団活動の制御）

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

の四つが挙げられる。また環境開発者への支援機能も必要である。

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

上記四つの機能はプロジェクトチームやプラットホームによってその必要度や実装法が異なる。例えば、数人からなるスキルレベルの高いチームの場合はCoordination支援が支配的となりCommunication支援も軽く実装すればよい。一方、100人以上の構成員からなるチームの場合はControl支援が支配的となり、プロジェクトの目標達成にあたって予想される問題点（開発過程で生じうる制約）をクリアしつつ大勢の人間の歩調をそろえさせるプロジェクト立案が重要である。Communication支援も合意形成や意思伝達確認に力点をおく必要がある。すなわち、プロジェクトの規模[Per88]、対象分野の特性、チーム構成員の分散度、データの安全性や機密度、利用する計算機設備等のパラメータに応じてメタ SDE（Software Development Environment）からスペシフィック SDE を生成する環境開発者の支援が必要である [Tul90]。

7.4.3 Vela 開発の背景と目標

図7.4-2 は、図7.4-1 における五つのサービス機能のうち Adaptation を除く四つの機能について、研究室で開発中のプロトタイプ Vela における機能とその相互関係を示したものである [落水90]。プロトタイプ開発の目的は、前節でのべた要請に対する有効な仮説を設定し、モデルを設計するための実験設備を構築することである。

第7.4.1 項で述べた問題点に対して以下のような問題点の切り出し方をすることにより、ナビゲーション支援とコミュニケーション支援に関して、現在の研究の力点をおいている [落水92-2]。

(1) 協調開発における個人の問題

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

(2) 協調開発におけるチームの問題

「多大な工数を費やして開発しなければならないようなアプリケーションにおいては、すべてを理

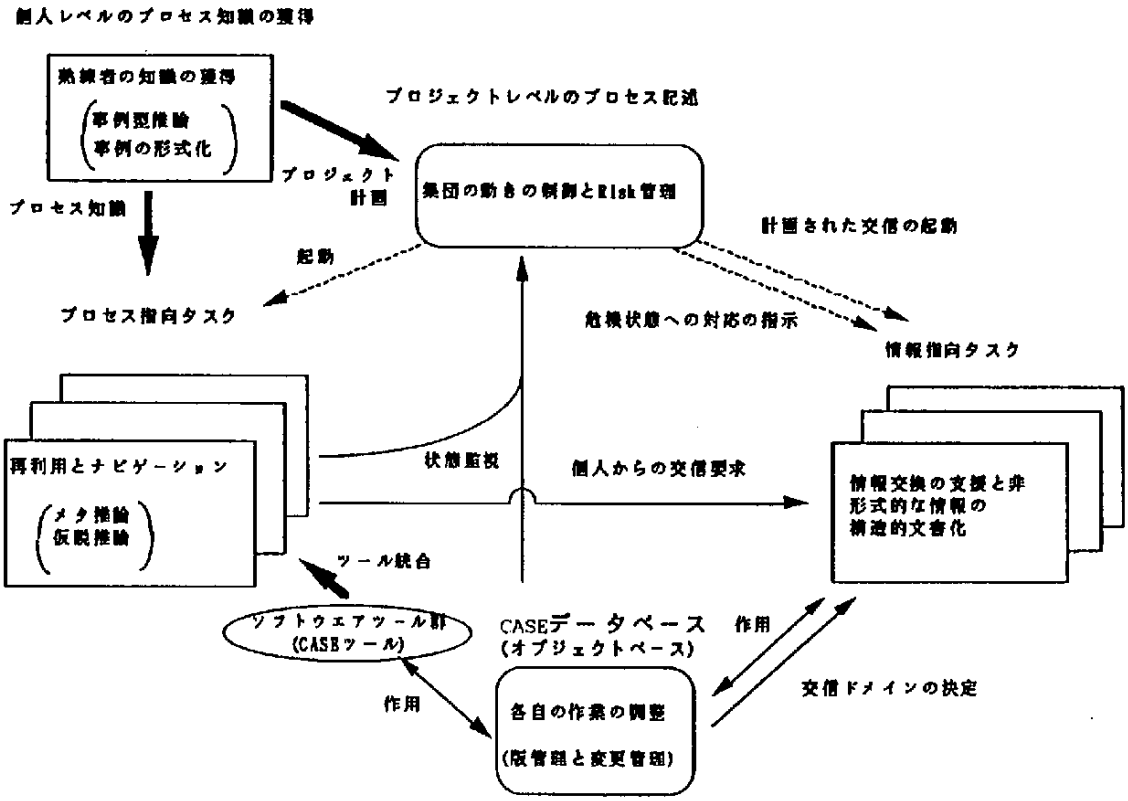


図7.4-2　Vela機能の相互関係

解している人は組織のどこにも存在せず、組織として理解していることの全体は多くの経験豊富な専門家一人一人に分散している[シュレ90]」。この事実は開発規模の増大につれて顕在化し、分散環境下においてはさらに増幅される傾向にある。

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

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

このような組織モデルに支援された人間集団が図7.4-3に示すようなソフトウェア開発環境モデルに以下のようにアクセスする。

(1)　並行的に操作され詳細化されるソフトウェアオブジェクト群を

(2)　CASEツールや個人の活動により、その状態を変化させ、

(3) ツール群や各自の活動のプロセスモデルによる統合により、

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

を保証されつつ操作する。

以下の項では、それぞれの問題に対する研究の進行状況をまとめる。

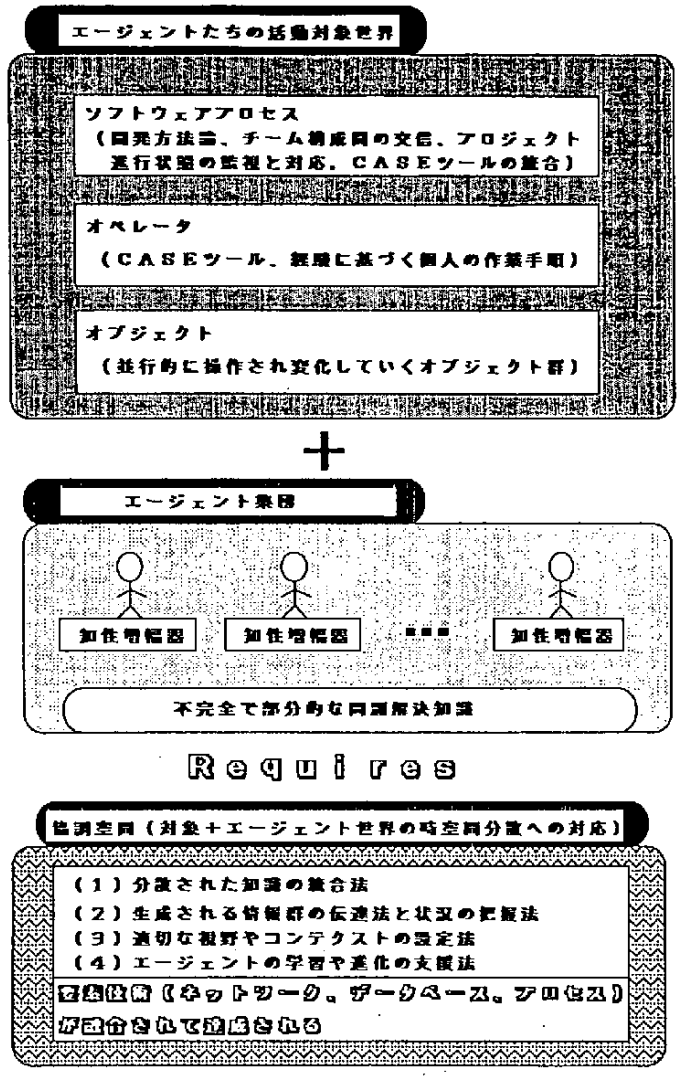


図7.4-3 開発環境と人間集団

7.4.4 Navigation 支援に関する研究の現状

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

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

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

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

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

① ソフトウェアプロセスに関する知識の分類

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

② 知識獲得・形式化機構の開発

前項で述べた知識の獲得・形式化を支援するプロトタイプを知識工学における事例ベース推論、モデル推論の技法を適用して開発した（図7.4-4）。

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

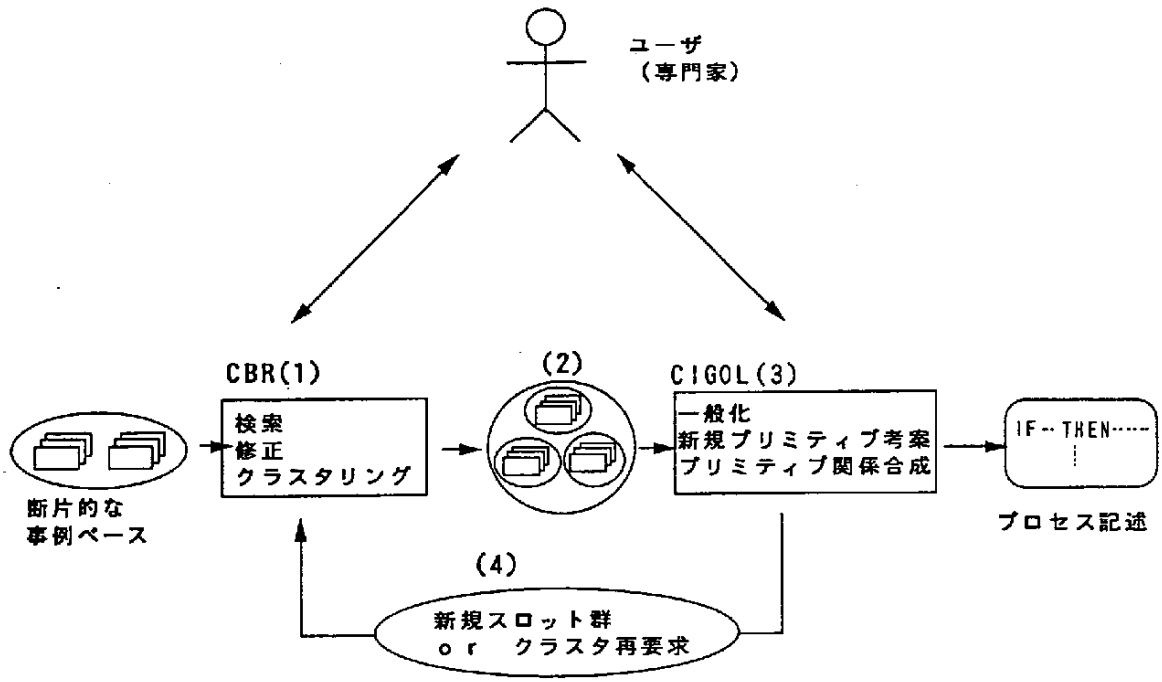


図7.4-4 Velaにおける知識獲得機構

③　上記機構の SA/SD 法への適用実験

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

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

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

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

(2)　大学にいる優秀なデザイナの行動分析をおこなう。

(3)　また、知識利用機構に関してもプロトタイプの開発を進める

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

7.4.5　Communication 支援に関する研究の現状

分散開発環境下にあっては、協調作業のための時空間の制約が厳しく、そこを緩和する方式が明確でない。一般に協調支援機構は、データベース技術が支援する成果物の管理機構やネットワークによる情報交換機構を基に、第7.4.1項で述べた機能に融合することで開発可能である。最近 CSCW の技術が注目を集めているが、ソフトウェア開発固有の交信を支援するには、本項の①で述べるようにまだ十分とはいえない。

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

(1)　CSCW に関する技術現状のサベイ

①　e-mail の効果と限界[Fel86]

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

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

② 準構造化メッセージの効果と限界 [Mal87]

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

③ ハイパーテキストの効果と限界 [Con88]

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

(2) 交信ドメインの概念

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

これに対して、論点対応に、関係者の間に永続的な仮想リンク（交信ドメイン）を設定し、グループ内の検討内容を発言の型に応じて構造的に記録するような交信モデルを設計し（図7.4-5）、TCP／IPのアプリケーション層の一つとしてプロトタイプを開発し、さらに、デザインレビュー支援機構を試作した [落水91-2]。

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

(3) 協調空間の概念の導入 [佐塚92]

ソフトウェア開発にたずさわる人々をその作業の型(エージェントの型)によって、以下の三種に分類する。

① 部分問題解決型エージェント

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

② 調整型エージェント

部分問題解決型エージェントの協調を支援する。すなわち、部分問題解決型エージェントの個々の

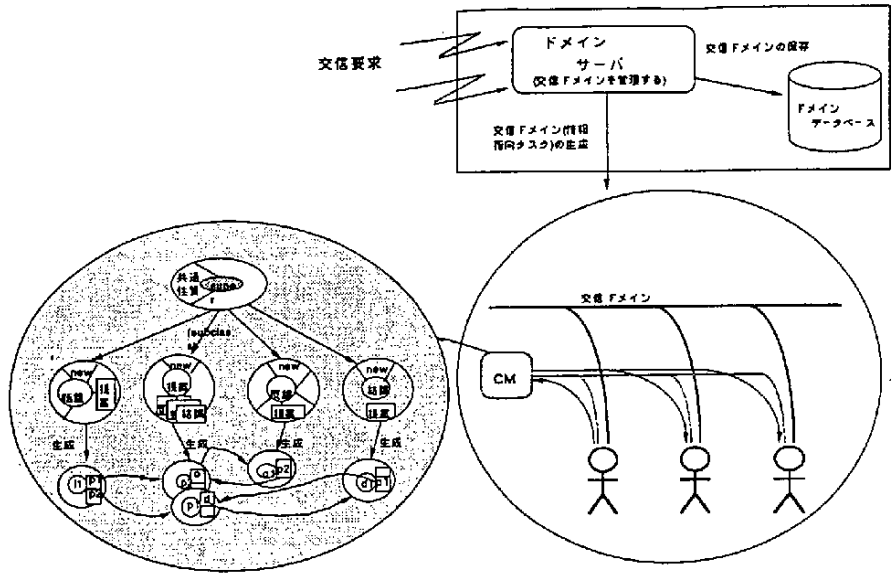


図7.4-5 交信ドメイン（論点対応の仮想リンク）

活動や全体の状況を把握しながら、チーム作業の進行状況分析、二次的な問題の解決、異常状況の検出とその対応を行う。それに加えて、組織全体の状況を広く把握し適切な対応をとるために必要な情報源とその探索法を知っている。

③ 開放型エージェント

現在の課題を達成することより、むしろ新しい問題解決法を工夫するために、発想し、思考し、開拓する。問題解決チーム間にまたがって情報交換を行い、また外部世界との情報のやりとりを行う。

各エージェントの相互関係を図7.4-6 に示す。

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

エージェント間の相互作用とは、ある作業環境から出ていく情報が別の作業環境に入る時におこる情報の蓄積と型変換の操作であると定義しこれを協調プリミ

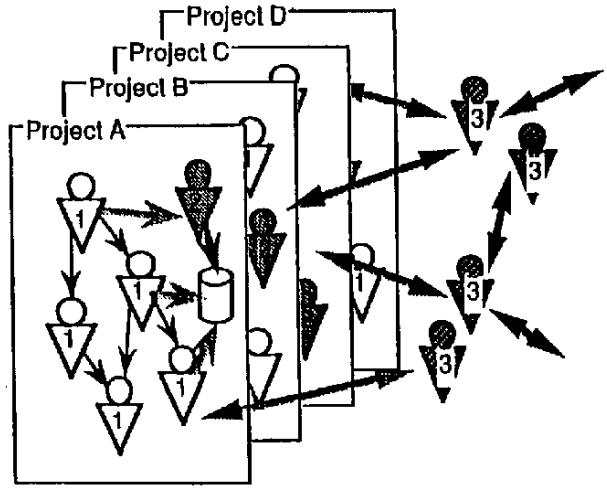


図7.4-6 各エージェントの相互関係

ティブとよぶ。現在抽出した協調プリミティブは以下の五つである。

① 協調プリミティブ1（生成物の蓄積と伝達）

情報の蓄積と伝達の機能を同時に実現し、プロジェクトに公開する。

② 協調プリミティブ2（状況の収集・蓄積と検索時のフィルタリング）

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

③ 協調プリミティブ3（計画情報の分配）

作業計画を関連するすべてのエージェントに分配する。

④ 協調プリミティブ4（状況把握）

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

⑤ 協調プリミティブ5（情報パイプ）

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

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

7.4.6 おわりに

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

筆者が開発中の協調支援環境 Vela は以下の二点からその問題にアタックしている。

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

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

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

【参考文献】

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

7.5 RTPとPerson to Person／2による分散開発支援

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

7.5.1 リアルタイム・プレゼンテーション・システム（RTP）

RTPは、日本アイ・ビー・エム東京基礎研究所において開発された試作システムであり、遠隔地にいるパーソナルシステム／55™のユーザ同士が、マルチメディアの会議資料を同時に共有しながら、リアルタイムの打ち合せができるシステムである。ネットワークとしてはISDN基本インタフェース（2B＋D、NTT INSネット64）およびトークン・リングLAN（4または16メガビット／秒）を使用する（システム構成については図7.5-1参照）。

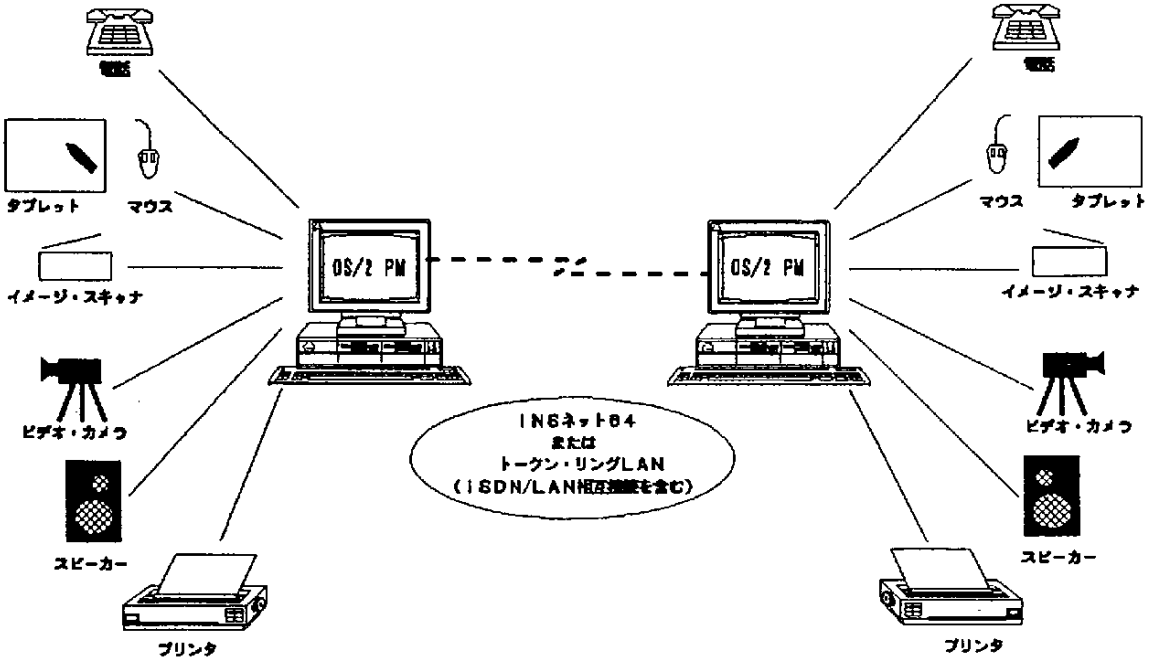


図7.5-1 ＲＴＰのシステム構成

RTPは、あたかもOHP（オーバ・ヘッド・プロジェクタ）を使ったプレゼンテーションを行うように、予め作成された『マルチメディア・フォイル』と呼ばれる会議資料を使って、遠隔地にいる相手と対話的な打ち合せを行うことができる（RTPの画面サンプルを図7.5-2に示す）。現在一般に利用できるRTP試作システムでは、二人のユーザが、LANあるいはISDNを通じて一対一で使う構成のみをサポートしているので、今後この二人のユーザを、しばしば『自分』と『相手』という表現で呼ぶことにする。P2Pも二人のユーザ構成のみをサポートしている。

図7.5-2の画面サンプルで、画面の右上に表示されている相手の顔は、会議用電話帳と共に予め用意された静止画であるが、既にH.261（P×64）等のリアルタイムの動画も実験済みである。P2Pは『アクションメディアII』というボードを使って、DVI™（ディジタル・ビデオ・インタラクティブ）のRTV（リアル・タイム・ビデオ）の圧縮方法によるリアルタイム動画を実現している。

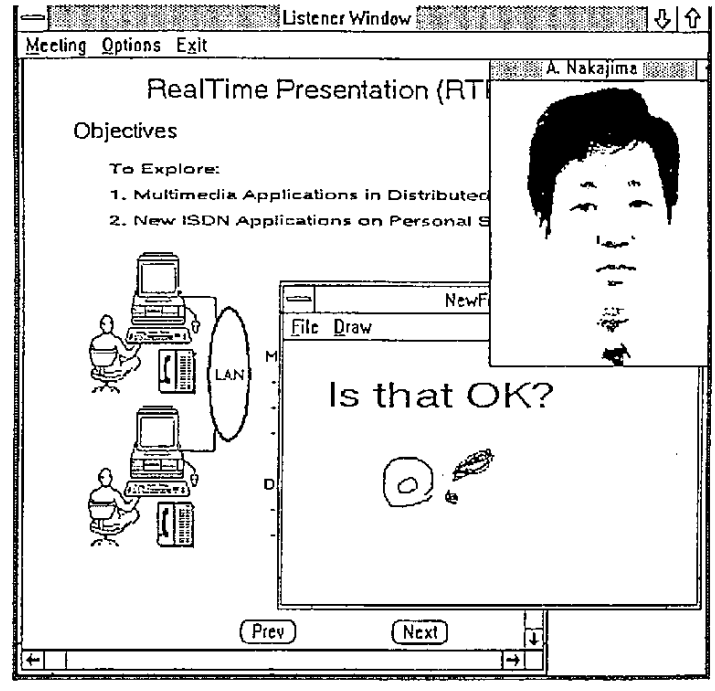


図7.5-2　ＲＴＰのサンプル画面

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

(1)　マルチメディア・フォイル表示機能

文字、グラフィックス、イメージ・データから成る複数ページのメタファイルを、OHPシートを順番にOHPの上に乗せていくように、相手との共有ウインドウに順次表示していき、タブレットやマウスを使ってメタファイル上に注釈を付けることができる。

図7.5-3にマルチメディア・フォイル・ブラウザのウインドウの様子を示す。また、MAVC（マルチメディア・オーディオ・ビジュアル・コネクション）を使って取り込んだ音声と静止画を、相手と同

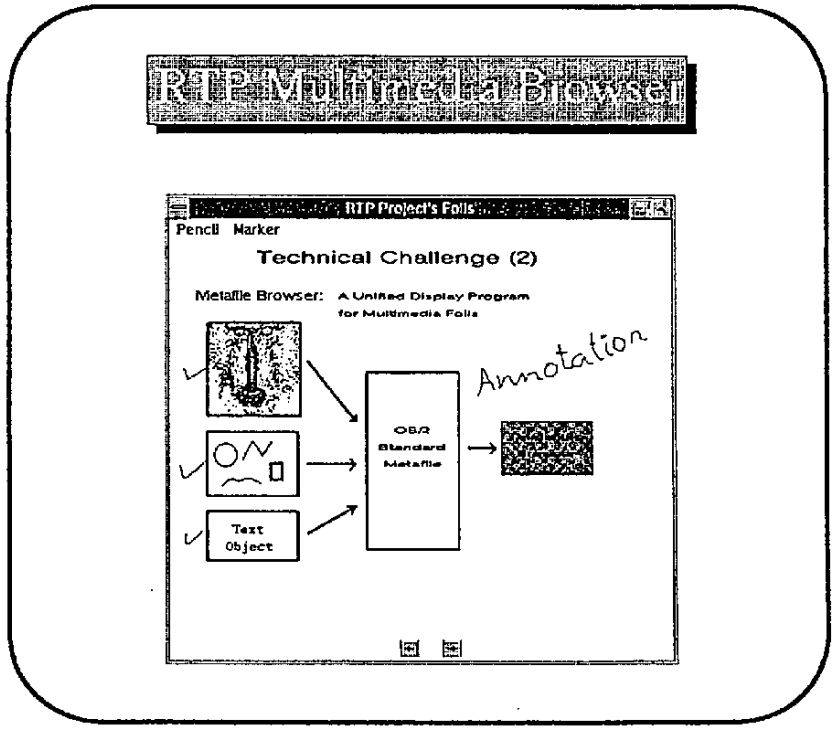


図7.5-3　マルチメディア・フォイル・ブラウザのウィンドウ

時にプレイバックする「メッセージ・フォイル機能』も存在する。会議資料としてのマルチメディア・フォイルはデータ量が多いため、会議実行前に予め転送しておくことで、RTP実行時のデータ転送量を最小限とし、ISDN基本インターフェースの64キロビット／秒のBチャネルでもリアルタイム性を持たせることができる。この会議資料の転送はRTPのファイル転送機能を利用して行われる。

(2)　リモート・ポインタ

共有ウインドウ上でのある参加者のマウスやペンの動きが、他の参加者にリアルタイムに伝わる機能で、一般的にグループウェアではテレポインティングと呼ばれている。リモート・ポインタの制御方法として様々なものが検討されたが、ここでは『疎結合（Loosely Coupled）方式』ついて紹介したい。この方式は、自分がポインタを動かした時に、もし或るタイムアウト時間（ネットワークの遅延時間等から決められる）内に相手のポインタが動かなければ、自分がリモート・ポインタの制御権を取り（相手のポインタは自分のポインタの動きに従って動く）、もしこのタイムアウト時間内に相手も動いた場合は、自分のポインタも相手のポインタも独立に動く、というものである。図7.5-4にこの疎結合方式リモート・ポインタのメカニズムを示す。一方P2Pにおいては、自分と相手それぞれが別のリモート・ポインタを、『チョークボード（Chalkboard）』と呼ばれる共有ウインドウ上で使うことができる。

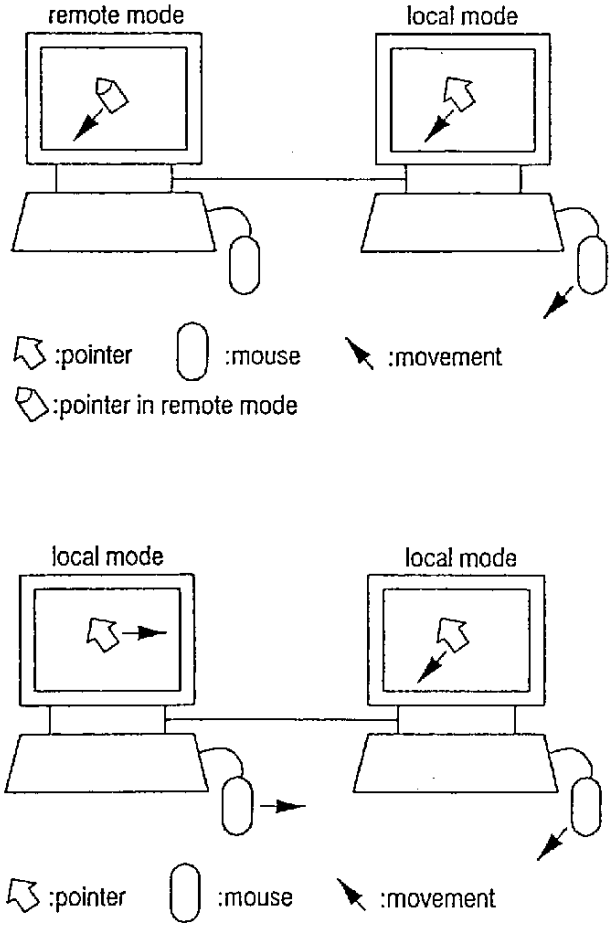


図7.5-4　疎結合方式リモート・ポインタ

(3)　対話的なフォイルの作成

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

(4)　静止画の転送

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

7.5.2 Person to Person／2

米国IBMで1991年10月に発表された製品（顧客独自の必要性から開発された特殊製品）であり、パーソナルシステム／2™のユーザ同士が、トークン・リングLAN、イーサネット、またはISDN基本インタフェースを通じてリアルタイムの打ち合せができるシステムである。図7.5-5にP2Pのシステム構成図を示す。主な機能としては、上で言及した『チョークボード』と呼ばれる共有ウインドウと、DVIの動画を利用した『ビデオ・ウインドウ』があり、以下により詳細に解説したい。P2Pは、さらに『チャットライン（Chatline）』と呼ばれるテキスト・メッセージの交換機能、ファイル転送機能などを持っている。リアルタイムの音声については、構内ならばPBX、遠隔ならばデータ通信に使われるISDNとは別の電話回線を利用してやり取りする。

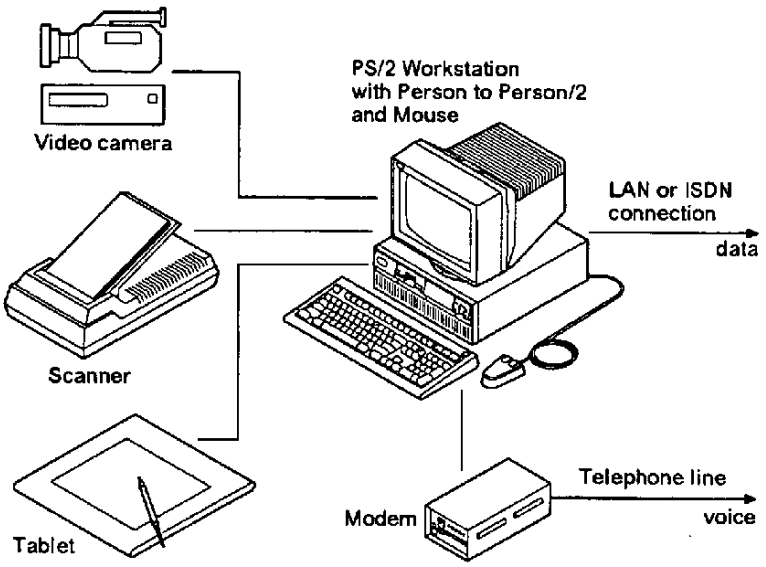


図7.5-5　ＰＴＰ／２のシステム構成

(1)　チョークボード

チョークボードは、OS／2 PMのウインドウ上で、一般的にグループウェアで『共有黒板』と呼ばれているような機能を提供する。ユーザは複数個のチョークボード・ウインドウを画面上に開くことができ、それらを相手のユーザと共有化することにより、遠隔地でネットワークを介した打ち合わせができる。図7.5-6にチョークボードのウインドウと文字やグラフィックスのデータをチョークボードに書き込むためのパレットを示す。それぞれのチョークボードは、『フォアグラウンド』と『バックグラウンド』の二つの層が重なったもので、ユーザは、バックグラウンドにはスキャナ、ファイル、他のPMウインドウからイメージ・データをロードすることができ、フォアグラウンドに文字やグラフィックスのデータを上書きすることができる。作成されたチョークボード上のデータは、イメージ・データとして保管することができる。またリモート・ポインタをチョークボード上で利用できる。さて、他のPMウインドウのビット・イメージをロードする機能は、『ミラー（Mirror）』と呼ばれており、これによって相手のユーザと任意のPMアプリケーション・プログラムを、たとえ相手がそのアプリケーションを持っていなくても、ウインドウのビット・イメージの形で共有することができる。あるチョークボードからミラーとして一度選択されたウインドウは、そのチョークボードと関連付け

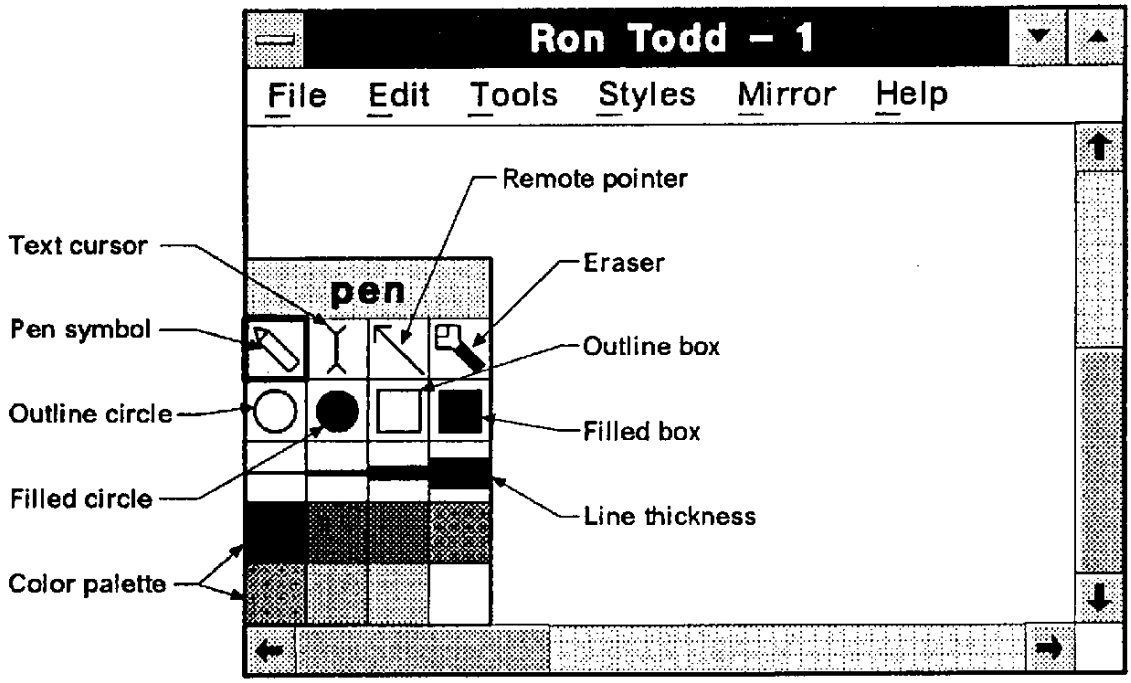


図7.5-6　チョークボード

られ、そのウインドウが更新された時には、ウインドウのタイトル・バーにあるアイコンをクリックするだけで、チョークボードには新しいイメージがロードされる。

(2)　ビデオ・ウインドウ

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

7.5.3　分散開発環境での適用可能性

これまで RTP および P2P という OS／2 上のワークステーション会議システムの事例について解説してきた。これらのシステムは、コンピュータ・システムを通じて人間同士のコミュニケーションを支援する事ができるが、単に電話、ファックス、テレビ電話、OHP、黒板などの機能を電子的に組み

合わせただけでなく、ワークステーションのアプリケーション・プログラムとどのように組み合わせて使えるかという点が重要であろう。本項では、RTPまたはP2Pとアプリケーション・プログラムとの関係について考察し、実際にアプリケーションとしてプログラム開発環境を取り上げた時、その分散開発環境での適用可能性について議論してみたい。もちろんシステム構成としては、現在の二者で一対一の構成ではなく、将来複数ユーザがサポート可能となり、分散している複数の会議参加者（プログラム開発者）が、P2Pのチョークボードのような共有ウインドウを持てるという前提で考えてみたい。

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

しかし、このインタフェースは、ウインドウのビット・イメージという、アプリケーションからみて低いレベルのものなので、より効率的なデータの転送やデータの再利用（取り込んだデータを変更してツールに入力すること）などが要求される場合には、別の方法を考える必要がある。それではつぎに、ファイルのレベルで、共有ウインドウがアプリケーション（プログラム開発用ツール）とインタフェースする場合を考えてみたい。例としてワープロを考えてみると、ユーザは、ワープロで作成した文書ファイルを共有ウインドウに読み込むことによって相手に転送し、その共有ウインドウに表示された文書を相手と同時に見ながら、打ち合わせをすることができることになる（図7.5-7参照）。この場合、文書ファイルのみ転送すればよいので、データの転送時間は短くなるが、共有ウインドウは、ワープロの文書ファイルのフォーマットと、その文書の表示方法を理解している必要があり、共有ウインドウ開発のワークロードは大きくなる。この方式は、特定のツールに特化した会議システムに有効なアプローチと考えられるが、共有ウインドウをそのツールに対応させる方法と、それともツールそのものを協同作業用に書き換える方法のどちらを取るかを、まず議論するべきであろう。

今までは、会議参加者であるプログラム開発者が、P2Pのチョークボードのような共有ウインドウを通じて打ち合わせをするという前提であったが、プログラム開発用ツールとしてのアプリケーションそのものを、どうしたら協同作業用に変更できるか、という点について考えてみたい。既存のアプリケーションを変更することなく、そのまま協同作業に使えるようにするためのメカニズム開発が何

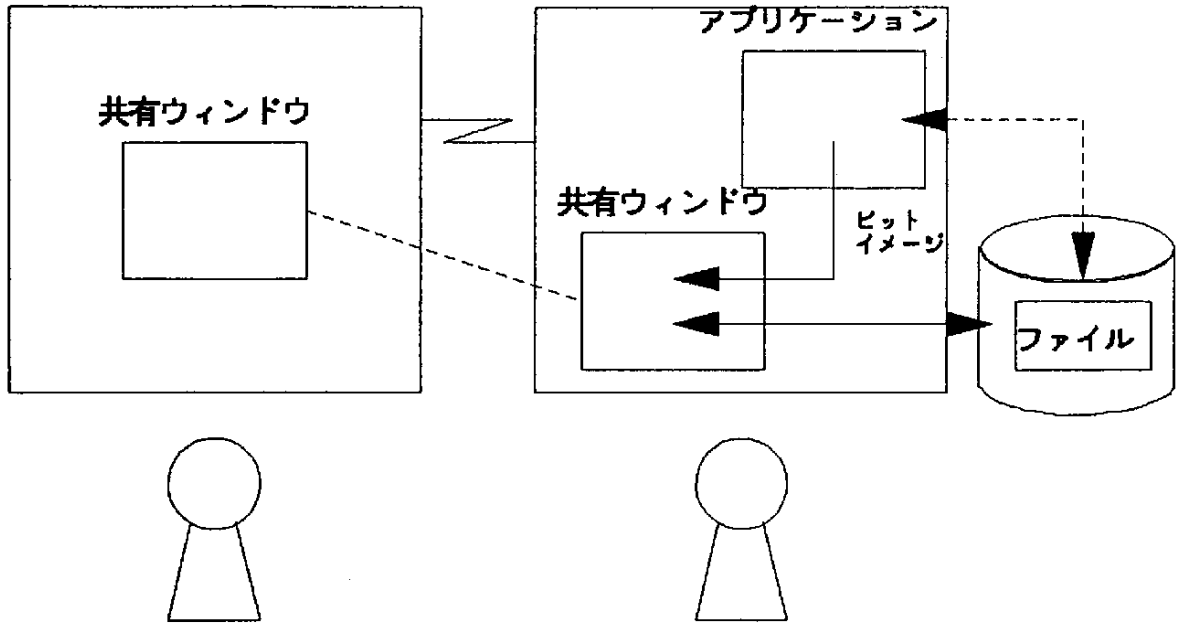


図7.5-7　共有ウインドウとアプリケーションとのインタフェース

度か行われてきたが、OS そのものに協同作業用の仕組み（ウインドウ、ポインタ、ファイルなど必要なシステム資源をネットワークを通してマルチユーザ環境で共有できること）が組み込まれていない場合、サポートできるアプリケーションの範囲、システムのバージョン・アップへの対応、ローカルとリモートの環境の相違への対応、ローカルとリモートでのオペレーションの同期など様々な問題を解決しなければならないことがわかった。

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

7.6 マルチメディア分散グループレビューシステム

7.6.1 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。

7.6.2 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。

7.6.3 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7.6-1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7.6-1 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7.6-1に表示例を示す。

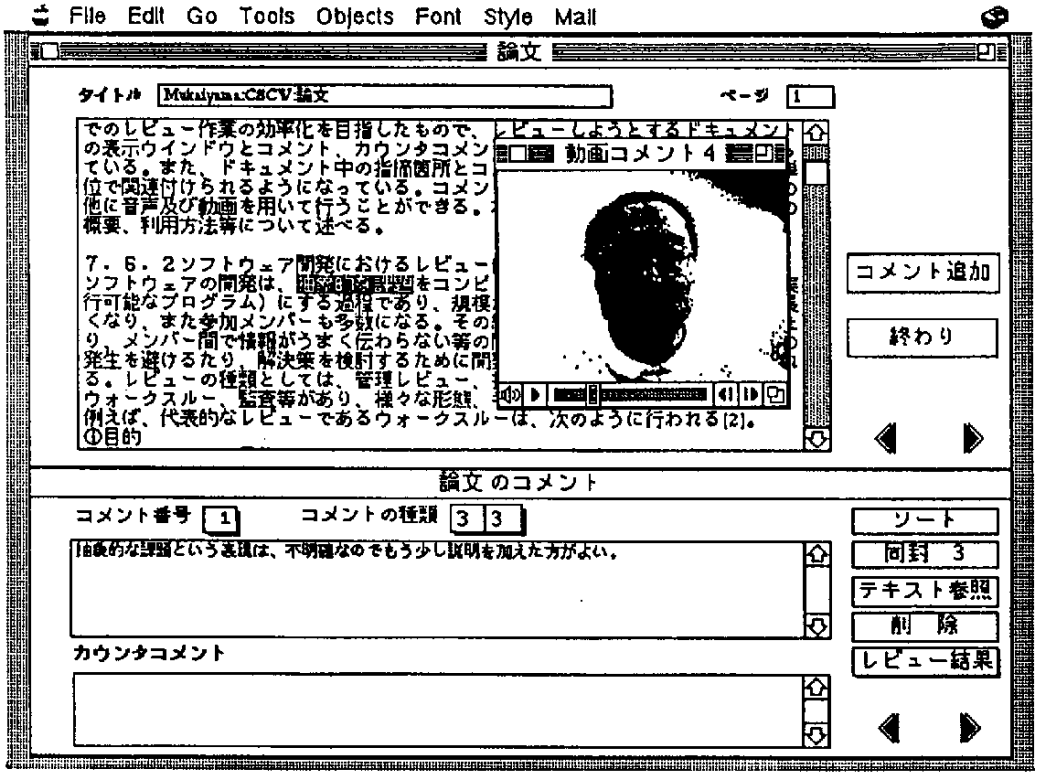


図7.6-1 画面表示例

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7.6-1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7.6-2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。

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。

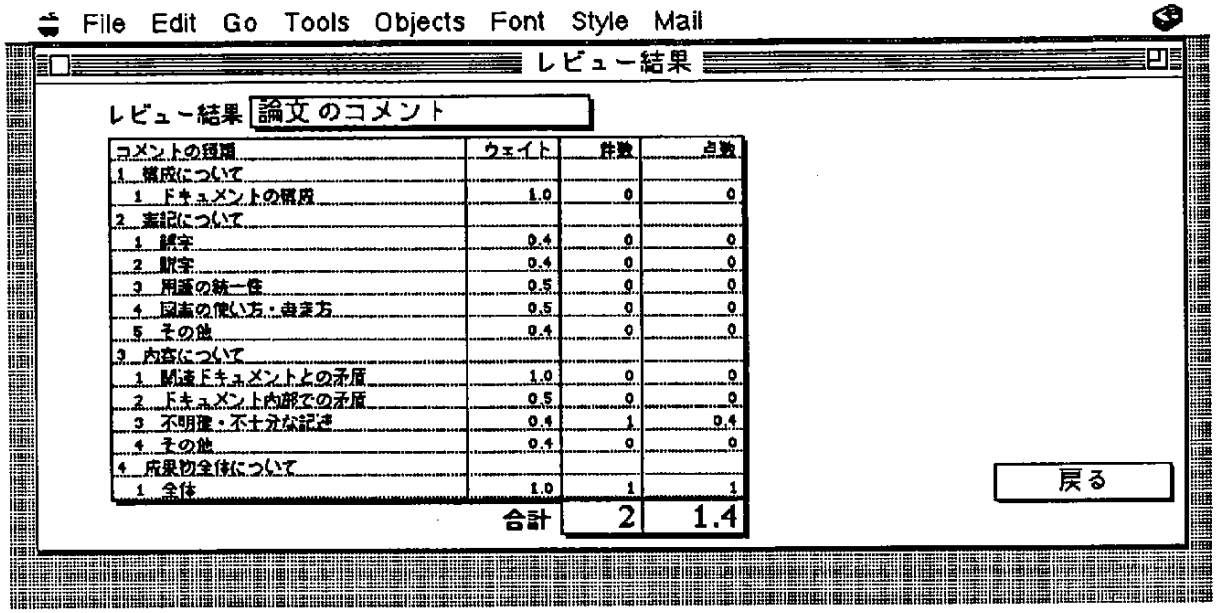


図7.6-2　レビュー結果

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。

7.6.4　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7.6-3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。

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。

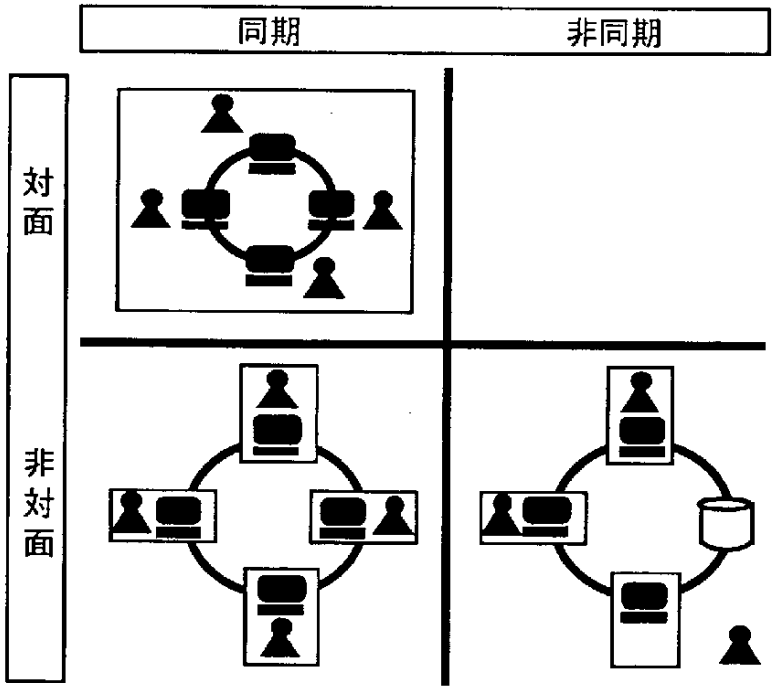


図7.6-3　利用形態

(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。

7.6.5 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).

7.7 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。

7.7.1 CASEプラットフォーム

図7.7-1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7.8-1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。

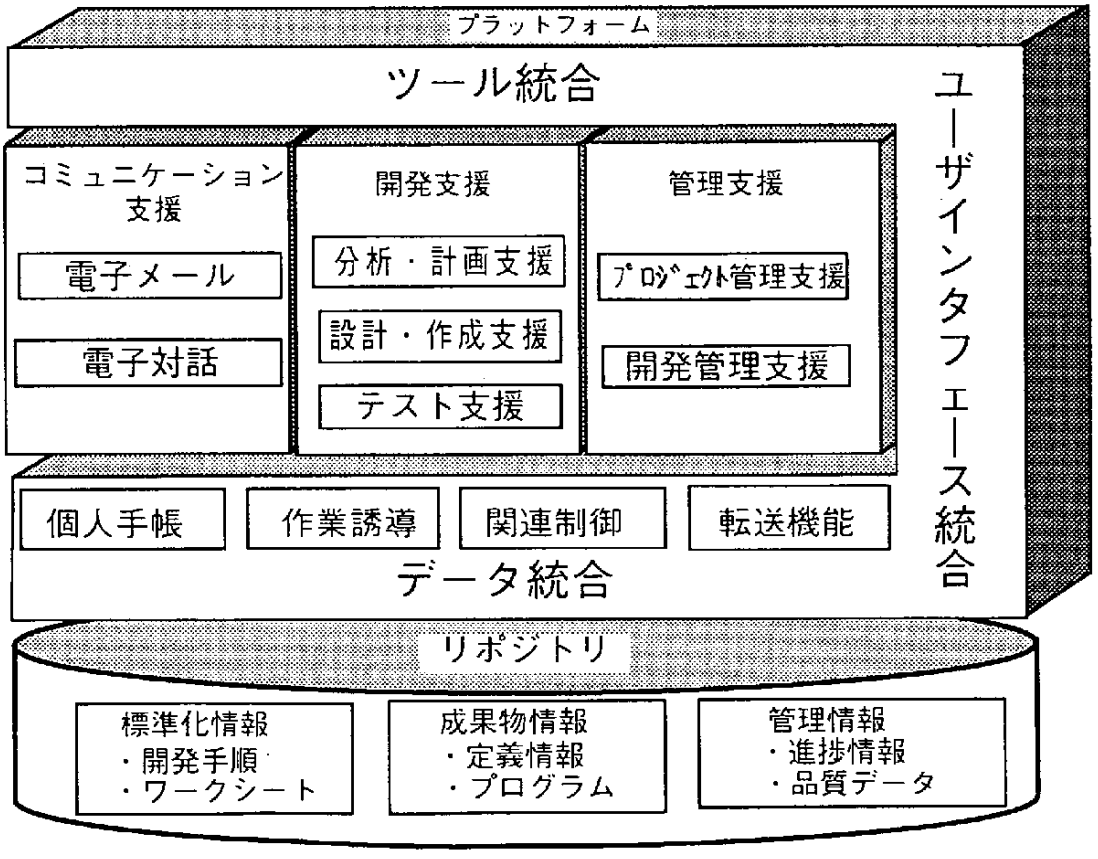


図7.7-1 統合型CASEの構成

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]。

7.7.2 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7.7-2で説明する。開発者はログイン後個人

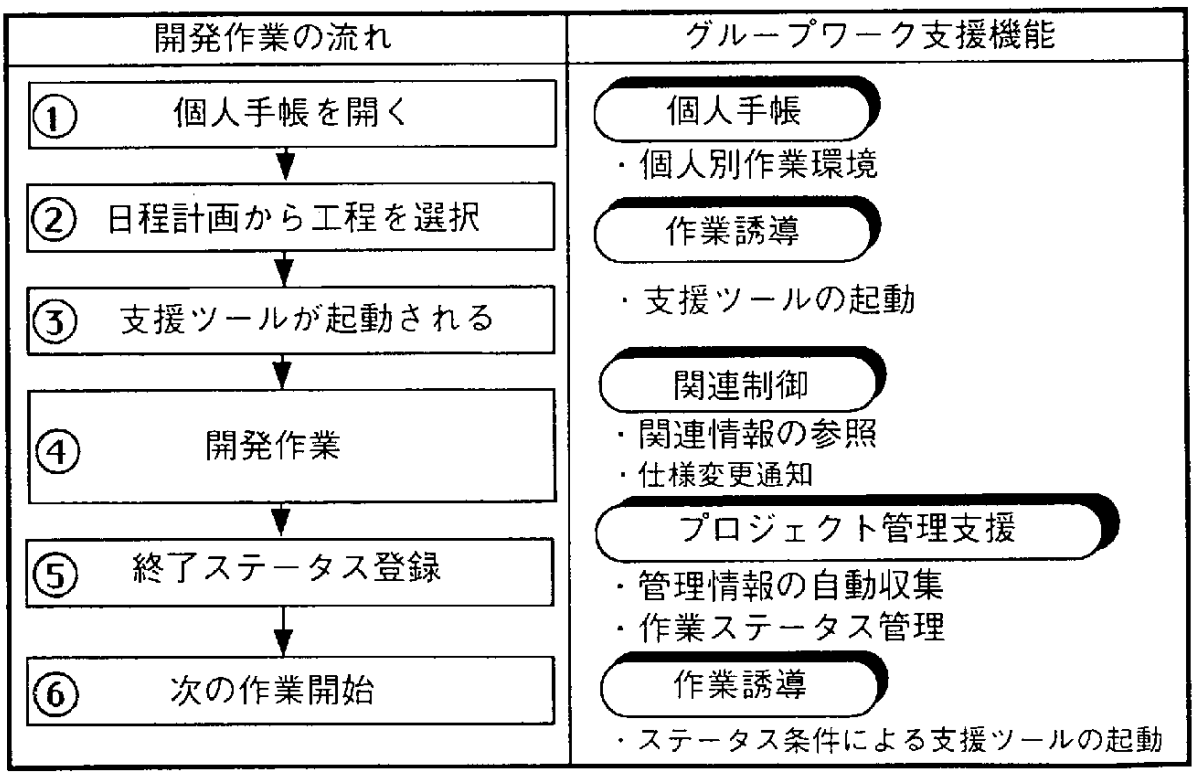


図7.7-2 開発作業の流れ

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。

7.7.3 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7.7-3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。

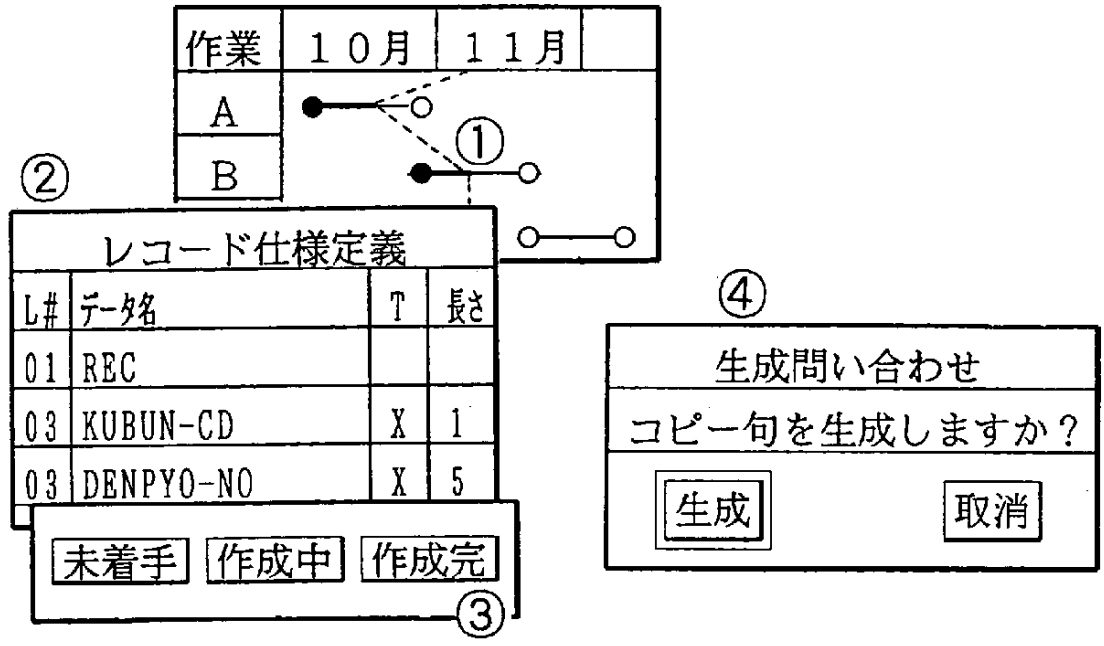


図7.7-3　作業誘導

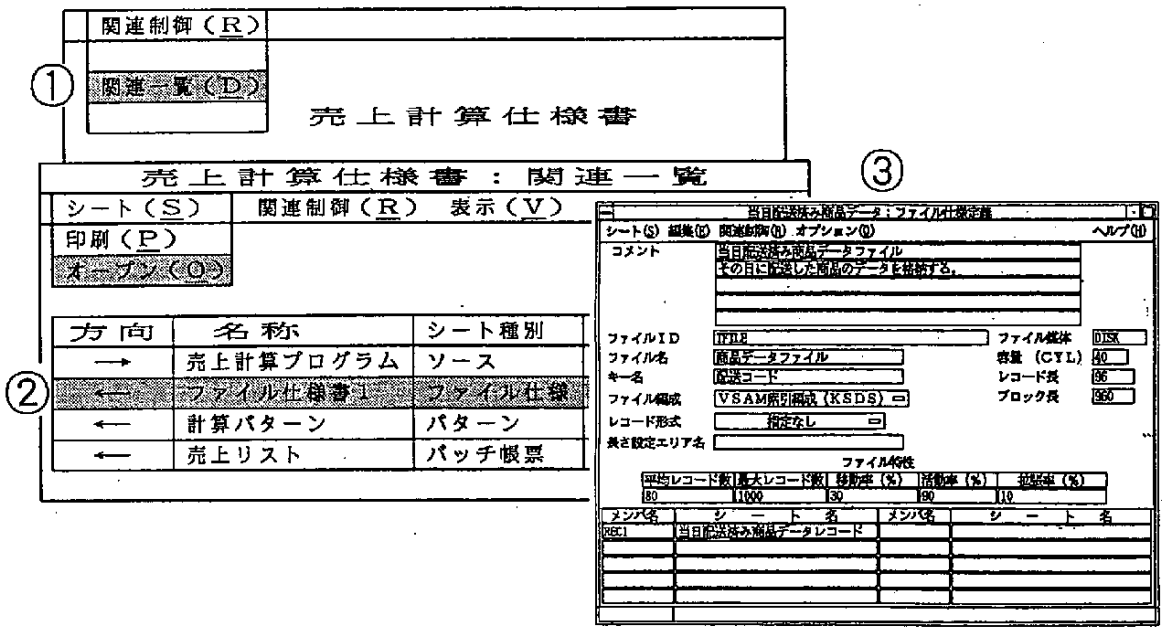


図7.7-4　関連情報の参照

7.7.4 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7.7-4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7.7-5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).

[未宗92] 未宗英利他: 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).

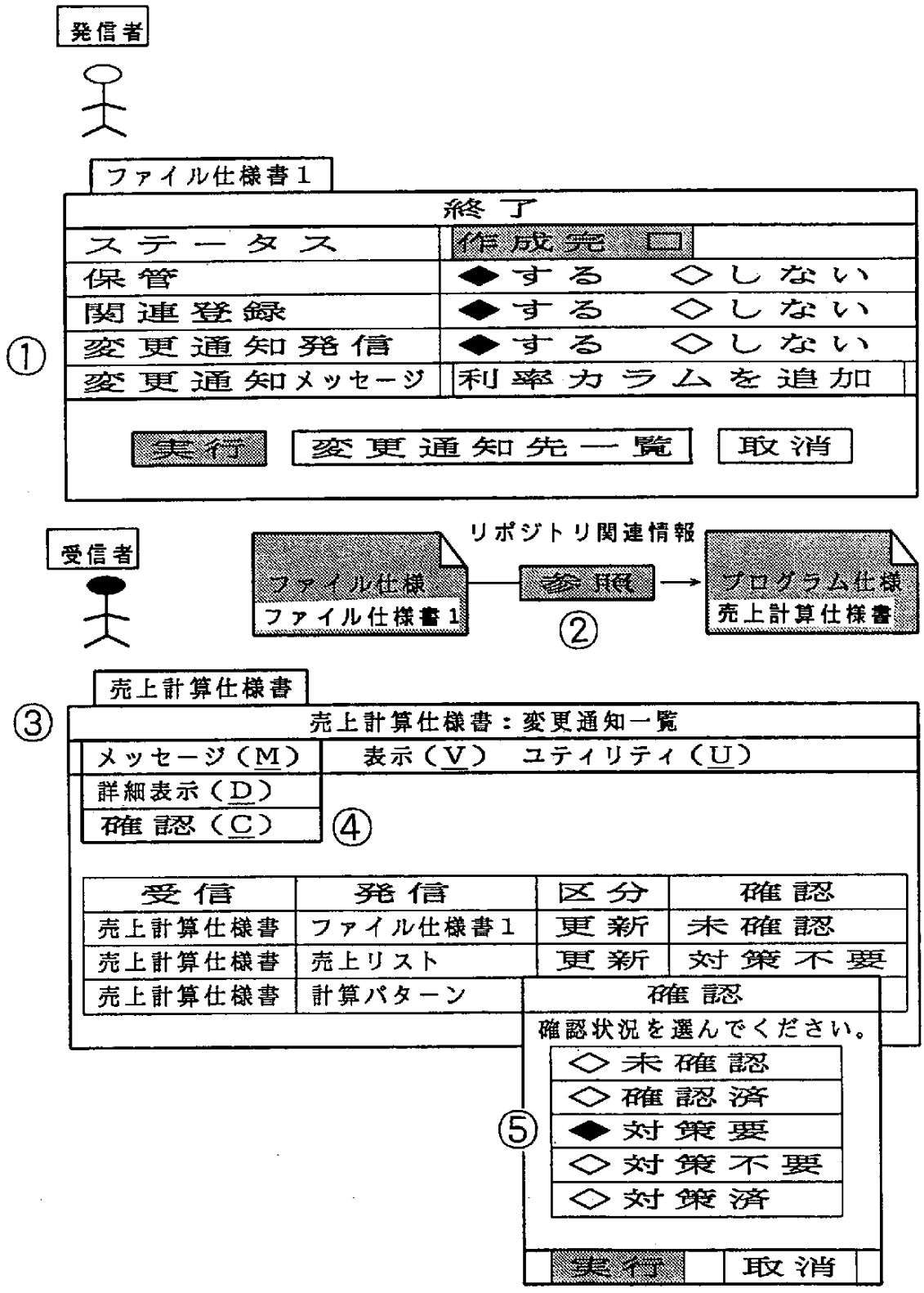

発信者
ファイル仕様書1
終了
ステータス
作成完
保管
する
しない
関連登録
する
しない
①
変更通知発信
する
しない
変更通知メッセージ
利率カラムを追加
実行
変更通知先一覧
取消
受信者
リポジトリ関連情報
ファイル仕様
ファイル仕様書1
参照
②
プログラム仕様
売上計算仕様書
③
売上計算仕様書
売上計算仕様書：変更通知一覧
メッセージ（M）
表示（V）
ユティリティ（U）
詳細表示（D）
確認（C）
④
受信
発信
区分
確認
売上計算仕様書
ファイル仕様書1
更新
未確認
売上計算仕様書
売上リスト
更新
対策不要
売上計算仕様書
計算パターン
確認
確認状況を選んでください。
未確認
確認済
⑤
対策要
対策不要
対策済
実行
取消

図7.7-5　変更通知

7.8 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]。

7.8.1 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7.8-1）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7.8-2）。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。

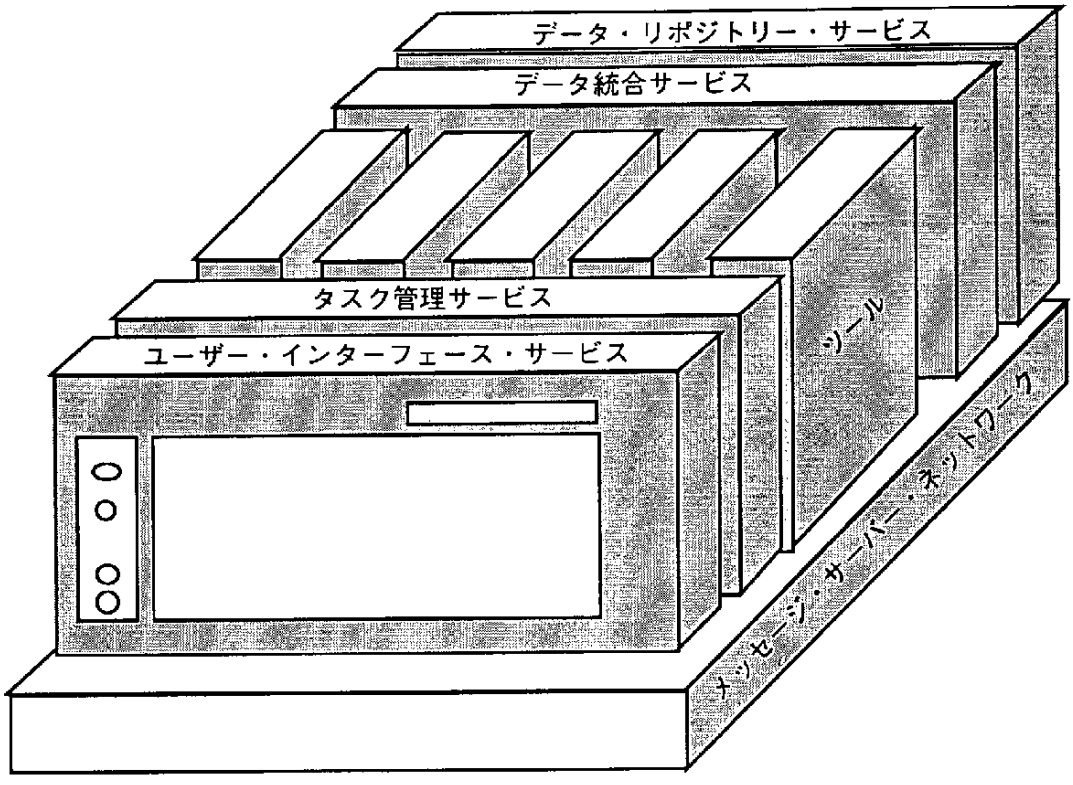


図7.8-1　ECMA リファレンス・モデル

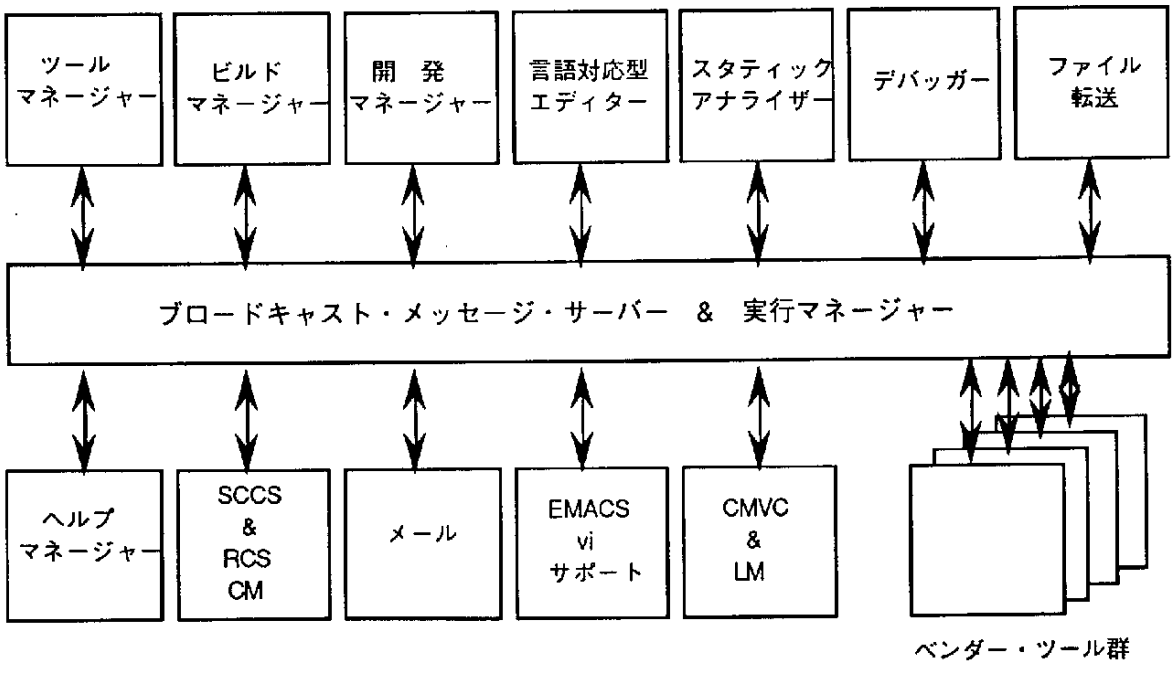


図7.8-2　SDEワークベンチ/6000 の構成要素

7.8.2　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7.8-3）。

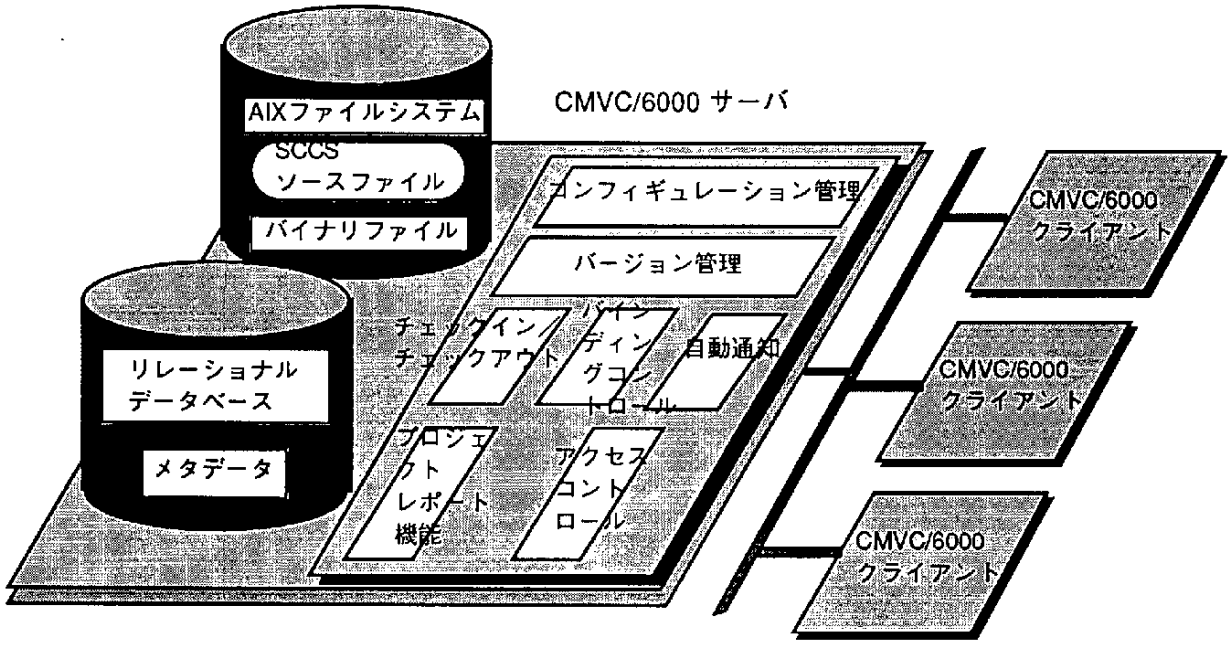


図7.8-3　CMVC/6000 概念図

*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).

8．グループウェアに関する調査

8．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
- ワークステーション／ビデオ会議
- グループ意思決定支援システム
- オフィスプロシージャシステム
- メッセージ構造化システム
- グループエディタ
- 設計議論システム
- 協調作業システムの設計支援
- ツールキット
- 協調支援環境構築のためのアーキテクチャ

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と、グループウェアの定義や分類、概念、具体例等を示してその全体像を概観する論文を紹介する。

② ワークステーション／ビデオ会議

ここでは、リアルタイムの対面型／分散型の会議を支援するためのシステムが対象となる。ワークステーションやビデオを利用した会議システムの開発例やシステム構築の際の設計概念等を示す論文を紹介する。ここで紹介するシステムは、対面型のシステムとしてColab、分散型のシステムとしてMMconf、Rapport、Commune、CAVECAT 、CRUISER および VideoWhiteboardの 7 点である。

③ 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。GDSSの全体像を概観する論文と、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。

④ オフィスプロシージャシステム

ここでは、オフィスにおける各種事務処理の流れを構造化し、複数の担当者とコンピュータが協調的に作業を進めるシステムが対象となる。電子メールのメカニズムとオフィスのプロセス記述を用いて、処理を半自動的に進めていくシステムに関する論文を紹介する。ここで紹介するシステムは、ECF、DOMINO、Action Workflow および EuroCoOp である。

⑤ メッセージ構造化システム

ここでは、情報のフィルタリングに関するシステムが対象となる。電子メールで送受信されるメッセージを半構造化することにより、メッセージの分類や優先度付けを自動的に行えるシステムである Information Lens と、この概念を拡張して多様なオブジェクトの記述と操作が行えるようにした Object Lens 、さらにこれを改良して、エンド・ユーザが広範囲の異なったアプリケーションを開発できる柔軟なシステムであるOval に関する論文を紹介する。

⑥ グループエディタ

ここでは、文書の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、Quilt、CoAUTHOR、DistEdit、PREP および PREPに組み込まれたテキスト差分検出システムであるFlexible Diff の 5 点である。

なお、DistEdit はツールキットとしても捉えることができる。

⑦ 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に基づいて、グループによるソフトウェア設計の上流工程における討論を支援するハイパーテキストである gIBIS と SIBYL という二つのシステムに関する論文を紹介する。また、IBIS 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と、グループによる「打合せ」を構造化した論文も併せて紹介する。

⑧ 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。

⑨ ツールキット

ここでは、グループウェアの個々の機能要素を部品として、これらの部品の組み合わせとカスタマイズによって、状況に合わせたグループウェア・アプリケーションを自由に構成できるシステムが対象となる。リアルタイムの８種類のグループ・ツールを提供する LIZA と、会話と行動の管理のための支援システム構築用部品群からなるツールキットである Strudel に関する論文を紹介する。

⑩ 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。

なお、取り上げた論文の出典は次のとおりである。

- Communications of ACM,Vol.30,No.1, January 1987
- ACM Transactions on Office Information Systems, Vol.5, No.2, April 1987
- ACM Computing Surveys, Vol.20, No.2, June 1988
- CSCW '88 Proceedings, September 1988
- CHI '89 Proceedings, May 1989
- Proceedings of the 11th. International Conference on Software Engineering, 1989
- ACM SIGOIS Bulletin, Vol.11, No.2, April 1990
- CSCW '90 Proceedings, October 1990
- Communications of ACM, Vol.34, No.1, January 1991
- CHI '91 Proceedings, April 1991
- Proceedings of the 13th. International Conference on Software Engineering, May 1991
- ECSCW '91 Proceedings, September 1991
- Proceedings of the 24th. Annual Hawaii International Conference on System Sciences, 1991
- International Journal of Man Machine Studies, 1991

- Studies in Computer Supported Cooperative Work, 1991
- MIT Technology Paper, January 1992
- ACM SIGOIS Bulletin, Vol.13, No.1, April 1992
- IEEE Software, May 1992
- GMD-FIT News (AC Special), No.10, June 1992
- CSCW '92 Proceedings, November 1992

8.1 グループウェア全般

8.1.1 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。

8.1.2 グループウェア

原文タイトル： Groupware: Some issues and experiences

翻訳タイトル： グループウェア：問題と体験

著　者： Clarence Ellis
(Software Technology Program, MCC, Austin, Texas, USA)
Simon Gibbs
(Centre Universitaire d'Informatique, University of Geneva, Switzerland)
Gail Rein
(Software Technology Program, MCC, Austin, Texas, USA)

出　典： Communications of ACM, Vol.34, No.1, January 1991, pp.38-58

本稿では、広義のグループウェアについて研究するとともにグループウェアの開発者が直面している設計上のいくつかの問題について概説されている。本論文は次のような五つの部分に分かれている。

まず「概観」ではグループの共通の仕事と共有環境の必要性の見地からグループウェアを定義している。ここでのグループウェアの定義は多くのシステムに及んでいるため、次の部分では「グループウェアシステムの分類」について述べられる。さらにこれらのシステムを構築する者の「視点」について述べられる。4番目の「概念と例」では一般的なグループウェアの概念がいくつか紹介され、グループウェアの一例として新しいグループエディタとして開発されたGROVEにこれらの概念を適用した結果が示される。5番目にグループウェアの設計者や開発者が直面している設計上の問題点についての議論が示される。ここではリアルタイムのグループウェアにおけるシステムレベルでの問題に焦点が当てられている。

(1) 概　観

グループのインタラクションの支援方法を考えるに当たって留意しなければならない点は、通信、協同作業、調整作業の三つである。

① 通信、協同作業および調整作業の重要性

電子メールのようなコンピュータベースの通信は、他の形式のコミュニケーションを完全には統合化していない。本来、電子メールや電子掲示板のような非同期のテキストベースの世界は、電話や対面の会話といったリアルタイムの世界とは区別される。電気通信とコンピュータ処理技術の統合はこれらのギャップの橋渡しになる。

通信と同様に、協同作業はグループ活動の基盤である。効果的な協同作業には、人が情報を共有することが必要となる。今の情報システムは、ユーザをお互いに大きく引き離してしまっている。CAD

データベースシステムで作業する二人の設計者を例に考えて見ると、彼らは同時に同じオブジェクトの異なる部分を修正することや、相手の変更について知ることはほとんどできない。グループの最新の状況を把握し、必要な時には各ユーザの行動をはっきりと通知するような共有環境が必要である。

通信と協同作業の効果はグループの活動がよく調整されていればより強化される。例えば、調整がなければプログラマやライタのチームは対立したり、何度も同じ作業を行っているような状況になるであろう。調整は、仕事そのものとも、また何人かの仲間が仕事を遂行するときに必要なオーバーヘッドであるとも見なせる。

② グループウェアの定義

グループウェアの目標は、グループ活動の通信、協同作業、調整作業を援助することである。したがって、ここではグループウェアを

「共通の仕事（または目標）に携わる人のグループを支援し、共有環境のインタフェースを提供するコンピュータベースのシステム」

と定義する。共通の仕事と共有環境の観念はこの定義にとって重要である。この定義は、ユーザが共通の仕事を共有できないタイムシェアリングシステムのようなマルチユーザシステムを除外する。定義では、ユーザが同時に活動するとは言っていないことに注意されたい。

グループウェアという用語は、最初にJohnson-Lentzによって「コンピュータシステムに社会的なグループのプロセスを付加したもの」と定義された。Johansenはその定義をコンピュータベースのシステムに限定した。本稿は基本的にシステムレベルの問題に関係しているので、Johansenの論に従っている。

(2) グループウェアシステムの分類

ここでは、グループウェアの様々な種類を見分けるために有用な、二種類の分類を取り上げる。最初の分類は、時間・空間に基づく分類である。2番目はアプリケーションレベルの機能的なものによる分類である。

① 時間・空間的分類

グループウェアは、対面のグループまたは多くの場所に分散しているグループを手助けすると考えることができる。さらにグループウェアは、リアルタイムのインタラクションや非同期（非リアルタイム）のインタラクションにおいて通信や協調を強化するものと考えられる。これらの時間・空間的な考慮事項は、図8.1-1に示されるような2×2のマトリクスによって表現された四つのカテゴリになる。会議室技術は左上、リアルタイムの文書エディタは左下、物理的な掲示板は右上、電子メールは右下のマスに入る。

包括的なグループウェアシステムは、四つの全てのカテゴリの要求に応えられる必要がある。例えば、(a) グループ内で、リアルタイムで文書を編集するのにコンピュータを使用する（同時／同じ場所、

または同時／異なる場所）という場合と、(b) 自分の家かオフィスで一人で編集する（異なる時間）という場合とで、基本的機能やインタフェースの Look and Feel が共通であることはたいへん重宝である。

	同じ時間	異なる時間
同じ場所	対面対話	非同期対話
異なる場所	同期分散対話	非同期分散対話

図8.1-1　グループウェアの時間・空間のマトリクス

② アプリケーションレベルの分類

二番目の分類は、アプリケーションレベルでの機能に基づいており、包括的という意味ではない。また、これらの定義したカテゴリは重複するものも多い。

(a) メッセージシステム

グループウェアの最も親しみ深い例は、コンピュータベースのメッセージ交換システムであり、非リアルタイムのテキストメッセージのユーザグループ間の交換を支援する。電子メールやコンピュータ会議、あるいは電子掲示板システムの例がある。こういったシステムの増大は「情報オーバーロード」現象をもたらしてきた。最近のメッセージシステムには、ユーザが処理する負荷を軽減することによって情報オーバーロードの管理を助けるものがある。「知性」がメッセージ配送システムにつけ加えられる場合もある。例えば、Information Lens システムやImail システム等の例が挙げられる。

(b) マルチユーザエディタ

グループのメンバは文書の協同執筆、編集にマルチユーザエディタが利用できる。ForComment のようなエディタは非リアルタイムに使用され、さまざまな批評のコメントと著者の作成したテキストをうまく分けることができる。リアルタイムのグループエディタは、同時に同じものをグループ内の人が編集できる。エディタは透過的にロックと同期を管理しユーザは個人用の文書であるかのように占有できる。このような例としては、Collaborative Editing System (CES)、Shared Book および Quilt 等がある。

また、マルチユーザエディタには、他のユーザの活動をはっきりと通知できるものもある。Mercury は、プログラムチーム用のエディタであるが、他人がプログラムを変更したことによって自分のコードの変更が必要になるときユーザに通報する。DistEdit システムは、マルチプルグループエディタを作ったりサポートするためのツールキットを用意しようとしている。

(c) グループ意思決定支援システムおよび電子会議室

グループ意思決定支援システム（GDSS）は、体系化されていない問題をグループで検討するためのコンピュータベースのシステムである。目標は意思決定会議の効率化、意思決定会議の高速化、意思決定の結果の質の改善である。GDSSは代替案の順序づけや投票用のツール、発想支援や論点解析のような意思決定の構造化を補助できる。

多くのGDSSは、数台のネットワーク化されたWS、コンピュータに制御された大型共有ディスプレイ、オーディオ／ビデオ装置から構成される電子会議室として構成されている。これらの施設のいくつかは特別に訓練されたオペレータが必要であり、また他のものはグループのメンバの操作能力を前提としている。アリゾナ大学のPlex Center Planning and Decision Support Laboratoryがよく知られた例である。

(d) コンピュータ会議

コンピュータは、さまざまな方法で通信の媒介として役に立つ。特に、会議進行の方法に三つの新しいアプローチ、すなわちリアルタイムコンピュータ会議、コンピュータ遠隔会議、デスクトップ会議を与える。

リアルタイムコンピュータ会議は、電子会議室に参集するか物理的に分散しているユーザのグループが、WSまたは端末を通じて同時にインタラクトできる。グループが物理的に分散しているとき、コンファレンスコールといった音声リンクがしばしば設けられる。リアルタイムコンピュータ会議のソフトウェアをインプリメントするためのアプローチとしては、単一ユーザ用のアプリケーションを修正せずにアプリケーションの出力を各参加者のディスプレイに表示できるように組み込む方法と、複数のユーザの存在を考慮した特別のアプリケーションを設計する方法の2種類がある。前者の例としては端末接続や複製ウィンドウがあり、後者の例としては、会議スケジュールシステムのReal Time Calendar（RTCAL）や、リアルタイムグループ注記システムのCognoterなどがある。

コンピュータ遠隔会議は、グループの会議に対する電気通信による支援である。例としては、電話会議やビデオ会議がある。遠隔会議は特別な部屋や訓練されたオペレータが時々必要となるといった不便な点がある。最近のシステムはWSベースの会議用のインタフェースがあり、より利用しやすくなっている。例えば、ゼロックス社はポートランドとパロアルトに分割されたプロジェクトチームで使うための音声／画像システムを設置した。同様のシステムの一つであるCRUISERは、ビデオ回線をブラウズすることで電子的に廊下を歩き回ることができる。

デスクトップ会議は、遠隔会議とリアルタイム会議の欠点を減らし長所をまとめたものである。この方法はやはり会議用インタフェースにWSを使用しているが、参加者によって共有されるアプリケーションを実行することができる。現在のデスクトップ会議システムは、WSごとにマルチビデオウィンドウが出来るようになっている。これは、情報の動画像や参加者のビデオイメージが表示できる。

例としては、MMConfシステムやRapportマルチメディア会議システムが挙げられる。

(e) インテリジェント・エージェント

インテリジェントエージェントは、ある特定の仕事の型に対応し、ユーザインタフェースの動きを人間のユーザに似せたものである。代表例として、Lizaという名のエージェントを持つグループウェアツールキットがある。

(f) 調整システム

調整システムは、全体の目標といった状況の下で関連する他人の行動を見ることができると同時に、各個人は自分の行動を見ることができる。システムはたいてい、ユーザの行動や待ちの状態の現状を知らせたり、自動的に助言や警報でユーザの行動のきっかけをつくる。調整システムは、形式（フォーム）、手順、会話、通信構造の四つの内どれに基づくかにより、四つのモデルの型に分類できる。フォームに基づいたモデルの例としては電子循環フォルダ（ECF）があり、会話に基づいたモデルの例としてはThe Coordinatorがある。

(3) 視点

グループウェア成功のためには、少なくとも五つのキーとなる分野または視点がある。それは、(i) 分散システム、(ii) 通信、(iii) 人間－コンピュータ間のインタラクション、(iv) 人工知能（AI）、(v) 社会理論である。グループウェアとこれら五つの研究領域の関係は相互に有益なものである。各分野はグループウェアの理論や実践における理解を進めるだけでなく、グループウェアは五つの領域全部に対して研究のやりがいのある分野あるいはトピックスを提供する。こうしたトピックスはグループウェアがなければ研究されないものである。

同じく重要なことは、与えられたグループウェアシステムはたいていこれらの分野の二つ以上の視点が関係しているという点である。例えば、デスクトップ会議のパラダイムは、(i) 通信技術に始まりコンピュータの能力および受信機の表示デバイスでこれを強化するという方法と、(ii) パーソナルWS（分散システムの視点）に始まり通信機能と統合するという方法の、二つの方向から導き出されたことがわかる。

(4) リアルタイムグループウェアの概念と例

① グループウェアの用語

グループウェアにおいて具体化された用語と考えは進化している。ここでは、グループウェアのシステムの研究と比較に役に立つ重要な用語を説明する。

(a) 共有状況

共有状況とは一組のオブジェクトのことであり、オブジェクトおよびそのオブジェクトの上になさ

れる行動が一組のユーザによって見ることができるオブジェクトの集合である。例として、協同執筆システムにおける文書オブジェクトや電子教室におけるクラスノートがある。

(b) グループウインドウ

グループウインドウとは、そのインスタンスが異なるディスプレイに表示されているウィンドウの集合である。各インスタンスは接続される。例えば、あるインスタンスに円を描くと他のインスタンスに円ができ、また、あるインスタンスでスクロールすれば他のインスタンスもスクロールする。

(c) テレポインタ

テレポインタは、二つ以上のディスプレイに現れるカーソルであり、異なるユーザが動かすことができる。あるディスプレイで動かされれば、全てのディスプレイで動く。

(d) ビュー

ビューは、共有状況のいくつかの部分のビジュアルまたはマルチメディアによる表現である。異なるビューには、表現は違っていても（例えば、数列は表やグラフで表現できる）同じ情報を含めることができる、また、共有状況の中で参照される異った部分に対して、同じ表現を使用できる。

(e) リアルタイムおよび非リアルタイムのインタラクション

リアルタイムのインタラクションにおいては会話のようにリアルタイムで会話できる。非リアルタイムのインタラクションは郵便物による文通のように長い時間をかけて行われる。

(f) セッション

セッションとは、グループウェアシステムによって支援されたリアルタイムのインタラクションのある期間のことである。例として公式のミーティングや非公式のワークグループの討議がある。

(g) 役割 (role)

役割は、人またはシステムモジュールに対して与えられた特権および義務である。役割は、公式または非公式に持たせることができる。例えば、多くの人を訪問して話したいと思い立った人は、自然に（非公式に）話題の提供者の役割が与えられるであろう。また、グループのリーダは、公式にマネージャの役割が与えられる。

② GROVE：グループウェアの一例

(a) GROVEの概要

The GRoup Outline Viewing Editor（GROVE）は、ここで紹介した概念を実現したグループウェアの一例である。MCCでインプリメントされたGROVEは、ワークセッション中にグループの人がアウトライン（概要）を同時に編集できるように設計された簡単なテキストエディタである。

GROVEのセッションでは、各ユーザは各人の WS とビットマップディスプレイを持っている。各ユーザは、スクリーン上に何重にも重ねられたウインドウの上で複数のテキストのビューを見たり操作できる。GROVEは、ビューとビュアとを区別している。ビューとは読み込みアクセス権によって限定

されたアウトラインにおける事項の部分集合である。ビュア(viewer)とは連続したビューの部分集合を見るためのグループウインドウである。GROVEのビューとビュアは、プライベート、共有、公開に分類される。プライベートビューとは、ある特定のユーザのみが読むことができる事項である。共有ビューとは、列挙されたユーザのみが読むことのできる事項である。公開ビューとは、全てのユーザが読むことができる事項である。

GROVEのグループウィンドウのイメージを図8.1-2 に示す。グループウィンドウはユーザの同期インタラクションのためのビュアを共有できる。

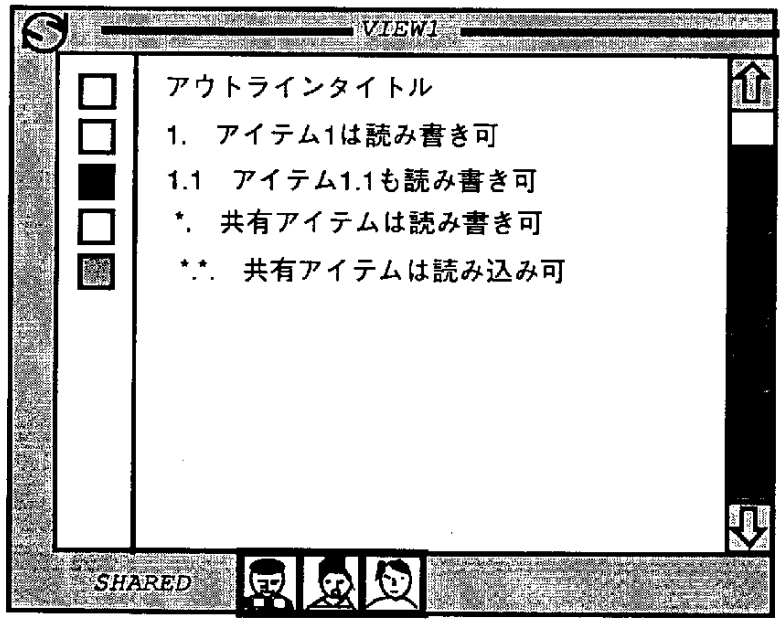


図8.1-2 GROVEのグループウィンドウのイメージ

グループウィンドウは、ビューの表示だけでなく、誰がウィンドウを使用でき誰が与えられた時間のセッションに参加しているかを示す。この情報はビューのメンバの人のイメージ写真等（イメージが使用できない場合は単に氏名）をウィンドウの下のボーダに沿って表示できる。ユーザがセッションに出入りすると彼らの写真がそのとき可能なグループウィンドウに表示されたり消されたりする。図8.1-2 のウィンドウは下のボーダ沿いに示されている三人のユーザの WS に表示され、各ユーザは他の人とセッションがつながっていることがわかる。ユーザは標準の編集操作（挿入、削除、カット、ペースト等）を行ってグループウィンドウ中の編集中のアウトラインを修正できる。これが実行されると、三人全員ともすぐにその修正を見ることができる。スクリーンで灰色をしたアウトラインの項目（図8.1-2 の一番下の項目のように）はそのユーザによって修正することはできない（修正は黒い色の項目に限られる）。ユーザは（マウスクリックで）アウトラインをオープンしたりクローズしたり

でき、また読み書き許可を変更できる。

関係者はGROVEのセッションにいつでも参加／退出できる。ユーザがセッションに入った（または再び入った）とき、以前に保存したバージョンの再生を選択しなければ、最新の文書を受け取ることができる。たとえセッションに参加していない間に変更が行われたとしても最新の状態は保持される。セッションは参加者が誰もいなくなったとき終了する。

(b) GROVEの使用体験

GROVEは、数グループの協同論文の計画からブレーンストーミングの提示までさまざまな設計の活動に使われてきた。一般にセッションは三つの型に分けられる。

① 対面セッション： 三台のSun WSと電子黒板のある研究所の電子会議室におけるセッション

② 分散セッション： 関係者が自分のオフィスでマシンを使って仕事をし、音声通信用のスピーカフォンで会議電話を行うセッション

③ 混合形式セッション：対面の参加者もいれば、分散した参加者もいるセッション

表8.1-1 は、15 のグループセッションにおけるセッションの型、グループ規模と作業内容を表にしたものである。分散および混合形式のセッションにはしばしば5、6人の参加者がいる。

表8.1-1 GROVEを利用したセッション

セッション形式	人数	作業内容
分散	3	プロジェクト記述の問題の確認
対面	3	プロジェクト記述の問題リストの精選
対面	3	技術報告のアウトライン
分散	3	管理上の発表計画
対面	3	管理上の発表計画（続き）
対面	2	チュートリアルの計画
対面	3	プロジェクト計画の討議
対面	3	システムのソフトウェア拡張の討議
対面	3	プロジェクト計画の討議（続き）
対面	3	プロジェクト計画の討議（続き）
混合形式	5	二つの計画の類似点/相違点の確認
分散	3	遠隔セッションのテスト
分散	3	二つの関係したトピックのブレインストーム
分散	5	論文のアウトライン
混合形式	6	論文のアウトライン

ユーザから見れば、分散編集セッションは対面の編集セッションとは明らかに異なった体験である。分散セッションについての観察結果としては次の点があった。

- 情報アクセスを増加させる。
- グループ内の並行作業を促進する。

・ 論議をより困難にする。
・ グループの焦点を集めることをより困難にし、気分を集中させる必要がある。
・ 社交的インタラクションが減少する。

また、グループ編集（リアルタイム）の単一ユーザの編集に対する比較結果とては次の点があった。

・ インタラクションは曖昧で、焦点が定まらず、混沌としている。
・ 他人との衝突は滅多に起こらない。
・ 一人で実施した場合より効率的にできる。
・ 情報の無駄を無くし、効果的なグループの成果に導く。
・ ツールの使用を自然な態度で習得できる。

(5) 設計上の問題

ここでは、小～中規模のグループが使うよう設計されたリアルタイムのグループウェアを対象として、その設計者や開発者が直面している問題について述べる。

① グループインタフェース

グループインタフェースは、グループの行動を表現し、かつ単一のユーザよりむしろ複数のユーザによって制御されることにおいて、単一ユーザのインタフェースとは異なる。グループインタフェースの一例は、GROVEのグループウィンドウである。他の例では、リアルタイムコンピュータ会議用のインタフェースやマルチプレーヤー用のゲームのインタフェースなどがある。

グループインタフェースでは単一ユーザのインタフェースでは出てこなかった設計上の問題が出てくる。基本的な問題は、複雑さの管理法である。複数のユーザは単一のユーザよりも、より活動のレベルが高く、高度の並行性を引き起こす傾向がある。インタフェースはこれらの複雑な作用を支援しなければならない。

他の重要な問題は、単一ユーザインタフェースのどのような技術と概念が、グループインタフェースの構築に役に立つかということである。単一ユーザインタフェースのままであるとどこでうまく行かなくなるかということを把握することにより、新しい概念の必要性が示される。例えば、スクロールバーは一人以上で操作する場合役に立つかとか、単に気を散らせるだけであるかとかである。

また、WYSIWISの問題、グループの焦点と散漫の問題、グループ力学に関連した問題、スクリーンのスペースの管理に関連した問題、グループインタフェースのツールキットに関連する問題等についても考慮する必要がある。

② グループプロセス

プログラムのウォークスルーのように明確に定められた仕事は、ユーザが集まって参加する必要があり、これはグループプロセスといわれる。グループプロセスはシナジー（協力作用）と並行性を増

大させるが、必要となる調整のオーバーヘッドはグループの負荷となり、効果を低下させる。グループウェアの技術は、利益を増大する一方でオーバーヘッドを最小化しようとしている。

考慮すべき問題としては、グループプロトコル、グループオペレーション、組織的・社会的要因、例外と調整、行動支援の統合等の点がある。

③ 同時実行制御

グループウェアシステムは、参加者の同時操作による競合を解決するために同時実行制御を必要としている。GROVEのようなグループエディタの場合、一人が文に単語を挿入する一方で、もう一人がその文を削除することも有り得る。グループウェアには独特の同時実行の一連の問題がある。明確なロックやトランザクション処理のようなデータベースのアプリケーションにおける同時実行処理の多くのアプローチは、グループウェアにとって不適当であり、緊密に結合したチームワークを妨げてしまう。

考慮すべき問題としては、応答性、グループインタフェース、広域分散、データの複製、堅牢性(rubstness)等がある。

また、同時実行制御方法についても考慮すべきである。制御方法のアプローチとしては、単純ロック(Simple Locking)、トランザクションメカニズム、交替(turn taking)プロトコル、集中化コントローラ、独立性検出、可逆実行、オペレーション変換等が考えられる。

④ 他のシステムの問題

グループウェアは、電子メールシステムそのものに近いものから、最新技術のリアルタイムのマルチユーザツールまで幅広いシステムを含んでいる。グループウェアの設計者は共通のインプリメント上の問題に直面する。ここで考慮しなければならない問題点は、通信プロトコル、アクセス制御、（リアルタイムの）通知などである。

8.2 ワークステーション／ビデオ会議

8.2.1 ビューの共有

原文タイトル： Sharing views and interactions with single-user applications

翻訳タイトル： シングル・ユーザ・アプリケーションによるシェアリング・ビューとインタラクション

著　者： Saul Greenberg
(Advanced Technologies, Alberta Research Council, Alberta, Canada)

出　典： ACM SIGOIS Bulletin, Vol.11, No.2, April 1990, pp.227-237

本稿では、元来シングル・ユーザ用に設計されたアプリケーションを、特別な「ビュー・シェアリング・ソフトウェア」を通じて物理的に異なるワークステーション間で共有するというコンセプトの基で、ビュー・シェアリング・ソフトウェアの設計と評価において考慮しなければならない役割と責任（ビュー管理、発言権制御、参加者の会議への参加登録、メタ・レベル通信の処理等）について述べている。また背景として、既存のビュー共有システムに関する概観を述べている。

(1) ビューの共有による協同作業能力の向上

ビューの共有とシングル・ユーザ・アプリケーションとのインタラクションによって、人間の協同作業能力は大きく向上する。可能性としては次のようなものが考えられる。

① 参加者は地理的に分散していられる。十分な帯域幅とボイス・チャネルのネットワークがあれば、物理的に離れた小人数のグループがコンピュータ・ベースの作業をリアルタイムに容易に共有できる。

② 高度なテキストおよびドローイング・ツールがグループで利用できるようになる。紙やホワイトボードのような従来のメディアに比べて、シングル・ユーザ・アプリケーションの共有によって機能が豊富になるばかりでなく、グループがディスプレイ上のオブジェクトを作成、操作、セーブする際の柔軟性が高まる。同様に、（アイデア・アウトライナのような）専門的なアプリケーションによって、参加者は、より作業に適したツールを利用できるようになる。

③ 物理的に互いの邪魔にならないという単純な理由から、ワーク・サーフェスへのアクセスを交代するのが容易になる。

④ 共有ビューは、「肩越し」の相談に有益である。経験者は、新人に問題を指摘させたり、問題解決に「ユーザを導いたり」、新人が自分で行おうとするのを見守ったりすることで、容易に援助を行うことができる。

⑤ 共有ビューは、参加者に資料を示したり、参加者が会議中に資料を参照したり利用したりするためのメディアとして利用できる。

さらに、情報シェアリングや調整活動、協同設計、共同監督、進行記録など、あらゆる協調プロセスが共有ビューの恩恵を受けることができる。

(2) 既存の共有ビューシステムの概観

本論文では、60年代中頃から続いている共有ビューシステムに関わる研究開発の一部について、簡略に述べられている。ここで取り上げられたシステムは、次のとおりである。

- NLS（1968：Engelbart と English）とこれを改良したAugment（1982, 1984：Engelbart）
- RTCALとMBLink（1985：Sarin と Greif）
- Cantata（1987：Chang）
- Shared X（1989：Garfinkel 他）
- Dialogo（1986：Lantz）
- Diamond Multimedia Conferencing System（1989：Crowley と Forsdick）
- Rapport（1988：Ensor 他）
- Xerox PARC（1987：Stefik 他）
- Capture Lab（1988：EDS）
- EMCE（1988：Garcia-Luna-Aceves 他）
- Timbuktu（1988：farallon）
- VideoDraw（1989：Tang）
- Telecollaboration プロジェクト（1989：Corey 他）とCruiser（1988：Root、1989：Fish）
- SharedArk（1988：Smith）

(3) ビュー・シェアリング・ソフトウェアの役割と責任

総合的なワークステーション・ベースのビュー・シェアリング・システムを構築する場合、設計上考慮しなければならない問題が数多く存在する。システムの責任は、次の四つの役割によって果される。

① ビュー・マネージャ

アプリケーションの分散ビューは、WYSIWISの考えに（多少とも）従わなければならず、誰もが自分の画面かウィンドウで同じ画像を見なければならない。ビュー・マネージャは、これらのビューを同期、伝達する責任を負う。

② チェア・マネージャ

全ての参加者が妥当な方法でアプリケーションとインタラクトできなければならない。アプリケーションは一つの入力ストリームを処理するためにのみ構築されているため、一般に、一度に一人だけ

に制御を与える発言権制御メカニズムが望ましい。チェア・マネージャは、発言権制御方針の設定や変更の責任を負い、参加者間の発言交代を調整、実行したり、選択した入力ストリームをアプリケーションに送ったりする。

③ 登録マネージャ

登録マネージャによる会議登録は、会議の開会と閉会、会議中の参加者の入場と退席、アクセス・コントロール、会議状況のフィードバックという四つの問題を取り扱う。

④ メタ・マネージャ

参加者は、アプリケーションに実際に影響を与えることなく、ジェスチャや注釈によってアプリケーションの「周辺」で話し合いができなければならない。メタ・マネージャは、このようなメタ・インタラクションの分離と制御に責任を負う。

各マネージャの役割と関係は、同機種のターミナルベースのコンピューティング環境と異機種のターミナルベースの環境とでは異なってくる。

同機種のターミナルベースの環境では、WYSIWIS なビュー・マネージャを作ることは比較的簡単である。アプリケーションの出力ストリームを分岐して、ストリームのコピーを他参加者のターミナルに送るだけでよい。接続された全参加者が最初のターミナルや画面構成と互換性のあるターミナルを使用していれば、各画面は同じ画像を表示する。

これに対して、異機種で互換性がないターミナルによるコンピューティング環境では、ビュー・マネージャはアプリケーションに「仮想ターミナル」宛てに書かせてから、出力ストリームを翻訳し、異なるターミナル全体に同じ（又は近似の）画像が現れるようにしなければならない。

(4) 障害と問題

共有ビュー・システムに関わる障害や問題点には、次のようなものがある。

① 規格

あらゆるビュー・シェアリング・システムや、ひいてはマルチユーザ・システムで最大の制約となるのは、プレゼンテーション規格がないことである。テキストベース・ターミナルのビュー・シェアリングは、ASCII 規格とターミナル・ドライバの統合データベースがあるためかなり単純だが、現在のワークステーションが採用しているウィンドウ・システムは、一つか二つ注目に値する例外はあるが、多くは独自のものである。オフィス内やオフィス間のワークステーションの多様性を考えると、全ての参加者が同じウィンドウ・システムを使ったりこれに精通したりすることはなさそうである。

② ネットワークとマルチメディア

既存の共有ビュー・システムの多くは、ローカルエリア用や専門化された高帯域幅ネットワーク用に構築され、遠隔地の参加者を一堂に集めるという目的にはそぐわないように思われる。通常の電話

回線では遅すぎるし、特別な高帯域幅接続は利用できないかコストがかかりすぎる。技術が発展して高速回線が安価になるまで、マルチメディアを含むリアルタイムのビュー・シェアリングは、遠隔オフィス間の重要なコミュニケーション・デバイスとはならないだろう。

③　ウィンドウ・シェアリングの技術的問題点

効果的なウィンドウ・シェアリングの設計には、様々な技術的問題がある。例えば、カーソルは普通ウィンドウ・マネージャの考えで動かされるので、一つのカーソルの共有や複数のカーソルの表示は、システムの根幹に大きな変化を与えることになる。また、画面に個人のウィンドウや一つの会議に接続された多重共有ウィンドウ、そして複数の会議が混合して表示されることになれば、システムは所属の同じウィンドウを識別するだけでなく、関連ウィンドウをまとめて管理するためのスキーマを持たなければならない。

④　データ・シェアリング

共有ビューを通じてどのようにデータを共有、操作するかという問題もある。アプリケーションが一人の参加者のデータ空間（すなわちその人のファイル空間内）で動くとすると、その人のファイルが他の会議参加者の故意又は不注意でダメージを受ける危険性がある。さらに、他の参加者はその人のアクセス許可によって制約を受けるが、このため他参加者が会議に参加したり会議へ自分の作業結果を持込んだりすることが防げられる可能性がある。アクセス・コントロールは、アプリケーションがグループの所有するデータ空間内で動く場合でも、問題として残る。

⑤　人間的な問題

効果的なビュー・シェアリング会議については、様々な非技術的問題がある。まず、すべての参加者にその共有アプリケーションを使った経験があるとは限らないことである。エディタを共有する際の問題を考えてほしい。何人かがシステムにいるとして、それぞれが自分のエディタがよくて、他の人のエディタを知らない場合である。自分のエディタが共有用に選ばれなかった参加者は、会議では低レベルのメンバになってしまう。一つのエディタで合意が得られたとしたとしても、カスタマイズされたバージョンに慣れていれば問題が起きる。

二つ目は、共有ビューを個人化する方法がない点である。WYSIWIS は、参加者の役割や知識、能力が大きく異なる場合、制約過剰になってしまう恐れがある。技術者が使うビューは、（例えば）管理職が理解するには細かすぎることがある。このような基本的な制約は、アプリケーションに協同作業認識を持たせることによってしか解決できない。

三つ目は、会議のダイナミクスについて我々はほとんど何もわかっていないことである。対面式の会議における発言権制御は、非常に微妙ではあるが自然なボディ・ランゲージによるところが大きい。会議に招いたり参加したりする際の人間的な慣習も微妙である。これに対して、共有ビュー・システムの明示的な発言権制御機構や登録スキーマは、不自然で制約が多い。

8.2.2 Colab

原文タイトル：	Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings
翻訳タイトル：	黒板を越えて：会議における共同作業・問題解決のコンピュータ支援
著　者：	Mark Stefik Gregg Foster Daniel G.Bobrow Kenneth Kahn Stan Lanning Lucy Suchman (Xerox Palo Alto Research Center,Palo Alto,CA)
出　典：	ACM SIGOIS Bulletin, Vol.11, No.2, April 1990, pp.227-237

本稿では、対面式の会議における協調問題解決のためのコンピュータ支援を研究することを目的としてXerox PARCにおいて開発されたColabと呼ばれる実験的会議室について概観されている。

Colabで利用するために開発された会議ツール（Boardnoter、CognoterおよびArgnoter）、そのために開発したコンピュータの要素技術およびアーキテクチャ、Colab会議から得られた観察結果および今後の研究問題について述べられている。

(1) Colabの概観

Colabでは、コンピュータは対面会議における協同作業を支援する。Colabは、LANで接続されたパーソナルコンピュータを用いた２～６人の小さいワーキンググループ用に設計されている。設計する際には、従来の会議室で馴染んでいる要素が使われた。Colabプロジェクトの主目的は、コンピュータ科学者である本研究者自身の会議をより効果的にすることと同時に、コンピュータツールが会議の進行にどの程度影響を及ぼすかについて広範囲な研究を実施する機会を提供することである。

(2) 協同作業用のツールの要件

会議ツール（会議におけるグループの対話および問題解決を支援するプログラム）の基本的必要条件は、参加者全員に対して共有できるインタフェースを供給することである。このようなマルチユーザ・インタフェースは、会議参加者がコンピュータメディアを通じて、容易かつ即座に互いに対話できることを意図したものである。

ここでは、会議ツールの重要な概念を表現するために、WYSIWYG（What You See Is What You Get）と似た表現で、WYSIWIS（What You See Is What I See）という用語を定義した。これは、参加者全員に共有情報の一致したイメージを提示するということである。会議参加者全員が全く同じものを見て、

他の者が指し示している箇所を見るならば、会議ツールはまさにWYSIWISである。

① 緩和されたWYSIWIS

厳格なWYSIWISは全員のディスプレイに同じイメージを表示することになるが、実際上、これはあまりにも制限となることがわかり、代りにWYSIWISの概念を緩和して使用した。例えば、グループの全員がアクセスできるパブリックの対話型ウィンドウと、アクセスが限定されるプライベートウィンドウ（例えば、電子メールなど）が区別できることは便利である。

また、ポインタ表示を画一化しないことも厳格なWYSIWISの概念に反している。会話中にものを指すことは効果的な方法であるが、参加者全員のカーソルを表示するとかえって注意散漫になりがちである。必要な場合のみポインタを見えるようにすることは、妥協策ではあるが効果的である。

WYSIWISの緩和の別の例として、パブリックウィンドウが各自のスクリーン上で違う場所に表示されることを認めている。この場合パブリックポインタは、ウインドウ内の相対的な位置を判断して表示される。これは、スクリーン上の位置によってものを参照する可能性を犠牲にするが、個人ごとのスクリーンレイアウトを可能にする。

② 並列処理の支援

他のプロセスと同様に、会議は、いくつかのことが同時に実行されることにより効率的になる。Colabツールは同時行動の支援を目指しているので、並列処理を行うためには活動をどのように分解して支援するかという問題が、ツール設計の際の鍵となる。並列処理を行うためには、タスクは、グループの各メンバがある程度独立して実行できるような適切なサイズの操作に分割されなければならない。操作が小さすぎるとあまりにも相互依存し、その干渉によって基本的な並列処理までが妨げられてしまう。例えば、共有テキストを作る場合には、個々のキー入力の操作のレベルまで分解すべきでない。とは言っても、操作が不必要に大きいと、協同で作業することができなくなってしまう。

また共有するオブジェクトを同時に扱えるようにすることも、衝突の原因となる。その際には衝突解決の方法が必要になることもあるが、人為的な強制で解決できる場合もある。衝突発見システムやビジーシグナル機能は、誰かがある項目を既に編集しているとか使用中であるということを利用者にグラフィック的に警告する（例えば、使用中の項目は他の全てのスクリーン上では灰色になる等）。

(3) Colabにおける会議ツール

① Boardnoter

Boardnoterは打ち解けた自由形式のスケッチを重視したインフォーマルな会議のためのものであり、黒板の機能を厳密にまねている。Boardnoterを使って描くためには「チョーク」機能を使い、消すには「黒板消し」機能を、文字を書くためには「タイプライタ」機能を使い、そして、指すには「ポインタ」を使う。Boardnoterで四角を描くには、単に「チョークをつまんで」四つの点をストロークすれば

よい。Boardnoterの新しいバージョンでは、複写、移動、拡大／縮小、ゴムヒモ状の線で囲む、グループ化、推敲（整理）などの機能を追加し、あらかじめ準備されたイメージを使ったり、拡大縮小をすることができるようになるので、通常の黒板よりも優れたものになる。

② Cognoter

Cognoterは、一緒にプレゼンテーションの準備をするためにアイデアを整理するColabツールである。そのアウトプットは、注釈が付いたアイデアのアウトラインとそれに関連した文章である。

プレゼンテーションの計画には、アイデアは何で、どのアイデアを一緒にして、どのアイデアが先にきて、プレゼンテーションの順番はどうすべきか、さらに、どのアイデアを除去するのが正当かをグループが決めることが必要である。

Cognoterは、一つの会議を、各々異なった活動である三つのフェーズ（ブレーンストーミング、整理、および、評価）に分ける。ただしグループは各フェーズを進める際に、そのフェーズに与えられた活動しかできないわけではない。例えば、ブレーンストーミング（第1のフェーズで重視される）は、最後のフェーズにおいても行うことができる。

Cognoterは、多くの点でThinkTankなどのツールとは全く異なるプロセスを支援する。Cognoterが複数の参加者（それが具体化するプロセスは単一利用者にとっても有益であるが）が同時に利用できるよう設計されているという最も明白な違いもあるが、それ以上の特徴として、Cognoterは思考プロセスをより小さく、段階的に効率よく行えるステップに分割するという点がある。Cognoterがアイデアの生成と整理のタスクを分けているのに対して、ThinkTankでは、アイデアはアウトラインでしか整理できない。Cognoterはまた、リンク形成操作によってアイデアの部分的整理が一歩一歩洗練されていくような段階的な整理ができる。アイデアの移動操作やグループ化操作は、わずかのリンクで能率的にアイデアを整理することができる。

ただし、Cognoterではプレゼンテーション計画プロセスの重要な部分について明瞭ではないところがある。例えば、Cognoterは、適切な専門的レベル、論文の目標、アイデアの削除、整理の主張については関与しない。Cognoterを修正してそのような点を明瞭な状態にすることはできるだろうが、これは本ツールの有効範囲外である。

(i) ブレーンストーミング

ブレーンストーミングのフェーズの目的は、プレゼンテーションのための最初のアイデアを生成することにあるので、グループ内の対話での協力作用を促進し、アイデアが出てくることに干渉や禁止をしないことは重要である。したがってCognoterでは、このフェーズにおいては、アイデアは評価や除去はされず、それらを整理することは重視されない。

その代りに、参加者はパブリックウィンドウの空いているところに、アイデアを特徴付けるキャッチワードまたはキャッチフレーズを入力する。参加者は、いつでも同時にアイデア項目を追加したり

文で補足説明できる。項目を移動させることはできるが、一旦書かれた項目は（自分のでさえも）削除できない。文章で補うことは項目の意味を明瞭にし、プレゼンテーションの用語としてはっきりさせるために使われる。一旦入力されれば、どんな参加者にもパブリックに表示され、編集することができる。

(ii) 整理

整理フェーズでは、グループは、ブレーンストーミングフェーズで生成されたアイデアの順序付けを行おうとする。Cognoterを使えば、アイデアの順序は、プレゼンテーション順にアイデアをリンクすることと、サブグループに分割することによって円滑に行うことができる。さらに、整理やリンクする前にお互いに項目を動かすことによって、それらを実際に実行する前に討議できる。

順序は、項目の矢印リンクで視覚的に示される。リンクは、プレゼンテーションの完全な順序を決定するために集団で行われる。いくつかの項目を一つのグループにまとめることができ、グループごとのウィンドウに移動される。このとき、上のレベルのウィンドウにはそのグループの名前にカッコが付いた表示で、他の項目と区別される。グループ化された各項目は別のウィンドウに表示される。

また、グループ間でもリンク付けを行うことができる。カッコ付き項目からの、あるいは、カッコ付き項目へのリンクは、そのグループ全体からの、あるいはグループ全体へのリンクのように扱われる。このようにグループ分けやグループ間のリンク付けの操作によって、少数のリンクでアイデアの全体的な順序を整理することができる。

(iii) 評価

三番目の評価フェーズでは、プレゼンテーションの最終形式を決定する。参加者は、アイデアを再編成するために全体の構造をレビューし、欠けている詳細を記入し、枝葉で関係のないアイデアを除去する。

整理フェーズで行われるリンク操作は、通常、二つのアイデアに対する局所的な考察に基づくので、評価フェーズにおいて、リンク付けされたプレゼンテーションの全体を見ることによって、ユーザが全体の構造を評価し、より広範囲の関係（例えば、バランスがとれているかどうか）を考察することができる。

Cognoterでは、プレゼンテーションの出発点は系統的に確認できる。出発点は、入ってくるリンクがない項目である。そしてCognoterは、アウトラインの準備をし、任意にどのアイデアを整理するかを示すことにより、アイデアを最終的に整理することを助ける。項目の関連図をよく検討することによって、テキストを追加しながら、アウトラインは作られる。

③ Argnoter

Argnoter はプロポーザルの呈示・評価のためのColabツールである。グループの一人以上のメンバが何らかのプロポーザル（例えば、プログラム計画や研究進行計画等）を持っているとき、プロポーザ

ル会議は開かれる。そのとき会議の目標は最も良いプロポーザルを選ぶことになる。白紙状態で始まるCognoter会議に対して、プロポーザルは、会議の前に少なくともある程度は案出されている。Argnoterの参加者は、すでにこれらのプロポーザル作成にエネルギーを注ぎ込んでいるで、会議にはより大きな争議および不一致の可能性がある。したがって、これらの会議では、反対意見の発見、理解、評価が意思決定の本質的部分となる。

プロポーザルを展開する際、通常その提案者は、何が要求され、何が可能であるかを正確に知らずに始める。彼らは、その時点での目標に基づいて提案を展開し、何が可能であるか学びつつ目標を明確にしていく。協同考案作業においては、メンバ間のコミュニケーションにより、目標と提案の間のこのような相互作用や緊張は、最後まで続けられる。このとき、提案目標は必ずしも最初から一致するとはかぎらない。通常は目標を推敲していくことも協同プロセスの一部であり、提案を段階的に考案・選択していくこともまた協同プロセスの一部である。

Argnoterによる会議は三つの明瞭なフェーズ（提案、議論、評価）によって構成される。

(i) 提案

提案フェーズでは、プロポーザルは明確に定められる。各プロポーザルは、短いテキスト、また必要であれば図で記述され、その特徴または機能に応じて名前が付けられる。Argnoterでは、プロポーザルは、プライベートでもパブリックでも可能なプロポーザル「フォーム」と呼ばれる一まとまりのウィンドウ上に作られ、表示される。パブリックのプロポーザルフォームは、WYSIWISである。一方、プライベートフォームは、それをコントロールする参加者のマシンにのみ表示される。プライベートフォームは、作業を共有せずに、各参加者が新しいプロポーザルを見たり作成することができる。このとき、会議で考察中のプロポーザルは、別のウィンドウで見ることができる。新しいプロポーザルは、現存するものを修正したり、複数のものから特徴を結合することによって作られる。新しいプロポーザルは、その親プロポーザルからテキスト、図および判断理由を自動的に継承する。

(ii) 議論

議論フェーズでは、個々のプロポーザルを選択する／しない理由の呈示を行う。これらの理由は、書き留められなければならない。Argnoterの場合には、ただ気に入ったプロポーザルに賛成、残りは反対というのではなく、参加者が全プロポーザルに関して賛否両方の判断理由を書く。したがって、全てについて賛否の判断理由を見て熟考することになるので、参加者は注意深くそれらを考案する時間をとるようになる。

議論フェーズでは比較を目的として、互換性、コスト、開発期間、効率、実現可能性、簡素性、および有用性などのカテゴリで、各プロポーザルを賛成／反対に分類分けできる。コンピュータ支援で、これらのカテゴリに基づいてプロポーザルを比較した補助用の表を自動的に作り出すことができる。

議論段階では、参加者は、判断理由を追加したり、現プロポーザルを修正しても良い。そうするこ

とにより、アイデアを協力して出すことが促進され、プロポーザルや理由付けを協同で行う役に立ち、プロポーザル間で共通した理由をシステマチックに見つけだすことができるようになる。

(iii) 評価

評価フェーズでは、まず個々の議論の背後にある前提を考察する。Argnoterにおける前提は、判断理由に関する判断理由として表現される。例えば、「これは労働コストを無視できる」という判断理由は、「このプロポーザルは安価である」という判断理由に基づいている。Argnoterを使えば、これらのさまざまな判断理由を結び付ける議論の構造を見ることができるようになる。

会議参加者は、判断理由の妥当性についてしばしば意見が合わない。Argnoterでは、明確な「blief set（信念セット）」によってこれらの差異のモデル化を行おうとしている。blief setは一組の判断理由を、有効（信じられる）または無効（信じられない）に分類する。

blief setを明確にすることにより、Argnoterが一種の議論スプレッドシートとして使えるようになり、プロポーザルを特定の信念のセットに関連付けて見たり評価することができるようになる。プロポーザルの表示は、プロポーザルに関する議論を整理し、前提を調べ、特定のblief setに基づいたそれらの議論を表示することによって生成される。複数のblief setを共存させることができ、どんな参加者でもblief setを作る（または特殊化する）ことができる。blief setは、異なる観点（例えば、保守派対自由主義者、開発担当対マーケティング担当）の特性を示すことを意図したものである。

(4) インプリメント

Colabのツールをインプリメントするに際して、以下の点を考慮した。

① データベースを維持するための制御方式

Colabデータベースを維持するためにいくつかの制御方式（集権モデル、集中ロックモデル、協力モデル、依存発見モデル、移動ロックモデル）の実験を行い、この結果、協力モデルによるアプローチを採用した。

協力モデルでは、各マシンはデータベースのコピーを持ち、変更を行う場合には、同期をとるのではなくその変更をブロードキャストすることで行われる。このアプローチは次のような競合状態を本質的に持っている。二人の参加者が同時に同じデータを変更すると、どちらの変更が最初に有効かを決める競合が起こり、各マシンで結果が異なってしまうことがある。

これらの欠点を和らげる要因として次のものがある。第一は、データベースを変更する順序は、ほとんどの場合、行われた順番に無関係であるということである。さらに、Colab参加者はその問題を知っており、言葉による合図を使って彼らの行動を調整する。したがって参加者が同じデータを同時に変更することはまれである。

第二の要因は、分散データベースの完全性を保証するメカニズムとは別に、Colabにおいては、参加

者の活動を調整し、会議が本来持っている作業の分割と合意達成の両方のための社会的な仕組みを支援するメカニズムを提供する必要があることである。そのようなメカニズムの一つとしてビジー信号が挙げられる。この信号は現在使用中のスクリーン項目を灰色に変え、その項目を変更しないように他の参加者に警告する。しかし、誰かがその項目を作業し始める瞬間とビジー信号が他の人に伝達される時間との間に必然的に遅延があるため、次の参加者が矛盾した修正を開始してしまう恐れがある。この場合でも、ビジー信号は二人の参加者が矛盾した作業していることをすぐに発見できることを保証する。

② アーキテクチャ

Colabソフトウェアは、Ethernetで接続されたXerox Lispマシン上に構築されている。ソフトウェアはLoopsで記述されている。LoopsはLispにオブジェクト指向を持たせたものであり、データを持つことができるオブジェクトでプログラムが構成されているという点でSmalltalk-80に似ている。計算結果はオブジェクトとしてメッセージを相互に送る。Loopsはパーマネント・オブジェクトの概念をサポートしている。パーマネント・オブジェクトとは、マシン間でユニークであることを保証された識別子によって指定できるオブジェクトである。これらのパーマネント・オブジェクトのバージョンはいくつかのマシンに同時に存在し得る。アソシエーションは複数のマシン上で同じオブジェクトを示す一組の表現である。個々の表現はアソシエートと呼ばれ同じユニークな識別子を持っている。

Colabでは、マシン、Colabツール、および問題解決のために一緒に働いている参加者の三つを一まとめにして、会話(conversation)という言葉で表現している。新しい参加者が会話に加わったとき、参加者全員は新しい参加者が加わったことが分かり、新参加者は他の参加者の存在が分かる。新参加者のマシンはデータベースを表すオブジェクトのコピーをもらう。

会話では、通信はEthernet上のプロトコル階層によってサポートされる。このメカニズムはリモートプロシージャコールのプロトコル上に実現され、その最上層にリモートマシンのオブジェクトにメッセージを送信する仕組みと、会話でのあるオブジェクトの全アソシエートにメッセージを送信する仕組みを付け加えた。

(5) 予備観察

① 記述プロセスの構造

Cognoterの設計は、発想の早期段階から方針または概要の生成および展開を通じて最終プレゼンテーションに至るまでの記述プロセスに関する一連の推測を反映している。しかし、実際にCognoterを利用してみると、設計とプロセスの適合の点ばかりではなく、その間の微妙な不連結が明らかになった。

Cognoterを使った初期のセッションにおいて、整理フェーズに移行する前に、メンバがブレーンス

トーミングウインドウでアイデア間の関係を表示するために空間的なグループ分けを始めてしまうことが分かった。

整理および評価プロセスは、ブレーンストーミングで生成されたアイデアの組み合わせが完全であったかどうか見るのを容易にした。設計する際の最初の前提では、アウトライン作成ツールは評価フェーズが完了した後で使用すべきものであったが、実際には参加者は、アウトライン作成ツールが現在の構造の中間状態を表示する際にも便利なツールであることに気づいた。これらの観察により、もともと前提にしたこととは異なったプロセス間の「繋がり」があることが分かった。

② 協同作業の維持

全ての参加者がディスプレイや共有データへ平等にアクセスすることによって、Colabのインタフェースは役割の柔軟性を高め、一人の参加者によって活動が規制されてしまうのを防止している。

しかし初期のセッションでは、当時の技術上の制約条件は単なる協同作業の限界ばかりではなく、いくつかの点でリソースの限界をも示した。とくに記述技術については、同時に一人しかテキストを入力できないことでグループで共通の文章を維持する際に一つの（すなわちその人の動作による）視点を共有することになる。

初期のCognoter会議において、共有視点を維持する作業は会議活動の活発度合いの変化において明白だった。とくにアイデアが練られる整理フェーズの間、参加者は数分の間、当面の目標の説明や行動の短期計画の作成を言葉で対話し、グループが彼らの「宿題」にした後、しばらくの間熱心にタイプする傾向があった。並行編集を始めて数分後、人々は他人が何をしていたか分からなくなり、他人が次に何をするかを見失う。するとグループはシステムとのインタラクションを止め、再び彼らがどこにいて何をすべきかを議論するといった状況が見られた。

8.2.3 MMconf

原文タイトル： MMConf: An Infrastructure for Building Shared Multimedia Applications

翻訳タイトル： MMConf：共有マルチメディア・アプリケーション構築のための
インフラストラクチャ

著　者： Terrence Crowley
Paul Milazzo
Ellie Baker
Harry Forsdick
Raymond Tomlinson
(Bolt Beranek and Newman Inc. Cambridge, MA, USA)

出　典： CSCW '90 Proceedings, October 1990, pp.329-342

本稿では、MMConf プロジェクトの目的と、採用されたアーキテクチャ、そのインプリメント、さらにこれに付随するアプリケーションに関して紹介されている。

(1) MMConf プロジェクトの目的

MMConf プロジェクトは、分散したリアルタイムのグループ・インタラクションをコンピュータでサポートしようとするものである。本プロジェクトは、投票やブレインストーミングのためのツールのような会議プロセスそのものを支援するアプリケーションではなく、会議に内容をもたらすためのアプリケーションを取り上げ、共有環境で動作するシングル・ユーザ・アプリケーションのサポートと、本質的な共有アプリケーション開発基盤の提供を目的としている。

MMConf プロジェクトの最終的な目標は、協調作業の種類を制限している距離による障壁を、コンピュータによって乗り越える方法を見つけることである。

作業の指針となった要件には次のものがある。

① MMConf は、広範囲な帯域幅と待ち時間特性を持つ広域ネットワークで実行できること。

② グラフィックとインタラクティブ性が高度に発達したアプリケーションをサポートすること。アプリケーションの会議での対話能力は、単一で使った時と同じくらい高いこと。また、公平に全てのサイトでほぼ同じ能力が達成されなければならない。

③ MMConf は、長時間の会議も、電話に似た短い「軽量」会議も同様にサポートすること。

④ コンピュータ遠隔会議には、従来の音声やビデオによる遠隔会議も伴うことになる。MMConf では、従来の電話会議とWideband ネットワークを使ったパケット交換の音声ビデオ会議システムと共に用いた。

(2) 設計上の問題

従来のグラフィック・ユーザ・インタフェースは、シングル・ユーザを想定している。このようなインタフェースを何人かのユーザに分散させるアーキテクチャでは、コンピュテーション・モデルを集中型にするか複製型にするか、参加者のスクリーン全体を使うかウィンドウのみ使うか、メタ討議を可能にしたまま、どうやって複数のポインティング装置を一致させるかという問題を解決しなければならない。

① 集中型のアーキテクチャと複製型のアーキテクチャ

コンピュータ会議アーキテクチャを設計する上で重大な決定となるのが、集中型にするか複製型にするかである。これによって、アプリケーションの実行が変ってくる。

集中型アーキテクチャでは、アプリケーションには一つのコピーしか存在しない。中央サーバが、ローカルなスクリーンを含む全会議サイトに出力を分配し、各サイトのサーバが中央サーバに入力を送り返して、これがアプリケーションに転送されることになる。

複製型アーキテクチャでは、会議の全サイトでアプリケーションのコピーが動作する。各サイトのサーバは、全コピーが入力事象を同じ順序で見られるようにするための何らかの制御メカニズムを使って、コピーに入力を分配する。全てのコピーが同じ入力を見て、状態を同期したまま各サイトで同じ出力を発するため、コピーの出力はローカルにしか現れない。

二つのアーキテクチャでは、集中型の方が本質的に単純である。アプリケーションの変更もいらず、アプリケーションとウィンドウの間に新しい会議ウィンドウ・サーバを挿入するだけでよい。

MMConf では複製型アーキテクチャを使っている。複雑さは増すが、複製型にすることで次のような大きな利点が生まれる。

(a) 性能

複製型アーキテクチャの二つの特性によって、大きく性能が向上する。第一に、入力と出力両方ではなく、入力のみをサイト間に分配すればいいので、帯域幅が低くてすむことである。第二に、複製型システムは、ネットワーク待ち時間の変化に対する感受性が低いことである。会議の全参加者は、アプリケーションのローカルのコピーとやり取りするので、インタラクティブな性能が高い。

(b) 汎用性

集中型アーキテクチャは、サーバ・ベースのウィンドウ・システムで最もうまく動く。サーバ・ベースのシステムでは、入力事象と出力要求はネットワーク接続を通るので、分配する出力を得るためのハンドルができる。

一方、複製型アーキテクチャもサーバ・ベースやカーネル・ベース・システムを使った場合に同じくらいうまく動く。カーネル・ベースのウィンドウ・システムでは、ライブラリやシステム・コールが入力事象や出力要求を扱うので、出力のハンドルがない。複製型アーキテクチャは、一つの入力ハ

ンドルしか必要としないので、割込みがはるかに容易になる。ウィンドウ・システムには、入力を得るための入口点は一つだが、出力を発生する入口点はたくさんあるのが普通である。

出力のローカルな発生も、異種環境での実行を容易にする。ウィンドウ・システムのまったく異なるサイト間での複製型遠隔会議では、入力事象を翻訳するだけでよい。

(c) 容易なサイト指定インタラクション

複製型アーキテクチャでは、新しいアプリケーションでも共有環境でのローカルなインタラクションを容易に実行できる。集中型アプリケーションでは個々のインタラクションができるが、会議出席者一人一人について別個に事象処理スレッドを維持しなければならない。複製型アプリケーションは、状態の通知を他の会議出席者にマルチキャストするだけで、シングル・スレッドのままでいられる。

複製型アーキテクチャの大きな欠点は、アプリケーションの全てのコピーについて同じ状態を保つことが難しい点である。この場合障害となるのは、初期アプリケーション状態の相違、順序の誤った入力事象、非決定的なアプリケーション、会議遅参者の四つである。

また、これ以外に、プロセッサとファイル・システムの速度の差のため、複製型アーキテクチャは一時的な矛盾が発生する。会議出席者がエディタに文書を読みこませた後に、全てのスクリーンで文書が見えるかどうか口頭で話し合う場合（「もう見えた？」）などがそれである。集中型アーキテクチャにも、ネットワークとローカルなウィンドウ・サーバの待ち時間によってディスプレイへの出力が矛盾することがあるが、遅れは小さいものであることが多い。

② スクリーン・ワークスペース管理

コンピュータ会議アーキテクチャでは、ユーザのスクリーン全体を管理するか、一定のウィンドウのみを管理するかを選択しなければならない。後者の場合には、どのウィンドウ制御がローカル・スクリーンだけに影響を与えるかも選択しなければならない。ウィンドウ制御には、ウィンドウ・サイズ、スクリーン上の位置、他会議ウィンドウとの相対位置、スタッキング順、アイコン化などがある。

会議ウィンドウのみを管理することで、ユーザは自分のワークステーションで独自の活動を行うことができる。これは、軽量会議が既存ウィンドウに影響を与えないで現れたり消えたりするためにも重要である。MMConf はこのモデルに従っている。

参加者のプライベート・ウィンドウのレイアウトがそれぞれ異なっていて、会議中もそのプライベート・ウィンドウで作業を行っている場合、ウィンドウ位置とスタッキング順をローカル決定とすることが重要であると考えた。アイコン化を独立して制御することで、共有状況が減り、アプリケーションは、アイコン化されているかオープンかに応じて、異なった行動を取る。MMConf では、アイコン化はアプリケーションの全コピーに影響を与える。アプリケーション事象の解釈にウィンドウ・サイズが重要なことが多いが、MMConf では、ウィンドウのコピーは全て一緒にサイズを変更する。

③ メタ討議

分散グラフィック・ユーザ・インタフェースの問題には、メタ討議能力を保持しながら、会議の要素に注意を向けるためのポインティングやスケッチング等複数のポインタを一致させる点がある。コンピュータ遠隔会議システムでは、一つの会議にポインタが一つしか使えないか、会議出席者と同数のポインタを同時に使えるかのどちらかで、いずれのアプローチも先例がある。

MMConf のテレポインタは、会議ウィンドウ内にしか現れない非常に大きな目立つ矢印である。現在、使用中のユーザが真中のマウスボタンを押すことでテレポインタを呼出す。既存のアプリケーションの機能性を排除することのないよう、このテレポインタは通常のポインタ事象も送る。

また、スケッチングについては、透明のウィンドウなら、どんなウィンドウにも重ねられる単純で一貫したスケッチング機能が得られるが、残念なことに、本システムがサポートされるウィンドウ・システムに透明なウィンドウはない。そのため、MMConf ではいくつかの討議向きアプリケーションの中に直接、スケッチング・サポートを加えている。

(3) MMConf アーキテクチャ

MMConf アーキテクチャは、次の五つの部分から構成される。

- 会議管理と通信
- アプリケーション・プログラミング・モデル
- アプリケーションへの入力に割込む方法
- プログラミング・モデルをウィンドウ・システムに合うようにインプリメントする方法
- ウィンドウ・システムに特有の入力事象と MMConf の一般的な事象記述との間のマッピング

MMConf のユーザ・インタフェース・ツールキットには、会議中に行動を変更する機能が多く含まれている。MMConf のアプリケーションにも、会議認識コンポーネントが入っている。会議認識には、同期確保、サイト間のデータ共有、副作用の制限、並列活動の容易化の四つの目的がある。

このツールキットによって、アプリケーションには次のような基本的な会議認識メカニズムが得られる。

① 会議情報

最も単純な照会は、アプリケーションが会議の下で動作しているかどうかである。他にも、どの会議出席者が最新の事象を発したか、ローカルなアプリケーション・ピア（アプリケーション・プロセスの各サイトにおいて対応する複製）が発言権を持っているかどうか、会議には何人参加しているか等の照会がある。

② 発言権管理

発言権管理が稼働していると、アプリケーションは発言権をロックして一定のインタラクション中

に変更できないようにすることができる。アプリケーションは、発言権がロックされている間、システムが事象を分配すべきかどうかも決定できる。アプリケーションは、ファイル名収集等、一つのサイトでのプライベートなインタラクション中は発言権をロックするのが普通である。

並列活動をサポートするため、ツールキットによってアプリケーションが発言権管理を禁止して、事象を自動的に分配するかどうか指定できるようにする。発言権を保持しないアプリケーションでは、全会議出席者が同時にアプリケーションとインタラクトでき、各アプリケーション・ピアは、異なる順番で事象が到着するのを見ることになる。これは通常のアプリケーションでは混乱の原因となるが、会議認識アプリケーションは、会議出席者ごとに制御スレッドを維持するため、各事象の起点を使うことができる。

③ メッセージ転送

アプリケーションに固有のプロトコルのための基本的な転送メカニズムは、メッセージを全てのアプリケーション・ピアにマルチキャストする機能である。この基本的な機能の上位として、任意のデータを引数としてパスして、登録機能のファミリの一つを呼出す機能がある。この機能は完全な遠隔手続き呼出し機能ではなく、呼出された機能は値を戻すことができない。

どちらの転送メカニズムも非同期で、送信プロセスは応答待ちをブロックしない。受信プロセスは、メッセージを入力待ち行列の事象の一つとしか見ない。

④ ファイル転送

複製型アーキテクチャの欠点に、データ・ファイルも複製しなければならない点がある。会議が始まる前にファイル分配が必要とされることは、自発的な会議や、計画された会議に新しい資料を導入する妨げとなる。このため、アプリケーションが新しいファイルを容易に導入でき、全ての会議出席者が同じファイルを入手できるようなファイル転送機能を MMConf に持たせた。

この基本的なメカニズムによって、一つのサイトのアプリケーションがファイルの転送を要求できるようになる。遠隔サイトは、転送ファイルの名称かエラー表示を含んだメッセージを受取る。

(4) インプリメント

UNIX 上では、MMConf は各アプリケーションにリンクされたユーザ・インタフェース・ツールキットと、ツールキット内でルーチンが通信する会議マネージャ・プロセスからなる。会議マネージャは、会議の出席者を表し、会議出席者間の全ての通信を取り扱って、出席者が会議に参加したり退席したり、新しい会話を開始するためのユーザ・インタフェースを提供する。

現在 TCP/IP 上にインプリメントされている一つのデータ・ストリームが、ローカルな会議マネージャを遠隔会議マネージャに接続し、もう一つのデータ・ストリームは、ローカルなマネージャをローカルなアプリケーション・ピアに接続する。アプリケーションは、全ての会議トラヒックを会議マネ

ージャに送り、マネージャはそのコピーを進行中の会議ストリームに書込むことでこれをマルチキャストする。

① 会議開始

会議を開始するには、会議発起人である一人のユーザが会議マネージャを呼出し、ユーザと他会議出席者のホスト名と最初の会話のトップレベル・アプリケーション名を指定する。発起人の会議マネージャは、他会議出席者のホストの周知のポートに接続して、ホストが会議マネージャのコピーをスタートさせる。

遠隔会議マネージャは、発起人が誰であるかと、ユーザが会議招待を受けるか拒否するかの応答を要求した対話ボックスを、全出席者のスクリーンに表示する。被招待者全員が応答するか時間切れになると、発起人の会議マネージャが他のマネージャ全員に全会議出席者のリストを送り、最初の会話を開始する。

② アプリケーション開始

会話をスタートさせる会議マネージャは、まず会話に独自の識別子(ID)を割当てる。そしてマネージャはIDとトップレベル・アプリケーション名、コマンドライン引数をマルチキャストする。

すると各マネージャは、アプリケーションのコピーを生成して、IDとマネージャが接続要求待ちしているIPCポート名の入った変数をアプリケーション環境に入れる。アプリケーションが第1回目にツールキット機能を呼出すと、その機能は環境変数を通知して、自動的にマネージャのポートへのIPC接続を開く。

③ 発言権管理

会話には一つの発言権しかなく、これが会議に同時に事象をもたらす出席者を識別する。発言権は、トークンと順序カウントからなり、カウントは、参加者二人以上の会議での発言権譲渡中、事象の順序が誤るのを防ぐために利用する。

MMConfは、発言権管理の概念をメカニズムと方針に分け、メカニズムでは発言権トークンの受渡しと全参加者のための同期事象ストリームの保持という低レベルの活動を扱う。方針では、会議出席者がどのように発言権を要求してこれを認めるかを決定する。会話毎に発言方針を異なるものとすることが可能で、出席者はいつでもこれを変更できる。

④ 入力事象の分配

ツールキット内のコードはアプリケーションの全ての入力事象を受取る。アプリケーションが発言権を持っている場合、ツールキットはまず事象をアプリケーションの待ち行列に入れ、会議マネージャとのインタラクションを省いて、インタラクティブな性能をよくする。次に、ツールキットは事象をウィンドウ・システムに依存しない形式に翻訳して、これを会議マネージャに転送すると、会議マネージャはアプリケーション・ピアIDをスタンプして、他会議出席者にマルチキャストする。

アプリケーションに発言権がないと、事象を受取った時のツールキットの行動は、その時の発言権要求方針で決まることになる。暗示的要求方針では、ツールキットは発言権要求を出して入力待ち行列をブロックし、明示的要求方針では事象を放棄する。

会議マネージャは遠隔の入力事象をバッファし、一連番号で並べる。発言権譲渡中は、受取の順序が狂うことも有りえる。するとマネージャは、IDで順序付けた事象をローカルなアプリケーションにディスパッチする。

マネージャから事象を受取ると、各アプリケーション内のツールキットは事象をウィンドウ・システムに対応する形式に翻訳し直し、ローカルに発生したかのように待ち行列に入れる。しかし、会議認識アプリケーションは、事象を最初に発生した参加者を常に認識することができる。

(5) アプリケーション

MMConfプロジェクトにとって大きな目標は、有益な共有アプリケーションを構築することである。コンピュータ遠隔会議に用いて、遠隔会議アーキテクチャの要件を考察したアプリケーションとしては、次の五つのものが挙げられる。

① BBN/Slate Document Editor

元々はシングル・ユーザ・アプリケーションであるBBN/Slateを遠隔会議に使うことにより、ユーザはテキスト、グラフィック、画像、スプレッドシート、音声の入った文書を見て編集することができる。

② ViewShell

MMConfアーキテクチャの柔軟性を持つ端末エミュレータである。エミュレータ自体は、ユーザ入力の管理に発言権管理と事象分配を使う通常の複製型プログラムだが、コマンド・インタープリタ（シェル）とこれが起動させる全てのプログラムは一つのサイトでしか動作しない。

ViewShellは本来、端末ベース・プログラムのための集中型会議アーキテクチャを形成し、集中型アーキテクチャの利点を全て持っているため、MMConfユーザは、修正を行っていないアプリケーションやハードウェアに依存するアプリケーションにさえアクセスできる。

③ ビデオ・ツール

各種ビデオ・アプリケーションによって、近接したワークステーション間でビデオ資源を共有するための強力な環境が生まれる。ユーザ・インタフェース・ツールキットでのファイル名収集機能同様、ビデオ・ツールキットにおける機能は、ビデオ・アプリケーションに必要な全ての会議認識を持っている。

④ Sketch

発言権管理や自動的な事象分配なしに動く単純なグラフィック・エディタで、全会議出席者が同時

にグラフィックにアクセスすることができる。Sketch は、アプリケーションに固有のメッセージを使って、各出席者がグラフィックに加えた変更を告示する。Sketch にはまた、会議参加者がプライベートなウィンドウから画像やテキストを取込むためのスクリーン・グラブやテキスト・グラブ機能がある。

⑤ Presenter

遠隔会議でのスライド・ショーを実現するものである。スライドにはすべてアプリケーションと関連データ・ファイルの名前が付いていて、これを選ぶことでアプリケーションがスタートし、データを読みこんで表示する。

8.2.4 Rapport

原文タイトル：	A Comparison of Application Sharing Mechanisms in Real-Time Desktop Conferencing Systems
翻訳タイトル：	リアルタイム・デスクトップ会議システムにおけるアプリケーション共有メカニズムの比較
著　者：	S.R. Ahuja J.R. Ensor S.E. Lucco (AT&T Bell Laboratories, Holmdel, NJ, USA)
出　典：	ACM SIGOIS Bulletin, Vol.11, No.2, April 1990, pp.238-248

本稿では、リアルタイムのコンピュータ・ベースの会議においてアプリケーション・プログラムを共有する方法を比較するために、Rapportというマルチメディア会議システムの三つのバージョンにおける、異なった手法が比較検討されている。

(1) リアルタイム会議システムの分類

リアルタイムのコンピュータ・ベースの会議システムは、次のように分類できる。

① クローズド・システム

クローズド・システムは、会議参加者のコンピュータに分散される単一の専門化されたプログラムとして開発されたシステムである。この場合、各参加者のコンピュータに存在する分散プログラムは、それが受け取るコマンドを解釈し、そのコンピュータ画面上に表示を生成する。これらの分散プログラムは、すべての参加者にとって状態の整合性をとるために、お互いにメッセージを送り合う。

② オープン・システム

オープン・システムとは、既存の一般に利用されるプログラム（アプリケーション・プログラム）を、修正することなく会議中で実行できるようにしたシステムである。例えば、それぞれの参加者の自分の気に入りのエディタ、VLSI CADシステム、財務計画エキスパート・システム等が、コンピュータ・ベースの会議の最中に利用できるのである。クローズド・システムと同様に、会議参加者のコンピュータの上のオープン・システム・モジュールは、状態の整合性を保持するため、お互いにメッセージを送り合う。また、オープン・システムは、アプリケーション・プログラムの出力表示を作り出すために、入力あるいは出力コマンドを各参加者のコンピュータに送る必要がある。かくして、オープン・システムは、一般的に言って、クローズド・システムよりも多くのメッセージ・トラフィックを作り出す。しかし、オープン・システムは拡張の際クローズド・システムのような必要性がない。新しい機能は、会議の間に、追加のプログラムを実行するだけで得られるのである。

オープン会議システムは、どの会議コンピュータの上でアプリケーション・プログラムを実行すべきであるかという基本的な問題を解決しなければならない。このためのアプローチとして、次の二つがもっとも一般的である。

(a) マルチ・サイト・アプローチ（複製実行方式）

このアプローチでは、会議の中の二つ以上のコンピュータが、同期と入力制御に関するある種の制約条件のもとで、それぞれのアプリケーション・プログラムのコピーを実行する。プログラムの複数のコピーは、すべて同期をとって開始され、正確に同一の入力を与えられることによって、歩調を合わせる。種々の入力制御プロトコルをインプリメントするために、入力にはフィルタをかけることができる。このようにして、プログラムのコピーは、同一の出力を生成し、それぞれが会議参加者のコンピュータの画面上に表示される。

このアプローチの利点は、会議コンピュータの間で入力コマンドだけが伝達されるのであるから、ネットワークのトラフィックが相対的に少なくて済むという点である。これは、コンピュータが広域ネットワーク（WAN）によって接続されているような会議に対しては、ネットワークへの負荷要求が少ないマルチ・サイト・アプローチが適しているということを示唆するものである。

このアプローチの大きな欠点は、アプリケーション・プログラムの複数のコピーの間で、整合性を保つことの困難さである。このような同期性の喪失を検知し、会議参加者の間の整合性を回復する技術が必要であるが、ほとんどのアプリケーション・プログラムは、以前の整合性のある実行状態を回復することができない。さらに、この技術は、会議参加者のそれぞれが、それぞれのアプリケーション・プログラムを同一のハードウェアおよびソフトウェア環境で実行することを必要とする。したがって、このアプローチは、共有アプリケーションが状態保持を殆ど必要とせず、再同期を取ることが容易である場合、つまり「スライドショー」アプリケーションである場合にのみ適しているように思われる。

(b) シングル・サイト・アプローチ（中央集中実行方式）

このアプローチでは、単一のサイトのみがそれぞれのアプリケーション・プログラムを実行する。会議の参加者からアプリケーションへの入力は、実行サイトへ送信される。アプリケーションからの出力コマンドは会議参加者全員のコンピュータへブロードキャストされる。このアプローチの主な利点は、会議中に同期と整合性を保持する重荷の大部分が、アプリケーション・プログラムから会議システムへ移されることである。それぞれのアプリケーション・プログラムについて、ただ一つのコピーしか実行されないので、整合性を取る問題は避けられる。さらに、このアプローチでは、会議参加者は自分ではプログラムを実行しなくても実行結果を見ることができる。したがって、自分の環境におけるソフトウェアとハードウェアに、他者とは幾分かの相違があっても、大勢の人々が会議に参加することができる。これによって、プログラム・コードを共有することなく、プログラムを共有する

ことができ、プログラムのセキュリティおよび所有権が保持される。このアプローチの不利な点は、入力メッセージと出力メッセージの両方が通信されるため、マルチ・サイト・アプローチよりも、ネットワークのトラフィックが大きいことである。

(2) Rapportのインプリメントの概要

Rapportマルチメディア会議システムは、一群の人々がオフィスにあるコンピュータと電話を使用して、音声、データ、イメージを共有しながら、リアルタイムの議論ができるようにしたものである。比較のために三つのバージョンが開発され、会議においてアプリケーションの共有をサポートするためのもっとも一般的な方法がインプリメントされた。Rapportの異なったインプリメントについての測定結果は、上記のアプローチについての、定量的な比較を提供するものである。

Rapportシステムには二つの基本機能がある。まず、呼び制御機能が提供されている。これは人々が会議を開始し、実行し、終了するためのメカニズムである。第二に、Rapportは、会議参加者の間で表示を共有するための機能、アプリケーション・プログラムの実行を制御する機能、それらの間の整合性を保持する機能、およびそれらに関連したウィンドウを管理する機能を持っている。Rapportの呼び制御機能は、通常の電話の呼び制御機能に基づくものであるが、画面ベースの電話からアクセスされる。アプリケーション・プログラムの管理とそれらの表示は、仮想会議室のメタファに基づいている。Rapportは電子会議室を表示し、参加者は自分のアプリケーション・プログラムを携えてその仮想的会議室に入ることができる。

Rapportの三つのバージョンにおいては、それぞれ異なったアプリケーション・プログラム実行環境が提供されている。一つのバージョンにおいては、アプリケーション・プログラムに対するマルチ・サイト実行環境がサポートされている。そこでは、それぞれの共有アプリケーションの入力コマンドが会議参加者へマルチキャストされる。他の二つのバージョンでは、シングル・サイト実行アプローチが取られており、会議参加者にはそれぞれの共有アプリケーション・プログラムの出力コマンドがマルチキャストされる。

Rapportの三種類のインプリメントのすべては、特別に設計されたボードによってSunワークステーションと電話を結合したノードに基づいている。ワークステーション間のデータ通信には、10Mビットの Ethernetが利用される。音声と制御信号は別のネットワークで通信される。このネットワークの集合体は、ISDNの通信チャネルを真似たものであり、本開発においてはISDNへの移行を容易にするために活用された。データおよび音声ネットワークによって結合されたこれらのワークステーション上で、Rapportの機能は完全にサポートされている。Rapportは、UNIXおよびX Windowシステムによってサポートされたソフトウェア環境で実行される。X Windowサーバとそれぞれのアプリケーション・プログラムとの間の通信は、Rapportの会議マネジャによってフィルタされる。アプリケーショ

ン・プログラム、X Windowサーバ、および会議マネジャ間の関係は、三つのバージョンの間で異なっている。

以下に、この三つの実行環境の各々を利用して、三台のコンピュータ（A、B、C）間でやりとりされたRapport会議の動作を説明する。

① マルチ・サイト・アプリケーション実行環境（MULTI）

Rapportの最初のバージョンは、アプリケーション・プログラムのマルチ・サイト実行をサポートしており、MULTIと呼ばれる。それぞれのコンピュータは、X Windowサーバ、会議マネジャ、およびアプリケーション・プログラム（P1とP2）を実行する。各アプリケーション・プログラムは、X Windowサーバに接続されている。X Windowサーバは、会議マネジャに接続され、会議マネジャはお互いに接続されている。すべての接続はソケットを介している。このバージョンにおいては、X Windowサーバは、会議マネジャとの間でメッセージをやり取りできるよう改造されており、これによってRapportは、アプリケーション・プログラムへの入力をフィルタし、マルチキャストすることができる。

まず、コンピュータBにいる参加者がアプリケーションP1への入力を行うと、B上の会議マネジャは、この参加者がこのアプリケーションへの入力制御を持っていることを認識し、Bの上のX Windowサーバに入力ストリームを与える。B上のX Windowサーバは、その入力ストリームをローカルの（B上の）アプリケーションに渡す。また、会議マネジャは、その入力イベントをコンピュータAおよびCの上で実行されているマネジャに送る。これらのマネジャは、入力をそれぞれのローカルなX Windowサーバに与え、そこで実行されているアプリケーションに渡るようにする。コンピュータ画面に表示を行うためのX Windowシステムの出力コマンドは、アプリケーション・プログラムのそれぞれのコピーによってそれぞれのローカルなX Windowサーバに送られる。

② シングル・サイト・アプリケーション実行環境（SINGLE）

Rapportの第二と第三のバージョンは、アプリケーション・プログラムのシングル・サイトでの実行をサポートしている。ここで説明するインプリメントでは、LANにもWANにも適しており、SINGLEと呼ばれる。それぞれのコンピュータは、X Windowサーバおよび会議マネジャを実行する。しかし、それぞれのアプリケーション・プログラムは、ただひとつのコンピュータの上でのみ実行される。このインプリメントにおいては、アプリケーション・プログラムはX Windowサーバには接続されておらず、会議マネジャに接続されている。 会議マネジャは、擬似的なX Windowサーバとしての役割を果たしている。X Windowサーバは改造されていない。それぞれの会議マネジャは、自分自身のコンピュータの上のX Windowサーバに接続されているが、リモートなX Windowサーバにはまったく接続されていない。会議マネジャは互いに接続されている。ここでもすべての接続はソケットを介している。

コンピュータAは、アプリケーションP1の実行サイトであり、コンピュータCは、アプリケーションP2の実行サイトである。アプリケーションP1への入力はリモートな会議ノードであるコンピュータBにおいて生成され、アプリケーションP2への入力はローカルに生成されている。Bにおける会議マネジャは、この参加者がこのアプリケーションへの入力制御を持っていることを認識し、入力ストリームをアプリケーションへ送る。アプリケーションが出力を生成したとき、それはローカルな会議マネジャに直接与えられる。会議マネジャは、それをローカルなXサーバに与える。また、会議マネジャは、このXの出力要求をコンピュータBおよびCの上で実行されている会議マネジャに与える。Xのリソース識別子の変更が必要であるが、これは、BおよびCの会議マネジャによって行われ、これらの会議マネジャは、それぞれのローカルなX Windowサーバに適切なXの出力要求を与える。それぞれの会議マネジャには同一のメッセージが送られるので、この通信は真にマルチキャストである。このマルチキャストの特性によって、このアプローチはWANに適したものとなっている。WANにおいては、メッセージの解釈は期待できないからである。さらに、このアプローチでは、一台のコンピュータ当たり、ただ一つのコネクションしか必要ではない。必要なのは、会議マネジャと何らかのネットワーク・スイッチの間の接続だけであり、これはWANにとって単純な構成である。

③ LANのためのシングル・サイト・アプリケーション実行環境（SINGLE／LAN）

ここでは、②に対する改造を説明する。これは、ある種のX Windowシステムのプロトコル・メッセージに対する経路の長さを短縮するものである。このバージョンでは、X Windowサーバの遠隔通信機能が利用されており、現在のところでは、LAN環境でのみサポートされている。このため、このバージョンはSINGLE／LANバージョンと呼ばれている。それぞれのコンピュータは、X Windowサーバおよび会議マネジャを実行する。しかし、それぞれのアプリケーション・プログラムは、ただひとつのコンピュータの上でのみ実行される。ここでも、会議マネジャは、擬似的なX Windowサーバとしての役割を果たしている。X Windowサーバは改造されていない。X Windowサーバのリモート・ホスティング機能を利用して、アプリケーション・プログラムに対応した会議マネジャは、会議コンピュータのそれぞれのX Windowサーバに直接接続される。

他の例と同様に、現在、アプリケーションP1への入力はコンピュータBにいる参加者によって生成されている。Bにおける会議マネジャは、この参加者がこのアプリケーションへの入力制御を持っていることを認識し、入力ストリームをコンピュータAの上のアプリケーションへ送る。アプリケーションが出力を生成したとき、それはローカルな会議マネジャに与えられる。会議マネジャは、それを、コンピュータA、B、Cの上のX Windowサーバに渡す。マネジャは、必要なリソース識別子の変更を行い、その後、コマンドをX Windowサーバに送る。会議マネジャがこれらの変更を行うことによって、Xサーバは会議を意識しなくて済む。しかし、マネジャは、すべてのリモートなX Windowサーバへの直接の接続を必要とする。会議マネジャは、相互に接続され、会議の状態情報を通信し合

っている。ここでもすべての接続はソケットを介している。

(3) 性　能

Rapport の三つのバージョンである、MULTI、SINGLE、SINGLE／LAN の性能を比較する。比較は、会議コンピュータの間のメッセージの数に焦点をあてる。これは通常の会議活動の間にカウントされたものである。

① 比較の環境

三種類のインプリメントの比較の要素となる、会議マネジャとアプリケーションとの間のメッセージは、三つのカテゴリに分けられる。管理メッセージ、入力メッセージ、出力メッセージである。管理メッセージとは、会議を設定し、参加者の追加削除を行い、会議のインタフェースのプロセスと通信するものである。入力メッセージとは、X 入力イベントをアプリケーションへ送るもので、会議マネジャのプロセスの間に送られる中間的なメッセージもこれに含まれる。また、このカテゴリのメッセージには、情報問い合わせへの応答も含まれる。出力メッセージとは、アプリケーションの出力要求を X Window サーバへ送るものである（中間的メッセージも含む）。

また、三種類のインプリメントの各実験環境において、アプリケーション、およびアプリケーションの実行サイトとその入力元の関係を合わせるために、会議「スクリプト」を作成することで対応した。すなわち、測定においては下記の三者会議のシナリオを使用した。

> A が B と C を呼び出す。B と C は受話器を取り上げる。A は端末エミュレータ・アプリケーションを開始し、テキスト・エディタ vi を起動し、事前に決められた編集操作を実行する。A は、端末エミュレータの制御を B に渡す。B は別の一連のコマンドを実行する。C は対話的図面作成プログラムを開始し、事前に決められた図を作成する。C は制御を A に渡し、A は図を修正する。

② 比較結果（メッセージの数と応答時間）

ここでは、SINGLE と MULTI の比較に焦点を当てる事にする。SINGLE／LAN の測定結果については簡単に触れるにとどめる。

(a) バッファリングのない場合のメッセージ数

三つのバージョンすべてが同一の管理メッセージを会議マネジャに送信した。会議の間に、98 個の管理メッセージが発生した。その大部分は会議インタフェース・プロセスからの状態問い合わせによるものであった。すべての場合において、これらのメッセージは、メッセージ・トラフィック全体の 15% 以下であった。メッセージ・トラフィックの大部分はディスプレイからの入力と出力メッセージであった。MULTI は、全体で 636 個のメッセージを送り、そのうち 538 個は X の入力イベントを運んでいた。SINGLE は、全体で 3875 個のメッセージを送り、そのうち 87% は X の出力イベント、11%

はXの入力イベントであった。SINGLEが送るメッセージに入力イベントを運ぶものが少なかったのは、SINGLEの場合、入力はポイント・トゥ・ポイントで送るだけで良いからである。LANの送ったメッセージはSINGLEのそれのおよそ半分であった。これは会議マネジャとX Windowサーバが直接接続されることが多いからである。

会議シナリオにおいては、アプリケーションから出される出力要求の数は、生成されるX入力イベントの数のおよそ5倍であった。全体的に見れば、バッファされない場合においては、シングル・サイト・システムは、マルチ・サイト・システムに比較して、およそ4倍のXイベント・メッセージを生成している。

出力メッセージの平均パケット・サイズは、25バイトであった。管理メッセージは98バイト、入力メッセージは24バイトであった。入力メッセージはほとんど常に単一で来る（50ミリ秒以上の間隔）。入力メッセージに関して観察された最大の突発は、10ミリ秒以内に生成された6個のメッセージであった。出力メッセージは、時間的に良く分離されている場合が多い（50ミリ秒以上の間隔）。しかし、出力メッセージは、入力メッセージよりも、束になって発生する傾向がある。出力メッセージの最大の突発は、10ミリ秒以内に生成された89個のメッセージであった。これらの突発は、出力メッセージの数を入力メッセージの数よりずっと多くする原因となる。

(b)　バッファリングを行った場合のメッセージ数

Rapportの三つのインプリメントのすべては、ペンディングとなっているすべての到着メッセージを読んで処理し、生成された出力メッセージをバッファの中に集めるメッセージ・バッファリング・パッケージを利用するよう構成することができる。このパッケージは、その後、バッファを単位としてそれぞれの目的地へ送る。UNIXのソケットは、部分的メッセージも送ることができるから、受取側では、バッファリングなしの場合と同じ振舞いで良い。バッファリング・パッケージは、無視できるほどのソフトウェア・オーバヘッドしか持たないが、どのインプリメントにおいて会議スクリプトを実行する場合にも、必要なメッセージの数を劇的に減少させる。出力メッセージは、入力メッセージよりも、束になって発生する傾向が強いから、バッファリング・パケットを利用することの効果は、SINGLEの方がMULTIよりも大きい。SINGLEのインプリメントにおいて、バッファリングが使用された場合、出力メッセージの入力メッセージに対する比率は、3.4対1である。さらに、会議スクリプトを実効するために、バッファリングを使用するとき、SINGLEは、MULTIに比較して、3.6倍のノード間メッセージしか使わない。バッファリングをしたときのSINGLEの平均パケットサイズは、出力メッセージが58バイト、入力メッセージが26バイトであった。これらのパケットサイズは、ナローバンドISDNおよびブロードバンドISDNの予想されるハードウェア容量の範囲内に十分おさまっている。したがって、SINGLEの場合のソフトウェア／ネットワーク・パケット通信速度のピーク要求は、MULTIの場合のそれのおよそ3.5倍であると結論する。

(c) 応答特性

端末エミュレータ (xterm) を実行するすべてのバージョンについて、このアプリケーションをローカルで操作しようと、リモートで操作しようとにはかかわりなく、本質的には即時の応答時間を観測した。すべての表示は、1秒以下の時間内で同期が取られた。

黒板アプリケーションで、細い線をリモートで描く場合、すべてのバージョンで即時の応答時間が得られた。太い線をリモートで描く場合、SINGLE のユーザは、若干の遅延時間を観察した。これは、太い線を描くために必要な余分の出力メッセージが原因で発生したものである。バッファリングを行った場合には、すべてのバージョンにおいて、最大応答時間は1秒以下であった。すべての平均応答時間は 0.1 秒以下であった。X Window システムに基づく典型的なアプリケーションによる測定は、SINGLE と SINGLE／LAN は、MULTI の場合よりも、6倍のバンド幅のネットワークを必要とすることを示している。しかしながら、静的な分析および、動的性能計測の両者によって、三種類のインプリメントはインタラクティブな応答性能においては、同程度であると結論する。

8.2.5 Commune

原文タイトル： Commune: A Shared Drawing Surface

翻訳タイトル： Commune：共有ドローイング・サーフェス

著　者： Sara A. Bly
(Xerox Palo Alto Research Center, Palo Alto, California, USA)
Scott L. Minneman
(Center for Design Research, Mechanical Engineering Department, Stanford University, Stanford, California, USA)

出　典： ACM SIGOIS Bulletin, Vol.11, No.2, April 1990, pp.184-192

本稿では、遠隔作業を行う設計者同士がドローイング・サーフェスを共有し、対面で作業する場合と同じインタラクションの多くを共有することができるプロトタイプのツール、Commune について紹介されている。

(1) Commune の初期の研究

Commune は、共有ドローイング・サーフェスの設計と利用に関して行われている研究の一部である。本研究のテーマの一つは、ビデオテープで記録した作業状況と、その作業を支援する新技術の設計開発との関係を理解することである。本研究は、共有ドローイングに関して、設計中にドローイングスペースで起こる活動の特性を明らかにして、技術で支援できるドローイング活動分野を特定しようとするものである。このような活動には、話し合いやドローイング・サーフェスの利用、参加者のジェスチャなどが含まれる。Commune では、ドローイング・サーフェスそのものに関連する非言語的な活動に焦点を合わせた。

本研究の初期の段階で、Bly は二人の人間による設計セッションを、①対面、②オーディオ／ビデオでリンクされているが地理的には分離、③電話でのみ接続という三つの異なる状況で実験を行った。この研究から、設計者は共同設計プロセスの一部として広範囲にドローイング・サーフェスを利用し、ドローイングされたものだけでなく、ドローイング活動そのものが重要であることが結論づけられた。このような結論はデータの観察に基づくもので、どのセッションでも、設計者たちは数秒間の内にドローイング、ライティング、ジェスチャをしばしばすべて実行しながら迅速に行っており、状況が許せば規則的にドローイング・サーフェスを参照していた。これは対面のケースでは特に顕著で、当然のことながら電話の場合は不可能だった。どのセッションも、強調のためのドローイング・サーフェスの頻繁な利用が特徴となっており、設計者たちがお互いのドローイング・サーフェスやジェスチャを見ることができない電話セッションの場合でさえ、話し合っていることについて図を書いたりジェスチャを行って話し合いの内容を強調し続けていた。設計者たちが一つのドローイング・サーフェス

上で指示したり書き入れたりすることのできる対面セッションでは、全体の時間の約50%で同じドローイングに関してやり取りを行っていた。

(2) Commune

Commune は、これまで研究してきた同時に二人から三人が作業するグループによる設計活動の支援を目的としたものだが、特に参加者が異なる場所に存在する場合に焦点を当てている。初期の研究から、次の設計目標を満たすことが、共有ドローイング・システムの効果的な利用に役立つことがわかった。

- 各設計者の書いたものやジェスチャを、他設計者が同時に見ることができること。
- 設計者は、ドローイング、ライティング、ジェスチャの迅速な切換えができること。
- 全設計者が、共有スペースで同時に書いたりジェスチャを行えること。
- ツールはできるだけ「ナチュラル」なものであること（よく使われているライティング／ドローイングツール、入力した位置にその入力した図や文字が現れることなど）

このような目標を満たすCommune のモデルとして、次のような形態を考えた。すなわち、各ユーザは、指示やドローイング、ライティング用のペンを持っており、そのペンで書かれたものは各々他と識別可能とする。システムとしては、ユーザが同一の面に書くと、その面に書いたものが現れる。面は水平に設置され、ライティング・ツールはペンのようなものとした。

① インプリメント

初期プロトタイプは、MS-DOS の '386 PC のプラットフォーム上に構築し、スタイラス付き透明デジタイジング・タブレットをライティング面とする。各デジタイザは、スクリーンが水平面を向くようにしたフラット・スクリーン・ディスプレイ・モニタ上に直接搭載する。Commune の各ワークステーションにはモニタとデジタイザが配置され、モニタは、同じ画像が全ユーザに同時に現れるようデイジーチェーン式になっている。

各デジタイザ・タブレットは、それぞれのスタイラスの位置を連続してプロセッサに知らせ、Commune は各ユーザからのスタイラスの現在位置にカーソルを表示する。スタイラスで押し下げると、カーソルの跡に線が引かれる。このように、設計者はスタイラスを動かすだけで指示したり、スタイラスを押し下げて動かすことでドローイングやライティングを行うことができる。全設計者は、このような動きを、知覚できる遅れなしに同時に見ることができる。

各ユーザのスタイラスは、スクリーンの上ではそれぞれ異なった色の鉛筆形のカーソルとして表れる。書かれたものはピクセル（これも色で識別される）で、直線や曲線、テキストといった区別はない。最初のプロトタイプでは、Commune は三つのメニュー・オプションしか持っていない。ドローイング・サーフェス上に書かれたものを全てクリアする「ERASE」と、現在書かれたものをセーブしな

がら１ページ先に進む「→」、以前にセーブしたページに戻る「←」である。

② Communeの利用結果

Communeを使って、二人による三つの設計セッションに関する初期の研究と同様の研究を行った。遠隔協調作業をシミュレーションするため、二つのオフィスにそれぞれ2台のCommuneドローイング・ステーションを設置し、各オフィスのモニタとカメラが、もう一方のドローイング・ステーションにいる人間の顔を映す。音声のやり取りには電話を使用した。被験者である本論文著者の一人とその同僚の製品設計担当者の二人の設計者は、初期の研究にも参加している。製品設計者が現在従事している問題について、この二人がCommuneを使って2時間の設計セッションを行った。

この設計活動全体を完全に記録するため、数種のビデオが集められた。各ワークステーションを肩越しに撮ったビデオは、各設計者がいつどのようにシステムとやり取りをするかをはっきりと記録していた。

ビデオテープ上の行動は、「アクション」、「用途」、「インタラクション」、「リアクション」という四種類のカテゴリに分けて記録した。「アクション」については、ユーザが面に英数字を書いたり（ライティング）、図面を描いたり（ドローイング）、書かれたものにカーソルでジェスチャを行うごとに事象を記録した。一つの事象は、ユーザがアクションを変更したり現在の書かれているものからカーソルを移動させたりした時に終了したものとみなされた。各事象は、図解や強調、おぼえ書き、参照、その他といった、「用途」によっても分類された。

観察結果は、上記の四種類のカテゴリのいずれの面でも、遠隔セッションよりも対面の設計セッションに近い傾向を見せた。

(a) アクション

Communeでは、ユーザは他の遠隔状況よりジェスチャを多く行った。Communeは、設計上ドローイング・サーフェスに関連するジェスチャがカーソルを使った二次元の動きに限定されるため、ジェスチャの利用は阻害されると思われたが、設計者はスクリーン上に書かれたものを参照したり（「この点」、「ここ」等）、対応する話し合いを強調するため、カーソルによるジェスチャを頻繁に利用した。参照や強調という予想されたジェスチャの利用に加えて、設計者はカーソルによるジェスチャを、実際に面上に書かずにポイントを示すために使うことがあった。このような一時的な図解の実行と機能については、さらに研究が必要である。

(b) 用途

ジェスチャ、ドローイングとも、ドローイング・スペースが初期の遠隔セッションより広く参照された。設計者は、初期のオーディオ／ビデオ・リンク・セッションでも互いのドローイング・サーフェスを見ることができたが、書かれたものやコンセプトへの参照はこれほど行われなかった。

Communeのセッションでは、対面セッションのように、互いのドローイングに何度も参照が行われ

た。Communeでは、おぼえ書きや、いたずら書きその他無関係な書き込みも大幅に減少した。

(c) インタラクション

設計者は、同じ場所で互いに書いたものを使ってやりとりすることができ、アクションの75%近くが、参加者双方が使う書かれたものに関連したものだった。セッション中、設計者たちは何度か同時に同じ場所に書いたり差し示したりした。どのセッションでもそうだが、設計者たちは、ライティング、ドローイング、指示を頻繁かつ迅速に混合していた。

(d) リアクション

概して、設計者たちはシステムがうまくいったと述べている。

この他、新しい能力や問題も見つかった。Communeには、初期の三つの状況のどれにも不可能な二つの特長がある。まず一つは、設計者が常にドローイング・サーフェスに対して同じ向きを向いていられることである。そのため、書かれたものはどちらの設計者にとっても常に上向きとなる。これはオーディオ／ビデオ・リンク・セッションにも当てはまるが、相手の図面はビデオモニタに現れるのであって、ドローイング・サーフェスではない。二つ目は、設計者が文字通り同じ場所に同時に書き込みや指示ができることである。対面でも理論的には可能だが、実際には物理的に手が邪魔になる。

Communeの利用においては、三つの問題が特に目についた。一つは、特殊なスタイラスが、短い線からなるスケッチの書き写しや作成には利用しにくいことである。二つ目は、ドローイング・スペースが小さいため、ユーザが頻繁にページを変えなければならないことである。本論文の研究では、設計者は、当初の対面セッションでは大きな紙3枚（ビデオ・リンク・セッションで5枚、電話のみのセッションで3枚）に対して12ページを使用した。三つ目は、Communeではジェスチャが指示アクションに限られることである。対面セッションで見られる表情豊かな手や指の動きはできない。討議に関連したカーソルの動きと、無作為な手の動きも常に区別できるとは限らない。設計者がスタイラスを持たずに手のジェスチャを行う場合もあった（この場合、当然、Communeの相手側には見えない）。

8.2.6 CAVECAT

原文タイトル： Experiences in the Use of a Media Space

翻訳タイトル： Media Space の利用経験

著　者： Marilyn M. Mantei
Ronald M. Baecker
Abigail J. Sellen
William A.S. Buxton
Thomas Milligan
(Department of Computer Science, University of Toronto, Ontario, Canada)
Barry Wellman
(Department of Sociology, University of Toronto, Ontario, Canada)

出　典： CHI '91 Proceedings, April 1991, pp.203-208

Media Space とは、ビデオ、オーディオ、コンピュータを統合利用し、空間的および時間的に分散した複数の個人あるいはグループが協力して仕事することを可能にするものである。本稿では、このような Media Space の一つである CAVECAT (Computer Audio Video Enhanced Collaboration And Telepresence) について、その初期の利用経験を要約するとともに、未解決の技術的問題点、およびこの技術の心理学的、社会的影響について考察している。

(1) CAVECAT システム

CAVECAT システムは、デジタル＋オーディオ＋ビデオのネットワークによって接続された多くの高機能ワークステーションから構成される。それぞれのワークステーションは、パソコン、TV モニタ、TV カメラ、一対のスピーカ、マイクロフォンから構成されている。1 台につき 4 枚のビデオ・ボードがあり、最大 4 ヶ所までの映像を表示する。場所によっては、ビデオ・ボードは、ワークステーションの画面上に低解像度のビデオイメージを直接表示することができるので、独立したモニタは不必要である。

このシステムの心臓部は、Rank Xerox EuroPARC で開発されたIIIF サーバを参考に開発されたスィッチング・ネットワークである。オーディオおよびビデオ通信はアナログであるが、ワークステーション上に存在する IIIF サーバのソフトウェアによってデジタル的にスィッチされる。それぞれのオフィスにあるパーソナル・ワークステーションは Ethernet を介して、接続要求メッセージを IIIF サーバへ送る。また、IIIF サーバは、それぞれのオフィスのプライバシ設定を調べ、他のオフィスからのアクセス要求を許可できるかどうかを決定する。

それぞれのパーソナル・ワークステーションには、サーバ・エージェントが常駐している。それぞれのオフィスの人々は、このエージェントへのユーザ・インタフェースを介して多くのコミュニケー

ション・メタファを選択することができる。タスク指向のメタファ（例えば、会合の呼び掛け）、空間指向のメタファ（例えば、誰か他の人のオフィスへ入る）、オブジェクト指向のメタファ（例えば、自分のオフィスのマイクロフォンを切る）などである。

(2) CAVECATの予期しなかった利用方法

Media Spaceのユーザに対するインパクトを理解するために、二人の教官のオフィス、システム・プログラマのオフィス、および大学院生の作業場所にCAVECATのノードを設置し、自分自身で利用することとした。コミュニケーションのインタフェースとしては、関係するオフィスのレイアウトからなる空間的メタファを使用した。CAVECATユーザ各人のビデオ・イメージをデジタル化し、それらのミニチュア化されたイメージを画面上でそれぞれユーザが所属するオフィスの内部に置いた。これらのユーザ・イメージをある仮想オフィスから別の仮想オフィスに移動させることによって、選択したオフィスとのビジュアルなリンク、および音声リンクが確立される。

これらの利用実験によって、次のような予期せぬ利用方法が発見された。

① グループによる会議

当初の意図は、このシステムがまず第一に、それぞれのオフィスにいるひとりひとりの人間によるコミュニケーション装置として利用されることであったが、実際には、ビデオ会議の自然な需要が存在しており、このシステムはビデオ会議装置としても利用された。グループのメンバを訪問したお客があったとき、物理的に歩きまわってそのお客を他人のオフィスに連れて行って紹介するのではなく、CAVECATを利用して紹介したのである。

② 鏡としての機能

このシステムは会議の他のメンバを表示するよう設計されたのであるが、自分自身を表示することには予期しない利点が発見された。利用者は、この「鏡」機能を利用して、自分がカメラに対して正しくフレームされているかどうかを確認したのである。鏡機能は4地点会話用の分割スクリーンに自動的に組み込まれていた。

③ モニタリング機能

もう一つの予期せぬ利用法は、自分自身のオフィスに仮想的に存在させることである。自分自身を仮想的に他人のオフィスに置くためにこのメディア・リンクを利用するのではなく、自分が自分のオフィスにいないときに、自分のオフィスを見るためのウィンドウとして、このリンクを利用したのである。利用者は、だれが自分を探していたか、あるいはいつ電話が鳴ったかをモニタすることができた。また、このシステムをセキュリティ目的に利用することができた。

(3) 技術的問題点

本実験において、以下のような技術的な障害が問題となった。

① システムの応答時間の遅れ

仮想オフィス間のコネクションを実現するために、最初に開発したプロトタイプのソフトウェア・パッケージでは、接続の確立、進行中の会議への参加、あるいは会議から抜けるための時間はおよそ2秒であった。2秒の待ち時間とは非常な遠距離通信におけるネットワーク切り替え時間および衛星通信による遅れに相当し、利用者にとってはもどかしいものであった。

② 音声のレベルと雑音

話者のオフィスにおける周辺雑音は音質にとって大きな問題となった。家具や設備の置き方の違い、オフィスにいる人数、ドアが開いているかどうか、オフィスの所有者の座っている場所とスピーカやマイクロフォンとの相対的位置関係、これらすべての要因が音質をさらに劣化させる潜在的要因となった。この結果として、音のレベルは常に調整されなければならなかった。現在、各参加者に自分自身のオーディオを制御する能力を与えることによって、システムを修正しつつある。しかし、そのような調整を簡単にする方法を考えなければならないし、個人による調整がフィードバックによって全体の音質を劣化させないことを保証する方法を考える必要がある。

③ 音声の局所化

CAVECATの参加者は、共有コミュニケーションの音声が、話し手の方向からではなく、空中から聞こえるような感じがするとコメントした。複数の参加者がコミュニケートする場合、参加者が音声を局所化できないとき、だれが話しているのかわからなくなることがある。また、他者のオフィスでの電話のベルがネットワークを介して伝達されてしまうため、それとの混同によって、自分の電話が鳴っていることに気がつくのが困難になることもしばしばであった。

④ ライティングとカメラのアングル

カメラの設置場所、ライティング、部屋の背景の色、カメラと人間との距離などすべての要素が他のオフィスへ送られるイメージの大きさと品質に影響した。カメラ・アングルが悪い場合には、話者の印象が傷つけられる恐れがある。自動焦点機能のあるカメラは、オフィスの中を歩きまわる人間に対して常にズーム・インしたりズーム・アウトしたりする。これによって他のオフィスにいる参加者は、多少乗りもの酔いになったような感覚を感じる場合がある。カメラと人間の位置関係を注意深く検討し、望ましいビデオイメージを提供するための適切な制御を提供しなければならない。

(4) 心理的および社会的影響

① オフィス間および内部での会議

グループ間の会議の管理は困難な課題である。同じオフィスにいる人々が互いに感じる「存在感」

は、メディアを介してコミュニケートしている仮想オフィスにいる人々に感じるそれよりも強いからである。同じオフィスにいる人々の物理的な近さは、ビデオによる隣人よりも、物理的な実際の隣人をより強く意識させる。この事実、およびネットワークを介した音声の品質の悪さによって、別のオフィスにいる人々との会話よりも、同じオフィスの人々との会話の方が盛んになるという傾向が見られた。

しばしば、二つのタイプの会話が同時に進行した。ひとつはパブリックなもので、人々はカメラに向かって発言する。また、各オフィスにいる人々の間でプライベートな会話が進行した。これら二つのタイプの会話をコーディネートし、適切なときに、パブリックな討論を主体とすることは、難しい課題であった。

このような大きな会議でのもう一つの問題は、多くの個人の表示サイズが非常に縮小されてしまうため、インタラクションの細かな点がしばしば見えなくなってしまうことであった。表情や非言語的な身振りなどが目立たなくなってしまう。システムを介した交流は、同じ部屋に居る人々の間の交流ほどの「現実感」を失ってしまう。

② 見つめることと目を見ること

参加者たちは相手のビデオ・イメージを見ているため、TVカメラを直接見つめることをしない。したがって、互いの目を見つめることは実現されなかったのである。

通常の対面コミュニケーションにおいては、相手を見つめたり、互いに見つめあったりすることは重要な一部分である。会話の61% は相手を見つめるし、31% は互いに見つめあうと推定されている。相手を見つめることは少なくとも五つの機能を果たす。すなわち、(i) 会話の流れを整理する、(ii) コミュニケーションが聞き手によってどのように受け取られているかのフィードバックを提供する、(iii) 感情を伝える、(iv) 人間関係の性質を伝える、(v) 地位の上下関係を反映する、の五つである。

これまでに得た最善の解決策は、広角レンズを持つカメラを人物の前、上、およびモニタのすぐ上に設置することである。カメラは人物にあまり近付け過ぎてはいけない。人物を近く見せるためにはズーミングを利用する。

③ 会議参加者の地位

もう一つの興味ある観察は、地位の情報を伝達する通常の空間的および非言語的な手がかりを失わせることによって、CAVECAT が社会的な地位の上下関係を変化させたことである。

対面の会議では、通常部屋の中の着席順が人々の地位の順序を反映し、地位の高い人ほど中央よりあるいは上座に席を占める。

CAVECAT の設計は、意図しないで独自の社会的な地位の手がかりを導入していた。4 人の参加者による会議では、CAVECAT は参加者のイメージを 2 × 2 の枠の中にデタラメに配置する。CAVECAT は、また、会議を招集した人を基準にして会議のためのビデオ・イメージを構成した。つまり、短い

休会の後別の人が会議再開を要求した場合、前とは異なったイメージの配置になった。これは、参加者にとって非常に気になることであった。それはあたかも、一旦出たあと部屋へ戻ったとき、みんながテーブルの新しい位置に付かなければならないようなものであった。

④ 会議のコーディネーション

このシステムにおいて、地位への伝統的な手がかりが失われたことと、新しい手がかりが生成されているという観察は、討論の制御という、もっと一般的な問題に関係している。重要な手がかりが失われたり、わかりにくくなったりすると、仲介役が発言順序やグループ意思決定のプロセスをコントロールする必要性が大きくなる。例えば、会話で主導権を握りたいと考えている人々は、しばしば発言したいという希望を示すため上体を伸ばすものである。この手がかりはビデオでは発見しにくい。

⑤ イメージのサイズと個人への影響

参加者が会話において効果的にふるまうことができるかどうか、および、各参加者が他者からどのように認識されるかは、一部分、ビデオ・イメージのサイズによって決定されるようである。大きなイメージで表示される参加者は、より大きな影響を討論に与えるように思われた。小さなイメージの参加者は、会話から離れており、効果的にふるまっていないと思われた。

ビデオ・イメージのサイズは、四つの要因によって決定される。すなわち、(i) モニタの画面サイズ、(ii) 見る人と TV モニタとの距離、(iii) 人間とカメラとの距離、および(iv) カメラのズームの設定である。しばしば参加者のイメージのサイズは異なったままであった。これらの変数は、めったに調整されないからである。

⑥ ビデオ・イメージと社会的距離

イメージのサイズと角度は、人々が自分と他の参加者との社会的関係をどのように認識しているかということと関係がある。不適切なイメージ・サイズは、人々が会話において過度に個人的であるか、あるいは過度に非個人的であるという印象を与えた。

お互いの間の距離が、その人間関係にとって不適切であると知覚されたとき、人々はたちまち居心地悪く感じることは、よく知られた事実である。接近し過ぎているならば、人々は自分の空間に侵入されたように感じる。離れ過ぎていても、居心地悪く感じるのである。

Media Space の場合には、問題なのは人々の間の「知覚された」距離である。物理的な距離よりも仮想的な距離である。観察によれば、一般的に言って、ビデオ・イメージは個人性が低く、控えめに受け取られる。Media Space について、もう一つ通常と異なっていることは、、コミュニケーションに参加しているグループのメンバのそれぞれについて、異なった人間間距離が同時に与えられているということである。これは、実世界における物理的な距離の場合と異なっている。実世界においては、人々の間の物理的距離は、ある意味ではネゴシエーションされ、共有されるのである。CAVECAT では、参加者の個人的な空間は、侵害者がそれと意識しないままに侵害される可能性がある。

⑦ プライバシと監視

自分が他のオフィスに接続されたときそれを「知る」こと、およびコネクションを拒否することができることは、非常に重要な必須機能であることは非常に初期の段階から認識されていた。最初のMedia Spaceには、他者が突然自分のオフィスに入り込んできたことを示すフィードバック機能が不足していた。さらに、IIIFサーバにはプライバシ保護機能があるにはあったのであるが、使いにくいものであった。適切なフィードバックを提供するための良いアプローチの一つは、人の声ではないオーディオ・キューを利用することである。

8.2.7 CRUISER

原文タイトル： Design of a Multi-Media Vehicle for Social Browsing

翻訳タイトル： 散策会談のためのマルチメディア装置の設計

著　者： Robert W. Root
(Bell Communications Research, Morristown, NJ, USA)

出　典： CSCW '88 Processings, September 1988, pp.25-38

本稿では、分散化された共同作業を支援するための、コンピュータを利用した通信技術への新しいアプローチについて述べられている。ここでは、事前に計画されたのではないインフォーマルな社会的インタラクションを可能とするツールに焦点が置かれている。開発されたCRUISERと呼ばれるシステムは、マルチメディア通信チャネルを介して、他者への直接的アクセスを低コストで提供する「社会的インタフェース」であり、これまでの研究論文および仕事場についての三つの基本的コンセプト（散策会談、仮想的作業場、インタラクション・プロトコル）に基づいて設計されている。これらの設計要素を利用して、新しいシステム・コンセプトが提案され、さらに、デスクトップ・ワークステーションを介して社会的インタラクションを自動化したことのCSCWにとっての意味が検討されている。

(1) 「散策会談 (social browsing)」の重要性

職場における人間関係と共同作業の形成と保持の鍵となるのは、個人の間の良好なコミュニケーションである。オフィスの研究によれば、オフィスワーカの活動の65%は個人間のコミュニケーションを含んでおり、32%は直接対面によるインタラクションとなっている。物理的な近接性は、個人間のコミュニケーションへ与えるその効果を介して、人間関係の形成と保持にとって重要である。しかし、分散業務のもっとも顕著な特性は、関係者が物理的に離れているという事実である。この物理的な分離の与える影響を克服するために、ユーザはコンピュータによって仲介されたコミュニケーションを隣のオフィスに歩み入るのと同じくらい効果的で容易なものにするようなツールを必要とする。

接近性の効果は、構造的要素とダイナミックな要素の組合せとして現れる。構造的要素は、建築設計的および組織的な考察から引き出される。それらは、環境の物理的な特性および協同作業の相手となる人々を定義することによって、社会的インタラクションの舞台を設定する。ダイナミックな要素とは、単に、人々は日常の活動の中で職場の中を動きまわるという事実である。人々は昼食をとりに外出したり、コーヒーを飲んだり、仲間と雑談したり、コピーを取ったりする。これらオフィス内移動の過程において、彼らは同様の用事をしている他人に出会ったり、それぞれのオフィスで勤務して

いる人たちの横を通り過ぎる。職場の中を歩きまわることは、それぞれの場所での仕事仲間との予定された、あるいは予定されていないインタラクションの機会を頻繁に提供する。ここでは、人間のオフィス内での動きに基づいて発生する、このインフォーマルかつ直接的な社会的インタラクションのダイナミックなプロセスを「散策会談 (social browsing)」と呼ぶ。散策会談は、職場内での社会的関係を形成し保持するための基本的メカニズムであると考える。

(2) CRUISER の概要

CRUISER を設計するに当たっての前提は、次のとおりである。

- デスクトップの完全動画ビデオ・コミュニケーション
- 高品質な全二重のオーディオ設備
- ローカルなコンピュータによって制御された交換マルチメディア・ネットワーク
- ビデオ・イメージとコンピュータによって生成されたグラフィックスの統合が利用可能

実際においては、プロトタイプ・システムのユーザは、本質的にはビデオ電話の設備、つまり、カメラを同軸的に装備されたビデオモニタ、モニタ上中央に設置されたハンドフリーなオーデイオ設備、および交換マルチポイント・マルチメディア・ネットワークを介したオーディオ・リンクとビデオ・リンクに対するコンピュータ制御を持っていることを仮定する。

① 仮想的作業場

マルチメディア・ネットワークによって、物理的には同じ場所にいない人々も含むよう、社会的インタラクションの範囲を拡大することができる。本質的には、それによって、高忠実度の仮想的環境を創造することができるのである。その環境の住人は、ネットワークへの加入者に限られる。この仮想的世界は、柔軟性に富みダイナミックである。物理的な近接性という制約条件は取り払われ、ユーザは社会的な環境を、建築設計上の制約ではなく、人間関係と仕事上の任務を中心に組織することができるようになる。

CRUISER は、仮想的な社会環境へのインタフェースのためのモデルであり、インフォーマルで自然発生的かつカジュアルな社会的インタラクションを支援するために設計されている。このネットワークは、他者へのマルチメディア接続（社会的リンク）を提供する。インタフェースは、いくつかの仮想的な通路を提供し、ユーザはそこを自由に歩きまわることができる。これらの通路には、ユーザが選択した、あるいはシステムがランダムに選択した人々が歩いていると考える。ユーザは仮想的空間を歩きまわり、自然発生的な他者との出会いを経験することができる。これは物理的な現実世界において、コーヒー・ルームへの道すがらだれかに出会うのとまったく同様に行われる。

CRUISER システムのユーザ・インタフェースは、ブラウジング・インタフェースとビデオ・ウィンドウという二つの基本的要素を持つ。ブラウジング・インタフェースは、CRUISER ソフトウェアとの

インタラクションを可能とし、ビデオ・ウィンドウは遠隔地のカメラからの映像を表示する。実際のインタラクションは多くの形を取り得るが、例として仮想的世界の地図が、ブラウジング・インタフェースによってユーザのワークステーション上に、二次元のグラフィックスとして表示されていると仮定しよう。ユーザは、マウスあるいはその他のポインティング装置を使って、ブラウジング・インタフェースによるインタラクションを行い、場所や機能を選択する。この地図は、仮想的な通路を表現するから、それは任意の便宜的なレイアウト、例えば円周形でも良い。この地図が現在のコミュニティを表現する。その中には、プリンタなどの装置があってよいし、共通エリアや会議室のような「実際の」場所があってもよい。また、他の人々（コミュニティのメンバ）が含まれる。ユーザはマウスを利用して経路を選択したり、特定の目的地を選択することによって、この仮想的世界の中を旅行するのである。

② 「社会的交流インタフェース」の要素

CRUISER システムは、その力の大部分を二つのアイデアから得ている。まず、ユーザは社会的な出会いを求めて仮想世界を歩きまわるための多様なメカニズムを持つことができる。つぎに、仮想的世界は物理的世界からは独立しており、ユーザのニーズに応じて組織し、その中の住人を選択することができる。

(a) 仮想世界を歩きまわるためのメカニズム

自然発生的な社会的交流を求めて歩きまわる（social browsing）というメタファは、実世界においても歩きまわるという行動が、自然発生的な社会的インタラクションの基礎になっているという考え方に基づいている。仮想的世界において、計画されたインタラクション、自然発生的なインタラクションの両方を支援するためには、少なくとも三つの異なったメカニズム（ジャンプ、計画された経路、ランダム・ウォーク）が必要である。ジャンプとは、ビデオ電話をかけるのと同じことである。ユーザが目的の相手をマウスで指定すると、その場所のイメージがビデオ・ウィンドウに表示される。これは直接的な計画された動きのためのメカニズムである。このアイデアを拡張して、CRUISER システムでは、仮想的世界における経路というものを設定する。経路とは、いくつかの場所（オフィス、共通エリア等）およびそれらをアクセスする順序の集合である。つまり経路は、ユーザが仮想的通路をクルーズするときに訪問する場所の順序を表わしている。これが自然発生的なインタラクションを生成するための基本的メカニズムである。経路をアクセスする二つの方法を CRUISER は提供している。計画された経路においては、経路はユーザによって事前に定義されており、ユーザがそのオプションを選択する度に CRUISER はその経路を生成する。ランダム・ウォークでは、CRUISER は、仮想世界において計算されるある種の関数によって経路を作成する。関数には、完全にランダムな方法、共通の関心に基づく方法、あるいは、ある特定の人がオフィスに在席しているかどうか等なんらかの条件に基づく方法などがある。これらの関数はユーザが定義する場合もあるし、ソフトウェアにまかせる

場合もある。どちらにしても、それらの目的は、社会的インタラクションのリンクをカジュアルな形でアクセスすることであり、可能な多くのインタラクションに対して予測のつかない機会を提供することである。

経路とは、通路を歩きながら多くのオフィスのドアを通り過ぎるようなものである。また、たまたま立寄った形での出会いのサポートも考えられる。つまり、通路で進行中の会話（立ち話）に第三の人物が一寸立ち寄る形で参加することである。

(b) 視点

ビジタが仮想的通路をクルーズしているときに、事前に設定された経路をたどっているならば、彼は経路に従って、ドアの開いているオフィスを一つ一つ自動的に「覗き込む」。ビジタが開いているウィンドウを覗くと、そのオフィスのカメラが撮影したイメージが見えるし、内部のマイクロフォンが捕らえた音声が聞こえる。クルーズする人がオフィスからオフィスへ動くにつれ、映像は変化する。まず移動しているというイメージ（つまり、仮想的通路のグラフィカルなイメージ）が与えられる。つぎに、経路上の次の場所の映像が与えられる。一方で、オフィスの住人のウィンドウは、仮想的通路に向かって開かれている。ビジタがオフィスを覗き込むと、そのオフィスの住人には、ビジタのオフィスのカメラからのビデオ・イメージが見える。そして、クルーズする人が通り掛けに言うことはすべて聞くことができる。通路が無人である場合には、オフィスの住人は通路のニュートラルなイメージを見るだけである。このような設計は、日常のオフィス生活の観察から得られた社会的対称性の哲学から導出されたものである。実世界においては、自分が見られることなく、自分だけが相手を見ることは一般的に言って不可能なことである。物理的世界のこのような特性を保存することによって、コンピュータによって仲介されたインタラクションの世界に、ある種の社交的な礼儀正しさを導入でき、人が勝手に他者から観察されないことを補償することによってプライバシ保護の要素を導入できるのである。

(c) 仮想的世界のイメージ

CRUISER システムにおいては、他者は社会的なリンクとして示される。一人のユーザと他のあるユーザとの間の、あるいはすべての他のユーザとの距離は、正に1単位、つまり1ノードである。仮想的な社会空間のすべての住人は、機能的には正確に同一の距離しか隔たれていない。この事実は二つの重要な意味を持つ。まず、空間的近接性の概念は仮想的空間においては意味がなくなる。すべてのユーザは同一の近接性を持つ。第二には、他者へのすべてのリンクはコンピュータによって制御されているから、仮想的空間の部分世界、つまりはビューを無限の多様性をもって扱うことができる。つまり、特定の目的のための部分世界（コミュニティ）を作成することができる。例えば、組織のメンバ、社交クラブ、作業グループなどである。コミュニティという考え方は、社会的背景を形成し、散策会談のための参加者のサブセットを定義するうえで有用である。

③ インタラクションのプロトコル

CRUISER は、実世界とのアナロジーによって設計されている。社会的なインタラクションを開始するための強力なツールを提供するとともに、仕事場での社交上のニュアンスを保つことを試みる。設計に当たって最も重視したことの一つは、オフィスでのプライバシと社会的インタラクションを管理するための適切なインタフェース機能を提供することであった。ここでは、仮想的職場の住人へのアクセスを規制するための、一組のプロトコルを提案する。このプロトコルは、職場での規範についての観察に基づいている。

現実の世界においては、視覚的・音声的な手がかりを利用して、訪問したい相手の状況を把握することができる。このような潜在的な手がかりを知覚するのは、われわれが社会的環境において経験をつんでいるからであり、われわれ自身が他の人の仕事中に割り込んで良いものかどうかを常に考えるからである。また、実世界でのオフィス住人が自分の面会可能性について明らさまに制御を行おうとする場合には、オフィスのドアを閉めたり、電話に出なかったりすることによって割込みを排除することができる。これらの行動は、明確なシグナルとなるとともに、オフィスに流入流出する情報の流れに対する物理的な障壁としての機能も果たす。これらの手がかり、シグナルおよび障壁は、一纏まりのプロトコルを構成している。われわれはこれらのプロトコルを利用して、職場の社会的構造を定義したり強化しているのである。

一方、仮想的世界においては、手がかり、シグナルおよび障壁は独立に操作可能であり、オフィスの住人に割り込んで良いかどうかや、オフィスと通路の間の情報の流れに対して、広い範囲の制御能力を提供する。例えば、ビデオは ON にしておき、音声の方は OFF にしておくなどの仮想的世界独自のプロトコル条件を提供することが可能である。さらに、システムはビジタが誰であるかを知っており、また、彼らのユーザ・プロフィルからのなんらかの情報を持っていると仮定するならば、ビジタあるいはビジタのクラスごとの個別の割込み許可状態を設定することも可能である。例えば、組織上の地位をもとにしてデフォルトの割込み特権を指定しても良いし、あるいは、ある友人には普通よりも高い割込み特権を与えても良い。CRUISER システムの仮想世界においては、ユーザが割込み許可（つまりプライバシ）特権状態の階層に従って、インタラクションを制御できるようにすることを提案する。それぞれの状態には、あらかじめ定義した情報チャネルあるいは通信チャネルの構成、割込みレベル、およびオーバライドのための必要条件が付属しており、これがオフィスの住人およびビジタの双方に影響する。ユーザは、自分の利用可能な変数を用いて、新しい状態を定義することができる。しかし、ユーザ間の規範の整合性を保存するためには、システムの提供する標準的状態を変更することはできない。情報／通信チャネルの利用は、オフィス住人にとってもビジタにとっても対称的である。住人はビジタがチャネルを使えないようにしておいたままで、しかもチャネルをオープンにしてオフィスの外の世界をモニタすることはできない。

これらの状態は、ビデオ・ウィンドウ上のグラフィックス・オーバレイ（例えば仮想的なブラインドなど）を利用して、ビジタに対して知らされる。例えば、もし住人が「ビジタ受入れ可能」であるならば、ウィンドウはオープンであり、クルージングする人は、住人のオフィスの中の音を聞くことができるし、覗き込むことができる。このウィンドウは完全に閉めてしまったり、部分的に閉じることもできる。閉じられたウィンドウと切断された通信チャネルは、シグナルとしての役割も障壁としての役割も果たすことができる。

プロトコルに加えて、CRUISER システムは、インタラクションを求めるためのドアベルと、外出中の人々を扱うためのメカニズムを提供している。住人のビデオ・ウィンドウを覗き込むだけで、その人がいるかいないか、自分と接触可能であるかどうかをチェックすることはできる。しかし、プライバシ上の理由から、CRUISER システムでは、住人の活動についての連続的なチェックができるような機能は、表立ってサポートしていない。人々は仕事の上で動きまわるものであるから、クルージングする人がオフィスを訪ねてもその住人は外出中であるかもしれない。現実世界でのこのような状況では、われわれは椅子の上や端末の画面と言った目につきやすいところにメモを残す。CRUISER システムの仮想的世界では、ビジタはオフィス住人のインタフェース・ウィンドウに、デジタル版の「メモ用紙」を貼り付けておくことができる。同様に、オフィスの住人もインタラクション・ウィンドウにビジタに宛てたメモを残しておくことができる。自動的な「お返し訪問」機能を提供することによって、メモを残すことの有用性を拡張することができる。メモにマウスを置くことによって、住人は直接的にメモを残したビジタのオフィスへジャンプすることができる。これらの機能は自然発生的なインタラクションを支援する上で有用である。

(3) 社会的インタラクションと仕事との統合

CRUISER は、職場での社会的インタラクションを自動化するためのツールである。このシステムが設置されると、社会的インタラクションは一般的なコンピュータ環境に機能的に統合される。これによってわれわれは、社会的インタラクションをまったく新しい観点から見ることができるようになる。つまり、社会的インタラクションは、職場のリソースの一つであり、仕事を進めるためのツールであり、作業が実行される背景の重要な要素と見ることができるのである。このような統合は、多くの形を取り得る。

基本的な統合コンセプトは、CRUISER のリンクとアプリケーション・プログラムの間の結び付けである。リンクは、個人、経路、あるいはコミュニティに対応する。一つのリンクがあるアプリケーションに結び付けられると、そのリンクは、ユーザが事前に記述しておいた条件に従って、対応するアプリケーションによって自動的に起動されるようにできる。また、リンクとアプリケーションとの間の結び付きは、そのアプリケーション環境の一般的な背景の一部分として保持しておくこともできる。

第二のコンセプトは、適応である。社会的関係が静的なものであることはめったにない。それらが共同作業の一部分である場合には特にそうである。CRUISER にメモリを与えておけば、そのユーザの間のコミュニケーションのパターンのモデルを作るであろう。コミュニケーション・パターンを時間軸の上で比較し、結び付きのモデルを利用状況に適応させることによって、CRUISER システムは、社会的リンクの利用を、ある作業の領域におけるあるユーザ集合の中における活動に合わせて調整する、ダイナミックなシステムに変身するであろう。

また、リンクとアプリケーションとの結び付きというコンセプトを利用して、CRUISER において偶然的な出会いといったインタラクションを可能にすることができる。例えば、ユーザは、計画された経路あるいはランダム・ウォークを、文書の印刷のような、ある典型的なワークステーションでの活動に結び付けることができる。このようにしたならば、文書を印刷するという活動は、仮想的な散歩に出るための機会としての役割を果たすこととなる。文書がプリンタに送られるときには常に、CRUISER は指定された経路を起動する。ときどき、CRUISER は計画されたインタラクションに予測不可能な要素を取入れ、ビジタが最終目的地へ行く途中で彼をどこか別のところへ連れて行く。現実世界を真に模擬するためには、ビジタはときどきは会話ノードに衝突するようにすべきであろう。これは現実世界では、人々が通りすがりに通路での会話にぶつかることがあるようなものである。結び付けの可能性はほとんど無限である。重要な点は、CRUISER の社会的インタフェースが、電子的作業環境の社会的インタラクションへの拡張をサポートし、コンピュータによって仲介されたコミュニケーションの新しい機能と力を増大させるためのメカニズムを提供していることである。

8.2.8 VideoWhiteboard

原文タイトル： VideoWhiteboard: Video Shadows to Support Remote Collaboration

翻訳タイトル： VideoWhiteboard：遠隔協同作業を支援するためのビデオ影絵

著　者： John C. Tang
Scott L. Minneman
(System Sciences Laboratory, Xerox Palo Alto Research Center, Palo Alto, California, USA)

出　典： CHI '91 Proceedings, April 1991, pp.315-322

本稿では、VideoDrawという協同図面作成用プロトタイプ・ツールと、これを改良して開発されたVideoWhiteboard（遠隔協同図面作成を支援するためのプロトタイプ・ツール）について紹介されている。まず、VideoDrawについてレビューを行い、VideoDrawについての経験がVideoWhiteboardの開発に導いた経過が説明され、その後VideoWhiteboardとそれを利用する人々の反応について述べられる。最後にVideoWhiteboardのこの最初の利用経験で観察されたいくつかの問題点について、プロトタイプの設計および協同作業による図面作成活動一般の両面から検討されている。

(1) VideoDrawの概要

VideoDrawは、遠隔地にいる二人の協同作業者がビデオ・スケッチパッドを共有できるようにしたプロトタイプ・ツールである。このツールは、ビデオ表示画面に向けられたカメラを接続している。ユーザはビデオ・ディスプレイの画面に対して絵を描き（消去可能なホワイトボード用マーカを使用）、これらの図形とそれにともなう腕の動きがカメラによって撮影され、他方の画面に表示される。このメカニズムによって、合成された共有図面作成領域が作成され、その上で協同作業者たちは互いの描いた図形、それに関係する相手の腕の動きを観察することができる。

VideoDrawを利用する人々の観察によって、彼らはしばしば腕のジェスチャを使ってシミュレーションをしたり、相互の間のインタラクションの効果を上げていることが明らかとなった。また、しばしば、これらの身振りは図面作成スペース中の関係したスケッチを参照しながら行われている。VideoDrawを利用することによって、遠隔地の協同作業者たちは、（結果としての図面を見るだけでなく）図面の作成およびそれへの参照のプロセスを見ることができる。これは、協同図面作成作業において重要な資源である。協同作業者たちは、ときどき、同時に同じ場所へ書き込むためにVideoDrawを利用することさえあった。これによって単一のドローイング・サーフェスでは不可能なインタラクションが可能になったのである。さらに、VideoDrawは、遠隔地にいる協同作業者たちの間に、共存感覚を提供するための新しい方法について示唆を与えた。一緒に作業している相手の腕の

ビデオ・イメージがドローイング・サーフェスの上に見えることは、例えばコンピュータ化されたスケッチパッドの上でカーソルが動く場合とは別の種類の存在感覚を与えるのである。

VideoDraw の利用において観察された限界は次のとおりである。

ビデオ表示画面が相対的に小さい（20 インチ対角）ので、画面一杯にしても、描くことの出来るテキストとグラフィックスの量が制限される。ディスプレイの蛍光体レイヤ（相手の図形が表示される部分）とディスプレイのガラス面（自分の図形が描かれる部分）との間のガラスの厚みによる視差によって、双方のユーザの作成した図形を適切に位置付けることが困難となった。ユーザは自分の画面上に自分が作成した図形を消去することができるのみであり、ときどき相手の図形を消去するよう相手に要求する必要があった。上向きに置かれた CRT ディスプレイを相手に作業し、しかも自分の頭でオーバヘッドカメラの邪魔をしないようにするのは、居心地の悪いものであったし、視差の問題をさらに難しくした。これらの観察結果は、VideoDraw の設計を修正することを示唆するとともに、VideoWhiteboard の設計へ導いたのである。

(2) VideoWhiteboard の概要

VideoWhiteboard は、遠隔地の間で共有される大領域の図面作成スペースを提供するビデオベースのプロトタイプ・ツールである。それぞれのサイトには、壁掛け型の背面投射型スクリーン（約 140 cm × 180 cm）、ビデオカメラ、ビデオプロジェクタ、および適当なオーディオ装置が設置されている。ユーザは、スクリーンの前面のなめらかな表面に、標準の乾式消去可能ホワイトボード・マーカを使って絵を描く。ビデオカメラは、ユーザから見てスクリーンとは反対側に設置されており、画面全体のイメージを撮影し、他のサイトのビデオ・プロジェクタに送信する。送信されたイメージは、スクリーンに投影される。それぞれのユーザがスクリーン上に図を描くと、それらの図形は、カメラによってイメージされ他方のスクリーンに投影されるのである。図形のイメージのほかに、カメラは遠隔地の協同作業者の影絵を送信する。協同作業者は、自分で書いた図形と「ビデオで投影された」図形の合成像、および遠隔地の協同作業相手の身振りや行動の影絵を見ることになる。音声のコネクションによって、作業者同士は互いに会話することができる。ユーザは、ホワイトボードを共有して協同作業している場合とまったく同様に、VideoWhiteboard のスクリーンに対して、書き込み、描き、消去し、ジェスチャを行うことができる。

それぞれのサイトのユーザは、表示画面に対して正しい方向感覚を共有する。「右」と「左」は、どちらのサイトにいる人にとっても同じ意味を持つ。カメラはユーザから見てスクリーンと反対側に設置されているため、実際には右左が逆転されたイメージが撮影されるのである。ビデオ・プロジェクタは、実際にはスクリーンの裏側に向かって投影しているのであるが、フロント・プロジェクション・モードで投影することによって、この鏡像は補正される。

VideoWhiteboard においては、図形を描いたり、影絵を投影したりするための入力スクリーンは、遠隔地の協同作業者の図形や影絵を投影するための出力スクリーンと同一である。したがって、ユーザは、遠隔地の協同作業者の描いた図形の上に直接自分の図形を追加して描いたり、ジェスチャを加えることができる。

(3) VideoWhiteboard と VideoDraw の比較

VideoDraw と同様に、VideoWhiteboard は、それぞれのユーザが図面作成スペースを共有し、その共有スペースにおいて、自然な方法で描き、身振りし、インタラクトすることを可能にする。それぞれのユーザに対する図面作成スペースの見え方は共通であり、図面作成スペースの中の対象に対する口語的表現での参照が意味を持つ（例えば、「これ」「ここの」など）。ユーザは遠隔地にいる協同作業者が作成したスケッチを直接補正したり、インタラクトすることができる。彼らはスケッチに対してジェスチャし、遠隔地の相手は、参照しているスケッチに対するそのジェスチャを理解することができる。遠隔サイトのユーザは、図面作成スペースに対して同時並行的にアクセスすることさえできるので、一度に同一の領域に対して複数の人間が作業することが可能である。異なったサイトにいる二人の協同作業者の影絵のイメージは互いにスーパインポーズされる。

VideoDraw とは異なって、VideoWhiteboard は、大きなスクリーンを持っており、ユーザはより多くの図面作成領域を利用できる。この大画面によって、同じサイトでも、複数の作業者が作業に参加することが可能になる。VideoDraw では、ジェスチャのフルカラーの画像が表示されたのに対して、VideoWhiteboard は、遠隔地の協同作業者の影絵を見せる。VideoDraw では主として腕の動きを伝達したのに対して、VideoWhiteboard は、上半身のジェスチャを伝達する。VideoWhiteboard では、カメラとユーザはスクリーンの反対側に位置しており、ユーザの頭部がカメラの画面撮影を妨害することのある VideoDraw の問題点を解決している。VideoWhiteboard では、視差の問題も軽減されている。VideoDraw における視差の問題点は、軸から外れたオーバヘッド・カメラから画面作成表面を撮影しなければならないことによってより困難なものとなっていたのである。このようにして、VideoWhiteboard は、VideoDraw における限界のいくつかを解決したのである。

(4) VideoWhiteboard の利用状況の観察

同一ビルディングの二つの部屋に設置された VideoWhiteboard を二人の協同作業者が約 1 時間にわたって遠隔作業するのを 2 回観察した結果、次のような観察結果が得られた。

① VideoWhiteboard の利用における特徴

VideoWhiteboard の利用に対する最初の共通的反応は、遠隔地の協同作業者が、遠隔のサイトではなく、スクリーンの反対側にいるように感じるというものである。遠隔地の相手の影絵をスクリーンで

見るというビジュアルな効果は、いく人かのユーザにとっては、自分が反透明なガラスのスクリーンを介して相手に話しかけているような印象を与えるのである。あるユーザは、相手の言うことを明瞭に聴くことができなかったので、耳をスクリーンに対してそばだて、もう一度言って下さいと相手に呼び掛けた。その音声が相手に届かなかったにもかかわらず、その身振りが目的の反応を相手から引き出し、相手はコメントをもう一度大声で繰り返したのである。

半透明のスクリーンの反対側にいる相手とインタラクトしているという印象は、本当ではないのにかかわらず、この錯覚が相手とインタラクトするためのユーザの能力の邪魔をすることはほとんどない。ユーザは、自分が遠隔地の相手とスクリーン上の図形を共有しているのであることを、すばやく認識する。また、彼らはスクリーンの上に投影される相手の影絵をそれとして認識することができる。

VideoWhiteboard についての興味ある特徴の一つは、遠隔地の協同作業者の影絵が、図形やスケッチの表示されたスクリーン・イメージの上に直接スーパーインポーズされることである。この機構によって、注意の分散が大きく軽減される。他の協同作業支援システムにおいては、参加者は、相手を見るか、作業領域を見るかを常に選択しなければならないのである。ホワイトボードを囲んでの直接対面によるインタラクションの場合には、仲間の作業者はホワイトボードに向かって図面を描いている自分の横あるいは後にいる。VideoDraw のプロトタイプを利用する場合には、協同作業者は、うつむいて図面を見るか、顔を上げて相手を見るかを交互に繰り返す必要があった。VideoWhiteboard は、図面作成領域と遠隔地にいる相手の影絵を同じ観察角度に置くので、相手の動作と図面作成領域に同時に注意を向けることを可能にする「ヘッドアップ・ディスプレイ」効果が提供される。

VideoWhiteboard は上半身全体のジェスチャを伝えるのであるから、ある意味では、VideoWhiteboard は、腕のジェスチャを伝達するという VideoDraw の能力をさらに拡大したものである。VideoWhiteboard は両腕を含む大きなジェスチャや、人々がインタラクションにおいて相手の反応を求めたり、反応を表明する際に自然に使っているボディランゲージ（肩をすくめるなど）さえも伝達できる。

VideoWhiteboard は、フルカラー・イメージを表示する VideoDraw のようには三次元感覚を与えないが、スクリーンの周囲のライティングによって、投影された影絵にシェードの違いをつけ、物体がどの程度スクリーンに近づいているかの感覚を与えることができる。したがって、ユーザは、投影された影絵の密度および鮮明さによって、遠隔サイトにいる協同作業相手、あるいはその他の物体がどの程度スクリーンに近いかを知ることができる。

② VideoWhiteboardの利用における限界

VideoWhiteboard の利用におけるいくつかの限界も観察された。影絵はある種のジェスチャ情報を伝達するには効果的であるが、フルカラーのビデオ・イメージの持つ豊かさを欠いていることは、明らかである。特に、影絵では、遠隔地に居る協同作業者間のアイコンタクトは不可能である。遠隔地の相手の影絵だけを見ていて、長時間のインタラクションが可能であるかどうかは、不明である。

影絵によるマスク効果は、一つのサイトに二人以上の協同作業者がいる場合に特に問題となる。これはVideoWhiteboardのインフォーマルな利用において、何回か実際に発生したことである。複数の遠隔協同作業者とインタラクトし、しかも彼らが影絵でしか見えない場合には、その判別がしばしば困難となる。幾人かのユーザは、このような状況においてはだれとインタラクトしているのか分かりにくいため居心地悪く感じたと報告した。この問題は、ステレオ・オーディオ装置を利用し、どの声がどの影絵に対応しているかについての位置的手がかりを提供することによって軽減すると考えられる。

ユーザが実際に行うことと、遠隔地の協同作業者が投影された影絵によって見ることができるものとの間の本質的な非対称性は、インタラクションにおけるいくつかの問題を引き起こす可能性がある。一つの位置を正確に差し示す腕のジェスチャ（例えばポインテイング）と、スクリーンから多少離れたところで行われるあいまいなジェスチャ（例えば、首肯くこと）は、認識することが困難である場合がありえる。同様に、もし、ユーザがスクリーンから十分離れたところに立ち、認識可能な影絵を投影しなかったならば、遠隔地の協同作業者は、相手と接触できなくなったように感じるかもしれない。影絵は大部分スクリーンの背後に置かれた光学機器によって生成される人工的な現象であるから、ユーザは、自分がどのような影絵を遠隔地の協同作業者に投影しているのかについては、視覚的フィードバックを持たないのである。この点については、VideoWhiteboardを介したインタラクションを保持するため、ユーザがいろいろの方法でこの潜在的問題を補っている様子が観察された（例えば、大げさなジェスチャ、スクリーンの近くに居て互いに常に影絵を見ることができるようにするなど）。

VideoWhiteboardには、大きなスクリーン領域があるが、使用する特定のビデオ技術の解像度による制約は存在している（プロトタイプにおいては、およそ330×240本のテレビ走査線である）。したがって、ユーザは、テキストとグラフィックスを解読可能にするため、やや大きめに書く必要があった。また、ビデオ投影技術は、フリッカ現象を起こす傾向がある。これは、このシステムの場合のように接近した位置で見る場合（すなわち、スクリーン上に描く場合）には特に問題である。それぞれのカメラによって撮影される領域は、それぞれのビデオプロジェクタによって投影される領域と正確に一致していなければならないから、VideoWhiteboardシステムを構成するには、光学機器の注意深い調整が必要である。スーパーインポーズされたイメージの光学的調整は、スクリーンの中央でのみ最適化されており、スクリーンの端に行くに従って段階的に劣化する。

VideoDrawの場合と同じく、ユーザは、自分のサイトのローカルなVideoWhiteboardスクリーンに描かれた図形を消去することができるのみであり、相手側のスクリーンに描かれた図形を消すことはできない。また、ユーザは、スクリーン上のイメージを保存、検索、印刷するための便利な手段を与えられていない。また、以前に作成したイメージに対するアクセス手段も与えられていない。

8.3 グループ意思決定支援システム

8.3.1 GDSS全般

原文タイトル： Computer-Based Systems for Cooperative Work and Group Decision Making

翻訳タイトル： 協調作業およびグループ意思決定のためのコンピュータ利用システム

著　者： Kenneth L. Kraemer
John Leslie King
(University of California, Irvine, California, USA)

出　典： ACM Computing Surveys, Vol.20, No.2, June 1988, pp.115-146

本稿では、グループが作業する際のニーズに合うように作られた「グループ意思決定支援システム（GDSS）」についてレヴューされ、これらのシステムの現在までの導入結果について評価されている。まずGDSSの基本的概念がレヴューされ、米国において現在稼働中あるいは開発中のシステム例が示され、また、現状におけるGDSSの導入結果が評価される。さらに、意思決定を始めとして、コミュニケーションや情報処理を含むグループ活動を支援するGDSSの今後の発展について述べられている。

(1) GDSSの定義づけ

さまざまな論文において、GDSSを構成するものは何かという点についての一致した見解は、ほとんど見つけられない。DeSanctisとGallupeは、GDSSを「ひとつのグループとして一緒に作業を行っている意思決定者が、込み入った問題を解決する際に手助けとなる、対話形式のコンピュータ利用システム」と定義した。この定義は、グループ活動の手助けとなる方法の発展のために、隠されていたニーズを理解するという点で、最初の適切な定義であると言える。

Huberは、ニーズとして次のようなジレンマがあることを示した。意思決定者にとっては、検討すべき多くの問題があって長時間の会議が増える一方であるが、彼らには他にも差し迫った問題があるため、会議に出席する暇が無い。このジレンマを解決するためには、会議をより効率的に行うことであり、このためのキーコンセプトがGDSSなのである。この方法は、決定内容の質を低下させることなく決定のスピードアップを図り、さらに決定内容の質を向上させることまで期待している。

GDSSは、このような目的をどのようにして達成できるのか。Huberは、論文中で次のように簡潔に述べている。

(a) グループ意思決定を効果的に行うためには、出席者が決定に満足し、後々出席者がうまく一緒に仕事ができるように、状況のニーズに合致することが必要である。

(b) 討論が特定の少数者によって仕切られてしまうことや、下位のメンバが上位のメンバのいいなりになってしまうこと、グループが各メンバの意向・考えを統制してしまうこと、さらに、普段からメ

ンバ間のコミュニケーションがうまくいっていないことや、問題点を究明し別の策を講じる時間が不十分であることなどが原因となって、グループ意思決定の生産性が落ちてしまう。

(c) GDSS はこのような問題に対して、次のような支援を行うことができる。すなわち、出席者一人一人が PC を持ち、全員が一度に見ることのできるディスプレイスクリーンを備え、データベースにアクセスしたり、スクリーンを通してグループリーダとコミュニケーションを行ったり、ワープロソフトを使用したり、あるいはグループ共用のデータやグラフィクスにアクセスしたり、メンバ相互のコミュニケーションをコントロールしたりといったことができるシステムを提供する。

Huber が他の論文で指摘しているように、意思決定時における GDSS の役割は、特に「理性的（rational）」に意思決定を行えることを目指したものである。つまり、意思決定会議の出席者は、グループ全体の役に立つために情報を理性的に使用しなければならない。

(2) GDSS の分類

① GDSS の分類の基礎

ここでは、グループが意思決定を行わなければならない状況において使用できる技術によって、GDSS を分類する。この場合、コンピュータ、通信、意思決定技術、あるいはこれらの組合せが考えられる。表8.3-1 に示すように GDSS のベースとして六つの技術に分類される。

表8.3-1 技術を基準としたGDSSの分類

技　　術	要素技術の組み合せ
電子会議室	コンピュータとＡＶ
遠隔会議設備	コンピュータと通信
グループネットワーク	コンピュータネットワークと相互会話
情報センタ	コンピュータ、データベース、及び検索ツール
意思決定会議システム	コンピュータと意思決定モデル
協同研究所システム	コンピュータと協同作業用ツール

② GDSS の要素のパッケージ

GDSS を理解するためには、これを(a) ハードウェア、(b) ソフトウェア、(c) オーガニゼイションウェア、および(d) 要員の 4 種類の要素から成る社会技術的なパッケージ（組合せ）と考えた方が分かりやすい。

(a) ハードウェア

ハードウェアとは、会議用施設、コンピュータ、通信設備、AV 機器等を指す。会議用施設としては、

最低限会議用テーブルと附属設備を備えた個室でなければならない。会議室には、会議のオブザーバ用のスペースを設けることもある。また、事務局や出席者が休憩できるようにラウンジルームやAVを備えた会議室が必要な場合もある。

コンピュータ機器としては、グラフィック処理が可能なコンピュータで、通常はグループ全員が見ることのできるような大きなディスプレイあるいはプロジェクションスクリーンを備えていなければならない。もう少し高度なシステムになると、各メンバに一つずつPCあるいは端末を備えたり、メンバ相互の通信の制御や、データベースやモデルにアクセスができる大規模な処理装置を置いたり、ローカルメンバ間の通信を行うためにLANを敷設したり、外部のメンバやデータベースとリンクするための遠距離通信システムを備えたり、複数の大スクリーンあるいはマルチウインドウの表示ができるスクリーンを備えたりする。また、高度なシステムでは、コンピュータ会議用やテレビ会議用にマルチポイントの相互通信ができたり、メンバ全員が見たり修正できるグループ共有のファイルやディスプレイと各メンバのみが見たり修正できる個人用のファイルやディスプレイとを分けたりする。

(b) ソフトウェア

ソフトウェアは、GDSSの技術的機能のキーとなるものであり、一般的な情報処理、意思決定モデル化、そして通信を支援するために利用される。一般的な情報処理のためのソフトウェアとしては、個人用／グループ用に、データベース管理システムや高級プログラム言語処理、あるいはグラフィック処理／スプレッドシート／統計分析等が可能な統合型パッケージなどがある。一方、意思決定モデル化用のソフトウェアは、特にグループの意思決定を支援することを目的にしており、モデル化用言語（例えば、SIMSCRIPTやDYNAMO等）、意思決定構造化技術、ブレーンストーミング、一般的なグループ活動のための技術、デルファイ技術、意思決定分析技術などのためのものがある。通信用のソフトウェアは、グループワークを協同的に行うことの支援を目的としており、ローカルあるいは遠距離間においてテキスト、データ、音声、ビデオ画像等の交換を行うためのツールなどがある。

(c) オーガニゼイションウェア

オーガニゼイションウェアには、組織として利用するデータ、グループとしての活動、グループの協同作業を管理する活動などがある。各メンバのアイデアや意見をまとめ、各々の重み付けや判定を行ってデータを統合するようなGDSSもある。また、組織内のデータだけでなく、外部のデータベースも利用できるGDSSもある。これらのデータベースには、戦略的・作戦的プラン、予算、マーケットや顧客のデータ、生産データが含まれる。また、オーガニゼイションウェアには、グループとしての活動が含まれる。これは、グループのメンバ間の関係が、独裁的であるか民主的であるか、友好的であるか闘争的であるか、あるいは政治的であるか論理的であるかによって変わってくる。グループとしての活動は、グループリーダの選ばれ方や議題の選ばれ方によって、物事の流れやグループ意思決定会議が左右される。また、グループとしての活動は、各メンバの組織の中での位置づけによって

規定されることもある。さらに、オーガニゼイションウェアには、グループの協同作業を管理する活動もある。これは、グループ活動に積極的に参加させたり、決定事項について秘密を守らせたり、グループ活動を円滑にさせるといったように、グループ活動の業務と社会性との対立を緩和させることを目的としたものである。

(d) 要員

要員とは、グループの参加者（出席者）およびグループ活動を支援するスタッフ（進行役）を指す。「進行役（ファシリテータ）」の存在がGDSSのキーとなる。この役目は、非常に幅が広い。進行役は、最低限グループ活動を支援する技術のオペレータであることが要求される。例えば進行役は、計算を行ったり、ディスプレイを操作したり、会議のドキュメントをとったりする。また進行役は、実際に会議を進行させたり、個々の意思決定やグループ活動のリーダ役を務めたりもする。新規のグループにおいては、トレーナ役や障害復旧役を務めることになる。この場合進行役は、メンバにハード／ソフトの使用法やグループ活動の技術を教え、問題が起こったときに助言したり、グループにフィードバックさせたりする。

表8.3-2に、上に述べた四つの要素とGDSSの六つの技術との対応を示す。

(3) GDSSシステムの実例

現在あるGDSSの実例を全てあげることは不可能に近い。電子会議室は、何百というオーダで存在する。AV業者の技術的アドバイスを受けて、建設業者やエンジニアリング企業が新設あるいは改装したものである。現在でも、電子会議室は増えつつある。これに比べて、ビデオ遠隔会議設備は少ない。米国においては、私設の遠隔会議設備は30～40の数であり、公共のものは15～20に過ぎない。また情報センタは、比較的新しく誕生したケースであり、米国においてはコンピュータのメインフレームメーカが、今までの情報処理機器供給の一環として始めたものである。情報センタの供給は、最初は組織の情報処理部門によって行われるが、次第にその情報処理部門にコンセプトや機器やソフトウェアを納入するコンピュータベンダが行うようになる。

意思決定会議システムは、他のGDSSに比べて明確な特徴を持っているため、市場に出回っている製品をあげることができる。表8.3-3に、主なGDSSの供給元と、供給しているシステム、およびそのシステムが持つ機能を示す。

これらのシステムについて調べてみると、いくつかのことが分かる。まず第一は、ほとんどの意思決定会議用GDSSは次の三つの機能を持っていることである。

① 意思決定ツリーやマルチアトリビュートユーティリティ分析といった構造的な意思決定分析機能

② 社会的な判断分析やデルファイ技術、一般的なグループ活動のための技術といった構造的なグル

表8.3-2　GDSSの代表的な要素

要　素	電子会議室	遠隔会議設備	グループネットワーク
ハードウェア	会議室、AV、ｸﾞﾗﾌｨｯｸﾃﾞｨｽﾌﾟﾚｲ、ｺﾝﾋﾟｭｰﾀ	会議室、AV、AVとｺﾝﾋﾟｭｰﾀの通信制御装置	ｵﾌｨｽ、ﾌｧｲﾙｻｰﾊﾞとWS、電話、ｺﾝﾋﾟｭｰﾀﾈｯﾄﾜｰｸ
ソフトウェア	会話的ｸﾞﾗﾌｨｯｸｽ	通信	同時／非同時のｺﾝﾋﾟｭｰﾀ会議、端末接続、ﾘｱﾙﾀｲﾑの会議のｽｹｼﾞｭｰﾘﾝｸﾞ、ﾋﾞｯﾄﾏｯﾌﾟﾃﾞｨｽﾌﾟﾚｲの共有
オーガニゼーションウェア	AV、規格化されたﾚﾎﾟｰﾄ、標準的な会議運用規定	AV、遠隔会議用運用規定	議長主導型ﾐｰﾃｨﾝｸﾞ
要　員	出席者、AV技術者	出席者（複数の場所）会議の進行役	出席者（ﾛｰｶﾙでの複数の場所）、ﾘｰﾀﾞ
例	組合せ式のAVﾙｰﾑもあるが通常はｶｽﾀﾑﾒｲﾄﾞ	静止画電話ｻｰﾋﾞｽ	MIT Lab for Computer ScienceのRTCAL と Mblink EIES, NOTEPAD, PARTICIPATE CONFER II

要　素	情報センタ	意思決定会議システム	協同研究所システム
ハードウェア	会議室、大ｽｸﾘｰﾝのﾋﾞﾃﾞｵﾌﾟﾛｼﾞｪｸﾀ、ｺﾝﾋﾟｭｰﾀ、端末	会議室、大ｽｸﾘｰﾝのﾋﾞﾃﾞｵﾌﾟﾛｼﾞｪｸﾀ、表示端末、票決用端末	会議室、電子黒板、WS
ソフトウェア	DB管理、統計用ﾊﾟｯｹｰｼﾞ、検索・ｸﾞﾗﾌｨｯｸｽ・ﾃｷｽﾄ処理用ｿﾌﾄ	意思決定分析用ｿﾌﾄ、ﾓﾃﾞﾙ化用ｿﾌﾄ、投票集計・表示用ｿﾌﾄ	ﾏﾙﾁﾕｰｻﾞｲﾝﾀﾌｪｰｽ、WYSIWIS、ｱｳﾄﾗｲﾝ化（Cognoter等）、評価（Argnoter等）
オーガニゼーションウェア	組織内外のDB、標準的な会議運用規定、標準的な会議（年次 総会等）	民主的な意思決定規定（一人一票、重要な議題を全て検討すること、多数決等）	標準的な会議運用規定
要　員	参加者、ｺﾝﾋﾟｭｰﾀ専門家ﾓﾃﾞﾙ化専門家	出席者、決定分析者、ｸﾞﾙｰﾌﾟ活動の進行役	出席者
例	HOBO System, SYSTEM W, EIS, EXPRESS, XSIM	Group Decision Aid, DDIのDecision Conferences, SUNY AlbanyのDecision Tectronics, Arizona大のPlanning Lab, Minnesota大のGDSS Lab	Xerox PARCのColab, MCCのNICKﾌﾟﾛｼﾞｪｸﾄ

ープ処理機能

③ データ管理、グラフィックディスプレイ、意思決定経過のドキュメンテーション、チュートリアル、意思決定分析のコンサルテーション、グループ活動支援、会議支援、票決集計・表示といった協同作業支援機能

表8.3-3 主要なGDSSシステム

供給元	システム	機能
Applied Futures,Inc.	CONSENSOR（ハード／ソフト）	投票の集計・表示
Compshare	SYSTEM W（ソフト） EXPRESS（ソフト）	ﾃﾞｰﾀ管理、ﾓﾃﾞﾙ化、統計分析、ｸﾞﾗﾌｨｸｽ、ﾚﾎﾟｰﾄ作成、PC通信
Decisions and Designs,Inc	Decision conference（完全なパッケージ）	会話的決定分析(6ﾓﾃﾞﾙ)、会議機能、決定分析、ｺﾝｻﾙﾃｨﾝｸﾞ
EXECUCOM	IFPS（ソフト）	会話的会計ﾓﾃﾞﾙ化、ﾃﾞｰﾀ管理、ｸﾞﾗﾌｨｸｽ
Institute for the Future	研究のみ	遠隔会議の研究・ｺﾝｻﾙﾃｨﾝｸﾞ
MIT Laboratory for Computer Science	MPCAL, CDS, RTCAL, MBlink（研究のみ）	ｶﾚﾝﾀﾞ管理・ﾘｱﾙﾀｲﾑの会議・協同文書編集など物理的に離れて散らばるﾛｰｶﾙｸﾞﾙｰﾌﾟの支援
Perceptronics,Inc.	GROUP DECISION AID（ハード／ソフト）	会話的意思決定ﾂﾘｰ、分析、ﾁｭｰﾄﾘｱﾙ、文書作成
SRI International	QUICKTREE, APLTYER（ソフト）	個人向けの会話的意思決定ﾂﾘｰ分析
SUNY,Albany Design Techtronics Group	公共サービス（完全なパッケージ）	会話的決定分析(6ﾓﾃﾞﾙ)、ﾃﾞｰﾀ管理、ｸﾞﾗﾌｨｸｽ、決定ｺﾝｻﾙﾃｨﾝｸﾞ
UCLA Cognitive Systems Laboratory	研究（数学的モデル／ソフト）	ｸﾞﾙｰﾌﾟ意思決定ｾｵﾘと分析
University of Arizona Planning Laboratory	PLEXYS（ソフト），研究・公共サービス（完全なパッケージ）	電子的なﾌﾞﾚｲﾝｽﾄｰﾐﾝｸﾞ、stakeholder分析、組織の分析、知識管理、ｸﾞﾗﾌｨｸｽ、ﾚﾎﾟｰﾄ作成、PC通信
University of Minnesota GDSS Laboratory	CAM－研究のみ（ソフト）	Snow card技術、一般的なｸﾞﾙｰﾌﾟ技術、stakeholder分析、ｽﾌﾟﾚｯﾄﾞｼｰﾄ・割付ﾂｰﾙ、ﾃﾞｰﾀ管理
Xerox PARC	COLAB－研究（完全なパッケージ）	対面式のｸﾞﾙｰﾌﾟﾜｰｸに対するｺﾝﾋﾟｭｰﾀの支援

これら全ての機能を全ての要素（ハードウェア、ソフトウェア、オーガニゼイションウェア、要員）で満たして販売されている例はほとんど無い。現在全要素のパッケージとして購入できるものは、Perceptronics社のGROUP DECISION AIDだけである。その他のシステムは、各組織の内部向けシステムであり、外部からは研究用として使用料を支払って利用できるのみである。Arizona大学のPLEXYSというソフトウェアは販売されており、約1ダースの大学と企業が導入して、各組織向けにアレンジしている。

第二は、一つのGDSSの中にいろいろな種類の機能を組合せる傾向が見られることである。例えば、SYSTEM Wのような情報センタ用の製品は、一つのシステムの中にいくつかのソフトウェア機能を装備している。従来このような各機能は、各々別のソフトウェア製品として分かれていたものである。同様に、GROUP DECISION AIDにはDeccisions and Designs社によって意思決定モデルが一覧できるように機能追加された版や、グループ活動にとってさらに役立つようにマーケティングプラン機能を追加した版等もある。Minnesota大学のCAMとArizona大学のPLEXYSは、グループ意思決定とグループ活動の研究の両方を支援するシステムを目指している。

第三は、GDSSは様々な形式をとりうるということである。一般にGDSSの技術が進歩するほど、GDSSがグループ活動へ介入する度合はより深くなる。

(4) GDSSシステム導入後の評価

① GDSS研究の発展経過

GDSS研究の発展経過には、大きく二つの流れがある。一つは、個人レベルおよびグループレベルでの意思決定そのものの研究である。この方面では、まず、個人およびグループにおける心理学的あるいは認知科学的な働きを見つけることや、小グループ内の相互関係の社会学といった研究がある。また、組織としての開発という観点から、グループにおいて意思決定までの時間を早め、そして友好的な合意に達せられるように、グループの活動を支援する方法を見つけるといった研究もある。これらの研究は、技術という面には係わらない。むしろ、意思決定を行う人々が何を考え、どのように行動するかということを問題にしている。

もう一つの流れは、意思決定に役立つ情報の収集・管理・提供を行う技術に係わる研究である。この方面は、エンジニアリングという面が中心になる。意思決定の新しい方法や技術を生みだし、これに基づいて意思決定を行う人の役に立つツールを開発することを目的としている。この方面における主な研究領域は、意思決定過程の分析や意思決定のモデル化、およびコンピュータや電子的なデータの記録・通信・情報表示等を対話的に利用した場合の人間の情報処理のあり方等である。

意思決定会議システムや協同研究所システムは、社会心理学と技術の両方の面から開発されたものである。この技術は、情報の獲得と共有を容易にするために、他のGDSSの技術を集めたものである。

グループ意思決定活動の研究成果は、その活動を円滑にするのシステムの技術開発のために適用されている。今のところ、意思決定会議システムや協同研究所システムは技術の面に片寄っている。それは、実際の意思決定がどのように行われるかということが、依然としてよく分かっていないからである。グループ意思決定活動についての社会心理学的な見方は広く指示されているものの、依然として「理性的な」モデルの方から影響を受けている。

② GDSS を使用して得られる効果

(a) 気分的な効果

GDSS は、会議を活発にし、グループ間の結束を固めるという気分的な効果をもたらす。コンピュータを使ったグラフィックス表示や投票、通信、データ分析、モデル化のシステムによって、グループ活動に関心を持つようになる出席者もいる。これらの技術の中には、その動きを見ているだけでおもしろく、エキサイトさせるものがある。

ただし、GDSS のこのような気分的な効果が、意思決定の結論の品質に寄与するかどうかは、まだはっきりしていない。

(b) 議事進行の手助け

GDSS の技術は、グループ意思決定の議事進行の手助けとなる。グループ意思決定活動には、決定すべきキーとなる問題に注意を向けさせたり、出席者から問題に対する考えを引出したり、どのような行動を起こすべきかについての出席者の合意の糸口を得るために、議事進行のルール（プロトコル）がある。

GDSS の技術は、決定すべき問題の中で重要な項目は何かということを、出席者により早くより正確に理解させることを目指していると考えられる。簡単な例としては、コンピュータによるグラフィックディスプレイがあげられる。多くの人々にとって、文章や表からではデータの特徴を読み取ることは難しいが、チャート図やグラフを用いれば、特徴のある部分についてその違いを際立たせることができる。投票システムはこれに似ているが、さらに明確な効果がある。電子投票システムは、合意があるか無いかをはっきりさせ、さらに討議すべき項目をはっきりさせ、討議のために新たな情報を提供するといった手助けを目的として設計される。

このような GDSS 技術が、グループ意思決定の議事進行ルールを確立させる。このようなルールが常によりよい意思決定の役に立つか否か、研究からははっきりとは分からないが、研究だけでなく実際の経験を通して、このルールには何らかの知恵があるだろうということは分かっている。とは言っても、確立したルールは、広い範囲で適用できるのか否かは明確ではない。

(c) 意思決定の際に利用できる情報の品質向上

GDSS は、意思決定者が利用する情報の品質を向上させることがある。これは、次の二つの場合である。まず、意思決定会議において、他の方法では出席者が入手できない情報を、オンラインデータ

ベースを利用して得ることである。もう一つは、モデル化やシミュレーション機能を利用して、意思決定活動において発生する情報を切り詰めたり、評価したり、統合したりすることである。

GDSS のこのような手助けによって、会議の出席者の一人一人が情報を取捨選択したり、意思決定における問題点を理解できるような、品質の向上が望めるであろうか。単に多くの情報を提供するだけでは意思決定の向上にはつながらない、ということは広く認められている。多くの情報がどんどん与えられると、出席者の能力を超えてしまい問題を検討できなくなり、意思決定活動が混乱してしまうこともある。意思決定会議の場において、単純にオンラインデータベースを利用するだけでは、意思決定の活動やその品質を劇的に向上させるというわけにはいかないようである。とは言え、出席者が選択できる様々な仮定をテストできる GDSS の効果は、GDSS の技術の中でも重要なものであるだろう。

(5) GDSS 導入の障害となる問題

① 技術的な問題

GDSS は、コンピュータの情報処理や記録、グラフィックディスプレイ、データベース管理システム、統計処理、意思決定分析やモデル化のプログラム、通信、分散入力装置等の様々な技術に支えられている。技術開発の現状において、これらの技術が GDSS で用いられる際に、以下のような問題点がある。

- コンピュータを使用する際のアクセスのし易さや柔軟性（ネットワーク通信上の問題）
- 表示技術（ディスプレイの解像度の問題）
- グラフィックスの能力（表示速度と表示可能な大きさの問題）
- モデル化や意思決定分析用のソフトウェア（高度なモデル化の問題）
- データベースとモデル化を統合する能力（モデル化機能のリンクの問題）

② GDSS のパッケージの問題

前述したGDSS 技術を構成する四つの要素がパッケージとしてうまく組み合わされなければ、GDSS のアプリケーションはうまく働かない。ハードウェアやソフトウェアの技術が無ければ、GDSS は夢のようなもので実現不可能である。しかしそれだけでなく、技術的な熟練や組織としての安定、会計上の安定、クライアントや顧客に対するデモ効果等がパッケージとしてまとまらなければならない。このような意味で、GDSS は単に個々の作業を達成するためのツールではなく、生産と結び付いた大規模なコンピュータシステムとして捉えられる。

③ 意思決定活動に対する理解不十分

個人やグループの意思決定活動に係わる研究は、比較的新しい分野であり、意思決定の振舞いについて未解決の問題が多い。認識活動に対する理解が不足していることや、複雑な意思決定状況にある

曖昧さ等が原因となって、意思決定の研究を難しいものにしている。これが、意思決定のための支援技術を開発する際に、比較的狭い範囲で合理性を主体に意思決定活動を捉えてしまう大きな理由である。現在のシステムで採用されている理性的なモデルによる捉え方は「現実の世界」の意思決定活動の中では限られたものでしかない。なぜなら、人々が時々見せる、訳の分からない非理性的な振舞いというものを考慮していないからである。たしかに、理性的な捉え方も重要な役割を果たしており、現実の意思決定活動において当てはまることもある。しかしそれは、多くの場合、現実とは似ていない。

Huberによれば、理性的なモデルは単純であるという点で役に立つが、この他に、意思決定に係わる文献から得られた三つの重要なモデル（「政治的／競合的モデル」、「ごみ箱モデル」、「プログラムモデル」）があると述べている。そしてGDSSの技術が、この三つのモデルにどの様に対応できるかについて述べている。この三つのモデルにおいて、GDSSの技術は問題や機会を明らかにし、選択した結果についてよりよく把握させ、問題を扱う際の意思決定者の役割を明確にする。「政治的／競合的モデル」では、GDSSは他の派閥の存在と、それらの派閥の強みと弱みを評価する手助けとなる。また「ごみ箱モデル」では、GDSSは問題が起こったときに、それまで棚上げされていた解決策を取り出す手助けとなる。そして「プログラムモデル」では、GDSSは問題や解決策に係わるさまざまなグループ（あるいは小グループ）の振舞いを明確にする手助けとなる。意思決定に役立つこのような種類のGDSSは、開発して導入することは可能と考えられるが、今のところは、このようなことができるシステムは無い。

GDSSがもっと効果的に発展していくため（特に理性的な意思決定の振舞いのモデルとは異なる場合に対応していくため）には、現実の意思決定活動について研究を続けていくことが必要である。とは言っても、おもしろいことに、必ずしも新しいGDSSを開発していく前に意思決定活動の研究が行われなければならないということはない。逆に、GDSSのツールを開発して導入することが、意思決定活動を研究する上で重要であると考えられている。これらのツールは、正しく設定されて上手にコントロールされれば、意思決定の実験的な研究に大いに役立つ。この問題について討論する最近の研究者会議において、こういった実験的研究はますます一般的になってきている。

(6) GDSSの近い将来

① GDSSの成長する度合

意思決定会議システムや協同研究所システムの高度な適用という面でのGDSSの成長は、今後数年間は比較的緩やかであると思われる。これらのシステムは、ユーザベースではまだ少数であり、現在のシステムが持っている機能からみれば、短い間に広く適用されるということはまだ充分保証できるものではない。しかし、政府や企業の予算に見られるように、この方面への関心はまだ続いているの

で、開発は引続き行われると思われる。

② 技術

GDSS の技術面で現在抱えている問題の多くは、数年後には解決されていると思われる。コンピュータのプロセッサ技術におけるコスト／パフォーマンスは向上し続け、これにより多くの組織がGDSS 技術を導入するようになると予想される。ディスプレイ技術についても、コスト／パフォーマンスは向上するであろう。データのアクセス・分析、意思決定分析、モデル化等のためのソフトウェアは、今後も開発が続けられると思われる。これらのソフトウェア製品は、マイクロコンピュータ市場への投資が続く中で、開発に拍車がかかるであろう。エンドユーザのアクセスやデータベースの利用が便利になるように、もっとユーザフレンドリなインタフェースが開発されると思われる。

③ 意思決定活動の知識

意思決定活動を把握するという点では、さほど目ざましい発展は望めないと思われる。この方面の発展は、認知科学や管理工学などの分野の発展に依存している。この分野においては、ブレークスルーがあるかも知れないが、急速な進歩は望めそうもない。意思決定活動の知識の面では、緩かな成長となるであろう。ＡＩ技術が人間の認知機能の研究に役立っているように、GDSS 技術の実験的な導入が意思決定活動について新しい知識を得るために役立つと思われる。

8.3.2 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。

8.3.3 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ブロッキングや、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。

8.3.4 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）がそれらを分析して、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。

8.4 オフィスプロシージャシステム

8.4.1 ECF

原文タイトル： Support of Cooperative Work by Electronic Circulation Folders

翻訳タイトル： 電子循環フォルダによる協調作業の支援

著　者： B. Karbe, N. Ramsperger, P. Weiss

出　典： ACM SIGOIS Bulletin, Vol.11, No.2, April 1990, pp.109-117

本稿では、ProMInanD (Extended Office Process Migration with Interactive Panel Displays) プロジェクトで開発された電子循環フォルダ (Electronic Circulation Folders : ECF) について、その機能やインプリメント、および評価が紹介されている。

(1) 循環フォルダ (CF) と電子循環フォルダ(ECF)

オフィス作業はいくつかのステップからなり、オフィス・ロール（オフィスにおける役割）を果すオフィスの人間がこれを実行する。これはより高いレベルの活動の一部と理解することのできる包括的な活動であり、判断や失敗、予期しない反応などによって、予想できない、あるいは非決定的な行動を取ることのある人間がかかわることで問題は難しくなる。

オフィス・タスクの処理を支援するツールとして、これまで広く用いられてきたものに循環フォルダ (CF) がある。これは、オフィス・ワーカが作業を行うべき、任意ではあるが作業に関連した文書を内容としており、CF の表紙にはオフィス・ワーカの宛名が記載されている。社内のメッセンジャ・サービスが、あるオフィス・ワーカの発送書類入れの密閉した CF を、そのオフィス・ワーカが CF の表紙に書いた宛先の到着書類入れに運ぶのである。CF の内容に対する作業と、CF の組織内部での移送ははっきり分れている。

CF は、オフィス・タスクのための作業を移動するツールとしては非常に柔軟性の高いものだが、欠点もいくつかある。まずメッセージ・サービスに時間がかかる点があり、また宛先のオフィス・ワーカが不在である可能性が考慮されていない。さらに、ファイルの行方が重大な問題となることもあり、これは関連するオフィス・タスクが形式化されていないとさらに大きな問題となる。

ProMInanD システムは、CF の代りに電子循環フォルダ (ECF) を使って、オフィス・ワーカが CF で慣れている柔軟性を提供するものである。ProMInanD システムは、従来の CF と ECF 間の関係を示すことでオフィス・ワーカにその全内容を説明できるよう設計されている。あるタスクに関連したタイプの ECF インスタンスが、移送仕様に基づいて、そのタスクにかかわるオフィス・ワーカに自動的に

移送される。ECFの内容に対する作業は、いわゆるステップ・プログラム、すなわちProMInanDではない任意のシステムに支援され、その自動呼出は移送仕様で決定される。移送は組織の情報によって決り、オフィス・ワーカが影響を与えることもできる。一つのオフィス・タスクを分割した複数のサブタスクに対する並行作業は、一つのECFを先祖とし、従属するECFをその子孫とするECFファミリによって支援することができる。

(2) ECFの仕様

① 表現方法

ECFは、説明部と内容部という大きな二つの部分からなっている。説明部は、ECFの移送仕様とECFのタイプ別の状態情報、システム全体で一意の識別、他ECFとの関係、進捗状況および実際にオフィス・ワーカがこれまで実行したステップの履歴に分れる。内容部は、ECFの作業内ステップを実行するのに必要な文書やフォームを保持するワークパート、オフィス・ワーカが編集できるオプションのフォルダ・スリップ、オフィス・ワーカが追加した文書を入れるためのオプションの付録からなる。

オフィス・ワーカが関心を持つのはECFの内容部で、オフィス・タスクに対する本質的な作業、例えば「請求オーダ」タスク用のECFであれば、請求オーダ・フォームの記入などの作業を行う部分である。行うべき作業の種類は移送仕様が決定するが、ECFがどこからきてどこへ行こうとしているかはオフィス・ワーカも関心を持つことなので、移送仕様をグラフィック表示で探索できるようになっている。

② 移送仕様

ECFの移送仕様は、実行すべきステップの形で移送ルートを決定する。簡単に言うと、どのオフィス・ワーカがステップを実行しなければならないか、どの文書を使うか、そのためにどのアプリケーション・プログラム、いわゆるステップ・プログラムを呼出すかをステップが定義するのである。オフィス・ワーカは、組織としての機能、いわゆる「オフィス・ロール」によって定義される。個々のステップに関するこのような情報以外にも、移送仕様は次のような情報を持っている。

- 全体的に見てタスクを実行するのに必要なすべての文書
- かかわりのあるすべてのオフィス・ロール
- すべてのステップの順序と、それがオフィス・ワーカによって実行されるのか、自動的に実行されるのかどうかの区別
- 複数のステップのサブタスクへのグルーピング
- ステップ・プログラムの実行結果によるそれ以後の移送方法
- 従属ECFの自動起動

ECFの各タイプについてそれぞれ移送スキーマが存在する。あるECFのタイプの新しいインスタン

スを生成すると、その移送仕様は対応する移送スキーマを適正に加工し、パラメータを定義することによってつくられる。

「パフォーマ・ロール（実行者の役割）」も移送仕様を決定するパラメータの一つで、一部がECFの生成時に、一部が移送中に設定される。「組織説明」は、どのオフィス・ワーカがどのオフィス・ロールを果すことができ、そのロールでどのECFのインスタンスが生成されるかを指示する。具体的なオフィス・ワーカ（すなわちパフォーマ）が、適したロールになってECF、例えば請求オーダのインスタンスを生成すると、その生成時、オフィス・ワーカがシステムに知らされる。移送仕様は、この例では、次に部門長による「認可」が実行されなければならないと記述されていることもあり得る。このような場合、実際に誰が実行しなければならないかは、ECFのインスタンスを生成したオフィス・ワーカが、組織説明から具体的な部門を探索し、その部門の長の役割を果す具体的なオフィス・ワーカを探すことで行う。このような宛先の解明は、移送中できるだけ後の方で行われる。

ステップとECFの特性が、例外処理に関する情報を与える。一定の状況下で省略できるステップもあるが、「認可」のように強制的なもので省略できないステップもある。

ステップ・プログラムの結果と、そのECFの以後の移送との間には従属関係が存在し得る。上の例で認可が行われなかったら、そのECFをそれ以上通常に処理することは意味を持たない。このような従属関係は、アプリケーションに固有の決定プログラムによって評価される。このプログラムは移送仕様の中でも参照される。

複雑なオフィス・タスクは、組織の各部門に関連するサブ・タスクに分割できることが多い。サブ・タスクをどのように実行するかは、その部門の責任となる。他部門のオフィス・ワーカは、そのサブ・タスクがどのようなステップで処理されたかということを知る必要はないので、移送仕様のグラフィック表示では、そのサブ・タスクはステップのない一つのアイテムとして示される。サブ・タスクの概念を取り入れることにより、特に移送仕様の定義を簡略化できる。

ECFの移送仕様のスキーマを作成あるいは修正するため、ProMInanDはオフィス作業のオーガナイザが利用するグラフィック移送仕様エディタを備えている。

③　ステップ・プログラム

オフィス・ワーカの本質的な作業、すなわちECFのワークパートに記述された文書に取り組むことによるオフィス・タスクの処理を支援するには、いわゆるステップ・プログラムがアプリケーション開発者によって提供されなければならない。標準的なステップ・プログラムは、テキスト・エディタ又はフォーム・ハンドラである。これらの支援するアプリケーションは、より専門化されたステップ・プログラムであるのが普通で、ProMInanDがインストールされたオペレーティング・システム上で動くプログラムは、すべてステップ・プログラムとして使うことができる。移送システムのデータベースに適切な情報を入力するだけで、新しいステップ・プログラムを移送システムに知らせること

ができる。

(3) ProMInanD の機能

ProMInanD の機能としては、オフィス・ワーカの自分自身の作業を支援するオペレーションとその結果を移送する作業を支援するオペレーションの二通りがある。

オフィス・ワーカが ProMInanD にログインすると、電子デスクが表示される。電子デスクによって、自分のオフィス・ロールによる選択を行ったり、他のツール、すなわち到着 ECF の入っている到着書類入れや、作業を行ってデスクを離れたがまだ完全に終了していないECFに関する情報を入れた発送書類入れ、作業の中断した ECF バイルや、スタートできる「空の」ECF の入ったフォーム・ボックスなどを起動することができる。

① ローカル・ワークを支援するオペレーション

どの時点でも、オフィス・ワーカのデスクにある ECF は、ツールの中にあるか表示されているかのどちらかである。オフィス・ワーカが役割上実行を許されていれば、到着書類入れやフォーム・ボックス、バイルの中の ECF を選択することができる。選択されると ECF は表示され、オフィス・ワーカは ECF の履歴と状況に関する情報を得る。ECF をオープンして内容に関する作業を開始することができる。作業を行うことのできるステップ・プログラムがいくつかある場合、オフィス・ワーカは好きなステップ・プログラムを選択できる。オフィス・ワーカは、ECF をバイルにしまうことで、作業を現在のステップで中断することができる。作業を完了したい場合は、いくつかあるモードの一つで転送することで、これが発送書類入れに入れられる。

デスク上の ECF に、通常のローカルなワークフローからの逸脱とも考えられる影響を与えることもできる。cancel step と delete ECF がそれで、オフィス・ワーカがステップをキャンセルすると、それまでそのステップで実行された作業はすべてキャンセルされ、ECF は発送されたばかりと同じ状態になる。オフィス・ワーカが ECF を削除すると、スタートさせたことのなかったかのようにデスクから除去される。しかし ECF の削除は、ECF が最初のステップにある場合にのみ可能である。

② 移送を支援するオペレーション

ECF の通常の移送を取り扱うオペレーションは、forward、postpone、inform である。これらは、表示された ECF あるいは発送書類入れに対して、オフィス・ワーカが利用することができる。

ステップの作業が完了すると、オフィス・ワーカは ECF を forward（転送）することができる。これによってこの ECF はデスクを離れる。ステップが最後まで行っていなければ、ProMInanD は次のステップに定義されたオフィス・ロール X を果すオフィス・ワーカに、ECF を配送しようとする。この役割を果すオフィス・ワーカが二人以上いる場合は、ProMInanD が選択を行う。この役割を果すものが誰もいない場合は、ECF は「X ロール のオフィス・ワーカに移送中」という状態に遷移し、後にオフ

ィス・ワーカがこの役割に入った時に配送される。配送することができないでいても、次のオフィス・ワーカについて詳しいことがわかったら、「X ロールを Y オフィス・ワーカ に指定」という状態に遷移することもある。

ステップの作業が中断した場合、オフィス・ワーカは ECF を postpone（延期）することができる。ECF は、「X ロールの Y オフィス・ワーカ の再提出まで延期」という状態に遷移し、オフィス・ワーカが指定したタイム・リミット後に提出される。

オフィス・ワーカのデスクを離れた ECF は、システムを離れない限り発送書類入れに記録されている。発送書類入れの ECF に inform（通知）オペレーションを行うことで、オフィス・ワーカは ECF の現在の所在に関する情報を得ることができる。オフィスでよく発せられる「今ファイルはどこ？」という質問に対する答えを得られることは、オフィスの作業スタイルと協調にも影響を与えることになる。

通常の移送ルートから逸脱させるためのオペレーションもあり、これらは、not me、refer back、append、delegate、shortcut、shift、fetch back、cancel ECF である。どれも、表示された ECF あるいは発送書類入れに対してオフィス・ワーカが利用することができる。

オフィス・ワーカが、現在のステップで作業しなければならないのは自分ではないと苦情を言う場合には、not me（自分ではない）モードで ECF を転送することで、前のオフィス・ワーカに返送される。

さらに情報が必要な場合があるが、オフィス・ワーカはその要求をフォルダ・スリップを記載するよう促される。ECF を refer back（差戻し）モードで転送することで、ECF は前のオフィス・ワーカに配送される。

append（追加）によって、オフィス・ワーカは、移送仕様の指定に加えて、現在のステップの後にさらにステップを追加することができる。新しいステップのステップ・プログラムは、現在のものから引継がれるが、これを受取るオフィス・ワーカか組織のユニットを指定しなければならない。これは、オフィス・ワーカが現在のステップを実行あるいは補足するため他のオフィス・ワーカ（通常は部下）を必要とすることがあるという考え方によるものである。これを簡単に一般化したものがフリースタイルの ECF となる。これはたった一つのステップで始まり、続くステップが全て追加となるものである。これによって形式化されていないオフィス・タスクがすべて支援できることになる。

delegate（委任）オペレーションによって、現在のステップに二つのステップを追加することができる。一つ目のステップは append と同様に追加されるが、二つ目は委任するオフィス・ワーカに戻される。これは、委任したオフィス・ワーカが結果を取り戻して責任を負うという考え方によるものである。

一定の状況下では、ステップを後ろへ shift（シフト）したり shortcut（近道）したりすることができ

る。シフトされたステップは後に作業を行うが、近道したステップは排除される。シフト・オペレーションは、シフトしたステップ担当のオフィス・ワーカがしばらく不在の場合などに、間のステップを一時的に優先させるものである。

発送書類入れに記録されているECFをfetch back（取戻し）したいオフィス・ワーカもいる。ECFが延期されていると、これによって不完全な再提出が行われることになるが、そうでない場合、ProMInanDは、次のオフィス・ワーカが到着書類入れから選択していない限り、ECFを取り戻そうとする。

進行中のオフィス・タスクが陳腐化することもある。そのような場合、対応するECFのイニシエータは「ECFをキャンセル」したいと望むが、ECFはキャンセルによる放棄が既に不可能な状態に達していることがある。これは移送仕様で決定される。放棄がまだ可能な場合でも、放棄までに既に実行したステップの作業を補償するためアプリケーションに依存したステップを追加して行う必要がある場合もある。

(4) ProMInanDのインプリメント

ProMInanDの現行バージョンは、TCP／IPを有するUnixとSun Viewのウィンドウ・システムが稼働するSunワークステーションで実現される。ProMInanDは全てObjective-Cで記述されている。

ProMInanDのユーザ・インタフェース部分は、他の主要部分から完全に独立するよう開発された。さらに、Objective-Cのグラフィック・ファウンデーション・ライブラリであるICpak201を使って開発されており、特にグラフィックおよびウィンドウ向きユーザ・インタフェースに適している。ICpak201の最下部コンポーネントである、いわゆる「アースペース」が、ワークステーションにあるウィンドウ・システムにインタフェースをマップする。そのため、ProMInanDのユーザ・インタフェースを他の目的システムに移植する場合は、アースペースを取替えるだけでよい。

ProMInanDの移送システム(MS)は、各オフィス・ワーカのワークステーションで走る複数のローカル移送サーバ(LMS)と一つのグローバル移送サーバ(GMS)からなる。LMSはワークステーションに配送されたECFを扱い、GMSはワークステーション間のECFの移送を制御することになる。

移送仕様は複雑なオブジェクトで、ECFを指定すると同時にこれを制御するが、そのためには、移送仕様を解釈し、特に例外処理に関してはダイナミックに修正しなければならない。Objective-Cで書かれているためオブジェクトの受け渡しや、格納、読取り、再起動ができるので、移送仕様はECFに追従できる。そのため、オブジェクトを異なるプロセス間で簡単に共有することができるのである。

ProMInanDは、分散型データベースシステムに保持される電子的組織ハンドブックに広範囲に依存しているため、スロット・メカニズムの拡張と組合わされて、ECFを配送すべきオフィス・ワーカの宛先が配送直前にわかるようになっている。このように、組織説明がMSのアルゴリズムと別になっ

ているため、指定されインスタンス生成までされたECFは組織のある程度の変更には影響を受けない。

(5) 評価

ProMInanDは比較的成熟した状態にあるが、さらに付け加えられれば望ましいものや開発中のものは次の通りである。

- ECFの寿命は数日あるいは数週間以上であるため、その間、ECFの新バージョンが開発されたり、組織が変更されたりすることがあり得る。このようなケースを適正に処理するため、ECFのバージョン管理が必要である。
- 終了したECFの内容は記録することができる。この記録を活用するため、有効な記録システムが必要である。
- 大きな組織では、グローバル移送サーバが一つでは大人数のオフィス・ワーカを処理できない場合が十分に考えられる。そのため、広域ネットワークも統合できるクラスタ・コンセプトを開発中である。

8.4.2 DOMINO

原文タイトル： Experiences with the DOMINO Office Procedure System

翻訳タイトル： DOMINOオフィス・プロシージャ・システムの利用経験

著　者： Thomas Kreifelts
Elke Hinrichs
Karl-Heinz Klein
Peter Seuffert
Gerd Woetzel
(German National Research Center for Computer Science (GMD), Germany)

出　典： ECSCW '91 Proceedings, September 1991, pp.117-130

本稿では、GMDにおいて開発され利用実験されたDOMINOと呼ばれるオフィス・プロシージャ・システムについて紹介されている。

(1) オフィス・プロシージャのモデル化

オフィスにおける事務処理の流れの円滑化を図るためにDOMINOと呼ぶシステムを開発し、これをGMD内で試験的に運用した。

DOMINOシステムは、オフィス内の各個人の物品購入願や出張・休暇願等を該当する管理者に提出し、管理者がこれを処理するといった事務処理を半自動化して、オフィスのコミュニケーションの円滑化を図るものである。

本システムの設計の際に前提となった点は、次のとおりである。

① オフィスワーカは、各々プライベートな作業ドメインを持っており、これらのドメイン間をメッセージが流れることによって共同作業が成り立つこと

② これらのメッセージは、共同作業においてスピーチアクトの理論に基づいて処理されること

③ 組織化されたグループにおける共同作業は、各作業ステップが特定の入出力によって結び付けられて処理されること

④ 各作業ステップにおけるこの特定の入出力は「理想的」に行われること

このような前提に従って、本システムではオフィスにおける様々な共同作業をPetri netの考えに基づいてモデル化を行った。

(2) DOMINOシステムの構成

システムの構成は、各個人が所有するMacとホストであるUNIXマシン（SUN）とをEthernetで接続している。ホスト上には「仲介役（Mediater）」と呼ばれるエージェントが動作しており、これが電

メールの自動的な疑似ユーザとなって各ユーザとのメールのやり取りを行う。この電子メールはCoPlanXと呼ばれる独自のプロトコルで通信が行われる。「仲介役」は組織やプロシージャに関するデータベースを管理しており、このデータベースを解釈して共同作業の処理の流れに沿ったメールの送信を行う。

(3) DOMINOシステムのインタフェース

ユーザインタフェースは、Macintosh のHyperCardを利用してインプリメントされている。

購入願などのウインドウは定型のフォーマットで、必要な箇所だけをキー入力すればよいようになっているが、不定型のデータ（図形など）もクリップで留めるようなイメージで一緒に送受することが可能である。

(4) DOMINO システムの利用結果

本システムをGMD内で運用してみた。ユーザとして、五つのグループに分かれた120 人の研究員を対象とした。この利用結果として、次の点が判明した。

- ユーザインタフェースの点では、簡単で使いやすく、処理の流れが把握しやすい。
- 流れが構造化されていない処理に対しては、対応しきれない。
- オフィスの事務処理環境は、各エンドユーザに合わせて変更することは難しい。
- 組織の役割あるいはグループの役割といったものは、完全には表現できない。

今後は、DOMINOシステムでの経験を踏まえて、さらに各メンバの要求に合わせて柔軟に対応できるシステムを開発する予定である。

8.4.3 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。

8.4.4 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。

8.5 メッセージ構造化システム

8.5.1 Information Lens

原文タイトル： Semistructured Messages Are Surprisingly Useful for Computer-Supported Coordination

翻訳タイトル： コンピュータ支援協同作業にとって非常に有益な「半構造化されたメッセージ」

著　者： Thomas W. Malone
Kenneth R.Grant
Kum-Yew Lai
Ramana Rao
David Rosenblitt
(Center for Coordination Science, Massachusetts Institute of Technology, Mass., USA)

出　典： ACM Transactions on Office Information Systems, Vol.5, No.2, April 1987, pp.115-131

本稿では、組織内部で情報の共有を図るために Information Lens というシステムを開発・導入したことで得られた成果が紹介されている。まず初めに、グループのコミュニケーションや協同作業用のシステムを開発する際に、半構造化されたメッセージのテンプレートを応用するというアイデアが紹介され、このアイデアを取り入れると、メッセージを作成したり処理したりする一般的な機能を設計することが簡単になることが述べられている。つぎに、この機能は作業割付（task tracking）やカレンダ管理（calendar management）、コンピュータ会議等のためのさまざまなアプリケーションに応用できることが示されている。最後に、あるコミュニティがその内部で使われるメッセージのタイプをどのように決めていくかという問題について若干述べられている。

(1) 半構造化されたメッセージとは

ここでは「半構造化されたメッセージ」を、次のように定義する。すなわち、さまざまなタイプに識別できるメッセージであって、各々のタイプはいくつかのフィールドを含んでいる（その内のいくつかのフィールドには、構造化されていないテキストやその他の情報が入っているものもある）。例えば、「セミナーのお知らせ」の場合には、「日時」、「場所」、「講演者」、「演題」といったフィールドが通常のヘッダフィールドや講演内容の記述の他にあるテンプレートが当てはめられる。

このアイデアが重要であるというのは、次のような理由による。

① 半構造化されたメッセージは、他のものよりもコンピュータが自動的に処理できる情報の範囲が広い。

② 半構造化されたメッセージは、厳密な構造化に比べて自由に情報の交換ができる。

③ 情報処理を行う人々は、今までもメッセージをある程度構造化されたタイプのものとして扱ってきている。

④ メッセージが自動的に処理されない場合でも、準構造的なテンプレートはメッセージを作る人にとって役に立つ。

⑤ 半構造化されたメッセージは、段々と拡張していくシステムの設計をやりやすくする。

このような一般的な理由に加えて、⑥一般的な性質を持ついくつかのタイプを継承して固有のタイプを作れるようにメッセージのタイプを継承できる機能を持たせることや、また⑦メッセージや、メッセージを処理するルール、そして新しいメッセージテンプレートをディスプレイから編集できるエディタ群を用意することにより、半構造化されたメッセージをより簡単に使えるようになる。

(2) Information Lens システム

Information Lens は、半構造化されたメッセージの特長を活かした知的な組織内情報共有システムである。このシステムは、人々がすでに受け取ったメッセージについて取捨選択したり、並べ変えたり、優先順位を付けたりする際に手助けとなる。また、受け取っていないメッセージの中で役に立つものがあるかどうかを知る際の助けにもなる。

さらに本システムは、電子メールシステムに応用するときに、次の四つの重要な機能を提供することができる。

① メールの送信者は、メッセージを作る際に構造化されたテンプレートを使うことにより、そのメッセージにどんな情報を入れたらよいかがわかり、さらに個々の情報について選択していくだけでほとんどメッセージができてしまう。

② メールの受信者は、送信者が構造化して送ってきたメッセージを、自動的に取捨選択・分類して該当するフォルダに入れるようにルールを指定することができる。

③ 送信者は、通常の宛先（個人あるいは受信者リスト）の他に、ある特別なメールボックス（現行では「Anyone」と名付けられている）に送ることができ、この場合このメッセージに関心を持つと思われる人に宛てて自動的に送られる。

④ 受信者は、Anyone宛であって自分のところには送られてこないメッセージを受け取るようにルールを指定することができる。

宛先の中にAnyoneを含むメッセージは、Anyone以外の宛先にはメールサーバが直接送信し、それと同時にそのワークステーション上の自動メールソータに送られる。自動メールソータは、ルールを指定している受信者にそのメッセージを送る。こうしてメッセージは、関心を持つと思われる人に対してのみ送られる。もし機密保持の問題があるときには、「Anyone-in-受信者リスト」という宛先にしておけば、指定した受信者リストの中でルールを指定した人に対してのみ送られる。

本システムの全てのアプリケーションは、Loopsという、LISPのオブジェクト指向版を使用して、Xerox 1100シリーズのワークステーション用のInterlisp-Dのプログラミング環境で作成された。これらのアプリケーションは、既に人々が使っている機能も一緒に使えるようにすることを考慮して、既存のメールシステム（Lafite）の上に構築されている。ユーザは今まで通りの使い方で、メールをやり取りしたり、受信者リストを利用したり、受信したメッセージを手作業で分類してフォルダに入れたりすることもできる。このようにして本システムは、本研究グループの５人のメンバにより１年以上の間継続して使用された。

(3) 半構造化されたメッセージで可能になること

① メッセージを作成する際の自動的な補助機能

さまざまなタイプの構造的なメッセージが用意されているため、メッセージを作成する際に、いろいろと自動的な補助機能を提供することができる。例えば、Information Lens システムでは、メッセージのタイプごとにさまざまなテンプレートを用意し、そのテンプレートはそのフィールド特有の情報を扱うことができる。このようなテンプレートを利用してユーザがメッセージを作成していくグラフィックインタフェースの例を図8.5-1 に示す。ユーザは、マウスで希望するメッセージのフィールドを選択した後、そのフィールドの「既定値」か、そのフィールドの意味の「説明」、あるいはそのフィールドに記述されると考えられる内容の「選択リスト」のいずれかをマウスで選択できる。内容の選択リストの中から希望する項目を選ぶと、その内容が自動的にメッセージのテキストに挿入される。

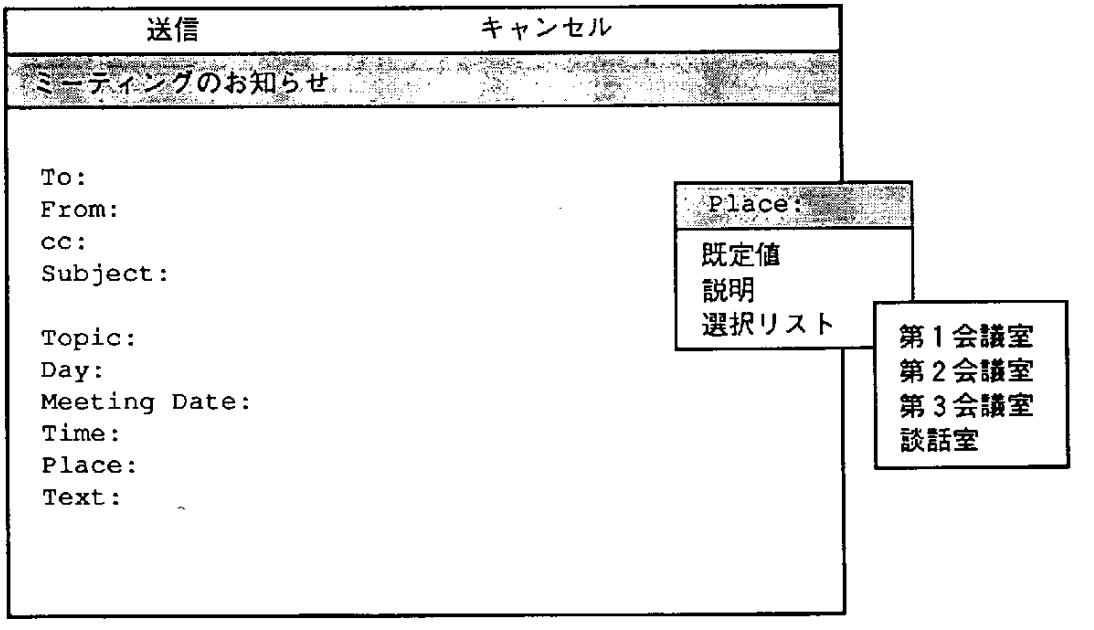


図8.5-1 メッセージ作成用テンプレートの例

メッセージのタイプごとに、既定値や選択リストに関する知識が豊富になってくるにつれて、システムはもっと簡単にメッセージの作成ができるような機能を提供できる。例えば、毎週の定例ミーティングのお知らせ等の場合、ほとんどの項目（送信先や曜日、時刻等）はあらかじめ既定値で埋めておくことができる。そのフィールドの既定値や意味の説明、あるいは内容の選択リストは、個々のユーザが、普段の自分のコミュニケーションのやり方や好みに合わせて設定することができる。

現行のシステムでは、全てのタイプのメッセージにTopicというフィールドが存在する。このフィールドは、そのメッセージのトピックを表わすキーワードを記述するためにある。このフィールドの選択リストは、他のフィールドのような単純なリストではなく、ネットワーク構造化されたさまざまなトピックの中から特定のトピックを得られるようになっている。

このようなメッセージ作成補助機能を望まないユーザは、messageという名の一般的なメッセージタイプを選択しておけば、既存のテキストエディタを使って、メッセージの各フィールドを直接編集することもできる。例えば、メッセージの中に希望するフィールドを自由に付け加えることもできる。

② メッセージを自動処理させるルール

メッセージを構造化することで、届いたメッセージを処理するルールを作成する処理も簡単になる。例えば図8.5-2は、テキストエディタを使ってInformation Lensシステムのルールを作成しているところを示したものである。このエディタは、メッセージを作成する時に使用したものと同じ種類のメッセージタイプを利用してルールのテンプレートを作っている。そして、メッセージ作成時と同じような操作方法で、各タイプごとに、既定値、意味の表示、選択リストを選択することができる。

本システムには、現在ローカルルールとセントラルルールという2種類のルールがある。ローカルルールは、メッセージがメールサーバからユーザの使用しているワークステーションに送られてきた時点で適用される。よく使われるルールとしては、特定のフォルダへメッセージを移動（move）する指示やメッセージを消去（delete）する指示、他の人へメッセージを再送（resending）する指示等がある。メッセージの「移動」と「消去」という指示は、どちらも消去（delete）することであるが、物理的に抹消（expunge）してしまうことではない。だから、同じメッセージをいくつもコピーして異なるフォルダへ入れるという複数のルールも書けるのである。「再送（resending）」は「転送(forwarding)」と似ているが、後者が元のメッセージを全てコピーして新しいメッセージとするのに対し、前者は宛先（To）とカーボンコピー（cc）のフィールド以外をコピーして新しいメッセージを作るところが異なっている。このとき宛先（To）フィールドには、ユーザがルールに記述しておいた宛先が適用され、またルールに記述した送信者（Sender）のフィールドが付け加えられる。

フィールド間あるいはフィルド内の条件を記述する際には、ブール関数を使うこともできる。さらに、メッセージの「特徴（characteristics）」を定義するルールを使用することによって、複雑な組み合わせのルールを記述することもできる。

図8.5-2　ルール作成用エディタの例

このルールにさらに発展性を持たせるために、「LISP code」という名のルールも用意している。このルールが指定されていると、ルールが発火したときにそこに記述されたLISPの関数名が呼ばれるというものである。

一方、セントラルルールは、個々のユーザがAnyone宛のメッセージの中から見たいと思うものを指定するために記述される。セントラルルールとしては、「show」と「set characteristic」という2種類の指定が可能である。「show」を指定すると、（宛先としてAnyoneが含まれている）メッセージの宛先の中にそのユーザが含まれていない時に、そのメッセージが送られる。こうして送られてきたメッセージは、その後他の通常のメッセージと同じように、ローカルルールによって処理される。

ルールに関しては、本システムを使用してみた結果に基づき、次の点が改良されている。まず、メッセージタイプに対するルールは、ルールセットエディタに現われた順序に従って適用される。また、特定のメッセージタイプに関するルールは、一般的なメッセージタイプから継承されたルールよりも先に適用される。さらに、何らかのアクションを起こすルール（メッセージの移動や削除等）につい

ては、「特徴（characterisstic）」として定義しておく（例えば、MOVEDやDELETEDという名で定義する）。こうすれば、例えば「そのメッセージがまだ移動も削除もされていなければ」といった条件のルールを設定することができる。また、ユーザがルールを理解したり修正する際の手助けとなるように、指定したメッセージに対してどのルールが発火したかを示す履歴を見ることができる機能を用意した。

③　メッセージを返信する際の知的な補助

半構造化されたメッセージでそのタイプが認識できると、ユーザがそのメッセージを見た後でしたいと思われる選択肢を、システムが知的に判断して提供することができる。

(a)　返信するタイプの提示

メッセージに対して返事を出したり転送したりする際には新しいメッセージが作られる。そこで、本システムでは、受け取ったメッセージのタイプごとに、作成する新しいメッセージのタイプについて選択肢を提示する。例えば、ユーザが「バグフィックスの要望」というメッセージに対して「返信」の選択をしたときに、「バグフィックスの約束」、「詳しい情報の要望」、「その他」という三つの選択肢を持つポップアップメニューが現われる。ここで「約束」を選ぶと、このメッセージに答える新しい「約束」タイプのメッセージが作られる。

(b)　対応するアクションの提示

さらに重要な機能として、本システムでは、受け取ったメッセージを見てユーザが対応するであろうと思われるアクションの選択肢を提示することができる。例えば、ユーザが「ミーティングのお知らせ」というメッセージを読んだときには、通常の「返信」や「転送」等の選択肢のほかに、「カレンダに登録」という選択肢が自動的に付け加えられる。ここでユーザが「カレンダに登録」を選ぶと、メッセージの中の構造化されている項目（日付、時刻、議題等）がそのユーザのオンラインカレンダに編集されて付け加えられる。また、もう一つの例として、ユーザが「ソフトウェアリリース」というメッセージを読んだときには、「ファイルをロード」という選択肢が自動的に付け加えられる。ユーザがこの選択肢を選ぶと、メッセージの中に記述されたファイルがそのユーザのシステムに自動的にロードされる。

(4)　メッセージタイプの特徴の継承

メッセージテンプレートを作成したり利用したりする際に、テンプレートがいくつかの子タイプのテンプレートのネットワーク構造として提供されれば、もっと使いやすくなる。テンプレートの子タイプは、親テンプレートを継承することにより、その親テンプレートのフィールド名や特性を自動的に継承する。そして、各子タイプは、そのタイプ独自のフィールド名や特性を、継承した親テンプレートのフィールド名や特性に追加することができる。例えば、「セミナーのお知らせ」というテンプ

レートは、親テンプレートである「ミーティングのお知らせ」を継承し、さらに「講演者」というフィールド名を追加している。継承してネットワーク化することにより、既にあるテンプレートと似たタイプのテンプレートを作成する際に、重複する情報を記述する手間を省くことができる。また、テンプレートを体系化することになるので、メッセージの送信者は適切なテンプレートを容易に選ぶことができるようになる。

(5) アプリケーションの例

半構造化されたメッセージのアイデアが一般的であることを示すために、ここではアプリケーションの例をいくつか挙げてみる。ただし、ここでもっとも重要な点は、協調作業を支援するさまざまなアプリケーションシステムが、(i) ユーザの立場に立ってスムーズに組み合わせられること、および (ii) 構造化されたメッセージを支援する基本的な機能を利用することによってインプリメントが容易になること、の二点である。

① コンピュータ会議

コンピュータ会議システムが持つさまざまな機能は、「Information Lensシステム」に容易に組み込むことができる。まず、(a) 会議名と（オプションとして）親会議の名というフィールドを持つ「会議の開催（conference opener）」という新しいメッセージタイプを加える。また、(b) このメッセージタイプの返信用の選択肢として「参加（join）」という項目を加える。この項目が選ばれると、自動的に (i) この会議名を持つ新しいフォルダが作られ、(ii) トピックフィールドにこの会議名が記述された「Anyone」宛のメッセージを取り込むようにルールが記述され、(iii) このトピックのメッセージを新しく作成したフォルダへ格納するようにルールが記述される。

このようにして本システムは、コンピュータ会議システムが、予め定義されたさまざまなトピック（あるいはサブトピック）を用意することによってコミュニケーションの構造化を図っているような機能を実現できるのである。

② カレンダ管理

本システムは、「ミーティングのお知らせ」というメッセージに対してそのユーザのオンラインカレンダに自動的にスケジュールを登録するという選択肢を提供して、カレンダ管理の簡単なやり方を実現している。さらに、本システムでは、半構造化されたミーティングスケジュール用のもう少し複雑なプロトコルを実現した。このプロトコルでは、「ミーティングの提案（meeting proposals）」や「ミーティングの承諾（meeting acceptances）」等の新しいメッセージタイプを利用している。「提案」や「承諾」のタイプは、「ミーティングのお知らせ（meeting announcement）」タイプのフィールドを全て持っている。ユーザは、ミーティングの提案が合意するまで、提案（あるいは再提案）のやり取りを続ける。

本システムでは、次のようにしてこの処理の自動化を図っている。まず、あるタイプ（例えば、「ミーティングの提案」等）のメッセージに対して返信するため、全て他の人（例えば、秘書）に送りたいと考える人がいるであろう。本システムは、このような処理を自動化する。また、本システムは、これらのメッセージに対して返信メッセージを作成する際に支援する。例えば、「ミーティングの提案」に対して「ミーティングの承諾」を返信しようとすると、全ての内容（時刻、場所、議題等）が自動的に「ミーティングの承諾」にコピーされる。「ミーティングの承諾」を受け取った際には、そのメッセージに対するアクションとしてメッセージの「confirm（確認）」という選択肢が用意されている。この項目を選択すると、ユーザのカレンダにそのミーティングが登録され、開催時刻を確認するために、他の出席者に「ミーティングのお知らせ」メッセージが送られる。

③　プロジェクト管理と作業割付

構造化されたメッセージを利用するシステムは、誰にどんな作業を割り付けたかを把握することを支援できるので、プロジェクト管理等の調整作業の手助けとなる。本システムでは、アクションの要望とそれに対する約束のメッセージを全て特別のフォルダに格納させるというルールを用意することによって、この処理を簡単に実現している。例えば、あるユーザは、受け取ったアクション要望のメッセージを、自分の関係するプロジェクトごとに分類したフォルダに格納しているという場合もあるし、他のユーザは、自分の送ったアクション要望メッセージを、そのメッセージの送り先ごとに分類したフォルダに格納しているという場合もある。

ここではもう少し複雑な応用例として、ソフトウェアメンテナンス処理用の簡単な作業割付システムを開発した。この例では、まず、ソフトウェアシステムのユーザが作業割付担当者に対して「バグレポート（problem reports）」メッセージを送る。作業割付担当者は、ユーザに対して「了解（acknowledgement）」メッセージを返すとともに、開発担当者に対して「バグレポート」メッセージを転送する。開発担当者はバグをフィックスすると、「バグフィックスレポート（fix report）」メッセージを作業割付担当者に送り、受け取った作業割付担当者は、元のユーザに対して、バグがフィックスしたことを示す「ユーザレポート（user report）」メッセージを送る。

この応用例では、本システムは三つのメッセージタイプを定義し、新しい対応オプションをいくつか設定して実現している。作業割付担当者となったユーザには、「バグレポート」を読んだ際に取ると思われる二種類のオプションを用意した。一つは、そのメッセージに対してまだ「了解」を返していない場合に選択できる「了解」オプションである。このオプションを選択すると、「バグレポート」の内容を基にして該当するフィールドの内容が自動的に記述された標準的な「了解」が作成される。「了解」を送ると、「バグレポート」の中の「特徴（characteristics）」フィールドに「了解済み（acknowledged）」というフラグが立てられる。「了解済み」フラグが立てられた「バグレポート」を読んだ際には、「作業割付（assign）」が対応オプションとして追加される。このオプションを選択

すると、作業割付担当者が「バグレポート」メッセージを転送する際に宛先を選択できるように、開発担当者の一覧が表示される。

(6) 個々のグループに特化したメッセージタイプの定義付け

構造化されたメッセージタイプをどのように定義するかという点は重要である。各々のグループやアプリケーションで、自分達に関係のある特別の情報を扱える詳細な構造を作ることができる。例えば、商品企画チームでは、商品という面から見たさまざまなメッセージタイプ（市場規模の評価、レスポンスタイムの評価、同業者のパワー等）のネットワーク構造を作ることができる。

このアプローチの面白い点は、定義したメッセージタイプの数が増えるにつれて、これらのメッセージタイプを使う人の数も増えるという点である。例えば、最初に情報共有システムを使い始めたユーザは、全てのメッセージに共通のフィールド（To、From、Subject、Date等）だけを利用してルールを作成する。ユーザのグループが一般的なメッセージタイプのセットを使い始めると、さらに特種化したメッセージタイプを扱えるもっと複雑なルールを作成して便利にしようとする。例えば、ワークグループの個々人は、手始めに構造化されていない電子メールを使って、さまざまなミーティングをスケジューリングしようとする。このシステムを利用することが一般的になり、重要性を持ってくると、メッセージを適切に分類したり優先順位を付けることができるように、特別のメッセージタイプ（「ミーティングのお知らせ」や「ミーティングの提案」等）が定義されるようになる。そして最終的には、ミーティングのスケジュールを自動的に処理できる機能が付け加えられる。組織の原理という立場から見ると、「組織内部コード」というものが組織の生産するものの中でとりわけ重要であると言える。結果的には、本システムは、組織におけるこの一まとまりの言葉を定義、再定義できる道具であると言える。

8.5.2 Object Lens

原文タイトル： Object Lens: A "Spreadsheet" for Cooperative Work

翻訳タイトル： Object Lens：協同作業のための「スプレッドシート」

著　者： Kum-Yew Lai
Thomas W. Malone
(Center for Coordination Science, Massachusetts Institute of Technology, Mass., USA)

出　典： CSCW '88 Proceedings, September 1988, pp.115-124

本稿では、Information Lens システムのコンセプトを拡張して開発されたObject Lens と呼ばれるシステムの初期のプロトタイプについて紹介されている。Object Lens は、初心者ユーザでも広範囲の協同作業活動のための（限定されてはいるが）非常にフレキシブルで有用なコンピュータ処理機能および通信機能にアクセスすることを可能にしている。本稿では、(1) Object Lens の設計において利用したキーとなるアイデアが説明され、(2) Object Lens の機能においてそれらのアイデアがどのように実現されているかが示され、(3) このフレームワークの中で開発したいくつかの単純なアプリケーションの例（タスク・トラッキング、知的メッセージ・ルーティング、データベース検索）が紹介されている。

(1) Object Lens のキーとなるアイデア

Object Lens のもっとも重要な特性は、それが、セミフォーマルなシステムであることである。ここで、セミフォーマルなシステムとは、以下の三つの性格を持つコンピュータ・システムであると定義する。

① ある種の情報をフォーマルな形で指定された方法によって表現し、かつ自動的に処理する。

② ①の情報、あるいは別の種類の情報を、フォーマルに指定されたのではない形で表現し、人間が処理することを容易にする。

③ コンピュータによるフォーマルな処理と、人間によるインフォーマルな処理の間の境界を容易に変更できるようにする。

これらの特性は協同作業アプリケーションにおいて、特に重要である。協同作業においては、人々の行動のいくつかのパターンについては、十分理解されているけれども、それ以外に潜在的に重要ではあるが、特定することの困難な知識が大量に存在しているからである。

そのような柔軟かつセミフォーマルなシステムを開発するためには、システムの中に埋め込まれる知識は、可視的 (visible) かつ可変的 (changeable) な形でユーザに提示されなければならない。つまり、ユーザが、情報やシステムに含まれた処理ルールを見たり変更したりすることが簡単にできなければ

ならない。Object Lens においては、知識をユーザに提示する方法に関して二つの中心的なアイデアがある。

- 「受動的 (passive)」な情報は、半構造化オブジェクトとして提示される。
- 「能動的 (active)」な情報処理ルールは、半自律的エージェントとして提示される。

以下、これら二つの構成要素によって、いかに可視的かつ可変的な形で知識がユーザに提示されるかについて説明する。

① 半構造化オブジェクト

Object Lens システムのユーザは、メッセージ、人々、会合、仕事、製造された部品、ソフトウエアのバグ等、物理的あるいは概念的に親しみのある物を示すオブジェクトを作成、検索、表示することができる。この「オブジェクト指向データベース」においては、それぞれのオブジェクトは、フィールドとフィールド値の集合を含む。例えば、PERSON オブジェクトは、Name（名前）、Phone（電話番号）、Address（住所）、および Job Title（職位）等のフィールドを持ち、TASK オブジェクトは、Requestor（要求者）、Performer（実行者）、Description（内容記述）、Deadline（期日）等のフィールドを持つであろう。Information Lens システムにおけるメッセージと同様に、これらのオブジェクトは、ユーザがどのフィールドにも任意に情報を記入でき、また書き込まないことも任意であり、さらにフィールドの中の情報は、特定のタイプに当てはめる必要がない（例えば、「I don't know」のように、自由形式のテキストであって良い）という意味で半構造化オブジェクトである。

(a) テンプレート・ベースのユーザ・インタフェース

ユーザはこれらのオブジェクトをテンプレート・ベースのユーザ・インタフェースという、非常に自然な形式で見たり変更したりすることができる。このインタフェースにはいくつかの利点がある。例えば、(i) ユーザが既に馴れ親しんだフォームに似ている。(ii) オブジェクトに含まれるフィールドについての情報、および異なったフィールドの間の区別や比較等の情報をユーザに提示するのに便利である。(iii) 多くの異なった種類のフィールドを一括して整合性のある使い方が可能である。

(b) オブジェクト間の関係

ユーザは、オブジェクト間にリンクを挿入したり削除することによって容易にオブジェクト間の関係を見たり変更することができる。例えば、TASK オブジェクトの Requestor フィールドと Performer フィールドには、それぞれその仕事を要求した人と、それを実行する人を示す複数の PERSON オブジェクトへのリンクが含まれるかも知れない。その場合、例えば、ユーザが TASK オブジェクトを見る時、その仕事に関係した人間についての情報（例えば電話番号等）を得ることは、もっと簡単になるであろう。

(c) 変更可能な表示フォーマット

ユーザはいくつかのオプションを用いてオブジェクトの表示方法を変更することができる。(i) ユー

ザは、例えば、表示するフィールドを選択したり、特定のフィールドの名称を変更したりすることによって、個々のオブジェクトの表示フォーマットを簡単に変えることができる。(ii) オブジェクトの集合から特定のフィールドを選択し、表形式で簡単に表示することができる。(iii) ユーザは、オブジェクトの集合の間の関係を選択し、オブジェクト間の関係が一つの図形的なネットワーク構造として示される、ツリー形式で表示させることができる。

(d) オブジェクト間の継承階層関係

タイプ定義の作成と修正は、オブジェクトのタイプを継承階層関係の中でアレンジすることによって簡略化されている。オブジェクトの新しいタイプは、既存のオブジェクト・タイプを特定することによって定義され、特にオーバーライドされた部分を除いて、既存オブジェクトのすべての性質が自動的に継承される。新しいオブジェクトについての情報のほとんどは、毎回再入力されるのではなく、既存のタイプから継承されるので、新しいオブジェクト・タイプの作成は簡単になる。また、後になってオブジェクト・タイプの定義を変更する場合にも、その修正は自動的に下位のオブジェクト・タイプに継承される。

② 半自律的エージェント

Object Lens システムのユーザは、他のユーザのために自動的に情報を処理するルールベースの「エージェント」を作成することができる。これらのエージェントは、システムが自動的に実行するタスクを分離するための自然な方法を提供している。エージェントは、新しいメールの到着、指定したフォルダ内への新しいオブジェクトの出現、事前に指定した時刻の到来、他のエージェントからの陽表的な起動、ユーザによる陽表的な選択等のイベントによって起動される。起動されたとき、エージェントは、指定されたオブジェクト集合に対して、ある一まとまりのルールを適用する。あるオブジェクトがルールに指定された判定基準を満たすとき、そのルールは何らかの事前に指定された処理を実行する。これらの処理は、検索、分類、メールの発送、オブジェクトの削除のような一般的な処理である場合もあるし、ファイルをロードしたり、カレンダにイベントを追加する等、特定のオブジェクトに対応した処理である場合もある。

Object Lens のエージェントが「自律的」であるのは、それらが一度作成されたならば、人間のユーザが特に注意を払わなくても、処理を実行することができるからである。ただし、それらが「半自律的」であるに過ぎないというのは、(i) エージェントは常に人間のユーザによって制御されていること（つまり、人間のユーザはそのすべてのルールを容易に見ることができるし、変更することができる）(ii) エージェントは、独自の処理を実行するよりも、人間のユーザにオブジェクトを提示し、処理を求めることがしばしばであること、等の理由によるのである。

エージェントもルールもそれ自体がオブジェクトであるから、ユーザは他のすべてのオブジェクトに対して利用されるのと同じテンプレート・ベースのユーザ・インタフェースを介してそれらを見た

り修正したりすることができる。与えられたオブジェクトに対してあるルールが何時発火すべきであるかの判定基準を指定するためには、ユーザは、ルール適用対象オブジェクトの「記述 (description)」を作成する。記述とは、特定のタイプのオブジェクトのための部分的に記入済みのテンプレートである。また、記述には「埋め込まれた記述 (embedded description)」を含めることができる。これは、もとのオブジェクトがリンクされたオブジェクトによって満足されなければならない特性を指定するものである。例えば、Task の記述には、そのタスクを実行する Person についての「埋め込まれた記述」が含まれる場合があるであろう。

(2) システムの特徴

Object Lens システムは、Ethernet によって接続された Xerox 1100 シリーズ・ワークステーション上に、Interlisp-D によってインプリメントされている。このシステムは、Loops の提供するオブジェクト指向プログラミング環境を活用して開発された。基本的なメール処理機能が開発グループの 3 人よって 3ヶ月間本格的に利用されている。

① インスタンスの編集

ユーザは、内蔵されたテキスト・エディタを利用してインスタンスのテンプレートの編集を行う。ユーザは、任意のフィールドに値を挿入したり、変更することができる。さらに、ユーザがマウスであるフィールドの名称部分をクリックすると、そのフィールドの値の候補のリストがポップアップ・メニューで表示される。これらの候補は、他のオブジェクトへのリンクである場合もあるし、単なるテキスト・ストリングである場合もある。これらの候補の一つを選択すると、その値が自動的にそのフィールドに挿入される。候補リストに現れていないオブジェクトへのリンクを挿入することも可能である。

また、あるテンプレートに、別のオブジェクトのテンプレートを埋め込むこともできる。この埋め込まれたテンプレートのフィールドは、テンプレートの他のフィールドとまったく同様に編集することができる。

② 新しいインスタンスの作成

ある既存のオブジェクト・タイプの新しいインスタンスを作成し表示するには、ユーザは、マウスでそのオブジェクト・タイプのテンプレートをクリックする。そうすると、該当するオブジェクトのインスタンスが画面に表示され、ユーザは、それに値を記入することができる。

③ 新しいオブジェクト・タイプの作成

新しいオブジェクト・タイプを作成するには、ユーザは、親オブジェクト・タイプに対応するテンプレートをクリックする。これによって、テンプレートへの操作の選択枝を示すメニューが現れる。このメニュー項目の中の「Create a subtemplate （サブテンプレート作成）」を選択すると、親テンプレ

ートのすべてのフィールドと性質を持った新しいテンプレートが作成される。その後、ユーザは、この新しいテンプレートにフィールドを追加したり、その表示フォーマットを変更したり、その他の性質を変更したりできる。

④ オブジェクト・タイプの表示フォーマットおよびその他の性質の変更

オブジェクト・タイプの表示フォーマットおよびその他の性質を変更するには、ユーザは、そのオブジェクト・タイプを定義するテンプレートを編集する。例えば、ユーザは、以下の変更を行うことができる。(i) オブジェクトのどのフィールドを実際に表示するか、(ii) 表示されるフィールドの名称、(iii) それぞれのフィールドに表示されるべき値の候補。また、ユーザは、それぞれのフィールドに対して、そのフィールドにおけるリンクがリンク・アイコンによって表示されるべきか、あるいは埋め込まれたテンプレートとして表示されるべきかを指定することができる。

⑤ エージェントとルールの作成

エージェントは、ルールの集合および、いつどこでそれらを適用するかの指定から構成される。ルールは、それを満足するオブジェクトの記述を含み、それらのオブジェクトに対して適用する処理を指定する。ルールには、IF フィールド（条件部）とTHEN フィールド（処理部）がある。ルールのこの両方の部分は、他のオブジェクトへのリンクを含んでおり、これらのリンクは、埋め込まれたテンプレートとして示されている。ルールの IF 部は、「記述」であり、これは、フィールド値によって一連のインスタンスを記述する特殊なテンプレートである。ルールの THEN 部は、Action オブジェクトである。

ルールの IF 部を作成するには、ユーザは、(i) IF フィールドをクリックし、(ii) 提示されたメニューから「Descriptions」を選択し、(iii) 提示されたオブジェクト・タイプのツリー・フォーマットから目的のオブジェクト・タイプを選択する。これによって、適当なタイプの記述をルールの中に、埋め込まれたテンプレートとして、挿入することができるようになり、ユーザは、この記述の中のフィールドに記入することにより、オブジェクトがルールを満たすために特定のフィールドに現れるべき値を指定することができる。Information Lens の場合と同様に、文字列を and、or、not、あるいはカッコによって組み合わせることによって、フィールド値に対するもっと複雑な指定を作成することもできる。

ルールの THEN 部を指定するには、ユーザは単に THEN フィールドをクリックし、そのとき提示される処理の候補リストのメニューから選択するだけでよい。いくつかの場合には、ユーザは埋め込まれたテンプレートの中のいくつかのフィールドに記入して処理を指定する必要がある（例えばオブジェクトをどこへ移動するか等）。

(3) アプリケーションの例

① タスク・トラッキング

協同作業アプリケーションにおいてしばしば指摘される機能は、人々の仕事の進行度合いをトラッキングする能力である。例えば、このようなシステムは、「他の人々が自分に要求した仕事は何か」「それらの中で期日を過ぎているものはないか」「自分は他の人々にどのような仕事を要求したのか」等の質問に答えることができる。

Object Lens において、このような機能をサポートすることは、簡単である。例えば、このシステムには、既に、行動の要求や約束に対応したオブジェクト・タイプが含まれている。Information Lens においてさえも、これらのメッセージを自動的にソートして、誰がその仕事をすることになっているか、どのようなプロジェクトを含んでいるか、等によって分類してフォルダに入れることが可能であった。しかし、Information Lens においては、フォルダの内容の要約表示は、From、Date、Subject 等の標準的なメッセージ・ヘッダ・フィールドしか表示できなかった。タスクについてのより多くの情報を見ようとすれば、個別のメッセージを一つづつ表示しなければならなかった。Object Lens においては、ユーザの選択した任意のフィールドを表示することによって、フォルダの中のメッセージを簡単に要約することができる。

② 知的なメッセージ・ソート：技術的変更通知

協同作業問題における興味ある例の一つは、製品設計仕様における変更の情報を組織の中の適切な人々に伝達する問題（しばしば技術的変更通知と呼ばれる）である。Information Lens においても、技術的変更通知を、「part affected （影響される部品）」、「type of change （変更のタイプ）」、「severity （重要度）」等のフィールドの内容によってソートすることが既に可能であった。Object Lens においては、追加の知識を利用してもっと知的なソートを行うことが可能になった。

③ データベース検索

個人としての仕事においても、協同作業においても、データベースからある種の条件を満たすオブジェクトを自動的に検索する機能が役立つ場合は多い。Object Lens においては、このための単純な方法を提供している。ユーザは単に、一つのフォルダの中のオブジェクトをスキャンして、選択されたオブジェクトへのリンクを別のフォルダに納めるエージェントを作成するだけでよい。このエージェントのルールは、オブジェクトを選択するための判断基準を指定する。

Object Lens では、メッセージの受け手を決定するための特別の機能をインプリメントした。この機能によれば、「記述」をメッセージの To フィールドや、cc フィールドに埋め込むことができる。こうすれば、メッセージが送られたとき、これらの記述は知識ベースの中のすべての Person オブジェクトに自動的に適用され、検索された人々が To フィールドや、cc フィールドに挿入される。この機能によって、メッセージの送り手は、メッセージの送信時に、データベースの中の人々に関する現在の

情報に基づいて、配布リストをダイナミックに計算し、作成することができる。

④ ハイパーテキスト

Object Lens においてハイパーテキスト・システムの機能を利用することは簡単である。例えば、本システムには Text というオブジェクト・タイプがある。これは Name と Text という二つのフィールドだけを表示する。Text オブジェクトの Text フィールドは、他のオブジェクトへのリンクを必要なだけ持つことができる。

ハイパーテキスト・システムの通常の利点に加えて、Object Lens では、ハイパーテキストを他のデータベース、メッセージング、および計算能力と統合しており、さらに有用なシステムとなっている。例えば、ハイパーテキスト・システムの別のノードへのリンクを挿入するには、ユーザは、まずリンクを張るべきノードを見つけなければならない。これまでのハイパーテキスト・システムにおいては、すべてのノードはリンクを追いかけることによって人手で発見しなければならなかった。Object Lens においては、上記のデータベース検索機能を利用して、ある種の基準を満たすオブジェクト（人々や、履歴情報等）を自動的に発見することができる。その後、それらのオブジェクトへのリンクをテキストに挿入することができる。

8.5.3 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。

8.6 グループエディタ

8.6.1 Quilt

原文タイトル： Collaborative Document Production Using Quilt

翻訳タイトル： Quilt を使用した協同文書作成

著　者： Mary D.P. Leland
Robert S. Fish
Robert E. Kraut
(Bell Communications Research, Inc., Morristown, NJ, USA)

出　典： CSCW '88 Proceedings, September 1988, pp.206-215

本稿では、文書の協同執筆を支援するコンピュータ・ベースのツールとして開発されたQuilt について紹介されている。まず、協同執筆を支援するために必要な要素について述べられ、つぎにQuilt システムの概要が示される。そして、協同作業における Quilt システムの利用法を示す一つのシナリオを提示して、Quilt システムの機能と制限が解説されている。

(1) 協同執筆を支援するための要素

協同執筆や文書は非常に広い意味で捉えることができる。共著になる文書は非常に多数に上り、たとえ最終版の著者名が一人であっても、その文書が協同作業の成果であるケースは多い。この協同作業は、同僚にコメントをメモしてくれるように頼む非公式なプロセスから、より構造化された規則に基づく審議や承認のプロセスまで、実に様々な形を取り得る。後者のプロセスは、しばしば、提案書、論文、仕様書、その他の「公式」文書を発行したり、出版したりするときに必要とされるものである。このように文書の種類や協力の程度は様々であるが、そこでは必ず協同執筆者間のコミュニケーションが必要となる。協同執筆者は文書の内容を読み、それにコメントを付け加えることができなくてはならないし、また、文書を効率よく完成させるためには、調整、計画、情報の共有等が不可欠である。例えば、文書の校閲はタイムリに行う必要があるし、この校閲がいつ終るかを編集者や執筆者たちに通知して、彼らが効率よく必要な訂正を行えるように取り計らう必要がある。また、これらのコミュニケーションは、文書の作成に協力者が果す社会的役割に応じて、適切な人々だけがアクセスできるように保護されなければならない。さらに、協同執筆者間に良好で生産的な作業関係を保つためにも、コミュニケーションは欠くべからざるものとなる。

(2) Quilt システムの概要

Quilt は、ハイパー・テキストと直接操作インタフェースの概念を導入し、協同執筆の社会的局面に

関する研究に基づいて設計・開発された、協同文書作成作業の幾つかのコンポーネントを支援するシステムである。特筆すべきは、Quiltが、テキストまたは音声のどちらかを使って、文書に注釈を付ける構造化されたメカニズムを提供することである。デフォルトの注釈タイプとしては、訂正の提案、公のコメント、宛先指定メッセージ、プライベート・メッセージ等が用意されている。協同執筆者たちは、必要に応じて、このデフォルトのセットを拡張することができる。この他、協同執筆者たちは、作成者や作成日時等を初めとして、様々な属性を注釈に定義することができる。これらの属性は、文書やその注釈のビューを選択するための述語を定義するときに使用できる。また、Quiltには、協同作業を円滑に調整するためのメカニズムとして、アクティビティのログ、通知、および、トリガの諸機能が用意されている。アクティビティのログには、利用者が文書に対して行った活動が、リテラルなマシン生成レベルと、より抽象的な人的生成レベルの両方で記録される。また、通知メカニズムが電子メールと注釈プロセスを統合しているので、協同執筆者は、文書に直接アクセスしなくても、彼ら宛の注釈を見ることができる。さらに、協同執筆者は、トリガ・メカニズムを使って、所定の述語が適用されたときにQuiltに実行させる特定の動作を定義することができる。

Quiltのインプリメントは、比較的一般的で入手の容易なソフトウェアを使用することを前提としている。協同作業のアイテム、アイテム間のリンク、および、Quiltの協同執筆者および利用者に関する情報を格納するための基礎的なデータベース・システムとしてOrionが利用されている。また、Quiltのユーザ・インタフェースとしては、X WindowとXt Tool-kitが使用されている。

(3) Quiltの使用例

Quiltの機能を解説するために、数人の人々が、Quiltを使って、一つの文書を計画し、作成し、訂正し、吟味し、読むケースを想定して一つの潜在的なシナリオを説明して行くことにする。このシナリオには、二人の協同執筆者AnneとBillが登場する。この二人は、会議で数回顔を合わせたことがあり、Log Cabinという名称の文書作成プロジェクトを協力して実行することにした。彼らは同一の機関に属している訳ではないので、頻繁に会うことは難しい。しかし、彼らは共にQuiltを動作させるマシンにネットワークを介してアクセスすることができる。この協同作業を行っている間、彼らはプロジェクトの様々な局面について討論し、彼らが書こうとしている論文のアウトラインを決定し、草案段階の原稿をオンラインでやり取りしたり、互いの原稿にコメントを付けたり、原稿を訂正したりしながら、個別に著作作業を進め、最後に彼らの原稿を一つにまとめなければならない。さらに、論文を発表する前には、Billのボスであるの承認を受けなければならない。Cherylのコメントに基づいて、彼らはさらに検討と訂正を重ね、論文をCherylの要求水準を満たすものに仕上げなければならない。

予備的な計画作業の大部分は、Anneの大学で、二日間に渡って、AnneとBillが一日中顔を付き合せて会議を行ったことで完了できた。しかし、内容や手順の詳細の多くがこの二日間の会議では決定

し切れなかったので、彼らはQuiltを使ってこれらの問題点について討議することにした。この討議のテーマはすべてQuiltシステムの中の文書となり、AnneとBillは両方とも、これらを読み、訂正し、これらに注釈を付けることができる。この初期段階で、Quiltは、伝統的な会議システムに似たコンピュータ・ベースの同期会議システムとして使われた。しかし、Quiltには優れた注釈機能があるので、討議で出されたアイディアを、従来のように間接的に関連ポイントを参照して指示するのではなく、直接的に関連ポイントに付加することができる。この直接操作によって、会議の筋道を非常に容易にたどることができるので、AnneとBillはこの論文の協同執筆を継続できると判断した。

ここで、AnneとBillが展開した協同執筆作業のアウトラインをたどり、Quiltの下でどのような展開を経て最終文書ができ上がったのかを見てみることにする。まず、AnneはQuiltにLog Cabinプロジェクトの協同作業をセットアップした。Anneは、彼女自身とBillと彼女のStudentsを初期メンバとしてリストした。StudentsはQuiltの利用者名であり、ここでは論文を読んで注釈を付けることができるAnneの生徒を意味する。つぎに、Anneは協同作業のスタイルを選択した。協同作業のスタイルによって、文書に付けることができる注釈のタイプや、協同執筆者が果す社会的役割が決定される。このスタイルは、次に示す様々なスタイルの中から選択することができる。

① 論文の協同執筆のためにQuiltに予め用意されているスタイルのセット
- 排他的スタイル　－　その章の著者だけがそれを修正できる。
- 共有スタイル　－　共著者はどの章でも修正できる。
- 編集者スタイル　－　任命された編集者だけがすべての章を修正する権利をもち、他の共著者は編集者に提案を行う権利しかもたない。

② Anneが既に定義している数種類のカスタム・スタイルのいずれか
- Anneが彼女の協力者のために既に定義しているデフォルトのスタイル。
- Anneが既に定義しているもう一つのスタイル　－　協力者全員がすべてのアイテムを修正、付加、または、削除できる。

③ Anneが、この協同作業のために、これから新たに定義する新しいスタイル。

ここで、Anneは、協同執筆者が互いに著作部分を交換できる「共有」スタイルを選択した。

表8.6-1に「共有」スタイルの表を示す。この表は、コメントのタイプ（基本文書、訂正の提案、公のコメント、宛先指定メッセージ、プライベート・コメント、ヒストリ）別に作業を行うことができる役割を示している。コメンテータは読み手に与えられるすべての権利と認可（並びに、認可テーブルで許諾されたその他の権利）を持っている。同様に、共著者は少なくともコメンテータの権利をすべて持っている。最後に、Anneは、メンバの一人一人に、選択したスタイルに応じて定義された社会的役割を一つまたは二つ以上割り当てた。この結果、AnneとBillはLog Cabin協同執筆プロジェクトの共著者となり、Anneのすべての Studentsはこのプロジェクトのコメンテータとなった。

表8.6-1　共同作業における役割と権利の定義

	基本文書	改訂版の提案	公のコメント	宛先指定メッセージ	プライベートコメント	ヒストリ
作成	共著者	共著者	コメンタ	コメンタ	読み手	共著者
修正	共著者	作成者 共著者	作成者	作成者	作成者	——
削除	共著者	作成者 共著者	作成者	受信者 作成者	作成者	——
改訂版の追加	コメンタ	コメンタ	コメンタ	作成者 受信者	作成者	——
コメントの追加	コメンタ	コメンタ	コメンタ	作成者 受信者	作成者	——
メッセージの追加	コメンタ	コメンタ	コメンタ	作成者 受信者	作成者	——
プライベートコメントの追加	読み手	読み手	読み手	作成者 受信者	作成者	——
読み込み	読み手	共著者	コメンタ	作成者 受信者	作成者	許可を受けた者のみ

Anne と Bill は同一機関に所属していないので、互いの存在を刺激としてプロジェクトを進めることができない。そこで、彼らは、作業の調整と完成を助けるために、Quilt の通知およびトリガ・メカニズムを利用することにした。彼らは、パートナが文書に大きな変更を加えたときはいつでもメッセージを送信し、論文の提出期限が近付くにつれて間隔を狭めて頻繁に「注意を喚起」するように Quilt に指示した。Anne は、彼女と Bill が彼らの原稿のために開発したアウトラインを Quilt に入力した。これは大きな変更に当たるので、Quilt はその旨を電子メールで Bill に通知した。一方、このメールを読んだ Bill は、Quilt のリンクを伝ってアウトラインにたどり着き、彼が妥当だと考える変更を加えたり、彼が Anne の意図を理解できない箇所を示すコメントを付け加えたりすることができる。Anne と Bill は、二つのメディアを使って、討議を続け、アウトラインを洗練した。メディアの一つは Quilt である。彼らは Quilt を使って、現在のバージョンのアウトラインを読み取り、アウトラインに変更を加え、互いに照会し合い、訂正の意図を説明した。もう一つのメディアは電話である。彼らは、電話を使って、現在のバージョンのアウトラインに変更を加えた。討議を行う度に、Anne はアウトラインを修正し、アウトラインに付属しているアクティビティ・ログに修正の理由を記録した。さらに、Anne は、彼女が討議中に考えついたアイディアを、彼女自身に宛てた音声コメントの形で、アウトラインの注釈として記録することができる。Quilt と電話を使ってコミュニケーションを行い、かなりの討議

を重ねた結果、彼らのアウトラインは、まだルースな形ではあったが、再編成され、どちらがどの章を書くかについて合意に達することができた。彼らは訂正済みのアウトラインをQuiltに格納し、論文の中の各自が分担する章をLog Cabinの各アイテムとして作成した。

Quiltの利用者に与えられている認可のレベルに応じて、利用者は、(i)アイテムを読むことができる、(ii)アイテムを読み、注釈を付けることができる、または、(iii)アイテムを読み、注釈を付け、修正することができる。その他の動作（アイテムを標準テキストに変換する、アイテムを削除する、アイテムの名称を付け変える、アイテムをコピーする等）も、利用者が得ている認可のレベルに応じて、実行することができる。

実際に論文の作成を進めている間、AnneとBillは、各自が分担する章に修正を重ね、文章の洗練に努めた。例えば、Anneはテキスト・エディタを使って彼女の章のテキストを編集した。利用者とエディタの間にモニタ・プロセスを挿入すれば、Quiltでテキスト・エディタを使用することができる。このプロセスは、注釈リンクへのQuiltの内部参照を、人間が読み取れる形に変換する。論文作成中にも、Anneは彼女のテキストに注釈を付けることができる。このケースの注釈の一つのタイプは、例えば「この参照をチェックせよ」というような自分自身への（パーソナルな）コメントである。この種の自分自身へのコメントは、目的を達したら、削除すればよい。もう一つのタイプは、例えば「ここでもっと詳しく説明したほうがよいだろうか？」というようなBillへのノートである。この種の注釈は、Quiltによってオリジナル・テキストにリンクされ、電子メールでBillの元へ送られる。各ワーク・セッションが終るごとに、Quiltは、BillまたはAnneが実行したコマンドに関する基本情報と、彼らが見た文書や文書に加えた変更に関する情報をアクティビティ・ログに記録し、彼らが行った作業を文書化するように求めるプロンプトを表示する。

AnneとBillは協同作業のスタイルとして「共有」を選択しているので、BillはAnneのテキストを直接変更することができる。選択したスタイルによっては、Billの権限がAnneのテキストにコメントを付け、訂正を提案するだけに制限されることもある。Quiltは、共著者が互いのテキストを変更する際に、共著者によって選択された規則を強制的に適用する。共著者の一人が訂正を提案した場合、Quiltを利用して、二つのバージョンの相違を容易に調べることができ、改訂バージョンを交換することができる。一つの章にアクセスしたBillは、その章のテキストを読む傍ら、公の注釈とAnneからBillに宛てたメッセージを読み取ることができる。文書を修正する場合、Billはこれらの注釈のどれにでも応答することができ、さらに、基本文書またはその注釈のどれにでも彼自身のコメントを付け加えることができる。

AnneとBillは論文をCherylに見せる段階に達したと判断したので、Cherylをコメンテータとして協同作業のメンバ・リストに追加し、Cherylに文書の承認を求めるメッセージを送った。このメッセージはCherylの電子メールに表示され、Cherylはそこから文書にアクセスし、文書を審査することがで

きる。Quilt は、Cheryl が文書を読んだり、公の注釈や Anne と Bill から Cheryl 宛てに送信されたメッセージを読んだりすることを許可する。Cheryl はコメンテータとしての認可しか得ていないので、Anne と Bill がやり取りしているプライベート・コメントを読むことはできないし、文書のテキストを変更することもできない。論文を読み終った Cheryl は、Quilt の中で幾つかのプライベート・ノートを作成して、総合的な感想を記録し、幾つかの訂正を提案した。Cheryl は、いくつかのテキスト・コメントを付け加えて、二三のミスを指摘し、参照資料を提示した。しかし、音声でコメントを述べた方がより詳しく説明でき、理解されやすいと判断した Cheryl は、応答の大半を音声コメントとして記録することにした。

Cheryl が論文に関するコメントを作成し終えると、Quilt はAnne と Bill に Cheryl のコメントが入手可能になったことを通知する。また、Cheryl は、Anne と Bill がCheryl の注釈の一つ一つを処理し終わったときにその旨を記録し、すべての注釈の処理が終ったときにその旨を Cheryl に通知するようなトリガの設定を Quilt に要求した。Anne と Bill は Cheryl の注釈を吟味し、これに応えて論文を適宜訂正した。あるケースでは、彼らは Cheryl に宛先指定メッセージを送って、コメントの内容を詳しく説明するように求めた。Cheryl は（電子メールを介して）直ちにこのメッセージを読み、即座にこの要求に応えた。しかし、Cheryl は、Anne と Bill が一つの注釈を処理するたびに、これに煩わされることはなかった。Cheryl がセットアップしておいたトリガのお陰で、Cheryl は彼女が行った提案の総合的な効果を見ることができた。このような再検討プロセスを数回繰り返した後、Cherylは論文の発表を承認し、Anne と Bill は論文を雑誌に提出した。

(4) Quilt における制限

Quilt は、協同作業に係わるすべての問題の解決を目指している訳ではない。まず第一に、Quilt は協同作業の開始を支援するものではない。協同作業は、通常、非公式なコミュニケーションを重ねている内に開始されるものであり、物理的に近接しているところで発生する。しかし、Quilt の会議システムは、通常協力者たちが初期計画を立てるために集って行う会議を補うことができる。協同作業の社会的足場が固まってしまえば、協同執筆プロジェクトの強力な要素としての役割を Quilt に期待することができる。

第二に、Quilt は同期コミュニケーションのためのメカニズムを提供しない。協力者たちは作業を進めながら互いに直接話し合い、話し合いながら、彼らが協同で執筆している Quilt 文書を拾い読みしたりする。現行のプロポーザルでは、協同執筆者たちは、恐らく Quilt をオンラインで使いながら、あるいは、Quilt のプリント出力を手にもちながら、実際に集って会議をしたり、電話を使ったりして、このタスクを遂行することになる。

第三に、Quilt は、理念、思考、調査結果等を獲得し、これらを文書にまとめるタイプの著作システ

ムとしては設計されていない。ただし、Quiltの中で、これらのシステムを文書エディタとして使用することは可能である。

最後に、Quiltの基盤になっているデータベースは現在のところ集中制御されるので、地理的に遠く離れた協同執筆者たちはQuiltを使用するために同一マシンへのアクセス権をもたなければならない。

8.6.2 PREP

原文タイトル： Issues in the Design of Computer Support for Co-authoring and Commenting

翻訳タイトル： 協同執筆およびコメント付けのコンピュータ支援の設計に関する問題

著　者： Christine M. Neuwirth
David S. Kaufer
Ravinder Chandhok
James H. Morris
(Carnegie Mellon University, Pittsburgh, PA, USA)

出　典： CSCW '90 Proceedings, October 1990, pp.183-195

本稿では、協同執筆とコメント付けの関係を調査するために開発されたPREP エディタのプロジェクトについて紹介されている。このプロジェクトでは、協同執筆およびコメント付けに対するコンピュータ支援を設計するに当たって次の三点が重点項目とされた。すなわち、①共著者とコメンテータとの間の社会的インタラクションの支援、②協同執筆と外部コメントにおける認識面の支援、③この二種類のインタラクションの実際的な面での支援である。本稿では、これらの重点項目のそれぞれと取り組むためにPREP エディタが採用した方式について説明されている。

(1) 協同執筆およびコメント付けの支援に関する問題点

① 協同作業の社会的局面の支援

一般に、我々は長期間の協同執筆関係の社会的局面についてはほとんど何も知らないも同然と言える。現在のところ、協同作業の社会的局面に関する知識は非常に限られているが、協同作業の社会的局面の一つとしての調整作業の問題については、ある程度の理解を得ている。調整の問題は、「著者が、単独ではなく、協同で著作作業を進めるに当たって、どのような活動をする必要があるか？」という疑問に対する答えとして発生してくる。

(a) 社会的役割の定義の支援

調整の問題に対する一つの答えは、社会的役割の定義を支援することである。様々な協力者の「適切な職務分担」（責任、対話のパターンなど）を規定することによって社会的役割を定義すれば、調整の問題は軽減される。Quilt システムはこの戦略を採っている。Quilt では、三種類の社会的役割、六種類のオブジェクト、一セットの動作を定義している。これらを定義したことで、Quilt の中では、様々なオブジェクトに対して、どの動作をどの役割が実行できるかを指定できるようになった。Quilt は、さらに、協同作業のタイプを三種類（排他タイプ、共有タイプ、編集者タイプ）定義している。利用者にタイプを定義する機能を与えたことは、Quilt の特筆すべき特色である。この機能がなければ、「著者」とか「コメンテータ」とかいった抽象的な概念に対して、オブジェクトに実行可能な動作を

定義する際に多くの問題が持ち上がったと思われる。

社会的役割の定義を支援するシステムには、潜在的な問題がある。社会的役割の定義が「未熟」であると、予期せぬ結果を招く恐れがある。例えば、プロジェクト開始直後には、誰がプロジェクトに「大きな貢献」をするか常に分かっているとは限らず、したがって、誰が著者としての役割を取るべきか分からないことが多い。しかも、開始時点で著者を定義してしまうと、「著者になれなかった」メンバの意欲が削がれ、この人はあまり貢献しなくなるかも知れない。したがって、執筆の社会的局面についてさらに研究が必要があるのと同じように、社会的役割を定義するシステムを実際に使用する著者たちについても、さらに研究を深める必要がある。

(b) 計画に関するコミュニケーションの支援

執筆作業の分担についての話には、論文の計画が出てくることが多いが、著者はプログラムのような感覚で計画を「実行」する訳ではない。その代わり、著者は執筆中に実行する事柄を決定するための資源として計画を使用する。このプロセスでは、部分的に完了した成果が重要な役割を果すケースが多々ある。部分的に完了した成果がタスク環境の一部となり、設計の後続過程を拘束するのである。さらに、著者たちは、言いたいことが見えてくるにつれて、彼ら自身に新しい目標を設定する。著者が単独で執筆活動を進める場合、彼らが自らに課した制約や、彼らが設定した目標を調和させる必要はないが、共著の場合には、彼らが設定した目標に関する見解を練り直したり、目標に見合った成果を得るために、しばしば、制約について話し合う必要がある。さらに、展開し続ける計画や制約についてコミュニケーションをもつことで、共著者やコメンテータが他人の計画を推測する必要がなくなるので、協同作業の性能が向上する可能性もある。

本プロジェクトの方式では、計画が執筆を制御しないことや、実際、執筆に先立って完全な計画を立てるのは不可能であることを受け入れた上で、その代わりに、計画に関するコミュニケーションの支援に努力を集中している。

(c) コメント付けに関するコミュニケーションの支援

著者たちはコメントを理解せず、自分のテキストに問題があると考えるよりは読者の読み方が混乱していると受け取りがちで、コメント間に一貫性が欠けていたり、相い反するコメントを受け取ったりするとイライラする。著者とコメンテータの関係の問題は、著者が複数の読者にコメントを要請した場合に、一層急を要する問題になる。著者とコメンテータの関係について、経験から発せられた興味深い疑問は、著者グループが押し寄せるコメントをどのように管理し、有効利用すれば最適であるかというものである。

本プロジェクトでは、コメンテータ間のコミュニケーションにたくさん見られる混乱や困難を許すコメント方式を採用している。コメンテータ間のコミュニケーションを促すシステムを提供することによって、コメンテータ間のインタラクションがより有用で活発なものとなるよう期待している。

② 協同執筆における認識面の支援

(a) 執筆における認識的活動に対するタスク特定型支援

協同作業で行われるタスク特定型の活動（メモ書き、製図、執筆、身振りによる意思表示など）を理解して、これらのタスクを支援する特殊なツールを設計しようという試みは、これまでに幾つか行われているが、執筆の実質的作業と作成された草稿とを同等視する傾向があるため、ほとんどのテキストコメント用ツールは作業中のドラフト（草稿）のアウトラインについてのコミュニケーションだけを支援している。しかし、経験豊富な著者は、テキスト自体とは直接関係ない中間的な外部表現を作成するのが普通である。アイデア間の概念的関係を差し示す矢印、ボックスなどの図形を作成する機能や、詳細部の可視表示を抑止する機能をサポートしていない環境で働く場合、フラストレーションを感じたり、重要な計画活動が省略されたと感じたりすることがあると、著者たちが報告している。著者たちは、通例、この種の中間表現を迅速に作成（および廃棄）することを好み、これらの中間表現を、自らの論旨を把握するため、または自らの論旨と他の共著者の論旨を調整するための、ほんの一時的な「スケッチ」、「いたずら書き」、または、「落書き」として使用することが多い。また、彼らは、ドローイング・エディタはこの種の「調整」機能に使用するには速度が遅過ぎ、結果としてこれらを調整のためには使用していない、と報告している。

執筆プロセスを研究し、コメントツールを使用している著者たちを観察したところによれば、著者が執筆を進める際に行うメモ書き、製図、ノート取りの支援などの認識的な問題は、執筆において特に重要であり、協同執筆およびコメント付けのためのコンピュータ支援を設計するときには、是非ともこれを考慮にいれなければならない。

外部コメンテータは、通常、作業中の草稿の作成に至る非公式な計画に関与していない点が、共著者と異なる。さらに、外部コメンテータは既存の作業を行った草稿や未来のいかなる草稿についても所有権を主張することができない。にも係わらず、草稿よりも執筆プロセスの方をより強力に支援するツールを利用して、外部コメンテータと著者グループとの潜在的な新しい様々な関係を定義することができる。本プロジェクトの重要な目標は、外部コメンテータが共著者に対して果たす興味深い役割の調査を手助けする一セットのツールを構築することにある。この興味深い役割の中で最も重要と思われる点は、(i) 計画対象へのコメンテータのアクセスを可能とすること、(ii) コメンテータによる「著作テスト」を可能とすること、(iii) コメンテータに「著作テスト」の実行をリクエストまたは要求すること、の三点である。

(b) 協同執筆に係わる認識的活動に対する支援

協同執筆に関連した認識的活動は非常にたくさんあるが、ここではその中の一つであるコメント付けへのアクセスだけを採り上げることにする。ほとんどのテキストコメントシステムは、ハイパーメディア・モデルに基づいている。これらのハイパーメディア・システムは、主要な情報アクセス方式

として、ノードからノードへとリンク・アイコンをたどる方式を採用している。利用者は、通常、ノードをスクリーン上に提示し、その内容を読み、いずれかのリンクに着目し、その後、幾つかのリンクを選択的にトラバースする。この種の集中的なリンクの追跡はブラウジング作業には適しているが、その他の作業には問題が多い。ハイパーメディア・システムの誘導リンキング・システムを利用者の執筆ニーズに合わせてカスタマイズする研究を進めている研究者もいるが、まだアクセスの問題が未解決のままになっている。

本プロジェクトの方式においては、プログラムを利用者の認識的活動とマッチするようにカスタマイズする必要がある。

③ 協同作業の実際的な面に対する支援

著者用のコメントシステムのほとんどは、基本的に、すべての協力者が同一のテキストを見られるものと仮定している。この仮定は、作業グループのメンバが遠隔地で作業を進めているときには実際的でない。多くの学術的協同執筆グループは遠く離れて作業を行っているので、遠隔協同作業の実際的な障害について心配する必要がある。協同執筆を実際に行うに当たっては、互換性と浸透性の二つが重要な問題となる。

(a) 互換性

協同作業のコンピュータ支援を行う既存システムの多くは、原稿へのマルチ・ユーザ・アクセス機能を採り上げているが、互換性のないシステムからのアクセスの問題を解決しているシステムはまだない。しかも、潜在的な協同作業資源（時間、金、コンピュータ・コンサルタントやプリンタなどの支援環境）がネックとなって、この種のシステムはなかなか採用されにくい。

(b) 浸透性

協同作業のための現在のコンピュータ・ツールは比較的浸透性が低い。つまり、電子テキストとハード・コピーの間の境界を越えるのが難しい。この結果、システムの有効利用や作業習慣に対する要求と、利用者の要求の間にミスマッチが存在する。例えば、研究者が飛行機で旅行している間にハード・コピー原稿にコメントを入れるのは、そうめずらしいことではない。しかし、忙しい研究者が帰宅してから、電子的な形でこれらのコメントを入力する余分な手間を喜んで引き受けるとは思えない。近い将来、参加者がハード・コピーの入出力をあまり必要としなくなるとはとても思えないので、この問題を解決し損った場合には、互換性の問題に失敗したのと同じような否定的結末が予想される。したがって、協同作業を支援する次世代のコンピュータ・ツールは、システムの浸透性を高める必要がある。すなわち、電子媒体からハード・コピーへ、ハード・コピーから電子媒体への移動にかかる利用者の手間を最少限度まで減らす必要がある。

(2) PREP エディタ

PREP エディタは、MacApp が動作する Macintosh II の上にインプリメントされている。また、学生たちがロー・エンドのマシン上でもこのシステムを使えるように、モノクロディスプレイだけに限定して研究を進めている。

① 基本構造

構造レベルでは、PREP エディタは、Intermedia 、Neptune 、NoteCards などの文献に報告されている多くのハイパーメディア・システムと基本的特徴を共有している。このシステムは、アイデアにほぼ相当するチャンクを定義する。チャンクはテキスト、表、ツリー図、または、任意のイメージを含むことができる。このシステムは、著者が議論を構築するときに便利に使える外部表現として、幾つかのチャンク・タイプ（合成表、合成ツリー図）を特に標的としている。このワーク・スペースには利用者が初期にアイデアや議論を構成するときに可視的な理解を助ける作図ツールも含まれている。このシステムはチャンク間のリンクも定義するので、概念のネットワークを構築することができる。

チャンクは、協同作業者間で共有されるデータベースに格納される。しかし、アイデアのネットワークへ単に共有アクセスできるだけでは、協同作業には十分でない。協同作業者のスクラッチ・スペース全体を通読することは、通常の場合に、机の上に散らばっている本、ファイル・フォルダ、および、論文を厳密に調べることにほぼ相当する。相互理解を促進するために、PREP エディタはワーク・スペースの各部分についてコミュニケートするための規則を提供する。特に、このシステムでは、著者が「草稿」を定義できるようになっている。草稿というのは、著者が他のパートナーにアクセスさせることを目的として作成したワーク・スペース内の一領域のことで、所々が埋ったチャンクのグリッドで構成されている。グリッドの各カラムは、ワーク・スペースの様々な内容を格納するために使われるのが普通である。カラムは互いに（または、メインカラムに）関連付けることができる。例えば、一つのカラムには論文の内容を、もう一つのカラムにはその内容の構成するための計画を格納する。コメンテータは、カラムを読むときに、独自のコメントをカラムに付け加えることができる。慣れないうちは、PREP エディタの草稿は伝統的な直線的なテキストに基づく草稿とは大いに違って見えるかも知れないが、作業を進めるうちに、だんだん伝統的な草稿に似てくるのが普通である。これは、内容が、通常、計画やコメントよりも重要なものとして扱われるためである。

② インタフェース

草稿の可視表現については、現行のシステムでは注解の多い古いタイプの学術テキストに似たスタイルのコメントをサポートしているが、「ダイナミック注解」とでも呼ぶべき経路を追求中である。これは、フォント・サイズや空間関係を使ってシステム内のチャンク間のリンク状態を示すことができるものである。本システムでは、コメントを可視的に整列すると協力者は「一目で」コメントを見分けられることを発見したが、フレキシブルなシステムにおいては、一般的に、任意の形状と、動的

に選択されたアイテム間の複雑なリンク状況を扱うことができる制約ベースのレイアウト・アルゴリズムが必要である。

③ バージョン化

協同執筆関係で最も共通なイベントの一つは、まず著者が草稿を他のメンバに提出し、次にこれを審査し、最初の著者が新しい資料を組み込む「編集→審査→組込み」のサイクルである。幾つかのシステムは、「変更バー」または他のヒストリ・メカニズムをサポートして、テキストが変更されたポイントを指示することによって、このプロセスを支援している。PREP エディタはさらにこれを推し進め、草稿とは別個のバージョンとして既存テキストへの訂正を可能としている。さらに、固有の計画スペースがあるので、訂正の背後の論拠もコミュニケートできる。このバージョン化による訂正機能を利用すると、著作の専門家たちは、既存資料を失う心配なく、草稿のスペースを操作することができる。

④ PREP エディタと既存のハイパーテキスト・システムの関係

PREP エディタは、一般的な意味で良好なハイパーテキスト・システムを提供するものではない。その代わりに、PREP エディタは、やや新しい可視的な構造を介して、リンクされたチャンクをサポートしている。これらの構造は、他のハイパーテキスト・システムの上に構築することが可能である。

8.6.3 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。

8.6.4 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8.6-2に示す。

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

表8.6-2　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

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。

8.6.5 DistEdit

原文タイトル： DistEdit: A Distributed Toolkit for Supporting Multiple Group Editors

翻訳タイトル： DistEdit：マルチプル・グループ・エディタを支援する分散型ツールキット

著　者： Michael J. Knister
Atul Prakash
(Software Systems Research Laboratory, Department of EECS, University of Michigan, MI, USA)

出　典： CSCW '90 Proceedings, October 1990, pp.343-355

本プロジェクトは、分散型環境での協同作業をサポートするアプリケーション構築のためのツールキットの提供を目的としている。本稿では、そのようなツールキットとして、分散型環境でのインタラクティブなグループ・エディタ構築に使える DistEdit について紹介されている。ここでは、まず本ツールキットの要件が述べられ、つぎに関連する他の研究との比較が行われ、さらに、試験システムのオペレーション、ツールキットの設計、このツールキットを使ったグループ編集に適合させるためのエディタの修正について述べられている。また、ツールキットの利用で得られた結果を提示し、設計の決定に関与したトレードオフについて述べられている。

(1) DistEdit ツールキットの要件

DistEdit ツールキットの要件は次のような非常に単純なものである。

① マルチユーザ協同作業

DistEdit ツールキットを使って構築したエディタは、ユーザが物理的に離れたまま協同してテキスト・ファイルの編集を行えるものでなければならない。

② リーズナブルな性能

DistEdit 内で使用する通信プロトコルは、グループ編集を不便に感じないくらい、少ない遅延で全てのユーザが一貫してファイルを見ることができなければならない。

③ マルチプルな既存エディタとの互換性

一つのグループ・セッションで異なるエディタを利用することができなければならない。誰でも自分の好きなエディタがあるので、グループ・セッションに参加する時、それ以外のエディタの利用を強制されないことが望ましい。

④ 耐故障性

グループ・セッションは、マシンのクラッシュや人の参加、退席にかかわらず、スムーズに継続さ

れなければならない。

⑤ 通信プロトコルの隠蔽

エディタを DistEdit を使ったグループ編集に適合させる際、通信プロトコルのような分散型システムに関する知識が必要であってはならない。

(2) 関連する研究との比較

グループ編集のために特別に設計されたエディタとしては、GROVEとShrEdit がある。DistEdit はこれらのシステムとは異なり、エディタではなく、シングル・ユーザ・エディタをグループ編集タスクに適合させるために使うツールキットである。DistEdit ツールキットを使えば、一つのグループ・セッションで、さまざまなハードウェア上のさまざまなエディタを利用することが可能である。例えば、あるグループ・セッションで、一人が Emacs、もう一人が vi（どちらも端末上で稼働）、さらにもう一人が xedit（X が稼働するワークステーションが必要）を使うことも、それぞれのエディタのコードをほんの少し修正するだけで可能となる。

さまざまなエディタをフロントエンドとして DistEdit を使うので、Lantz が述べている構成可能ユーザ・インタフェースに似ている面もある。Lantz は、ユーザ・インタフェースを、単一のバックエンドとインタフェースする個別の交換可能なモジュールにすることで、カスタマイズ可能なユーザ・インタフェースを作ることを提案している。同様に、DistEdit ではさまざまなエディタが交換可能なフロントエンドとして作動するが、Lantz の述べたモジュラー・フロントエンドがかなり小型で開発が簡単なものを意図していたのと対照的に、DistEdit はサポートするフロントエンドより実際には大幅に小さくなる。

DistEdit ツールキットの設計によって、非常に高度な耐故障性が提供される。グループ・エディタを設計する際、単純な方法として多くのエディタや協同作業用ツールに利用されているものに、クライアントーサーバ・モデルがある。これは、エディタ・バッファの状態を保持する一つのサーバが存在し、すべての編集コマンドが共有サーバを通るようにすることで、ユーザのビューの相互一貫性を確保している。この設計は単純だが、サーバのクラッシュに弱い。さらに、一人の人間がファイルを編集している場合でさえ、更新にはサーバを通らなければならないため、編集が遅くなる。これに対して、DistEdit は各ユーザのエディタ・バッファ状態のコピーを保持する。

グループ用ツールキットに対するもう一つのアプローチとして、LIZA を挙げることができる。LIZA は、グループウェアを迅速にプロトタイプ化するための高度なツールの集合体で、メッセージ送信やムード表示、協同編集、スライド展示、グループのモニタ等のサポートを行う。このようなフレームワークと基本的なツールセットを持った LIZA は、主に協同作業の新しいアイデアを迅速かつ容易に試験することを目的としている。DistEdit は、これよりもかなりレベルの低い、アプリケーション

が特定されるツールキットであり、ユーザが既に慣れている状況でのグループ編集の可能性を深く探求するために設計されている。

密接に関連した協同作業システムとして、より非同期つまり非リアルタイムなインタラクションをサポートするものがある。この例としては、CES や Quilt 等のエディタが挙げられる。これらは、ユーザが同じ文書に対して作業できるが、通常は異なる時期に異なるセクションで行うものであるため、システムのインタラクションの持続期間は非常に長く、何日もかかることさえある。このようなシステムでは、耐故障性や更新のリアルタイムな伝播に関する問題の多くは重要ではない。DistEdit ツールキットはより密接に連結されたリアルタイムのインタラクションを提供することに焦点を当てている。

(3) エディタ例：DistEdit ベースの MicroEmacs

ここでは、DistEdit を使えるように修正した MicroEmacs 試験システムのオペレーションを、ユーザの立場から検討する。GNU Emacs のオペレーションも、編集オペレーションが強力になることを除いて、同様である。一般に、グループ編集セッションの各参加者は、DistEdit ツールキットを使うために修正が行われているのであれば異なるエディタを使うことができる。

MicroEmacs エディタのオペレーションは、DistEdit システムを使わない通常の利用とほとんど変らない。ユーザはプログラム名を指定してエディタを起動してから、編集するファイル名を指定する。エディタは最初のユーザのために動くので、すべては通常と同じで、状態表示行にこのユーザが「マスタ」であることを示すインジケータが出る点だけが異なる。

別のユーザが同じファイル名でプログラムを動かした場合には、エディタ・バッファにロードされたテキストは最初のユーザのものとなり、最初のユーザがすべての変更を行うと共に終了する。最初のユーザが「マスタ」で、すべての編集機能を持っており、二番目のユーザは編集機能のない「オブザーバ」である。オブザーバである二番目のユーザは、マスタの行うすべての変更とカーソルの動きを見るが、オブザーバのカーソルはここでマスタのカーソルに「ロックステップ（連動）」されている。

オブザーバは、テキストを変更するようないかなるオペレーションも行えず、変更を行おうとしてもそのようなオペレーションは無効である。しかし、ファイルをセーブしたりウィンドウの大きさを変えたり、テキストを変更しない編集コマンドを発することはできる。

オブザーバが自分のカーソルを独自に動かしたいと思ったら、自分のカーソルをマスタのカーソルの従属から解放する「ロックスイッチ」コマンドを呼出すことができる。この「フリーカーソル」モードでは、オブザーバは自分のカーソルをどこへでも好きなところに動かすことができるが、やはり変更を行うことはできず、マスタの行った変更は全て同時に更新される。ロックスイッチとフリーカーソルは、オブザーバが自由にオン／オフ切換えができる。

常に一人のユーザがマスタで、他の人はオブザーバである。マスタは、「コントロールスイッチ」コマンドを呼出していつでも制御を放棄することができる。全てのエディタの状態表示行が更新されて、マスタがいないことが示される。マスタがいなくなると、どのオブザーバもコントロールスイッチ・コマンドを使って制御を得てマスタになることができるが、コントロールスイッチを発した最初のユーザがマスタになる。そうすると状態表示行が更新されて、マスタ／オブザーバ状態が表示され、新しいマスタが上記のオペレーションを行うことになる。マスタがいない時は、誰も変更を行うことができない。制御は、電話のような外部ラインやオンラインのトーク・セッションでの話し合いで譲渡する。

一つの編集セッションにユーザは何人でも参加することができる。ユーザは、他ユーザに影響を与えずに、いつでもセッションに入ったり抜けたりすることができる。一人のユーザのマシンが故障しても、そのユーザがセッションから抜けるだけである。マスタがセッションを抜けたり故障に遭遇したりすると、他ユーザはマスタなしという状態を見て制御を得られることになる。編集中のテキストは、全てのユーザがセッションを抜けて誰もローカルにセーブしておかなかった場合にのみ失われる。

一つのセッションのユーザは全員同じテキスト・バッファを持つ。見え方は、ウィンドウのサイズによってユーザごとにやや異なる。ロックステップの場合、オブザーバのカーソルはウィンドウ内を動いて常にテキストの同じ論理的位置にあるが（同じ行および桁）、実際のスクリーン上の位置は異なることがある。

キーポイントは、サポートされているエディタすべてで、通常の編集コマンドが普通通りに動くことである。ユーザ側から見た変化は、ロックステップとコントロールスイッチ・コマンド、マスタ／オブザーバとロックステップ／フリーカーソル状態インジケータだけである。

(4) DistEdit ツールキットの設計

DistEdit システムは、多種のビジュアル・エディタに容易に適用できるモジュラ・ツールキットとして設計されている。適用の要件としては、エディタがテキスト・バッファを直接修正する少数のサブルーチンを持っていることのみである。これを持っていないと、エディタには広範囲な修正が必要となる。エディタは、テキスト・バッファから読み取るルーチンについては修正する必要はない。

① DistEdit プリミティブ

DistEdit プリミティブとは、エディタのコードから呼出すことのできるオペレーションである。プリミティブは、エディタからの一つのファンクション・コールで開始されるが、実行場所に着く前に、何層かのコードやネットワークを進むことができる。この意味で、DistEdit プリミティブは、多くの遠隔実行を生じさせることになる点を除けば、リモートプロシージャコールのようなものである。

DistEdit のプリミティブには、(1) 更新プリミティブ、(2) カーソル位置通知プリミティブ、(3) 制御

プリミティブの三種類がある。DistEdit には、すべてのテキスト更新オペレーションを実行する更新プリミティブが三つ（ストリング挿入、ストリング削除、ストリング置換）と、カーソル位置の変更を同報通信するカーソル位置通知プリミティブが一つ、状態と制御情報の譲渡のための制御プリミティブが二つ（制御交代、ロックステップ切替え）ある。

更新プリミティブとカーソル位置通知プリミティブを発することができるのは、マスタ・エディタのみである。制御プリミティブは、オブザーバでもマスタでも使うことができる。

② 基礎となる通信ソフトウェア

同報通信機能と誤り制御が高品質であり、プロセス・システムが軽いという点から、通信パッケージとして分散型アプリケーションをプログラムするためのツールキットである ISIS が選ばれている。

ISIS の同報通信機能によって、DistEdit が低レベルの通信を取り扱う必要がなくなった。メッセージは失われず、すべての同報通信がグループの全参加者に同じ順序で到着することが保証される。

ユーザは、いつでもグループ・セッションに入ることができる。ユーザがセッションに入ると、エディタの状態が現在のユーザの一人から、新しいユーザのエディタに転送される。ISIS には、新しいユーザの通知メカニズムと状態転送のための耐故障性のある通信プロトコルがある。

(5) エディタに必要な修正

DistEdit ベースのグループ編集セッションで通常のエディタを使うには次のコード修正が必要となる。

① エディタ・バッファの全てのテキスト更新オペレーションを、三つの DistEdit 更新プリミティブへの呼出しにマップしなければならない。

② 三つの DistEdit 更新プリミティブを、エディタのテキスト更新ルーチンにマップし直さなければならない。これらのマッピングでは、エディタの全てのテキスト更新オペレーション（テキスト・バッファを直接更新する全てのルーチン）が機能が 5 個から 15 個の小さなルーチン・セットに入っていることが理想的である。

③ DistEdit 制御プリミティブへの呼出しは、エディタのユーザ・インタフェース部分に挿入して、ユーザがロックステップ・コマンドとコントロールスイッチ・コマンドを実行できて、ユーザにマスタ／オブザーバとロックステップ／フリーカーソル状態を示せるようにしなければならない。

④ DistEdit がカーソルの移動や照会を行って、テキスト・バッファの行を読み取れるよう、アクセス・ルーチンを備えなければならない。DistEdit がテキスト・バッファの状態を参加ユーザに転送するためにテキスト・バッファの行を読むルーチンが必要になる。

⑤ テキスト・バッファの変更をスクリーンに表示するため、DistEdit 内から呼出せるルーチンを備えなければならない。

システムの通信プロトコルやマルチプル・プロセス、耐故障性機能を扱う部分はすべてDistEditツールキット内に隠されている。上記の変更はどれも分散型プログラミングについて何も知らなくても実行できるようになっている。

(6) 結果

MicroEmacsとGNU Emacsのエディタを、DistEditツールキットを使った分散型グループ・エディタに変換し、両方のエディタを同じグループ・セッションで使った。

オリジナルのエディタ・コードの変更はほとんど必要なく、性能は十分に受入れ可能なものであった。

① MicroEmacsとGNU Emacsに必要な変更

(a) 変更量

DistEditを使うためのMicroEmacsの修正では、オリジナル・プログラム本体内で、9行のCのコードの追加と6行の変更が必要であった。マッピングと特にMicroEmacsのためのサポート機能を入れた追加ファイルには、約60行のコードが必要であった。

DistEditを使うためのGNU Emacsの修正では、オリジナル・プログラムの本体内で10行の追加と12行の変更が必要である。GNU Emacsのための追加サポートファイルには、約300行のコードが必要であった。

(b) 変更の困難さ

エディタの修正では、コード内の次の三つの領域の位置決定と修正を行った。(i) バッファ修正ルーチン、(ii) トップレベルのユーザ・インタフェース、(iii) 入力ブロックポイント。

このうち入力ブロックポイントの位置決めが最も難しい作業であった。これは、ユーザからの入力待ちをエディタがブロックするポイントである。入力ルーチンは、DistEditプリミティブがISISを通じて同報通信しながらキーボードかマウスの入力を待つようにするため、修正しなければならない。コードのこの部分の位置決めと修正は、入力抽象とマシン従属性が多レベルにわたるため困難であった。

② 性能

本試験では、それぞれ8 MBのメイン・メモリを持ち、Ethernetを介して接続されたDec 3100ワークステーションを使ってエディタを動作させた。ユーザが数人いても性能は妥当なものであった。マスタ・エディタは、全体として大きくスローダウンすることはなかった。オブザーバのエディタでは、おそらく同報通信にかかる時間が常に同じではないためか、カーソルがジャンプすることがあった。

(7) 考察

① マルチプル・エディタ・インタラクション

上の二つのエディタは一つのグループ・セッションでうまく動いたが、オペレーションやテキスト・バッファの見え方はエディタによって異なる。ウィンドウ表示とテキスト・マークがその例で、エディタがそれぞれ互いの機能を補完することもできる。

DistEditでは、全ユーザが同じテキストを見るのではなく、テキストが常に一貫していることが保証されるだけである。表示のされ方はエディタによって決り、長い行の処理などでは見え方が大きく異なる。（規定値の）GNU Emacsでは長い行はスクリーンの次の行に表示するが、MicroEmacsは水平スクロールを使っている。ユーザの選ぶウィンドウ・サイズも大きく異なることがあり、同じテキストでもユーザによって見え方が変ることになる。

異なるエディタを使う大きな利点として、互いに補完しあうよう複数のエディタの機能を散在できる点がある。例えば、MicroEmacs（「取消」機能がない）ユーザの実行したオペレーションを、制御を切換えた後、GNU Emacs（「取消」機能がある）のユーザが取消すことができる。このように、異なるエディタと異なるユーザを組合せて、一つのエディタよりも強力なパワーを生むことができる。

② 通信層構成

セッション内のエディタ間で通信を行うにはいくつかのレベルがある。本システムでは、通信用に基本的な更新と制御プリミティブを持たせることにした。各エディタの低レベルのオペレーションが、これらプリミティブへの呼出しに翻訳される。

代替案としては、ユーザのキー入力に直接割込んでこれを全ユーザに同報通信する方法もあったが、これには二つの問題がある。一つは、エディタが同じグループ・セッション内で異なっていると、他エディタからのキー入力を理解するよう全エディタを修正しなければならないため、実行が難しいことである。二つ目は、セッション内のエディタを同一にして単純化しても、同時更新を可能にするための拡張が簡単にはできないことである。異なるユーザからのキー入力は、予期せぬ行動につながる可能性があるため、一つのストリームに単純に混合してしまうことはできない。

③ プリミティブの選択

通信プリミティブを選ぶ際、いくつかの選択肢があったが、エディタから独立した非常に基本的なプリミティブとした。代りに複雑で強力なプリミティブ・セットを選ぶこともできた。特定のエディタ、例えばGNU Emacsで利用できる機能として通信プリミティブを選択することだが、このような強力なプリミティブには、ストリングの挿入、削除、置換ではなく、全体的な置換やブロック・インデンテーションのようなオペレーションを含むことになる。

基本的なプリミティブを採用した要因は次の三つである。一つ目は、強力な基本プリミティブ・セットの場合、システムに新しいエディタが加わるごとにプリミティブ・セットを更新ルーチンの呼出

しに翻訳するためのコードを追加しなければならない点である。第二に、非常に強力な基本プリミティブ・セットの大半をあまり頻繁に使わない場合の性能低下の可能性である。長いメッセージ・フォーマットが必要になるため、システムの帯域幅ニーズが増すことになる。エディタ・オペレーションの大半は、シンプルな挿入や削除であろうと予測して、これらをプリミティブとした。第三は、エディタ・プリミティブの数が多くて複雑な場合、同時更新の取扱いが複雑になる可能性である。この場合には、エディタの提供する各オペーレション間の矛盾を解決するアルゴリズムが必要となる。単純なエディタ・プリミティブを使うことで、同時更新の解決タスクは容易になる可能性が大きい。

8.7 設計議論システム

8.7.1 gIBIS

原文タイトル： gIBIS: A Hypertext Tool For Exploratory Policy Discussion

翻訳タイトル： gIBIS：調査方針の議論のためのハイパーテキスト

著 者： Jeff Conklin
Michael L. Begeman
(MCC, Software Technology Program, Austin, Texas, USA)

出 典： CSCW '88 Proceedings, September 1988, pp.140-152

本稿では、初期の設計議論状況の獲得を支援することを目的として設計された用途特定型のハイパーテキスト・システムについて紹介されている。このシステムは、MCC Software Technology Programの中のDesign Journalと呼ばれるハイパーテキスト・プロジェクトの一環として開発されたプロトタイプ・ツールである。本稿ではまず、このシステムのベースとなるIssue Based Information Systems (IBIS)（問題点に基づく情報システム）と呼ばれる特殊な方式について説明され、つぎにgIBIS (graphical IBIS)と呼ばれるこのシステムの機能について述べられ、最後にこのツールに関する予備的な観察について説明されている。

(1) IBIS方式

Horst Rittelによって開発されたIBIS方式は、複雑な問題の設計プロセスが、基本的には関係者（設計者、顧客、開発者など）が設計上の諸問題を解決するために各自の専門知識や観点を伝え合う関係者間の会話そのものである、という原則に基づいている。問題、関心、疑問などはいずれも設計上の「問題点(Issue)」と成り得るし、設計を進めるためには（合意が得られていなければ）これらの問題について議論する必要がある。実際、IBISモデルにおいては、設計プロセスを構成しているのはこの「議論」に他ならない（自分自身との「議論」を排除するということではない。gIBISは、対話的にも独白的にも、どちらにも使用できる）。

IBISモデルは、設計上の問題の主要なIssueを互いに結びつけることに焦点を当てている。一つ一つのIssueが多くの「解決案(Position)」を持ち得る。Positionというのは、Issueを解決するステートメントつまり主張のことである。Positionは互いに排他的であることが多いが、IBIS方式はPositionが排他的であることは要求していない。つぎに、Issueの各Positionごとに、そのPositionを支持する「議論(Argument)」または、そのPositionに反対するArgumentが、一つまたは二つ以上存在し得る。したがって、独立した一つ一つのIssueがツリーのルートとなり（空のツリーの場合もある）、Issueの子供

としての Position と、Position の子供としての Argument を持つことになる。

IBIS には、9種類のリンクがある。例えば、Position は、Issue との間に、「応答 (Responds-to)」リンクをもつ。この Responds-to リンクは、Position と Issue の間にしか使えない。Argument は、その親 Position との間に、「賛成 (Supports)」リンクまたは「反対 (Objects-to)」リンクをもたなければならない。Issue は、他の Issue との間に、「一般化 (Generalize)」リンクまたは「特殊化 (Specialize)」リンクをもつことができる。また、Issue は、他の Issue、Position、および、Argument との間に、「質問 (Question)」リンクまたは「ヒント (Be-suggested-by)」リンクをもつことができる。

代表的な IBIS の議論は、誰かが「X をどのように処理すべきか？」といった疑問を含む Issue ノードを提示することで始まる。その人物は、同時に、X の処理方法を提案する Position ノードを提示することもできるし、その Position を支持する Argument ノードを提示することもできる。これに応答して、あるユーザは別の Position を提示し、これを支持する独自の Argument を提示するかもしれない。その他のユーザもまた、それぞれの Position を提示したり、いずれかの Position に対する支持または反対の Argumentを提示したりすることができる。これに加えて、Argument によって発生した新しい Issue が提示され、直接この Issue の元となったノードにリンクされたりする。図8.7-1 に、IBIS 方式の中で有効とみなされる動作を規定した状態遷移図を示す。

IBIS 方式には、停止のための規則は存在しないし、幾つかの Position に合意に達して Issue が解決されたことを記録する特定の方法も存在しない。むしろ、議論の目標は、関係者のそれぞれが、互いの提案の特定の要素を理解すべく努めること、および、個人の観点を他の関係者に説きつけるべく努め

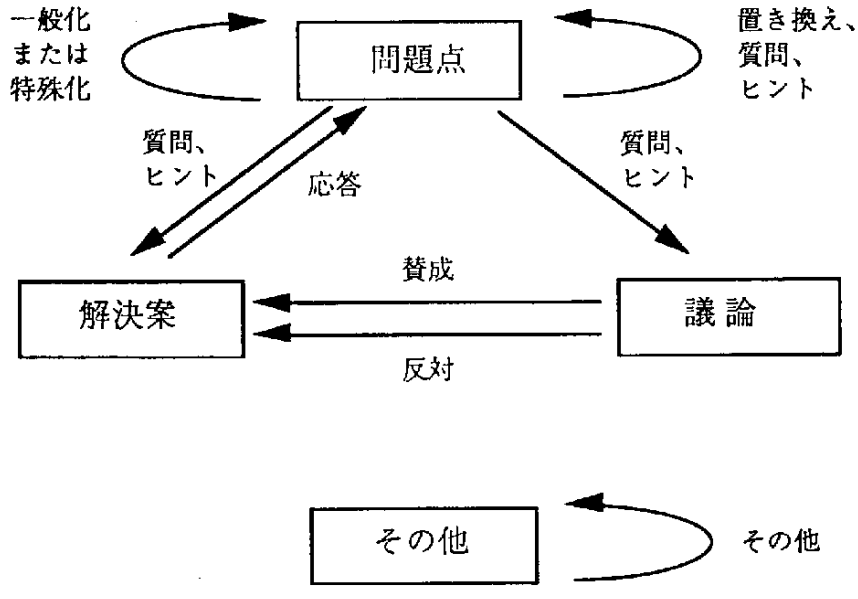


図8.7-1　IBISモデルの要素とリンク

ることにある。さらに、この方式は、中心的なIssueの探索、量的に片寄りのない質疑応答、一つの観点を証拠を挙げて支持するなどの、より建設的な動作を支援する。

gIBISをインプリメントするに当たって、IBIS方式に次のような変更と拡張を加えて、必要なフレキシビリティを確立した。(i) IBISのフレームワークの中で自身の思考を表現する方法を発見できなかったユーザのための「エスケープ」メカニズムとして、ノードとリンクに「その他(Other)」というタイプを追加した。(ii) 要求文書、設計スケッチ、コードなどの非IBISマテリアルを含むノードのために、「外部(External)」というタイプを追加した。(iii) 他のPositionを特殊化または一般化する機能をPositionに与え、同様に、Argumentにもこれらの機能を与えた。

(2) gIBISツール

基本的なgIBISインタフェースは四つの独立したウィンドウ、すなわち(i)画面左側のグラフィカル・ブラウザ、(ii)右側上のノードへの構造化された索引ウィンドウ、(iii)索引ウィンドウのすぐ下のコントロール・パネル、(iv)ノードの属性と内容およびリンクを表示する検査ウィンドウから構成されている。ユーザがノードまたはリンクを選択すると、その内容が即座に検査ウィンドウに表示される。

① ブラウザ

ブラウザは、IBISのネットワーク構造（すなわち、Issue、Position、Argumentの各ノードとその間のリンク状態）を目に見える形で表示するためのウィンドウである。ノードとノード間の相互接続リンクは、実質的にサイズが無制限のキャンバスに表示される。ブラウザは、ローカル・ビューとグローバル・ビューからなり、ローカル・ビューがウインドウのほとんどの部分を占める。この部分には、現在関心をもっているエリアがズームインされたビューが表示され、ノードとリンクの完全な詳細を見ることができる。これに対してグローバル・ビューの部分には、ノードのラベルやリンクタイプのアイコン、ネットワークの細かい構造を作り上げている二次リンクなどを一切排除して、ネットワーク全体の概観が表示される。また、グローバル・ビューは、長方形の囲みによって現在のローカル・ビューの範囲と位置を示している。ブラウザのウィンドウ領域内で、キャンバスはスクロールすることができる。

ブラウザは、ノードやリンク（すなわち、表示したいオブジェクト）への直接操作式のインタフェースをサポートしている。マウスクリックで表示したいオブジェクトを選択すると、ブラウザの中でそのオブジェクトがハイライト表示されてボックスで囲まれ、その内容が検査ウィンドウに表示され、その索引ラインが索引ウィンドウの一番上にスクロールされる。

また、マウスクリックにより表示されるメニューから、オブジェクトの作成、編集、削除、移動などの操作を指示することができる。例えば、タイプがIssueのノードを選択すると、Issueに対して実行できる有効操作を反映したメニューが表示される。このメニューには、「従属ノードの生成」、

「ノードの移動」、「ノードの編集」、「他のノードへのリンク付け」、「ノードの削除」などの項目がある。ここでユーザがタイプがPositionである従属ノードの生成を選択すると、この従属ノードは選択されているIssueの右側に配置され、タイプResponds-toのリンクによって自動的にIssueとリンクされることになる。このメニュー選択を行うと、検査ウィンドウ（右下のウィンドウ）が半分に分かれ、下部にノード作成ウィンドウが表示される。このノード作成ウィンドウは、構造化されたテンプレートをウィンドウ領域一杯に表示するように予め設定されている。そこでユーザは、テンプレートの構造化されたフィールド（Subject、Keywordsなど）にデータを入力し、つぎにノードのトピックに関する任意の記述を入力する。ユーザがノードに関するデータを入力し終えてコントロール・パネルのSUBMITボタンを押すと、ノードが解析されて格納され、これに伴ってブラウザ・ウィンドウと索引ウィンドウが更新される。

また、ユーザが「他のノードへのリンク付け」というメニュー項目を選択すると、現在選択されているノードに有効な1セットの出リンクタイプのメニューが表示される。このメニューには、「質問」、「一般化」、「特殊化」などの項目がある。ユーザは、これらのリンクタイプの中から、いずれかを選択した後、ブラウザ・ウインドウ上でリンクすべきノードをマウスクリックで指示する。

gIBISは、カラー・モニタ付きのSUNワークステーションで使用するように設計されており、ノードやリンクタイプの情報、さらに「currently selected（現在選択中）」や「matches the current query（現在の照会と一致）」などの特殊なノードの状態が、色分けされて示される。白黒モニタを使用しているユーザのためには、色分けされた情報をアイコンで表示する機能を設けている。gIBISツールは、デフォルト設定のままでは、ユーザにカラー情報とアイコン化情報を両方とも提示するが、どちらの情報もユーザの手で抑止できる。

② ノード索引ウィンドウ

ノード索引ウィンドウには、現在のIBISネットワーク内のノードを、その順序に従って階層構造に配置したビューが表示される。Issue、Position、およびArgumentには通し番号が振られる。Issueは単純に作成順に並べられる。ユーザは、ビュー構成パネルを利用して、Subjectだけでなく、Author、Keyword、または、ノードのLabelに基づく索引情報を、自由に構成することができる。

ノードは、ブラウザからだけでなく、索引からも選択できる。ノードの索引ラインの上でクリックすると、そのノードが現在のノードになる。これに伴い、そのノードのアイコンがブラウザ・ウィンドウの中でハイライト表示され、そのノードの内容が検査ウィンドウに表示される。

③ コントロール・パネル

コントロール・パネルは1セットのボタンで構成され、これらのボタンを使用することによってgIBISツールの機能性を単なるノードとリンクの作成機能を越えたものに拡張することができる。各ボタンをマウスクリックするとメニューが現われ、このメニューを使って基本機能を拡張したり、基本

機能をニーズに合わせて変更したりすることができる。例えば、NEXTボタンを選択すると、通常、gIBISはユーザが読み終った現在のノードを記録してから次のネットワーク・ノードを表示する。しかし、NEXTボタンの上でマウスの右ボタンを押すと、mark current node as unread（現在のノードに未読マークを付けよ）という項目を含むメニューが表示される。

また、MISCボタンを選択すると、さらにユーザのニーズに合わせてインタフェースの特定の状況を作り変える機能や、ユーザから開発者へツール・グリップ／提案を送る機能、すべてのノードに読了／未読マークを付ける機能、IBISネットを直線化およびプリントする機能、Issueグループに名前を付ける／消す機能などを選択することができる。

QUERYボタンを選択すると、小さな照会・構成ウィンドウが表示され、その中の項目（Type、Date、Author、Subject、Keywordsなど）から照会したいものを選択してタイプインする。照会の結果は、ブラウザ・ウィンドウ（ローカル・ビューとグローバル・ビューの両方の中で、選択したノードが明るい黄色に変わる）と索引ウィンドウ（照会を満足するノードの索引ノードだけが表示される）の両方に表示される。

④　リレーショナルDBMSの利用

gIBISをインプリメントするに当たっては、工数が非常に限られていたため、既存のリレーショナルDBMSであるUNIFYの上にgIBISを構築することにした。データ蓄積マネジャとしてリレーショナルDBMSを選択したことで、同時制御、レコード・レベルのロッキング、信頼性の高いデータ蓄積、高速アクセス方式、および、妥当な検索エンジンを手に入れることができた。さらに、このDBMS(UNIFY)は無解釈データタイプ（基本的には、任意の長さのテキスト、デジタル化した音声、グラフィック・ビットマップ等々を書き入れることができるフィールド）を提供しており、データベースの記録に不可欠な一部としてノードのボディを格納することができた。

しかし、DBMSをデータ蓄積マネジャとして利用したために、閉鎖的なシステムになってしまうという大きな不利益も生じた。本質的に、Issueネットワークが参照するすべてのオブジェクトは、DBMSの中に置かれていなければならないのである。残念ながら、Issueベースの議論に出てくる多くのオブジェクト（要求書、構造説明書など）やネットワークから得られる結果（コードやドキュメントなど設計の成果物）は、このデータベースの範囲外にあるので手が付けられない。このため、特殊な代理タイプのノードを作成して、「盲信的な」方法でgIBISが外部オブジェクトを参照できるようなメカニズムを作っている。この代理ノードは、外部オブジェクトを指示するポインタ（通常は、ファイルへの完全パス名）と、gIBISがオブジェクトを表示するときに呼び出す任意のディスプレイ・プログラムの二つのパートをもつ。デフォルトのディスプレイ・プログラムを呼び出すと、外部オブジェクトはテキスト・ファイルと仮定され、gIBISの標準検査ウィンドウにロードされる。一方、ユーザがディスプレイ・プログラムを指定すると、このプログラムが呼び出され、引数として外部パス名を

パスする。この機能を利用すれば、外部データおよびプログラムをgIBISにスムースに統合することができる。

(3) 観察所見

ここでは、IBIS方式とgIBISツールの両方について、その長所・短所と研究上の問題点を述べる。

① ネットワークの構造

1987年の半ばから1988年の半ばにかけての一年間のgIBISツールの利用状況を、ネットワークに関して統計を取った。32人の人々が、全部で33のIssueの作成に参加した。1988年2月現在で、作成されたノードの数は2,091個、すべてのIssueグループがほぼ同数のIssue、Position、および、Argumentを持っていた。

Issueノードの31%はPositionを持っていないが、残りのIssueノードは平均1.9個のPositionを持っていた。一方、Positionノードの59%はArgumentノードを持っていないが、残りのPositionは平均1.7個のArgumentを持っていた。

また、これらのノードを接続しているリンクの数は2,214個であった。リンクのタイプによる分類結果から、ユーザが反論Argumentよりも支持Argumentをより多く提示する傾向を持っていること、および、ノードを一般化するよりは特殊化する方が自然であることが示された。

さらに、この試験期間中の大きなIssueグループ15個における個人参加のレベルについての調査から、多くのIssueグループが主として一人の参加者によって構成され、参加者のバランスが取れているIssueグループはごくわずかであることが判明した。

② 明示修辞構造の有効性

幾つかのIssueグループに協力して参加していた何人かのユーザは、gIBISツールが議論に強制的に課す構造が非常に有効であり、「胸に秘めた一物や、密かな合図や、巧妙な修辞法」を白日の元に晒す役割を果すと報告してきた。彼らは、また、仮定や定義が明らかになる傾向も評価している。

これらの利点の幾つかは、IBISネットワークの半構造化された性質に、その源をたどることができる。この性質のお陰で、書き手は表現に何の制約も受けずに完全なメッセージを構造化することができたし、読み手は調査と総合的理解の両方を支援するテキストの反復構造を読み取ることができたのである。読み手と書き手の両方が、IBISの明示修辞構造の恩恵を受けており、この構造は、少なくとも、言葉による議論の一般的な構造を明らかにしている。

③ ツールと方式の相互促進作用

コンピュータ化されていないIBIS方式は扱いが厄介なので、高速gIBISツールによるサポートがなければ、現在得ているような人気を博することはなかったと思われる。また、gIBISは本研究所の環境で使用できる唯一のハイパーテキスト・システムという訳でもないのに、短時間のうちに、より広く

長時間使用されるようになった。これは、IBIS 方式に対する要求とgIBIS ツールのハイパーテキスト機能の間に、特に良好な整合性が存在したためであると思われる。

④ 未熟なセグメンテーションの危険

ハイパーテキスト・システムに一般的に見られる微妙な問題の一つは、個人の思考を独立した単位に分割するのがしばしば不自然なことである。特に、問題がよく理解されていないと、その思考は不明瞭で、混乱した、移ろいがちなものとなる。IBIS方式はユーザにノードタイプとリンクタイプの厳格な選択を強制するので、gIBIS を使用するとこの傾向が顕著になる。特に、設計の会話は、「X を試してみよう —Y の利点がある」といった形のコミットメントを特徴とすることが多い。これは Position と、その支持Argument であるが、Position に Issue が結合されていない。ユーザの中には、Issue と Position を即座に見分けることができない、アイディアを構造化する前に単に記録しておく「プロトノード」が欲しい、というような苦情を寄せた人もいる。

⑤ Issue の解決の把握

審議プロセスには、Issue の解決をどのように表現し、表示するかという、やや補足的な問題が存在する。IBIS 方式は、「正しい回答」または少なくとも「我々が現在コミットしている Position」として Issue に応答する Position の一つを選択することによって（選択の方法は問題でない）、Issue が解決されるとしている。このためには、Position ノードに「SELECTED」マークを付けて表現すればよいし、ブラウザ・ウィンドウの中でその Position ノードをはっきり区別して表示すればよい（例えば、選択されていない Position とは異なるカラーで表示するなど）。

⑥ 非直線的文書の文脈の問題

gIBIS の実験の重要な要素の一つは、協同作業の媒体としてのハイパーメディアの利用性を調査することにある。数人のユーザが、Issue グループを共有して、協同で作業を行っている幾つかのケースで、予期せぬ問題が出現した。書き手が細心の注意を払って明瞭かつ完全に文を書かなかったために、読み手は、個々のノードを理解できたと感じたものの、数十ものノードにまたがって曲りくねっている書き手の思考の糸をたどることができなかったのである。つまり、ハイパーテキスト・ツールは、書き手のアイディアを、強制的に、細かく分割してそれぞれの粒子を独立して表現させるので、書き手の頭の中で展開しつつあるより大きなアイディアが曖昧になってしまう、ということである。

この問題の改善には、幾つかの技術が有効である。例えば、ノードを 1 セットにまとめる技術が考えられる。これは、IBIS サブツリーのすべてのノード（Issue、Position、Argument）の表示を単一のノードにまとめる新しいタイプの IPA ノードで、このノードに IPA 特定テキストを添付することもできる。この新しい構造は個々の Issue の議論を直線化するので、gIBIS ネットワークを読む時によく感じられるような分裂した感じが減少すると思われる。

⑦ 注釈的なまたは「上位（メタ）の」議論

例えば、「しかし、それはここでは問題でない」など、「上位（メタ）」、つまり議論の（内容ではなく）プロセスにコメントを付けることは、人間の会話によく見られることである。同様に、IBISの議論においても、Issueグループの参加者たちが、仲間の誰かがIBIS構造を十分に使いこなしていないとか、アイディアの出し方が不正確であるなどと感じたときには、メタの議論が必要となることがある。例えば、BがAのIssueノードの内容について、実は、これが二つのIssueと、一方のIssueに対するPositionであると感じたとすれば、Bは何等かの方法でこれを表現し、Issueグループの状況の中で、Aとこの「メタIssue」に関する議論を始める必要がある。

この問題を解決する方法は幾つかあるが、本研究では各ノードに独自の「メタ層」をもたせる方法を採用しており、ノードの内容がそのIBISタイプと一致しているかどうか、といった議論は、ノード内のこの特殊化された部分で追跡するようにし、リクエストに応じて可視表示できるようにしている。現行バージョンでは、どのユーザも任意のノードのボディ末尾に「メタ・ライン」を付加することができる。メタ・ラインを追加したら、コメントを入力し、サインすることによって、ノードのこの部分の中で、注釈的または手続き的議論を開始することができる。ノードの書き手は、ノードの末尾に応答を付加してもよいし、構造エラーを訂正するためにネットワークを修正してもよい。

⑧ ブラウザ・スペースのマクロ・レベル組織

ハイパーテキスト研究の重要なテーマの一つは、ネットワーク内のノード数が数十以上のケースで、グラフィカル・ブラウザを如何にして有効利用するかという問題である。現在の形のgIBISブラウザは、ノード・ビューイング・ウィンドウとコントロール・パネル・ウィンドウがスクリーンのスペースを共有しているので、スクリーンの半分以上を占めることはできない。この大きさのブラウザ・ウィンドウに一度に表示できるのは、せいぜい40個から50個のノードである。ブラウザには各ノードについてほんの一語か二語のごく短いラベルだけしか表示されないので、ノードに関する詳しい情報が欲しいときは、その上でマウスクリックして、ノード・ビューイング・ウィンドウの中でその内容を読まなければならない。

⑨ 展開を続けるネットワークの変化に対処することの重要性

すべてのデータベースは、そこに内包されるデータの変化を管理するためのメカニズムを持たなければならない。「Issue base」の本質は、古い版の方が正確で重要性も高いとか、いや、不正確でヒストリとしての関心しかあり得ないとか、あるいは、その中間であるとかいった、展開し続ける議論のためのツールとなることにあるので、gIBISのようなアプリケーションでは、変更が非常に大きな意味をもつ。

このため本システムでは、ネットワーク・マテリアルの改版と関連性をシステマティックに指示するメカニズムを必要としている。例えば、最近議論の対象となったノードや、更新されたノードを除

いて、古いノードは黄色で表示するとか、端部をギザギザにして表示するとかいった方法が考えられる。重要性、やま場、信頼性に関して示したメカニズムと同じように、改版と関連性もまた、部分的にしか自動化できない主観的な尺度であると言えよう。

変化を管理するもう一つのメカニズムは、言うまでもなく人間である。Issue ネットワークのサイズと重要性が増すにつれて、組織に Issue ベースの通用性と健全性を管理するスタッフがどうしても必要になってくる。

8.7.2 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。

8.7.3 SIBYL

原文タイトル： SIBYL: A Tool for Managing Group Decision Rationale

翻訳タイトル： SIBYL：グループ意思決定原理の管理ツール

著　者： Jintae Lee
(Center for Coordination Science, and Artificial Intelligence Laboratory, MIT, MA, USA)

出　典： CSCW '90 Proceedings, October 1990, pp.79-92

本稿では、グループの意思決定支援システム、SIBYL について紹介されている。まず、SIBYL の基本となる動機について述べられ、つぎにSIBYL が意思決定過程を表現するために使う言語 DRL (Decision Representation Language) について簡単に述べ、SIBYL の模擬セッションでの用法が示される。さらにSIBYL と同じような目的を持つ他のシステムとの比較が行われている。特に、設計原理を記録することを目標とするハイパーテキスト・システムである gIBIS と比較されている。

(1) 動機

SIBYL は、大きな二つの動機が基本となっている。知識の共用と、質的な側面からの意思決定支援である。

① 知識の共用

通常、意思決定を行うには、ある基準に沿って選択肢を評価する際に関係する知識の断片を集めたり関係づけたりということを行う。こうした知識は、いったん明白に表現されると、いくつかの方法によって、他の者と共用することができる。グループの意思決定では、特に表現された知識は、新たな知識を取り込み、それを支持する主張、拒否する主張、条件付与などによってその知識を膨らませることができる。

また、意思決定プロセスの一部として表現された知識は、同じような意思決定を行う他のグループにとっても、有効であることが多い。過去の意思決定は、必要とする実際の情報だけでなく、ある意思決定にいたるまでの方法、違った目標にも到達しうる方法、選択肢の評価の対象となるべき属性について理解する際の手助けとなる。さらに、過去の意思決定は、選択したアプローチが成功したか否かという知識も提供してくれる。知識を共用する方法としては、一つのグループの中でも時を隔てることで行われることがある。意思決定がどのようになされたかという記録をドキュメントとして提供すれば、これが転じて公式化と学習の基本となる。

② 質的な意思決定支援

意思決定過程の質的構造を表現する言語を持つことができれば、そのシステムは、意思決定を支援

する多くのサービスを提供することができる。例えば、対象間の依存性をシステムで管理することができる。追加される知識の効果や不確実性について伝達することができる。そして、現在の意思決定に有効な知識の断片を過去の意思決定から取り出すことができる。

(2) SIBYL

SIBYL は、次の三つの部分からなる。(i) DRL (Decision Representation Language)、(ii) DRL で表現されたものを利用して質的な意思決定支援を行うサービス群、(iii) DRL をより簡単に利用できるようにするユーザ・インタフェースである。ここでは、SIBYL の模擬セッションという条件のもとでこれらを図示することとし、すなわちユーザ側に立って見た SIBYL について説明する。

① DRL (Decision Representation Language)

DRL は、Goal、Alternative、Claim、Qestion、Procedureという５種類のノードとそれらの間の関係（14 種類）によるグラフで議論の過程を表現するものである。

Goal（目標）は、理想的なオプションが持つべき特性を表す。Alternative（選択肢）は、選ぶべきオプションを表わす。Decision Problem（意思決定課題）は、目標を最適に満足させる選択肢を選ぶ問題を表す。各Alternative は、Achieves（達成）という関係を通してGoal と関連づけられ、文法上では `Achieves(Alternative,Goal)` と表す。DRL でいう「関係」とは、Claim（主張）のサブクラスであり、特に、関係 `Achieves(A,G)` は、選択肢 A が目標 G を達成するという主張を表す。選択肢の全体的な評価は、関係のもっともらしさ、すなわち、主張 `Is-the-Best-Alternative-For(Alternative,Decision Problem)` で表わす。逆に、この関係のもっともらしさは、選択肢とすべての目標との間の関係 Achieves のもっともらしさに比例する。ひとつの選択肢は、その選択肢を各 Goal と結び付ける主張 Achieves のもっともらしさと、Goal の重要性について論じることによって評価される。これは、Claim を作ることによって、他の Claim に対して Supports（賛成）、Denies（反対）、Presupposes（前提）を行う Claim を作り出し、DRL で議論するということである。これらの関係（Supports、Denies、Presupposes）は、すでに述べたように「主張」であり、それらもまた、議論の対象となるものである。あるClaim のもっともらしさが Qestion（疑問）にどう答えるかによるとすれば、その Question は Claim に影響を与える。

② SIBYL を利用したセッションの例

ここでは、SIBYL が利用する知識を、人間がどう与えるかについて説明する。この知識は、DRL意思決定グラフで表される知識ベースを構成する。

(a) 課題の提出

あるグループで、Zeus というシステムの開発に使用するプログラミング言語を何にするかを決定しようとしている。グループ・マネージャの Jane は、Decision Problem（意思決定課題）事例を作成する

ことで意思決定プロセスを開始する。彼女は、画面の枠の中に全タイプを表示しているタイプ・ブラウザの中から、Decision Problem というオブジェクトタイプをクリックして作業を始める。新しい画面とメニュー項目を表示するテンプレート・エディタが呼び出されて、そのオブジェクトについて実行可能なアクションが示される。

(b) 当初の目標と選択肢の指定

エディタ中のAdd An Alternative というオプションをクリックして、考慮すべき選択肢を表す Alternative というオブジェクトを作成する。ここでは、C と Common Lisp である。同様に、Add A Goal をクリックして、最適となる選択肢が持つべき属性を表す Goal というオブジェクトを作成する。ここでは「Zeus の開発が可能」「開発コストの最小化」というオブジェクトが生成される。これらの Goal と Alternative は、SIBYL が自動的に意思決定課題例と関係づける。つぎに、この意思決定課題に対する意思決定マトリクスを作成する。意思決定マトリクスは、横軸に Goal、縦軸に Alternative が示される。各セルの値は、そのセルの Goal に対する Alternative の現在の評価を表す。はじめは、各セルとも unevaluated と表示される。Alternative に対して賛否両論の主張を作成するに従って、セルの値は更新される。

(c) グループ・メンバからの入力の受付

Jane は、意思決定マトリクスができたことをグループのメンバに告げて、各人の入力を要請する。グループのメンバは、Goal に関する Alternative を評価したり、既存の評価に対する疑問を提示したり、新たな Goal や不要な Goal を指摘したりすることによって意思決定に寄与する。

(d) 目標の練り上げ

John は、Zeus のチーフ設計者で、この知識ベースに追加を行う最初の人物である。彼は、Lisp の強力な信奉者である。Zeus はオブジェクト・システムが必要であるため開発には Lisp が良いと思っている。また、Zeus のウィンドウ・システムは X ウィンドウ・システムで書かれなければならないと思っている。X が C で書かれていることは知っているものの、X のライブラリおよびツールキットの Lisp 版である CLX とCLUE があるので Lisp でうまく行くだろうと考えている。Lisp 支持を表すために、John はまず意思決定マトリクスを検討し、「Zeus の要件を支援する」というGoal をいくつかの Subgoal に分けることにする。「Zeus の要件を支援する」と書かれたセルでマウスをクリックし、ポップアップ・メニューから Add A Subgoal を選択し、「オブジェクト・システムの提供」と「X ウィンドウでのインタフェース」という二つの Subgoal を追加する。また、Zeus には電子メールが必要であることがわかっているので、もう一つ「電子メールのサポート」を Subgoal として作成する。これらの Subgoal は、Zeus の開発要件として John が考えていることを表す。こうすれば、例えば、Zeusと似たシステムの要件を考える必要が出てきた場合など、これらの考えを検討したり、再利用することができる。

(e) 論拠の反論の追加

John は、Common Lisp と「オブジェクト・システムの提供」と「X ウィンドウでのインタフェース」という Subgoal とを関係づける意思決定マトリクス内のセルでマウスをクリックし、ポップアップ・メニューから Show Argument Browser を選択して、自分の論拠を追加する。

はじめは、そのセルに関する Alternative（Lisp）が Subgoal（X ウィンドウでのインタフェース）を達成するという主張が一つあるだけで、ブラウザのなかでは一番上の主張である。DRL の中では、関係は Claim のサブクラスであり、特に、Alternative と Subgoal をリンクする Achives（達成）という関係は、Alternative は Subgoal を達成するという主張である。この主張は、各Alternative についてSubgoal が作成されるごとにシステムが自動的に生成し、意思決定マトリクス内の各セルごとのアーギュメント・ブラウザには、はじめにたたき台としてこの主張が入る。

各人は、このきっかけとなる主張を支持または拒否する主張として、賛成または反対の論拠を表現する。すなわち、その主張のもっともらしさは支持または拒否の主張が追加されるに従って更新され、目標に対してその選択肢がどう働くかというものさしを表わす。SIBYL では、ユーザは対象上でマウスをクリックすることで、その対象について実行できる全操作のメニューを出すことができる。John は最初の主張Achives でマウスをクリックし、詳細（In particular）を出す。ある主張ノード上でマウスをクリックすれば、Add A Supporting Claim（賛成意見の追加）、Add A Denying Claim（反対意見の追加）、Add A Question（疑問の追加）、Specify A Presupposition（前提条件の特定）といった可能な操作メニューを表示することができる。ここで、 Add A Supporting Claim を選択すると、新しい Claimのためのテンプレート・エディタが表示されて、この新しい主張は Supports という関係で元の主張と自動的にリンクされる。John が Add A Supporting Claim を選択すると、SIBYL は新しい主張のためのテンプレート・エディタを表示し、新しい主張と元の主張とを Support という関係でリンクする。

このようにして、何人かのユーザがそれぞれの知識を入れた後、John は、また意思決定マトリクスを検討する。意思決定マトリクスのセルのうちunresolved となっているのは、そのセルの目標については、その選択肢の評価を出すには解決すべき問題がいくつかあるということを意味する。また、そのセルの目標についてその選択肢がどう働くかというものさしを三段階で表すことができる。この値は、中心となる意思決定者が与えることもできるし、賛成・反対の主張のもっともらしさから導き出すこともできる。John は、自分が最後に SIBYL を使ってから追加された主張のすべてを表示するように SIBYL に要求する。SIBYL は表形式または、ネットワーク形式でこれを表示することができる。また、作成者、作成日、関係する主張などそこにあるフィールドのどれを使ってでもソートすることができる。このようにして今ある知識ベースを検討したあと、その間にうかんできた疑問に答えるか、先に進んでいる主張にある程度対応したり、新しく得た知識を追加したりすることにする。最終的にこれらの検討結果を示す知識ベースを描いた意思決定グラフが作成される。

(3) 関連する研究との比較

最近、意思決定原理や論拠を表すという同様の目的を持った研究がいくつか出てきた。Argnoter は、なかでも早くに出てきたもののひとつであるが、立論の構造を明確にし、その構造における依存性の管理を補佐することでグループ意思決定を助けようとしている点で、SIBYL の考え方にもっとも近いものである。しかし残念ながら、その機能を理解するためにはどのような表現構造を用いるべきかが明確でないなど、Argnoter は上位レベルの記述に留まっている。

また、SYNVIEW は、ネットワーク上に分散したユーザ間での議論を表現しようとするもののなかで、最も初期のもののひとつである。しかし、そのために、その表現が厳密に制限されており、インデントがテキストにあると、時には選択肢となったり、証拠関係となったりするのである。SYNVIEW は、合意の物差しを提供しようとしている点では興味深い。

Potts と Bruns は、設計歴を設計の産物の交代、理論的解釈のセットとして捕捉すべきものとし、逆に、それぞれは IBIS 構造によって捕捉されると提唱した。彼らは、この構造を使ってソフトウェア設計の例を表した。SIBYL の DRL 構造は、理論的解釈を表すもう一つの構造とみなすことができる。

gIBIS (graphical Issue Based Information System) は、「調査方針議論のためのハイパーテキスト・ツール」である。gIBIS の目標は、設計原理、すなわち、「設計の問題点、代替解決案、これらの代替案間のトレードオフ分析、意思決定過程において行われた仮または正式の言質」の捕捉にある。gIBIS は良く知られ、その目標とするところも SIBYL とまったく同じであり、gIBIS が使う IBIS 構造は、他の多くのシステムで採用されている。SIBYL と gIBIS とは、設計や意思決定の過程で蓄積する知識を表現し、見直しや再利用を可能にするという共通の目的を持つ。SIBYL、gIBIS のどちらも、議論を通して選択肢を評価することを伴うある種の課題、すなわち、意思決定に共通の対象と関係をあらわす用語を持つ言語を提供することで、この目的を達成している。

SIBYL は gIBIS を拡張したものと考える。gIBIS も SIBYL も、構文上の型（テキスト・ノード、イメージ・ノード等）だけを持つ単純なハイパーテキスト・システムと、すべての知識がシステムによる自動処理のために形式に従って表現される完全な形の知識ベースとの中道をとっているのであるが、SIBYL は、gIBIS に比べ、知識ベース寄りである。gIBIS は主にハイパーテキスト・システムであり、そのサービスの焦点が構造の表現におかれており、何を表現しているかという構造を理解し、新たな知識を加えることが簡単にできるようにすることを目標としている。一方、SIBYL は、主として知識ベースのシステムで、セミフォーマルな表現を使うので、知識のすべてを形式化することを強制しない。SIBYL が提供するサービスは、いわば、知識ベース・システムに一般的な、依存性管理、不確実性管理、ビューポイント管理、優先度管理である。

SIBYL と gIBIS が提供使用としているサービスの違いは、互いの持つ用語の違いで説明できるかもしれない。gIBIS でいう Issue は、DRL では Decision Problem といい、同様に Position は Alternative、

Argumentは Claim という。しかし、gIBIS に欠如しているという点で特に目だつのが、選択肢が評価されるべき目標や目的という概念である。gIBIS では、対象は暗黙のうちに表されるだけである。例えば、「Lispを使用する」という立場を「CLOS があるので Lisp はオブジェクト・システムを提供する」という論拠を使って支持することができる。この関係から、オブジェクト・システムの提供が目的なのだということは、おそらく類推されるだろうが、なぜそれが目的なのかとか、他の目的とどう関わっているのかは明確ではない。さらに悪い点は、中間的な目的については論拠の中でまったく触れられないことが多いということである。例えば「C には C++ がある」という論拠が「C を使用する」という立場を支持する場合を考えてみると、この場合オブジェクト・システムを持つという目的は明らかにされていないので、それについて論じることは難しい。また、「オブジェクト・システムの提供」という目標を明確に表わさないのでは、片や CLOS に言及し、他方が C++ に言及している前出の二つの論拠は、選択肢の同じ属性についての比較を行っているのだということがはっきりしない。こうした理由から、少なくともある程度、人々を合意へ導く手助けをするに当たっての gIBIS の難しさが説明できる。

それゆえ、目標を明白に表現することの利点は多い。そうすることで、考えを表現し、選択肢を評価すべき対象がわかるようになる。明白に表現することは、また、その目的について議論したり、必要に応じてそれを変更したりできるということである。目標を変えようとするなら、仮に「オブジェクト・システムの提供」は「Zeus の開発が可能」という目標のサブ目標としてはもはや重要ではないとすれば、SIBYL では、そのサブ目標を削除する（または、その重要度をゼロにする）だけでよい。そうすれば、それに関連するすべての主張が自動的に消されるが、gIBIS ではこうした変更を受け入れるモジュラーがない。

目標は、SIBYL のサービスすべての中で重要な役割を果たしている。目標を明白に表現することによって、その目標が変更されたとき、依存性管理機能が知識ベースを更新できるようになっている。例えば、このシステムを教育的なものにするという目標の重要度がかなり高くなるとすると、Pascal が選択肢の一つとして登場する。Is-the-Best-Alternative-For という関係のもっともらしさは、その目標の重要度と、ある選択肢をその目標のそれぞれと結びつけている Achives 関係のもっともらしさとに比例している。gIBIS では、なにが目的であるかを語ることができないので、それぞれの重要度の違いもそのままである。

その他にも、関係というものが、議論の対象となり得る上位クラスの対象であるかどうか等、SIBYL と gIBIS の構造的違いはいくつかある。例えば、gIBIS では、「A と B には賛成であるが、A が B を支持するということには賛成できない」とか、「G を目標にすべきであるとは思わない」などということはできない。全体的にみて、DRL の構造はより表現力に富み、gIBIS が人間の使い勝手の良さに主として動機を持つの対し、SIBYL は提供するサービスに動機づけられているように思われる。

8.7.4 Project Nick

原文タイトル： Project Nick: Meetings Augmentation and Analysis

翻訳タイトル： Nick プロジェクト：「打合せ」の支援と分析

著　者： Peter Cook
Clarence Ellis
Mike Graf
Gail Rein
Tom Smith
(Software Technology Program,MCC,Austin,TX,USA)

出　典： ACM Transactions on Office Information Systems, Vol.5, No.2, April 1987, pp.132-146

Nick プロジェクトは、MCC の Software Technology Program の中の Design Interface Groupによって行われたシステム設計の領域における打合せの分析と支援に関する研究である。

本稿では、Nick プロジェクトで検討中のいくつかのアイデアとそれらの実現方式が紹介されている。まず、いくつかの定義と研究目的が述べられ、つぎに、本研究の構成要素について説明される。ここでは打合せ進行モデルを提示し、構成要素はこのモデルに沿って議論されるが、このモデルは Nick プロジェクトおよびシステムの本質となっているものである。さらに、打合せという概念およびポットの理論について定義を行うとともに、この定義を用いて打合せを分析し、典型的な「打合せの問題点」を発見する方法が概観される。また、プロジェクトの打ち合わせ内部での側面が説明される。電子的な打合せ支援機能が説明され、打合せ情報マトリックスについて議論されている。

(1)　定義と研究範囲

打合せとは、二人以上の人間が協力して行う、構造化されたコミュニケーション活動であると定義される。これらの集まりは、法人団体におけるあらかじめ予定された正式な会合(取締役会等)から、特定の問題を話し合うため二人の同僚が非公式にアドホックな形で打合せをすることまで、幅広い範囲を持つ。打合せの参加者は空間的および時間的に同じ場所にいる場合もあれば、いない場合もある。つまり、分散会議もここでの定義の中に含まれている。また、非リアルタイムの電子会議およびメールシステムは、会議参加者が実際に同時点において打合せ資料を見ていなくても良いような打合せの形式である。

本研究では直接の面談による打合せを対象としている。その理由の一つは、MCC における一般的な関心と資源である。また、ここでは主として小規模の設計打合せを対象としている。つまり、最小限二人の設計者が解決策を検討しているような場合から、最大 50 名程度の人々が集まり、ネゴシエー

ション、問題解決、情報交換を行う場合までを想定して、打合せ活動を支援することを対象とする。

また、本研究は主として、設計チームの打合せを対象とする。設計チームの活動には、ブレーンストーミング、探索、設計構造の定義、問題の分析、タスクの割当て、問題解決などがある。設計に関する打合せは、効率性と有効性についてはよく分かっていない。これは特に、責任がまだ明確になっておらず、組織構造が変わりやすい設計の初期段階において言えることである。

(2) 主要構成要素

本研究では、理論の構築とシステムの構築の二つの面のアプローチをとっている。この両面は、並行して進められ、また互いに相補いあうものである。

まず、理論構築の側面では、設計プロジェクトのビデオテープを観察することや筆者自身のグループにおける打合せにおいて実験することによって得られた洞察を基に、打合せの主要要素を取り出しモデル化を行っている。これを以下に紹介する。

「打合せ」には、獲得コンポーネント、分析コンポーネント、プレゼンテーション・コンポーネントの三つの主要要素があると考えられる。一般的に、ある打合せで生成された情報は、獲得され、分析され、次の打合せで利用されたり、プレゼンテーションされたりする。これをモデル化する際に、一つの打合せを円で示し、その打合せに入／出力される情報の記憶装置（および処理）を箱で表す。そして円から箱（あるいは箱から円）への矢印は、情報の流れを意味する。ある打合せXの円への矢印の出所が示されていないのは、これらの情報源が指定されないことを示している。この円から出る矢印は、出力情報を示し、矢印の終端にある箱は、情報のリポジトリである。情報リポジトリのそれぞれは、打合せ情報のある一部を獲得する。避けられないことであるが、打合せで生成された情報の一部は、獲得されないままで終わる。打合せを撮影したビデオテープでさえも、参加者全員の表情までは記録しない。したがって、打合せXの円から出る矢印の中には、いかなるリポジトリにも届かないものがある。

このモデルでは、知的で高度な情報解釈選択機構を持つリポジトリとそうでないものとの間を区別していることが重要である。高度なフィルタを持つリポジトリを区別するために、ポット（瓶）という用語を作った。ポットは、箱の入り口に斜線を付けることによって示される。「ビデオ」というラベルを持つ箱は、技術的な支援機構の例（ビデオテープもしくはビデオディスク）である。これは打合せにビデオテープもしくはビデオディスクを利用したことを示している。「パブリック・ポット」というリポジトリは、打合せの過程でホワイトボードに記入された情報、あるいは公式な議事録に記録された情報を示す。これは、パブリック・メモリとも呼ばれ、会議の過程ですべての参加者が見ることができるものである。

分析コンポーネントは多くの異なったリポジトリからの情報を総合し要約する方法を研究するもの

である。本モデルでは、この分析は打合せの途中ではなく、打合せが終了した後に行われることを前提としている。このコンポーネントには、打合せのシンタックス（打合せの進行過程）およびセマンティックス（技術的内容）を獲得し、理解し、改善するための情報分析が含まれる。また、効果的なプレゼンテーションを行うための、入力情報の変換方法の研究も含まれる。例えば、ビデオテープは有効な獲得手段であるかも知れないが、入力情報を注意深く再編集しない限り、プレゼンテーション手段としては時間がかかり不便であるように思われる。打合せの過程の特定の部分にランダム・アクセスできるビデオディスクは、より優れたテクノロジである。典型的な変換手段としてはタイプ入力から音声出力への変換がある。

分析コンポーネントから出る矢印は、分析され変換された情報を示す。この情報は、電子的な形で入手可能であり、次の打合せにおけるプレゼンテーションのための情報源として利用可能である。プレゼンテーション・コンポーネントは、コンピュータ化された情報の表示だけでなく、ブレーンストーミングや、黒板の前での創造的かつアドホックな「おしゃべり」のための、構造化されたプリミティブを利用可能にすることも考慮している。設計者たちはしばしば、箱、円、リスト、マトリックスなど、ある種の構造体を利用する。

(3) 打合せの分析

本研究では、打合せ要素の間のスタティックおよびダイナミックな関係を説明するために、ポット理論を開発し洗練しつつある。打合せの概念を定式化する定義と、ポット理論の説明を以下に述べる。

打合せとは、八つの要素の組 M = < G, R, P, I, A, S, C, E, >であると定義する。それぞれの要素は次のとおりである。

G: 打合せの目標の集合
R: 打合せで利用する資源の集合
P: 打合せの参加者の集合
I: 打合せで操作される情報
A: 打合せの過程で発生する活動の集合
S: 打合せのタイプ、議題、議事進行規則等、打合せの構造
C: 打合せが行われる状況
E: 打合せに周辺的な影響を与える外部要因

集合 G は打合せに参加する人たちが心に抱いている目標ではなく、組織としての目標である。前者は、参加者集合 (P) の一部分である。集合 P の個別要素は、参加者のプロフィールであると考えることができる。つまり、各参加者の目標、能力、および信念の記述である。集合 P についてもう一つ注

意しておくべきことは、参加者とは人間である必要はないということである。

資源Rは、打合せの間に利用され、あるいは利用可能である実体的なオブジェクトである。例としては、ハードウェア、ディスプレイ装置、あるいは会議室の椅子やテーブルである。他方、情報Iは、文書、議事録、および打合せの多種多様な入力データ、出力データである。

Aは打合せの間に発生する活動の集合であり、コミュニケーション、知覚、相互作用を含む。これらにはエピソードのようにマクロなレベルで記録することができるものがあるし、あるいは打合せ相互作用グラフのようにより詳細なレベルで記録されるものもある。また、身振りなどのような非言語的行動もある。

打合せの構造Sは、議題、議事進行規則、あるいは会議の予定時間などを含む。また、議長、書記のような役割と、どの人がどの役割を務めるかという指定が含まれる。

打合せの状況Cとは、非常に豊かな構造を持つものである。これはその打合せに関係する組織的構造、社会的構造、および歴史的、政治的、経済的要因を含む。集合Eは、その打合せに直接関係しない外部的要因を含む。

上記の要素の多くは、打合せの進行と共に、進化発展する。集合I（情報）とP（参加者）は特に変化しやすい。打合せは、I,P,Rを主要入力とする一つの関数と考えることができる。この関数は、G,S,Cの影響のもとに、入力に対して、Aによって記述されるある種の変換を加える。打合せの出力のうち最も目に見えやすいものは、生成されたり変換されたりする情報（I）、および、参加者（P）の知識の変化、態度の変化である。しかし、多くの場合、他の集合も変化する。目標は再検討されるかもしれない。議題は変更されるかもしれない。参加者間の関係は変化するかもしれない。あるいは組織構造について重要な決定がなされるかもしれない。

ここでは打合せにおける抽象的かつグローバルなリポジトリNの存在を仮定し、それをヌーメノン(noumenon)と呼ぶこととする。ある特定の打合せのためのA（活動）およびI（情報）のすべての要素は、このヌーメノンの中において存在すると考える。ヌーメノンへのアクセスは、現象(phenomena)、つまり打合せにおける活動と情報を通じてのみ行われる。そして、いかなる個人もそれらのすべてを吸収することはできない。したがって、個人のリポジトリ（個人は打合せでの出来事に高度な解釈を施すから、そのすべてはポットである）は、すべてヌーメノンの中からフィルタをかけられ取り出された部分集合もしくは抽象化されたものである。二人の参加者の個人的ポットP1,P2の内容を考えて見よう。それは、フレーム、つまり打合せの過程においてこれらの参加者が重要であると認識したところの、構造化された要素を含んでいる。さらに、個人的ポットの要素からヌーメノンの要素への写像（P1—>NおよびP2—>N）を考えて見よう。これら二つの写像の組合せから、二つの集合P1,P2の間の写像を作ることができる。この合成写像の良さが、これら二人の参加者間のコミュニケーションの良さの尺度であると仮定する。

(4) インプリメント

本プロジェクトの電子的打合せ支援設備は、当初次のような構成であった。

① 打合せの参加者一人一人に提供されたパソコン、キーボード、マウス、プライベートな表示画面。

② 会議室の中でパソコンを結ぶLAN。このネットワークは、建家全体をカバーする大きな電子ネットワークに接続されており、会議室の中には準備されていない情報に対しても必要に応じて即座にアクセスすることができる。

③ 参加者全員が見ることのできる協同作業用表示装置。打合せ支援システムの出力を表示するためのソフトウェアが用意されており、これによってメンバは打合せの結果や進行状態を正確に知ることができる。

④ 議題情報を表示し、参加者間の意味のあるコミュニケーションを提供し、グループ・メモリへの情報入力を可能とするソフトウェア。

⑤ 参加者間の二つのサブチャネル。一つはグループ・プロセス情報用であり、他方はグループの情緒的状態を登録するためのものである。

⑥ 打合せに関連した統計情報を獲得し、処理するソフトウェア。これによって打合せの有効性についてのいくつかの側面を定量化することが可能になる。

多様な打合せの構造のそれぞれにおいて、打合せの継続中に表示され、操作され、獲得される情報のタイプは、どの程度の構造化がなされているか（バイナリ、構造化、非構造化）、また情報の共有がどの程度許されるか（完全なプライベート性保護、サブグループ内部での公開、完全な公開）に対応して分類されるであろう。このカテゴリを表8.7-1のマトリックスに示す。

表8.7-1 打合せ情報マトリックス

	プライベート情報	サブグループ情報	公開情報
バイナリ情報	リストのチェックマーク	指示フラグ	投 票
構造化情報	リスト	促進用メッセージ	プレゼンテーション、リスト
非構造化情報	スケッチ図	ノートとコメント	電子黒板上のスケッチ図

プライベート情報とは、会議の間に参加者個人によって獲得される情報であり、打合せの席上では他者に開示されない（例えば、非常にカジュアルなメモ等）。これは会議中自由に使用できるパーソナル・ワークステーションを介して獲得される。サブグループ情報とは、会議の間、二人以上の個人によって共有される情報である。通常、会議中秘密のメモを回覧することは悪いマナーであると考えられている。しかし、パーソナル・ワークステーションを利用して、一人または複数の他者にメモを

送る可能性は、より高度の並列性を持つ新しい打合せスタイルの開発へ導くかもしれない。

バイナリ情報とは、イエス／ノウ、真／偽など１ビットで伝達することのできる情報である。このタイプの情報は、参加者にとって作成、送信、受信が特に容易である。表8.7-1に示した打合せ情報マトリックスの１行３列目には「投票」とある。これはバイナリのパブリック情報チャネルをサポートする電子装置を利用して行える打合せ活動の一つの例である。打合せ参加者は、自分の意思に応じて、自分のワークステーションのボタンを押すだけで、現在討論中の問題に関する無記名投票を行うことができる。イエス／ノウは、ネットワークを介してパブリック・マシンへ伝達される。投票集計結果は即座にパブリック表示画面に表示される。

構造化情報とは、定形化されており、コンピュータが即座に解釈することのできる情報（通常は非バイナリ情報）である。このタイプの情報には、リスト、マトリックス、テンプレート、フローチャートなどがある。表8.7-1のマトリックスの２行２列目には、「促進用メッセージ」とある。これは、参加者と進行役の間でやり取りされる定形メッセージの一つである。これはサブグループ・コミュニケーションの一例である。参加者全員がこのメッセージを見る必要はないからである。進行役は、「そろそろ次の議題へ進むべき時ですよ」というメッセージや、「現在の話題についてぜひ発言させて欲しい」というメッセージを受け取る。これらのメッセージは事前に用意されており、一つのキー・ストローク、マウス・クリック、タッチ・スクリーンへの指の接触、あるいはその他簡易なユーザ・インタフェース技術によって起動されるべきである。本システムでは、気分表示メッセージをインプリメントしている。これは誰でもいつでも起動することができる。これは、「退屈したよ」「この打合せは非常に興味がある」「そろそろコーヒーブレークが必要だ」などのメッセージである。これらのメッセージによって進行役は会議を生産的、効果的に運営するための判断をおこなうことができる。ある種の条件のもとにおいては、電子黒板あるいは何らかのパブリック表示媒体上のムードメータを介して、これらの入力メッセージの平均値を公開して表示することも役立つかもしれない。そうすれば、これは構造化された公開情報（２行３列目）の一例となる。

非構造化情報とは、定形化されておらず、コンピュータが即座には解釈できない情報である。設計者にとっては、これはしばしば、パーソナル・ワークステーションの画面上に描くプライベートなスケッチ図（３行１列目）あるいは、電子黒板上のパブリックなスケッチ図（３行３列目）の形を取る。電子的媒体上でスケッチすることの利点は、スケッチによって作られるビットマップは保存、再生、連続的編集（構造体に変換された場合）が可能であるという点である。グラフィカルなソフトウェア設計支援ツールや打合せ用のグラフィックスとアニメーションの開発には特に重点が置かれている。ソフトウェア設計者のためのグラフィカル言語の開発が進歩すれば、これらのスケッチ図をコンパイルし、構造化情報として取扱うことが可能になることが期待される。これによって、打合せ分析のフェーズで利用できる構造化情報が拡大されるであろう。

8.8 協調作業システムの設計支援

8.8.1 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。

8.9 ツールキット

8.9.1 LIZA

原文タイトル： LIZA: An Extensible Groupware Toolkit

翻訳タイトル： LIZA：拡張可能グループウェア・ツールキット

著　者： S.J. Gibbs
(Software Technology Program, MCC, Texas, USA)

出　典： CHI '89 Proceedings, May 1989, pp.29-35

本稿では、まずグループウェアの定義が行われた後、「アクティブ・オブジェクト」に基づくグループ・ツールのモデルが提示される。このモデルは、LIZA として知られる拡張可能グループウェア・ツールキットの設計と実現に使われている。マルチ・ユーザのインタフェースを構築するシステムである LIZA について紹介され、さらに LIZAで構築したグループ・ツールの例について述べられている。

(1) 本プロジェクトの研究範囲

グループウェアと言えば、ユーザがグループである場合のソフトウェアを指すが、ここではさらに特定して次のように定義する。

「共通のタスクについて（同時に）作業する二人以上のユーザをサポートし、共有環境にインタフェースを提供するソフトウェア・システム」

この定義においては、「共有環境」の概念が重要である。この場合共有環境とは、一人のユーザの行動がもう一人のユーザに見えることを意味する（例えば通知メカニズム等によって）。わかりやすい例が、共同執筆編集システムや、マルチ・プレーヤ・ゲームである。データベースのようなものは、多くのユーザが一つのデータベースにアクセスして一つのタスクについて作業を行うが、ユーザが他のユーザの行動を知らない（また実際にデータベース・システムはそれを保証するようになっている）ため、グループウェアではない。

本研究では、リアルタイムのグループ・ツールに焦点を当てて LIZA の開発を行った。これによってマルチ・ユーザ・インタフェース設計の問題（すなわち、グループ・オペレーション、WYSIWIS、社会的プロトコル対ソフトウェア・プロトコルなどの問題）を明らかにできることを望んでいる。

(2) LIZA ツールキット

LIZA は、LAN 上に接続されたハイパワーのパーソナル・ワークステーションで実行される。「セッション」とは、LIZA を使う参加者グループのことで、分散していたり（参加者がそれぞれのオフィ

スにいる）、集中していたり（参加者が同じ場所にいる）、この二つの組合せであったりする。会議開催コールを行って、物理的に離れた参加者間に聴覚チャネルを設定する（スピーカホンかマイク付きヘッドホンを使う）。集中型参加者は、特別な電子会議室を使うことができる。この会議室には、多数のワークステーションと、ワークステーション・ディスプレイを背面映写する大型（5インチ×7インチ）スクリーンが設置される。

① アクティブ・オブジェクト

LIZAのツールは協同作業を行う一組の「アクティブ・オブジェクト」である。アクティブ・オブジェクトとは、オブジェクト指向のプログラミング・パラダイムを拡張してオブジェクトの同時実行を可能にしたものである。アクティブ・オブジェクトは、Hewittのアクタ理論や、EmeraldおよびKnos等、アクティブ・オブジェクトをサポートする最近の言語と密接な関係がある。

分散制御や並列処理や分散知能が必要な場合、アプリケーション内にアクティブ・オブジェクトを採用する。例えば、鳥や魚の群のような動物グループの動きをモデル化するのには、アクタ・ベースの言語が使われている。集合体の各メンバはアクタによって表され、その行動によって集合体の動きのアニメーションを駆動する。Knosは、「メッセンジャ」オブジェクトの指定に使われ、行き先を探索して電子メールを配達するのを補助している。Knosは、ネゴシエーションを行うオブジェクトの行動を指定するのにも使われている。

LIZAのグループ・ツールによって、グループ・メンバが同時に作業できるようになる。システムの最も基本的なレベルに同時性が必要であるため、LIZAの構築にはアクティブ・オブジェクトが適していると考えられる。

② ツール構造

LIZAでは、各グループ・ツールは「カーネル・オブジェクト」と呼ばれる一つのアクティブ・オブジェクトと、「イメージ・オブジェクト」と呼ばれる多数のアクティブ・オブジェクトからなる。イメージ・オブジェクトは、ツールのユーザ・インタフェース部分を担当し、一般にツールを使う参加者一人につき一つのイメージ・オブジェクトがある。参加者が有意な行動（ボタンを押したり、メニューを選択したり）を取ると、イメージ・オブジェクトが「イベント」の信号を送り、カーネル・オブジェクトがこれを捕まえる。カーネル・オブジェクトは、イメージ・オブジェクトから入って来るイベントに対する対応を処理、調整する。カーネル・オブジェクトは、一つ以上のイメージ・オブジェクトに対して「ディスプレイ指令」を送り返して対応することもある。

グループ・ツールを形成するオブジェクトは、一つのツール・クラスのインスタンスである。ツール・クラスの一般的な構造を定義するためにテンプレートを使うこともできる。このテンプレートは、上位クラス、クラスのインスタンスが保持する変数、ツールの行動指定（述語）を定義する部分からなる。

グループ・ツールの全てのクラスは、イメージ・オブジェクトを生成／抹消する方法（参加者がセッションに入ったり抜けたりする場合に必要）等の機能を持つ上位クラスを継承している。ツールの行動を指定する述語（メソッド）は四つのタイプに分かれており、最初の二つは、カーネルーイメージ間のインタフェースを形成するための述語である。カーネル・オブジェクトがイメージ・オブジェクトから送られたイベントを処理する述語と、イメージ・オブジェクトがカーネル・オブジェクトから送られたディスプレイ指令を処理する述語がある。次の二つは、ツールが他ツールに提供するパブリック述語と、ツール内部で使うプライベート述語である。

③ システムのアーキテクチャ

LIZA は、多数の「クライアント」と一つの「サーバ」からなる。イメージ・オブジェクトはクライアント上で動作し、サーバはカーネル・オブジェクトを動作させる。サーバとクライアントの間の連絡は、Unix のソケット・メカニズムを介して行われる。

このようなアーキテクチャを選んだのには、単純さと即応性という二つの理由がある。集中型サーバでグローバルな情報を維持することで、同時制御と通知が容易になる。一方クライアントも、多くのユーザ・インタフェース・オペレーション（ウィンドウ再表示やメニュー表示等）を自動的に実行できるので、即応性が失われることがない。

LIZA は、サーバとクライアントに新しいツール・オブジェクトを追加することで拡張可能である。例えば現在の LIZA システムには 8 個のツールがあり、それぞれが既存システムに接続されたものである。もう一つの拡張方法としては、イメージ・オブジェクトからカーネル・オブジェクトへの通信にメッセージではなくイベントを使うことが挙げられる。イメージ・オブジェクトがイベントの信号を送ると、そのイベントは、関連するカーネル・オブジェクトの処理述語で処理されることになる。

(3) グループ・ツール

ここでは、LIZA の 8 個のツールの内の 4 ツールの例と投票ツールについて説明する。

① ツールの例

(a) ルーム・ツール

ルーム・ツールは、セッション参加者の位置マップを示す。参加者は、それぞれオフィスや会議室にいる。オフィスのレイアウトは物理的な位置には対応していないが、各オフィスには部屋番号や電話番号が付いている。会議室内では、参加者は会議テーブルの物理的位置に応じて配置されている。

(b) グラフ・ツール

グラフ・ツールを使って、ユーザ・グループは共有グラフを編集することができる。このツールは、Cognoter に似ている。オペレーションとしては、生成、削除、ノードのリンクやリンク解除、ノード内容の更新やノードの移動、コピー、サイズ変更がある。ノードはそれぞれ、プライベート、パブリ

ックまたはチェック・アウトの三つの状態のいずれかになっており、オペレーションでノード状態を変更する。

(c) スライド・ツール

スライド・ツールを使って、参加者は一連の「スライド」イメージをロードしたり複数のスライドを流し読みしたりできる。このツールは、スライド選択用ボタンのあるウィンドウと、スライドを表示する「エピソード・ウィンドウ」の二つのウィンドウからなる。参加者が特定のスライドを選ぶと、全参加者のエピソード・ウィンドウにそれが表示される。

(d) メタ・ツール

このツールは「操作盤」を表し、横の各段が参加者、縦の各列がツールに対応している。項目はマウスで選択することが可能で、特定のユーザがツールをスタートさせたりストップさせたりするのに使える。これは、一部のメンバがシステムをよく知らない場合、経験のあるユーザが手助けできるため、有益である。

② 投票ツールとこれに伴う社会的問題

ここでは、LIZA を使って投票ツールを設計する場合を考えてみる。参加者の一人が投票ツールを起動すると、「投票ウィンドウ」と呼ばれるウィンドウが全員のディスプレイに現れる。投票ウィンドウには、投票すべき問題を入力するタイプイン領域と、「投票開始」、「投票終了」の二つのボタンがある。前者を押すと、参加者全員のディスプレイに「スコア・ウィンドウ」が現れ、後者を押すとこれがウィンドウから消える。スコア・ウィンドウには投票する問題が表示され、「はい」、「いいえ」、「どちらともいえない」、「棄権」の4種類から一つを選んでマウスで入力できるようになっている。ユーザの現在の選択が強調表示される。スコア・ウィンドウはグループの投票状況を示す棒グラフも表示する。投票開始時は全投票者が「棄権」であるが、投票が進むにつれて棒グラフが変っていく。

この投票ツールは非常に単純なもので、LIZA 内に数ページのコードを記述するだけで実現できたが、グループ・ツールの設計者が、ツールの利用に影響する社会的組織的問題をいかに認識していなければならないかをよく表している。投票ツールの設計の際に見過ごされた問題には次のものがある。

(a) 途中結果

この投票ツールは、参加者が進行中の投票結果を見ることができるため、投票に影響を与えることがある。ツールを使用すべき背景を理解していないと、このような影響をそのままにしておくべきか、全員が投票を終わるまで合計がわからないようにする「目隠し」投票にすべきかどうかがわからない。

(b) 匿名性

スコア・ウィンドウは匿名性を保持し、自分のスコア・ウィンドウを見ているユーザが他ユーザの投票を決定することができない。しかし、他ユーザのウィンドウを盗み見したり、他の参加者による

マウスのクリック音をスコア・ウィンドウでの変更と結びつけたりして見当をつけることはできる。

(c) 投票提案

この投票ツールでは、どのユーザも投票のための議題を提起することができる。これは、それぞれの地位が同等のグループには適しているかもしれないが、民主主義をそれほど高度に発達させていないグループも多い。また、同等のグループの中でも、ある問題に関する投票を強制されることを好まない参加者がいることが考えられる。そのため、誰が投票議題を提案できるか、いつこれを行うかは、投票ツール設計者が考慮しなければならない問題である。

(d) 投票変更と棄権

この投票ツールでは、ユーザは投票を変更したり取下げることさえできる。このツールを、コンセンサスを得るための状況に使うのであれば、変更も有益だが、得票集計に使う場合は、変更によって結果がゆがめられることがあるので、望ましくない。

社会的要因に対する感受性では投票ツールの例はやや極端だが、一般的にグループ・ツールには、程度の差はあっても同様な問題が起こることが予想される。社会的組織的要因は、グループ・ツールの容認に重要な役割を果すため、まずLIZAのようなシステムをグループ・ツールのプロトタイプ化に使って、実験的にグループによる受入れを評価していくアプローチが考えられる。

8.9.2 Strudel

原文タイトル： Strudel - An Extensible Electronic Conversation Toolkit

翻訳タイトル： Strudel — 拡張性ある電子会話ツールキット

著　者： Allan Shepherd
Niels Mayer
Allan Kuchinsky
(Hewlett-Packard Laboratories, Palo Alto, CA, USA)

出　典： CSCW '90 Proceedings, October 1990, pp.93-104

Strudel は、会話と行動の管理のための支援システム構築用部品群からなるツールキットである。本稿では、まずStrudel の概要が述べられた後、利用シナリオ例に基づいてStrudel の概念モデルが説明される。また、アクティブ・メッセージ・システム相互間の相互運用可能性等の問題についても手短に説明されている。

(1) Strudel の技術的アプローチの概要

Strudel は、エンド・ユーザが電子メールに基づく会話と行動項目を管理するための部品からなるツールキットを提供する。ここでは、ツールキットの部品の設計について現在のプロトタイプの例とともに説明する。このツールキットにおいては、標準化、ツールキット部品のユーザによる拡張可能性、既存のアプリケーションとの相互運用可能性、および広範囲に普及したプラットフォーム上での実行時性能の良さ等に重点が置かれている。Strudel は、既存システムとコンパチブルである。例えば、既存の電子メールのユーザ・エージェントと同様のルック・アンド・フィールを持つ。その目的は、UNIX、X Window システム、および ARPA Internet メール等の標準的ソフトウェアが動作する相対的に低価格なグラフィック・ワークステーションの上で、グループウェア・アプリケーションを構築可能な、小型で高速でポータブルな、C 言語によってインプリメントされたプラットフォームを構築することである。さらに Strudel は、OSF/Motif UI Toolkit の提供するグラフィカル・インタフェースを利用することによって、ユーザに受け入れやすくする配慮がなされている。カスタマイジングを支援するために、Strudel では、Winterp に基づく拡張言語が提供されている。

Strudel は、会話（conversation）、会話手番（conversational move）、行動（action）、行動項目（action item）、通知（notification）等に対するユーザによるカスタマイズ可能な定義を含む部品ライブラリを持つ。プレゼンタによってこれらのオブジェクトの表示、編集、検索が可能である。

既存の研究は、到着する大量の電子メールをメッセージ・フィルタ用のルールによってメッセージ・クラスに分類し、ブラウザによって表示し、さらに、「X からのすべてのメッセージをY へ転送

せよ」など、あるクラスに属するメッセージに適用する行動を指定できるようにしたものであった。Strudel では、この研究をさらに拡張し、ユーザによる拡張可能なメッセージ・タイプおよび会話部品の管理が可能なライブラリ機能を提供することによって、ユーザがメッセージを作成中にメッセージへ構造を追加することを可能にしている。

また、Strudel においては、メッセージはWinograd の「行為のための会話」における Request のようなタイプ化された会話手番を含めることができる。ここでは、メッセージ「タイプ」ではなく「会話手番」という用語を使うが、これは、一つのメッセージには複数の「手番」が含まれることがあり得るということと、それぞれの手番は会話の相手によって典型的に取られるであろう次の手番を示唆するからである。半構造化されたメッセージの場合のように、手番はフィールドを持つことができる。例えば Request Meeting という手番は、日付、場所および議題のフィールドを持つ。手番は行動を含むことがある。例えば Request Meeting という手番は、Add to To-Do List という行動を含み、これはその手番のフィールドをアクセスするかも知れない。ユーザはトップレベルのメニュー、ボタン、あるいは以前のメッセージに表示されたメニューや項目リストから手番を選択することによってメッセージを作成する。

メッセージが集まると、連続した会話手番の間がつながれ、会話が構成される。会話の中で、従来の電子メールのメッセージと Strudel のメッセージを自由につなげることができる。従来の電子メールのメッセージは、既存の In-reply-to や、Subject フィールドのエントリを利用して、可能な限り前のメッセージにトレースされる。

Strudel が従来のアプローチと異なっているのは、ユーザがダイナミックに会話手番と会話タイプ定義を発展させることができる点である。したがって、これらの定義を特定のタスクに合わせた特別のものとすることが可能である。例えば、ユーザは、Ask who is responsible for repairing a medical instrument defect という新しい会話手番タイプを作成することができる。ユーザはその後、この手番を Medical Instrument Defect Repair という会話タイプの最初の手番として追加することができる。しかし、現在のところ、新しい手番定義や会話を中央集中化されたライブラリに統合することはほとんどサポートされていない。Strudel は、エンド・ユーザが、会話とタスクにおいて、次の手番を選択し作成することを支援するための単純なスクリプトのみをサポートしている。

メッセージやその他のオブジェクトによってサポートされている「行動」も、特定のタスクに合わせた特別のものとすることが可能である。他のツールに対するインタフェースもサポートされており、会話の中で指定された行動とタスク・オブジェクトは、他のツールによって管理されたオブジェクトへの操作を開始することができる。特に、会議室予約システムなどの汎用アプリケーション、および、ソフトウェア保守や欠陥追跡システムなどの特定領域の協調ツールへの簡単なプログラム・インタフェースを定義することができる。例えば Request meeting メッセージは、Make room reservation という

行動を持つことができる。この行動は、会議室予約アプリケーションの手番を起動することができる。

ユーザがメッセージ・システムを自分のタスクと結合できるようにするために、Strudel ではユーザは「行動項目」を作成することができる。行動項目とは、彼らが実行したいと考える活動、およびその進行状態を記述するためにユーザが作成するメモである。例えば、ソフトウェア開発技術者は、欠陥修理の予定、その優先順位等を記入した行動項目を作成するかもしれない。さらに、Strudel ユーザに外部ツールで発生したイベントについての情報を知らせるため、そのツールからの呼出しによって「通知」と呼ばれる特別の行動項目を作成することができる。会話の中におけるメッセージと同様に、行動項目は、半構造化されタイプを持つフォームとして表現され、タスクの中に結び付けられる。行動項目フォームは、行動とは別のものである。

Strudel の会話タイプは、その次に行われる手番のタイプを制約するものではない。したがって、会話の任意の時点で、異なった会話タイプを自由に開始することができる。例えば、欠陥の解決を話題とする会話において受け取った Defect notification への返事として、Request meeting や Post design issue などの手番を送ることが可能である。

(2) Strudel 利用のシナリオ例

これまでプロトタイプ開発されたシナリオをサポートするために、いくつかの会話タイプがライブラリに定義された。これらのタイプには、設計問題の討論に利用される会話型の IBIS、「行為のための会話」、会議スケジューリング、ソフトウェア欠陥追跡および修理などがある。

ここでは、欠陥修復のシナリオを例として用いる。このシナリオにおいては、ソフトウェア開発システムが Strudel に製品組立てにおいて欠陥が発見されたと通知する。その開発プロセスに責任を持つソフトウェア技術者は、Strudel を利用して、ペンディングになっている通知レポートを読む。欠陥通知は、グラフィカルなフォームとして表示される。フォームの上のボタンによって、ソフトウェア技術者は、特定のタイプのメッセージを作成し、送ることができる。例えば、いく人かの開発技術者に、その欠陥の処理に責任のある人は誰であるか知っているかと問い合わせるメッセージなどが考えられる。一人の開発技術者が、その欠陥は自分のものだと認めるメッセージを返して来る。また、この開発技術者は、メッセージ・フォームの上の別のボタンを押し、修理タスクのための最初の行動項目フォームを作成することによって、修理プロセスを開始する。後になって、この開発技術者は、報告された欠陥に関した設計上の問題点を提起するメッセージを他の技術者たちに通知することによって、関係した会話を開始する。行動項目フォームから、技術者は、ソフトウェア開発システムからの関連した欠陥報告のコピー、その欠陥に関する他のメール・メッセージ、欠陥修理の進行状況等にアクセスすることができる。

(3) Strudelの概念モデル

ここでは、Strudelの抽象的モデルにおける基本的なタイプについて、(2)のシナリオ例に基づいて解説する。

会話とタスクは、それぞれメッセージと行動項目の集合である。メッセージは会話手番を伝えるために利用される。例えばRequest to repair a defectという手番などである。通常、ひとつの会話手番は次の会話手番を示唆し、そのメッセージを読む人によって実行される行動を提示する。例えばRequest to repair a defectという手番は、次の会話手番としてAgree to repair a defectという手番を示唆する。会話とタスクを開始し、展開し、終了する経路を導くために、会話タイプとタスク・タイプを定義することができる。例えば、欠陥修理の会話においては、Request to repair a defectという手番あるいは、Are you the right person to handle this defect?という質問を最初の手番として定義する。欠陥追跡のタスクは、製品開発システムなどの外部のツールからStrudelへ欠陥の通知が報告されたとき、あるいは、ユーザがRepair Defectという行動項目を作成したときに開始される。

「会話手番」、「行動項目」、「通知」等の基本タイプは、「タスク手番（task move）」というルート・タイプのサブクラスである。これらのタイプのインスタンスは、タイトル、オプションのフィールド、およびその手番の状況の中でユーザが適用できる行動を持っている。例えば、Repair Defect行動項目のインスタンスの場合には、REPAIR DEFECTというタイトルと、Defect Reports(欠陥内容)、Rapair Location（修復すべき箇所）、Status（修復状況）、Design Issues（修復に係わる問題点）、Clients waiting for repair（修復を待つクライアント）等のオプション、およびActions（取るべき行動一覧）を持っている。

通知は、ユーザがインスタンスを作成するのではなく、アプリケーションからの呼出しへの応答としてインスタンスができるという意味で、擬似的な行動項目として取扱われる。この点を除いては、通知は行動項目と同一の性質を持つ。例えば、ユーザに欠陥を教えるために、ソフトウェア開発ツールからの連絡によって通知インスタンスが作成される。

特定の手番タイプに対応した行動を定義した場合、これらの行動はボタンとして表示される。例えば、Repair Defect行動項目のActionsの中にはInitiate Repairボタンがある。手番に対して定義された行動は、例えば、行動項目や次の会話手番を作成するために利用することができる。フィールドの上で定義された行動は、例えば、カレンダやスケジュールの上である時間が空いているかどうかを確認するために利用できる。また、行動をユーザが定義し、特定のタスクのための操作をタスク・オブジェクトに施すことができる。例えば、技術上の変更指示を製品の設計履歴に記入するための行動を、生産管理システムおよび在庫管理システムへのインタフェースを呼ぶことによって、インプリメントすることができる。欠陥のあるソフトウェア・モジュールを検索する行動は、ソフトウェア保守ツールを呼ぶことによってインプリメントすることができる。手番が作成されているのか、あるいは、読ま

れているのかの違いによって、その手番のタイプ定義に指定された異なった行動を提示することができる。

会話手番は、半構造的であってよい。この点では、手番は「半構造化されたメッセージ」とよく似ている。Strudelにおいては、それぞれの手番がフィールドからなる構造を持つけれども、フィールドの内容は構造化されていないという意味において、手番は半構造化されたものであるといえる。ユーザは、フィールドに好きなだけの情報を書き込むことができるし、書かなくてもよい。また、フィールドの中の情報は特定のタイプである必要はない。会話手番が作成されたとき、それは電子メールのメッセージの中に挿入され、他のユーザに送られる。一つのメッセージにいくつかの手番を含めることができる。

メッセージをつなぎ合わせると、一つあるいは複数の会話となる。会話は、オープニングおよびそれに続くメッセージから構成される。オープニングは、参加者、最初のメッセージ等々を指定する。ユーザは一つの会話手番タイプを選択し、そのインスタンスを作ることによって、新しい会話を開始する。この選択は、トップレベル・メニューであるStart conversation、あるいは既存の行動項目あるいは通知の中でおこなわれる。その後、ユーザはメッセージを編集し送信する。受け取ったユーザは、そのメッセージへの応答としてのメッセージを送る。このようにして会話が進行する。ユーザは会話をコピーしたり仲間に入ったりすることができる。

会話手番のための特定化された行動によって、ユーザは、会話の中の以前の手番およびそれにリンクされた行動項目と通知へ遡ることができる。このように探索する能力があるので、会話の中の行動をネットワーク形状のグラフィックで表示することができる。このグラフィックでは、各手番は長方形の箱で示され、関係する手番は矢印で結ばれる。矢印が破線のリンクは、可能ではあったが実行されなかった手番を結んでいる。

タスクとは、タスクの最初の行動あるいは通知によって生成される、行動項目あるいは通知の集合である。会話も最初の会話手番から生成されるものであり、タスクは構造的には会話と類似している。会話は特別のタスクとして扱われる。会話タイプおよびタスク・タイプは、一組の初期手番タイプを指定する。会話において宣言された初期手番タイプおよびタスク・タイプのメニューが現在、新しい会話とタスクを開始するために利用されている。

行動項目、通知、メッセージ等多様なクラスのためのブラウザを含め、タスクと会話のいくつかのデフォルトの表示方法が提供されている。現在の電子メール・ブラウザに似た、単純なメッセージ・ブラウザによって、従来の電子メールのメッセージとStrudelのメッセージを表示することができる。カレンダとTo-Do listを表示するために、ブラウザは、ユーザが行動項目とメッセージを、時間の関係とキーワード・マッチに基づいてソートし、フィルタをかけ、グループにまとめることを許している。例えば、メッセージを送信時間でソートし、参加者によってグループ化するなどが可能である。会話

ブラウザによって、ユーザは、会話の内部を探索したり、会話を保存したりすることができる。

(4) メッセージの作成

ユーザが会話手番を含むメッセージを読むとき、彼は、前の手番のタイプと内容に基づいて、次の会話手番を作成することができる。ユーザは、現在の手番の中に示唆されたものの中から次の手番タイプを選択する場合もあるし、または、ライブラリに定義されたタイプから選択しても良い。あるいは、タイプの付いていない手番を含むメッセージを送っても良い。また、新しいタイプを作成して使用してもよい。会話タイプは、会話の進行中に拡張することができる。ユーザは既存の手番タイプの中に、next-conv-move-types として、新しい手番タイプを定義することができる。同様な方法によって、タスクのタイプを定義し進化させることができる。

Strudel の現在のインタフェースにおいては、次の手番タイプの候補は、ボタンあるいはメニューを介して提示される。到着したメッセージを読んでいるとき、ユーザはボタンを押し、次の手番タイプを選択する。メッセージの送り手は、次のデフォルト手番タイプを設定することができる。これは、方法や方針を表現する、望ましい選択枝を示唆するためでもあるし、あるいは、ただ談話におけるデフォルトの焦点を設定するためでもあろう。ユーザは次の手番を編集するか、選択したタイプのデフォルトの手番を送るかのどちらかを選択する。この次の手番は、現在の手番への応答（in response to）である。これは、従来の電子メールにおける in-reply-to によって作られる結び付きに類似している。しかし、in response to の手番は、誰に宛てて送っても良い。このようにして、その時点で他の人たちを会話に入れることができる。例えば、問題の記述を受け取ったとき、ユーザは異なった人々に異なったリクエスト・メッセージを送り、問題に対する異なった解決方法の提案を要求することができる。

(5) 問題点

従来のオフィス・フォーム・システムにおいては、フィールドとボタンの上のテキスト・ラベルは固定である。Strudel においては、ユーザはラベルの表面のテキストおよびデフォルトのフィールド内容を容易に変更することができる。例えば、表示された挨拶文章の礼儀正しさの度合いを変更することができる。この柔軟性から多くの問題が発生する。読む人は固定したフォームの方がすばやく理解できるといった利点は失われる。既存の電子メールと同じく、変更したというしるしがないので、送り手が行った小さな変更によって誤解が生じるかもしれない。

重要な問題は、会話管理システムと協調システムの間の相互運用可能性のためのフレーム・ワークの開発である。この問題に対する本研究の最初の実際的なアプローチは、従来からの「構造化されていない」電子メールおよび他のプロトタイプの会話管理、アクティブ・メッセージ、あるいは協調システムとの実験的な相互運用性を許すよう Strudel を設計することであった。このため、会話手番のタ

イプ、フィールドの追加、および行動を互いに独立に定義した。これによってStrudelは、これらの機能の一つ以上をサポートしている他のシステムからのメッセージを実験的に解釈することが可能となった。例えば、特定のアプリケーションは半構造化メッセージだけを送るかもしれない。あるいは、Requestのような非構造化のタイプされたメッセージだけを送って来るかもしれない。または、アクティブで構造化されたメッセージを送って来るかもしれない。キーとなる問題は、（メッセージの中の）行動に対して、相互運用的な参照を提供することである。

この他、サポートすべき適切な会話「単位」を発見すること、ユーザ主体の設計と技術主体の設計との間に適切なバランスを取ること、機械によるサポートに最も適した会話クラスの発見などの問題を考慮しなければならない。

8.10　協調支援環境構築のためのアーキテクチャ

8.10.1　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。

8.10.2 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。

——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

CSCW

JIPDEC