SEAMAIL

Newsletter from Software Engineers Association

Volume 10, Number 9-10 May, 1997

目　次

編集部から		1
ネットワークを介した共同作業の支援環境構築にむけて	落水 浩一郎	2
「オブジェクト指向技術の実態調査」結果報告書	中小路 久美代・松原 順	38
家庭内乱 ^H^H 内 LAN を張る	太田 俊哉浩	59
Software Symposium 97 参加案内		73
ISFST'97 論文募集		77

ソフトウェア技術者協会
Software Engineers Asociation

ソフトウェア技術者協会(SEA)は，ソフトウェアハウス，コンピュータメーカ，計算センタ，エンドユーザ，大学，研究所など，それぞれ異なった環境に置かれているソフトウェア技術者または研究者が，そうした社会組織の壁を越えて，各自の経験や技術を自由に交流しあうための「場」として，1985年12月に設立されました．

その主な活動は，機関誌SEAMAILの発行，支部および研究分科会の運営，セミナー／ワークショップ／シンポジウムなどのイベントの開催，および内外の関係諸団体との交流です．発足当初約200人にすぎなかった会員数もその後飛躍的に増加し，現在，北は北海道から南は沖縄まで，700余名を越えるメンバーを擁するにいたりました．法人賛助会員も30社を数えます．支部は，東京以外に，関西，横浜，長野，名古屋，九州，広島，東北の各地区で設立されており，その他の地域でも設立準備をしています．分科会は，東京，関西，名古屋で，それぞれいくつかが活動しており，その他の支部でも，月例会やフォーラムが定期的に開催されています．

「現在のソフトウェア界における最大の課題は，技術移転の促進である」といわれています．これまでわが国には，そのための適切な社会的メカニズムが欠けていたように思われます．SEAは，そうした欠落を補うべく，これからますます活発な活動を展開して行きたいと考えています．いままで日本にはなかったこの新しいプロフェッショナル・ソサイエティの発展のために，ぜひとも，あなたのお力を貸してください．

代表幹事： 山崎利治

常任幹事： 大場充　熊谷章　坂本啓司　中野秀男　深瀬弘恭

幹事： 青山幹雄　荒木啓二郎　市川寛　伊藤昌夫　菊地俊彰　君島浩　窪田芳夫　酒匂寛　塩谷和範
篠崎直二郎　杉田義明　高橋光裕　武田淳男　田中一夫　玉井哲雄　中来田秀樹　中谷多哉子
野中哲　野村行憲　野呂昌満　端山毅　平尾一浩　藤野誠治　二木厚吉　堀江進　松原友夫

事務局長： 岸田孝一

会計監事： 辻淳二　吉村成弘

分科会世話人 環境分科会(SIGENV)：塩谷和範　田中慎一郎　渡邊雄一
教育分科会(SIGEDU)：君島浩　篠崎直二郎　杉田義明　中園順三
ネットワーク分科会(SIGNET)：小林俊明　人見庸　松本理恵
プロセス分科会(SEA-SPIN))：青山幹雄　伊藤昌夫　坂本啓司　高橋光裕　田中一夫　増井和也

支部世話人 関西支部：臼井義美　中野秀男　盛田政敏　横山博司
横浜支部：野中哲　藤野晃延　北條正顕
長野支部：市川寛　小林俊明　佐藤千明
名古屋支部：筏井美枝子　角谷裕司　外山徹　野呂昌満
九州支部：武田淳男　平尾一浩
広島支部：大場充　佐藤康臣　谷純一郎
東北支部：菊地俊彰　野村行憲　和田勇

賛助会員会社： アイシーエス　PFU　SRA　アスキー
オムロンソフトウェア　カシオ計算機　キヤノン新川崎事業所
さくらケーシーエス　サン・ビルド印刷　ジェーエムエーシステムズ　ジャストシステム
ダイキン工業　安川電機　構造計画研究所　新日鉄情報通信システム
新日本製鉄エレクトロニクス研究所　中央システム
東芝アドバンストシステム　東電ソフトウェア　東北コンピュータ・サービス
SRA東北　日本NCD　日本情報システムサービス
日本電気ソフトウェア　富士通　富士通エフ・アイ・ピー
オムロン　SRA中国　富士電機　ブラザー工業　（以上30社）

SEAMAIL Vol. 10, No. 9-10　　1997年5月15日発行
編集人　岸田孝一
発行人　ソフトウェア技術者協会（SEA）
〒160　東京都新宿区四谷3－12　丸正ビル5F
T: 03-3356-1077　F: 03-3356-1072　sea@@sea.or.jp
印刷所　サンビルド印刷株式会社　〒162　東京都新宿区築地町8番地
定価　1，000円（禁無断転載）

編集部から

☆

およそ半年ぶりのSEAMAILです．

☆☆

イタリアから帰国されたばかりの落水先生からは，御自身の研究室でのこれからの研究開発計画についての，力のこもった説明をお寄せいただきました．読者各位のコメントをお待ちします．

☆☆☆

中小路さん·松原さんの調査報告は，函館のソフトウェア·シンポジウム'94の参加者を対象に行なわれた調査がきっかけで，もともとはアメリカやヨーロッパでも同じようなアンケート調査を行なって比較分析する予定だったのですが，御承知の通り，外国ではこの手の調査をやっても回答がなかなか集まりません．ということで，昨年日本で２度目の調査を行なって，時系列的な分析をまとめることになったものです．

☆☆☆☆

太田さんからは，マイホームへのLAN敷設の貴重な体験談がお寄せいただきました．ありがとうございました．

☆☆☆☆☆

さて，今後の発行予定ですが，編集部サイドでは，まず，昨年２度にわたって行なわれたMCCとのJoint Forumの記録とりまとめがあります．これは今月ICSE (Boston) に出かけるまでに終って印刷にまわせるでしょう．もう一つは，昨年暮れに行なわれた国際シンポジウムの記録 (OHP原稿etc) のとりまとめ．６月までにはなんとかしなければ! それと並行して，１月の若手ワークショップのレポートも上がってくると思います．

☆☆☆☆☆☆

ということで，ここ何カ月かはなんとか定期刊行が期待できそうな気がしますが.....?

☆☆☆☆☆☆☆

ネットワークを介した共同作業の支援環境構築にむけて

落水 浩一郎
北陸先端科学技術大学院大学
情報科学研究科
〒923-12 石川県能美郡辰口町旭台 1-1
Phone: +81-761-51-1260
Fax: +81-761-51-1149
e-mail: ochimizu@jaist.ac.jp

概要

本論文では、ソフトウェア・プロセス実行の支援とコミュニケーションの支援を統合した機構を提供することで、ネットワークを介した共同作業を円滑に支援する計算機支援環境の参照モデルを提案する。環境開発の目標は、開発方法論の明解性、個人の作業環境の独立性、コミュニケーションの円滑性、作業場所の任意性 (計算環境の携帯性)、作業遂行に必要な知識・ツールの獲得容易性 (遍在情報へのアクセス容易性) の五つの特性で特徴づけられる「快適性の向上」である。

環境の構成は以下の通りである。共同作業を通じて生成・変更されていく「中間成果物や決定事項とそれ等の状態」を、現実世界の状況を正確に反映しつつ管理する情報リポジトリを中心に置く。次に、そのまわりに共同作業者間の協調や調整を可能にする目的で、中間生成物や決定事項の矛盾に満ちた状態を把握し、それを減少させるための機能を配置する。そのような機能の支援の下に、チーム構成員の分散度や作業の規模等のパラメータに応じてＣＡＳＥツールやグループウェアを配置する機構を置き、さらに、上記環境の種々の内部状態を視覚的に把握することを支援するユーザインタフェースを置く。

本報告で提案する「自在」CSCSD モデルは、上記環境の構成に対するフレームワークであり、また、そのような環境を容易に構築するためのオブジェクト群を整備していくための手掛りでもある。

以下、１章では、「自在」プロジェクトの目的と目標を明らかにする。２章では、モデルを定義し、環境を構築するための原則と考え方を明確にするために、ソフトウェア設計方法論、CSCW、ソフトウェア・プロセス・モデルに関する現状の問題点と今後の課題を検討する。３章では、「自在」CSCSD モデルの構成と詳細を説明する。４章では、「自在」CSCSD モデルに従って開発中の自在プロトタイプの進捗状況を紹介する。５章で今後の課題をまとめる。

目次

1 「自在」プロジェクトの目的

2 ソフトウェア設計方法論、CSCW、ソフトウェア・プロセスの研究分野において解決されるべきいくつかの課題
- 2.1 ソフトウェア設計方法論におけるソフトウェア・アーキテクチャの役割
- 2.2 コミュニケーションはいつ、どのような条件下で成立し、かつ進行するのか？
- 2.3 ソフトウェア・プロセス・モデルでは何を記述すべきか？
- 2.4 2章のまとめ

3 CSCSD モデル
- 3.1 決定事項の管理に基づくプロセスとコミュニケーション支援の統合
  - 3.1.1 決定とは
  - 3.1.2 会話の何を記録すべきか
  - 3.1.3 中間生成物間の粗粒度レベルの依存関係を統合の土台とすることについて
  - 3.1.4 粗粒度レベルの依存関係を基にしたタスクとソフトウェア・プロセスの定義
  - 3.1.5 ソフトウェア・プロセスの実行と発生するコミュニケーションの関係について
- 3.2 情報リポジトリへの要請
  - 3.2.1 各自の責任範囲をタスクとして定義することについて
  - 3.2.2 粒度の異なる決定事項を系統的に管理することについて
  - 3.2.3 討議空間
  - 3.2.4 討議プロセス
  - 3.2.5 討議の型
  - 3.2.6 グループウェアベース
- 3.3 共同作業のモデル：CSCSD モデル

4 自在プロトタイプの研究現状
- 4.1 携帯性向上のための CSCSD サーバ
- 4.2 独立性を保証するための分散サーバ
- 4.3 円滑性を提供するためのグループウェアベース
- 4.4 探求的学習を支援するアクティブチャネル
- 4.5 評価のためのプロトコル解析実験
  - 4.5.1 分散に依存する問題とその評価尺度の発見
  - 4.5.2 快適性を特徴づける特性とその評価尺度の発見
- 4.6 3章のまとめ

5 まとめと今後の課題
- 5.1 一貫性管理層を追加した CSCSD モデルによる調整支援
- 5.2 オブジェクト管理システムの構成法
- 5.3 分散に依存する問題のより精密な把握
- 5.4 分散開発時代に適合する方法論とプロセスモデルの定義

1 「自在」プロジェクトの目的

ソフトウェア諸技術を、明解な構成原理を持つ建築や音楽のように、より健全で我々の感動を呼ぶものにするためには、我々は今、何をするべきであろうか？ まず、小島氏によるイタリアの建築に関する随筆[29]を紹介しよう。彼の建築論に関する考察は、ソフトウェア環境論の考察にあたって、類推により得られるいくつかの有用な視点を我々に示唆してくれるだろう。

- 建築論とは、建築によって何を表現するか？、よい建築とは何か？、そしていかによい建築を作るか？である。古代ローマ帝国における建築が表現し続けていたのは「力」であった。「力」は物量すなわち「大きさ」によって表現された。物理的な大きさを建築の「大きさ」として感じさせるため、形体は単純化された。古代ローマの建築は幅、奥行、高さなどの比率が注意深く検討された直方体と円筒を基本にしている。それを可能した技術はセメントであった。ルネサンスの建築に求められたものは「調和」であった。「調和」とは空間全体が整理され、秩序が感じられる状態といってよい。この「調和」は「比例」によってつくりだされた。建築の各部分の寸法がそれぞれ比例によって整えられ、それによって空間の秩序がつくられた。初期のルネサンス建築では最も単純な比例、*1*：*1*が基本となった。具体的には平面なら正方形と円、立体では立方体と球の形をとる

類推により、我々はソフトウェア開発環境によって何を表現するか？、よいソフトウェア開発環境とは何か？、そしていかにしてよいソフトウェア開発環境を作ることができるか？ という問いかけを設定することができる。何を表現すべきかを明らかにすることは、時代の共感を呼び次の時代に生き残るソフトウェア開発環境を創出するために必要である。ソフトウェア工学の世界では、速さ、堅牢さ、柔軟性などの目標が検討され続けてきた。著者は、次の時代に重要となる一つの目標は「快適性の向上」であると考える。

目標達成のための技術課題を設定するためには、快適なソフトウェア開発環境が持つべき特性を論じることが必要である。著者は、少なくとも明解性、独立性、円滑性、携帯性、増幅性の五つの要因が必要であると考えている[50]。

明解性 とは、「設計対象が評価可能であり、開発手順が明確かつ具体的であること」

独立性 とは、「必要なコミュニケーションを保証しつつも、個人の作業の独立性が保持できること」

円滑性 とは、「チームの構成員間のコミュニケーションが円滑であること」

携帯性 とは、「活動の場から、必要な情報に迅速にアクセスできること、さらには活動の場にそれを支える情報が追随できること」

増幅性 とは、「ネットワーク上に遍在する、作業遂行に必要な情報にタイムリーにアクセスできること」

をそれぞれ意味する。

2 ソフトウェア設計方法論、CSCW、ソフトウェア・プロセスの研究分野において解決されるべきいくつかの課題

本章では明解性、円滑性、独立性が快適性の重要な要因であると考えるにいたった理由を述べる。携帯性と増幅性については第4章で述べる。

2.1 ソフトウェア設計方法論におけるソフトウェア・アーキテクチャの役割

小島氏の随筆に戻ろう。

- 建築の様式とは、「空間の組立て方」と「特徴的な細部」とからなる。細部の少ない建築は様式の変革期に現われ、空間の組み立て方を中心とした新しい様式が生みだされた。一方、変革のない時代には、多様な細部が発展した。一方、変革を推し進めなかった建築もあった。ビザンティン、ゴシックなどに分類される中世の建築やバロックの建築は建築空間に支配されない多様な細部を持っている。

建築における「空間の構成法」は、ソフトウェア設計においては「ソフトウェア・アーキテクチャ」に対応する。著者はソフトウェア・アーキテクチャに関する研究を充実させることが、開発対象および設計方法論の明確化に最も貢献すると考えている。ここで、ソフトウェア・アーキテクチャという用語を吟味してみよう。

ソフトウェア・アーキテクチャという用語の定義は人により様々である。 CMU の David Garlan は、彼の論文中で [17]、大規模システムの設計においてクリティカルな問題、計算要素とそれら要素間の相互作用を高レベルの抽象度で構成すること を論じた。彼は、ＳＥＩの研究グループによる定義に基づき、ソフトウェア・アーキテクチャの定義を以下のように与えている。「プログラムやシステムの構成要素の構造、構成要素間の相互関係、設計やその変更を支配する原理やガイドライン」。

彼は、ソフトウェア・アーキテクチャに関する研究動向を以下の二つに分類した。

1. 複雑なソフトウェア・システムを構造化するための手段、技法、パターン、いいまわし (idioms) などの、設計者間で共有される様々な共有レパートリーを開発する。これらの用語はシステムの厳密な定義を与えることはほとんどできないが、システムの全体像を理解可能にする抽象を用いて、複雑なシステムを設計者たちが記述するのを可能にする。

2. 一群の製品に対して再利用可能なフレームワークを提供する特別なドメインを開拓する。その根拠は、「共有されるデザインを実体化することで、互いに関連する一群のシステムの共通部分を、比較的低コストで構築できる」という考え方にある。代表的な例としては、コンパイラの標準的分解、標準化された通信規約、第 4 世代言語における共通パターンなどがあげられる。

彼はまた、ソフトウェア・アーキテクチャに関する研究の関心を以下のように整理した。

- 全体的構造と大域的制御構造
- 通信プロトコル、同期、データ・アクセス
- 設計要素への機能の割り当て
- 物理的分散
- 設計要素の構成法
- 規模への対応と性能
- 設計代替案の選択

おおざっぱな言い方をすれば、設計方法論およびソフトウェア・アーキテクチャ共、その目的は、要求と実現の間のギャップを埋めることにある。違いは以下の点にある。ソフトウェア設計方法論はある問題クラスをその解に変換するステップを定義する。ソフトウェア・アーキテクチャは要求と実現の間に独立して存在し、D.Garlan によって与えられた論点を検討し、可能なアーキテクチャのクラスから最適なものを選択することを可能にする。設計方法論とソフトウェア・アーキテクチャは互いに補完的な役割を果す。

筆者等は、設計方法論の効果と問題点を認識するために、SA/SD や OMT などの既存のソフトウェア設計方法論を実世界における大規模モデリング問題や設計問題に適用した経験を持つ [31]、[32]、[43]。しかし、ほとんどの既存の設計方法論は、ソフトウェア・アーキテクチャ設計フェーズを明解かつ具体的に定義していないので、以下のような評価を既存の方法論に与えた。

- 既存の設計方法論は、問題依存の解を類形化された部品 (例えば、オブジェクト・ライブラリ) に接続する役割を果たすにすぎない

次に、ソフトウェア・アーキテクチャの柔軟性の問題を論じよう。オペレーティング・システムやオンライン・リアルタイム・システムなどの様々なモニュメントが構築された時代における価値感は、「大きさ」にあった。ソフトウェア工学は「大きさ」と「複雑さ」を制御するために生まれた。そのための原理としては「分割統治」や「機能の階層構成」が採用された。階層は「抽象のレベル分け」によって達成される。階層構成においては、ある特定の層の実装法を効率改善のために変更することは容易である。しかし、ソフトウェアにおける抽象は「最適化」を伴なうため、「階層構造自体の再構成」は困難である [44]。細部の手直しの積み重ねによっては「空間の再構成」はなしがたい。

その反省を受け、「類形化」を目標とするオブジェクト指向技術が試みられている。その構築原理は「型」の概念に基づく。デザインパターンに関する研究の成果を待った上で結論を下す必要があるが、オブジェクト指向方法論においてはシステムの構築がボトムアップになり空間の構成が困難である。さらに、ネットワーク時代においては、計算要素間の相互通信からなるネットワークにまたがる計算構造の柔軟性を保証する、新しい目標と構成原理が必要である。

次の時代における「計算」が以下に述べる形態をとることに関しては、我々は合意できると思う。

- 計算要素はオブジェクトにより表現され、その間の相互作用はメッセージ通信によってなされる (全体的構造と大域的制御構造)。制御は分散制御になり、制御およびデータの授受に関する通信プロトコルは標準化される。永続オブジェクトの管理は一貫性保持の目的から多分集中管理されるだろう。CSCW アプリケーションの発展により、アプリケーションオブジェクトと永続オブジェクト間の交信はロングトランザクションが主流となるだろう [9]。ネットワークの遅延を考慮に入れた書き込み制御技術も発展するだろう [10]。

上記の諸機能を実現するための基礎 /応用研究はすでに、世界の各地で実施されている。これらの研究があるレベルにまで達した時、次に問題になるのは何であろうか？

とくに、ネットワークを介した共同作業を支援する分野に話題を絞る時、重要な問題の一つとして「規模への対応 (アーキテクチャを規模対応に整備すること)」が挙げられる。アーキテクチャの設計において、「規模」を特徴づける主要なパラメータは何であろうか？ 「遅延時間」は重要なパラメータの一つである。「遅延時間」はネットワークの規模により大きく異なり、システムの性能に大きな影響を与える。

「規模」に応じたソフトウェア・アーキテクチャのクラス分けをまず実施し、次に、設計者にシステム構築の枠組と部品を提供するために、全体的構造と大域的制御構造、通信プロトコル、同期、データ・アクセスなどの諸機能を支援する制御オブジェクトを整備・提供する必要がある。この時、分散度を考慮しつつ、アプリケーション・ドメインのオブジェクトを、物理的配置に割り当てる設計方法論および記述言語を開発する必要がある。

2.2 コミュニケーションはいつ、どのような条件下で成立し、かつ進行するのか？

グループウェアは以下のような技術を開発することにより、コミュニケーションにおける時間／空間分散を克服する手段を我々に提供してきた [11]。すなわち、リアルタイム・グループウェアによる即時性、臨場感の向上。マルチメディアによる適切な情報表現手段の提供。人間間の face-to-face コミュニケーションの前段階におけるコーディネーション・コストの e-mail による低減 [13]。各自の仕事の独立性を保証しつつ必要なコミュニケーションを達成させる非同期コミュニケーション (電子メール) の効果も広く世の中に受け入れられている。

しかし、これらの効果だけでは膨大なネットワーク設備に対する社会的コスト投入の見返りが少なすぎる。コミュニケーションにおける「知性増幅」の道具として、計算機とネットワークを最大限に活用するアプリケーション分野を、我々はさらに開拓する必要がある。

新しい領域を開拓する一つの手掛りは、「会話の背景状況」を明示的に取り扱い、かつ関係者間で共有させることにある。例えばソフトウェ開発においては、ほとんどの会話は膨大な量の文書や設計者やプログラマの知識の一部分に関連しており、それ等の状態に基づいて様々な活動や会話が駆動される。既存のグループウェアは、頼む、伝える、説明するなどの一つ一つの会話を支援することはできるが、その背景情報や状態に関しては、例えば文書を例にとると、今、我々ができることは、関連する文書をスクリーンに表示し操作する程度のことである。ソフトウェア文書の量が膨大であり、記述内容の相互関係も複雑であることを考えにいれると、既存のグループウェアの能力は必ずしも十分でないように思える。大量の情報とそれに付随する状況を参加者間で共有でき、かつフォアグランドで取り扱うことができるような機能をグループウェアは充実すべきであろう。

以上述べた問題点を明確に指摘した論文がある。C. Ellis は [12]、ソフトウェア・モデリングに関する研究成果に基づいて、グループウェアに対する三つのユーザ視点モデルを定義・検討することにより、上記の問題点に関する今後の研究方向を示唆した。それらは、ユーザが操作可能なオブジェクトの静的記述であるオントロジカルモデル (Ontological Model)、ユーザによって実行されるべき活動を構造的に表現した調整モデル (Coordination Model)、システムとユーザおよびユーザ間のインタフェースであるユーザインタフェースモデルである。

オントロジカルモデルによって、会話の背景情報の内、静的情報を保持するリポジトリを定義する。一般に誰も全体のことは知らない。多くの場合、各会話参加者はリポジトリの内容の一部を知っており、さらに、それらの情報の一部は他人と共有されている。誰かが各自が保持する情報内容の理解に差があることを認知した時に、コミュニケーションの契機が成立し、情報の均一化の試み (会話) が始まる。会話の内容は、各参加者が持つコンテクストに依存して理解される。構造的コンテキスト (structual context) は「リポジトリの動的状態」を表現する。社会的コンテキスト (social context) や組織コンテキスト (organizational context) は、「会議参加者の動的心理状態と社会的背景情報」をそれぞれ表現する。コンテクストにより会話の流れが制御される。話合いのルールや常識的な手順 (プロセス) を調整モデルが表現する。ユーザインタフェースモデルが、関連資料、会話の内容、会議の進行状態などを視覚的手段で会話参加者に提供する。

ここで、考察の範囲を「会話に通じて下される決定事項」に限定しよう。例えば、ソフトウェア開発過程は「決定の積み重ね」であるとみなしうる。各自は各自の仕事の責任範囲において、みずからが下した決定を管理している。例えば、ソフトウェア設計者は、設計対象に関する仕様を作成しプログラマに引き渡す。プログラマはその仕様に従ってデータ構造や制御構造を設計しコードを書く。決定 (引き渡される仕様) は、引き渡された時点から、設計者とプログラマの間で共有されることになる。引き渡されたものの内容の理解に関してギャップが存在する時に、いろいろな種類の会話が発生する。また、プロジェクトマネージャとデザイナやプログラマ達と、主に時間資源の管理に関する決定について様々な話題が生じる。

この時、会話の流れを円滑にするためには、以下ような情報が必要になる。

- 決定事項の全体像の表現
- プロジェクトの目標に対する進捗状況
- ある話題に対する、部分的決定の積み重ねの構造とそれらの決定の際に存在したさまざまの前提条件
- 一つの部分的決定に関する確信度と手間

これらの要請は、相互理解や決定の変更管理支援のために、決定事項とその状態を、その全体像から詳細レベルにいたる決定の粒度で、系統的に管理できるソフトウェア・リポジトリの必要性を示唆している。リポジトリのモデリングにおいては、決定の時間的特性を考慮に入れるべきである。多くの場合、決定は一時的なものである。時間がたち、状況が変化するにつれて、合意が不合意になり、不合意が合意に変化することはしばしば経験されることである。

決定事項の管理技術を現在のグループウェアの成果に併合することにより、長期間にわたる会話を円滑にし、参加者の間に一体感を作りだす効果を期待できる。

2.3 ソフトウェア・プロセス・モデルでは何を記述すべきか？

ソフトウェア・プロセスに関する研究分野においては、「プロセス記述言語とその実行環境」および「プロセス成熟度」に関する二つの主要な研究成果がある。前者は開発工程設計の道具を与え、後者は、組織の開発能力の判定基準、開発プロセス改善へのガイドラインを与える [45]。本節では、ソフトウェア・プロセス・モデリングの必要性、有効性に関する筆者の意見をもとに、今後の研究方向を論じる。

ソフトウェア・プロセス記述のねらいの一つは、ソフトウェア設計方法論の形式化にある。これは成功するであろうか？ この議論を進めるにあたり、ソフトウェア設計方法論そのものの有効性と、ソフトウェア設計方法論の形式記述の有効性の問題は注意深く区別する必要がある。ほとんどのソフトウェア設計方法論はソフトウェア開発にあたってのガイドラインを与えているにすぎないので、ソフトウェア・プロセス・モデル (ソフトウェア設計方法論の形式記述) も当然同じ欠点を持つ。ここで我々が議論したいことは、形式的記述そのものの有効性である。

本題に戻ろう。「ソフトウェア設計方法論の形式化が有効であるか？」という問いは、「手続きの抽象は有効であるか？」という問いと同じ意味を持つ。ソフトウェア・プロセス・モデルは、(操作、入力、出力) の 3 組で表現される「活動」の集合と、それらを構造化する制御構造からなる。(操作 1、入力 1、出力 1) と (操作 2、入力 2、出力 2) に、「型」と「ポリモルフィズム」の概念を適用することにより、抽象表現 (操作、入力、出力) を得ることができる。オブジェクト指向法では、さらに、(操作-x, di, do) と (操作-y, di, do) を ([操作-x, 操作-y], di, do) としてまとめることにより一つの表現単位を得る。しかし、これらはプログラミング言語分野での成果であり、ソフトウェア・プロセス・モデリングに固有のものではない。制御構造に関しては、ソフトウェア・プロセス・モデリングに特有な特徴をいくつか観測できる。その一つに「反復」または「バックトラッキング」がある。ソフトウェア・プロセスにおいては、反復における戻り先は、その非決定性の特徴よりあらかじめ決定することはできない。戻り先は動的に決定される [42]、[28]。

プロセスの実行時変更もソフトウェア・プロセス・モデリングに特有の性質である。人間の行動は不確実で再現性がないので、プロセス実行時にその実行順序や実行内容が状況に応じてしばしば変更される。すなはち、「非決定性」や「不確定さ」の記述が、プロセス・モデリングに固有の特徴である [25]。

これらの制御構造を記述する見返りとして我々は何を期待できるのであろうか？この点について、はっきりした見通しを我々は持たないように思える。例えば、ウオータフォール・モデルにおいてもオブジェクト指向開発法においても、「反復」は必ず出現する。繰り返しは、前者では悪であり、後者では善である違いがあるだけにすぎない。ソフトウェア・プロセスを形式的に記述することは、ウォータフォール・モデルの欠点を定量的に評価することを可能にするのだろうか？ オブジェクト指向ソフトウェア開発におけるプロトタイピング・アプローチに対して、有用で具体的なガイドラインを提供できるのだろうか？ 柔軟な標準開発工程の設計を助けてくれるのであろうか？ そのような効果を期待できないとすると、形式化は、実世界の活動を別の世界の言葉で単に表現しなおしているにすぎない。

問題の起源に立ち帰る必要がある。人間は完全なものを一度には完成できないので、繰り返しが発生する。その度合は、ソフトウェアの規模が増大するにつれて大きくなる。また、人間は機械ではないので、その振舞いを予測し再現するのが困難である。ソフトウェア・プロセス・モデリングにおいては、これらの人間要因を考慮に入れる必要がある。

我々のアプローチでは、手順を書き下すことではなく、仕事の静的/動的コンテキストを定義することに力点を置く。試行錯誤は自然なものであるとして、達成目標 (品質基準)、犯してはならない制約や規則、他人との仕事のつながりとその優先度などを各自の作業環境のまわりに配置する。品質基準はオブジェクトベース中のオブジェクトの属性とする。制約、規則、他人との仕事の関係は作業インタフェースの属性とする。上記の事柄に違反した際の調整支援は今後の重要な研究課題である。

プロセス成熟度に関する関する研究は、実世界における様々な開発プロセスを改善するための有用なガイドラインを提供しうるだろうか？ F.Cattaneo は論文 [5] で、CMMの適用が必ずしも現実の改善にはつながりないことを指摘している。何が原因なのであろうか？

いくつかの可能性のうち、ここでは、問題発生源の一つとして、個人の活動とチームの目標の間の「調

整」の問題を吟味してみよう。例えば、他人のやった仕事を引き継ぐ。他人に仕事を引き渡す。社内やプロジェクト固有の規則を順守する。一定の品質が保証されたプロダクトを生成するなどの例がある。最も重要なことは、これらすべてのことを納期にあわせて、限られた時間資源で実行しなければならないことである。

個人の目標をプロジェクトの目標の間には大きなひらきがある。前者は、「自身の能力の範囲内で納得しながら良い仕事を楽にする」という自己中心の仕事のタイプを容認し、後者は、「限られた時間範囲内にすべての仕事を終了する」という自己犠牲、献身型の仕事のタイプを要求する。双方を同時に達成できる人もいるがそれは稀な例である。われわれは、そのようなノウハウの共有を可能にする理論とシステムを、事例推論とモデル推論を利用して開発したことがある [46]、[55]。しかし、膨大なコストを必要とするので現在のところ実用化の見通しはない。個人の目標とプロジェクト・チームの目標をつなぐことができる調整支援のためのプロセス・モデルが必要である。

2.4 2章のまとめ

本章では、ソフトウェア・エンジニアリング活動を対象として、計算機ネットワークを介した共同作業支援のためのモデルを定義し、環境を構築するにあたっての基本方針と原理を述べた。我々の目標は、以下の五つの特性で特徴づけられる「環境の快適性の向上」にある。

明解性: 中間成果物の評価が可能であり、その手順が明確かつ具体的である設計方法論を利用できること

独立性: 必要なコミュニケーションを保証しつつも、個人の作業の独立性が保持できること

円滑性: チームの構成員間のコミュニケーションが円滑であること

携帯性: 活動の場から、必要な情報に迅速にアクセスできること、さらには活動の場にそれを支える情報が追随できること

増幅性: ネットワーク上に遍在する作業遂行に必要な情報・ツールに要求に応じてタイムリーにアクセスできること。すなわち、個人の能力を拡大し得るようなツールが必要であり、それを開発できる環境が整いつつある。

次に、今後解決されるべき研究課題を洗いだす目的で、ソフトウェア設計方法論、CSCW, ソフトウェア・プロセス・モデリングの分野における研究現状を吟味した。その結果、以下のような課題を認識した。

- ネットワークにまたがる計算構造を前提として、規模に応じたソフトウェア・アーキテクチャ・クラスを整備すること
- 長期間にわたる会話を円滑にし一体感を生みだすために、リポジトリ管理技術 (特に決定事項の管理技術) を既存のグループウェアの研究成果に融合させること。この際、決定事項に関する状態 (確信度、完遂度など) を合せて管理することが必要である。
- 個人の目標とプロジェクト・チームの目標をつなぐことができる調整支援を目的としたプロセス・モデルを開発すること
- 予測不可能性、非再現性、不完全性などの人間要因を上記のモデルや環境の開発にあたって十分に考慮すること

3 CSCSD モデル

我々の共同活動の主要な要素は何であろうか？ それらはどのように関連して共同活動を成立させているのだろうか？ 少なくとも、プロダクト、コミュニケーション、プロセス、人間の四つを挙げることができる。本節における考察の目標は、これらそれぞれの要素が共同作業で果たす役割を吟味し、要素間の相互関係を定義することである。考察の結果として、CSCSD モデルと名付けた共同作業の参照モデルを提案する。

著者はすでに、ソフトウェア・エンジニアリング諸活動を支援するためにソフトウェア・リポジトリで管理されるべき情報を検討し、(コミュニケーション支援、プロセス管理支援、プロダクト管理支援)x(個人レベル、チームレベル、プロジェクトレベル) の 9 つの分類した [47]。本論文では、それに加えて、第 2 章で議論した以下の要因を加味した考察を行なう。

- 一体感を生みだし、かみくだく過程を支援することの必要性
- その振舞いは予測しにくく、完全なものを一度には完成できず、その過程には、一般に再現性がないという人間の能力の限界への配慮
- 個人 (自身の能力の範囲内で仕事を進める) とプロジェクト (限られた期限内に成果物を達成する) 間の目標のひらき

これらの要請を満たす筋道の通った解が存在するのであろうか？ もし存在すれば、それが CSCSD モデル構築の基礎を与える。その解の実行にあたって、計算機の特徴を活用できる場面は何であろうか？ これは「自在」環境設計の基礎を与える。

本節では、まず、分散計算システム上に共同作業支援環境を構築するにあたっての一つの切り口を示すことから始め、決定事項の管理を基にしたソフトウェア・プロセス実行支援とコミュニケーション支援の統合法を検討しつつ、上記問題解決への一アプローチをまとめる。その考察結果を基にして、共同作業支援の参照モデルである「CSCSD モデル」(Computer Supported Cooperative Software Development model) を提案する。

さて、技術的、経済的に以下のような利点を持つ分散計算システムは、今後ますます普及していくものと思われる [1]：

- 分散アーキテクチャは、多くのアプリケーションとその利用者に固有の、地理的分散をより良く反映する
- 複雑な問題を、複数の計算機上で並列に解くことで性能向上が達成された
- 独立なジョブを異なる計算機上で並行実行することにより、スループッツが改善された
- 情報、プログラム、プロセッサ等の共有により、資源効用が増大した
- データ資源やプログラム資源を複写することにより、可用性が増大した
- プロセッサや記憶装置を、増大する負荷に応じて、増強することが可能になった
- 冗長な計算資源により信頼性が増大した

他のエンジニアリング分野と同様に、分散システム設計における中心的課題はコストに対して利益をバランスさせることである。単一の計算機設計に比べて、分散アーキテクチャは並行性、可用性、故障許容性等において利益を与えるが、これらの利益はアーキテクチャの複雑さや性能オーバヘッドの増大の代償の下に達成されるものである（図 1下部参照)。

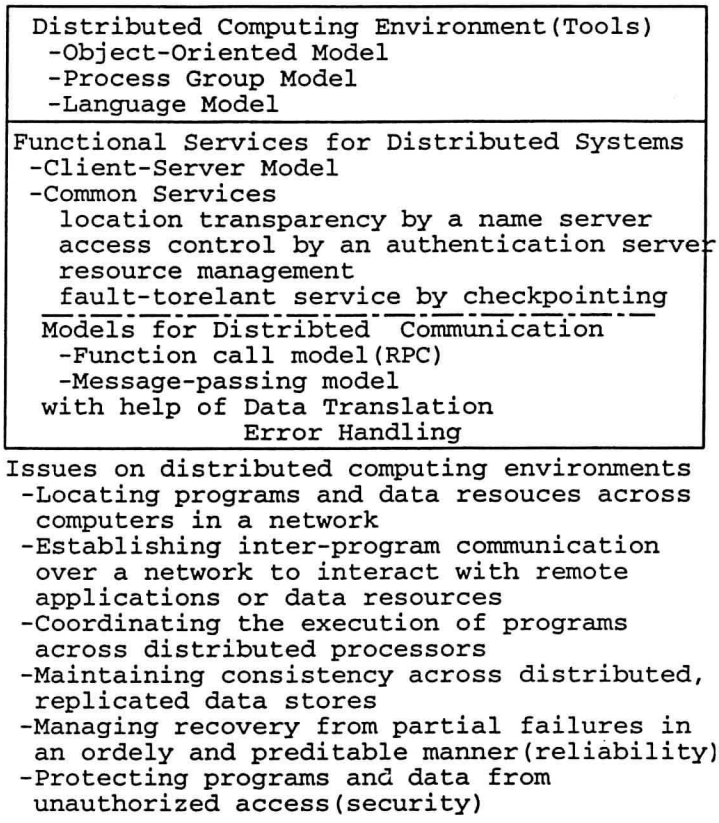


図 1: 分散計算環境における課題と研究成果

ところで、資源共有、一貫性保持、安全性、耐故障性、負荷バランス等の問題を理想的に解決する分散仮想計算機構が実現されたと仮定して、人間間の協調作業の実現にあたってそれでも残る「分散に依存する問題」は何であろうか?その一つとして、「協調活動の状態を共有することの困難さ」が挙げられる (図 2)。

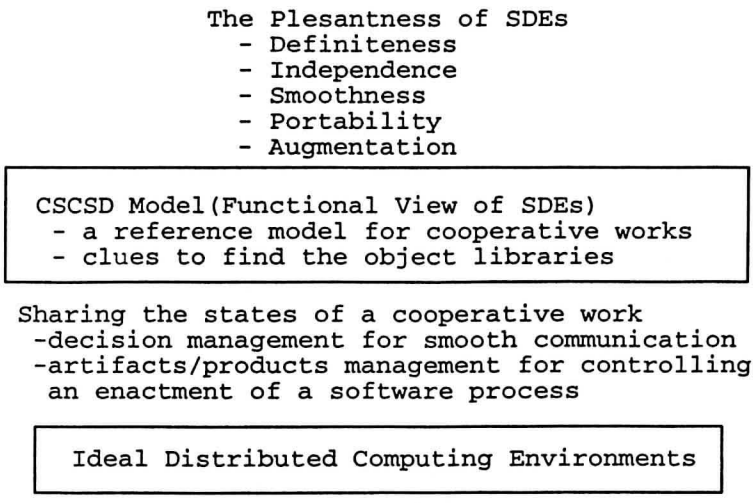


図 2: CSCSD モデルの役割

例えば、決定事項とその状態の共有はコミュニケーションを円滑にするために重要であり、また中間生

成物とその状態の管理と関係者間での共有は作業の進行を適切に制御するために必要である。ＣＳＣＳＤモデルの狙いは、明解性、独立性、円滑性、携帯性、増幅性の諸要因を支援するための情報リポジトリの構成とそれを利用した調整機能を、協調活動に関する状態の共有に基づいて構築することにある。この際、人間要因をモデル化の直接の対象とはしない。ここまで述べてきた協調を阻害する、様々な人間要因に基づく混乱を低減または緩和させ得るものを土台に置く。すなわち、共同作業によって生成される「もの」(中間生成物や決定事項など) を基にしてモデル化を試みる。

3.1 決定事項の管理に基づくプロセスとコミュニケーション支援の統合

ソフトウェア・プロセスの実行時に下される様々な決定と、プロダクトやコミュニケーションとの関係を明らかにしつつ、決定事項の記録・管理法について考察する。

3.1.1 決定とは

ソフトウェア開発における決定事項は 2 種類ある。一つは、設計者やプログラマによって作成され、検査チームによって検証される中間生成物や製品である。もう一つは、そのような中間生成物や製品を作成していく過程でなされる決定、すなはちいくつかの代替案の中からの一つの案を選択する過程でなされる決定である。前者は、後者が複数個集まって構造化されたものである。後者の型の決定 (デザイン・ラッショナレ) は伝達コストの安い会話によってなされる傾向がある。

共同作業においては、参加者各自が、現在の状況が全体目標に対してどのような位置づけにあるのかを理解できることが重要である。また、「完全なものを一度には作成できない」という人間の特性に起因して、決定事項はしばしば変更される。そこで、決定は多くの場合一時的であり、「完全な決定」はありえない。決定の変更とその波及の解析が容易に行なえる記録スキーマが必要である。

会話によってなされる決定事項を、中間生成物とその依存関係に統合する形で記録・管理することが必要である。両者の型の決定をどのように記録管理すべきであろうか？ これが CSCSD モデル定義にあたっての主要な論点となる。

3.1.2 会話の何を記録すべきか

ほとんどの研究者は、デザイン・ラッショナレを管理するのに、会話内容そのものをその流れに沿って記録すれば良い、または、すべての会話内容を構造的に記録すれば良いという立場を取る [6]。しかし、これは冗長である。なぜならば、このような会話は、内容をかみくだいて説明したり (パラフレージング)、正確に内容を理解してもらうための質疑応答を数多く含むからである。まず、コミュニケーションの重要な要因である、「かみくだく過程の支援」や「一体感の形成」についての取り扱いを明確にする必要がある。共同問題解決活動に携わる場合、結論にいたる速度、質は人によって異なる。早く達成した人が、ゆっくり、かみくだくように、要点を「他人の能力」を勘案しつつ説明する状況が話合いでは良く出現する。これを「かみくだく過程」と呼ぶことにする。かみくだく過程は、話者相互間で、理解の差や決定内容に関するくい違いが自覚され、それをなくそうとする努力がなされている場合にも生じる。

かみくだく過程において発生する会話を記録することは、その内容が人の経験・知識・体調等に依存するものであり再現性がないので、あまり意義はないと思われる。もちろん、かみくだく過程を支援することは重要であり、参加者の間に「一体感」を生み出し、進行状況や次の議論の目的 (決定事項と未決事項) の共有を助ける。かみくだく過程の支援はグループウェアやユーザインタフェースの役割であると判断する。

会話の流れは会話参加者の関心の推移を表現する。そこでは参加者の間で最も関心の高い事項から会話が始まる。その会話の前提条件の欠落に途中で気付き、一時話題がそちらに移ることもある。ある話題について話が一段落した時、続いて話合うべきことが話題として持ち上がる。一度の話し合いでは結論に達せず話を中断することもよくある。この時、別の話題に話が移った後で、突然、あることに気付いて、話題がもとに戻ることもある。このような会話の流れの中で決定されていく事項をどのように記録すれば良いの

だろうか。日記を書くように、会話の流れに沿って記録することは一番容易であるが、決定事項間の因果関係を明示的に表現できない欠点がある。

我々は、話合いの途中で、そこまでの話を整理し、解決された問題とそうでないものを整理した上で次に話し合うべきことを決定するような場面をよく経験する。この時、我々は以下のような情報を整理していることが多い。

- すでに決定されたことと、これから決定されるべきことに対する全体像
- 全体像に対する、現状の位置づけ
- 最終目標に対する進捗状況
- 次に話し合うべき話題

このような事柄は当然記録の対象とすべきであろう。

ところで、議論が進むにつれてそのような整理結果が変化することを考慮に入れる必要がある。また、いったん決定したことが不完全であったことに後になって気付くこともしばしばある。討議結果の再構成や蒸し返しを支援できる情報も記録の対象に入れるべきである。そのためには以下のような情報を記録しておくことが必要である。

- ある話題に対する、部分決定の積み重ねの構造とそれらの決定にあたって必要なさまざまの前提条件
- 一つの部分的決定に関する確信度と手間

さらに、話し合いの話題はソフトウェア・オブジェクトに起因するので、

- 中間生成物や中間生成物間の依存関係との関連

を記録対象とする必要性は明白であろう。

3.1.3 中間生成物間の粗粒度レベルの依存関係を統合の土台とすることについて

分析・設計活動においては、ある中間生成物が入力され、既存の成果物、標準規約、品質基準等を参照しつつ、ツールによる変換や人間の思考によって、出力となる中間生成物や製品が生成される。すなわち、中間生成物や製品の間には入出力に関する依存関係が存在する。

粗粒度レベルの依存関係は、仕様書や設計書などの文書間の入出力依存関係を定義する。細粒度レベルの依存関係は、それら文書中の項目間の「利用する／される」または「参照する／される」という依存関係を定義し、設計根拠を理解するための基礎を与える。このような理由で、我々は、中間生成物間の粗粒度レベルの依存関係をソフトウェア・プロセス・モデルの定義とコミュニケーションの記録を統合する基礎として採用する。

もちろん、中間生成物間の粗粒度レベルの依存関係が上記の機能実現の土台として必要十分であるとはいいがたい。しかし、変換作業の実行順序の設計にあたって制約を与え、会話の発生順序に関する概略を与えている意味において、統合の主要な土台の一つであると考える。

3.1.4 粗粒度レベルの依存関係を基にしたタスクとソフトウェア・プロセスの定義

中間生成物／製品の部分集合を、チーム構成員それぞれに各自の役割に応じて割り当てる。これをタスクと呼ぶ。それぞれのタスクは、例えば、割り当てられた中間生成物や製品の記述に関する様式や規約、作業遂行上の制約、他人との仕事の関係、タスク開始／終了条件、品質基準などの属性を持つ。属性定義の詳細は個々の組織やプロジェクトに依存するので、ここでは、プロジェクト・マネージメントに有用な、すべてのタスク属性定義に必要となる共通性質を検討する。少なくとも、以下の機能の実現に必要な情報をタスク属性として定義すべきであると考える。

- 順守すべき制約事項が妥当で明確であること。
- 制約が犯された場合には、システムがそれを検知できること。
- 制約違反に対しては、システムが適切な対応を指示できること
- その処置が他人に悪影響をおよぼす場合には、その調整をシステムが支援できること。

中間生成物／製品間の粗粒度レベルの依存関係に基づき、それを基にして作業遂行者への仕事の割り当てを定義したタスク間に、実行順序を付加することにより、ソフトウェア・プロセスを定義する。プロジェクト全体に割り付けられる時間資源と、能力や容量に応じて各自が必要とする時間資源をバランスさせるところにプロセス定義の難しさがある。

3.1.5 ソフトウェア・プロセスの実行と発生するコミュニケーションの関係について

ところで、コミュニケーションを通じて決定されていく事柄の間の因果関係[1]と、タスク間の時間的順序関係は同じ構造を持つのであろうか？ 後の例に述べるように、話題が状態に依存して発生することがあるので、討議空間の構造はタスクの実行順序の構造と必ずしも一致しない。そこで、「決定事項間の因果関係」と「タスクの実行順序関係」はそれぞれ独立に管理する必要がある。

討議の流れとタスクの実行構造が対応する場合

タスクに割り当てられた人間が複数人である場合と、タスク間で中間生成物が直接または間接に共有される場合にコミュニケーションが発生する。発生するコミュニケーションにはいくつかの種類がある。例えば以下の例は代表的である。

- 複数人で一つの仕事を共同で実施する場合、それぞれが持つ専門知識を交換するため、代替案の中から解候補を選択するため、スキルや経験の差を補うため (かみくだく過程) にコミュニケーションが発生する
- タスク間で中間生成物が共有されている場合、生成物の伝達のため、伝達内容に関する合意形成のため、中間生成物の不完全さに起因する誤りを検知・修正するためにコミュニケーションが発生する

この場合、討議の流れは、タスクまたはタスク間の構造に直接関係する。

討議の流れとタスクの実行構造が対応しない場合

機能仕様書は複数の機能要求から成り立っている。設計段階において、それらの機能の内、一部は性能要求を満たすため早いプロセッサに割り付けられ、残りの機能は遅いプロセッサに割り付けられる。それぞれはまとめられてモジュールになる。それらのモジュール間のインタフェースが決定され、大域データと各モジュールのデータ構造が決定され、各モジュールが実装される。テスト時にロードモジュールが十分な性能を出せなかったとしよう。すでになされた以下の 2 種類の決定を変更しなければならない。

- 粗粒度レベルの決定：機能仕様書、性能仕様書、プロセッサの選択、二つのモジュール、ロードモジュール。
- 細粒度レベルの決定：ある機能のプロセッサへの割りつけ、割りつけられた機能のモジュールへの統合、モジュール間インタフェース・データの選択、データ構造とアルゴリズムの選択。

プロセッサの選択は妥当であったか？ 機能の振り分けは妥当であったか？データやアルゴリズムの選択は妥当であったか？ という疑問に関連する決定事項を再検討しなければならない。

[1] これを後に討議空間として定義する

結果として、さまざまの仕様書やプログラムを変更する必要がでてくる。これに対しては、独立した討議を起動し、その討議のサブ討議として、関連する仕様書やプログラムに関する決定内容を検討し、必要なら変更した上で、変更結果をその理由と共に追加記録することになる。

この例のように、予測不能な異常状態が出現した場合には、ソフトウェア・プロセスとは独立した討議を起動する必要がある。すなわち、一般に「決定事項間の因果関係」と「タスクの実行順序関係」は異なるものである。

3.2 情報リポジトリへの要請

「自在」環境の中核となる情報リポジトリにおいては、「共同作業を通じて生成・変更されていく中間成果物や決定事項とそれ等の状態を、現実世界の状況を正確に反映しつつ管理すること」を目標にしている。その目的は、予測不可能性、非再現性、不完全性などの人間要因に基づく混乱を軽減させる一つの手段として、共同作業によって生成されていく生産物とその状態を関係者間で共有させることにある。

このとき、粗粒度レベルの文書間の依存関係を、作業遂行に関する基本的な制約としつつも、

1. 共同作業参加者の各自の責任範囲と他者との関係の明示
2. 共有情報の生成と変更の管理
3. それらに関して発生するコミュニケーションの円滑な支援

等が情報リポジトリ設計の基本的な要請である。このため、自在における情報リポジトリでは、中間生成物や決定事項にそれぞれ状態遷移図を付加することにより、状態に関する情報を明示的に記録・管理し、また、それにより現実世界の状況の変化を正確に反映することを試みる。最終的には、振舞いに関する制約の基に並行動作する自律オブジェクト群の管理機構として定式化することが目標であり、故障オブジェクトの出現にともなうネットワークワイドな故障検出機構として調整支援機能を実現することが目標であるが、ここではその目標達成のための基礎考察を行なう。

3.2.1 各自の責任範囲をタスクとして定義することについて

独立性 (独立性という用語は「他人との間で不必要なコミュニケーションや相互干渉を持つことなく、各自が自身の仕事を進められること」という意味合いで使用している) という目標を達成するための一つの課題として、共有情報の管理法が挙げられる。

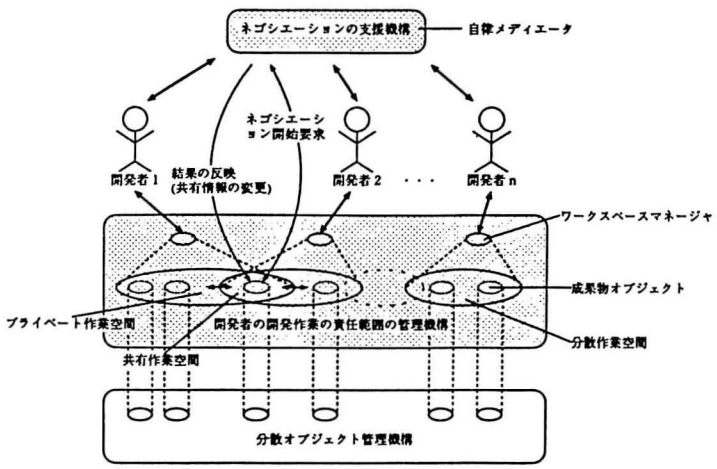
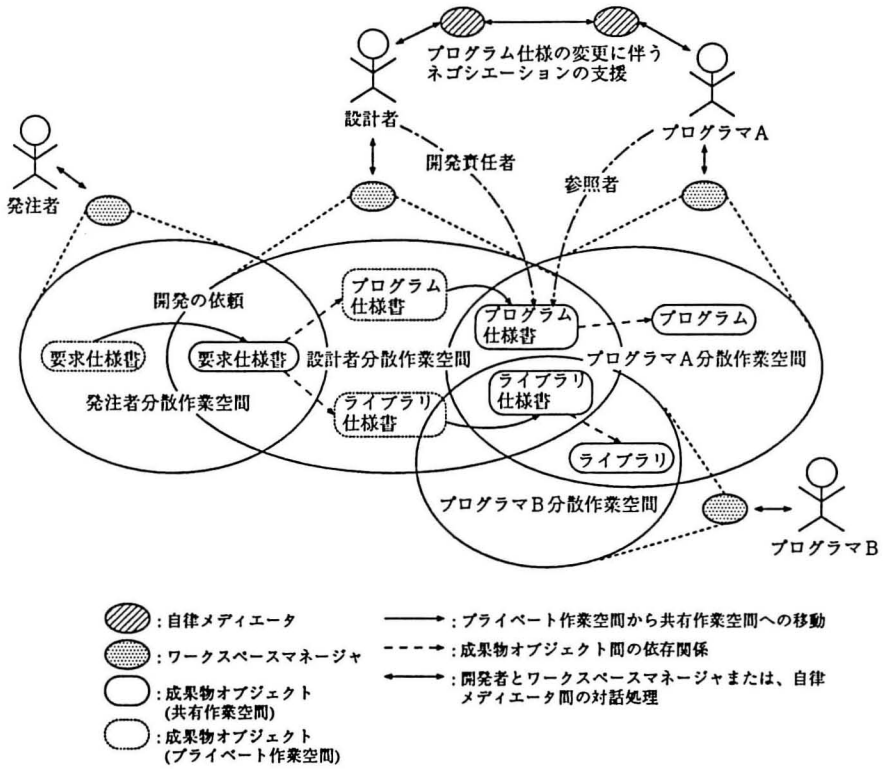


図 3: 共有情報の管理機構

図 3に我々のアプローチを示す。図 3において、共同作業で生成される中間成果物オブジェクト (文書等) 全体の部分集合を「仕事の責任範囲」として定義し、各参加者に割り当てる。この部分集合を分散作業空間 (またはタスク) と呼び、作業空間管理者オブジェクトが管理する。共有情報は分散作業空間の積集合として定義しこれを共有作業空間と呼ぶ。共有作業空間に属する中間成果物オブジェクトはそれぞれアクセスリストを持ち、許可されていないアクセスに対しては作業空間管理者オブジェクトを介して自律メディエター・オブジェクトを起動し、関連者間での一時的な権限の委譲に関するネゴシエーションを支援する。

我々はすでに、作業空間管理オブジェクト、自律メディエターと名付けた二つの制御オブジェクトを定義・開発した [20]、[21]。作業空間管理オブジェクト、自律メディエターの主な特徴を図 4に示す。



	ベースオブジェクト	メタオブジェクト
自律 メディエータ	・開発者への問い合わせ ・他のオブジェクトのメソッド呼出し	・ハーフプロトコル（状態遷移図）によるベースオブジェクトのメソッドの起動制御
ワーク スペース マネージャ	・作業空間属内の成果物オブジェクトの管理（プライベート⇔共有作業空間の移動とそれに伴うアクセス権の変更） ・開発者による成果物オブジェクトへのアクセス要求の受付け	・共有作業空間に属する成果物オブジェクトへの変更許可を求める、管理責任者のメタワークスペースマネージャとのネゴシエーション ・ネゴシエーションを支援するための自律メディエータの起動
成果物 オブジェクト	・中間成果物の管理	・開発者毎のメソッドレベルのアクセス制御

図 4: 各オブジェクトの役割

図 4において、

1. 作業空間管理者オブジェクトと中間成果物オブジェクトは、ソフトウェア開発者各自の仕事の責任範囲を規定し、共有データの管理を支援する
2. 自律メディエター・オブジェクトは、共有データの変更にともなって発生するソフトウェア開発者間のネゴシエーションを支援する

3. それぞれのオブジェクトは, 対応するメタ・オブジェクトを持ち、状況に応じて適切な振舞いを動的に選択することを可能にする

これらのオブジェクトを利用することにより、ソフトウェア技術者が、彼の責任範囲に直接関係する知識と、彼とデータを共有する人との関連のみに注意を払って仕事を進めれるようにすることが目標であり、このとき、データ共有に関する方針を協調的かつ柔軟に変更することを支援することも目標である。

中間成果物オブジェクトに図 5に示す状態遷移図をそれぞれ内蔵させることにより、中間成果物の進捗状況を反映させる。

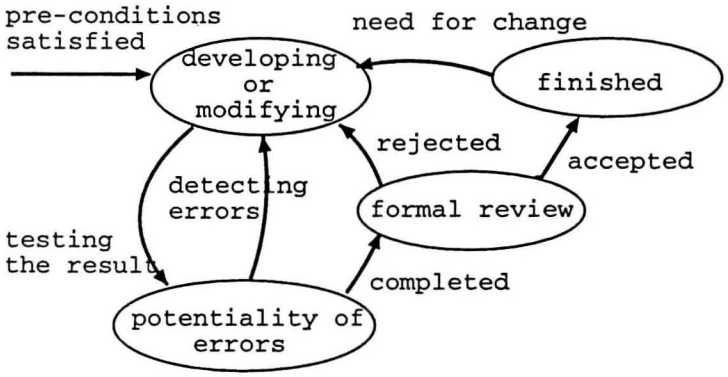


図 5: 中間生成物に付加される状態遷移図

3.2.2 粒度の異なる決定事項を系統的に管理することについて

会話を通じてなされる決定事項を、図 6に示すように、討議空間、討議プロセス、討議の型からなる 3 層記録スキーマで管理する [23]、[48]。

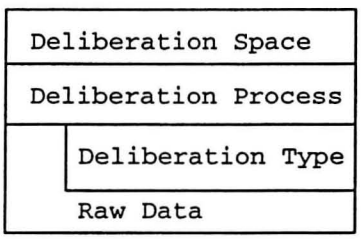


図 6: 会話を通じてなされる決定事項の管理法

上記スキーマを持つリポジトリをグループウェア・ベースと呼ぶ。次節以降に、討議空間、討議プロセス、討議の型をそれぞれ説明する。

3.2.3 討議空間

- それぞれの話題に対する決定事項間の因果関係を討議空間で表現する。その役割は、決定事項の全体像と最終目標に対する現在の議論の適切な位置づけを我々に提供することにある。

討議空間は、後に定義する討議プロセスのグラフであり、節は討議プロセス、枝は討議プロセス発生の原因を表現する。

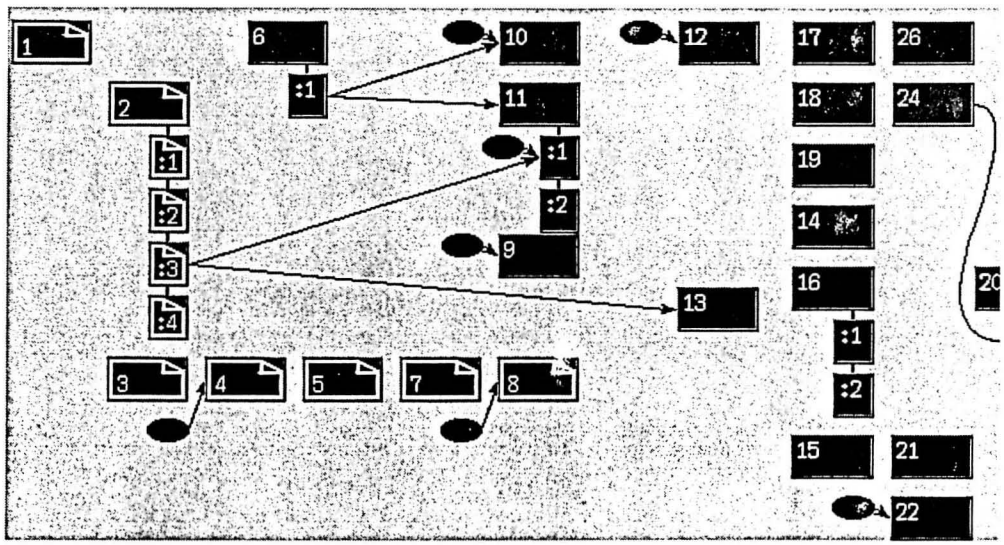


図 7: 討議空間

図 7は討議空間の一例を示している。長方形の箱は討議プロセスを表わしている。正方形の箱は、長方形で表現された討議プロセスを遂行するために発生した子討議プロセスを表わしている。 枝には三つの型がある。

継続 ある討議プロセスが、別の討議プロセスの完了に伴ない発生する

詳細化 ある話題に関する討議を完結するためには、いくつかの部分話題を討議する必要がある。

相互関連 二つの討議プロセスが、ある内容を共有して並行に走行している。

討議プロセスに割り当てられた数はプロセス ID である。プロセス ID が 1 から 8 の討議プロセスは決定事項を文書中の状態にある。他の討議は現在進行中である。他の状態は色によって区別される。箱をクリックすることによって、討議の型でグループ化された発言内容を確認することができる。発言の内いくつかは、討議の型を介さず箱に直接つながっている。楕円の記号は「コンテキスト」と呼び、利用されたグループウェアを使用しない状態で発言された内容を保持している。

3.2.4 討議プロセス

- ある話題に対する一連の会話を討議プロセスで管理する。その役割は、討議の型を話題対応に討議プロセスとしてまとめることにより、ある話題に対する、部分決定の積み重ねの構造と、それらの決定にあたって必要なさまざまの前提条件を管理することにある。

討議されるべき話題は数多くあり、その一つ一つが討議を生成する。いくつかの話題に対するそれぞれの討議内容は通常まざりあっている。討議の参加者がそれぞれの話題に対応する討議の進捗と筋道を把握できることは有用である。同じ話題に対する発言や会話をグループ化するものとして討議プロセスを導入する。

討議プロセスは、図 8に示す状態遷移図にしたがって推移する。討議プロセスは、それぞれの話題に対して生成され、討議の期間中生存する。討議はいくつかの前提条件が満たされたとき開始される。前提条件の例として必要な資料の準備、関連する人々の参加、指定された時間の到来などが挙げられる。いくつかの会話が状態「討議」においてなされる。権利を持つ誰かが宣言することで、討議は終了し、結果 (討議を通じてなされた決定) が文書化される。

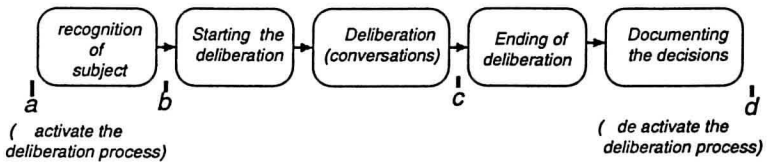


図 8: 討議プロセスに付随する状態遷移図

3.2.5 討議の型

- 田中等 [54] による、会話の型に関する研究成果に基づいて、我々は会話の型を：伝達と整合、決定、創造の三つに分類した。会話中の一連の発言を伝達・整合、決定、創造などの討議の型で管理する[2]。ソフトウェア開発プロセス時に発生するほとんどの会話は以下の三つの型に整理することができる。良く知っている人がそうでない人に決定事項や知識を「伝達」する。お互いの言い分を聞きながら、いくつかの代替案の中から一つの案を「選択・決定」する。部分的な知識を持つ人々が、それらを統合して新しい情報を「創造」する。討議の型の役割は、一つの部分的決定に関する確信度と手間を記録することにある。伝達・整合、決定、創造などの討議の型はいくつかの状態を持ち、どの状態で会話が中断・終了したかにより、確信度や完遂度を判定できる。時間属性や発言の回数は決定に要したコストを表現できる。

伝達・整合の図式表現

「伝達・整合」の目的は、質疑応答を繰り返すことにより、情報や知識が共有された状態に達することである。図 9の図式表現を状態遷移図により示す。

まず、アイデア、指示、生産物の提示・引き渡し等に駆動されて会話がはじまる。受け手はその内容を吟味する。もし、受け手が内容を理解した場合は、それを受理し、送り手と受け手は「相互理解」の状態になる。もし受け手が態度を鮮明にしない場合には「不一致の可能性」がある。そこで、受け手が何らかの質問を送り手に対して発する。この段階で送り手と受け手は「不一致」の状態になる。ここで、送り手と受け手の間でギャップを埋めるための努力が、例えば、いくつかの質疑応答を通じて開始される。会話は「受理」または「不一致」のどちらかの状態で終了するだろう。前者は伝達・整合の成功を意味し、後者は伝達・整合の失敗を意味する。

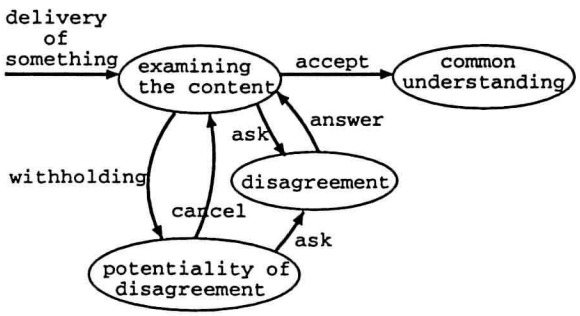


図 9: 伝達・整合

[2]それぞれは会話の筋道を記録する上で基本的かつ重要なものであるが、もちろんすべての会話の型をカバーするわけではない

例 1：生産物の引き渡し

それぞれが定められた役割を持つ二人の人間の存在を仮定する。図 10 にソフトウェア開発のドメインからの例を示す。ソフトウェア設計者は仕様を定義しそれをプログラマに渡す。この時、仕様内容について共通理解を達成するためにコミュニケーションの初期の段階で伝達・整合の会話が発生する。

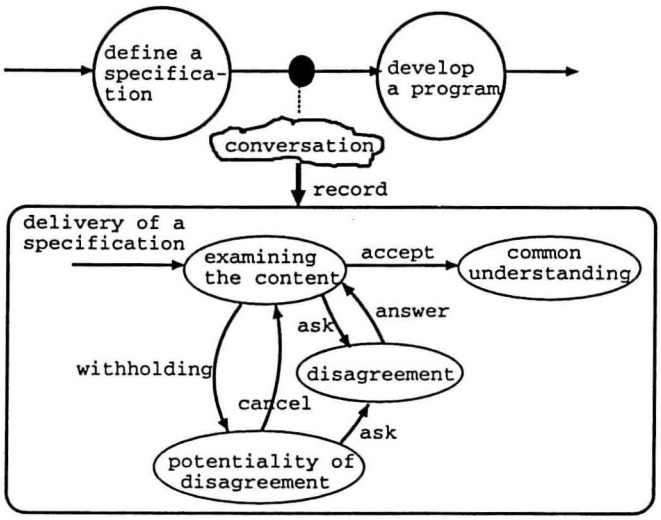


図 10: 生産物の引き渡し

両者または、後に関与することになるその時点での第三者にとって、何が合意され何が合意されていないかを明確にしておくことは有用である。通常、この型の情報は生産物の引き渡し後には消滅することが多く、種々の問題を引き起す場合が多い。

決定の図式表現

「決定」の目的は、図 11に示すように、解に対するいくつかの候補の中から、適切なものを一つ選択することにある。この型の会話は解に対する複数の解候補が存在し、選択・決定が必要であることから発生する。その意味で、初期状態は「複数の解候補の存在」とする。参加者の内誰かが、議論の対象範囲を狭めるため、理由と共に意見を開陳する。その意見を吟味した後、ある場合には、それが拒否され、またある場合にはそれが受理される。前者の場合はさらに他の意見が表明されるだろう。いくつかの議論を経た後、会話は収束し、候補の内のどれか一つが解として選択されることになる。

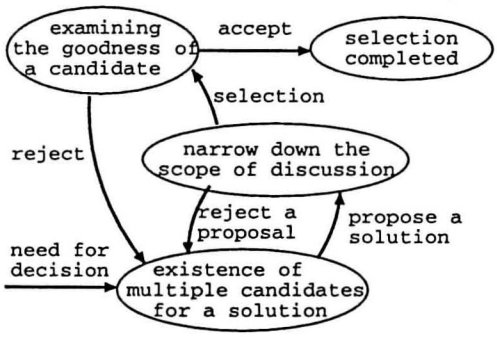


図 11: 決定

例 2：ソフトウェアレビュー

図 12もソフトウェア開発のドメインから取った例である。ソフトウェアレビューのプロセスはソフトウェア開発者とレビューアの間の意思決定のためのコミュニケーションプロセスである。

まず初期には、「受理」か「拒否」かの二つの可能性がある。意見はレビュー対象物のすべての欠陥を明示的に洗い出す形で表明される。もし結果が拒否であった場合は、これらの意見の内のいくつかがレビュー対象が拒否された理由と記録される。

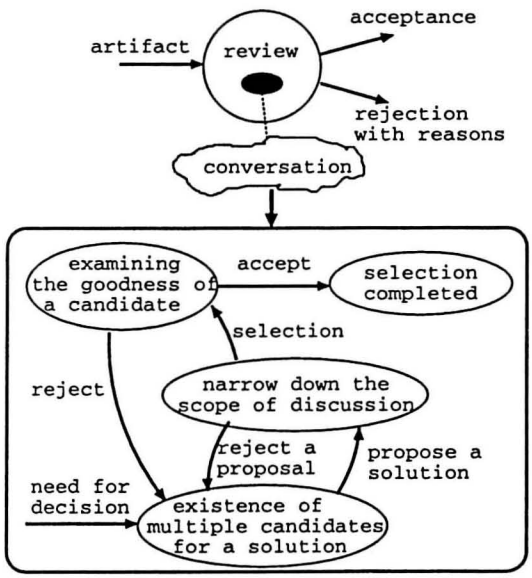


図 12: 決定

創造の図式表現

「創造」の目的は何か新しいものを創ることにある。「創造」に対する図式表現は「伝達・整合」と「決定」を合成することで図 13のように表現される。

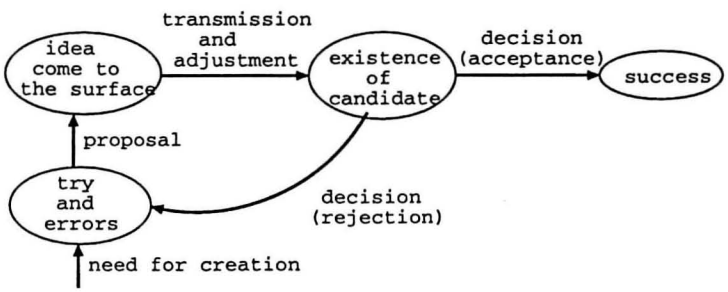


図 13: 創造

討議の型の組み合せ

いくつかの会話は、以下のような、討議の型の組み合せによって表現される。

「決定」と「伝達・整合」の組み合わせ

チームの構成や方針にしたがって、決定のあるものについては、チーム構成員の内、主要メンバーによってなされることがある。この場合には、「決定」後、数多くの「伝達・整合」の会話を利用して、全体的コンセンサスを得るための努力が決定に関した人以外のメンバに対してなされる。

「伝達・整合」と「決定」の組み合わせ

図 10に示す例においては、最初は伝達・整合の目的で開始された会話が、送り手と受け手の間での厳しい対立が発生し、会話は「伝達・整合」のみによっては終了せず、「決定」の型の会話が開始される。この時、「伝達・整合」の会話の状態は「不一致」の状態に留まる。

3.2.6 グループウェアベース

上記スキーマを持つソフトウェアリポジトリを、我々はグループウェアベースと呼ぶ。グループウェアベースは、図 14に示すような管理ページの集合である。管理ページは各討議プロセスごとに作成される。図 14において、項目 5 は議論の蒸し返しを支援するための情報を保持しており、項目 12 は討議の型のインスタンスに対するリンクを保持している。項目 14 の値は、完了、延期、キャンセルされた、開始されていない、の四つの内のどれかの値をとる。

1	process ID
2	subject
3	goal of a deliberation
4	preconditions for a deliberation
5	resteaming
6	coordinator
7	decision maker
8	participants
9	time resources for a deliberation
10	activation time
11	starting time

12	history of the deliberation(types)
13	deactivation time
14	achivement level
15	the time for documenting the decisions
16	decisions
17	raw data
18	parent process ID
19	child process ID
20	successive process ID
21	co-related process ID

図 14: 管理ページ：グループウェアベースの記録単位

3.3 共同作業のモデル：CSCSD モデル

前章までに検討した環境構成の諸要素を図 15に示すように、階層構造として構造化する。

<table>
<tr><td colspan="3">User interface
for works and context understanding</td></tr>
<tr><td colspan="3">CASE tools for production and communication</td></tr>
<tr><td>Process server (for defining order of execution)</td><td>Distribution server(for controlling shared artifacts</td><td>Grouware base (for recording reasoning of decisions)</td></tr>
<tr><td colspan="2">Definition of tasks and constraints</td><td></td></tr>
<tr><td colspan="3">Management of coarse-grained dependency relationsips among artifacts</td></tr>
<tr><td colspan="3">CSCS server linked to functional services for distributed computing systems</td></tr>
</table>

図 15: CSCSD モデル

図 15において、

ユーザインタフェース (最上層): ユーザインタフェースには、分散サーバ、グループウェア・ベース、プロセス・サーバの状態が表示されると共に、 Ellis が指摘した各種情報、構造的コンテキスト、社会コンテキスト、組織コンテキストが必要に応じて表示され操作される。共同作業の参加者が、「一体感」を形成することを補助する。

ツールキット層 (第 2 層): 作業の自動化の度合を高める CASE tool や、「かみくだく過程を支援するグループウェアを置く。これらはチーム構成員の分散度や作業の規模等のパラメータに応じて配置される。ツール統合機構により作業の連続性を保証する。また、アクティブ・チャネルのような新しいアプリケーションもこの層におかれる。

分散サーバとグループウェア・ベース (第 3 層): 分散サーバ、グループウェア・ベースを置く。分散サーバはさらにプロセス・サーバと分散作業空間からなる。プロセス・サーバは、タスクとその実行順序、消費可能資源等を定義する。分散作業空間は、作業分担と責任の範囲を明確にし共有情報の管理を行なう。グループウェア・ベースは、会話による決定事項とその間の因果関係を保持する。

中間生成物間の依存関係の管理 (第 4 層): 中間生成物間の依存関係を管理する。上層の分散サーバを構築する際、作業のガイドラインや各自の作業に関する制約を定義し、またグループウェア・ベースに対してはコミュニケーション記録にあたって、会話内容がどの中間生成物やその依存関係に関与するのかを定義する土台を与える。

CSCSD サーバ (最下層): 規模への対応、非均一性への対応、計算機環境への対応などを支援する。

4 自在プロトタイプの研究現状

本章では、1章、2章、3章における問題提起に一つの解を与える目的で進めている、落水研究室における「自在プロジェクト」[49] の進捗状況を紹介する。

ユーザインタフェース (最上層): におけるプロトタイプはまだ存在しない

ツールキット層 (第 2 層): においてはPCTEによるツール統合技術を試行している段階である。また、アクティブ・チャネルに関してはフィルタリング対象がその利用につれて拡大されていくような情報フィルタリング技術について調査を終了した段階である。

分散サーバとグループウェア・ベース (第 3 層): については、それぞれプロトタイプ「群舞」と「栞」が稼働しておりその評価実験を開始する段階である。後に詳述する。

中間生成物間の依存関係の管理 (第 4 層): については、すでに多くの研究成果があり、プロトタイプ開発の対象としない。

CSCSD サーバ (最下層): については、当面、様々な計算環境で市販または試験的に提供されているサーバ群を利用する予定である。

本節では、まず、上記の内、CSCSD モデルに基づくプロトタイプ「自在」の開発現状を以下の 4 つに関して紹介する。

- 「携帯性向上」のための CSCSD サーバ：インタネットワーキング、移動計算環境、遍在情報環境などの新しい計算環境下において、データの非均一性を吸収することにより、個人の計算環境の携帯性を増加させる。プロトタイプ「飛翔」を開発中である。
- 「独立性を保証する」分散サーバー：「分散作業空間オブジェクト」や「自律仲裁オブジェクト」などの制御オブジェクトにより、共有情報の管理を支援する。モデルとプロトタイプ「群舞」をすでに開発した。
- 「円滑性を保証する」グループウェア・ベース：討議の進捗と筋道を記録することにより、決定事項の管理を支援する。決定の粒度に基づいて構成された 3 層スキーマを定義し、プロトタイプシステム「栞」を開発した。
- 「探求的学習支援」のためのアクティブ・チャネル：上記サーバ上で稼働する新しいアプリケーション。プロトタイプ「探求」を開発中である。

次に、分散に依存する問題点をより精密に把握し、快適性の既存の要因を評価し、さらに新たな要因を発見する目的で実施しているプロトコル解析実験の中間結果を紹介する。

4.1 携帯性向上のための CSCSD サーバ

インタネットワーキング、移動計算環境、遍在情報環境などの、新しい計算環境に対する基盤が発展しつつある。これら一群の新しい計算機環境は、どのような新しい可能性を我々にもたらすのであろうか。一口でいえば、「活動の場から、必要な情報に迅速にアクセスできること、さらには活動の場にそれを支える情報が追随できること」であろう。これらの新しい計算機環境の特徴の一つは非均一性である。自在プロジェクトの一貫として開発中の CSCSD サーバの目的は、インターネットと移動体通信技術、移動計算環境と遍在計算環境を活用できるインフラストラクチャを提供することである。

このようなサーバを構築するにあたって考慮すべき点がいくつかある。例えば、データの非均一性や操作の非同期性への対応がある。Gio Wiederhold は、この問題に対して、従来の Client-Server モデルを、Client と Server の中間に Mediator をおく Mediatted Architecture に拡張することを提案している [56]。

Mediator は、適切な情報資源へのアクセス、データ選択、フォーマット変換、データを共通抽象レベルで整理すること、異なる情報資源からの情報を統合すること、アプリケーションの目的に応じたメタ情報を準備することなどを支援する。

このようなソフトウェアは一般にミドルウェアと呼ばれ、すでに、CORBA(Common Object Request Broker Architecture)、DOE(Distributed Objects Everywhere)、ILU(Inter-Language Unification)、KQML(Knowledge Query and Manipulation Language)、OLE(Object Linking and Embedding)、OpenDoc(Open Document Exchange)、PDES(Product Data Exchange using STEP) などの商品が出ており、今後標準化が推進される (http://dc.isx.com/I3)。

また、S. Heiler は、大規模分散システムにおける相互互換性 (interoperability) の問題を指摘している [19]。非均一なシステム構成でサービスやデータを交換するために、通常の、メッセージパッシングプロトコル、手続き名、エラーコード、引数の型などの合意に加えて意味的レベルの互換性も保証する必要がある。たとえば、データ名の与え方の問題に関して、 2 人のデータベース設計者がいるとして、それぞれはドメイン・エキスパートでもあるとする。このとき、同じドメインの同じデータ要素であっても、それに対して同じ名前が与えられる確率は 7%から 18%の間であるという事実を例にあげていて興味深い。

Frank Manola は情報システムをオブジェクト指向アーキテクチャで構築する際の相互互換性の問題を論じている [36]。古典的な情報システム構築法は、例えば、D.Pascott による DATARUN 方法論 [51] で述べられているように、ほぼ完成の域に近づいたといえる。均一な世界においてはこの手法で十分であるが、大規模分散システムでは、世界はヘテロになり、その間でデータ交換等が保証されなければならない。データベース、共有サービス、アプリケーションの各レベルで、データ表現、オブジェクト・インタフェースなどの相互互換性が保証されるべきである。

開発されるべき CSCSD サーバに対する機能要請をまとめると以下のようになる。

(1) 規模への対応: 規模 (遅延時間) に応じて、大域的制御構造、通信プロトコル、同期、データ・アクセス法などを適切に設定できること。または、基本部品を基に、規模に応じた適切なサーバを容易に構築できること

(2) 非均一性への対応: 適切な情報資源へのアクセス、データ選択、フォーマット変換などが保証されること

(3) 計算機環境への対応: マルチプラットフォーム上で稼働可能なこと

(4) アーキテクチャの柔軟性の保証: オブジェクト指向アーキテクチャに基づいて設計されていること

4.2 独立性を保証するための分散サーバ

すでに説明した、作業空間管理者オブジェクトと自律メディエター・オブジェクトを実装し、「協調書き込み」と名付けた動作の有効性を確認した (図 16)。

次の研究段階では、作業空間管理者オブジェクト、自律メディエター・オブジェクトに一定のレベルの知性を付加することにあり、トリオ環境 [35]、[3] における研究成果を適用することを予定している。トリオ環境における研究成果は、その振舞いが時制論理で仕様化されたオブジェクトの並行制御とコミュニケーションの問題を検討することでソフトウェアプロセスモデルに拡張可能であると思われる。例えばトリオ環境では、オブジェクトの振舞いは以下のセマンティックスを表現することで形式化される。「事象Aの発生に伴ない、事象BがX秒以内に発生しなければならない」。

トリオ環境の成果をソフトウェアプロセスモデルに適用する場合には、検討されるべき点が二つある。まず第一は、トリオモデルではＷＦＦは「単一の現在時刻」を基準にして表現される。ソフトウェア・プロセス・モデリングにおいては、それぞれのオブジェクトが固有の現在時間 (仕事の開始時間) を事前条件と共に持つように定義する方が自然である。また、ソフトウェア・プロセスの実行中に生じるオブジェクト間のコミュニケーションも定義する必要がある。これは、並行実行制御とコミュニケーションの問題である。

二番目は、ソフトウェア・プロセス・モデリングにおいては、「…時間以内」に「必ず…しなければならない」という表現はプロジェクト管理者の単なる期待であることにある。人間の行動は予測し難く、また再現性がない。

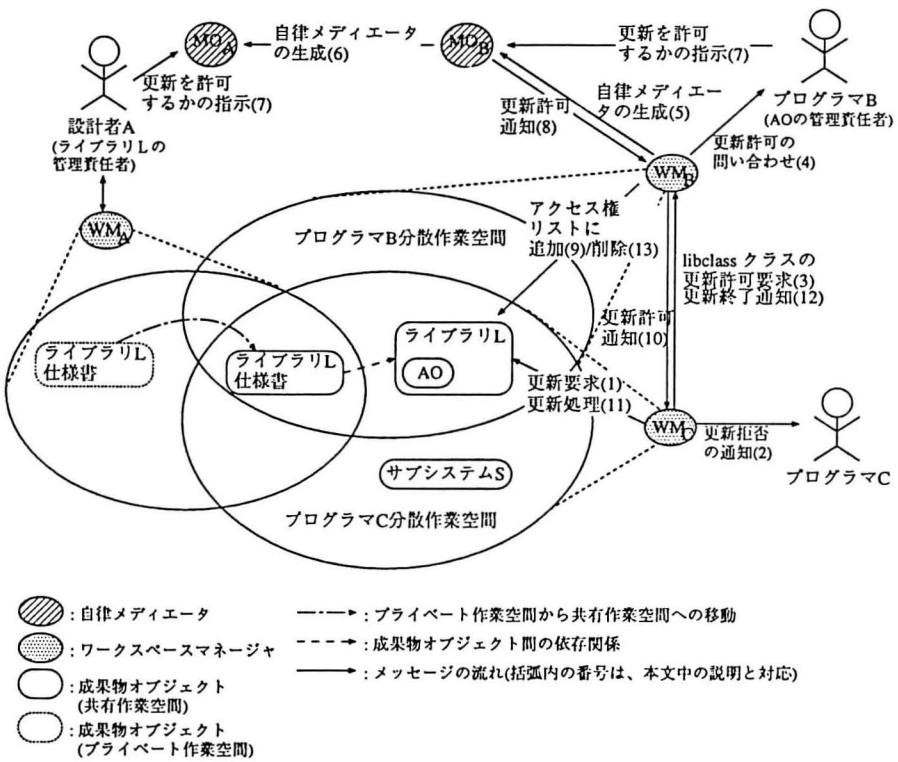


図 16: 協調書き込み

4.3 円滑性を提供するためのグループウェアベース

我々はすでに、上記のモデルに基づいてプロトタイプ・システム「栞」を開発済みである [30]。プロトタイプは、メーリングリストを利用した電子会議を対象として、討議参加者間のさまざまな意見の喰い違いの調整を支援し、また、決定事項の管理の不十分さに起因する冗長な会話の発生を減少させる目的で開発した (図 17)。

4.4 探求的学習を支援するアクティブチャネル

自在を利用した先進的ソフトウェア開発環境は、良いソフトウェアを開発するのに必要な日常活動; news や e-mail による情報の発信と獲得、内外の研究者とのディスカション、電子会議による合意形成・意思決定・共同設計作業、研究成果のリアルタイムな発表などの活動をネットワークを介して実施することも支援する [26]。この目的のために開発中のアプリケーションが、遍在情報に対するアクティブ・チャネルである。

今後、ネットワークに点在する膨大な量のデータの中から必要な情報のみを抽出・利用する情報フィルタリング技術を基にした情報検索技術の革新がおこるだろう。すでに WWW の普及により、ネットワークの各ノードに様々な情報が蓄積されつつある。このような情報を仮に遍在情報と呼ぼう。各地に散在する遍在情報の中から、情報獲得に関してごく限られた知識しか持ち合わせない人が、計算機の助けを借りて問題を解決する探求的学習の支援は今後の大事なアプリケーションの一つであると思われる。探求的学習を支援するツールをアクティブ・チャネルと呼ぶ。ここでチャネルとは情報の検索要求に応じて動的に張

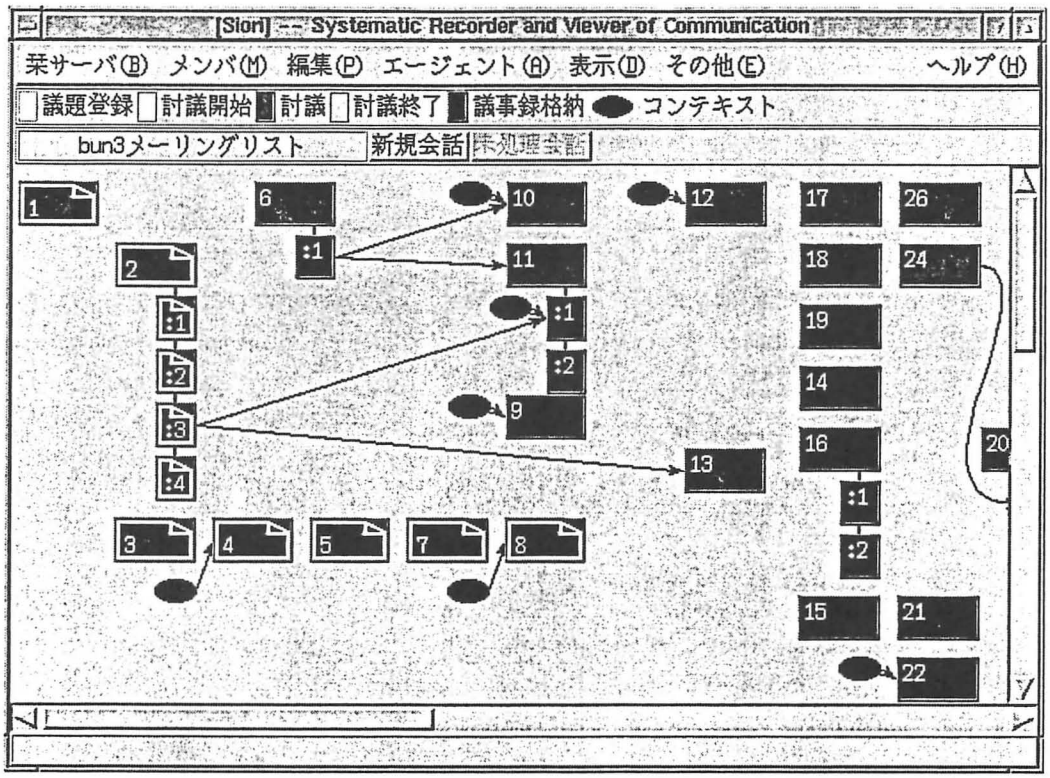


図 17: 菜クライアント

られる情報検索・フィルタリングのためのリンクを意味し、アクティブとは、一回の検索ごとに検索やフィルタリングに有用な情報が増加することを指す。

探求的学習に関する基本的な機能要請は Jennings によって議論されている [22]。Jennings は CSCW のタスクドメインを「手段」、「情報交換の媒体」、「探求的学習の支援」の 3 つに分け、各ドメインにおけるソフトウェア・エージェントが果たすべき役割を論じている。

(1) ここで、「手段」とは、既知のタスク・ドメインで、ある特定のゴールを達成する (出力を得る) 手段として CSCW を利用することである。

(2) 何が出力として得られるかは必ずしも予知できないが、それを達成するための手段を知っている場合には、CSCW は「情報交換の媒体」として利用される。ゴールはシステム利用者間のインタラクションによって達成される。

(3) 利用者はタスク・ドメインやある話題について、ごく限られた知識しか持ち合わせていず、かつ、特別な出力を持つわけではない。

第 3 番目のドメインに対するアプローチの一つとして協調的フィルタリングがある。協調的フィルタリングに関する研究動向を川瀬のサベイ [27] に基づき紹介する。

コンピュータ・ネットワークの著しい発達に伴い、コンピュータ環境下での知的生産活動においては、電子メール、ネットニュース、WWW を利用して多種・多量の情報を手にすることが可能になっている。しかし一方では、その情報の多種・多量性は爆発的に増大し続けており、もはや個人がそれらの情報すべてを処理するのは、事実上不可能になりつつある。したがって、ネットワークを流れている情報源から、自分にとって必要であると思われる情報を優先的に選択する何らかの手段を講じない限り、情報の収集・吟味に割く時間と労力は増加の一途を辿る羽目になる。知的生産活動における情報収集は、ある最終的な目的へ

至るためには不可欠ではあるが過渡的な作業である。したがって、その作業の負担を軽減することにより、活動がより生産的なものになるように支援することが可能になるのである。

こうした要求に対する具体的な技術として、情報フィルタリングが研究され用いられてきた。情報フィルタリングの基本的な考え方は次のようなものである。「ユーザのニーズ・興味・嗜好をプロファイルとして記述しておく。そして、コンピュータ・ネットワークに流れている大量の情報に対して、そのプロファイルを用いてふるいをかける (すなはち、フィルタにかける) ことによって、ユーザが必要とする情報だけを優先的に選択し、提供する」。

川の流れのように絶え間なく流通しているネットワーク上の情報群から、新しくかつ必要な情報を日常的に取り出すには、逐一検索するというよりもフィルタによってふるいにかけるというアプローチが有効である [4]。とは言え、情報フィルタリングのために必要な技術と情報検索のためのそれとでは、共通する部分が多く、両システムの最終的な目的は本質的には等しい [4]。実際に、WWW の有用なページを見つける場合は、検索というスタイルをとることが多い。すなはち、情報フィルタリングと情報検索は相まって機能すべきものである。

フィルタリング法は、ユーザが如何にして情報を選択あるいはふるいにかけるのかを基にして分類した、Malone による 3 つの分類 [33] がある。

Cognitive filtering 情報そのものの内容とユーザのニーズに基づいたプロファイルとをマッチングさせることで、ユーザに適切な情報を提供する。そのための方法としては、AND、OR、NOT のブーリアン操作子を使ったキーワード列からなるプロファイルを利用したり [33]、語彙間の関連性を空間ベクトルで表現したプロファイルを利用するものがある [14]。ユーザの振舞いをフィードバックすることで、自動的にプロファイルをアップデートするテクニックを持ったフィルタリングもある。たとえば、記事の注視時間の長さによってトピックスへの興味の度合を計り、それをプロファイルに反映させるものがある [37]。

Social filtering 社会的な人間関係と個人の主観による判断に基づいてフィルタリングを行う。つまり、この人はこの道の権威なのでこの人が言ったことには信憑性があるとか、ボスから来たメールは要注意だ、などの判断を用いる。

Economic filtering ある情報を得ることに、どれだけコストが掛かるかによってふるいにかける。このコストとは、情報何バイトにつきいくら、といった明示的な課金として表されることもあるし、「この情報は大勢の人に向けられたものだから情報単価は安い」とするような表現もある。

情報フィルタリングを用いる際の問題点は以下の通りである。

個人プロファイルのみに依存した場合の限界 先に述べたように、一般に情報フィルタリングは、ユーザが必要とする情報を優先的に収集するために、そのユーザのニーズ・興味・嗜好を反映させたプロファイルに基づいて行なわれる。しかし一方で、ユーザ個人のプロファイルのみに依存する場合、ユーザ個人の価値観によって限定された情報収集を繰り返すことになる。このことは、知識の広がりの硬直化を招くことになる。そして、新しい知識に出くわす楽しみを味わったり、積極的な探求欲求を刺激したりするのを狭めてしまう危険性がある [37]。

ゼロから新しい知識を求めるときの困難さ 個人が自分にとって未知の分野に関する情報を入手しようとするとき、何をキーワードにして情報源に当たればよいのかはっきりしない場合が多々ある [4]。

上記 2 つの問題点を克服するために、「協調的フィルタリング (collaborative filtering)」というアプローチが最近注目を集めている。その骨子は、以下の通りである。「ある情報に対して、それが有益であるのか、信頼性はあるのか、同僚にとっても役に立つのかなどの評価やコメントを、ユーザがつける。そして、その評価やコメントを仲間や同僚間で共有し参照できるようにしたり、評価結果を集計して順位付けの表示をし

たりすることで、フィルタリングの機能を果たす」。すなわち、協調的フィルタリング方式は、自分が情報を入手するときに他人の意見を参考にするという、我々が日常よく行っている行為に立脚している。

協調的フィルタリングは、前述の cognitive filtering と social filtering とを組み合わせたものであるとも言える。キーワードマッチングなどの機械的な処理やヒューリスティックな評価/コメントは cognitive filtering として行われ、新たに情報を得る際に評価/コメントを参考にすることについては social filtering として行われることになる。

協調的フィルタリングを利用したシステムは、既に幾つか登場しており、それぞれ成果をあげている。以下に 4 つの例を挙げる。

Tapestry [18] メールとニュースのためのシステムである。ユーザは読んだ記事に自由文の注釈(annotation)を付けることができる。この注釈は、記事本体と共にシステム内に保持される。そして、情報を欲するユーザは、query を発行することによって、記事内容だけでなく記事に付与されている注釈に基づいて記事を選択をすることができる。例えば、山田さんが fj.comp.music で何か面白そうな記事を探そうとしているとする。そこで、彼は query として 'midi' というキーワードを設定する。また、山田さんは、同僚の鈴木さんがこの手の分野に詳しくてニュースグループをいつもチェックしているということを知っているので、鈴木さんが注釈を付けた記事を探すための query も記述する。かくして山田さんは query に基づいてフィルタリングされた新たな記事を得ることが出来る。この場合、query がプロファイルとして機能している。また、鈴木さんが注釈を付け、それを山田さんが参考にすることで協調的フィルタリングが機能していることになる。このシステムは、query を記述する際に、SQL ライクな専用言語を使用する必要がある。そのため、何が知りたいのかを予めはっきりさせておくことが要求されてしまう。

Grouplens [52] ニュースのためのシステムである。ユーザは記事に対してその価値を例えば 5 点満点で投票する。そして、そうして行われた投票の履歴から、似たような評価傾向にある人、つまり興味の方向性が似ている人を割出し、その人達同士による評価を参考にすることができるようにする。例えば、木下さんが fj.soc.history の記事を Grouplens 用に改造されたニュースリーダで読んでいるとする。彼が行う採点は、Better Bit Bureaus と呼ばれるサーバに送られる。そのサーバでは、評価が集計され、どのユーザが似たような評価をしているのかが計算される。その結果、山本さんというユーザが浮かび上がった。そこで、木下さんが既に評価していて山本さんが未読であるような記事について、山本さんのためにその記事の予測スコアを BBB サーバは計算して配送する。予測スコアの計算は、木下さんと山本さんの類似度と木下さんの評価をもとになされる。その予測スコアは、改造ニュースリーダに表示され、山本さんにとってはその予測スコアが高いものほど有用な記事であろうことがわかる。木下さんが未読で山本さんが評価済みであるような記事に関しても同様である。なお、実際にこのシステムを利用するに当たっては、プライバシーの問題のため、実名ではなく匿名・変名を用いる。このシステムでは、評価履歴がプロファイルになっている。そして、評価を重ねていくことが、協調的フィルタリングを効果的に機能させる要となっている。このシステムは、ユーザが大人数存在しなければ十分に機能しない。そのために、対象となる情報ソースが限られてしまう。また、フィルタリングの機能を享受するまでに、ある程度時間がかかる。

Firefly [53] ユーザの興味を引きそうな映画・音楽に関する情報を提供するためのシステムである。これは WWW 上で展開されている。また、ユーザは変名を使用する。ユーザは Firefly のページにて、自分が今まで観たり聴いたりした映画・音楽作品について、7 点満点の評価点を付ける。その評価傾向に基づき、そのユーザが好みそうな作品が予測され、表示される。評価の集計とその結果を用いた予測計算のアルゴリズムは、Grouplens と似たものになっている。また、具体的な作品名などを query として与えれば、それに付けられた評価点・コメント・傾向が似ている作品名といった情報を手に入れることができる。さらに、ユーザ各人はページを持っており、そこに好きな作品名・アーティスト名・評論などを記述することができる。それによって、自分と似たような興味を持つ人の見識を参考にする

ことができる。このシステムでは、Grouplens と同じく評価履歴がプロファイルとなり、それによって協調的フィルタリングが機能する。また、自分でコメントを付けたり他人の意見を見たりすることもでき、そうした行為によってもフィルタリングの機能は増強される。このシステムは、Grouplens と同様に、大人数のユーザが必要である。また、評価がほとんどなされていない作品については、フィルタリングの効果が期待できないという点も、Grouplens と同様である。

Pointers and Digests [34] 小規模グループ内で、有用な情報を効率よく共有するためのシステムである。このシステムは、Lotus Notes の環境上で機能する。ユーザは、pointer なるファイルを作成する。この pointer は、a) 有用な情報本体へのハイパーテキストリンク b) その情報のタイトル、日付、保持されているデータベースの名前 c) pointer を作成した人によるコメント をパッケージングしたものである。pointer は、同僚や仲間にメールとして送信することもできるし、Information Digest なる形態で共有データベース化することもできる。このシステムでは、「ある特定の人物から送られる pointer を受け取りたい」あるいは「自分はある特定の仲間に pointer を送る」という設定が、プロファイルとしての役割になっている。なお、このシステムは、Firefly と同様に'recommender' と呼ばれることもある。このシステムを利用するためには、Lotus Notes を導入しなければならない。また、外部のネットワークに流れている情報群に対しての有効な機械的なフィルタリングをサポートしていない。

今後、従来の情報フィルタリングの技術に加えて、フィルタリング対象世界の拡大機能を追加することによりその効果の一層の増大をはかることが必要であろう。

4.5 評価のためのプロトコル解析実験

本論文においては、開発対象とその手段の明確化、作業の独立性の保証、コミュニケーションの円滑化、作業遂行に必要な知識の獲得、計算機環境の携帯性の向上、で特徴づけられる計算機環境の「快適性」を追求してきた。しかし、定義されたモデルや開発されたプロトタイプ環境の有用性を評価するには、評価のための手段・尺度が必要である。これらを発見・整備していく手段としてプロトコル解析が有用であると思われる。プロトコル解析実験の目的を次の三つに設定する。

分散に依存する問題とその評価尺度の発見

快適性を特徴づける別の特性の発見：携帯性、明確性、独立性、円滑性、増幅性を含め、快適性の要因を再吟味する

独立性や円滑性などの評価尺度の開発：上記各特性に対する代替メトリクスを発見する

4.5.1 分散に依存する問題とその評価尺度の発見

「分散」プロジェクトは、北陸先端大、九州大、東工大間の共同プロジェクトである [2]。「分散」プロジェクトの目的は、研究室における基礎的な研究成果をネットワーク上で評価する場を設けることである。北陸先端大では、「自在」プロジェクトのフィールドテスト環境としてして位置づけ、分散に依存する問題とその対応策について基本的な知見を得る立場で参加している。分散プロジェクトでは、以下のような共同実験を実施してきた。

- 遠隔プレゼンテーション、遠隔形式仕様レビュー、遠隔共同プログラミングなどを Internet、 ATM などのネットワーク設備上で、既存のマルチキャスト・ツールを利用して実施してきた。実験の目的は、既存のマルチキャスト・ツールやネットワーク設備に関する問題点を発見することにあった。すでに第一期の一連の実験を終え、ネットワーク層における遅延の問題、人間系におけるコミュニケーション・プロトコルの問題等、今後の研究・技術開発に有用な知見が得られた。

4.5.2 快適性を特徴づける特性とその評価尺度の発見

ネットワークを介した共同作業の阻害要因を把握するためのプロトコル解析実験も実施してきた[38],[39],[40],[41]。現在、一度の投げかける話題の量 (複雑さ) と会話における話題の完遂度との関係について基本的な成果が得られつつある。

4.6 3章のまとめ

決定事項の管理とその生成・変更の支援をモデル化の中心として、下記のような機能階層からなる共同作業の参照モデル CSCSD モデルを提案し、さらに、現在進行中の「自在」プロトタイプを紹介した。

ユーザインタフェース (最上層): ユーザインタフェースには、下層の各システム (分散サーバ、グループウェア・ベース、プロセス・サーバ) の状態が表示されると共に、 Ellis が指摘した各種情報、構造的コンテキスト、社会コンテキスト、組織コンテキストが必要に応じて表示され操作される。共同作業の参加者間に「一体感」を形成することを補助する目的を持つ。

ツールキット層 (第 2 層): 作業の自動化の度合を高める CASE tool や、「かみくだく過程を支援するグループウェアを置く。これらはチーム構成員の分散度や作業の規模等のパラメータに応じて配置される。ツール統合機構により作業の連続性を保証する。また、アクティブ・チャネルのような新しいアプリケーションもこの層におく。

分散サーバとグループウェア・ベース (第 3 層): 分散サーバ、グループウェア・ベースを置く。分散サーバはさらにプロセス・サーバと分散作業空間からなる。プロセス・サーバは、タスクとその実行順序、消費可能資源等を定義する。分散作業空間は、作業分担と責任の範囲を明確にし共有情報の管理を行なう。グループウェア・ベースは、会話による決定事項とその間の因果関係を保持する。

中間生成物間の依存関係の管理 (第 4 層): 中間生成物間の依存関係を管理する。上層の分散サーバを構築する際、作業のガイドラインや各自の作業に関する制約を定義し、またグループウェア・ベースに対してはコミュニケーション記録にあたって、会話内容がどの中間生成物やその依存関係に関与するのかを定義する土台を与える。

CSCSD サーバ (最下層): 規模への対応、非均一性への対応、計算機環境への対応などを支援する。

5 まとめと今後の課題

本論文では、ソフトウェア・エンジニアリング活動を対象として、計算機ネットワークを介した共同作業支援のためのモデルを定義し、環境を構築するにあたっての基本方針と原理を述べた。我々の目標は、明解性、独立性、円滑性、携帯性、増幅性の五つの特性で特徴づけられる、環境の快適性の向上である。その実現のために解決されるべき課題は以下の通りである。

- 規模に応じたソフトウェア・アーキテクチャ・クラスの整備
- リポジトリ管理技術 (特に決定事項の管理技術) とグループウェアの融合
- 調整支援を目的としたプロセス・モデルの開発
- 予測不可能性、非再現性、不完全性などの人間要因の考慮

次に、決定事項の管理に基づく共同作業のモデル、CSCSD モデルを提案した。CSCSD モデルは 6 層の階層構造から成る。それらは、ユーザインタフェース層、ツールキット層、分散サーバとグループウェア・ベース、タスク定義層、中間生成物間の依存関係の管理、CSCSD サーバである。

現在、「自在」プロトタイプについては分散サーバおよびグループウェアベースについて初期の研究開発が終了した段階である。新しい計算機環境と開発スタイルに対応できるソフトウェア開発環境の実現という目標に対して、今後の進展させるべき課題は以下の通りである。

5.1 一貫性管理層を追加したCSCSDモデルによる調整支援

前章までに、ネットワークを介した共同作業を支援するための情報リポジトリの構成法について考察を重ねてきた。ところで、情報リポジトリに記録される中間生成物と決定事項およびそれらの状態は矛盾に満ちたものであり、矛盾の存在を前提とした上での共同作業推進の支援機構がミラノ工科大学との共同研究として開始されたところである。図18に示すように、CSCSDモデルの情報リポジトリ層とツール層の間に、情報リポジトリ内容間および現実世界との矛盾を検出し可能なら解消する層を設ける。これにより、一貫性管理に浪費され作業者の負担を低減することが期待される。

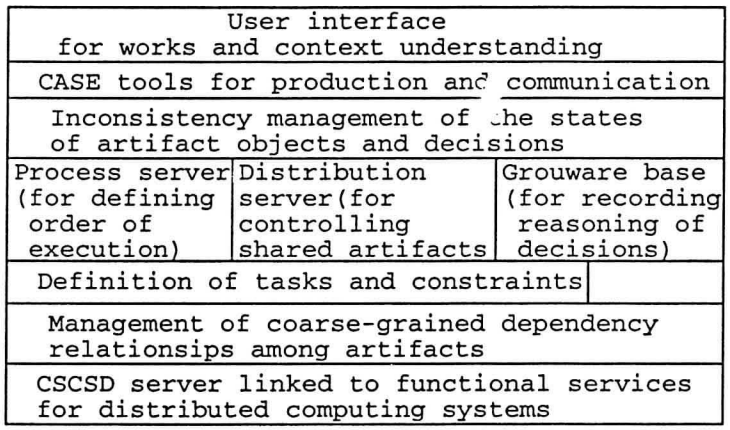


図 18: 一貫性管理層を追加したCSCSDモデル

すでに述べたように、協調的ソフトウェアプロセスの実行を制御する一つの確実な方法は中間生成物や製品の状態を観測することにある。一貫性管理層に置かれるコーディネータオブジェクトは、

- 振舞いに関する制約の下に互いに相互作用しながら、時間軸にそって状態を変化させていく並行動作実体群

の状態を監視する。

コーディネータオブジェクトは、状態を監視することにより、

- いくつかのオブジェクトにわたる状態間の一貫性 (外部制約) を維持することを試み、また、あるオブジェクトの内部状態間の一貫性 (内部制約) を維持することを試みる。
- 実体の動作履歴や関連と制約を利用して異常状態発生の犯人 (または故障オブジェクト) の検出を試みる。

コーディネータオブジェクトは以下のデータベースを利用する。

- 粗粒度レベルの依存関係
- 実体の状態遷移の履歴と実事象の履歴

われわれのモデリングにおいては、

1. 実体は、設計に関する中間生成物や成果物オブジェクト、および会話を通じてなされる決定事項であり、双方とも進捗状況を表現する状態遷移図を内蔵した自律オブジェクトとして形式化する予定である。許容される遷移や禁止される遷移および例外処理を表現するための時間に関する制約が、状態遷移図のそれぞれの遷移に割り当てられる (内部制約)。

2. 入出力依存関係グラフは、外部制約を表現するために使われる

3. 中間成果物オブジェクトの活動履歴は、実際に発生した事象/遷移の系列を版管理機構で管理する

コーディネータオブジェクトの主要な機能は以下の通りである。

- グループロールバック：故障オブジェクトが検出された時、それによる悪い副作用を可能なかぎり消去する。例えば、「中間生成物の集合に属する任意のAとBに対して、もしBがAから派生する時、Bの状態が" 完了" であるならば、Aの状態も" 完了" でなければならない」という粗粒度レベルの依存関係上で定義された制約に基づいて、関連する中間生成物の状態をすべてロールバックさせる。

- deviation[7]：いくつかのオブジェクトは、実事象の発生を待つことなくその事象の発生を仮定して遷移を進めることができる。実事象の発生後、もし矛盾が発生した場合は、それに関連する推移系列を検出し解消する。

5.2 オブジェクト管理システムの構成法

図 18の各層において、

- 中間成果物または製品間の入出力依存関係 (グラフ構造)
- 中間成果物のチーム構成員への割り当ての関係 (集合) とそれらを管理するオブジェクト群
- 上記集合をノードとする実行順序関係 (グラフ構造)
- 決定事項を文書化したものの間の依存関係 (階層グラフ構造)
- パラメータに応じてカスタマイズされる一群の同じ型のツール群 (集合)
- 上記各種構造間の対応関係
- 中間成果物や決定事項に貼り付けられる状態遷移図
- それらの状態遷移図の状態をモニタし制御するオブジェクト群

等の要素が存在する。それらを単一のスキーマを持つ単一のオブジェクト管理システムで管理するのは容易ではない。特に一部の変更に伴ないあちこち手直しする必要が生じる場合が予想される。複数のスキーマの提供とその管理を容易にするような、メタレベルアーキテクチャに基づくオブジェクト管理機構が必要である [15],[16]。

5.3 分散に依存する問題のより精密な把握

群舞と栞を中心とする自在プロトタイプの第1版を早期に完成し、「作業の独立性の保持」、「コミュニケーションの円滑化」に対する効果を評価するためのフィールドテストおよびプロトコル解析を実施する。例えば、

- 栞に関しては、「伝達・整合」、「決定」、「創造」の各討議の型毎にプロトコル解析実験を実施し、現在のグループウェアスキーマの有用性と改良されるべき問題点を把握・評価する必要がある

- 群舞に関しては、アクセスリストによる共有情報の変更管理にのみ限定せず、共有情報の管理に関して発生する問題点とその対応策を読み切るためのフィールドテストを実施する必要がある

現在の CSCSD モデルでは、共同作業における作業進捗の調整を、作業や会話を通じて生成される中間成果物や決定事項の状態の管理にもとづいて実施することを基本的な切り口としている。ところで、「状態の管理」は分散開発でなくとも対応すべき問題であり、その意味ではネットワークを介した共同作業を適確に支援するための必要条件ではあるが、それですべてではない。

「分散に依存する問題をより適確に把握することこそ最も肝要な事柄」である。分散開発支援の問題に関してブレークスルーをもたらすには、本稿の中心話題である「共同作業の状態に関する、各自の状況把握のズレをいかに調整し、必要な情報を共有させるか」という問題に関して一層の鋭い切り口が必要である。思考実験、プロトコル解析、現実世界の観測と分析等のさまざまな手段を通じて、多面的により良い切り口を定め、それを精密化していく必要がある。

例えば、本学篠田助教授は、阪神大震災のような大規模災害時に、通信衛星を使った臨時回線への切り替え、被災者の安否確認データベースの開発と全国規模の問い合せへの対応等を目的とするソフトウェアを、WIDE グループの仲間と電子メールのみを利用して開発した経験がある。彼の経験によると、このような本格的な分散開発において問題になったのは、モジュール間インタフェースの不整合の続発とそのような不整合を除去するためのコミュニケーションの遅延の拡大であった。一つのインタフェースの不整合を除去するのにやく半日かかり、状況を共有するのが困難であったとの感想がある。このような現象の背景にある問題点を深く洞察する必要がある。

5.4 分散開発時代に適合する方法論とプロセスモデルの定義

オブジェクト指向方法論、オブジェクト指向アーキテクチャを前提にして下記のような調査研究を実施することにより、分散システムの設計方法論、分散開発時代に適合するプロセスモデルの定義についての研究を今後開始する予定である。

- 「規模への対応」を考慮したソフトウェア・アーキテクチャに関する研究について、さらに詳細な調査研究を実施すること
- 「柔軟な」ソフトウェア・アーキテクチャの構成法とオブジェクト指向アーキテクチャに関する研究の現状を調査すること
- 上記 2 つの調査結果ををもとに、ソフトウェア・アーキテクチャのクラスを定義すること
- アーキテクチャ構成部品 (制御オブジェクト) の整備方針を検討すること
- システムへの統合法に関して考察を深めること
- その後で、いくつかの事例研究を実施し、オブジェクト指向ソフトウェア開発方法論におけるソフトウェア・アーキテクチャ設計の位置づけを明確かつ具体的にすること
- ソフトウェア・プロセスに関する研究に関して、ポリシーや制約の表現法を検討した研究成果をサベイすること
- また、決定事項の管理の問題に関して、管理の粒度、保護や安全性の度合等に関する現実世界の状況を調査・把握し、モデル化にあたっての具体的目標を設定すること

謝辞

本報告の基本構想はミラノ工科大学に文部省長期在外研究員として滞在した期間における著者の思索によって得られた。思索の過程における Carlo Ghezzi 教授による「一貫性管理」に関する示唆と、 Stefano Ceri 教授の「群舞」に関する有益なコメントに対して心より謝意を表する。

参考文献

[1] R.M.Adler and R.C.Paslay,"Design of Distributed Systems", Encyclopedia of Software Engineering, John Wiley and Sons.

[2] 荒木、岡村、佐伯、三浦、落水、篠田、海谷:「ネットワークを介した協調活動の実現にむけて」, 情報処理学会、サマー・ワークショップ・イン・立山報告集, pp.105-112, 1995.

[3] S.Bandinelli, A.Fuggetta, C.Ghezzi: "Software Process Model Evolution in the SPADE Environment",IEEE Trans. on SE, Vol.19 NO.12, pp.1128-1144, Dec.1993.

[4] Belkin, Croft "Information Filtering and Information Retrieval: Two Sides of the Same Coin?", CACM, 35, 12, pp29-38, 1992.

[5] F.Cattaneo,A.Fuggetta,L.Lavazza: "An experience in process assesment", Proc. of 17th ICSE, pp.115-121,1995.

[6] J.Conklin and M.Begeman:"gIBIS: A Hypertext Tool for Exploratory Policy Discussin", CSCW'88, pp.140-152, 1988.

[7] G.Cugola,E.Di Nitto Frank,C.Ghezzi: "How to deal With Deviations During Process Model Enactment", Proc. of ICSE 17, pp.265-273, April 1995.

[8] G.Cugola,E.Di Nitto Frank,A.Fuggetta,C.Ghezzi: "A Framework for Formalizing Inconsistencies and Deviations in Human-Centered Systems", (submitted to TOSEM).

[9] U. Dayal, M. Hsu and R. Ladin: "A Transaction Model for Long-Running Activities", Proc. of 17th International Conference of VLDB, pp.113-122,1991.

[10] C.A. Ellis and S.J.Gibbs: "Concurrency Control in Groupware Systems", Proc. of the ACM SIGMOD'89, pp.399-407, May 1989.

[11] C.A. Ellis and S.J.Gibbs and G.L.Rein: "Groupware: Some Issues and Experiences", Comm. of the ACM, Vol.34, NO.1, pp.38-58, Jan. 1991.

[12] Clarence Ellis, Jacques Wainer: "A Conceptual Model of Groupware", Proc. of CSCW 94, pp.79-88, 1994.

[13] Martha S Feldman: "Constraints on Communication and Electronic Mail", Proc. of CSCW 86, pp.73-90, 1986.

[14] Foltz, Duans "Personalized Information Deliverty: An Analysis of Information Filtering Methods", CACM, 35, 12, pp51-60, 1992.

[15] 藤枝, 落水,"メタレベルアーキテクチャを利用したオブジェクト管理システムの構成法について", 情報処理学会 ソフトウェア工学研究会, Vol.94 No.99, pp.41-48, 1994.

[16] 藤枝, 落水, "メタレベル・アーキテクチャを利用したオブジェクト管理システムのバージョン管理機構の構成法", 日本ソフトウェア科学会 第12回大会論文集, pp.109-112, 1995.

[17] David Garlan, "Research Directions in Software Architecture", ACM Computing Surveys, Vol.27, No.2, pp.257-261,June 1995.

[18] D.Goldberg, D.Nichols, B.M.Oki, D.Terry "Using Collaborative Filtering to Weave an Information Tapestry" Communications of the ACM, Vol 35, No.12, pp61-70 (1992).

[19] Sandra Heiler: "Semantic Interoperability", ACM Computing Surveys, Vol.27, No.2, pp.271-273,June 1995.

[20] 堀, 落水:「ソフトウェア開発における自己反映オブジェクト指向モデルに基づく共有情報の管理法」, コンピュータ・ソフトウェア,Vol.13, NO.1, pp.37-54,1996.

[21] M.Hori, Y.Shinoda, K,Ochimizu: "Shared Data Management Mechanism for Distributed Software Development Based on a Reflective Object-Oriented Model", LNCS 1080, Advanced Information Systems Engineering, pp.362-382, 1996.

[22] D.Jennings:"On the Definition and Desirability of Autonomous User Agents in CSCW", CSCW and Artificial Intelligence, Springer Verlag, 1994.

[23] 門脇, 落水:「ソフトウェア・プロセス実行における系統的コミュニケーション支援の一方式」, Jaist Research Report, IS-RR-97-0009S, ISSN 0918-7553, March 1997.

[24] C.Kadowaki, K.Ochimizu: "Recording the Progress and the Reasoning of Deliberations",(under submission).

[25] 片山:「ソフトウェアプロセスとその研究課題」, 日本ソフトウェア科学会第 11 会大会論文集, pp.433-436t, October 1994.

[26] 川上, 川瀬, 小寺, 鈴木, 萩原, 橋本, 落水:「ネットワークを介した協調活動の支援環境」, Jaist Research Report, IS-RR-96-0012S, 1996.

[27] 川瀬、篠田:「協調フィルタリングに関する研究動向」、研究室内部資料, 1996.

[28] K.Kishida, T.Katayama, M.Matsuo, I.Miyamoto, K.Ochimizu, N.Saito, J.H.Sayler, K.Torii, L.G. Williams: "SDA: A Novel Approach to Software Environment Design and Construction", Proc. of 10th ICSE, pp.69-78, 1988.

[29] 小島:「建築」, イタリア, 新潮社.

[30] 近野, 門脇, 落水:「グループウェアベース「栞」を用いた電子会議内容の進捗把握と文書化の支援」, 情報処理学会ソフトウェア工学研究会資料、107-12,1996.

[31] 倉谷、門脇、西山、落水:構造化分析の事例研究 (生産管理システム)、Seamail Vol.8 NO. 8-9, pp.3-40, 1994.

[32] 倉谷、東田、藤枝、鈴木、落水:オブジェクト指向分析の事例研究 (生産管理システム)、Seamail Vol.8 NO. 8-9, pp.41-76, 1994.

[33] Malone, Grant, "Intelligent Information Sharing Systems" CACM, 30, 5, pp390-402 (1987).

[34] D.Maltz, K.Ehrlich "Pointing The Way: Active Collaborative Filtering", Proc. of CHI '95, pp202-209 (1995).

[35] D.Mandrioli, A.Morzenti, P.San Pietro, E.Crivelli: "Specification and verification of real-time systems in a logic framework: the TRIO environment", , Vol., No., pp.-,.

[36] Frank Manola: "Interoperability issues in Large-Scale Distributed Object Systems", ACM Computing Surveys, Vol.27, No.2, pp.268-270,June 1995.

[37] 森田、篠田 「情報に対する行動の分析と情報フィルタリングへの適用に関する研究」, JAIST 修士論文, 1994.

[38] 村越, 海谷, 落水,"ネットワークを介した共同作業における阻害要因の分析", 日本ソフトウェア科学会第 12 回大会論文集, pp.233-236, 1995.

[39] H.Murakoshi, H.Kaiya, K.Ochimizu: "An Analysis of Obstruction in Cooperative Work over a Computer Network", Proc. of CICS'95, pp.98-103, Oct. 1995.

[40] 村越, 海谷, 落水, 佐伯,"非同期型のコミュニケーションを用いた共同作業における阻害要因の分析", 情報処理学会ソフトウェア工学研究会, 109-4, pp.25-32, 1996.

[41] 村越, 海谷, 落水, "共同ソフトウェア開発における非同期型コミュニケーションの生産物への影響", 近代科学社, ソフトウェア工学の基礎 III(日本ソフトウェア科学会 FOSE'96 報告集), pp.166-169, 1996.

[42] A.Ohki, K.Ochimizu: "Process Programming with Prolog", Proc. of 4th ISPW, pp.118-121, 1988.

[43] 落水、東田:「オブジェクト・モデリング」, ジャストシステム, 1995.

[44] 落水: 「ソフトウェア工学実践の基礎–分析・設計・プログラミング」, 日科技連出版

[45] 落水:「ソフトウェア・プロセスに関する研究の概要」, 情報処理,Vol.36, NO.5, pp.379-391, 1995.

[46] K.Ochimizu,T.Yamaguchi: "A Process Oriented Architecture with Knowledge Acquisition and Refinement Mechanisms on Software Process", Proc. of 6th ISPW, pp.145-147, 1991.

[47] 落水:「ソフトウェア・レポジトリ」, 情報処理,Vol.35, NO.2, pp.140-149, 1994.

[48] K.Ochimizu and C.Kadowaki:"A Framework of a Support Environment for Cooperative works over Distributed Computing System based on a Decision Management",Jaist Research Report, IS-97-0013S, ISSN 0918-7553, March 1997.

[49] 落水, 門脇, 藤枝, 堀:「ソフトウェア分散開発支援環境「自在」のアーキテクチャ設計」, 電子情報通信学会 ソフトウェアサイエンス研究会資料, SS94-18, 1994.

[50] K.Ochimizu,"Constructing a Support Environment for Cooperative works over a Computer Network by Integrating Software Process Enactment Support and Communication Support",Jaist Research Report, IS-97-0012S, ISSN 0918-7553, March 1997.

[51] D.Pascot 著、落水訳: 「C/S データベース設計入門」, 日経ＢＰ社,1997.

[52] P.Resnick, N.Iacovou, M.Suchak, P.Bergstorm, J.Riedl "GroupLens: An Open Architecture for Collaborative Filtering of NetNews", Proc. of ACM 1994 Conference on CSCW, pp175-186(1994).

[53] U.Shardananda, P.Maes "Social Information Filtering: Algorithm for Automating "Word of Mouth "", Proc. of CHI '95, pp210-217 (1995).

[54] H.Tanaka, K.Araki, Y.Masuda:"Relationship between Interpersonal Communication and Effects of Communication Media", IPSJ SIG on GW 93-GW-4, pp.53-60, 1993(in Japanese).

[55] 山口、樽松、下津、中尾、落水: "事例に基づく推論とモデル推論の統合に基づく知識獲得支援システム（２）-ソフトウェアプロセス知識の獲得-", 人工知能学会誌, Vol.11 No.4, pp.97-103, 1996.

[56] Gio Wiederhold: "Mediation in Information Systems", ACM Computing Surveys, Vol.27, No.2, pp.265-267,June 1995.

「オブジェクト指向技術の実態調査」結果報告書

中小路　久美代（SRA）

松原　順（奈良先端科学技術大学院大学）

1. はじめに

ソフトウェア開発に要する期間の短縮，再利用の増加，品質の向上など，オブジェクト指向技術を用いることの利点が多く挙げられている．しかし，実際の開発現場でどれだけオブジェクト指向技術が用いられているのか，本当にオブジェクト指向技術は有効か否か，そしてその理由についてはほとんど知られていない．そこで，オブジェクト指向技術の実態について理解を深めるため，調査を行った．

本報告は1994年6月，1996年11月の2回にわたり実施したオブジェクト指向技術の実態調査の結果をまとめたものである．第一回実態調査は，日本の企業，ワークショップ，奈良先端科学技術大学院大学においてアンケート形式で行われ，48名から回答があった．第二回実態調査は，第一回実態調査で対象となった企業5社に対して行われ，5社38名から回答があった．

以下，はじめに実態調査の結果報告，分析を行い，次に他の実態調査例との比較を行う．そして，最後に将来への展望を述べる．

2. 実態調査の結果

2.1 第一回実態調査

第一回実態調査は1994年6月，日本の企業，ワークショップ，奈良先端科学技術大学院大学において，アンケート形式で行われ，48名から回答があった．調査項目は3つのパートに分かれており，それぞれ，ソフトウェア開発に関する個人的背景，所属組織の背景，個人の関わったプロジェクトに関する背景についての質問で構成されている．以下，実態調査の結果を，各パートごとに報告する．

(1) ソフトウェアに関する個人的背景

このパートでは，回答者のソフトウェアに関する教育背景と実務経験について質問が行われた．その結果，ソフトウェア開発経験が4年以上あるという回答が約8割を占めた．また，「オブジェクト指向技術という言葉を知っているか」という質問にたいして，全員が知っていると答えた．以下，主な項目について詳述する．

- 通常使っている分析・設計技術

オブジェクト指向関連ではエンティティ・リレーションダイアグラムとオブジェクトモデリング技法の使用が多くみられた．（図1参照）

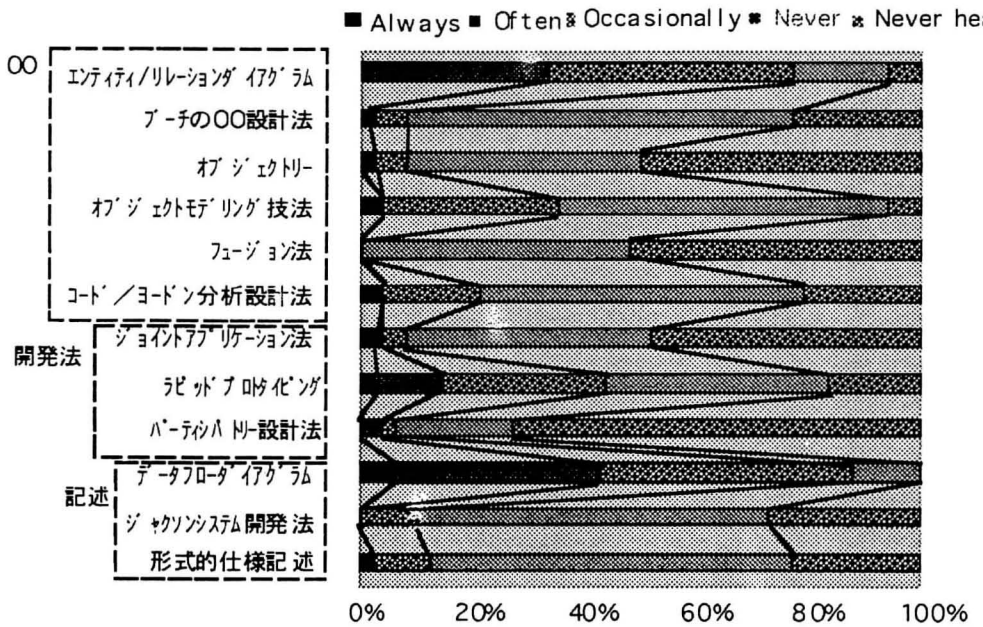


図1：分析・設計技術の利用状況

- 使用したことのあるプログラミング言語

C,COBOL,Fortran,Assemblerの熟練度が高かった.

- オブジェクト指向技術に関する専門度

約半数はオブジェクト指向技術に関わったことがあり，分析・設計・プログラミング・開発による差はなかった. (図2参照)

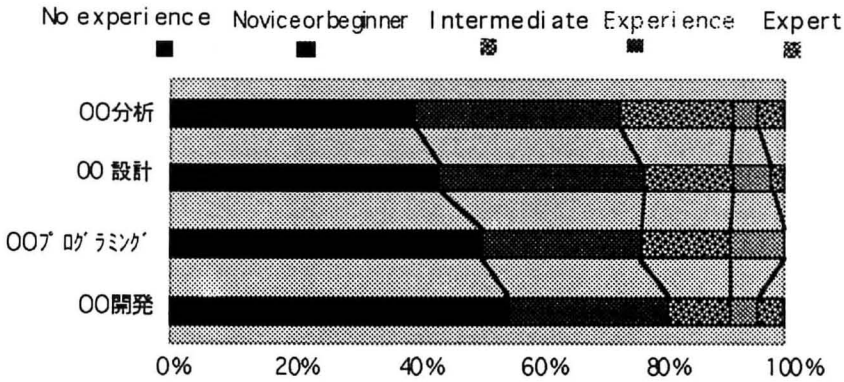


図2：オブジェクト指向技術に関する専門度

1 オブジェクト指向技術を用いたプロジェクトの経験

オブジェクト指向技術を用いたプロジェクトに関わった経験がない人が約半数を占めた.

1 オブジェクト指向技術の重要性

ソフトウェア開発のライフサイクルのフェーズにおいてオブジェクト指向技術が重要だと考えられているフェーズは要求分析・設計・プログラミングであるという傾向があり，大多数の人がオブジェクト指向技術を持つことは重要であると考えている.

(2) 所属組織の背景

このパートでは，回答者の所属する組織についての質問が行われた．組織の業種では ソフトウェア開発関係であるという回答が一番多く，約7割を占めた. 以下，主な項目について詳述する.

- 雇用人数

全体で100-1000人という回答が一番多く，約6割であった. (図3参照)

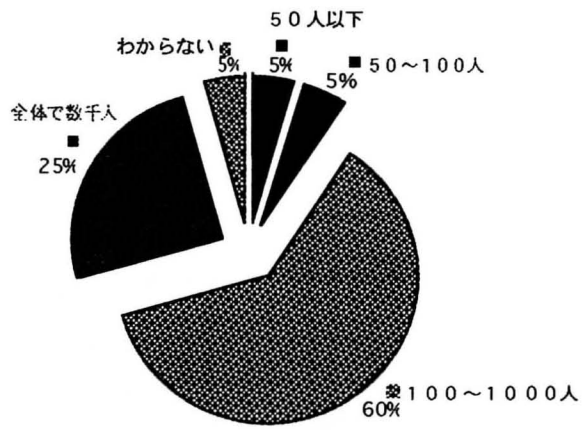


図3：組織の雇用人数

- 所属組織に在籍するオブジェクト指向技術専門家の人数

1人もいないという回答は4割，数人以下という回答が約5割で,たくさんいると答えたのは1割に過ぎない(図4参照)

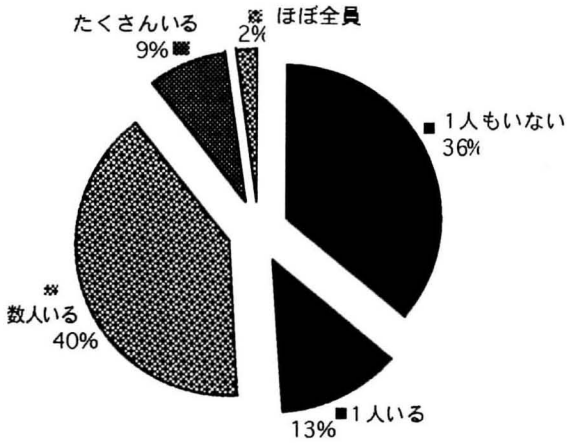


図４：オブジェクト指向技術専門家数

- 組織の成熟度

発展中であるという回答と成熟したという回答が大体半分ずつであった．

- 企業文化

所属企業の雰囲気は親しみやすくのんびりしている傾向にあった．

(3) プロジェクトの背景

このパートでは回答者が最近関わった３つのプロジェクトについての質問が行われ，９３のプロジェクトについて回答があった．以下，主な項目について詳述する．

- プロジェクトにおける役割

プロジェクトマネージメントを行なったという回答が約７割あり，開発の下位工程にいくに従って減少する傾向にあった．

- プロジェクトの開発言語

プロジェクトの開発言語はＣ言語が圧倒的に多く　約半数を占めていた．ついでCOBOL,Assemblerが多く，それぞれ約１.5割であった．

- プロジェクトのアプリケーションドメイン

半数以上をビジネスアプリケーションが占めた．

- プロジェクトメンバー数

2-5人で行なわれているプロジェクトが約３割と最も多く，6-10人，11-20人で行なわれているプロジェクトが各々約２割であった．（図５参照）

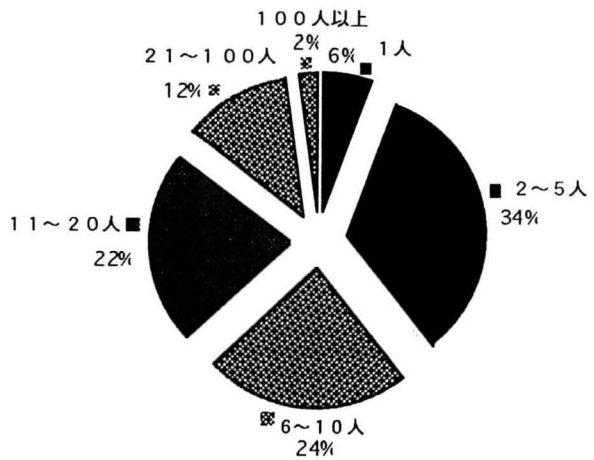


図５：プロジェクトメンバー数

- プロジェクトの開発期間

半年以上1年未満，2年以上　開発にかかっているプロジェクトが各々約3割であった．

- CASEtoolの使用状況

約7割のプロジェクトでCASEtool が未使用であった．残りのプロジェクトでは分析・設計に CASEtool を使用したとい回答が 一番多く使用されたCASEtoolの 一例を挙げるとSA/SD ER editorなどであった．（図6参照）

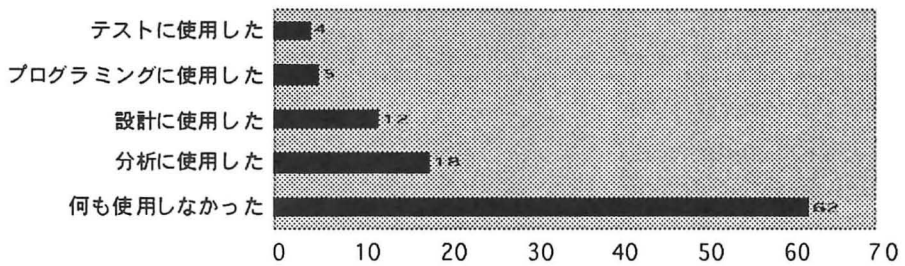


図6：CASEtoolの使用状況

- オブジェクト指向技術の利用状況

約7割のプロジェクトでオブジェクト指向技術は利用されていなかった．残りの33/93のプロジェクトでは分析・設計・プログラミングにオブジェクト指向技術が用いられていた．（図7参照）

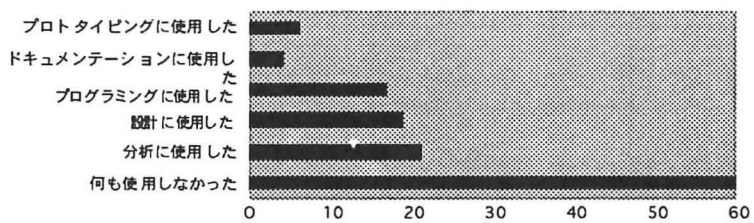


図7：オブジェクト指向技術の利用状況

- オブジェクト指向技術を用いたプロジェクトの結果

オブジェクト指向技術を用いたプロジェクトが成功したという回答が約7割を占め，不成功であったというプロジェクトは3/33であった．（図8参照）

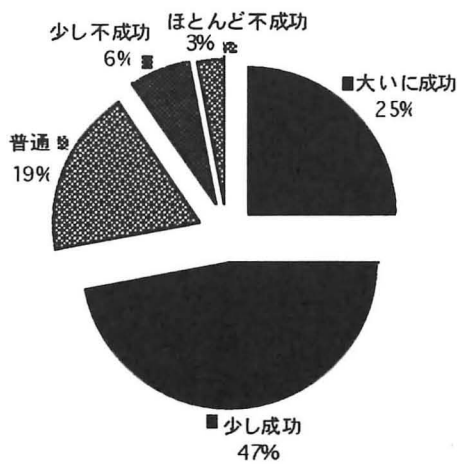


図8：オブジェクト指向技術を用いたプロジェクトの結果

- プロジェクト成功の理由

オブジェクト指向技術を用いたプロジェクトの成功理由として，「プロジェクトメンバーの士気が上がった」「システムの保守性や理解度が上がった」という回答が多くみられた．（図9参照）

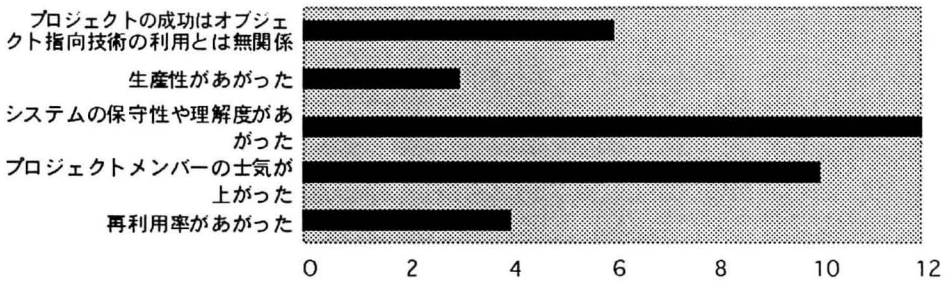


図9：オブジェクト指向技術を用いたプロジェクト成功の理由

- プロジェクト失敗の理由

オブジェクト指向技術を用いたプロジェクトの失敗の理由として、「オブジェクト指向技術に対する不理解」「オブジェクト指向技術の運用面における失敗」などが挙げられた.

2.2 第二回実態調査

第二回実態調査は、第一回実態調査で対象となった日本の企業5社を対象として1996年11月に行われ、5社38名から回答があった. 質問は主にオブジェクト指向技術に関する教育と、個人が関わったプロジェクトに関するものであった. 個人が関わったプロジェクトに関する質問は、回答者が最近関わった3つのプロジェクトについて行われ、81のプロジェクトについて回答があった. 以下、主な項目について詳述する.

- オブジェクト指向技術に関する専門度

分析・設計よりもプログラミングの熟練度のほうが全体的に高かった.

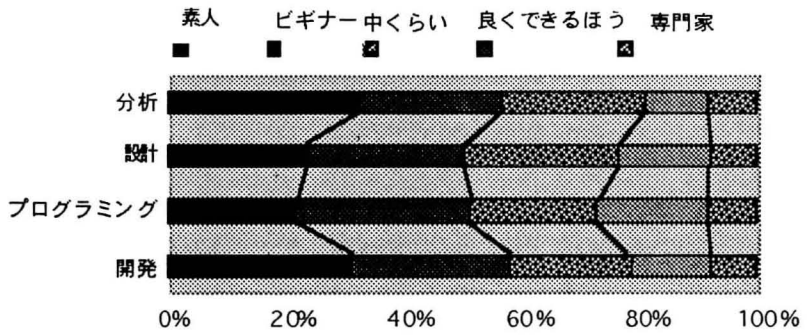


図10：オブジェクト指向技術に関する専門度

- CASEtoolの使用状況

7割強のプロジェクトでCASEtoolは使用されなかった.

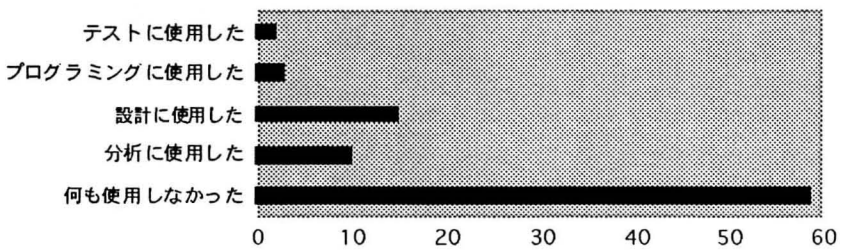


図11： CASEtoolの使用状況

- オブジェクト指向技術の使用状況

81プロジェクトのうち約2/3の54プロジェクトでオブジェクト指向技術が使用されており、プログラミングに使用したという回答が一番多かった.

セミナーの期間についての集計結果を示す．（図 10 参照）

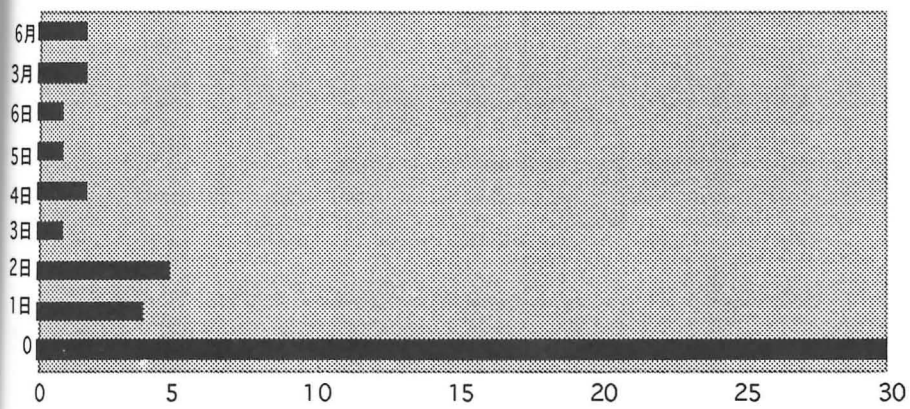


図 10：セミナーの期間

9 あなたのオブジェクト指向技術（オブジェクト指向分析、設計、プログラミング、開発）に関する専門度をそれぞれ教えてください．

oo 分析	oo 設計	oo プログラミング	oo 開発
[19] 全くの素人	[21] 全くの素人	[24] 全くの素人	[26] 全くの素人
[16] ビギナー	[16] ビギナー	[12] ビギナー	[12] ビギナー
[9] 中くらい	[7] 中くらい	[7] 中くらい	[5] 中くらい
[2] 良くできる方	[3] 良くできる方	[4] 良くできる方	[2] 良くできる方
[2] 全くの専門家	[1] 全くの専門家	[0] 全くの専門家	[2] 全くの専門家

10． あなたの所属している組織の中に、オブジェクト指向技術の専門家（もしくは専門家だと思われる人）がどれくらいいらっしゃいますか？また、その組織は全体で何人くらいですか？

[17] 一人もいない
[6] 一人いる
[19] 数人いる
[4] たくさんいる
[1] ほとんど全員がそうだ

11． あなたは、現在オブジェクト指向技術を用いた開発または保守プロジェクトに関わっていますか？

[22] 全く関わっていない いたことがない
[14] 一つだけオブジェクト指向関連のプロジェクトに関わっている いたことがある
[9] 2、3のオブジェクト指向関連のプロジェクトに関わっている いたことがある
[2] 多数のオブジェクト指向関連のプロジェクトに関わっている いたことがある
[1] これまで関わっているすべてのプロジェクトが、オブジェクト指向関連である

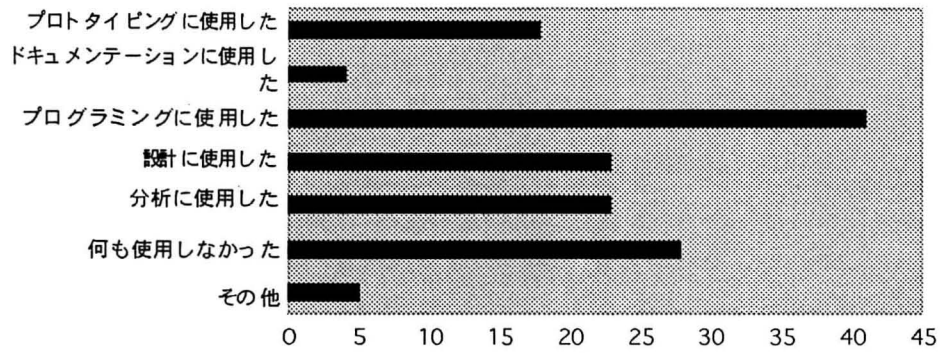


図 12：オブジェクト指向技術の使用状況

- オブジェクト指向技術の使用を決定した人

約半数のプロジェクトで回答者以外の上司がオブジェクト指向技術の使用を決定していた．また，その他の内訳は「顧客の要望」がほとんどであった．

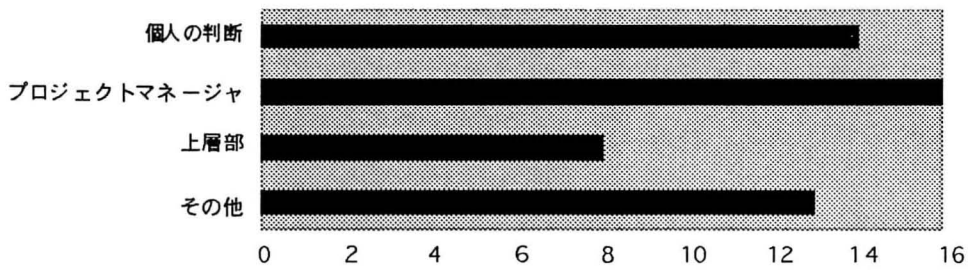


図 13：オブジェクト指向技術使用の決定者

- オブジェクト指向技術の使用目的

9割近くのプロジェクトでオブジェクト指向技術使用による実質的な効果を期待していることがわかった

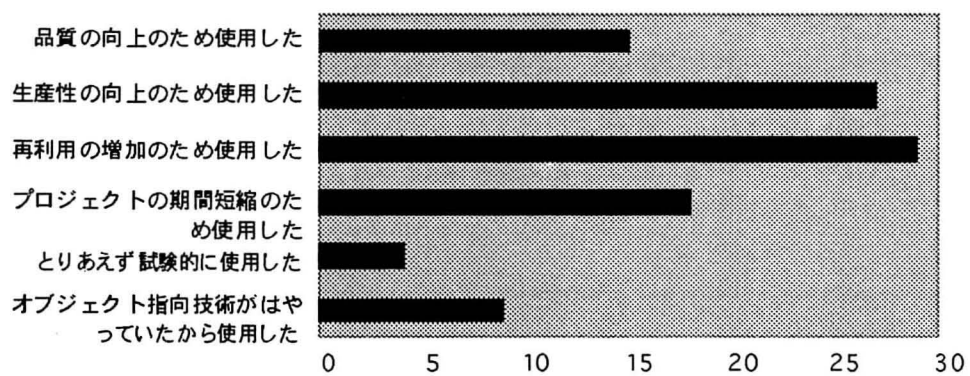


図 14：オブジェクト指向技術の使用目的

- オブジェクト指向技術使用の効果

9割近くのプロジェクトで何らかの効果があがった．その他の内訳は，プログラミングスタイルの統一化・部品化による生産性の向上・開発期間の短縮が 3，結果不明が 8 などであった．

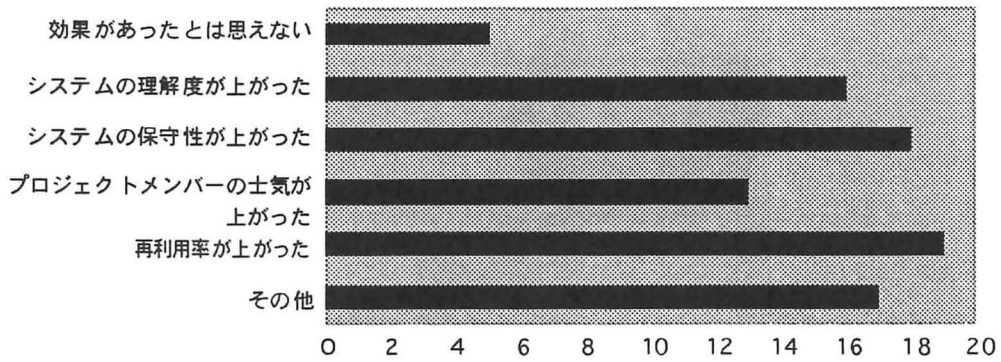


図 15：オブジェクト指向技術使用の効果

- オブジェクト指向技術の使用効果が見られなかった原因

プロジェクトサイズ・ドメインにはあまり関係なく，オブジェクト指向技術に対する不理解を挙げたものが圧倒的に多かった．

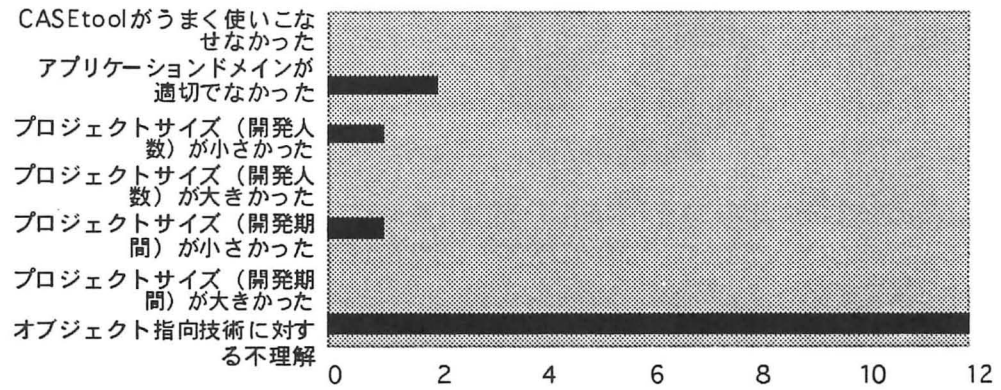


図 16：オブジェクト指向技術使用による効果がみられなかった原因

3. 実態調査の分析

3.1 第一回実態調査

第一回実態調査ではオブジェクト指向技術は良く知られていることがわかった．これは，「オブジェクト指向技術という言葉をしっているか？」という質問にたいして全員が「知っている」と答えたことからわかる．また，オブジェクト指向技術を用いたプロジェクトでは，ほとんどが，オブジェクト指向技術を使用したことによる何らかの効果が上がっている（図 8，図 9 参照）ことから，オブジェクト指向技術は有効であることもわかった．しかし，オブジェクト指向技術は有効な技術であるにもかかわらず，実際現場でオブジェクト指向技術が用いられたプロジェクトは 93 プロジェクト中 33 プロジェクトであった．

期待したほどオブジェクト指向技術が使用されていない原因として考えられるのは，オブジェクト指向技術の熟練者があまりいないという事実である．個人のオブジェクト指向技術に関する熟練度（図 2 参照），企業内のオブジェクト指向技術の熟練者の数（図 4 参照）をみても，オブジェクト指向技術の熟練者が少ないことは明らかである．オブジェクト指向技術を用いたプロジェクトが失敗した原因にも，オブジェクト指向技術に対する不理解が挙げられている．

この結果，オブジェクト指向技術に関する教育が不足しているということがわかった．

3.2 第二回実態調査

第二回実態調査では第一回実態調査に比べてオブジェクト指向技術の熟練度が上がっていることがわかった．その傾向は特に，プログラミングにおいて顕著である（図 10 参照）．また，オブジェクト指向技術を用いたプロジェクトの割合も 81 プロジェクト中 54 プロジェクト(67%)と第一回実態調査の時の 93 プロジェクト中 33 プロジェクト(35%)という割合に比べて上昇していることがわかった（図 7，図 12 参照）．この原因として考えられるのは，オブジェクト指向技術に関する教育が前回の調査のときよりも熱心に行われたのではないかということである．確かに，新たに設定した自主的な学習に関する質問では，38 人中 28 人(74%)が書籍などによる学習をしていることがわかったが，セミナー参加に関しては，第一回実態調査においてセミナーに参加した人数は 48 人中 18 人(38%)，第二回実態調査においてセミナーに参加した人数は 38 人中 18 人(47%)と，オブジェクト指向技術を組織的に使用する割合が増える主要因であるとは考えにくい．そこで，組織別に，オブジェクト指向技術使用の割合をみると，組織により片寄りがあることが判明した．（図 17 参照）

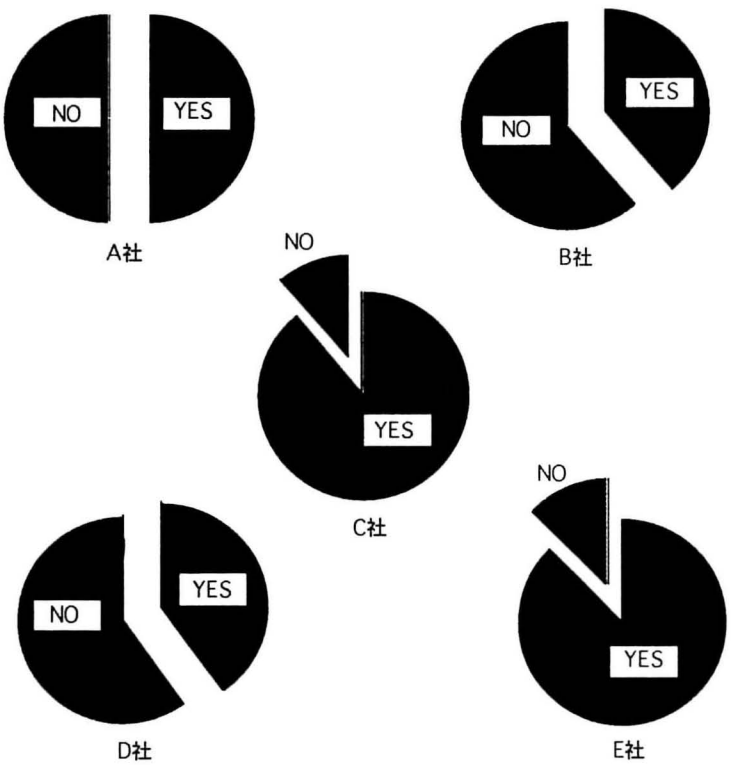


図 17：組織別オブジェクト指向技術の使用状況

C 社、E 社ではオブジェクト指向技術を用いたプロジェクトが用いなかったプロジェクトより明らかに多いが、B 社、D 社では、オブジェクト指向技術を用いたプロジェクトの割合が少ないことがわかる。組織によるオブジェクト指向技術導入の差の原因を探ると、オブジェクト指向技術が積極的に使用されている組織では、上層部がオブジェクト指向技術の導入を積極的に決定しているという事実があることがわかった。（図 18 参照）

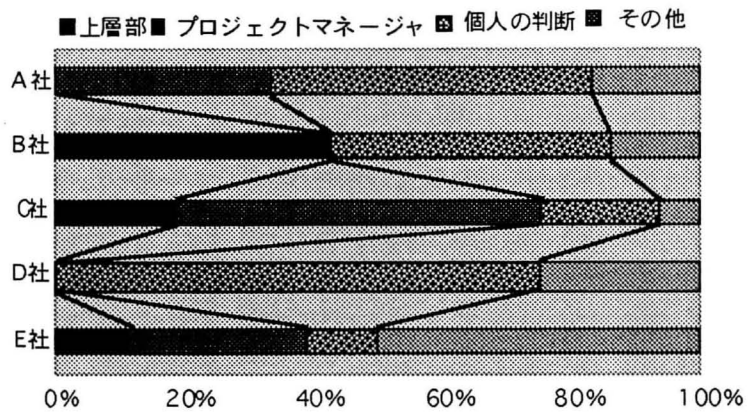


図 18：組織別オブジェクト指向技術使用決定者

これは、オブジェクト指向技術導入の 1 つのパターンであると考えられる。

4.まとめ

今回の調査で、オブジェクト指向技術がその効果を期待されながらなぜ導入が進まないのかといったことの原因をさぐってみた。その結果、オブジェクト指向技術に関する教育が現場の要求を満たしていないことがわかった。また、オブジェクト指向技術導入では、オブジェクト指向技術に関する教育の徹底と上層部のオブジェクト指向技術導入への積極的な姿勢が重要であることがわかった。

将来的には、この調査結果やさらなる調査をもとに、オブジェクト指向技術をスムーズに導入するための要求にはどういったものがあるか、例えばどの程度の教育が必要なのか、などといったことを明らかにしたいと考えている。

付録1： オブジェクト指向技術の実態調査1 アンケート及び集計結果

([]内に回答者数，総数48名)

パート1：あなたのソフトウェア工学に関する教育背景と実務経験についてお尋ねします

1. あなたがソフトウェア開発者 プログラマー アナリスト 設計者（一つお選び下さい）になってからの期間をお答えください.

[0] 1年未満

[6] 1～3年

[11] 4～7年

[12] 7～10年

[16] 10年以上

2. 現在の職位をお答えください. (省略)

3. 以下に挙げた分析技術、設計技術をお使いになったことがありますか？（それぞれの技術について、使ったことがない/時々 よく/常に使う その技術を聞いたことがない、の一つに印をして下さい. ）

	使ったことがない	時々	しばしば	常時	聞いたことがない
ジャクソンシステム開発法（JSD)	[30]	[6]	[0]	[0]	[13]
ジョイントアプリケーション開発法（JAD)	[21]	[2]	[1]	[1]	[22]
データフローダイアグラム（DFD)	[6]	[22]	[16]	[3]	[0]
エンティティ/リレーションダイアグラム (ERD)	[6]	[21]	[16]	[0]	[3]
形式的仕様記述（e.g. Z言語)	[31]	[1]	[0]	[5]	[11]
コード/ヨードンOO分析設計法	[27]	[8]	[0]	[2]	[10]
フュージョン法	[23]	[0]	[0]	[0]	[25]
オブジェクトモデリング技術（OMT)	[28]	[15]	[0]	[2]	[3]
オブジェクトリー	[20]	[3]	[1]	[0]	[24]
ブーチのOO設計法	[33]	[3]	[1]	[0]	[11]
ラピッドプロトタイピング	[19]	[14]	[6]	[1]	[8]
パーティシパトリー（ユーザ参画型）設計法	[10]	[1]	[2]	[0]	[34]

そのほか（具体的に手短にご説明ください）：

4.次に挙げたプログラミング言語の中で、使ったことのあるものがありますか？（それぞれの言語について、あなた自身がその言語を使う際にどう感じるか、ビギナー まあまあ使える よく使える まったくのプロ 使ったことがない、の一つに印をお付けください. ）

	ビギナー	まあまあ使える	よく使える	まったくのプロ	使ったことがない
C	[11]	[14]	[13]	[7]	[3]

12. オブジェクト指向技術を用いることが最も大切なのはソフトウェア開発のどのライフサイクルフェーズだと思われますか？（当てはまるものいくつにでも印をつけてください．）

[31] 要求定義と分析
[33] 設計
[20] プログラミング
[3] テスト
[5] ドキュメンテーション
[4] そのほか（説明してください）

13. オブジェクト指向技術に関する技術を持つことは、大切だと思いますか？

[0] 今後の将来のためには別に重要ではない
[8] まあ大切だが、それほど致命的ではない
[18] 技術を持っていれば将来役に立つであろう．
[18] オブジェクト指向を理解することは絶対に不可欠である
[3] よくわからない

パート２：あなたの所属する組織に関する質問です．

14. あなたの所属する組織は？

[3] 金融関係
[4] 製造関係
[34] ソフトウェア開発関係
[1] コンサルティングビジネス
[1] 販売関係
[6] そのほか（ご説明ください）

15. 創業（設立）以来何年くらいになりますか？

16. 雇用人数は？

[2] 全体で５０人以下
[2] 全体で５０−１００人
[27] 全体で１００−１０００人
[11] 全体で数千人
[2] わからない

17. あなたの組織の成熟度は？（当てはまるものすべてに印をつけてください）

[9] まだ果立ったばかり
[9] 研究／開発のステージ

[5] 急速に発展しつつある

[6] 成熟していて、伸びは穏やかである

[17] 現在再構築の際中である

[6] 少し縮小の傾向にある

[4] わからない

18. あなたの組織の企業文化をどのように表わされますか？下記の点についてそれぞれ、賛同するかしないかを記述して下さい.

企業文化は、フレンドリーで、リラックスした感じである

全く反対 少し賛成しない 関心がない 少しそう思う 全く賛成する

1 [4] 2 [8] 3 [3] 4 [18] 5 [11]

企業文化は、とても競争的で、生き馬の目を抜くようである

全く反対 少し賛成しない 関心がない 少しそう思う 全く賛成する

1[17] 2 [13] 3 [6] 4 [7] 5[1]

企業文化は、とても競争的だが、助け合いの精神がある

全く反対 少し賛成しない 関心がない 少しそう思う 全く賛成する

1[9] 2[13] 3 [9] 4 [11] 5 [1]

企業文化は、革新的で、新しいアイデアに対して寛容である

全く反対 少し賛成しない 関心がない 少しそう思う 全く賛成する

1 [8] 2 [13] 3 [3] 4[17] 5 [3]

企業文化はとても保守的で、官僚的である.

全く反対 少し賛成しない 関心がない 少しそう思う 全く賛成する

1 [9] 2 [10] 3 [3] 4 [12] 5 [10]

そのほかにあなたの働く企業の雰囲気を表わすことばがあれば、記述して下さい.

パート3：プロジェクトの背景（以下，集計の都合上プロジェクト1～3をまとめて集計する）

まずあなたがこれまでに関わった3つのプロジェクトを思い出してください．そのそれぞれについて、以下の質問にお答えください．もしもオブジェクト指向技術関連のプロジェクトがありましたら、特にそれを含んでください．もしも思いつくプロジェクトが3つに満たない場合は、思い出す限りのプロジェクトに関して質問にお答えください.

19. プロジェクトについて簡単にご説明ください．（省略）

20. それぞれのプロジェクトに関わった期間はどれくらいですか？（開始年月と終了年月をお答えください）

*プロジェクトの開始年月の集計を示す．期間についてはQ28参照

～`85年 [9] `90年 [12]

`86年 [2] `91年 [9]

`87年 [2] `92年 [19]

`88年 [6] `93年 [13]
`89年 [8] `94年 [4]

21. それぞれのプロジェクトについて、あなたの果たした役割は？（当てはまるものすべてに印を付けてください。）

[56] プロジェクト
マネージメント
[47] 要求定義
[49] 分析
[54] 設計
[38] コーディング
[43] テスティング
[39] ドキュメンテーション
[28] 保守
[3] そのほか（教育，プレビュー ）

22. それぞれのプロジェクトに関して、使用したプログラミング言語は？

[41] C
[15] COBOL
[8] C++
[4] PL/1
[5] Fortran
[5] Smalltalk
[4] Lisp
[0] Prolog
[0] Pascal
[22] そのほか(Assembler[11], Basic[2], ObjectiveC[2], etc..)

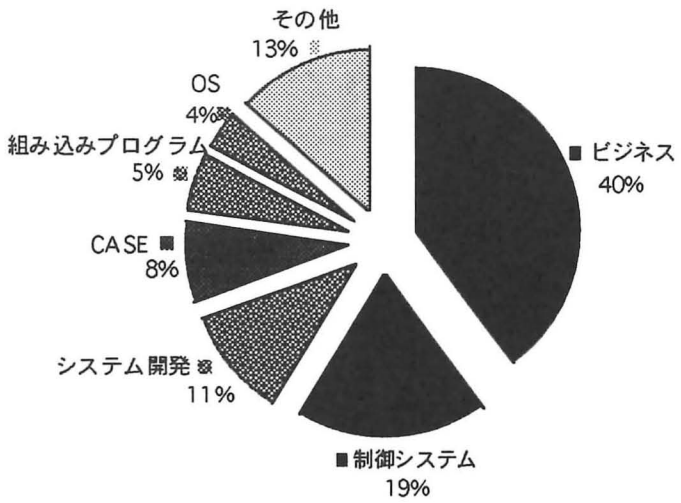
23. それぞれのプロジェクトに関して、ターゲットハードウェアは？
（最終成果システムをインプリメントしたハードウェア）
24. それぞれのプロジェクトに関して、開発ハードウェアは？
（ソフトウェアを実際に開発したしたハードウェア）

[20] PC(IBM compatible)
[22] Mainframe
[7] MacIntosh
[39] Workstation
[1] HP

[1] Dec-station

[23] そのほか(PC-98[13], Fujitu[2] etc...)

25. それぞれのプロジェクトに関して、アプリケーションドメインを説明してください。（たとえば、ビジネスアプリケーション、工場制御システム、帳票アプリケーション、ワードプロセッサーなど）



26.それぞれのプロジェクトを最もよく表わしているのは次のどれですか？（当てはまるものすべてに印を付けてください。）

[30] 前例のない新規のシステム開発

[24] 既存のシステムにとって変わる新規システム開発

[26] 既存システムのエンハンスメント（改造　拡張）

[2] 既存システムの保守

[11] 外部顧客のためのカスタムシステム

[14] 企業内部で用いるためのカスタムシステム

[9] 新規のパッケージソフトウェア

[4] パッケージソフトウェアのバージョンアップ

C++	[18]	[9]	[4]	[0]	[17]
COBOL	[8]	[8]	[11]	[4]	[18]
PL/1	[8]	[8]	[2]	[1]	[29]
Fortran	[15]	[15]	[13]	[3]	[2]
Smalltalk	[18]	[4]	[0]	[0]	[6]
LISP	[16]	[5]	[1]	[1]	[24]
APL	[6]	[1]	[0]	[0]	[41]
Assembler	[12]	[15]	[8]	[6]	[8]
Eiffel	[6]	[0]	[0]	[0]	[42]
Prolog	[8]	[4]	[1]	[0]	[35]

そのほか（具体的に手短にご説明ください）：

5. あなたの計算機もしくは情報工学に関する教育の背景についてお尋ねします．以下のカテゴリーに最も近いですか？ 一つ選んでください．

[7] 計算機または情報工学の大学院（修士 博士）修了

[8] 計算機または情報工学の学部（学士）修了

[6] 計算機または情報工学以外の修了で、計算機に関する授業は何もとったことがない

[3] 大学修了はしていないが、計算機に関する授業を、生涯教育プログラムや社内研修、または職業専門学校でとったことがある．

[8] 計算機に関する授業をまったく受けたことがなく、すべての知識は現場の作業を通じて学んだものである

6. オブジェクト指向技術という用語を聞いたことがありますか？（オブジェクト指向プログラミング、オブジェクト指向システム開発、オブジェクト指向分析設計）

[48] はい

[0] いいえ

7. 6番の質問にハイと答えた方に質問です．どこでオブジェクト指向技術について聞きましたか？（当てはまるものすべてに印をつけてください．）

[8] 学校で、オブジェクト指向技術やパラダイムについての授業をとったことがある

[22] オブジェクト指向技術やパラダイムについてのセミナーをとったことがある

[30] オブジェクト指向技術について、新聞やビジネスウィークなどの雑誌の記事で読んだ

[22] オブジェクト指向技術について、CACMやIEEE Softwareなどの専門雑誌や専門書で読んだ

[25] 実際にオブジェクト指向技術に携わっている人達が周りにいて、その人達から聞いた

[22] 所属している組織で実際にオブジェクト指向技術を使っているプロジェクトがあった

そのほか（説明してください）

8. オブジェクト指向技術に関するセミナーに出たことがある方は、そのそれぞれについて、そのセミナー名（e.g. C++によるプログラミング、オブジェクト指向の概念、オブジェクト指向分析設計など）、その期間（e.g. 半日、１０週間）、提供者（大学によるもの、あるコンサルタント会社によるものとか）、そしてそれを修了した時期（年 月）を教えてください．

27. それぞれのプロジェクトのサイズについてお尋ねします. プロジェクトメンバーは何人？

[5] 1 人
[30] 2-5 人
[21] 6-10 人
[20] 11-20 人
[11] 21-100 人
[2] 100 人以上

28. それぞれのプロジェクトの期間はどれくらいでした ですか？

[1] 1-2 月
[11] 3-6 月
[25] 7-12 月
[19] 13-24 月
[30] 2 年以上

29. CASE(Computer-Aided Software Engineering)ツールを使いましたか？（あてはまるものすべてに印を付けてください）

[62] CASE ツールは何も使わなかった.
[18] 分析に CASE ツールを使った（説明してください）
MacDraw, SA/SD, ER editor, STP
[12] 設計に CASE ツールを使った（説明してください）
SA/SD
[5] プログラミングに CASE ツールを使った（説明してください）
Smalltalk, C++
[4] テストに CASE ツールを使った（説明してください）
カバレッジ計測 tool
[ ] そのほか（簡単にご説明下さい）

30. それぞれのプロジェクトで、オブジェクト指向技術は使われましたか？（当てはまるものすべてに印を付けてください. ）

[60] オブジェクト指向技術は何も使わなかった.
[21] 分析にオブジェクト指向技術を使った（説明してください）
OMT
[19] 設計にオブジェクト指向技術を使った（説明してください）
OMT
[17] プログラミングにオブジェクト指向技術を使った（説明してください）

Smalltalk

[4] ドキュメンテ--ションにオブジェクト指向技術を使った（説明してください）

[6] プロトタイピングにオブジェクト指向技術を使った（説明してください）

[ ] そのほか（簡単にご説明下さい）

OMTのコンセプトを参考にした

以下の質問＝３０から３３には、上の＝３０のいずれかで”使った”と答えた方のみお答えください（すなわち、３つのプロジェクトの一つででも何らかの形でオブジェクト指向技術を使ったことのある方です）．それ以外の方は、＝３４へお進みください．

31. オブジェクト指向技術を使ったプロジェクトは、成功だったと思われますか、失敗だったと思われますか？

大いに成功　少し成功　普通　少し不成功　ほとんど不成功

1 [8]　2 [15]　3 [6]　4 [2]　5 [1]

32. オブジェクト指向技術を使ったプロジェクトが、多かれ少なかれ成功だったと思われる方にお伺いします（上の質問に１または２を付けた方です）．その理由は何だったと思われますか？（当てはまるものすべてに印を付けてください．）

[4] オブジェクト指向技術を用いることで再利用率が上がった

[10] オブジェクト指向技術を用いていることでプロジェクトメンバー--の士気があがった

[12] オブジェクト指向技術を用いることでシステムの保守性や理解度があがった

[3] オブジェクト指向技術を用いることで生産性が上がった

[6] プロジェクトの成功は、オブジェクト指向技術の利用とは無関係である

[ ] そのほか（ご説明ください）

OOPの勉強になった

High-Quality-Software

Easy-To-Transfer-Certain-Knowledge/Technology

33. オブジェクト指向技術を使ったプロジェクトが、多かれ少なかれ成功ではなかったと思われる方にお伺いします（上の質問＝３１に４または５を付けた方です）．その理由は何だったと思われますか？（当てはまるものすべてに印を付けてください）

[ ] 時間的なプレッシャーと非現実的な期待が大きすぎた
[3] プロジェクトチームの中に、十分なオブジェクト指向技術の専門家がいなかった
[ ] プロジェクトチームの中でオブジェクト指向技術に興味を持った人がほとんどいなかった
[ ] マネージメントサイドの支援がほとんどなかった
[ ] チームメンバーの支援がほとんどなかった
[1] 単にオブジェクト指向技術ではシステムが開発できなかった
[ ] そのほか（ご説明ください）
OMTに対する不理解

34. そのほかに、あなたのオブジェクト指向技術の利用経験に基づく感想やコメント、その効果や欠点、またあなたの組織の中で近い将来にオブジェクト指向技術が用いられる可能性などありましたら、ここにお書きください．

実用レベルでOO技術を用いるのは困難
ドメインによって特化された使い方がなされるべきである．
企業内で組織的に使用するのはまだ先である．
導入には絶対に成功させるという覚悟が必要
オブジェクト指向技術の専門家が必要
理解性のよい技術である
GUIの設計には必須
考え方はどんどん取り入れられるべきである

35. この調査そのものに関するコメントなどおありでしたら、ここにお書きください．
（省略）

ご協力に深く感謝申し上げます．ありがとうございました．

付録２：　オブジェクト指向技術実態調査２．アンケート及び集計結果
（[]内に回答者数，総数38名）

-----パート１：あなたの個人的背景 -----

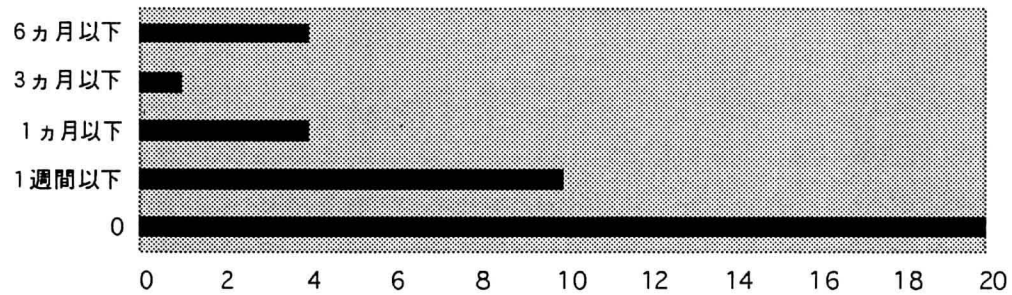
1.　「オブジェクト指向技術」という言葉を知っていますか？（オブジェクト指向プログラミング・オブジェクト指向システム開発・オブジェクト指向分析設計）

[38] はい
[0] いいえ

2.　あなた自身のオブジェクト指向技術（オブジェクト指向分析・設計・プログラミング・開発）の専門度をそれぞれ評価してみてください．

	素人		中程度		専門家
分析	[12]	[10]	[8]	[5]	[3]
設計	[8]	[11]	[8]	[7]	[3]
プログラミング	[9]	[10]	[10]	[6]	[3]
開発	[12]	[8]	[9]	[4]	[3]

3.　「オブジェクト指向技術」についてどの程度学習しましたか？会社を通じて参加されたセミナー（セミナー名・セミナーの期間・セミナーの終了時期）、また、あなたが自主的に行った勉強（オブジェクト指向技術に関する本を購入した　など）を教えてください．



-----パート２：プロジェクトの背景-----

4.　最近行ったプロジェクトを3つ挙げてください．（省略）

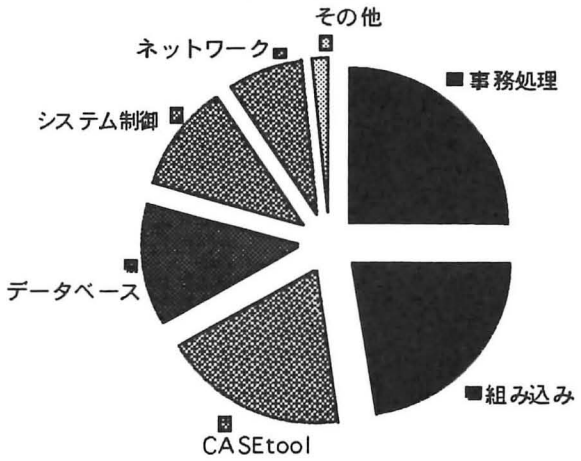
5.　それぞれのプロジェクトの期間はどれくらいでしたか？

[7] 1-2月
[25] 3-6月
[20] 7-12月
[12] 13-24月
[15] 2年以上

6.　それぞれのプロジェクトメンバーは何人ですか？

[7] 1人
[37] 2-5人
[13] 6-10人
[15] 11-20人
[7] 21-100人
[0] 100人以上

7. それぞれのプロジェクトについてアプリケーションドメインを説明してください.



8. CASE(Computer-AidedSoftwareEngineering)toを使いましたか？当てはまるものすべてに印をつけ、当てはまる場合はその名前、簡単な機能、メーカー、価格、使用期間など支障のない範囲で詳しく教えてください.

[59]CASEtoolはなにも使わなかった.
[10]分析にCASEtoolを使った. (説明してください)
[15]設計にCASEtoolを使った. (説明してください)
[3]プログラミングにCASEtoolを使った. (説明してください)
[2]テストにCASEtoolを使った. (説明してください)
[0]その他の目的にCASEtoolを使った. (説明してください)

9. CASEtoolを使うことで各フェーズの作業ははかどりましたか？当てはまるフェーズすべてに印をつけてください.

[5]分析
[12]設計
[5]プログラミング
[2]テスト
[0]その他

10. それぞれのプロジェクトでオブジェクト指向技術は使われましたか？当てはまるものすべてに印をつけ、その技術の名前など詳しく教えてください.

[28]オブジェクト指向技術は何も使わなかった.
[23]分析にオブジェクト指向技術を使った. (説明してください)
[23]設計にオブジェクト指向技術を使った. (説明してください)
[41]プログラミングにオブジェクト指向技術を使った. (説明してください)
[4]ドキュメンテーションにオブジェクト指向技術を使った. (説明してください)
[18]プロトタイピングにオブジェクト指向技術を使った. (説明してください)
[5]その他

以下の質問には上の#10のいずれかで“使った”と答えた方のみお答えください. (すなわち3つのプロジェクトのうち1つでも何らかの形でオブジェクト指向技術を使ったことのある方です.) それ以外の方は、#16へお進みください.

11. オブジェクト指向技術を使うことを決定したのは誰ですか？

[8]上層部
[16]プロジェクトマネージャー
[14]個人の判断

[13] その他（説明してください）

12. なぜオブジェクト指向技術を使いましたか？または何の目的で使われるようになったと思われますか？当てはまるものすべてに印をつけてください.

[9] オブジェクト指向技術がはやっているから使った.
[4] とりあえず試験的に使った.
[18] プロジェクトの開発期間短縮のため使った.
[29] 再利用の増加のため使った.
[27] 生産性の増加のため使った.
[15] 品質向上のため使った.
[ ] その他（説明してください）

13. オブジェクト指向技術を使うことによって効果があったと思われる項目すべてに印をつけてください.

[19] オブジェクト指向技術を用いることで再利用率が上がった.
[13] オブジェクト指向技術を用いることでプロジェクトメンバーの士気が上がった.
[18] オブジェクト指向技術を用いることでシステムの保守性が上がった.
[16] オブジェクト指向技術を用いることでシステムの理解度が上がった.
[5] オブジェクト指向技術を用いることで効果があったとは思えない.
[17] その他

14. 効果がなかったと思われる項目は何が原因であると思われますか？当てはまるものすべてに印をつけ,具体的に説明してください.

[12] オブジェクト指向技術にたいする不理解
[0] プロジェクトサイズ（主に開発期間）が大きかった
[1] プロジェクトサイズ（主に開発期間）が小さかった
[0] プロジェクトサイズ（主に開発人数）が大きかった
[1] プロジェクトサイズ（主に開発人数）が小さかった
[2] アプリケーションドメインが適切でなかった.
[0] CASEtoolがうまく使いこなせなかった.
[11] その他

15. どのようにプロジェクトを改良すれば（プロジェクトサイズを小さくする、オブジェクト指向技術についてもう少し学習する　など）よりオブジェクト指向技術を使用した効果が上がると思いますか？

（省略）

16. その他に、あなたのオブジェクト指向技術の利用経験に基づく感想やコメント、その効果や欠点などがありましたら、ここにお書きください.

（省略）

17. この調査に対するコメントなどがありましたら、ここにお書きください. なお、個人的にご協力いただけるのであればメールアドレスを教えてください.

（省略）

ご協力に深く感謝申し上げます. ありがとうございました.

家庭内乱^H^H 内 LAN を張る

1997/04/18

太田俊哉

1.はじめに

ずいぶん前なので忘れてしまったけど、SEAMAIL に長原さんの IDSN 導入記が載ったことがあった。あの当時は ISDN は個人で引くのは極めて珍しかったので、有意義なレポートだった記憶がある。

現在は、ISDN は通常の電話よりも申し込みが多くなったほど当たり前のものになりつつある。だから、いまさら ISDN を引いた、というのではパソコン入門誌の記事にしかならない。しかし、個人で LAN、それも、建物建築時にあらかじめ組み込んだ LAN の話ならどうであろうか。せいぜい 2〜3 台を繋ぐのであれば、SEA のメンバーならば当たり前ではあろうが、建物組み込みはまだ事例がそう多くはないのではないか。

というわけで、SEAMAIL の原稿不足救済(!?)の意味も込めて、木造軸組み在来工法における LAN 敷設の実際についてレポートする。

2.そもそものきっかけ

そもそものきっかけは社宅の利用期限である。筆者の務めている会社(電気連合系)では、社宅の利用期限が 7 年間と決まっている。また、コンピュータの台数も、いつのまにか、2 台、3 台と増えてきた(なぜか WS まで増えてしまった)。退出まであと 1 年半の時点で、そろそろ次の住み処を考えておかなければならなくなったわけである。

当時(94 年〜95 年)は、バブルが終わったとはいえ、まだまだ一軒家には手が届かない状態であった。では賃貸で、と見回してみると、これも意外と高い。筆者は以前に 2DK で 58000 円のアパートに住んでいたが、社宅の近辺では 12〜13 万が当たり前、3DK だと 17 万くらいしていた。これでは到底住めそうもない。

もちろん、もっと安い公団や都営(都民/公社)住宅もある。しかしこれらはみな抽選。20 数回チャレンジしたが、良い所は競争倍率が高いため全然当たらない。無抽選の所もあるにはあるが、遠くて高い(下手をすると民間よりも高い)ものばかり。20 回以上の優先枠も通勤不可の遠距離にしかない。

で、下がりつつある住宅貸し付けの金利なども考え、中古のマンションでも買おうかな、と思っていたわけである。

が、

この考えは、阪神淡路大震災で無残にも打ち砕かれることになった。

TV で見る限り、老朽化した中古マンションはやめようということになった。そこで、(その時点では若干無謀に思えたが)一軒家を中心に考えることになったのである。

3.SEA会員に贈る、家探しの極意

本来はLANを張る話をメインにすべきなのだが、その前に家をどのように入手するか、ということは(そろそろその年代の人にとってみては)かなりの関心事であろう。不動産情報誌や新聞の折り込み広告などで情報が入ってくる一方、欠陥住宅の話も事欠かない。そこで、LANを張る前に、

家の購入もSEA会員の関心事項であり、技術移転の対象となる

と、勝手に考え、極意というほどではないが、ちょこっとだけ筆者の体験談をまとめてみることにしよう。

3.1 まずは土地探し

一軒家と言えば、

理想は注文住宅一戸建て

である。自分&家族の希望どおりの家が立てられれば、これに越したことはない。

しかし、現実は単なるソフト系の仕事をしているサラリーマンには、

ヒジョーに厳しい

のであった。では、時系列で、どのように現実を思い知らされていったかを説明しよう。

3.1.1 Limit1 年半前(95/01 の段階)

家探しをするならば、通常は週刊住宅情報とか、新聞の折り込みチラシに掲載されている仲介中心の不動産会社に問い合わせるのが順当な手順であろう。しかし、筆者の場合、子会社に従業員福利厚生のための不動産部門があったのである。そこで、まずここ相談してみることにした。が、一言

「あなたの予算では埼玉の奥の方か山梨県じゃないと無理」。

一戸建てがいかに高いかということを思い知らされたのであった。

3.1.2 Limit1 年前(95/07 の段階)

ちょうど子供の入学式などで忙しく、しばらくは家探しをほっておいた。せいぜいたまにチラシを見て「高いなあ」とため息をつく程度(7000万台があたりまえ、億以上もたくさん)。まあ、住んでいるところがいわゆる高級住宅地のはずれだったので、地元向けのチラシがハイエンド指向になるのはしかたがないんだけどね。

ただ、いよいよ1年を切ったので、もう一度、今度は本腰を入れて探すことにした。

まず、近所の立て看板を見て、近所で売りに出ている不動産紹介(オープンハウス)を見学することにした。近所の物件は、金額的には2000万以上オーバーするものばかりだが、取り合えず新築の物件とはどんなものかを見ておくことは必要だからと思ったからだ。イメージがわかないと決断しようがないからね。

で、不動産屋の人と話をしてみたところ、値段的に手ごろな物件はそこそこあるとのことなので、とりあえず、資料請求のお願いをしておくことにした。

3.1.3 Limit 11 カ月前(95/08)

建築屋の友人が1年ほど前に家を建てたので、それを見学させてもらうついでに、建ててもらった不動産屋(兼工務店)を紹介してもらった。で、まず基本的な話をいろいろと教えてもらった。土地だけでは難しいとか、沿線によって出物があるないとか。で、時間を取ってもらって建て売りを4件ほど見せてもらった。公庫仕様(要するに100m*2以上のもの)とそうでないものがいっぺんに建っている現場とか、日当たりがいいけどちょっと狭いとかなど、非常に丁寧に(そりゃそうだ、となりに昔

の顧客の、しかも同業者が座っているんだもんね)説明してくれた。しかし、低予算での現実を実感してしまうことになった。この時はまだ理想が高かったので、現実とのギャップが大きかったからである。

3.1.4 Limit10 カ月前(95/09)

建てるんだったら丈夫なもの、ということで、最近の家はどんな構造で作られるかを知るため、住宅展示場回りを始めることにした。幸い歩いていける所に比較的小規模なものが 1 つ、電車で数駅の所にかなり大きいものが 1 つあるので、散歩がてら回ってみることにした(いろいろ景品もくれるし)。いくつか見学をしたが、たまたま、某 2x4 メーカーの所で捕まり、いつのまにか土地探しをお願いすることになった。ハウスメーカーは建てるだけではなく、ちゃんと不動産業社と連携取って、土地から面倒見ることがわかった次第。なるほど、単に建てるだけだの客(=土地持ち客)はそうそういないから、それだけだったら商売にならないということか。

この辺りから、新聞のチラシを見て、めぼしいものの資料をかたっぱしから資料請求することになる。もちろん紹介される度毎にいちいち現場見学していたのではたまらないので、FAX で図面だけを送ってもらうことにした。おかげで一次ふるい分けができ、FAX の利用は大正解であった(結局決まるまでに 100m 巻きロールがまるまる 2 本なくなった)。

3.1.5 Limit9 カ月前(95/10)

たまたま多摩川のそばで出物があったということ。まだ古家は建ったままだが、更地渡しにするということ。若干狭いが(公庫基準は満たせない)予算的には何とかなりそう。

建物のアウトラインを引いてもらうことになったが、結局売り主が諸般の事情で売るのをやめたため、この話は流れてしまった。

このとき、家のだいたいの構成と、費用がどのくらいかを大まかに知ることになった。子会社の不動産部を経由して、別のメーカーにも見積もりを出してもらい、

メーカーや工法によって数百万の差がつく

ことがわかった。また、仕様決定から着工、完成までのスケジュールがどのくらいかということもわかった。かなりタイトなスケジュールである。仕様決定まで 1 カ月程度しかない。そのあいだに、レイアウトから配線、内装まで決めないとダメなのだ。

3.1.6 Limit8 カ月前(95/11)

2x4 メーカから紹介された不動産屋さんが、色々と建て売り新築物件や中古物件を持ってきた。いくつか見学することになった。が、なかなかよい条件のものがない。

この時期に、別の人に私の希望をすべてかなえるとどのくらいかかるか、と聞いたら、半ば呆れ顔で「億だね」と言われてしまった。

裁判所の競売物件というのも 1 つぶつかった。金額的、かつ場所的にはよい所(駅 10 分)だったが、素人には危険過ぎるということで流した。

そのほかにも、

1)急斜面西向きの築 20 年の家。セットバック必要なため、傾斜部分の切り崩しが必要で、立て替えの費用が非常に掛かることが難点。
2)築 17 年の家。作りは頑丈だが、地下車庫が不便(車庫はいらん)。
3)同 17 年の家。作りは非常によい。全面改装のため新築同様。が、5 分差で別の人に取られた。
4)築 10 年の家。駅から遠いが、土地が非常に広い(145m*2)。が、どうも夜逃げした家らしい。

などというのがあった。

3.1.7 Limit7 カ月前(95/12)

このころになると年末のため、出物が減ってしまう。いいものがないので、しかたがなく、代替策として、アパートをも考え始める。長期戦を覚悟したのもこのあたり。

3.1.8 Limit5 カ月前(96/02)

朝ポストを見ると、近所で空き地になっていた土地に建て売りを立てて売るという案内が入っていた。日当たりなどが多少悪い(北接道)、やや狭い、など条件はあまりよくないが、その他の条件がよすぎる(歩いて5分の距離、周りに林がある、駅まで5分、小学校の学区代わらず、目の前が公道でセットバック不要、車はほとんど通らず)。

ざっと下見をして、とりあえず押さえておいてもらい、権利関係を確認したのち、ここに決めることにする。

3.2 ここまでのポイント

決めるまでの経験から、いくつかポイントをまとめてみよう。

1) 土地だけの売り物は少ない

少なくとも首都圏の都心に近い部分で、手ごろな土地だけ探すのは極めて困難であることがつくづくわかった。よさそうなものはほとんど建売業者に買われている。土地で出ていても建築条件付きのものが大多数。

2) 中古は税制面で不利になる場合がある

中古の一軒家でもよいものはあるが、優遇税制が使えないものがある(15年以上経ったもの)。年間数十万税金が違う場合がある。

3) 妥協が必要

どこかで妥協しないとだめ。日当たりも良く、土地もそれなりにあり、駅から近く、閑静な住宅地で、などというのははっきりいって無理。億の金が必要。なにかを捨てれば条件を満たすものは色々と出てくる。

また、安易な妥協は禁物。よいものが取られたとしても、必ず別のよいものが出ると信じること。

4) 根気よく探すこと

長期戦のつもりの方がよい。短期間だと、勉強/経験不足で後悔する可能性が大。

5) 勉強しよう

単に週刊住宅情報の記事を読むだけではなく、色々な住宅雑誌や本などで勉強しておくことが必要。本は図書館に行けば色々と置いてある。

また、住宅機器メーカーのショールームなどにいって、色々と見て回るのもよい。照明器具などはいやというほど種類があある。また、キッチンなどは高さがいろいろあるので(いまだと平均は80cmか)最適の高さというものの感覚をつかむことができる。

6) 情報の窓口は広く

複数の不動産会社に声をかけること。それだけ情報が増える。但し、かけすぎるのも問題。あとの情報の整理が大変になる。ま、しばらくすると、まともな情報を出す会社、わりといいかげんな会社などがわかってくる。そうしたら、本命、対抗、義理などのランクわけをすればよい(ただ、私の場合には大穴だったけど)。

7) FAXは必須

図面などを送ってもらえ、毎回出歩かなくても一時ふるい分けができる。ある程度なれてくると、図面だけでイメージがわくようになってくる。

8) 最低 20 件は見学すること

できるだけ多くの家を見ることによって目が養われる。特に、建築前の家と、建築後の家を見るとよい。建つ前に建ったイメージが沸くようになる。理想は、同じ家で、基礎から完成までを眺めることか。

9) 都市計画等の確認も怠りなく

住宅密集地ではほとんど不可能に近いが、都市計画道路に引っかかっていたりする場合がある。数年後立ち退きなどの話が出てくる可能性もあるので注意が必要。

10) 資金計画はじっくりと

購入代金以外にもかなりのお金が必要となる。引っ越し費用もそうだし、そのほかにオプションの什器、固定資産税の日割り分(購入日から年末までは購入者の負担となる)、火災保険、引っ越しの挨拶費用などなど。

また、月々のローン支払いは無理のないように。固定資産税や火災保険代を忘れずに。理想は、マンションの管理費と同じ分だけをメンテナンス費用として積み立てておくのがよい。数年後必ずどこか直さなければならなくなる。最低でも 3〜5 年間隔で畳は取り替えなければならないし。

3.3 建て売りのよい点、悪い点

今回は一応(何で一応かはこの後説明する)建て売りを買うことになったが、一から立てるよりも建て売りの方が便利な点も多かった。

建て売りも捨てたもんじゃない

のである。地鎮祭とか棟上げ式なんかしなくて済むし、細かな仕様は建て売り屋さんが決めているので考えなくてもよい。そのかわり

基本的に as is

で買わなければならない。どう作られたか(手抜きがないか)なども、でき上がってからではわからない。

もちろん、建っている最中、建ってからの仕様変更は難しい。また、建つ前の仕様変更をどこまで聞いてくれるかは業者による。通常は、レイアウトの全面変更などはやってくれないと思った方がよい。せいぜい、壁紙や壁の色、電源の配置、組み込み機材(システムキッチンなどの若干の仕様変更)程度である。また、後になればなるほど費用がかかる。

このあたり、つくづく思うのだが、家を買う(建てる)のは

ソフト開発と同じ

ではないかと(おー、SEA むけの話題になったぞ)。ソフトも上流過程で仕様をきちっと決めておけば後が楽だし。人(会社)によっても仕様変更(=わがまま)を聞いてもらえる範囲が違うし。まあ、もの作り一般に共通なことなのかもしれない。

また、工法の違いをしっかりと理解しておく必要もある。たとえば、パネル工法系は後配管が難しいことに注意しなければならない。特にコンクリート系は穴を開けるだけでも大変。もちろん増築も一筋縄ではいかない。壁で支える構造のため、増築して壁を取り去ると、構造計算のやり直しになってしまうからだ。

さて、コンピュータを大量に入れることを考えた場合、なんといっても重要なのは「電源」と「空調」である。空調も最終的には電気だから、結局は「電源」をどのように構築していくかが最大のポイントである。マシン室は別系統必須、空調用の電源は別配線必須である。あとはコンセントの位置やアース程度なので、その辺を押さえておけば、後は何とかなる。

4. 注文はどこまで受け付けてもらえるのか

とりあえず建て売りを買うことになったが、買う家がまだ図面の段階であった。そこで、LAN をあらかじめ引くことができるかを相談してみた。要するに、ソフト開発ではおなじみの「仕様変更」である。

4.1 Limit4 カ月前(96/3)

契約する。これがけっこう神経を使う。重要事項の説明から始めると、3 時間くらいかかった。ま、

そこらでパソコン買うのとは金額が 2 桁違う

からね(ちなみにマンションだと 1 時間くらいで済むときもあるんだって)。

契約と同時に家の詳細仕様検討に入る。ここで、LAN を張り、マシン室を確保するための設計変更が可能かどうか聞いてみた。なんと、2 週間程度の余裕期間で詳細を決められれば大丈夫ということなので、設計の見直し作業に入ることになった。

ちなみに参考までにどのような点を変更依頼したかというと、

- 駐車場をなくす(これで部屋が確保できる)
- 風呂をユニットバスに(メンテナンスが楽)
- オーブンレンジ追加
- フローリングを畳に(この方が冬温かい)
- 照明器具などの詳細指定
- 電話とドアホンの統合化(ホームテレホン化)
- 200 V対応
- 分電盤の 12 系統化(マシンのことを考えて)
- 窓をすべて断熱ペアガラスに(冷暖房費削減)
- マシン室を作る
- LAN 配線の追加工事

といったところ。要するに、

ほとんど全部

である。ま、メーカー品のパソコンを買って、マザーボードとグラフィックカードとサウンドカードと HD とを取り替え、メモリと SCSI カードと MO と DAT を追加するようなもんだと思えばよろしい。

LAN の配線は、家が在来軸組み工法なので、引き回しが楽なのですぐに OK が出た。但し、通常の電気工事屋さんでは最後の配線までは難しいので、部材の提供と、最後の配線はこちらもちということになった。

4.2 マシン室と LAN のプラニング

マシン室(書斎?)の確保ができたので、LAN を引くことを全体にしたレイアウト変更が可能かどうかを打診してみることにした。これも、ほとんどすべて OK が通る。そこで、早速家庭内 LAN のプラニングをすることにした。

まず、マシン室に 10BASE-T のハブを置き、そこに LAN の配線を集中させることにした。マシン室には ISDN の TA も置き、機材を全部ここに取りつけることにしてすっきりさせようとするのが狙い。ただ、ホームテレホン親機は別の部屋に置くことになったので、そこの配線が少々ややこしくなってしまった。今にして思えば、ホームテレホン親機もマシン室に置けばよかったと思う。
また、各部屋の LAN の出口は、壁取りつけ型のものにした。ただ、1 口だけの RJ-45 パネルが見あたらなく、しょうがないので 2 口のパネルで 1 口をダミーにすることにした(あとで、なんかのショーで配っていたパーツ屋さんのカタログには載っていた。たまたま秋葉原のなじみの店には置いていなかっただけ)。

マシン室の電源は、それだけ別アースを依頼した。洗濯機や電子レンジなどなどアースを別にすることで、できるだけノイズを削減しようと思ったわけだ。この辺は現地で調整ということになった。いわゆる

現場でデバッグ

である。

コンセントは、TA をラック上に置くことも考え、上、中、下と付けてもらうことにした。当然エアコンのコンセントも用意。熱を逃がさない換気扇(ロスナイ)の設置も考えたが、部屋が小さいのでそこまでの必要はないということで、やめにした。

また、電源回りは、電源容量を試算し、できるだけなにかあっても被害が少ないような引き回しにしてもらった。

・電灯とコンセント系列は別
・台所は大電流が流れるものが多いので系統を分ける
・どこかが落ちても隣の部屋の電気がついて暗くならないように

という工夫をしておいた。また、将来のことも考え、200V も引けるようにしておいた。当然入り口は単相三線 200V。おかげで、分電盤が 12 系統必要になってしまった。

マシンのケーブルの布線を考えると、やはり

フリーアクセスは理想

だろう。しかし、さすがに無理だった。理由は、部屋の高さがそれほどないこと(尺モジュール)、費用がかなり掛かってしまうこと、だいだいそんなにマシンを置く予定がない。フリーアクセスにすれば、エアコンも床下吹き出しを考えたのだが、そのようなことができる家庭用エアコンは見当たらなかったというのも遠因である。

電磁シールドも考えた。しかし、これも費用と木造在来工法等の相性であきらめた。でも建てた後 PHS を屋内で使おうとするとうまく使えないので、意外とシールド状況が良いのかもしれない。

また、部屋の鍵を子供のいたずら対策に備えて付けてもらうことにした。これは、UNIX マガジンの坂本文さんの連載で、子供がマシンをいたずらしてしまうということが書いてあったからである。実際、3 才くらいになると、勝手に電源を入れて、勝手に CD-ROM を入れて、勝手にゲームを始めるくらいはできるようになるのでこれは大正解であった。

4.3 そのほか

そのほか考えたものとしては、

・無停電電源→後で必要になったら買う。
・太陽光発電→お金がない!
・集中型エアコン→個別空調の方が安上がり。集中型の方がエネルギー効率が悪い。
・床暖房→そんなに居住しない。通常は会社だから。
・雨水再利用システム→お金がない

などがあった。

あと、建て売りの場合、照明はくくりつけのもの以外は自己調達である。しかし、引っ越していった人が置いていったインバータ型のコードペンダントをただでもらってきてあったのでそれを利用することにした。社宅だと引っ越し時にいろいろと融通してもらえるので、こういったとき便利である。

4.4 詳細仕様の指示

配線は、あらかじめ設計図面に

可能な限り詳細を記述しておく事が成功の鍵

である(コンセント、照明等)。今回の場合、配線が極めて複雑なのでかなり詳細に記述しておかないと誤配線の元になるからだ。また、分電盤配線の指示もする。

スイッチ等の型番もできる限り詳細に指定しておいた。ただ、電工品は、仕入れの関係があり、必ずしも師弟の物が付くとは限らないとのことであった。そこで、妥協できるものと妥協できないものをわけて指定しておくことにした。基本的には松下電工のものを指定したが、建ってみると一部は東芝製だった。

スイッチ等はあとで取替可能なので、気に入らなければ後で取り替えればよい。

4.5 工具などを買う

パネルはぷらっとほーむ(秋葉原のハイレベルユーザ向けパソコンショップ)と愛三電気(秋葉原のプロ向け線材専門店)で、線材は愛三電気でカテゴリ 5 をバルク購入。100m と 250m があったが、足らないと困るので 250m を購入。ただ、計算外だったのは、250m のバルクケーブルは非常に重い。そういえば、いくつかの雑誌記事で、100m 単位のイエローケーブルが届いたら、とても台車がないと運べないという記事が載っていたことがあった。さすがにカテゴリ 5 のケーブルはイエローケーブルよりは軽いが、持ち帰ることができそうにもなかったので、しかたがないから発送してもらう。

5. 配線の実際

配線は、軸が組み上がり、外壁/内壁/床の下地ができ上がるころに行なう。内壁は石膏パネルを張りつけるのだが(吉野石膏だった)、その前に配線しておかないと這い回しができないからだ。

工事はほぼ一日がかり。2 軒分まとめて数人がかりで行なう。配線の間違いや、細かな位置を現場合せで行なうため、現場立ち会いを行なうことにした。コンセントの垂直位置などは、図面上では上、中、下程度しか指定していないが、現場で「だいたいこの辺」と指定した。ま、在来工法というのが、

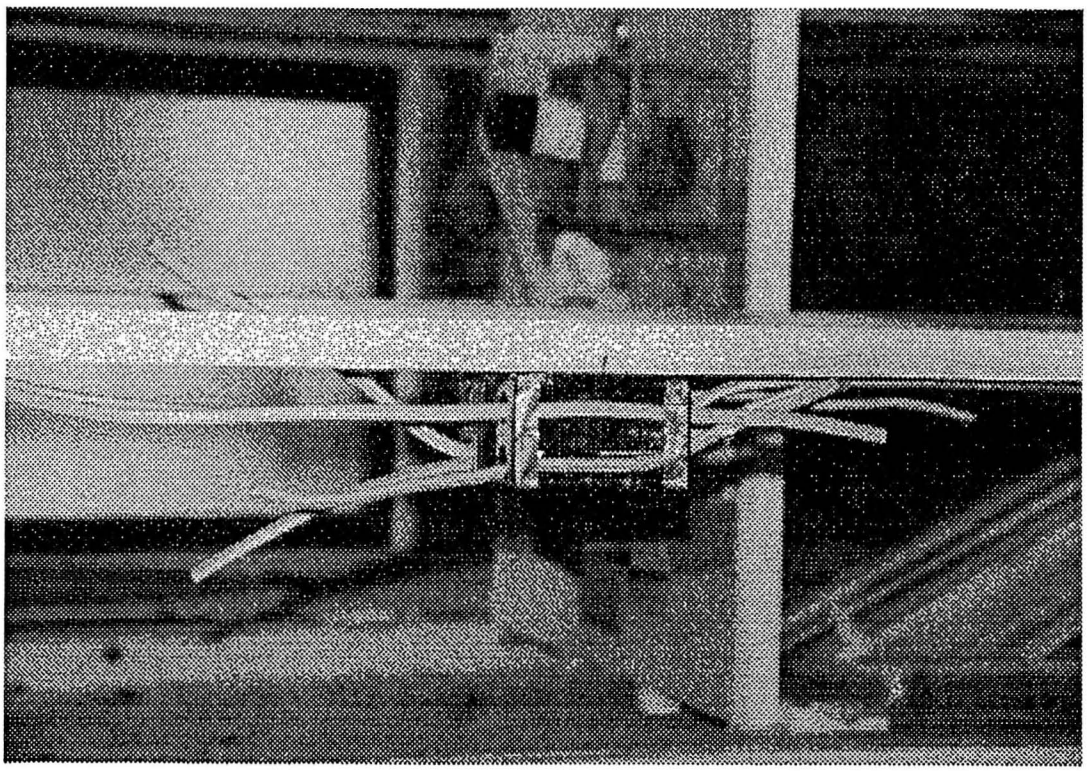
詳細部分はアバウトな作り方をしてる

ということだ。

また、電源系統の這い回しも現場で若干変更することになった。たとえば台所の照明スイッチは一箇所の予定だったが、それだと台所に入ってからでないと電気がつけられないことがわかったのでも

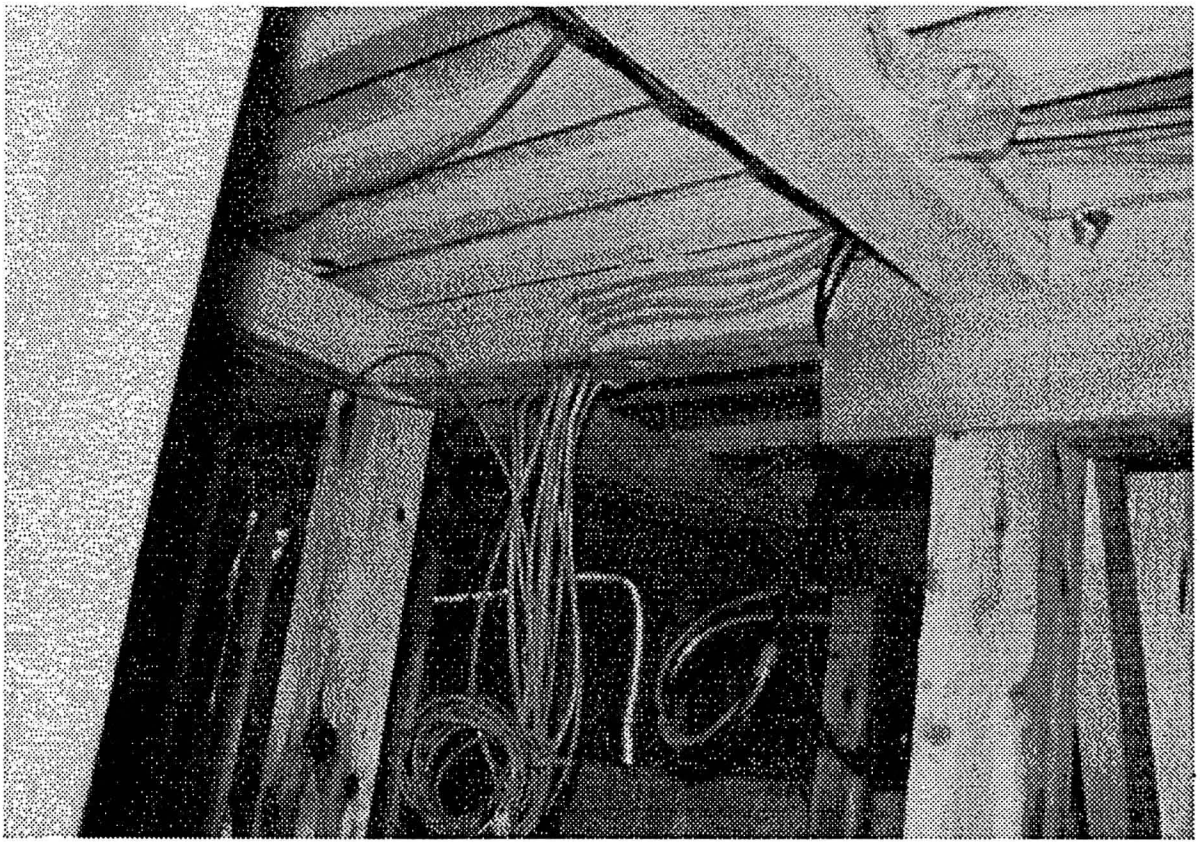
う１つつけてもらうことになった。この辺が在来工法の融通の利くところ。パネル工法系だとこうはいかない。

配線工事は極めて簡単。ケーブルを柱に沿ってステープルで止めるだけ。ただ、大変なのは、配線量。同じサイズの通常の家の３倍とか。工事にきていた人がかなり苦労していたようだ。特に、単に配線量が多い(距離が長い)のではなく、配線があっちへいったりこっちへ行ったりしたり、系統の違う配線が 4 種類(電源、TV、電話、LAN)あるので、一筋縄ではいかなかったのが苦労の原因のようだった。通常だとアウトラインの図面を見ただけで、ささっと配線できるのだそうだが、今回は毎回図面を見ながらでないと配線できなかったとか。この辺は LAN の引き回しと同じであろう。



たとえば、この写真は、電源線を引き回しているところである(90 度曲げてみてほしい)。壁面に同じ高さになるように２個配置している。これで、壁の反対側の同じ高さの所にそれぞれスイッチが繋がることになるわけだ。

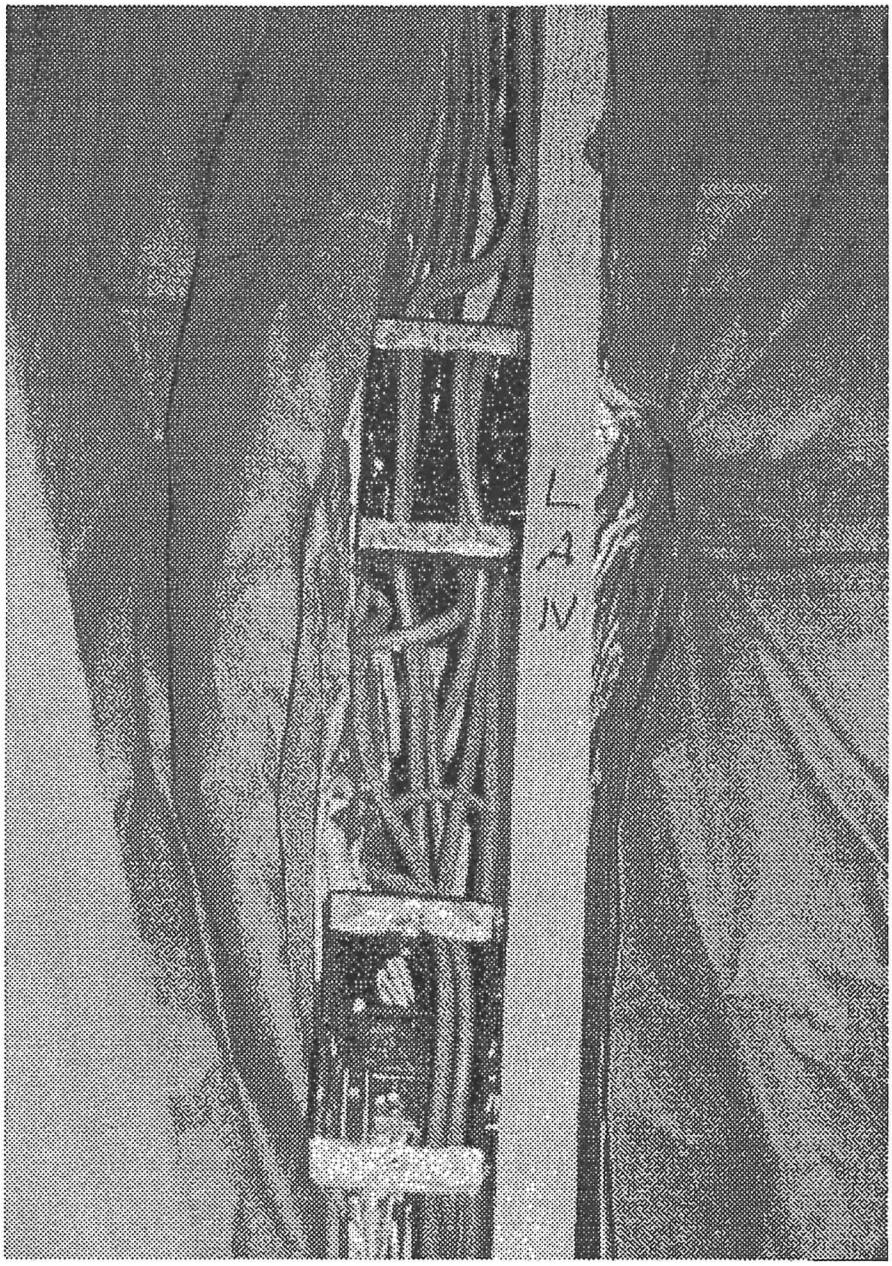
また、分電盤のあたりは、次のページの写真のようになってしまった。電源系統を凝って 12 系統としたため、相当ごちゃごちゃすることになってしまったわけである。



また、LAN の配線は次ページの図のようになっている。カテゴリ５は意外と太いので、はい回すのに苦労していたようだ。

ただ、現地である程度打ち合わせしながら工事をしても、最終的には電話線回りは間違えられてしまった。ISDN が２系統。しかも、一旦外部からの電話線を DSU/TA が受けたあと、再度ホームテレホン親機(これはマシン室とは別のところにある)に渡すため、配線が理解できなかった模様。

ただ、LAN のことは理解できていたらしく、工事屋産もカテゴリ５のケーブルを持ってきてくれていた。オフィスの LAN の配線もしたことがあるとのこと。



6. 引っ越し準備

6.1 Limit1 カ月前(96/05)

いくら近所といっても引っ越しは引っ越しである。段取りよく行なわないと徹夜で荷物を梱包するはめになる。最近では梱包サービスまでやってくれるのもあるが、けっこうな値段がするので、できるだけできることは自分たちでやることにする。

まず引っ越し業者を決める。近所の引っ越しの時に非常に対応が悪かったところを除き、数社に当たる。不動産屋さんがなじみの所を紹介してくれる場合が多い。そのほかにも、会社の共済組合とか生協とかで選べばよいだろう。

ポイントは、

必ず見積もりだけ取ること

である。

見積もりを取った後その場で決めろと迫る場合が多いが、旦那が帰ってきてからでないととか言いつくろってその場で決めるのはやめよう。比較してみないと高い安いがわからない。以前にすぐに決めてかなりふんだくられてしまったことがあった。

各社それぞれ、ダンボール代代別で安いとか、いろいろサービスはするけど高いとかいろいろあったが、中庸な所を選ぶことにした。

マシンのうち、一部のものはあらかじめ運んでおく。自分で運ぶのがやはり安全。いくら最近のコンピュータは丈夫になったとはいえ、ワークステーションなどはやはり心配である。引っ越しに伴う不用品を安く譲ることを条件に、友人を一日車付きでこき使うことにした。

また、新築につきものの新築病が怖いので、毎朝窓を開け、帰りがけに窓を閉めて換気を行なう。これは、新居がごくごく近所にあるメリットであろう。通常ではこんなことはできない。まあ、引越前に掃除にいった時に換気するのが精いっぱいだろう。

また、虫予防のため、念のためバルサンを焚いておく。新築なのでゴキブリなどはいないはずだが、工事中に虫が入り込む可能性があるので。それに、引っ越すとなかなかバルサンを焚くというのはやりにくい。

本当は、火事と間違われないように、消防署に連絡しておくのがベターとか。しかし、焚いてみたら、ほとんど煙が出なかったので、それほど気にすることはないように思えた。

引っ越し荷物は、必要度の低いものから順次梱包をしておく。新居のどこの部屋に入れるかを明記しておくのがポイント。もちろん箪笥などは配置位置をきちんと決めておく。マーカーや紙などでメモしておくとベター。

また、運べるものは先に運んでおく。そうすると、次に梱包したものをおく場所が確保できる。

6.2 引っ越し二週間前(96/06)

エアコンは、個別型だが、室外機１つで室内機２つというタイプを最初選んだ。システム空調タイプも考えたが、

カタログスペック上は意外と効率が悪い

のでやめにした。大型機よりも個別型の方が技術が進んでいる。

ただ、配管が長すぎ、冷えない可能性大と、工事に来た人が言ったので、当日急遽機械を取り替えた。近所の電気屋(中型のチェーン店)で購入したので無理がきいた。単なるディスカウントストアだとこうはいかないだろう。

一応居室はインバータタイプ、マシン室は(使用量が少ないことも考え)安価なタイプにした。

7. 引っ越し

7.1 Just Limit 当日

当日は、ただただ引っ越し作業をするだけ。ご近所の挨拶はそれまでに済ませておく。

マシンについてはていねいに運ぶことを依頼する。とはいっても、引っ越し荷物を積んだトラックより、自転車でいった人の方が早くつく距離なので、作業はほとんど積み込むのと下ろすことだけ。

到着後は、荷物をどこに置くかのデスパッチを行なう。このとき、

あらかじめダンボール表面に内容と共に置き場所を書くとすごく楽

である。この辺は、ユニマガのワークステーションのおと(1992 年ごろ)が参考になる。

また、旧居(?)の後片づけ、掃除は同じ社宅の人が手伝いにきたのでかなり早く終わった。まあ、それで足らなければ、日を改めて掃除にいけばいいんだけどね。

8. 引越後

8.1 マシン室備品の購入

本来は、マシンラック、机などの新規購入品はあらかじめ買っておいて、

引越前に運び込んでおくのがベター

である。しかし、ちょうど時期外れでいい品があまりなかったので、引越後に買い込むことになってしまった。これは失敗。なぜならば、すぐにマシンのセットアップができないからだ。結局机がくるまでの間は床にディスプレイを転がしてマシンを使うはめになってしまった。

机は、いわゆるパソコン机ではなく、通常の机を買うことにした。机の上に17インチを置くので、奥行きがほしかったのだが、ほとんどが規格サイズ(奥行き70cm)でなかなかよい物がない。そこで、机の後ろにファングル(組立式スチール棚)でディスプレイ台を作ることにした。奥行き45cmの台を作ると、ちょうど17インチディスプレイとSS2が載るので具合がよい。高さ方向は机の高さと一致するように、柱(アングル)部分を加工してもらったので、机との段差がほとんどない形でティスプレィ置き台を作ることができた。

はっきりいって、

通常のパソコン机、やJIS規格準拠の机は使い物にならない

のだ。キーボードを置いてしまうと他のことができなくなる。奥行き70cmしかないからだ。デスクワークならば十分な広さであるが、17インチディスプレィを置くと、下手な机ではキーボードすら置けなくなってしまう。そこで、奥行きがあるのならば、是非ディスプレイ置き台を作り、机の上を広く確保できるような工夫をすることをお勧めする。

さらに、資料を置く棚を買う。これは、最近見かける網型のもの。無印良品にも似たものがあった(ひょっとするとOEMかも。OEMという言葉が妥当かどうかはわからないが)。以前はいわゆる業務用のスチール棚を使っていたが、奥行きが30cmしかないので不便だった。そこで、今回は奥行き45cmにしてみた。

網型のものを買ったのはわけがある。機材を網棚に置いてひもで縛り、落下防止対策をしようと思ったからだ。が、結局機材以外の資料があまりにも多く、網棚に載せることができた機材がTAとDSU程度のため、あまりメリットはなかった。

あと、いざというときのために、悲惨^H^H飛散防止フィルムを張っておいた。これは

あとからちまちま張ろうと思ったのがこれも失敗

だったといえる。しろーとが張ると恐ろしく手間がかかるのだ。ガラス1面当たり1時間以上はかかると思った方がよい。また、フィルム代が1面あたり2千円くらいかかる。家全部だと数万円になってしまう。

可能であれば、工場出荷時点で張れないかを相談してみるとよいかもしれない。

8.2 LANの配線

あらかじめ配線されていたLANの線にジャックを取りつける。これはそんなに手間ではない。末端の被覆をはがし、専用工具で接続していくだけ。

但し、工具が7000円以上するのが難点。一回の使用でおしまいにするのはもったいない。ま、次にだれかが使うときに貸し出せばいいのだけど。

壁面のパネルからハブまでのケーブルは、配線に使用したケーブルの残りを流用。これは、友人に、ケーブルの一部と物々交換で作ってもらった。

接続チェックは二台の PC を使用。ノートを 1 台用意し、ping することで接続を確認。確認したら、どの部屋と繋がっているかをパネル出口に張っておく。

8.3 ただ、まだちゃんと使っていないのね

一応配線はしたのだが、実際はさほど使う頻度がない。なぜかというと、マシン室ができてしまったため、だいたいそこで作業すれば用が足りてしまうからだ。また、サーバーの 24 時間運転もしていない。だから、ノートを台所に置いて、他の人に説明しながら作業する、という特別な場合を除き、まだ LAN を活用するところまでいっていない。

間、そのうち子供が大きくなってきたら、インターネット繋ぎたいだの何だのいってくるからその時は有効に活用することになるのではないかと思っている。

9. 住んでみて

建て売り屋さんが、仕様変更に対して、できることはやってくれたし、できないことはできないと言ってくれた。そのため、無駄な出費が少なく、かなり希望どおりの家ができたと思っている。

仕様を早期に固めて、仕様変更がほとんどなかったのも幸い。また、仕様がかなり詳細まで決めたのもよかった。この辺はソフトの制作と同じである(と SEA むけにこじつける)。

ペアガラスは効果大。断熱効果は(比較対照の家に住んでいないので)よくわからないが、とにかく結露がない(金属部分は別)。

LAN の線は 1 部屋 1 本としたが、将来を考えるともう一本の方がよかった気がする。

電話の線は引いてあるが、複数の回線を処理できるタイプではないので(ホームテレホンなら当たり前?)、個人電話には対応できないのは失敗かもしれない。ただ、これは、テレフォニーシステムかインターネット電話システムを用意すれば解決可能ではないかと思っている。

かなり詳細に説明したつもりだったが、それでも施工ミス(配線ミス)はあった。一旦外部から受けた電話線を TA でアナログに戻し、さらにホームテレホン親機に突っ込むというところが理解できなかった模様。ま、これほど複雑な配線は普通しないからね。

10. おしまいに

以上が、LAN 付きインターネット対応の一軒家を買った顛末である。はっきりいって、これだけ無理がきいた建て売りを買うというのは非常に難しいと思う。よい建売業者、よい不動産屋、大工を初めとする腕のよい職人さんたちに「たまたま、運が良く」あったのでなんとかやってこれたというのが実感である。

ソフトウェア・シンポジウム '97

開催案内および参加者募集

1997 年 6 月 18 日 (水) ～ 20 日 (金)

於： 福岡ソフトリサーチパーク (福岡県福岡市)

主 催

ソフトウェア技術者協会

後 援

九州システム情報技術研究所　福岡ソフトリサーチパーク　九州通商産業局

協 賛

日本ソフトウェア科学会　情報処理学会　情報サービス産業協会　福岡県情報サービス産業協会
電子情報通信学会ソフトウェアサイエンス研究専門委員会　福岡エレコン交流会　計装研究会情報専門部会

ソフトウェア・シンポジウムは, さまざまな場で活躍している技術者や研究者, 管理者などが一堂に会し, ソフトウェア技術に関する多面的な経験や知識を交流するユニークで貴重な場を提供してきました。

第 17 回を迎えた今回は, 世界的な注目を集めるアジアに開かれた街であり, 日本で今最も元気のある街と言われている福岡市に所を移し, 情報技術および情報産業の一大集積地として新たに誕生した福岡ソフトリサーチパークを会場に開催致します。

西暦 2000 年の足音が間近に迫りつつある時代を迎えて, 情報システム技術も華やかな話題が一般紙をも飾るようになりました. これから技術はどこへ向かうのか, 何が情報の現場で実際の問題となっており, どのような解決が図られているのか, また地域社会における情報化への取り組みがいかに私たちの生活に影響を与えるのかを, 発表し共有する機会をもちたいと思います。毎年, 現場に立脚した質の高い論文が集まるシンポジウムのレベルは, 回を重ねるごとに着実に上がってきております。今回も, 研究論文と経験論文はフルペーパーによる厳しい審査の結果, 計 25 編の論文が採択されました。また昨年に引き続き今回も, 産業界での地道な研究・改善活動を広くご発表いただく場として, 発表用 OHP で審査する事例報告というカテゴリを設け, 8 編が採録されました。プログラムは, 招待講演 (3 件), 論文発表 (9 セッション, 25 編), 事例報告 (2 セッション, 8 編), パネル討論 (2 件) の構成になっており、活発な議論が展開されるものと期待されます。

シンポジウム基調講演では, コロラド大学教授 Alex Wolf 氏にソフトウェアプロセスのお話を、慶應義塾大学教授村井純氏に次世代インターネットのお話をお願いしております。

併設チュートリアルおよびツール展示は, 今回も開設します。チュートリアル (別料金) はプロセス成熟度モデル(CMM . カーネギーメロン大学の公式テキストを利用)、ソフトウェア CALS(メーカ数社による実演あり)、Java をテーマに, 3 講座を用意しました。ツール展示 (入場無料) では, バラエティに富んだツールの展示とデモンストレーションを行います。

多数の方々の積極的な参加をお待ちします。

【シンポジウムスタッフ】

実行委員長： 武田淳男 (安川電機)　高橋光裕 (電力中研)

プログラム委員長： 荒木啓二郎 (九大)　酒匂寛 (Designers' Den)

プログラム委員 (50 音順)： 青山幹雄 (新潟工大)　鯵坂恒夫 (和歌山大)　伊藤昌夫 (ニルソフト)　落水浩一郎 (JAIST)
楠本真二 (阪大)　熊谷章 (PFU)　小林允 (ユニシス)　佐伯元司 (東工大)　阪井誠 (SRA)　坂本啓司 (オムロン)　佐藤康臣 (広島市大)
澤田武 (キヤノン)　砂塚利彦 (NEC)　高野博秀 (富士通九州通信)　田中一夫 (YIS)　田中敏幸 (シャープ)　谷口悟 (NEC)
谷津行穂 (IBM)　玉井哲雄 (東大)　中来田秀樹 (Next-F)　中小路久美代 (NAIST)　中野秀男 (大阪市大)　奈良隆正 (日立)
布川博士 (宮城教育大)　乗松聡 (野村総研)　野呂昌満 (南山大)　端山毅 (NTT データ)　藤野誠治 (富士通)　二木厚吉 (JAIST)
古川善吾 (九大)　朴学載 (大宇電子)　増子泰弘 (松下通信)　松原友夫 (Peopleware)　松本健一 (奈良先端大)　山本修一郎 (NTT)

チュートリアル： 古川善吾 (九大)　片山徹郎 (NAIST)

ツール展示： 中来田秀樹 (Next-F)　平尾一浩 (IIJ 九州)　杉田義明 (SRA)

ローカルアレンジメント： 元村直行 (安川電機)　張漢明 (K-ISIT)

ソフトウェア・シンポジウム'97　プログラム (予定)

***** 6月18日 (水) *****

12:30 − 13:00　受付

13:00 − 13:15　**オープニング**

13:15 − 14:30　**基調講演 1: Does Better Process Lead to Better Product?**　Alex Wolf (U. of Colorado, USA)

14:30 − 15:00　＜休憩＞

15:00 − 16:30　**パネル討論:「プロセス改善に向けて：管理者，作業者，技術支援 − それぞれの提言」**

コーディネータ:　中小路久美代 (NAIST & SRA)

パネリスト:　飯田元 (NAIST)，伊藤昌夫 (ニルソフト)，阪井誠 (SRA)
高木徳生 (オムロン)，乗松聡 (野村総研)

16:30 − 17:00　＜休憩＞

17:00 − 18:10　**招待講演:「二都物語：福岡と博多」**　江頭光 (博多史研究家，元西日本新聞記者)

18:30 − 20:30　**＜情報交換パーティ＞** 於マリゾンホール

***** 6月19日 (木) *****

9:00 − 10:15　**基調講演 2: 次世代インターネットを語る**　村井純 (慶応大)

10:15 − 10:30　＜休憩＞

10:30 − 12:00　**パネル討論: 2000年問題への対応はこうすればよい!?**

コーディネータ:　長谷川亨 (SRA)

パネリスト:　桝田義一 (SRA)，田中創 (AST)，和田山雅之 (CCS)，他

12:00 − 13:00　＜昼食休憩＞

13:00 − 14:30　**Session A1: 事例報告 1**

電力系統制御システム自社開発の取り組み:　丸田理 (東京電力)

西暦2000年問題社内アンケート調査の結果報告:　桝田義一 (SRA)

Agent − based Technology: A Solution for System Complexity:　椎名アルバート (安川電機)

コンピュータシステムの修正変更に伴うシステム定数関連事故の低減施策:　千田直樹 (日立製作所) 他

Session A2: 形式手法 … 研究論文

Zの状態遷移に関する性質と制約の記述と証明方法の提案
田口 研治 (奈良先端科学技術大学院大学) 他

代数型仕様言語 CafeOBJ のライブラリ設計
海野 浩 (北陸先端科学技術大学院大学) 他

形式仕様の記述スタイルに関する考察
杉山 智倫 (北陸先端科学技術大学院大学) 他

14:30 − 15:00　＜休憩＞

15:00 − 16:30　**Session B1: オブジェクト指向 … 経験論文**

ワークフロー支援システム開発へのオブジェクト指向技術の適用と評価
遠藤 誠 (東北電力) 他

オブジェクト指向による異なるアーキテクチャ間のソフトウェアの移植
上田 秀樹 (オムロン)

日本企業におけるオブジェクト指向技術の実態
松原 順 (奈良先端科学技術大学院大学) 他

Session B2: 分散・ネットワーク … 研究論文

広域コンポーネント利用支援システム WACS
高木 浩則 (日本電信電話) 他

分散処理システムにおける信頼性向上の一手法
小椋 則樹 (奈良先端科学技術大学院大学) 他

ネットワークソフトウェアのアーキテクチャ
野呂 昌満 (南山大学) 他

16:30 − 17:00　＜休憩＞

ソフトウェア・シンポジウム'97　プログラム (予定)

17:00 − 18:30　**Session C1: 業務改善 … 経験論文**

CUM 分析によるグローバルイントラネットの構築
瀧野 修 (日本電信電話) 他

システムダウン原因の徹底的な絶滅を目指したデザインレビュー
遠藤 ひとみ (日立製作所) 他

パソコン環境におけるソフトウェア障害対応プロセスの分析とそれに基づく業務改善アプローチについて
新森 昭宏 (インテック・システム研究所) 他

Session C2: 実験的手法 … 研究論文

共同ソフトウェア開発における非同期型コミュニケーションの特徴と共同生産物に与える影響の分析
村越 広享 (北陸先端科学技術大学院大学) 他

プログラムスライスに基づくデバッグ支援ツールの実験的評価
西江 圭介 (大阪大学) 他

Z,ML,Smalltalk によるソフトウェア開発の試み
張 漢明 ((財) 九州システム情報技術研究所) 他

*******　6 月 20 日 (金)　*******

9:00 − 10:00　**Session D1: World Wide Web の活用 … 経験論文**

電子バインダシステム:WWW を用いた品質システムの適用とその初期調査報告
櫨山 淳雄 (日本電気) 他

WWW を利用した教育環境の構築
阿部 雅彦 (日本電気) 他

Session D2: ソフトウェア プロセス … 研究論文

オブジェクト指向開発プロセス生成システムの開発と評価
花川 典子 (奈良先端科学技術大学院大学) 他

ソフトウェアプロセスへのツールの段階的導入方法
池田 邦彦 (オムロン) 他

10:00 − 11:30　**Session E1: ソフトウェアの信頼性 … 研究論文**

冗長ソフトウェアシステムの性能評価モデルに関する考察
木村 光宏 (鳥取大学) 他

予測平方和を用いた改良型 GMDH ネットワークによるソフトウェア故障データの解析
篠原 康秀 (広島大学) 他

FT 図を利用した安全性検証についての考察
寺田 博文 (大阪大学大学院) 他

Session E2: プログラミング言語 … 研究論文

関数型プログラムの束縛評価グラフとその応用
合田 和正 (九州大学大学院) 他

字句拡張可能なオブジェクト指向言語の設計と試作
野呂 昌満 (南山大学) 他

チャネルを同期制御に用いるコンカレント言語 Joule
辻村 仁志 (PFU)

11:30 − 13:00　＜昼食休憩＞

13:00 − 14:30　**Session F1: 事例報告 2**

構造化手法の適用と自動生成ツールの開発:　三島 康二 (ダイキン工業) 他

VisualSmalltalk 社内勉強会:　岡田 資司 (SRA)

分散オフィス環境のためのイントラネット構築手法:　廣重 法道 (富士通九州通信システム)

ソフト開発におけるクリーンルーム手法適用の一事例:　太田 正己 (富士通九州通信システム) 他

15:00 − 17:00　**クロージング パネル: ソフトウェア開発における国際協調**

コーディネータ：田中一夫 (YIS)

パネリスト: DeHua Ju(ECUST, China), KyoChul Kang(POSTECH, Korea),
他にインドと日本からのパネリストを予定

17:00 − 17:15　**クロージング**

最優秀論文賞発表ならびに表彰，次回開催予告

併設チュートリアル･プログラム

6月17日午後(13:00 ～ 17:00)および18日午前(9:00 ～ 12:00)

CMMによるソフトウェア･プロセス改善の概要 乗松聡(野村総合研究所)

ISO9000やSPICEと並んで，米国国防総省の支援を得て開発されたソフトウェア･プロセス成熟度モデルCMMが，いま日本でも注目されています．今回のセミナーは，CMMの開発元であるCMU/SEIから提供された公認テキストを使用する日本で最初の本格的入門コースです．

6月17日午後(13:00 ～ 17:00)

SCALS Tutorial & Demo 大場充(広島市大)およびSCALSデモチーム

インターネットを活用した分散型ソフトウェア開発･管理のモデルとして次第に形を整えつつあるSoftware CALSについて，その基本構想と具体化のイメージに関する講義に加えて，メーカ各社によるプロトタイプのデモンストレーションをご覧いただきます。

6月18日午前(9:00 ～ 12:00)

JAVAプログラミングと応用 小俣裕一(日本サン･マイクロシステムズ)

インターネット時代の新しいプログラミング言語として，いまホットな話題になっているJava言語の基本概念とその最新の応用に関して，わかりやすく解説していただきます．

併設ツール展示 (6月18日午後，19日終日および20日午前)

要求仕様定義支援ツールREQUARIO(カーリオ)…(株)日立製作所

Smalltalkプログラム評価ツールOOM (Visual Object Oriented Metrics)…(株)SRA

テスト支援(メモリアクセス監視)ツールPurify for WindowsNT…(株)SRA

グループコミュニケーションツールWG2001…(株)東芝

WEBブラウザと地図ソフトの緯度経度情報を連携させた不動産取引事例･地図検索システム…(株)東芝

WWWの仕組みをベースとしたファイル操作システムWing Filer…富士通九州通信システム(株)

WWWベースの複数箇所間映像コミュニケーションシステムWing Viewer…富士通九州通信システム(株)

先進のオブジェクト指向イントラネット構築システムClassic Blend…(株)オブジェクトディメンション

システム構築用並列処理計算機PIM (Parallel Inference Machine)とその開発環境Parallel Navigator…(株)安川電機

他，数社が出展を検討中です。

**********　参加申し込み要領　**********

シンポジウム参加費(会員および一般の参加費には18日夜の情報交換パーティを含みます)

SEAおよび協賛団体会員: 40,000円，一般: 55,000円，学生: 10,000円

チュートリアル受講料

	会員	一般	学生
CMM (6/17午後 & 18午前)	30,000円	40,000円	5,000円
SCALS (6/17午後)	5,000円	6,000円	1,000円
Java (6/18午前)	5,000円	6,000円	1,000円

申し込み方法: 下の申込書に必要事項を記入し，E-mailまたはFAXまたは郵便で，SEA事務局までお申込み下さい．折り返し参加証および請求書をお送りします．

〒160 東京都新宿区四谷3－12 丸正ビル5F ソフトウェア技術者協会

TEL: 03 － 3356 － 1077　　FAX: 03 － 3356 － 1072　　E-Mail: sea@sea.or.jp

.....

ソフトウェア･シンポジウム'97参加申込書(申込日：　　月　　日)

氏名(ふりがな):＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿(＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿)

種別：□SEA会員　□協賛団体(＿＿＿＿＿＿＿＿＿＿＿＿)会員　□一般　□学生

申込：□シンポジウム　□併設チュートリアル(CMM　SCALS　JAVA)希望講座を○で囲む.

参加費：＿＿＿＿＿＿＿＿円

会社(学校)名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

部門&役職：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

住所：(〒＿＿＿＿＿)＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

TEL：＿＿＿＿＿－＿＿＿＿＿－＿＿＿＿＿(内線＿＿＿＿＿)　FAX：＿＿＿＿＿－＿＿＿＿＿－＿＿＿＿＿

E-MAIL:＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

SYMPOSIUM ANNOUNCEMENT & CALL FOR PAPERS

ISFST-97

International Symposium on Future Software Technology '97

October, 29-31, 1997
Xiamen, China
Sponsored by SEA (Japan) and UNU/IIST (Macau)

This is the second symposium dedicated to the future software technology, in succession to the series of annual International CASE Symposiums held in China since 1991, organized by Software Engineers Association of Japan and other academic and industrial institutions. ISFST-97 will bring together researchers, practitioners, and educators in the leading-edge software technologies.

The theme of the symposium is "Software Technology Catering for 21st Century". Suggested topics include, but are not limited to:

> Engineering of Large Software Systems, Hybrid and Distributed/Parallel Systems, Reengineering and Reverse Engineering, Domain Engineering, Software Architecture, Object Orientation and Reusable Components, Software Process and Management, Multi-media and CSCW, Intelligent Browsing and Internet, Technology Transfer, Computer Aided Software Engineering, Computer Human Interaction,

General Co-Chairs: Keijiro Araki(Kyushu Univ, Japan)
Yulin Feng(Academia Sinica, China)

Program Co-Chairs: Masao Ito(Nil Software, Japan)
Xichang Zhong(Academia Sinica, China)

PC Members: Huwang Chen(CIT, China) Seiji Fujino(Fujitsu, Japan)
Chris George(UNU/IIST, Macau) Jun Han(Monash Univ, Australia)
Kegang Hao(Northwest Univ, China) Dehua Ju(ECUST, China)
Kyochul Kang(POSTECH, Korea) Akira Kumagai(PFU, Japan)
Yoshitaka Matumura(SRA, Japan) Andrew So(SPL, HongKong)
Tetsuo Tamai(Univ of Tokyo, Japan) Lifu Wang (Beijing Univ, China)
Syuichiroh Yamamoto(NTT, Japan) Chengqiang Ye(Zhejiang Univ, China)
Zhizhuo Zhao (Xiamen Univ, China) Youliang Zhong(Hypersoft, Australia)

Important Dates:

Submission deadline:	July 15, 1997
Notification of acceptance:	August 15, 1997
Final paper due:	September 15, 1997

Paper Submission:

Six copies of a full/draft paper or an extended abstract should be received no later than July 15, 1997 by one of the program chairs:

Professor Xichang Zhong
Software Engineering Center
Chinese Academy of Sciences
4S 4th St. ZhongGuanCun, P.O.Box2719
Beijing 100080, China
Tel: +86-10-62560341
Fax: +86-10-62560341
xczhong@sun.ihep.ac.cn

Masao Ito
President
Nil Software Corp.
2-17-7, Kinuta, Setagaya-ku,
Tokyo 157, Japan
Tel: +81-3-3749-8601
Fax: +81-3-3749-8602
nil@nil.co.jp

Paper selection will be based on originality and contribution to the theme. Each submission should include a postal address and e-mail address, if available, for a contact author. All accepted papers will appear in the symposium proceedings and at least one of the authors of each accepted paper should present the paper at the symposium.

SEA
ソフトウェア技術者協会
〒160 東京都新宿区四谷3-12 丸正ビル5F
TEL.03-3356-1077　FAX.03-3356-1072