SEAMAIL

Newsletter from Software Engineers Association

Volume 5, Number 5 1990

目　次

事務部から		1
1990年度総会報告		2
新代表幹事あいさつ	熊谷　章	4
旧代表幹事あいさつ	岸田孝一	5
ICSE に参加して		
SEAの ICSE ツアー	中川　中	6
ICSE ツアーに参加して	吉田等・野村章・国立勉・岡本修明	
	大脇隆志・久保宏志・岸田孝一	11
Eurekaソフトウェア工場	中川　中	15
全ライン・コード・レビュー実施の定着化	勝田祐輔	18
SIGENV Forum in 神戸, October'89		
全体報告	渡邊雄一	23
ソフトウェアドキュメント管理ツール	元治景朗	25
メインフレームでのドキュメント作成支援	吉村博史	26
未来予測アンケート（その2）の集計結果		27
5月のフォラーム速報		
参加者アンケート		32
ヒューマンコミュニケーションとファジィ処理	向殿政男	35
Call for Papers		
第 11 回　ソフトウェア信頼性シンポジウム		41
International Workshop on UNIX-Based Software Development Environment		43
13th International Conference on Software Engineering		45

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，1700 名を越えるメンバーを擁するにいたりました．法人賛助会員も約 60 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: 熊谷章

常任幹事: 落水浩一郎　岸田孝一　中野秀男　野村行憲　平尾一浩　深瀬弘恭　松原友夫

幹事: 青島茂　天池学　新井秀明　市川寛　臼井義美　江木康雄　岡田正志　大木統雄　大林正晴　片山禎昭　片山卓也　加藤重信　加藤重郎　窪田芳夫　杉田義明　田中一夫　玉井哲雄　玉川滋　鳥居宏次　中来田秀樹　中山照章　野中哲　野村敏次　林香　人見庸　藤野晃延　二木厚吉　北條正顕　松本省二　盛田政敏　山崎朝昭　山田淳　渡邊雄一

会計監事: 辻淳二　吉村成弘

分科会世話人 環境分科会 (SIGENV): 田中慎一郎　渡邊雄一
管理分科会 (SIGMAN): 大久保功　加藤重郎　野々下幸治
教育分科会 (SIGEDU): 大浦洋一　杉田義明　中園順三
ネットワーク分科会 (SIGNET): 青島茂　野中哲　久保宏志
法的保護分科会 (SIGSPL): 能登末之

支部世話人 関西支部: 臼井義美　中野秀男　盛田政敏
横浜支部: 熊谷章　林香　藤野晃延　松下和隆
長野支部: 小林貞幸　佐藤千明　細野広水
名古屋支部: 岩田康　鈴木智
九州支部: 植村正伸　小田七生　藤本良子　平尾一浩　松本初美　中島泰彦　後藤芳美

ＳＥＡＭＡＩＬ　Ｖｏｌ．５，Ｎｏ．５　平成２年７月９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　新宿区四谷３－１２　丸正ビル５Ｆ
印刷所　サンビルト印刷株式会社　〒１６２　東京都新宿区築地町８番地
定価　５００円（禁転載）

事務局から

☆

この号は本来６月に発行できる予定だったのですが，実質上の編集長役を引き受けてくださっている旧代表幹事の岸田さんが，５月末に欧米へ出張され，引き続いて日中シンポ（大阪），ソフトシンポ（京都）とイベントが続いたために，原稿の最終チェックと割り付けが遅れて，ついにひと月遅れてしまいました．

☆☆

新年度の役員も決まったことですから，何とかこのあたりで編集委員をつのって，各人に数ページずつ担当してもらいたいと思い，ボランティアを探しています．大阪大学の中野先生は，毎月の SEA 関西のレポートを約束して下さいました．名古屋の筏井さんと米国コロラドにお住まいの酒匂さんは，書評欄と時々の地方だよりを担当してくださるそうです．各分科会や支部の世話人のみなさん，毎月の活動報告や今後の予定をレポートしてください．お待ちしております．

☆☆☆

今月号には，5月 15日に行われた総会の報告，新幹事の方たちの名簿に加えて，今年度で代表幹事が交代されましたので，新旧代表幹事のあいさつを載せました．また，春の ICSE ツアーの報告や，勝田さんの品質管理に関する論文(去年の信頼性シンポでの発表を整理されたもの)をいただきました．さらに，3月に行なった未来予測アンケート（Part-II）の結果と，5月のフォーラムの速報も収録して，なんとか１冊分のページを稼ぎました．

☆☆☆☆

第３回日中シンポジウム（６月５日，大阪），そしてソフトウェア・シンポジウム'90（６月６～８日，京都）と，今年前半に予定された２大イベントがそれぞれ成功裡に終わって，ほっと一息です．関係者のみなさん，おつかれさまでした．次は，秋のイベントですね．若手の会そして信頼性シンポジウムへの申込みを，よろしくお願いいたします．

☆☆☆☆☆

ところで，**だいじなお知らせ！** SEA 事務局は７月の初め（７～８日の週末）に引っ越します．新しい住所と電話番号は次の通りです：

〒160 東京都新宿区四谷 3 - 12 丸正ビル 5F
TEL: 03 - 356 - 1077
FAX: 03 - 356 - 1072

今度のオフィスは，地下鉄丸の内線「四谷３丁目」駅を出て徒歩１分の便利なところです．スペースも少し広くなります．この SEAMAIL がお手もとに着くころには，もう引っ越しも終わっているはずです．おひまの折りは，お遊びに（いや，事務局の雑用のお手伝いに）おいでください．

☆☆☆☆☆☆

1990年度総会報告

今年度 SEA 年次総会は，さる5月15日，5月のフォーラムの後，機械振興会館で開催された．

本年は，会則（第24条）「通常総会は毎年1回5月に開催する」通りに行われたわけである．

出席者は33名，他に委任状提出者278名で，合計311名．定足数101名を大幅に上回り総会は成立した．

総会では，まず前年度の会計決算（表1および表2）が前年度代表幹事の岸田孝一さんから報告され，満場一致で承認された．引き続き，前年度幹事会において検討された，新年度の役員候補者リスト（次ページ）が提示され，原案通り承認された．その後，新代表幹事の熊谷章さんにより今年度予算案（表3）が提示され，これも異議なく可決された．

残った時間で，新幹事の挨拶，参加者全員の自己紹介などがあり，和やかに閉会した．

表1 収支計算書

自 1989年4月 1日
至 1990年3月31日

支出の部	
人件費	1,256,327
事務所費	4,703,815
印刷費	5,844,894
通信費	3,074,751
会議費	340,375
広報費	0
消耗品費	190,012
雑 費	2,042,433
当期収支差額	687,892
合 計	18,140,499
収入の部	
新入会費	1,890,000
更新会費	5,493,764
賛助会費	3,600,000
寄付金	3,677,503
雑収入	3,479,232
合 計	18,140,499

この収支計算書は，公益法人会計基準に示されているような収支ベースのものではなく，
損益ベースによって作成してある（企業会計の損益計算書に該当するものである）．

表2 貸借対照表

1990年3月31日現在

資産の部	
流動資産	
現預金	7,409,746
未収入金	1,405,000
固定資産	
敷 金	2,400,000
合 計	11,214,746
負債及び正味財産の部	
流動負債	
未払金	518,000
正味財産	
正味財産	10,696,746
(当期収支差額	687,892)
合 計	11,214,746

表3 予算書

自 1990年4月 1日
至 1991年3月31日

支出の部	
人件費	2,400,000
事務所費	3,000,000
印刷費	6,000,000
通信費	3,600,000
会議費	360,000
広報費	240,000
消耗品費	360,000
雑 費	120,000
当期収支差額	1,120,000
合 計	17,200,000
収入の部	
新入会費	1,800,000
更新会費	7,000,000
賛助会費	4,200,000
寄付金	2,400,000
雑収入	1,800,000
合 計	17,200,000

1990 年度　ソフトウェア技術者協会幹事名簿

代表幹事	熊谷　章	ＰＦＵ＆富士通
常任幹事		
	落水浩一郎	静岡大学
	岸田　孝一	ＳＲＡ
	中野　秀男	大阪大学
	野村　行憲	岩手電子計算センター
	平尾　一浩	ヒラタソフトウェアテクノロジー
	深瀬　弘恭	アスキー
	松原　友夫	日立ソフトウェア･エンジニアリング
幹　事	青島　茂	日本サンマイクロシステムズ
	天池　学	カシオ計算機
	新井　秀明	日本電子開発
	市川　寛	電算
	臼井　義美	日本電子計算
	江木　康雄	大阪大学
	岡田　正志	日本電気ソフトウェア
	大木　統雄	オーケーマイクロシステムズ
	大林　正晴	管理工学研究所
	片山　禎昭	日本システム
	片山　卓也	東京工業大学
	加藤　重信	凸版印刷
	加藤　重郎	三谷コンピュータシステム
	窪田　芳夫	東京電力
	杉田　義明	ＳＲＡ
	田中　一夫	山一コンピュータセンター
	玉井　哲雄	筑波大学
	玉川　滋	日本ナレッジインダストリ
	鳥居　宏次	大阪大学
	中来田秀樹	ネクストファンデーション
	中山　照章	富士通ソーシアルサイエンスラボラトリ
	野中　哲	ライフボート
	野村　敏次	日本電子計算
	林　香	ＳＲＡ
	人見　庸	ジェーエムエーシステムズ
	藤野　晃延	富士ゼロックス情報システム
	二木　厚吉	電子技術総合研究所
	北條　正顕	シスプラン
	松本　省二	読売コンピュータスクール
	盛田　政敏	ケーシーエス
	山崎　朝昭	新日鉄情報通信システム
	山田　淳	東芝
	渡辺　雄一	アスキー
会計監事	辻　淳二	辻システム計画事務所
	吉村　成弘	公認会計士

SEA へのさまざまな思い

熊谷　章
(新・代表幹事)

SEA にはいろいろな思い出がある．

私の会員番号は 860358 だから，入会は86年ということになる．そのころは，PFU の研究開発部で新しいコンピュータシステムの研究開発をしていた．この仕事は大変楽しく，それからの約４年間の経験が，ここ暫くの私の人生観を左右している．いま座右の銘にしている，「不易流行」，「人間性中心主義」，「マイノリティ」という３つのスローガンもその頃考えたものである．

新しいコンピュータシステムを考えるためには，時代性や社会の動き，特にその時代の価値観を知る必要がある．つまり，一般の社会現象や他社の事業にも興味が持てなければならない．自分の会社の中に閉じこもっていては駄目だと強く感じた．

私は，いわゆる書斎作家や評論家が嫌いだ．だから，宮澤賢治や開高健等の思想や生き方には，心を打たれる．

この間，東京ビジネススクールの夏目学長から面白い話を聞いた．教養についての話である．勝手に自己流に解釈して，その概要を紹介する．

現代のビジネスマンは，教養人である必要がある．競馬，マージャン，ゴルフ，女だけを話題にするようではダメだ．もう少し高尚に，世界平和や米ソ交渉，日中友好といったテーマについて，語ることができなければならない．YES-MAN では駄目で，はっきりと自己主張を持たなければならない．会社などはいつでも辞める位の覚悟を持つことが重要である．そうした覚悟があればこそ，自分の考えや信念を上司や幹部にも正々堂々と言明できるのだ．人生６０年の計を自分の中に確立し，その信念にもとづいて生きて行く心構えが大切なのである．

このとき，一番大事なものは何かと言えば，友人である．友人は人生の宝である．友人なしには，ビジネスも人生もすべてがうまくいかない．そうした人生の宝を持つには，つねに相手の立場に立ってものを考える謙虚さ（心の余裕）と，人間に対する深遠な理解が必要である．

古来，人間には五欲あると言われている．食欲，眠欲，性欲，健康欲，自己実現欲である．

この中でもっとも強い欲望は何か．教養のレベルによって多少の差はあるだろうが，一番強いのは自己実現欲であろう．船長が船と共に死ぬのも，社会的な事件で多くの人々が自ら死の途を選ぶのも，皆そのためである．

したがって，教養のある人とは，相手の自己実現欲の対象が何かを洞察し，そのために助力してやれる人のことをいう．自立していない人間には，なかなかそういうことはできない．

教養は栄養と同じように，外部から摂取するものである．まず第一に，自分が所属している組織の外にいる友人との付き合いが大切である．栄養学にたとえれば，生野菜のようなものだ．異なった環境や視点，価値観から物事を見直し，かつ意見交流を行なうことは，まさに知的な活動の源泉である．昔から，交通の頻繁な所に多くの文化が繁栄したことを見れば分る．

２番目の方法は読書である．書物には，先人たちの知恵や知識が一杯詰まっている．栄養学にたとえれば，缶詰である．読書によって，時空を自由に闊歩でき，自在な考え方が身につけられる．

教養を身につける３番目の方法は一人旅である．夜汽車に揺られて車窓から町の灯を独りで眺める．そんな旅がいい．見知らぬ街，見知らぬ人々，見知らぬ風土，見知らぬ生活を，その土地からはみ出た目で眺め，その思いを我が身に移すことが面白い．さまざまなことに気づく．日常生活に埋もれ気づかずにいた多くのことがらを発見できる．いままでにない自分の方向が見えてくる．栄養学にたとえれば香辛料にあたる．これがないと，いま一つピリッとしたところに欠ける．

夏目さんの話の内容はこんなものだったと解釈している．私が新しいコンピュータシステムを研究開発していたときも，同じようなことを考えていた．SEA に加入したのは，そんな時期であった．そして，SEA はまさに教養を磨く場であった．産学官さまざまな世界に身を置く研究者や技術者が渾然一体となった交流の場が，（しかも国際的に）広がっている．いまや，時代は，Beyond The Company, Beyond The Nation である．

自立を説いた福澤諭吉，「雨ニモ負ケズ」の宮澤賢治らの志を継いで，新しいソフトウェアエンジニア育てる場としての SEA ができ上がりつつある．今後，日本を代表する数多くのソフトウェアエンジニアが SEA の中から輩出することを強く信じ，また期待している．微力ながら，SEA のこのような方向への発展のために尽力したい．よろしくお願い申し上げる次第である．

SEA — これまで，そしてこれから

岸田 孝一
(旧・代表幹事)

去る５月１５日の総会で，私の代表幹事としての任期は終わり，熊谷章さんにバトンを引き継いだ．この機会に，SEAのこれまでを振り返り，あわせて未来についての個人的な展望を語っておきたい．

SEAは，これまでの日本の社会にはなかった組織を目指してきた．そして，その狙いは着実に達成されつつあると感じられる．

新しく入られた会員の方々のなかには，SEA発足当時のことを御存じない方も多いと思うので，このあたりで簡単に歴史を振り返ってみるのもよいことであろう．

SEA結成の母体になったのは，初代の代表幹事であった鈴木弘さんや私などが中心になって運営していた旧ソフト協（現在のJISA）の技術委員会である．この委員会は，業界親睦団体としてのソフト協やJISAの枠をいささかはみだした格好で活動していた．つまり，あくまでソフト業界に働く技術者を中心としながらも，メーカやユーザの人々，あるいは大学の研究者もまきこんだ形で，さまざまな会合を組織した．今年で１０回目を迎えるソフトウェア・シンポジウムは，もともとこの委員会が企画して始めたものである．また，「若手の会」を初めとするワークショップの概念を日本で初めて具体化したのも，この委員会であった．

活動の実態と組織の形とを一致させるべく，SEA設立の話が具体的に論議されはじめたのは，ソフト協がセンター協と合併してJISAが生まれることになった１９８４年ごろのことであった．それから迂余曲折の経緯（「これは新しい形の労働組合ではないのか？」とか，「技術者のトラバーユを助長することになりはしないか？」とかいった業界の経営サイドの誤解を解く必要があったりした！）を経て，１９８５年の暮れにスタートを切ることができた．

学会や業界団体といったすでに社会的に認知された形ではなく，任意団体としてのソフトウェア・エンジニアの集まりであるSEAが，はたしてうまく運営できるのかどうかについては，多くの不安があった．われわれがモデルとして考えていたのは，欧米におけるボランティア・ベースのプロフェッショナル・ソサイエティであったが，日本の社会にはそうしたメカニズムが確立されていなかった（現在でもまだ十分に確立されているとはいえない）からである．

実際，設立して１～２年の間は，分科会や支部の運営も何となくぎくしゃくとしていたように感じられる．しかし，それぞれの世話人の方々の努力のおかげで，時間がたつにつれて，それも次第に安定してきた．

私が鈴木弘さんの後を受けて２代目の代表幹事になった１９８８年の春には，すでに会員数も千人を越え，賛助会員も４０社ほどになっていた．ただ大きな問題は，機関誌SEAMAILがまだ定期刊行の波に乗らず，またその見通しも立っていないことであった．また，財政的には，何百万円かの借入金が残っていた．

この２年間，いろいろな努力を試みたにもかかわらず，機関誌編集部はいまだに難破状態にある．代表幹事の責務（SEAのようなボランティア組織では，各種DMの印刷・発送といった雑用一切を含んでいる）から開放された（？）今年は，何人かの方々のお手伝いをいただいて，これを何とかしたいと考えている．SEANMAILは，会員のみなさんからの投稿で成り立つ雑誌である．現状は，ワークショップその他のイベントの報告（これも実際に参加できない会員各氏へに情報提供という意味では大切ではあるが...）や，アンケート集計で何とかページを稼いでいるにすぎない．よろしく御協力をお願いしたい．

財政面の問題は，とりあえずこの２年間で借入金の返済を終ったので，ほっと一息という状態である．しかし，これだけ大きくなってきた組織を現在の事務局（ほとんど中島さん１人を何人かのボランティアで支援している）で支えて行くのは苦しいので，組織の整備とそれにともなう予算規模の拡大に関して，そろそろ何らかの長期的なビジョンを考えて行かなければなるまい．

さて，最後に将来の展望だが，基本的にSEAのようなボランティア組織の活動エネルギーは，個々の会員のそれの総和以上にはならないと思われる．その意味で，これからのSEAにとって大切なのは，本来混沌とした形で存在している個人のエネルギーを一定の方向にまとめあげられるような活動（分科会，ワークショップ，その他のイベント）を，いくつかセットアップすることであろう．そうした活動の起爆剤としての役割を担うアクティブ・メンバーが何人あらわれるかがポイントだと思う．過去５年間の経験は，私を陽気なオプティミストに変えてしまった．今年もまた，多くの新しい友人たちと出合い，新鮮な議論を肴においしい酒が飲めることを期待している．

seaのicseツア

中川 中

1 はじめに

第12回国際ソフトウェア工学会議(以下、icse12)を機に組まれた今回の欧州技術調査団は、

- Berlin の Eureka ソフトウェア工場(以下、esf)本部
- Bath の Praxis 社
- Nice の icse12 会議場

の3箇所を訪問し調査・交流を試みた。本稿は調査団の一員である筆者の目から見た各訪問先ならびに調査行全体の模様をまとめたものである。筆者の関心は主に技術面にあるので、体制や機構・運営上の話題は他に譲り、研究開発の内容自体に的を絞ってある。

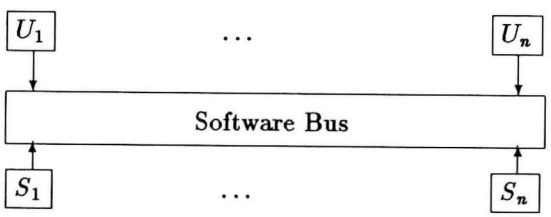
2 esf

esfはその名前にもあるとおり、ソフトウェア生産を工業化すること[1]を目的としており、そのための基盤技術全般を— しかも適用分野を特定することなしに — 対象とする野心的なプロジェクトである。今回の訪問では、全体のアーキテクチャ[2]のほか、特にプロセス・モデルと再利用の分野における研究成果が披露された。

(1) アーキテクチャ

esfにおける開発環境は、

図 1:



- ユーザ・インタフェイス[3]要素群
- ソフトウェア・バス
- サービス要素群

から構成される。エディタ、デバガ、コムパイラ等の通常のソフトウェア開発ツールは主に対象物(オブジェクト)への操作からなる純粋な機能の部分と利用者とのインタラクションを規定する部分とに分解され、それぞれサービス要素、ユーザ・インタフェイス要素と呼ばれる。ソフトウェア・バスは両者を媒介する(図1)。

この構成の最大の利点は、バスと各要素とのインタフェイスを形式化することによりツールの入れ換えや追加を容易にすることにある。しかも、ユーザ・インタフェイスは機能自体と分離されるため、こうした環境の改良や拡張は利用者の再教育を必要としない。また、対象物の統合性をサービス要素に専任し、プロトコル定義で各要素を結びつけることにより、ハードウェア上の分散環境にも自然に馴染む。

ユーザ・インタフェイスを独立させる試みはUIMS(User Interface Management System)等の名前で近年盛んであり、esfのアプローチもその文脈で捉えることができる。利用者の視点からは、デバガ等

[1] Software Factory という言葉はこのように一般的な意味で使われており、倭語の「ソフトウェア工場」から通常想起されるイメージとは必ずしも一致しない。

[2] この言葉はソフトウェアの構成・組織方法という意味で使われている。

[3] この言葉のニュアンスは筆者の忌み嫌うところであるが、本稿ではesfの用法にしたがった。

の機能は容易に拡張でき、しかも使用方法自体は一定していること、ツールの種類に拘らず一定の使用方法が提供されること、また使用方法それ自体が容易に改良・個性化 (customise) できること、が望ましい。一方、分散開発環境におけるツールの利用形態を考えると、バスを軸としたツールの統合は柔軟で巧妙な方法といえる。

問題は、この構成がどのように具現するかである。特に、バスと各要素とのインタフェイスにはデータ形式変換を含めやや過大な負担が期待されており、実現性ないし実現した場合の実用性は必ずしも明らかでない。アーキテクチャのデモンストレーション (以下、デモ) は 2 種類のデータ形式の相互変換を行なうもので、問題の所在は的確に把握されていたようである[4]。

(2) プロセス・モデル

Wilhelm Shäfer 等による esf におけるソフトウェア・プロセス・モデルの研究は、ソフトウェア工場における開発に従事する者 (以下、開発者) の視点を全体プロセスとして統合し、もってプロジェクトの組織や管理に貢献しようとする試みである。全体の印象は非常に意欲的であり、包括的である。モデルの構築、そのプログラム化[5]、その特例化 (instantiation)、またモデルのプロジェクト (工場) 組織における位置付けの機構をそれぞれ考案し支援システムとして実現してしまおうというのが究極の目的である。

これまでに実装されたシステムのデモではプロセスの進捗を監視し支配するツールと作業内容、その作業の行なわれる文脈および作業対象を受けまた作業結果を報告するツールが動作していた。いずれもとりたてて目新しい着想を踏まえたものではないが、実装したという事実は評価せざるをえない。稼働環境上の問題ゆえに個々の動作は鈍かったが、この類のツールではそれは問題ではない。問題はより本質的で、誰にとって役に立ちそうかである。これらのツールが行なうのは、一言では

コミュニケーションを形式化する

ことである。管理、とくにプロジェクトの進捗管理にはおおいに貢献するであろうが、ただそれだけならば、という疑問が強く感じられるのである。

(3) 再利用

ソフトウェア部品の再利用を実現する手段として大きく生成方式と構成要素方式[6]とがあるが、esf は後者の方式によっている。

特徴的なのは部品の整理の仕方である。再利用の促進で重要なのは

- 再利用可能な部品を作ること
- 必要な部品を容易に見つけ出せること

に尽きるが、後者の問題は部品を格納する書庫 (library) の構造と検索機構に帰着する。この問題をデータ指向のタームに換言する[7]と、個々の部品をどう特徴づけるか、となる。任意個の索引 (indices) を付加する、分類木あるいは (重複分類を認めるという意味で) 分類有向非環状グラフ (dag) を使う等の幾つかの選択のうち、彼らは複数分類基準による手法を採用した[8]。

聞きそびれたのは次の 2 点である。

- 一つの分類基準自体は平ら (flat) なのか、木のような構造を認めるのか。
- 分類基準相互の間の関係を考慮するのか、互いに無関係という前提に立つのか。
- それぞれの (for each) 分類基準につき、すべての部品が何らかに分類されるのか (すなわち、すべての分類関数は全関数なのか)。

最後の問いには否という回答が推測できるが、前 2 者は全く不明である。さらに、これらの点が明らか

[4] 筆者はこのデモを真剣に観察したわけではないので、この点はやや憶測による。

[5] 彼らの定義によれば、プロセス・モデルとは人間が解読可能な記述であり、プロセス・プログラムとは機械が解読可能な記述である。この定式では、おそらくプログラムはモデルの部分集合であり、それを特徴づけるのは観察者に要求される知能のレベルのみである。記述を解釈することにより得られる意味の世界 — 通常はそれこそがモデルなのであるが— を無視している点が気にかかる。

[6] これらの用語はそれぞれ、(a) 汎用的 (generic) な部品をその都度特化する方式、(b) すでに完成した部品を使う方式、を意味する。これらは相互に排他的というより、選択スペクトラムの両端にあるとされるべきである。

[7] 正確には、この言い換えは問題の一部についてである。ライブラリの構造自体の差異、例えば木構造かネットワーク構造か、が検索効率に影響することは勿論である。

[8] 例えば、ある部品 e は分類基準 C では c_e に $(C(e) = c_e)$、別の基準 C' では c'_e に $(C'(e) = c'_e)$ 分類される、というように。

になった上で、(a) 部品の追加の際に生じ得る無矛盾性等のチェック、(b) 検索手続きの完全性や検索した部品の最適性、(c) 追加・検索の (実際上の) 労力を問わねばならない。筆者は再利用の分野に親しくないために他との比較はできかねるが、着想を聴いた限りではとりわけ優れた研究とは思われない、というのが感想である。

3 Praxis 社

England は Avonshire 、一大産業ベルトを形成する M4 からやや南に下った所に位置する景光の地 Bath に本拠を構える同社は、「高級感」を漂わせるイメージで名を上げる。このたびの訪問では品質管理、プロセス記述および形式方法 (Formal Methods 、以下、fm) の適用の 3 テーマをめぐり議論がすすめられた。丸一日の日程のうち後半はサブグループ単位の討論であり、筆者はプロセス記述のグループに参加したこと、および興味の程度の差異により、以下では品質管理のテーマは割愛する。

(1) fm の適用

fm という言葉は雑多な対象を意味すべく使用されるのが慣例である[9]が、Praxis 社の場合、具体的には Z および vdm を用いている。もっとも、以下に述べるように、特定の言語ないし方法自体はあまり問題ではない。

Z 等モデル意味論に基づく方法は — 程度の差はあれ— 広く知られている反面、実プロジェクトに適用されたという例は少なく、伝え聞くところの適用例も概ね政策上の理由からであったりする。同社の経験はその意味で貴重であり、とりわけ

- どうして使えるのか
- どのように使っているのか

に関心が持たれた。この関心の背景には、この類の方法は非現実的・難解・非生産的・硬直的・不自然その他の口実で管理者も開発者も疎むのが一般的である、という事実がある。

結論を急ぐと、そもそもこれらの方法を適用できる理由は (a) コンサルテーション・教育指導等の活動を通じて普及に努めていること、(b) 開発方法を選択できる立場にあること、に集約される。(b) については羨むほかはない。(a) の利用者教育の点は重要である。Z を使えることは Z の記法を知ることとは異なる。後者は時分の単位の話であり、前者は月日の単位の話であると Praxis 社は正しく認識している。実際の教育法について知る機会を逸したのが筆者には悔やまれる。

次に、使用形態であるが、これについても明確な結論が得られた。fm(の一部) の効用は一般に

- 解析手段が多様であること
- 伝達手段として曖昧さがないこと[10]

とされるが、おそらく最大のメリットは

思考手段として適すること

である。詳細は知り得なかったが、Praxis 社の場合もこの点を認めており、とくに自然言語との対比で Z や vdm のタームで考えると不完全さ、曖昧さ、未解決の部分、矛盾等が自ずから明らかになる、という経験が指摘された。このメリットを強調すると、処理系の良否、さらには有無さえも、導入の是非にとって決定的ではない。また、記法や方法の差異も副次的な問題に過ぎないという[11]。

(2) プロセス記述

Clive Roberts 等によるプロセス・モデルやその記述においては、

- 人の要素を考慮すること
- (とくに) 外生的な変化に即応できること

の 2 点が強調される。モデルの中心にあるのは役割 (rôle) と相互作用 (interaction) である[12]。この役割と

[9] 文脈にもよるが、少なくとも意味 (モデル) の世界を厳密に定義する諸方法はそれら以外とは区別されるべきである。例えば、Rigorous Methods とか。

[10] この系として、検証の際の対置物として (against which) の仕様を記述するのに適すること、も挙げられる。

[11] もっとも、思考手段とみなすからには表現力や表現方法は決定的に重要である。この点、なぜ vdm のような低レベルの記述で満足できるのか筆者には不思議である。

[12] これらの用語は使用者によりそれぞれニュアンスが違う。事実、今回も用語の定義をめぐり約 2 時間程議論が続いた。特に、

は、人間、ツール、物、活動等多様な範疇を包括する。役割はそれぞれ固有の (可変な) 行動様式 (behaviour) を持ち、ソフトウェア・プロセスは既与の目標 (goal) を達成するために (a) 手法の選択[13]、(b) 手法の特化、(c) 役割の起動、という手順を経て実際に動き始める。このあいだ、可変性が強調され、新たな役割の導入や既存の役割の行動様式の変更をつうじてプロセスが変化する。

ここまでは何とか説明できるが、あまりに不明な点が多く、このモデル[14]については全く判断しかねる。特に、核となる概念の定義が曖昧である。憶測はつくのであるが、突然に憶測を覆すような用法が出現しすっかり混乱したりする。最終的にはこのモデルによったとされるプロセス記述言語の単位要素 (primitive) から推測する他はなかろうが、筆者はそれを吟味していない。そのうち、折あらば。

デモで稼働していたシステムを見る限りでは、esf での場合と同様に、プロセスを記述しまた「実行」[15]して一体どうするつもりか、という疑問が残る。ソフトウェア・プロセスはプロジェクト管理の下位の分野なのであろうか。

4 icse12

今回の当会議は開催地が France にあることから、(実践至上主義派[16]に対して) 理論派の優勢が目立つであろうという期待があった。プログラムに現れる論文タイトルを総覧して然りである。経験レポートと称されるカテゴリが設けられたが、この優勢を中和する程の存在感はない。

実際にセッションにでた印象はこの期待に違わなかった[17]。以下でいくつか目についたセッションを取り上げる。

(1) 分散プログラムの構造

Barbara Liskov 女史による 2 日目の全体セッションである。筆者は 1 時間 15 分に及ぶ女史の講演の大半を眠りの中で聴いていた。にも拘らずこのセッションを取り上げた理由は、二度とこの類の講演を全体セッションに持ち込まぬようにと訴えるためである。ここで「この類」というのは、チュウトリアルに相応しい内容のみからなる、という意味である。このようなスピーチを全体セッションに持ち込むことは聴衆の大半を当惑させる。分野を限定すること自体には問題はない。あるのは内容が一般的関心 (general interest) の対象とならない点にある。女史は筆者には初見であり、それなりに期待が合っただけに残念である。

(2) プロセス・モデルその 1

このセッションは筆者の発表を含んでいたため、敢えて取り上げる。筆者はソフトウェア・プロセスにおける産物の変遷過程を形式化する試みについての概略を例を交えずに話した。もう一人の発表者は Ada をベースにした漸増型 (incremental) のプロセス・モデルを使用経験を交えて披露した。単純化していえば、題材およびこの特定場での発表の両面で前者は抽象的・形式的・即理的であり、後者は具体的・非形式的・即物的であるという対照を成すこと、司会者 Perry の指摘のとおりである。終始居座っていた聴衆はこの予めの指摘なくしては — あるいは、指摘があったにせよ — 面食らったことと思われる。ただ、質疑応答の量で測るかぎり、まずまずのセッションではなかったか。

(3) 形式検証用ツール

論理的動画化 (Logical Animation) と銘うたれた一番目の発表は実に退屈であった[18]が、次の順序ソート・ホーン節仕様支援システムの発表は興味深い話題をいくつか含んでいた。詳細は省くが、完備化手続き (completion procedure) を実装したシステムを OBJ3 に類似した言語用に作り上げてしまったのである。後に行なわれたデモも観察したのであるが、

役割の抽象性、実体との結び付き、あるいはモデルにおける役割とモデルの外の世界の対象との関連、で相互の誤解を解消するのが大変であった。

[13] それぞれの役割をどう組み合わせるか、そしてどのような構成をとるか、を手法と呼んでいる。

[14] モデルなのかどうかも不明である。

[15] この言葉 (execute) はプロセス・プログラミングの世界で頻繁に現れるが、明らかな乱用であり、使うにしても引用符つきにするのが妥当である。

[16] ターミナルと戯れ合うのに夢中な者、能く動くことを良いシステムの十分条件とする者、金儲け即ちプロフェッショナルと解する者、自らの不能を弁護するために理論を罵倒する者、これら諸々の者共を軽蔑を込めてこう呼ぶ。

[17] ただし、筆者はそれらしいタイトルの発表ばかりを選んで聞いたので、この印象は殊更に個人的であるのかもしれない。

[18] 今更何をと言われんばかりに積木の世界の例で目新しくもない論理体系の話しを延々と続け、肝心の動画化の話しを殆んどしなかった。

Harald Ganzinger 等は非常に全うな、重要な研究を行なっているという印象を受けた[19]。システムの核になる部分は揃っているようなので、すでにある単体インタフェイスのチェックや書き換え規則の完備化などの機能に加え帰納法を用いた種々の解析機能をとりこむことも時間 (とセンス) の問題と思われる。筆者等の画策しているシステムが参考とするところ大である。

(4) 雛型作り

後半の抽象的及び具体的単体の混合、仕様・開発・雛型作りと題された Christina Choppy の発表を聴いた。代数型言語を用いたソフトウェア開発の話題であり、さきの順序ソート・ホーン節の発表と同じく、筆者にとって直接の関心事である。様々な開発段階にある単体を組み合わせて実行する機構を形式化し、厳密なモデル意味論を与えたもので、代数仕様ないし抽象データ型に基づく開発に頂上下り型 (top-down) や段階的詳細化の手法を取り込むことを目的とする。fm の適用を考える際にこの種の研究は重要であり、物作りや実験に取り組むことはそれとして手法を厳密に検討することも怠ってはならない、という定言にそったよい姿勢が伺える。彼女等は最終的には各単体は Ada 等既存のプログラミング言語程度の表現方法で記述されると考えている。このあたり、選択の当否の判断は難しいが、ひとつの現実的な見識ではある。

(5) 全体について

ソフトウェア工学一般を対象としかも理論的にあるレベルを越えるような会議を組織するのは困難である。この困難さは今回のセッションのタイトル及び組み方に端的に現れている。たとえば、「形式検証用ツール」という上のセッションのタイトルはいずれの発表にも相応しくない。また、プロセス・モデルその 1 では同じ聴衆を対象としているとはとても思えない発表が一括されている。立場やテーマを異にする研究者達が自らの位置付けや相互の意見交換のために集う、という趣旨からは会議全体のプログラムが変化に富むのはよいことであるが、個々のセッションにはもう少し統一性が欲しい。

5 その他、あるいは余録

今回の調査行でもっとも目についたのは、Dortmund の存在である。Ganzinger 、Shaëfer とも Dortmund を拠点ににしており、それぞれかなり活発に動いている。ドイツはもともと研究活動では定評がある[20]だけに、本格的にプロジェクトを組んでこうした開発拠点ができ始めると恐ろしい。この分野でもドイツの時代がやってくるのであろうか。

[19] 発表者が Renate というドイツ美人であったのはこの文脈では関係がない。

[20] 少なくとも、抽象データ型回りの論文を漁るとドイツ発のものが圧倒的に多い。

ICSE ツアーに参加して

吉田　等
千代田情報サービス（株）

1. ESF

地味ではあるが着実に前進しているとの印象を受けた．ただし，問題はやはり最終局面で First Product の開発に成功するかどうかである．多くの協同プロジェクトがその段階で失敗しているので，ESF がどのようにしてその壁を乗り越えるのかが，興味深い．

2. ICSE

はじめての参加であるため，良し悪しの判断などは到底できないのだが，中川氏のレポートにもあったように，1つのセッションの中での発表内容がおたがいにかけはなれているケースに，いくつか遭遇した．限られた時間やスペースのせいで，仕方がないのかもしれないが，セッションの統一性をもう少し考慮したらどうかという印象を受けた．

3. その他一般的な感想

やはりベルリンの印象は強烈であった．つい最近まで，あの壁が人間の往来を阻んでいたとはとても信じられない．よく，ソフトウェアの世界は日進月歩ですごいスピードだなどといわれるが，いまや，国際政治のスピードはそれさえも追い超してしまったように感じる．

野村　章
昭和コンピュータシステム

1. ESF

HAPPI というシステムは，異なる OS で動く複数のマシンと多数の開発者を含むソフトウエア開発プロジェクトに，ソフトウエア工場の環境を提供する．このシステムは1つのプロセス·モデルにもとづき，コミュニケーション·アーキテクチャを駆使して構築されていた．

また，SITE というシステムでは，人·仕事·ツール·データを結びつけ，ソフトウエア部品の集積構築機構である Software-Bus を介して，プロジェクトの動きをモニタすることができ，マネジャとプログラマの協調作業が支援されるようになっていた．

いわゆる Formalism は，このようなソフトウエア開発のマクロなプロセスについてはうまく働いて，実現性·実効性があるように思えた．

2. Praxis

ESF と異なり，ここでは広範な組織モデルを VDM や Z を用いて作り，これをプロセス·モデルとして実行しようとしていた．組織は営業·ソフトウエア開発チーム·顧客の資材購買支援部門·経理部門·品質管理部門などを含む．

ソフトウエア開発の現場では，顧客への対応·プロジェクト管理·コスト管理などに追われて，プロジェクトごとに管理手法が異なり，開発期間やコストが変動するのが常である．キッチリした企業単位のプロセス·モデルがあり，状況の変化に応じてその都度モデルを再実行してフレキシブルな組織的対応がとれるツールができれば，きっとすばらしい違いない．

コンピュータとツールを用いて，ソフトウエア開発全般に対するマクロな統制機能を働かすことができれば，ソフトウエア·プロダクトの品質も保証されるという Praxis 社の主張には賛同したい．

3. ICSE

ソフトウエア開発に工業化技術を取り入れようという目論見からすれば至極当然の成行きではあろうが，相変わらず Formalism の主張が巾をきかせていた．

「ソフトウエア工学はソフトウエア技術の進歩に貢献しない」などと手厳しい意見もあるが，いわゆる Formalism については，マクロなソフトウエア·プロセスのレベルとスペシフィケーション·レベル（あるいはプロダクト·レベル）とでは，その実用性において大差があるように思える．前者の好例は ESF や Praxis の試みであり，後者はプロダクト中心に数学モデルを用いた Formalism の議論である．

Formal Method が本当に有効なのは，コンパイラ開発のようなシステム·ソフトウエアの分野でしかないのではないか？　したがって，このテーマは，アカデミズムの興味対象にしかすぎないように見える．

いずれにしても，プログラム一般を扱うプロセス·プログラミングは，実用性に乏しい．メンテナンスや再利用のためのエンジニアリングの話は大変興味深かったが，もっと具体的な方法論について聞きたかった．最近話題の IBM の CASE ツールやレポジトリについて何等言及がなかったのは，ソフトウエア工学の興味の中心が汎用大型機からワークステーションに移ってしまったからであろうか？

部品化技術やオブジェクト·データベースによるプロセス情報の知識ベース化，AI 技術による部品検索や知識ベー

スの情報検索など，実用性の高い技術に関する研究や議論がもっと増えて欲しい．問題領域を狭めた実用的研究がもっとあってよいと思う．

4. 雑感

コンピュータの世界には，自分の関与しているパラダイムやツールにとららわれて，社会や法律･経済･文化などの事象に目をつむり，狭い「蛸壷」の中で肩肘張って外界からのアクションをかたくなに拒絶する人が多い．

技術移転は非常に重要であるが，このような人が実務を担当している部署に自主的に行わせるのは至難の技である．SRA の岸田さんの持論である Play-Ground の必要性は痛感しているが，いわゆる「遊び心」を含んだ意味での Play-Ground が保持できる企業は，かなり大手の余裕のあるところに限られる．

専門職として外の技術を漏れなくウオッチして，これと思うものについて自らその有効性を評価し，優れたものを担当部署に積極的に技術移転する人間がいればよい．残念ながら「遊び心」を持って創造性を発揮できる要員がいまだに少ないのも事実であり，トップの期待が大き過ぎても，せっかくの組織が潰れることがある．読者諸君の中で腕に自信のある人は是非当社へ“とらばーゆ”されたい（これは本音）．

国立　勉
ＮＴＴソフト研

1. ESF

ソフトウエア･バスは，ツールの統合化を容易にすることを狙ったものと考えられるが，どの程度まで標準化が達成できるか興味がある．

ヨーロッパは，多国間共同プロジェクトを熱心にすすめていることを実感した．ESF の場合，開発を参加企業が個別に実施しているとのこと．全体のオーガナイズにどんな工夫がなされているか興味がある．

2. ICSE

ICSE の参加は初めてであるが，中川さんの感想同様理論的色彩の強さを感じた．

ツールウェアでは，米国製ツールがここでも巾をきかせているなという印象を持った．

ソフトウェア再利用，リバースエンジニアリングの課題は依然として研究途上であると感じた．

3. その他

ニースはさすがに世界的なリゾート地だけあって，すばらしい所でした．路上にころがっている○○を除けば....

岡本　修明
ファコム･ハイタック（株）

1. ESF

ESF のデモでは，ユーザ･インタフェスにおけるデータ変換によるデータ交換のコミニケーション等がプロトタイプにせよ３種類稼働することを実演して見せていました．内部処理はないにせよ１つの実力であると思いました．実システムのレベルでのシステムを期待し，今後，本運用での仕掛け，使用方法をさらに確認したいものです．

2. Praxis

議論のための議論でなく，ワーキング･グループでの Praxis の検査課長の話はわかりやすく興味を持つことができました．特に，実際の社内のドキュメントを示しての検査方式の説明には実感があり，理解できるところがありました．検査効率が３％であるとのことでしたが，この数字は本当と思われます．外国におけるソフトウェア検査の実態を見るチャンスに恵まれ，非常に参考になりました．

3. ICSE

国際会議に初めて参加し，先進の技術論文に触れ，大きな驚きを持ちました．論文には，参考になるものが多々あり，帰国後，見させていただいております．次回以降も是非参加させていただきたいと思っております．今回は，日本からの論文発表が２件と少ないのが問題だと感じました．論文発表をするよう，周囲に宣伝したいと思っています．

（４）その他

英語を余り理解できないので，自分で何らかの努力，工夫が必要な気がしました．全体としては，異なった人の文化に触れ，非常に楽しく有意義なツアーでした．

大脇　隆志
日立製作所

1. はじめに

ICSE は，私にとっては昨年（米国，ピッツバーグ）に続き２度目，ヨーロッパは初めての訪問であった．

しかし，今回の ICSE は，初の欧州体験が与えた印象ほど刺激的ではなかった．３月２３日に，ベルリンの ESF 本部を技術訪問し，翌日ツアー本体が，英国バースの Praxis 社訪問のためにロンドンへ発ったのと別れて，私は一足早く

ニースへ入り，ICSE のチュートリアルを聴講したが，期待したほどの内容ではなかった．もう少し，新しい話題の提供もしくは体系化の試みが欲しかったと思う．あえて，Praxis 社訪問を断念して聴講しただけに，半分損をしたような気分である．

2. ESF

ここでは，EC 各国の援助のもとに，Software Factory の確立を目指し，ソフトウエア生産の工業化に関する技術開発および標準化を行っている．ESPRIT 計画が，現状では，CASE 環境やインタフェースの標準化（PCTE: Portable Common Tool Environment）などの，いわゆる「ソフトウエア・エンジニアリング」分野を対象としているのに対し，ESF では，Software Factory としての「システム・エンジニアリング」までを含めて対象としているようだ．

分散環境での，「ツール統合のための Software Bus」，「管理者からプログラマへのジョブのディスパッチングや，各ジョブの進捗管理を行なうための CSCW」をプロトタイプしたデモを見ることができた．アイデアとしては，プロセスモデル記述をベースに実行しているだけで，特に目新しいものはないが，素直にそれを実現してしまうところがいい．実に野心的なプロジェクトである．

3. ICSE

今回の ICSE の基調テーマ Building a Foundation for the Future であった．基礎をきちっと固めて，着実に実践していこうとする姿勢が見られる．日本では，なぜか理論と実践が分離する傾向にある．欧米では形式的仕様記述などの Formal Approach も，研究だけでなく，実際のソフトウエア生産現場で実践しようとする試みが盛んであるように感じられた．

今回，米国は一歩引いて，EC のお手並みを拝見といった感があり「盛り上がりは，今一つ」という印象を受けた．しかし，ヨーロッパの意気込みを示したという意味では，成功といえるのではないか？

4. まとめ

ESPRIT 計画，EUREKA プロジェクトなど欧州でのソフトウエア環境の統合化・標準化の動きは，個々の技術の重要性もさることながら，ツールを超越した標準のプラットフォーム構築の意義が極めて大きいと感ずる．

残念ながら日本では，この点での機運に乏しい（まだまだ，意識が低い）．米国を中心とした OSF などの標準化コンソーシアムにも，日本（特にメーカ）はもっと積極的に貢献していくべきであると思う．

ソフトウエア・エンジニアリング理論の実践という面でも，日本は欧米に比べると数年遅れているような気がする．現状は，小集団活動的な，人的管理により，何とかごまかしている（？）．研究者の方々は，ソフトウエア工学の高級な理論に精通しているが，一方現場では，基本的な理論・方法論さえ実践されていない．このギャップをいかに縮めるかが，今後の課題であると思う．

久保　宏志
富士通（株）

1. ICSE

Building A Foundation for the Future が ICSE-12 のテーマであることに気がついたのはいつのことであったろうか？　会期の途中でプログラムを見ていてふと眼にとまったような気がする．

このテーマがあってプログラムが編成されたにちがいない．Recent Advances in Measurement（プログラムには，Measurement ではなく Metrics とあったが，Basili 先生は意識して Measurement にかえた．冒頭，そんな断りの言葉があったと記憶する）や Recent Advances in Object-Management Systems などの特別セッションは，このテーマから導かれたものにちがいない．もしそうであるならその導出に同意する．

２つともなかなかためになるセッションであった．残念なのはレジメも OHP のコピーもないことである．Basili 先生の書きものがプロシーディングズに収められているが，レジメとはいえない．

Plenary Session もまたテーマがあって，そのトピックがえらばれたのであろうが，全般的に印象が薄い．Barbara Liskov 女史の Structure of Distributed Programs は，タイトルだけは会議のテーマにふさわしいが，内容はテーマにふさわしいとはいえなかった．

最後の Plenary Session で Stephen L.Squires 氏（DARPA）は，これからのソフトウエア工学の方向性は Computational Systems Engineering にあると結んだが，この主張には同意する．そのことを説くのに High Performance Computing を例題にすることが適切とは思わないが，結論には同意する．

2. 一般的な感想

今回のツアーの収穫は，ESF と Praxis をこの眼でみたことにあります．この２つの技術訪問は大きな感銘を与えました．これまでヨーロッパには関心を払わないできましたが今回の旅行で，ヨーロッパが視野にはいってきました．しかも，大きな位置をしめるようになりました．開眼の旅

であったことになるでしょう.

岸田 孝一
(株) SRA

1. ニースまで

ESF の存在を知ったのは, 去年の夏シスコの IFIP のパネル Software Factory around the World でディレクタの R.Thomas 氏と同席したときが最初である. そのあと10月の ISPW-5 で W.Scacchi 先生から「あそこは ESPRIT よりもおもしろそうだ」という噂を聞き, ぜひ訪ねてみたいと思っていた. もちろん, ベルリンの壁もついでに見たかったことも否定できないが...

たまたまわれわれが訪問した日は, アメリカから L.Osterweil 先生をリーダとする Alcadia のグループ, それに B.Boehm さん以下の DARPA のグループもベルリンを訪れて, 夕方のレセプションは思いがけぬ交歓会になった.

プロトタイプのデモを見て, 深瀬さん(アスキー)と顔を見合わせ「われわれが SDA プロジェクトでやろうと考えていて, まあできることはわかっているのだからわざわざ試してみることもなかろうと, ついついサボってしまった間隙をみごとに抜かれた!」と思わずぼやく.

Praxis は, フォーマル・アプローチの実践や品質保証への地道な取組みで以前から知られたユニークなソフトハウスである. ローマ時代の温泉で有名な観光都市バース (つまり Bath) に本拠を構えているというのも, わざとらしくキザな会社だ.

ロンドンからバースへの途中ちょっと回り道をして, ストーンヘンジの遺跡を見る. 風が冷たい. 3月にイギリス通産省の視察団のメンバーとして来日された常務の M.Bean 氏夫妻の御招待で, 古い牧師館を改装したしゃれた Manor House で夕食を御馳走になる.

翌日の訪問の冒頭, Bean 氏の Welcome Speech の中で,「イギリスのソフトハウスはすでにメーカを下請けにしてシステム・インテグレーション・ビジネスをしている. 日本のソフト業界は遅れているのではないか?」といわれたのには参った.「まああちらはメーカの力が弱いからネ」と, 一応は負け惜しみのセリフは吐いてみたものの. 内心ではもっとがんばらねばと思う. ISPW で一緒だった C.Roberts 氏によるプロセス・エンジンのデモは, ほぼ予想通り. ESF を見たあとのせいか, それほど感動的ではなかった.

2. ニースにて

ヨーロッパでの ICSE は, 第4回(ミュンヘン)そして第8回(ロンドン)に続いて3度目だが, 今度が一番つまらなかったと思う. プログラムを眺めて, 食指をそそられるセッションが少なかった(これはプログラム委員会の責任). そして, フランスのお国柄(会議の事前登録人数で主催国フランスがアメリカより少なかった!).

ミュンヘンの時は E.Dijkstra が世界中のソフトウェア・エンジニアたちから総スカンを喰うという歴史的パネルを目撃することができた. ロンドン会議では, ちょうど始まったばかりの ISPW 参加者たちを中心とする知的パフォーマンスが私たちを楽しませてくれた. そのような意味での興奮が, 今度の会議には欠けていたように感じた.

Belady - Barstow - 鳥居の3氏が運営をリードする来年のオースティン会議は, 私自身もまた PC に復帰するので, こんな風にならないよう, 少しがんばらなくてはと思う. みなさん, どしどし論文を応募してください.

3. ドルトムント行き

パリでツアーの一行に分かれて, また西ドイツに戻る. ESF 主催の小さな国際シンポジウムで何かしゃべってほしいというリクエストである. ドルトムントは, ルール鉄工業地帯の中心都市の1つ. 鉄鋼不況で落ち目になったのが最近ハイテクで息を吹き返しつつある. 西ドイツ版ピッツバーグといったところか. ドルトムント大学付属のソフトウェア技術センタは ESF の技術的リーダシップを担っている.

シンポジウムは, 3日間. いろいろな人がそれぞれ1時間弱しゃべり, 質疑と討論を行なうというインフォーマルで楽しいイベント. 私の出番は Osterweil 先生に続いて初日朝の2番手.「環境構築における技術移転」という題で, SEA セミナで使った材料をもとに日本での開発環境の実態を紹介した. 日経コンのシグマ批判記事ももちろん(?)引用し, 最後はオルテガ社会学を土台に, 環境における自己疎外と自己実現の問題提起でしめくくったら, 意外に評判がよかった.

ニースのあとイタリアのトリノで別の会議に出られた鳥居先生も2日目から合流し, 帰りはフランクフルト経由の直行便で一緒に成田へ. 気流の関係か10時間ほどしかかからない. アメリカ西海岸より近いのには驚いた. これからは, たびたびヨーロッパへも足を向けることにしよう.

Eureka ソフトウェア工場

中川 中
SRA

Eureka ソフトウェア工場(以下, esf)は欧州における一大研究計画 Eureka の一環として 87年頃から 10年計画で始められた産学共同プロジェクトであり, ソフトウェアの工業生産を行なうための汎用の環境の構築を目標とする.

1. 組織と目的

1.1 目的

esf はより効率的なソフトウェア製造環境を提供するための協力体制を敷くことを目的とする. 個々の支援環境を構築するのは利用者自身に委ねるが, esf はその構築を容易にするため諸サブシステムが標準のインタフェイスを持つような共同アーキテクチャを与える.

このアーキテクチャは (1) 柔軟性, (2) 組織性 (3) 計測可能性および (4) 進化可能性を備えていなければならない.

具体的な成果として期待されるのは,

(1) プロジェクト管理や文書化など特定の機能をもつ機能部品,
(2) プロセス·エンジンやソフトウェア·バスなど, 工場を構成するための骨格部品,
(3) (1) の部品の利用を容易にするためのオブジェクト管理·通信などのサービス,
(4) 工場の構成·運営手法を盛り込んだプロセス·プログラム,
(5) アーキテクチャやインタフェイスの標準仕様

などである.

さらに, プロジェクトの進捗状況を把握するために中間成果物の発表日程が決められている. たとえば, アーキテクチャ定義案は 88年第 4四半期, 90 年第 2四半期, 91 年第 2四半期にリリースされる予定である. また, デモンストレーション版は 89年第 2四半期, 最初の製品は 92年第 2四半期に完成する, という計画である.

1.2 組織

esf プロジェクトを組織する目的は以下の通りである:

- 既に個別の技術は存在するので, それらを広く現実に適用できるような運動を十分な規模で起こす,
- 標準化と技術の適用とを組織上の交流も含めて相互に補完しあう.

そして組織構成は:

- プロジェクト内外の変化に柔軟に対応できる,
- 組織内の研究開発単位や関連グループの蓄積を十分に活用できる, また
- 管理や責任を明確にする反面で官僚的にはならない,

ことを目指す.

以上の点を考慮して次のような指針が立てられた. まず, 数国の大企業および重要研究機関を募る必要がある. つぎに, 十分な規模を得るためには活動拠点は分散しなければならない. 第三に, 構成員の変動でプロジェクトが動揺してはならない. 第四に, 共同活動が必要な場合に構成員の利益が確保される必要がある.

この指針の下にコンソシアムが発足し, 89 年時点で 5 ヶ国 13機関が参加している. 構成員はシステム·ハウス, 大口ユーザ, 研究機関など多岐に亘る. 活動分野は 17を数える. 本部は Berlin に設けられ, ここで事務·会計手続き全般, 各活動の協調, デモンストレーション, 各種会議が行なわれる. そのためにあてられる人的資源は, しかし, 全体の 1 割程度で, 残りはすべて参加各国に散在する諸開発拠点に投入されている.

2. 活動分野

esf プロジェクトが携わっている分野は以下のとおりである.

2.1 標準化

構成要素の可搬性や交信可能性は esf がめざすアーキテクチャにとって重要であり, esf 内の活動の他, 特に ECMA, OSF 等外部の標準化団体との交流に力を

注: 本稿は「ESF Project Information」および ESF Demonstrator」の要約である.

入れている．

2.2 ガイド

ユーザ対話構成要素 (UIC) の生成システムの構築を目的とする．

2.3 適応型対対話型 UI サービス集合

UIC を構成するためのツール集合を提供することを目的とし，既存のユーザ·インタフェイスの評価，ソフトウェア·バスとの統合，および人機械間の通信の高度な面の研究を行なう．

2.4 UI サービス

個人と機械とのインタフェイスを設計，雛型作り，生成するためのツール集合を構築する．特に雛型作りに力点が置かれる．

2.5 プロセス制御言語

オブジェクト指向に基づき，並列性を記述できるようなプロセス記述言語を開発する．

2.6 プロセス記述および管理環境

プロセス·モデルを定義·特例化するために必要な方法論，言語およびツールを提供する．

2.7 ソフトウェア·バス

ソフトウェア·バスを構成する機能の雛型を実装する．通信サービス，操作サービス，はめ込み機能および統合能力の4活動に分かれる．

2.8 オブジェクト格納サービス仕様

esf における格納サービスと他のサービスとの統一インタフェイスの仕様とその根拠を明示する．

2.9 ソフトウェア再利用

再利用の概念を分析·定義し，esf における再利用手法·ツールを実装する．また理論面の研究も行なう．

2.10 ソフトウェア構成管理

柔軟性および広範性を特徴とするような構成·版管理，リリース管理，ベースライン管理，変更制御·履歴管理を行なうシステムを実装する．

2.11 技術著作家用統合支援ツール

GRIF をベースにして文書の編集，生成，書庫管理，表現および配布の機能を含む統合文書開発環境を作成する．

2.12 esf の ada 支援環境

esf の基準に合致するような ada の支援環境を構築する．雛型として既存のツールの統合を行ない，その後実用向けのシステムを作成する．

2.13 ソフトウェア品質援護作業台

品質支援活動計画，要求の設計やテストにおける反映，提出物の評価，計測値の獲得·分析やテスト計画とその適用，を支援するシステムを作成する．

2.14 中間表現

プログラミング言語と機械語との中間言語の処理系を作成する．

2.15 実時間組込型システム分野用の最初の esf

実時間システムのための esf 例を構築する．

2.16 妥当なプロジェクトを統合する高度の試み

PIMS, ASPIS, TWIST, SPECIMEN といったツールの統合実験を行ない，プロジェクト管理支援環境を提供する．

2.17 事務アプリケーション用生産環境

事務アプリケーションの開発に適する統合ソフトウェア生産環境を生成する．個々の構成要素の統合方法に力点が置かれる．

3. デモンストレーション

現時点で公開されているデモンストレーション用システムは以下の4種類である．

3.1 HAPPI (An Integrated Factory Environment)

種々雑多なツールや機種を統合して幾つかの役割モデルが協力しあうようなプロジェクト用の環境を構成する試みの実現性を確認することを目的とする．デモンストレーションでは，TWIST という進行中のプロジェクトで月次報告書を作成する，というシナリオが想定される．

役割モデルはプロジェクト管理者，設計者，技術著作家，会計士，および支店管理者の5つで，それぞれが単機能ツールを用いて作業を行なう．たとえば設計者は一般のエディタ，会計士は市販のスプレッド·シートを使う．これらのツールの起動·データ入出力がどの程度自動的に行なわれるかが統合レベルの指針となる．たとえば，設計者の使うエディタは自動的に起動されるが，データはプロジェクト情報と関連づけられない．一方，スプレッド·シートの入力データは自動的にロードされ，こちらの方が統合化の度合

が高い.

3.2 PEBA (Two Tools for Software Design)

異なるデータ構造を利用するツールをソフトウェア·バスを介して結び付けることの実現性や利点を確認することを目的とする. ここでは典型的な例として, テキスト·ベースのモデル実行系と汎用グラフィックス·エディタを結ぶ.

概念データ·モデルと組織プロセス·モデルの仕様がグラフィックス·エディタにより図の形で定義される. この定義は適当なデータ変換を行なったうえでモデル実行系に送られる. アニメータは文法チェックを行ない, 実行用の雛型を作成する. この雛型を実行するとグラフィックス·エディタの図上でもシミュレートが行なわれる.

以上の結合により, 一つのツールに他のツールの機能の追加, ユーザ·インタフェイスの共通化·高度化, 冗長データの除去等が可能になる.

3.3 SITE (SI Team Environment)

管理者とプログラマとの協調を支援する環境であり, 以下のような仕組みから構成される.

(1) 管理者用のプロジェクト·プロセス支援ツール: 階層構造の作業ネットワークを定義し, プログラマへ作業割り当てを伝える. プログラマからは作業報告を受け取る.

(2) プログラマ用の作業管理ツール: (1) の作業割り当てを受け, また作業状況を報告する.

(3) 作業割り当てには作業の遂行に必要なツール: データを解放するための作業文脈が付随する.

全体のアーキテクチャは esf のそれにしたがっており, ソフトウェア·バスにユーザ·インタフェイス要素とサービス要素とがつなげられている.

3.4 TPM (A Tool for Process Modelling)

プロセス·モデルを記述するための言語とその処理系の構築を目的とする. 処理系にはグラフィックス入力ツールやモデル·アニメータを含む. 力点はモデリングにあって, プロセスを実行してしまえるような詳細な記述は避ける.

言語の構成要素には:

(1) 活動の集合である段階

(2) 活動

(3) 活動を実行する役割遂行者

があり, 実行単位である活動は入出力文書の集合に依存する形で定義される.

アニメータはプロセスを活動単位で追跡し, デッドロックや未定義文書の検出を容易にする. ある時点である活動が実行可能かどうか, 実行した結果はどうかなどは主に文書の存在の有無として表現される.

参考文献

今回の訪問で入手した esf 関係のまとまったドキュメントとしては, 次の4種類がある:

(1) ESF Technical Reference Guide (Version 1.1), 1989, 45pp.

(2) ESF MINI - Functionalities and Characteristics, 1988, 88pp.

(3) A Methodology for describing ESF Components, 1988, 39pp.

(4) A methodology for the Design and Description of Complex, Object-Oriented systems (Version 0.1), 1988, 44pp.

(1) ～ (3) は ESF 発行の技術文書. (4) はノルウェイの工業技術センタ (SI) でまとめたオブジェクト指向設計法の解説文書で, ドイツの社会学者 Max Weber の説いた「理想の官僚制度」がオブジェクト指向パラダイムの起源だというユニークな巻頭言が載っている.

なお, これらのドキュメント (他に OHP のコピーなど) は1セット SEA 事務局に保管してある. 興味をお持ちの方は連絡されたい. ただし, ページ数が多いので, 無料のコピー配布サービスにはちょっと応じかねる.

全ライン・コード・レビュー実施の定着化

勝田　祐輔
日本ユニシス

1. 序言

本稿は，ソフトウェアの開発にさいして[1]，作成したソース・コードの全ラインを対象とするレビューの実施が定着するに至った経過の報告と考察である．

開発した全ラインのコード・レビューなど，とても時間がかかってできるものでないだろうと，だれもが考える．われわれのところでもそうであった．ところが，実施のタイミングややり方によっては，所要時間がそれ程でもないということが発見され，そうした認識が次第に全員に広まっていった．

われわれの部門におけるレビュー実施率は，当初，全コードに対して，たかだか数十パーセントであった．品質もよくなかった．ところが，いくつかのプロダクト開発[2]にさいして，初めての試みとして全ラインのコード・レビューを実施したところ，結果として予期外のよい効果が確認された．これにより，コード・レビューの価値が全員に認識され，その実施が広まっていったのである．

以下では，その経緯，および，全ラインのコード・レビューを効果的に実施して行く上で必要と思われる事項を要約する．

2. レビューの位置づけ

開発成果物の品質向上，要員の技術レベルの向上，仕事の進め方の改善，問題点や弱点の早期検出，必要な各種リソースの確認などのために，定期的（たとえば毎週または隔週）なレビューや，開発フェイズの区切りごとのレビューがある．後者のレビューの主なものは，たとえば次の通り：

- 要求仕様レビュー
- 開発実行計画レビュー
- デザイン・レビュー
- コード・レビュー
- テスト計画レビュー
- テスト結果レビュー
- 本番稼働前（or リリース前）総合レビュー

[1]われわれの部門は，WS上のソフトウェアの開発を行っている．開発しているのは標準ソフトウェア・プロダクトならびにユーザから受託した個別システムである．開発言語はCが主である．

[2]スクリーン・デザイナ：U-SD/MAPPER，U-SD/COBOLの2プロダクト

むかし，ある本で「デバッグ報告書がきちんと書けているような開発プロジェクトは，おおむねうまくいっている」という主張を読んだことがあり，非常に印象的であった．レビューには，この指摘と共通する何かがあると思っている．レビューという行為は，現在および将来のプロダクトやプロセスの改善をはかる上で基本的な行為であると，筆者は考える．

上にリストアップしたレビューのうち，いままで部分的にしか実施できなかったのが，コード・レビューである．ソフトウェア開発の改善のために，やった方がいいという手法はこれまで数多く提唱されてきた．そして，それらの多くは実際に効果があることが証明され，信じられている．にもかかわらず，現実には，全然実施されていないか，または不十分にしか実施されていない．人々は日常の忙しさに追われて，必要なトレードオフ分析すら行っていない．地道にやることこそが重要だと思う．

3. コード・レビューについての概観

ソフトウェアの開発・保守にかかる工数の総和（すなわちライフサイクル・コスト）を少なくするには，プログラミング段階でコード・レビューを実施し，初めから質のよいコードにしておくことが必要だといわれてきた．

実際，質の悪いコードを作ってしまったために，稼働してから安定性やパフォーマンス上の問題が続出し，延々と保守作業が続き，しかもコードの読みにくさのために作業の効率が悪くときにはシステム・ダウンの危険にさらされるといった類の現象は，現にあちこちに存在している．

E.Yourdonは，その著書（文献1）の中で，「ウォークスルーの基本的な目的はエラーの検出だ」と述べている．開発期間に比較して保守期間が長く，その負荷がさらに新規開発のバックログを圧迫して，事態が深刻さを増しつつある現在では，このYourdonの指摘についても，より深く吟味を加えることが必要であると考える．

4. 全ラインのコード・レビュー実施への経緯

4.1 以前の実施状況

われわれの部門におけるこれまでのコード・レビューの実態を示す典型的な例として，次に2つのプロジェクトの場合を示す．なお，ここでコード・レビュー率は，レビューしたライン数を新規作成コードのライン数で割ったもので

ある．

例 1. プロダクト F の開発（C 言語）

コード･レビュー率＝ 20 %

内訳	新規作成コード	24,050ライン
	流用コード	6,150ライン
	レビュー所要時間	15時間

このプロジェクトでは，テスト工程が当初予定の約 4倍になり，納期･品質･保守工数ともに深刻な状況であった．

例2. プロダクト M の開発（C 言語）

コード･レビュー率＝ 44.6%

内訳	新規作成コード	8,205ライン
	流用コード	5,990ライン
	レビュー所要時間	10時間

このプロジェクトは比較的うまくいったほうであった．しかし，テスト段階において，明らかに前工程に多くの問題があることが認識された．リリース後も，安定性･パフォーマンスの両面で，問題が次々と報告された．パフォーマンス改善のためにコードを直してすっきりさせたら，バグも一緒にクリアされたといったこともあった．全ラインのコード･レビューにどの程度時間がかかるかを試算した結果，23 時間と出た．この数値は，予想よりはるかに少ない値であった．

4.2 実施できない原因と考察

前節，例 2 の開発終了後，ただちに検討プロジェクトを発足させた[3]．最初に上がってきた結果は，おおよそ次のような皮相なものであった．

- コード･レビューが実施されない主な原因は「時間がない」ことである．
- 定着させるための主な対策は「計画段階でコード･レビューの時間を見積っておく」ことである．

しかし，「時間がない」というが，はたして本当にそうだろうか？ 仕事が忙しすぎて食事を取る時間もなく，餓死（！）したプロジェクトの例など，いままでに聞いたこともない．

また，あらかじめ「コード･レビューのための時間を見積っておく」ことは，当然必要であろう．しかし，もし開発ライン数が計画時の 2倍になってしまったとしたら（それは十分ありうることだ），予定通りにレビューは行われるだろうか？

要するに，プライオリティの認識の問題である．関連して考えなければならないことも，他にいろいろあるだろう．もっと本質に迫るような検討が必要だ．何のためにコード･レビューをやるのかを，もう一度徹底的に議論してみる必要がある．

そこで，筆者も一緒に入って検討をやり直した．その結果を次節にまとめる．

4.2.1 コード･レビュー実施の目的

コード･レビューの役割，テストとの関連，開発後の保守の負荷などについて議論を重ねた．その結果，コード･レビューの目的は単なるエラー検出以上のものがあるという結論に達した．議論の経過は以下の通りである：

(0) コード･レビューでなるべく多くのエラーを検出して，テストを効率化すること： エラーは早期に発見すればする程，その修正コストが少なくてすむので，この考え方は確かに重要である．

(0') さらに，テストが非常に難しい部分のコードを，しっかりレビューして確認すること： この考え方も重要である．しかしながら，人間の目によるレビューだけでコードの正しさを完全に確認するのは，容易ではない．コードは，最終的に計算機上で動くものであるから，動作確認を省くわけには行かない．

したがって，人間および計算機のそれぞれに固有な役割というものが，上記の２つの目的（0 および 0'）以外にあるのではないか？だとすれば，人間でなければできないような重要な役割を，コード･レビューのもう１つの目的として追加すべきではないだろうか？

以下にその検討結果を述べる．

(1) コードが書式標準に合致しているかどうかを確認すること： これは機械では難しい．書式標準が主として要請しているのは，変数名や関数名が十分に吟味されていて，意味がとりやすいこと（名前の吟味が不十分なまま安易にコメントでカバーするのは邪道である），移植性のない表現が避けられていることなどである．他にも，モジュール化，関数のサイズ，構造化などに関することがらもある．

保守を容易にするためには分かり易いすっきりしたコードが必要なことは明らかである．開発者以外の人間が保守を行なう（行なわざるを得ない）ことは日常茶飯事である．保守段階における負荷と安全性をいかに改善するかは，現在，ユーザにとっても，メーカにとっても，きわめて重要な課題である．また，コードを分かりやすくすることによっ

[3]品質改善活動の一貫として実施した．

て，バグを駆逐できるという復次的効果も期待できる．

(2) アルゴリズムの大筋および計算効率（時間·空間計算量）の妥当性を確認すること： アルゴリズムやデータ構造が不適切なためにパフォーマンスが悪くなる実例は，枚挙に暇がない機械で稼働させてからわかるようなのは，相当にひどいケースである．部分的な効率の悪さの積み重ねによって全体の効率が悪くなっているような場合には，対処が大変になる．

また，このような観点でのレビューは，エンジニアの自己研修に対する意識（計算量の理論，アルゴリズム）を促す教育的効果もある．

(3) インタフェースの正しさを確認すること： 他のモジュールやオペレータとのインタフェースの不整合は，後の工程で発見·指摘される場合が多く，しかも，この種の不具合は何ヵ所にも組み込まれてしまう性質のものである．したがって，それらをできるだけ初期の段階で検出することは，全体の生産性向上の見地からも望ましい．

(4) プログラム流用が適切に行われているかを確認すること．

(5) 当該コードが，保守の観点から見て，最善のコードであるか否かを確認すること： (2) では，アルゴリズムの大筋の適切さを要求しているが，保守の観点からの最善性までは要求していない．ここでは，そうした保守性の考慮を要求している．

リリース後のバグ修正においては，コード作成から長い時間が経ち，また担当者が別人に代わっているという場合がよくある．全体を十分に読み切らずに，急いでパッチしてしまうようなまずいケースもあり，危険この上ない．数ラインの誤ったコードのために，重大なトラブルが発生した例も世の中にはたくさんある．コードの内容および構成を十分にチェックし，保守が容易なコードを目指すことは，非常に重要である．

以上の諸目的を全員で確認し[4]，以後のコード·レビューを実施した．

4.2.2 コード·レビューとコード流用との関連

流用したコード自体については，レビューは行わなくてもよい．インタフェースの確認だけが必要である．したがって，レビューの負荷は，流用した分だけ減ることになる．コード流用は，生産性向上の決め手の１つであり，一般の関心も高い．しかし，下手な（汚い）プログラムは，いかに流用を勧めても流用されない．流用されるには，そのコードが美しいコードであることが必須である．

コード·レビューの場は，既存関数の流用を具体的に指摘する場である．と同時に，有用な関数を共用関数として，レビューの参加者が見出す場でもあり，「この関数は流用を意識して作るように」と促す場でもある．コード·レビューとコード流用の推進は，一緒に進めるのがよいと思われる．

4.2.3 見積り精度向上の必要性

開発計画時点で，プログラミング工数の見積りがしっかりできてないと，コード·レビューにしわよせがくる傾向が見出される．

議論の結果，見積りに必要な各種データを整理蓄積していくことが大切で，そのためには，開発途中で各種データをとって行き，開発完了時点でそれらを取りまとめて考察を加え，将来に活かそうという主旨で，「開発完了報告書作成要項」を作成することに行き着いた．

5. スクリーン·デザイナ開発における実績の紹介

この開発は，初めて全ラインのコード·レビューを実施し，その価値が認識できたということで，我々にとっては記念すべき出来事であった．

5.1 開発ライン数実績

言語	総行数	新規作成行数	流用行数	流用比率
C	12,905	10,619	2,286	21.5%
ASM	1,105	1,105	0	0%

表5-1 スクリーン·デザイナ開発ライン数実績

5.2 全ライン コード·レビュー所要工数

プログラミング全工数	レビュー工数	工数比率
1,646人·時	194人·時	11.7%

表5-2 スクリーン·デザイナ開発，全ライン コード·レビュー所要工数

5.3 コード·レビューで指摘された問題点

次ページの表で，性能の考慮，コーディング規約，コーディング·ミスが，それぞれ 20.7%で，合わせて 62.1%となっていることが注目に値する．このうちコーディング·ミスの 20.7%は，コード·レビューなしでもテストで検出できる可能性があるが，残りの 41.4%は，コード·レビューが実施されないと，ほとんどがそのまま残る可能性を持っ

[4]この検討は，品質改善活動の一貫として行われ，成果物として「コード·レビュー実施要項」が作成された．

ている．

内容 重要度	メジャ	マイナ	件	%
制御構造	2	4	6	4.6
データ構造	1	3	4	3.0
エラー処理	2	3	5	3.8
ARMの考慮	2	4	6	4.6
性能の考慮	27	27	20.7	
コーディング規約	1	26	27	20.7
注釈	5	5	3.8	
コーディングミス	6	21	27	20.7
仕様追加	6	3	9	6.9
コード欠落	2	6	8	6.1
操作性	1	1	0.7	
外部関数の使用法	5	5	3.8	
合計	22	108	130	
		(16.9%)	(83.1%)	

表5-3 コード・レビューで指摘された問題点サマリ

5.4 テスト・ケースと検出エラーの実績

マシン上での機能テストの結果は，次表の通りである：

区分	名前	行数	項目数	ケース数	エラー数	エラー率
M	SDM	1,175	1	21	4	0.34%
M+C	SD	4,911	33	683	44	0.9
M	SDMG	849	3	155	7	0.82%
C	SDC	1,200	1	22	0	0%
C	SDCG	2,022	10	163	6	0.3
C	SDCP	747	5	67	0	0%
C	Subroutine	3,106	17	409	3	0.1
C	合計	14,010	70	1,520	64	0.46%

表5-4 スクリーン・デザイナ機能テスト要約表

表の区分 M は MAPPER，C は COBOL のモジュール，M+C は両方のプロダクトで共用されていることを示す．エラー率は，エラー件数／ライン数である．プログラマは，4名であった．この表で，モジュール SD と，SDMG のエラー率が高く，一方，SDC, SDCP およびサブルーチンは非常に低いことが目立つ．いずれも，協力会社にプログラミングを依頼したものだが，これは担当プログラマの能力・経験による差であると推定される．コード・レビューにおいては，この点も考慮に入れる必要がある．

5.5 全体の工数サマリ

工程	工数(人時)	(人日)	協力会社工数	比率%
デザイン	925.5	6.6	0	0
プログラミング	1,646	11.76	8.56	72.8
		(225)	(1.6)	
テスト	885	6.3	4.53	71.9
		(150)	(1.07)	
マニュアル作成	279	1.99	0	0
合計	3,735.5	26.65	13.09	49.1

表5-5 全体工数サマリー

プログラミング工数と，系統的な機能テスト工数の比率は，上の数字で妥当であると考えている．以前の失敗プロジェクトでは，この値が逆転していた．

5.6 リリース後の状況

1988年 6月にリリースし，社内外で数多く使われている．現在（ 1990年4月）まで，バグの報告は皆無である．

6. コード・レビュー実施のための前提条件，実施により導かれた教訓及び提言

スクリーン・デザイナ開発の経験以後，当部においては，ようやく全ラインのコード・レビューを行なうことが，自然な考え方として実施されるようになった．現在までの我々の考察と経験をもとに，考慮すべき事柄や効果的な方法について，若干の補足説明を加えて以下にまとめる．

6.1 関係者の意識

コード・レビューの目的や重要性について，担当者全員（協力会社員を含む）に十分理解させることが，最も重要である．「コード・レビュー実施要項」を，担当者全員に十分に時間をかけて説明し，実施方法を検討した．時間はかかったが有意義であった．

6.2 事前準備

コード・レビュー以前に，コーディングの標準化についてのドキュメントが，まず必要である．われわれの場合，1つの標準化案を以前に全員の検討で作成し[5]，1年後に内容を更新してあった．全員が毎日守るべきことがらは，全員の手で検討し，本当にその日から継続して実施しできるかどうか，現実的な考察を加えることが大切である．なお，この種の標準は，その時点の技術レベルにおけるベストのもの

[5]職場の中の品質改善活動として数ヵ月の期間にまとめた．

をねらい，世の中の技術進歩に応じてどんどん更新して行く体制をととのえておく必要がある．

6.3 フォーマット類の設定と運用

われわれの場合，「コード・レビュー参加依頼表」の類を設定して[6]，毎週のプロジェクト・レビューで出席者を決める方法をとっている．「コード・レビュー結果報告書」の設定，日時，所要時間，出席メンバー，ライン表，指摘された内容，その問題がメジャーかマイナーかの区別，合計指摘件数，再レビュー必要性の有無などが，その内容である．この報告書を管理者が確認し，必要に応じて（特にメジャーな問題については），実際のソース・コードのジャーナルをもとに，担当者と技術的な討論を行なう．指摘件数が回数を追うごとに減ってきた場合には，それをはっきり評価してやることも大切である．

6.4 タイミングと1回のレビュー・ライン数

ソース・コード 500ライン以下のある単位で，コンパイル・エラーがなくなった時点でレビューを実施することにしている[7]．所要時間は 1時間から 2時間程度とし，それ以上長引く場合は，別の日時を設定する．これは，集中力が持続する限度内という意味である．

6.5 管理者の意思と姿勢

レビューがすんでないコードは未完成品とみなし，絶対に外へは出さないという管理者の強い意思も大切である．SPR（Software Problem Report）の処置欄に，必ずコード・レビュー実施ずみのチェック（およびレビュー結果報告書 ID）を記入することが有効である．これは，「SPR 運用要項」にも関係する事柄である．

開発全体の問題，すなわち工数見積りの精度を改善して行く努力を続けることも大切である．「開発完了報告書作成要項」の類を利用して，開発に関する基礎データを蓄積して行くといったことが必要である．

スケジュールを守ることに対する姿勢の甘さ（シビアさの欠如）があると，コード・レビューにしわ寄せが来る恐れがある．この点についても，管理者の強力な指導が必要である．

6.6 流用の推進

流用コードを多くするよう推奨することも有効である．流用コードは，レビューの対象外とする．なお，共用関数の運用についても，全員で討議し運用要項を定めて守って行くことが大切である．

6.7 レビュー参加者について キーマンの参加は絶対に必要で，これなしでは意味が少ない．一方，コード・レビューの場は，OJT の場をしてもきわめて有用である．初心者を，教育的見地からあえて参加させることもときには必要であろう．よいコード，まずいコードについての生きた教育の場として最適である．

6.8 SPR における不具合の分析

これはまだ十分に実施できてないが，コード・レビューで鋭い指摘ができるようになる能力を養う上でも，有効であろうと考えている．まだ断片的な実施にとどまっているが，今後の大きな課題である．

7. 結び

コード・レビューの重要性をいうはやすいが，それを定着させるのは容易でない．日常の管理，コントロールも重要である．われわれの場合，定着に至るまで 3年ほどの時間がかかった．急がず，着実に関係者の心理的な外堀を埋めて行くこと，そして担当メンバーとの徹底的な討議を通じて，レビューの必要性に関する認識を深めさせることが大切だと思う．

本稿において，筆者が特に訴えたかったのは，コード・レビュー実施の目的についての考察，および前章で述べた実施のための前提条件整備，導かれた教訓といったことがらである．実際のコード・レビューに要した時間は，すでに数値で示したように意外と少ない．

人間でなければむずかしい仕事と，機械を使わなければできない仕事（たとえば周到な系統的テスト）の区別をしっかりわきまえて，それぞれの作業をきっちりやることが大切である．前者を怠れば後者が非常に非能率になることはすでに述べた通りである．しかも，保守段階にも大きな影響を与える．コード・レビュー技術に関心を抱いておられる方々から，今後いろいろな助言をいただき，改善して行きたいと念願している．

[6]フォーマット類は，すべて担当者が案出した

[7]コード・レビューがテスト段階にずれこみ，混乱した苦い経験にもとづく．

参考文献

1) E. Yourdon 著　国友・千田訳（1981年）
ソフトウェアの構造化ウォークスルー

2) Victer R.Basili's Lecture Note（1989年 7月）
"Improbing The Software Process and Product In a Measurable Way"

SIGENV Forum in 神戸 全体報告

渡邉　雄一
アスキー

1. はじめに

SIGENV（環境分科会）では，地方会員とのコミュニケーション（呑みニケーション）を基本としたフォーラムをすでに３回〔盛岡で２回，神戸で１回〕開催してきました．

幸い，いずれのイベントも，地元会員の方々のご協力を得て，無事に成功をおさめることができました．それらのフォーラムでの発表や討論の記録は，特集号の形でSEAMAILに載せるつもりで，ずっと原稿をため込んでいたのですが，整理がほとんどできないまま，今日に至りました．その結果，いまとなっては内容の鮮度の点から見て，そのすべてを掲載するのもふさわしくないような状況になってしまいました．

せっかく寄稿してくださった方々，また会場でアンケートに協力したくださった参加者のみなさんには，まことに申しわけないのですが，ここでお詫び申しあげます．

それらの原稿の中から，昨年の１０月に神戸で開催した「ソフトウェア・ドキュメンテーション」に関するフォーラムのもようを簡単に紹介したいと思います．

2. ドキュメントの計算機支援を中心に

これまでにも何回か御紹介したように，SIGENVでは一昨年から昨年いっぱい（ほぼ１年半近く）の時間をかけて，プロトタイピングについての討論を進めてきました．

究極のプロトタイピング支援環境には，当然ドキュメントの作成も含まれるはずです．では，「実際のところ，ドキュメンテーションの機械化支援はどうなっているのだろうか？　特に，現場でアプリケーション開発に従事している技術者たちにとって，どのような恩恵があり，どの程度活用されているのだろうか？」といった疑問が浮かび上がってきたわけです．

もちろん，一般にCASEツールといわれる上流工程の開発支援を中心に据えたツールや環境では，当然ドキュメンテーションの支援が重視されているでしょうが，CASEが日本で話題を集めるようになったのは，ようやく今年になってからであり，しかも，まだまだ現場に普及しているわけではありません．

当時，前の会社でメインフレームをハッキングしていた私には，そこでの開発支援ツールの状況が，世間でいわれるほどヒドクはないことがわかっていたので，それらを叩き台に，メインフレーム以外のアプリケーション環境や，資産としてのドキュメントの管理といった話題もまじえて，幅広いバックグラウンドを持つSEAの会員たちが相互に交流できればと考えて，このようなフォーラムを企画したわけです．

幸い，実際にツールを開発している元治さん（KCS），吉村さん（富士通），平田さん（SRA）の３人をお招きして，お話をうかがうことができました．初めのお２人の発表要旨は，25～26ページを参照してください．

平田さんの発表内容は，アブストラクトがないので，ここで要約しておくと，WS（Sony-NEWS）の上でソフトウェアの部品化と簡単な合成を目指した試みであり，開発対象はC言語ベースの制御系アプリケーションです．いくつかのフェーズを経て部品の生成を行なうようになっており，各フェーズで得られる情報と雛型部品を用いて，ユーザから提出を求められる形式的なドキュメント類の多くを機械的に生成することができるシステムです．WS上のツールですから，個別のユーザへの対応も比較的容易で，ドキュメント形式のちがいなども簡単に扱えるそうです．

また，元治さんのお話は，ソフトウェア・シンポジウム'88で発表された論文の「その後」がどうなったのかを知りたくて，私が強引に話題提供をお願いしたものです．とかく，日本では，某国家プロジェクトのように失敗を失敗と認めなかったり，また，きちんと問題点の分析をすることなく，貴重な事例を闇に葬ってしまう風潮が強いのですが，そうした意味でたいへん参考になる有意義なお話を聞かせていただけたことをうれしく思います．

3. 会場でのアンケートとその後のお便りから

会場でアンケートをお願いしたところ３４名の方から回答をいただきました（ちなみに参加者は，一般３６名，関係者７名）．とても全部を紹介することはできませんので，代表的な意見を以下に要約しておきます．

3.1 ドキュメント管理ツールの試作

元治景朗〔KCS〕

－　ドキュメントのライフサイクルとして廃棄フェーズを作るのはおもしろい．

- 実際にドキュメントを廃棄したケースについて，その判断例を教えてもらいたい．
- システム開発時点で，何をユーザが求めているのかをもっと明確にしたほうがよい．
- 利用者にはどう使わせるのですか？
- できあがったドキュメントは一種の排泄物ではなかろうか？ 設計者のアイディアはもっと別のところにあるのでは？
- DB はすぐに巨大化してしまう．利用率を上げるには，いかに DB をコンパクトにするかだと思う．

3.2 メインフレームでのドキュメント支援

吉村博史〔富士通〕

- ソースのみを信用するという考え方には感心した．
- 既存のシステムからの情報（ドキュメントではない！）を用いて分析するという考え方は，保守の際に有効だ．
- ドキュメントの標準化ができてよい．
- COBOL 以外の言語もサポートしてほしい．
- 最終目標はペーパレスなのでしょうか？
- 工程についてふれられているが，ウォータフォール・モデルをベースにしないと話が展開できないのでしょうか？ そうした開発方法論（例えば富士通の SDEM）と実際の開発プロセスとがしっくりこないと感じている人は多いはずなのに....
- 現実の直面した問題にチャレンジしている点に感心した．

3.3 WS 利用によるドキュメント作成

平田淳史〔SRA〕

- ドキュメンテーションとプログラミングやテスティングといった一連の作業を統合化している点に感心した．
- いかにも Unix らしいツールの組合せで仕事をこなしている所がすばらしい．
- 現場での開発プロセスの実態になるべくそった形で支援しようとしている．
- 「ドキュメントは別に紙でなくても構わない」というのはショックでした．
- 入力データの量が多すぎはしませんか？ またデータの整合性についてもちょっと気になります．
- デモを見せてもらいたかった．

3.4 その他全般的な感想

- ドキュメントの目的は何なのか？ 作らせる側のものか？ 作る側のものか？ 使う側のものか？
- 利用者の立場の違いによってドLュメントに求められているものが異なる．
- 悲しいかなドキュメントは最終オブジェクトではなく開発·管理·コミュニケーションの道具として用いいられているに過ぎず，その結果，開発プロジェクトの運営方針１つで，その性格が一変してしまう．
- Unix ではマニュアル以外のドキュメントはない．それで足りているのはなぜ？
- ドキュメントには，再利用のための情報伝達物としての性格がある．
- 作業進捗の目安（量）←外注管理←ウォータフォール·モデルでの採用．
- 開発方法論，プロセスモデル，開発技法，ツールの関連やいかに？

4. おわりにかえて

ドキュメントというものの性格上，それらをユーザの立場で評価する方も多くおられたようで，そのような方にとって，今回の企画がちょっとピント外れであった感は否めません．特にそのような立場の人が指摘されるドキュメントの「読みやすさ」は，大変重要な問題だと思います．

しかし，そうした観点からの問題意識は，むしろテクニカルライタの仕事の範疇に属すると思います．今回の企画は，ソフトウェアを開発するエンジニアの立場から，どのような機械化支援が現実に可能であり，また有効であるかを考えようとしたのです．

ところで前月号でも紹介しましたが，今年の SIGENV は，これまでのような派手なイベントは避けて，地道に OOD の勉強会を進めています．その様子は，いずれ適当な機会に SEAMAIL 誌上で（もっと気軽な感じで）報告したいと考えています．たまには空気のおいしいところで羽を伸ばしてみたいので，勉強会の合宿を９月末に企画しており，近々その案内もさせていただきます．興味のある方はぜひ世話人までご連絡下さい．

ソフトウェアドキュメント管理ツール

＜その後のＤＤＢ＞

（株）ケーシーエス
ソフトウェア技術部
元治景朝

1．ドキュメンテーション環境の現状

当部門は、メインフレームを実行環境とする大規模な事務処理ソフトウェアの開発に携わっており、ソフトウェア開発効率の向上は我々の永遠の課題である．この開発現場においてドキュメンテーションの占める割合は高く、その対応策のひとつとして富士ゼロックス社製のＪ　Ｓｔａｒワークステーションを中核とする８０００ＩＮＳ環境を導入している．さらに、分散化された開発組織に対応した階層分類型のファイル構造を用いてドキュメントを蓄積保管し、それらを部品化・ひな型化することによりドキュメントの再利用を推進してきた．

2．ＤＤＢへの期待

ところが、日々のソフトウェア開発に並行してドキュメントは増え続ける．一方、我々の環境においてファイルサーバの物理的な容量には限界がある．当然のことながら蓄積媒体の増設が要求されるものの、それは根本的な問題解決にはならない．そこで、我々が考えたのはソフトウェアドキュメントのライフサイクルの中に廃棄というフェーズを設けることである．ソフトウェア開発の実績として残されるドキュメントの再利用性を定量化し、利用価値の低いドキュメントを捨てることを考えた．しかし、その定量化は容易ではない．そこで、蓄積保管されているドキュメントの利用状況をモニタリングし、多くの技術者にその価値判断を委ねることとした．

また、ドキュメントの再利用の現状は、分散化された開発組織の中で個別に行われてきた．しかし、ドキュメントはソフトウェア開発の事例集であり、多くの技術者の経験的知識を包含している．技術蓄積・技術移転の観点からも、ドキュメントを情報として、よりグローバルな世界で共有財産化することが有効であると考えた．

そして、これをドキュメントデータベースとして、ギガバイト単位のファイルサーバを確保できるメインフレームの環境に構築し、８０００ＩＮＳとＤＤＢでドキュメントの管理を行っている．

3．ＤＤＢの概要

データベースの実体はＲＪＥとＴＳＳにより８０００ＩＮＳと接続されたＦＡＣＯＭの環境にある．ドキュメンテーションのためのワークベンチであるＪ　Ｓｔａｒからの操作でそれにアクセスできる．そのデータベースの構造は、索引語ごとに該当するドキュメントのアドレスを管理するアスペクト方式を採用している．

ドキュメントの登録は、ＤＤＢがＪ　Ｓｔａｒ上に用意しているオペレーションシートをＭＭＩとして行われる．このオペレーションシートの情報をもとにシステムが索引を作成する．この時、利用者間の索引語に対する概念モデルのギャップを最小にするために、分類しをシステムが提示し、それを選択させている．さらに、抄録文からの索引作成は事後結合索引の方式を採用している．ドキュメントの検索は、メニューを用いた会話形式で行われ、キーワードによる多重検索でドキュメントの絞り込みが行える．

利用状況に関しては、データベースの運用管理のための情報と廃棄基準の設定と勧告のための情報を取得し、モニタリングしている．

4．その後のＤＤＢ

ＤＤＢはドキュメント単位のデータベース化を推進し、我々はそれに情報データベースとしての役割を期待している．しかし、残念ながらドキュメントの情報構造を踏まえていない現在のデータベースでは、どうやらそれは期待できそうにない．今後、今蓄積されているドキュメントを用いてその情報構造を分析する作業が、早急な課題であると考えている．

メインフレームでのドキュメント作成支援

富士通株式会社　吉村博史

1．はじめに

システム開発で作成するドキュメントについて論じる場合，ターゲットマシン・システム開発対象・作成工程・利用工程など様々な分野からの考察が必要である．

今回の報告では，我々の部門で開発・製品化している「ＳＩＭＰＬＥ／ＤＦシリーズ：ソフトウェア開発保守支援システム／ドキュメント機能」での経験をベースに，以下の分野のドキュメント出力プロダクトを説明した．

－　ターゲットマシン：大型のメインフレーム
－　システム開発対象：事務処理分野
－　作成／利用工程　：プログラミング工程以降
－　開発形態　　：ディクショナリの適用なし

2．ＳＩＭＰＬＥ／ＤＦシリーズとは

当プロダクトは，開発中又は開発済のＣＯＢＯＬプログラムソース・登録集原文（以下コピー句という），ジョブ制御言語及びロードモジュールライブラリを入力として，ドキュメントを出力するソフトウェアである．

出力するドキュメントは以下のとおりである．

入力	ツール名	出力ドキュメント
-COBOL コピー句	DF- LAYOUT	- データ領域定義 - レコードレイアウト - テストデータ記入表　他 1種
-COBOL ソース -COBOL コピー句	DF- CROSSCOB	- クロスリファレンス 8種 (データ項目⇨プログラム等) - マトリックス 4種 (コピー句⇨プログラム等)
同上	DF- PROGRAM	- プログラム一覧 - プログラム構造図 - プログラム管理表　他 1種
- ジョブ制御言語	DF- JOBFLOW	- ジョブフロー - データセット一覧
- ロードモジュール	DF- LOADCHART	- ロードモジュール一覧 - ロードモジュール構造図 - ソース日付チェック

3．基本的な考え方と特長

当社のドキュメント出力製品の適用経験や我々の部門のユーザプログラムの受託開発経験などをベースに，以下の考え方で当ソフトウェアを開発した．それは，そのまま当ソフトウェアの特長となっている．

(1)　日本語による出力

過去のソフトウェア資産を中心に，プログラム内のデータ項目名などは英数字で記述されている．しかし，ドキュメントのわかりやすさから，データ項目名などを日本語で出力することが望ましい．したがって，ユーザ毎の命名規約に対応できる日本語変換用の辞書を用いて，日本語によるドキュメント出力を可能とした．

(2)　用途に応じたドキュメント出力

誰が・いつ・何のために・何のドキュメントを・どこで・どのように使うかを検討し，用途に応じたドキュメントを出力できるように，日付指定など豊富な範囲指定とドキュメント種類毎のツール体系をサポートした．

(3)　ユーザカスタマイズ

既存のフォーマットに合うドキュメントを出力できるように，ドキュメント名の変更などができるようにした．

(4)　既存資産からのドキュメント出力

ソフトウェアよりドキュメントが古くなる問題はよく指摘されている．この解決策の一つとして，ディクショナリからのドキュメント出力があるが，より運用が手軽な，既存のソフトウェア資産からのドキュメント出力をサポートした．

以上を中心に，開発／保守時の使い方の違いや表ではなく図形による「見るドキュメント」なども述べた．

4．終わりに

各種のシステム開発方法論・開発手法について，「はじめに」のような各分野からの検討が必要との意見があった．したがって，ここで触れた内容はドキュメントに関する議論の一部である．

ソフトウェア開発の生産性・品質向上を図るためには，ドキュメントをできるだけ少なくかつ正しく作ることが必要と思われる．ソフトウェアとドキュメントを一致させるだけでなく，最近話題の地球環境問題などからも，ドキュメントは少ない方が良い．

このように奥が深い「ドキュメント」というテーマに関し，様々な観点で今後ともＳＥＡでの検討を続けていただければと思う．

以上

SEA 未来予測アンケート（その２）の集計結果

0. はじめに

好評だった９０年代予測アンケート（前々号）に続いて，SEAMAIL 編集部では，３月のフォーラム（ダウンサイジング）などの場を借りて将来の我が国におけるコンピュータのマーケット·シェアについての予測や，会員のみなさん自身の将来への抱負や夢などについて，アンケートを行い，約 65 通の回答を得ました．

1. 1995 年のマーケットシェア

「ある統計によると 1989 年のわが国におけるコンピュータのマーケットシェアは，表１のようになっています．この統計では，WS がどこに分類されているのかよくわかりませんが，90年代はどうやら WS を中心に世の中が回転して行くように思われます．いまから 5年後，日本のコンピュータ·マーケットはどうなっているでしょうか？」というのが最初の質問でした．

1989年度	台数	金額
汎　用	0.5%	44%
制御用	1.0%	24%
オフコン	7.2%	9%
パソコン	91.3%	23%

表1　1989年のシェア

これに対するみなさんの回答の平均値は，次の表２に示す通りです．

1995年度	台数	金額
汎　用	0.7%	32.6%
制御用	2.2%	20.3%
オフコン	4.5%	6.9%
パソコン	70.8%	21.2%
ワークステーション	22.5%	21.8%

表2　1995年のシェア予測

統計学上の大数の法則が働いて，けっかの数字はほぼ常識的な線にまとまっていますが，いずれにせよワークステーション時代の到来はまちがいないようです．なお，解答の中に，合計が100%にならないものが，いくつかありましたので，平均値も合計が100%になっていません．

2. ソフトウェアのダウンサイジングは？

「それでは，ソフトウェアのダウンサイジングはどうなるでしょう？　今日，大規模なユーザでは，ソースライン 1000 K 程度あるいはそれ以上の，戦艦大和的なアプリケーションが稼働していますが，こうした巨大システムは，いまから 5年後にはどうなっていると思いますか？」という第２の質問に対する回答の集計結果は，次のようになりました．

いまとあまり変わらない	30
われわれはみんなメインテナンスの重荷に押し潰されている	9
システムはバージョンアップされ，美しくなっている	2
WS ＋ LAN を用いた分散システムに作りかえられている	14
わからない	2
その他	10

ハードウェア·ダウンサイジングの傾向が確実に予測されるにもかかわらず，ソフトの世界では，「いまと変わらない」「メインテナンス地獄」の到来を予見している現実派が多数を占めています．

その他の中には次のような意見がありました．

笠原俊哉: システムの作り直しが可能であれば少しは楽観視できるかも知れないが，すでに数多くのおお規模システムがそれぞれの社会的役割を担って稼働中であり，５年程度の時間では，基本的に現状と変わらないと思われる．

松原友夫: 両極化する．旧来のスパゲッティ·プログラムの巨大なかたまりは，手をつけられないまま化石化する．置き換えを加速する技術はまだ未熟なので，塩漬けまたは廃棄·新設が主流となる．したがって，旧システムを捨てるだけの勇気と余裕のあるところは，WS ＋ LAN の分散型システムに置き換わる．

林　香: WS ＋ LAN の分散システムに作りかえるようという試みがあちこちでなされるが，結局うまくいかずにメインテナンスの重荷に押しつぶされる．,br

田中慎一郎: 現行のシステムと，分散型システムのどちらも存在しているはず．人間の頭の方は，５年ではあまり変化しないので，「作り直した方が得だ」という事実をユーザがまだ理解できないだろう．新規システムのすべてが分散型であるわけではなく，さらに一部だけである．,br

新保拓也: 部分的に分散システムに作り替えられるが，一部の大規模システムは残る．

吉岡智司: WS ＋LAN の方向を模索するが，過去の資産

が足かせとなり，多くのシステムはまだ変わらないままであろう.,br

中西和男: 大規模アプリケーションはそのまま維持されるが，新規開発分から WA + LAN に移行が始まる.,br

松田憲裕: ５年後程度では，まだ実際のバージョンアップの準備段階だと思われる ,br

船切誠: メインテナンスが重荷になるようなシステムは，「メインテナンスしなくても良い」ということにして，新しいシステムが開発される（といいな）.

田村整: 現在の状況とあまり変化はないが，WS が普及しシステム屋の OA が進む.

3. ソフト技術者としての生き方は？

「これからのソフト技術者の生き方は，アート派（ソフト工房派）と産業派（ソフト工場派）の２つの方向に分化するように思われます．あなたはどちらに属しますか？」というのが３番目の質問でした．回答の分布は次の通りです.

アート派（工房派）.....	34
産業派（工場派）.....	14
どちらともいえない.....	19

およそ２対１の比率で「工房派」が多いようですが，これは現実にそうなのか？　それとも願望でしょうか？

4. あなたのモットーは？

以下は，「1990年代を生きるソフト技術者としてのモットーは？」という第４の質問に対するみなさんからの回答です.

石原由紀夫: 個人的ソフトエキスポを目指す.

渋谷誠:（プログラマは）働きゃいいってもんじゃないよ.

西郷正宏: 技術者ではなく，デザイナーでありたい.

吉沢浩宣: きれいじゃなければ生きられない.

本沢宏: 柔軟の思考.

笠原俊哉: いかに作るかではなく，何のために何を作るかが大切だ．世界と未来を意識した技術者でありたい.

折田豊: 仕事を選ばない.

松原友夫: システムを生き物として有機的に見ること．ダウンサイジングは視点をせばめる危険をはらんでいる．分散化によりシステムは広域的な影響力を持つようになるので，マクロな視点がますます必要になる.

小林俊明: 合理的システム，合理的開発，合理的環境の追求.

片渕清: 高信頼性，高生産性.

島影明: 誠心誠意＋煙草（肺癌の危険はあるが，まくための煙も必要）.

余談ですが，喫煙未経験者より断煙者の方が嫌煙権を強く主張するのは，次の理由の中どれにあたると思いますか？

１．俺（私）はやめたんだという優越感

２．誘惑を感じるから

３．他人にもやめさせてあげようという親切心

上の１に該当する方へ．あなた自身がタバコをやめるまで他人に与えた迷惑の分は甘んじて堪える義務があると思うのですが，間違いでしょうか？

青木淳: 設計事務所にいるか，バスの運転手になっていたい.

新田稔: プログラミングは自由に，リソースは平等に．すぐれたソフトウェアは全人類の共有資産.

林　香: 明るく，楽しい，ソフト作り.

岸田孝一: ９０年代の私は，技術者でなく管理者として生きることになるでしょう（もし生きられれば...）．そのときのモットーは "Let it be!"

中野秀男: 皆さんと楽しく.

田中慎一郎: もっと自由に，もっと柔軟に！

林好一: Transfer できるテクノロジーを創ること.

野村敏次: 過信なく遠慮なく凡々と．１０年後には自由がある.

坂本文: ソフト技術者としては生きないでしょう.

大神原薫: センスとバランス感覚.

升田　誠: 自動プログラミング.

匿名: 技術の進歩に遅れない.

喜代田昭男: カスタマの保護．（色々な意味で）常に将来性を考える.

志賀方宣: Simple is the best!!!

佐藤貞義: 人に優しく接するソフト.

新保拓也: コンピュータ・システムもまた民主化に向かう．エンジニアはユーザと共存の道を選ばなければ孤立するだろう．したがって，モットーはデモクラシー.

中島頼義: ユーザ支援に徹する.

匿名: 自分を含めたソフト技術者と，コンピュータを使う人たちの人間性を守っていきたい.

田中初男: プライベートには，「興味本位でゆらいでいるのがベスト」だと考えている．オフィシャルには，「現実的問題をひとつひとつ解決し，成果を積み重ねていくのがベスト」だと考えることにしている.

窪田芳夫: 普通の人間が普通に利用できるやさしいシステム作り.

熊谷章: 不易流行・マイノリティ・人間性中心主義．システムの進化の仕組みとネタを革新してゆくこと.

佐藤千明: プログラミング・レスの概念によるシステム構築.

今野芳弘: 楽しんで仕事をする.

吉木伸彦: Small is beautiful.

吉村成弘: 趣味と実益の一致.

吉岡智司: 打倒デカルト！ 哲学をもって仕事をする.

咲間繁:「飲みにゅケーション」による他社他業種の方との交流. ソフトはソフト技術者だけでは創造できなくなるだろう.

渡辺雄一: 嫁さん探しとプログラミングの両立.

匿名: コンピュータが単なる道具の１つであることを常に念頭におき，主体性を持って道具を生かして行きたいと思います.

中西和男: ソフトウェアのハード化とハードウェアのソフト化が並行して進. 相互の融合を考えると，ソフトもハードもしっかりと身につけなければならない. そして考え方も.

稲垣浩一: いつまでも技術者であること.

福島秀明: 美しくスマートにありたい.

松田憲裕: 常によい技術をよい技術であると認め，それを受け入れる度量を持ちたい.

船切誠:「プログラムは文章である」と胸を張っていえるようになりたい.「読んで楽しいプログラム」が書けるようになりたい（少し違うか？）. もちろん，その前に基本は押さえておかねば....

笹倉有代: 優雅に生活し，美しいシステムを作る.

平尾一浩: 探る・動く.

磯村克彦: 作って楽しく，使って楽しいソフトウェア.

秦幸男: 外部の力を出来るだけ利用する.

関牧人: 常に，もっとよいものはないかを考えながら仕事をする.

野中哲: 自活.

柘植夏記: 個人的にはアート派として振舞いたいが，業務上は産業派として生きて行くしかない.

平岡信之: 身土不二，晴耕雨鍵（研），公私混同.

上原英美: 創造性を生かせる仕事がしたいし，そういう技術者になりたい.

川端正昭: ユーザー側の立場にたってのソフト開発.

芦塚浩一: プログラミング・テクニックの向上よりも，システム構成・操作性を重視するソフト開発を心掛けたい.

山本和文: 人間味あふれるソフト作り.

野口貢: ソフト技術者としては限界を感じる. 別ジャンルを開拓して楽しく仕事ができればよい，ただし，プログラミングができる人間でありたいと常に思う.

匿名: コンピュータは人間の仕事の一部を肩代わりするのが目的なので，その本質を外さないこと.

5. みんなの夢

さて，最後の質問では，「いまから 5年後には，いろいろ新しいコンピュータのアプリケーション分野が開拓されていることでしょう. あなた自身は，どんなアプリケーションをどんなふうに手がけてみたいですか？」ということで，皆さんに夢を語っていただきました.

渋谷誠: カナ漢字変換に代わる革命的日本語入力システム. 世の中からじゃまなキーボードを消滅させる革命的文章入力システム.

西郷正宏: アプリケーション・ソフトは，基本的にはユーザが作るものになるでしょう. では，われわれ（そして私）は，何を作ればよいのでしょうか？ ツールを作る一部の天才と，そのコピー商品を作るプライドのない技術者（商売人）たち. そして，あとの大部分は，あらゆる日用品の中に埋め込まれたチップのファームを作っているのではないでしょうか？ つまり，この時代に純粋なソフトウェアを作れる人間は，高い創造性をそなえていることを意味します.「プログラマの復権」はこうして達成されるのです.

吉沢浩宣: ネットワーク OS の中で増殖する電脳 B-ing !?

本沢宏: グラフィクスはすでにある程度コンピュータに取り込まれているが，サウンドはまだもう一歩である. こうした分野に関わって行きたい.

笠原俊哉: 参加したいプロジェクトのテーマは，地球環境・宇宙開拓・生命関連のシステム・アプリケーション. ぜひ，そうした開発を手がけてみたい.

折田豊: 具体的な分野はわからない. そのソフトを使うとワクワクしてきて，いろいろなアイデアが涌き出てくるようなものを作りたい.

松原友夫: 民主化を支え，社会の混乱を防ぎ，地球を救うシステム. 情報技術の進歩は，昨年の東欧における民主革命をどれだけ助けたことか？ その反対に，証券取引システムの電子化はデイレーなしの過剰反応によって混乱をもたらした. 夢としては，公害発生源を即時に摘発し，他に被害が及ばないうちに対処できるような公害モニタ・システムを開発して，地球を救うことはできないだろうか？

小林俊明: これまでにそれぞれのユーザあるいはメーカが勝手に作って来たアプリケーションを，もっと保守しやすく，機能拡張に手間のかからない姿に変えられるようなプラットホーム（分散オブジェクト指向環境）の開発を手がけていたいですね. それには，まず，自分自身の開発環境を

もっとよくしたい．在宅勤務で，仕事のファイルは自宅にリモート･マウント．保守作業も客先のマシンにリモート･ログインで OK．東京に出て行くのは SEA のフォーラムぐらいにしたいものです．

島影明：アプリケーションの開発の世界は相変わらずの日々でしょう．エンドユーザーにとって真に使い勝手のよいシステムを作りたいものです．

青木淳：メジャーになったら終わりよね．マイナーなアプリケーション分野にいると思う．標準化のプロセスに入ったようなものは手がけていないと思う．「パワー／スタミナ／スキル」のうち，どれかといわれれば，スキルが求められるアプリケーション分野．どんな分野なのかは不明．

新田稔：スーパ･ユーザ･インターフェイス，つまり，ユーザが何かやろうあと思った瞬間にそれができている．ファジィなオブジェクト，つまり，何やらわからないものが，わからないままとりあつかえる．世界中のマシンが階層的ネットワークにつながっていて，上位マシンのクラスやメソッドを使うことができる．

林　香：Mac のイメージ･ビデオの Knowledge Navigator のような人間と楽しく（フレンドリー）に対話しながら，人間の手助けをするソフトウェアを作ってみたい．

岸田孝一：プログラミング･グループがネットワークと WS を駆使して仕事をする環境を，何とか９０年代前半に完成させたい．その現場でのカオス的な並列プロセスの稼働状況を，何とかモニタリングし，コンロトール（よい意味で）することを考えたい．

中野秀男：本物の秘書つきの秘書システムを作りたい．そして，少しまじめに，アルゴリズム･アニメーションまたは Scientific Visualization もやってみたい．

林好一：分散ヘテロ環境上で OA ツールを使ってみたい．たとえば，バカでもわかる DB マネジャとか，バカでも使える文書作成システムとか，バカでは扱えないがリコーにはすごく便利なプロジェクト運営シミュレータとか．

野村敏次：正直申しまして，過去５年間には，目ざましいハードウェア（特に WS）の進歩とか，ネットワークの普及，さらには DB の進歩はありましたが，アプリケーション開発の分野では，多少のはやりすたりはあったものの，それほど新しい展開はなかったと認識しています．したがって，今後の５年間にも，残念ながら革命的に新しいアプリケーション分野は開拓されないだろうという悲観的見解に立たざるをえません．

大神原薫：対象はともかく，本来の意味での著作権で保護されるような，エレガントなソフトウェアを作りたいですね．あ，それから，使いやすいソフトウェアですね．そのためには，たぶん Window や GUI が必要だと思います．

本橋隆夫：総合経営システムのデザイナーとして仕事をしたい．そして，いろいろな新しいアプリケーションを取り込んだ形で，総合的なシステムのコーディネーションをやってみたい．

三浦眞太郎：EWS を使ったグループによるプログラム開発．しかし，EWS のパッケージ･ソフトの流通がどうなるか，なかなか先が見えない，NEWS の日本語処理は最低である．いずれ，すばらしい日本語処理パッケージが出てくることを期待しない．

升田誠：ホストのソフトウェアを何とか EWS 上で作りたい．よいツールを使いたい．

匿名：個人商店や小企業向けに，今の世の中で売られているようなパッケージではなく，すぐ役に立つ簡単な操作のソフトを作りたい．そうしたユーザが自分でソフトを作るのはむずかしいし，現在のようなパッケージでは小回りがきかないなどの問題がある．簡単にソフトがジェネレーションできれば，もっとコンピュータ化が進むだろう．

喜代田昭男：遺伝子解析システムの構築．

志賀方宣：ホーム･コンピューティング．５年後の家庭には，現在のパソコンとは異なる新しい形のコンピュータが入って来るでしょう．その時のコンピュータを考えたい．Mac でも NeXT でもなく TRON でもないもの．その上でのサービスを考えたい．

佐藤貞義：業務実行環境としてのシステム．

新保拓也：ニューロ･コンピュータ，ファジィ，ＡＩ等の分野で，より人間的なシステムの開発を手がけてみたいと思う．

中島頼義：電話のように，コンピューティング･パワーがどこからでも入手できる．どこからどこへでもつなくことができる．そんな世の中になるように，ソフトを提供したい．

匿名：ソフト開発支援，システム導入支援などのシステムを，人間工学や心理学のアプローチにもとづいて設計してみたい．

田中初男：人の感情に影響を与えるようなプログラムを手がけてみたい．具体的な方法は，まだよく考えていないが，ゲーム以外にも，手にアセを握るようなアプリケーション分野がでてくると思う．

窪田芳夫：スリムで単純なソフト．稼ぎは本数を売りまくればよい．

熊谷章：知的支援システム（知的活動を支援するツールとしてのコンピュータの開発）．マルチメディア･システム（出版，放送，コンピュータの３業種を混ぜた新しい業界の

確立).

咲間繁: ますます新しいアプリケーション分野が開拓され，増殖をつづけ，ユーザは自分の仕事に何を適用し，どれを応用すればよいかが判断できなくなる．そこで Fortran をかじったことのある私が，そうした人々のために，専用のシステム(決して汎用ではないスパゲティ・プログラム)を，goto 文を大量に使用してコーディングしてあげる．まさに歴史は繰り返す．

田村整: 十数年前の「2001 年宇宙の旅」が再認識されているが，そこに登場した HAL のソフトを作りたい．人間の言葉でコミュニケーションし，自分で判断し成長する(プログラムを蓄積する)機械の基本ソフトを作って見たい．

渡辺雄一: メンテナンス・フリーなアプリケーション・システム．そのためには，個々人の Viewpoint を保持する仕掛けが，システム内に必要になるはず．あとは，やっぱり Visual Programming の基礎的な裏付けと，CSCW との融合でしょうか．

匿名: 現在，情報管理の方法としては DB が一般的ですが，これはあくまで特定のターゲットを定めた効率重視の手法であり，人間の持つ偶発的な創造能力(連想など)に必要なマルチターゲットのランダムアクセスには適していません．そうした創造性を生かすためには，思い切って「コンピュータとお別れする」のが一番なのかもしれませんが，私はあえてコンピュータの持つ能力を生かし，なおかつ創造性を刺激するな情報管理方法について考えてみたいと思います．

中西和男: ユーザが CPU を意識しないですむとような，マルチ・マクロなコンピュータシステムにおける OS．ネットワーク上で動作し，ユーザがマシンを指定しなくてもすむような分散型の OS．残念ながら具体的アイデアは，まだない．

福島秀明: SE に停年はないと思っていた．しかし，何年か振りで現役に戻ってみると，開発環境の変化などへの適応が非常にむずかしい事を感じた．より高いマクロな視点から，システム・インテグレータとしての方向性を打ち出すという立場で，再度チャレンジしてみたい．

松田雅裕: ・巨大なオンライン・サービスなど，ネットワークを用いた情報流通に関するアプリケーションを，あくまで人間主体のシステムとして構築する立場に立っていたい．人間の心理的側面も考慮した MMI の研究をしてみたい．

船切誠: きっとプログラムを書くのは大変だとは思いますが，ものすごく小さくなったコンピュータが互いに協調してうごくような環境の仕事をしてみたいと思います．TRON の電脳住宅のような感じのもの．しかし，特に日常生活に密着したものではなく，「数十台で１台のコンピュータ」「群体コンピュータ」上での仕事がいいです(なんだ，ただの単に並列コンピュータの話になってしまったではないか！)．

曝村克彦: ターゲット・システムをもっと楽に設計したい．そのために MMI 設計支援ツール(NeXT Step のようなもの)や，オブジェクト指向データベースを有効利用して，自分では極力プログラムを作らずにシステムが構築できる環境を用意したい．分野としては，モノ作りにこだわる CIM 指向が続くと思う．

秦幸男: できるだけアプリケーションを手がけていないことを望む．無理なら画像処理関係かな？

関牧人: あれは名作だったと語りつがれるようなソフトを作りたい．

野中哲: 広い作業空間，ファンの無い静かな WS，東京脱出！(仕事の内容より，作業環境をどうにかしたい．なさけない話だが真実だ)

柘植夏記: あんまりアプリケーションを手懸けたくないなぁ．

山下俊弥: LA を中心としたシステムの解析・設計・提案を行なう．LA/FA 用アプリケーションを開発するための OS やツールを統合化して行く．

川端正昭: 現在はまだプログラマとしての経験が少なく，私自身，夢というよりも現実的な仕事の面で精一杯２～３年がんばってから，ゆっくり考えてみたいと思います．

芦塚浩一: ソフトウェア開発環境が整備され，生産性の向上が進み，除々にユーザ・レベルでの開発も可能になって来るようになるだろう？

平岡信之: 信州の森の中で，唯我独尊的に，あるいはネットワーク・コミュニティの協力をもらって，「地球にやさしい」エコマーク・アプリケーションを作る．ソフトウェアとネットワークの力で，紙やガソリンなどの無駄な資源消費を減らしたい(今は逆に増やしてるみたいだけど)．

山本和文: コンピュータをあまり知らない人でも簡単に使えるようなパッケージを作ってみたい．

笹倉有代: AI を使って的中率 90% 以上の占い(四柱推命)システムを作る．そして老後，ラップトップをひっさげた街頭占い師に転向する．どなたか，ディスプレイが水晶球の形をしたパソコンを開発していただけませんか？

平尾一浩: やっぱ，鉄腕アトムみたいなロボットを作りたい．そして，悪いやつらをやっつけなきゃ！

石原由紀夫: いわない．いったらまねされる．

5月のSEA フォーラム速報

はじめに

5月の月例フォーラムは，新年度の年次総会に先立って，5月 15日（火）の午後に，「ファジィ vs ニューロ !? − 90年代の新しい計算モデルをめぐる講演と討論」と題して，機械振興会館（東京·港区）で開催されました．

当日は，まず，基調講演者として明治大学の向殿政男先生をお招きし，「ファジィ理論とその応用」と題して，ファジィ理論の起源から今日に至るまでの経緯，ファジィ制御技術の応用の現状，さらにはファジィ·コンピュータの今後について，お話しいただきました．

それに続くパネルでは，向殿先生の他に，ニューロの立場から富士通の宇佐美仁英さん，アルゴリスム理論の立場から大阪大学の中野秀男先生を加えて，ファジィ，ニューロその他のキーワードを中心に，90年代を彩るであろう新しい計算モデルをめぐっての討論を FXIS の藤野晃延さんのコーディネーションのもとに展開しました．

テーマもタイムリーで，また，スピーカの選定も適切であったせいでしょうか，広い会場がほぼいっぱいになるくらいの盛況で，会場からの質問やコメントも活発に出て，楽しい Forum でした．そのことは，以下に収録した参加者アンケートからもうかがえるでしょう．

当日の講演および討論を録音したテープは，今回の企画を担当した新代表幹事の熊谷さんがいま整理中(？)ですが，ここでは，とりあえずの速報として，参加者のみなさんにお願いしたアンケートの結果をまとめておきます．

また，基調講演者の向殿先生が当日配布された資料(オリジナルは，電子情報通信学会のヒューマンコミュニケーション研究会の第1会会合で先生が発表された論文）を 35ページ以降に再録しました．あわせてお読みください．

アンケート結果

Q1 あなた自身は，今日の Forum で話題になった新しい計算パラダイムについて，どの程度興味を引かれましたか？

結果は次の表に示す通りですが，これを見ると参加者のほとんどが，ファジィやニューロに大きな興味を持たれたようです．みなさん，日常の仕事でガチガチのアルゴリズムに，よほど苦しめられているにちがいありません．

	ファジィ	ニューロ
非常に興味深い	48	25
興味を持った	9	25
それほど興味ない	0	7
ほとんど興味ない	0	0

Q2 それぞれのパラダイムが近い将来に実用化され，いろいろなコンピュータ·アプリケーションに利用される可能性については，どうお考えですか？

	ファジィ	ニューロ
かなり高い	36	17
ある程度は	23	32
それほどない	0	7
ほとんどない	0	1
わからない	0	2

この回答もほとんどがポジティブです．はたして，将来のアプリケーション·パラダイムは大きく変わるのでしょうか？ 未来予測アンケート（その2）のほうでは，かなりネガティブな回答も見られたのですが....

Q3 今日の Forum を聞いての一般的な御感想は？

参加者のみなさんから寄せられたコメントは以下の通りです：

(1) ファジィ理論については以前から興味を持っていたが，確率論的アプローチとの相違が，今日の向殿先生のお話しを聞いてよく分かった．ファジィ制御の考え方を，ソフツェア開発プロセスのマネージメントに応用したらどうなるだろうかと，直感的に考えている．そのためには，開発プロセスのモデルをファジィ·システムとして表現できなければならないが，さて？

(2) 向殿先生のいわれるように，「ソフトウェア開発·運用」のありかたを，そろそろコンピュータの世界から人間の世界へ持って行かなければならないと痛感しています．まさしく理系の発想から文系の発想への転換が要求されていると思います．

(3) 最先端の技術動向が聞けて楽しかった．実際の応用面において，どんな場面にどのパラダイムを適用すべきか迷うことが多いとおもわれるが，ファジィ·ニューロそれぞれ

の限界について，または利点·長所などをまとめて示していただければ，もっとよかったと思う．

(4) 問題を切り分けけて，どのパラダイムを適用するかの判断は，人間の知恵によるしかなく，コンピュータに頼るわけに行かないのは当然である．その意味で，ファジィやニューロの応用可能な範囲をもう少し明確にして欲しかった．

(5) ニューロについては当方の予備知識の不足のせいか，あまりよく理解できなかったが，ファジィに関する向殿先生の話し方にはつい引きこまれてしまった．これからも応用の動向を注目して行きたい．

(6) 大変おもしろかった．人間と機械のコミュニケーション問題には，以前からとても興味を持っていました．ファジィの話でそこまで聞けたのは予想外で，とてもうれしかった．

(7) 一般的な知識しかなかったので，楽しく勉強できました．もう少し時間をとって，先生方のお話をじっくりお聞きしたいと思います．ぜひ，秋のセミナー·ウィークで取り上げてください．

(8) Forum への参加は初めてでしたが，非常におもしろかった．今度は「朝まで生テレビ」の形式でやってみては？

(9) ソフトウェア開発の現場では，要求仕様の細部をすべて厳密に定着してからソフト作りに入るというよりは，瞬昧な仕様をそのままにしてプログラミングを行い，結果としてうまく動けばそれで OK ということが多いように思われます．そのような意味では，われわれの仕事は本来「とてもファジィ」なのだと思います．ただ，理論的な裏付けがあるのと，成り行きで（仕方なく）やっているのとでは大違いで，そのために色々なトラブルが起きていることも事実です．ファジィ理論が，アプリケーションの解法パラダイムとしてだけではなく，われわれ自身の仕事のプロセスに対しても，何らかの救いを与えてくれることを期待したいのですが....

(10) 満足しています．向殿先生のお話はたいへん参考になりました．今後は，ますます人間に近づいた形のシステムが必要とされるようになるでしょうし，記号によるコミュニケーションには限界があるのですから，それを認めたうえで，新たなアプローチを試みる必要があるでしょう．

(11) そろそろコンピュータ·システムの世界にも，パラダイム変化が起こっていいころだと思っていたが，自然界の物質モデル（脳や神経ネットワーク）や社会の経験モデル（メンバーシップ関係）など，われわれの身近な事実から，小さなシステムの全体を全体として把握するモデル化手法があらわれてきたのは，たいへん興味深い．

(12) ファジィ理論を応用して，人間の心の揺れをコンピュータで表現することができると，おもしろい．そうなれば MMI の設計のやり方も変わってくるだろう．フライデーや C3PO がつくれるかな．

(13) この業界の現実にどっぷりつかっていると，ファジィ的な考え方はトラブルの原因になるようなケースが多いので，仕事に利用するのはむずかしいでしょう．でも，設計フェイズでは，発想の転換のために，ときにはファジィ的なアプローチが必要かな，とも思います．

(14) ファジィにしろニューロにしろ，話題としてジャーナリズムなどではやしたてられているが，単なる可能性だけでなく，その欠点や限界についての話題も含めた今回のような議論が，もっと広まればよいと感じました．

(15) 生体細胞を利用したコンピューティングの話はどこまで進んでいるのかと，思いをめぐらせた．生体細胞の抗体原反応等は，非常に大きな超並列計算でもあり，文字通りのニューロであると思う．

(16) 向殿先生のお話にとくに興味を引かれました．特に，「コンピュータは形式的なものしか扱えないが，それでも，どうにかして意味的なデータを扱えるようにしなければならない」というあたりです．

(17) 今後のコンピュータがより人間に近づいていることをあらためて再確認しました．パネラーの方々のお話も非常におもしろく，参考になりました．

(18) あいまいさ，不確実さに対するさまざまなアプローチは，非常におもしろいものだと思います．特に感情の取り扱いについて興味をおぼえました．(19) 感情や感性をどう扱うか？ その表現方法や意味づけは興味ある問題です．MMI の改善のためにも，今後も勉強して行きたいと思います．

(20) 細かい技術面は別にしても，論理や考え方の方向性として新しい視野が広がりました．非常に有意義でした．

(21) 大変興味深くお話を聞くことができました．人間工学との統合が今後さらに期待されます．

(22) 今まで，あいまい（ファジィ？）であったものをはっきりさせていただいて，感謝しています．ファジィ·データベースのお話を聞いて，自分でも作ってみようと考えました．

(23) 全体的に，私にとっては難しい話しが多かった．向殿先生の講義は，学生の時に受ければよかったと後悔して

いる．

(24) 大変おもしろかったが，多少雑談や脱線が多かったかも知れない．もう少し現場に密着した話が聞きたかった．

(25) いろいろな計算パラダイムの現状が理解できた．

(26) 私は文系の人間ですが，ファジィ理論の説明は非常にわかりやすく，ためになりました．言葉のシャワーをあじわいました．

(27) ファジィ理論の応用は，コンピュータを人間に近づけるものではないかと思っていたが，その通りだったのでほっとしました．ファジィでありニューロであろうななロボットの制御を考えてみたいと思います．

(28) 思ったより楽しかった．また続きを聞いてみたい．

(29) ソフトウェア開発の泥沼にあえいでいるわれわれの心の中にたまりつつある色々なストレスを解消するための手助けとなるのではないかという漠然とした期待を持ちました．

(30) データ量が多くなってもシステムがつぶれないように，ファジィやニューロのパラダイムを応用して，データを認識する新しいインターフェイスが作れないかなと，お話を聞きながら考えていました．

(31) 今後はファジィ理論を応用した家電製品を買うことに決めました．

(32) それぞれのパラダイムの説明を聞いて，相互関係について整理することができた．こうした機会を得られた感謝している．私の中に結論めいたものが出たわけではないが，いくつかのヒントが得られたと思う．

(33) ファジィ，ニューロ，並列計算と，いままで知らなかった知識を得ることができてよかった．今後も Forum には出席したいと思った．

(34) ファジィ演算で結果として空集合が得られた場合の処理方法はどうなるのかな？

(35) 楽しい時間を過ごした．もう少し時間が長くてもよいと思う．

(36)「コンピュータと意味」という将来かならず必要となる問題を，わかりやすく扱っていただいて，大変有意義であった．

(37) ニューロはファジィ理論を実現するためのよい道具である．ニューロ＋ファジィ理論は，エキスパート・システム構築のプロトタイピング・ツールだ？ ニューロ＋ファジィ理論で，解が分からない問題を解くためのルールを見つけられる可能性がある．つまり，CASE に応用すれば効果があるだろう．

(38) 久々におもしろい話を聞いた．どうしても仕事に追われて，あの学生の頃の知的興奮が少なくなっていたところでした．

(39) 私はきょうの話題に関しては素人ですが，これらの新しいパラダイムは，本質的に分離できるものではないと感じました．たいへん楽しく，おもしろく思いました．

(40) 本には書いてなさそうな話がたくさん聞けてよかった．

(41) ファジィ集合の概念を聞いて，自分自身の数学観が大きく広がったような気がした．

(42) 最終的にルールは人間が作るという点では，AI と同じように感じた．

(43) 予備知識がまったくなかったが，３時間の講演とパネルを聞いて，何故 'vs' なのか若干わかったような気がするし，ファジィやニューロに関する興味も湧いてきた．自分なりに勉強してみたいと思った．

(44) 大変楽しい Forum であった．テーマも討論内容も，ともに聴衆（SEA メンバー）のレベルにあっていた気がした．これからの Forum も，このような新パラダイムをベースとした技術的テーマを取り上げ，その応用に関しての議論を展開したほうがよい．今回のテーマに関しては，いずれ，プログラミングのやり方やモデルの記述の具体例について，議論してみたい．

Q4 これからの SEA Forum のテーマについて

みなさんからの御希望をまとめると，およそ次のようになりました．これを参考に幹事会で秋以降の Forum の企画を考えて行きたいと思います．

これからの MMI (Multi/Hypermedia)	15
ネットワーク & 分散環境	12
オブジェクト指向	11
ハードウェアの動向（新アーキテクチャ）	9
これからの OS（MACH, OS/2, TRON, etc)	4
CSCW, グループウェア	4
ビジネス・トピックス（SI, SIS など）	4
AI の今後	3

ヒューマンコミュニケーションとファジイ処理

'Human-Communication and Fuzzy Processing

向殿政男
Masao Mukaidono
明治大学理工学部情報科学科
Department of Computer Science, Meiji University

1. 情報とコミュニケーション

"情報"という言葉を訳語として最初に創作したのは、文豪の森鴎外だと言われている。インフォメーションに相当するとドイツ語から訳出したものである。英語のインフォメーションは"型通り(form)"の否定形、すなわち通常とは異なったものという意味に由来していると思われる。一方、"情報"は情(なさけ)ある報(しらせ)と読める。情(なさけ)とは感情や人情の情であるが、ここでは"意味"ととることもできよう。こう考えると、情報という言葉は情報の本質を衝いている。情報とは意味の報せであり、報せるためには何らかの形としてその意味を表現しなければならない。ここに、情報が宿命的に持たなければならない二つの側面、すなわち、意味(セマンテック)と形式(シンタックス)とが存在する。情報とは意味と形式との混合物である。

現代は高度情報化社会に突入したと盛んに喧伝されている。この高度情報化社会をもたらしたものは、情報に関する科学技術の発達であることば論を待たない。すなわち、情報の処理(コンピュータ)、情報の伝達(コミュニケーションとネットワーク)及び情報の蓄積(データベース)に関する科学技術が、エレクトロニクス技術とディジタル技術との融合により実現され、高度に発展したことによる。ところで、ここで注意しなければならないのは、情報の意味の側面と形式の側面とを考えた時、上記の発達は、形式の側面のみに関してであることである。すなわち、形式としての記号を、いかに高度に高速に、誤りなく大量にそして安価に処理をし、伝達し、そして蓄えるかという技術が発達したのであって、決して意味の側面に関する技術が発達した訳ではないのである。よって、現代の高度情報化社会は、ある意味では偏った発展をしているということもできよう。情報の形式の面での科学技術のみが極めて高度に発展しているのに対して、意味の面に関する学問は、残念ながら取り残されているからである。

コミュニケーションとは、情報をやりとりすることにより、お互いに意志を通じ合わせることである。ここでの情報がはたしている役割のうち、意味の側面と形式の側面とを考えた場合、大事なのは、もちろん意味の方である。コミュニケーションを成立させるためには、意味を理解することが必要である。ところで、理解するということはいったいどういうことなのであろうか。ここで極めて困難な哲学的問題が発生する。

人間と人間との間でのコミュニケーションは、主として言葉が用いられる。言葉は、イロハやアルファベットという形式的な記号を用いて表現されているが、各記号列には、ある意味が対応している。人間はもちろん意味の方で理解をしている。言葉にも意味と形式とが同時に存在する。このように、言葉は情報の一つの典型的なモデルとなっているのである。あるコンピュータと他のコンピュータとがネットワークを介して通信をしているとしよう。これは、民族と

機械との間でコミュニケーションが成立していると言えるのであろうか。機械同士は形式上の世界であり、通信されている内容は実際は0と1との記号列である。ここで、両者は一般に記号体系が異なっているので記号列間の変換が必要である。そしてこの変換される記号列も通常、情報と呼ばれている。コンピュータが他のコンピュータからの情報を得て、人間が意図したように正しく、例えばあるものを制御したとしよう。これは二つの機械の間でコミュニケーションが成立して、意志が通じ合ったと言えるのであろうか。これも前出の哲学的な問題に少なからず関連している。

しかし、もっとむづかしいのが人間と機械であるコンピュータとのコミュニケーションである。これが本論で問題としているヒューマンコミュニケーションの問題である。人間（Man）同士、機械（Machine）同士、および人間と機械との間のコミュニケーションでお互いの接点であるMMインタフェースを考えたとき、ヒューマンコミュニケーションの問題は必ず最初にマン・マシンインタフェースの問題として浮上してくる。

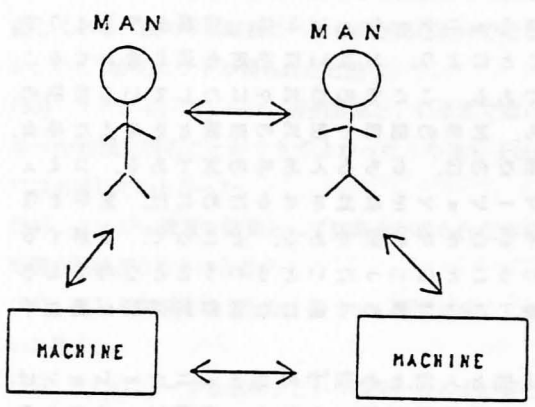


図1　コミュニケーションとMMインターフェース

2、マンマシンインターフェース

機械は人間の能力を拡大するための道具として発明された。人間と機械とが補完的にお互いの特徴を生かして、ある目的の仕事を遂行するためのシステムは、マン・マシンシステムと呼ばれる。ここで重要となるのが、人間と機械との機能的接点であるマン・マシンインタフェースである。従来、マン・マシンインタフェースの研究は、人間工学的な立場から、具体的な効率や、動特性などの人間も含めて機械的側面が重視されて来た。しかし、コンピュータの出現は、従来のマン・マシンインタフェースの発想を根本から考え直させるキッカケとなった。機械的側面ではなく、情報の流れというまったく異なった側面に注目しなければならなくなったからである。

コンピュータは、人間の知的活動を拡大するための道具であり、人間とコンピュータとが共同で、それぞれ単独ではできないような創造的で発見的な仕事をするのが目標になって来ている。このためには、人間にわかりやすい表示をどうするか、人間の意図をいかにコンピュータに容易に伝えられるかが重要な点となろう。従来、コンピュータはコードである記号の世界だけで人間とコミュニケーションを行ってきた。両者の間での情報の担手は主としてアルファベットや数字からなる記号列のみであったと言って良い。しかし、人間が言葉でコミュニケーションを行っていると言っても、顔の表情や声の上げ下げなど、記号としての言葉以外の情報を用いたコミュニケーションも大きなウエイトを占めていることは明らかである。このことは、パソコン通信によるコミュニケーション.がいかに意図が伝わりにくく、そして、微妙なニュアンスが伝わらないのでイライラさせられることからも良く分かる。よって、近年、人間とコンピュータとの間も、記号のみでなく、絵や音などマルチメディアを用いたコミュニケーションが重要視されるようになって来たのは当然の成り行きであろう。ただし、記号はもちろんであるが、絵も音もコンピュータは最終的にはすべて0と1のビット列で表現し、処理していることには変わりはない。ここで重要なことは、人間の論理とコンピュータの論理とは根本的に異なるという認識である。コンピュータは情報を取り扱っているが、結局は情報の形式的側面、すなわち記号のみに着目しているのであり、コンピュータは情報を形式的にしか取り扱い得ないという事実である。しかし、前述したように、情報の大事な面は意味の方なのである。人間は、分かる分からないという意味の世界の論理であ

り、コンピュータはイエス、ノーの２値に基づくビット列で表される記号の世界の論理である。

これまでは、人間とコンピュータとのヒューマンインタフェースにおいては、人間が無理に機械の論理に合わせていたといえよう。しかし、取り扱うべき情報が大量に、かつ複雑になってくると、コンピュータの専門家でさえコンピュータとの対話を思み嫌うようになるであろう。余りに非人間的であるからである。更に、これからの高度情報化社会では、専門家のみでなく一般の人もコンピュータとつき合うようにならざるを得ないが、非専門家は人間の論理でしかコンピュータと対応しない。従来のままでは、全な形でコンピュータが社会に浸透して来るとは思えないのである。コンピュータに情報の意味の部分も取り扱わせないと、人間と機械との真のコミュニケーションは実現できないのでないか。これがマン・マシンインタフェースにとっての根本問題である。ただし、何度も記すが、コンピュータはあくまでも情報を形式的にしか処理できないのである。よって、そこで意味を形式的に取り扱うための理論が是非とも必要となる。アナログ情報はディジタル情報よりも、より意味に近い表現と言える。表している内容とその表現形体（意味と形式）とがアナログ情報の場合には強い関係があるのに対して、ディジタル情報の場合には約束ごとにすぎないのである。アナログ情報にはリアリティがあるが、ディジタル情報は約束の世界なのである。よって、絵や音などのマルチメディアや、ウインドウやマウスなどは見方によっては意味を形式的に表現している一つの手法とも考えられる。確かにこれまで、意味ネットワークやフレーム理論など、意味を形式的に取り扱うための理論はあったが、それ程成功しているとは思えない。

先程、言葉は情報の典型的な一つのモデルになっていると言った。人間同士が言葉でコミュニケーションをするように、人間とコンピュータとが同じような言葉でコミュニケーションができれば、人間の側に立ったマン・マシンインタフェースと言えよう。しかし、言葉とそれが示す意味とは、１対１に正確に対応しているのであろうか。言葉の意味をイエス、ノーの２値論理を用いて正確に定義できるのであろうか。自然言語を考えてみれば、答えは明らかにノーである。言葉の意味には正確に定義できないようなあいまいさが存在している。それにしかかわらず、人間はコミュニケーションをし、理解し合えるのはなぜだろうか。人間にとって、あいまいさは本質的であり、避けて通る訳にはいかないようである。よって、マン・マシンインタフェースにとってもあいまいさの問題は本質的なものである。次にあいまいさについて考えてみよう。

３．ファジィネスというあいまいさ

あいまいさを真面目に取り扱っている数学的理論と言えば、まず、確率論が挙げられる。確率論で取り扱っているあいまいさは、サイコロの目の数のように、振ってしまえば１の目が出たか否かが明確になるようなあいまいさ、すなわち、本来はイエスかノーかどちらかで答えられるのだが、まだ起きていないからあいまいであるという生起に関するあいまいさである。ここで重要なことは、本来はイエスかノーかで答えられるという事象を対象としていることである。このようなあいまいさはランダムネス（ｒａｎｄｏｍｎｅｓｓ）と呼ばれている。

さて、それでは"スピードが早い"とか"中年である"とかの言葉の意味は、イエスかノーかで厳密に答えられるような事象であろうか。このように言葉の意味にまつわりついているようなあいまいさは、すなわち境界が不明確で、しかも主観によっても異なるようなあいまいさはサイコロのいくつの目が出るかのようなランダムネスのあいまいさとは本質的に異なると言ってよい。このようなあいまいさはファジネス（ｆｕｚｚｉｎｅｓｓ）と呼ばれている。ファジネスのあいまいさはよく考えてみると我々の日常生活には満ち満ちている。従来の科学技術は、あいまいさを排除し、物事を明確にする方向で発展して来ており、主観にまつわるようなあいまいさは研究対象として来なかったと言ってよい。しかし、前述のように、人間にとってあいまいさは本質的であり、避けて通る訳には行かないという認識の下に、ファジネスというあいまいさを定量的に表現して積極的に取り扱おうという理論が、ファジィ理論として提案さ

れ、最近、にわかに応用の場面を通して脚光を浴びている。

4、ファジィ集合とファジィ理論

ファジィ理論は、ファジィ集合、ファジィ測度及びファジィ論理を基本として、現在、理論体系として発展整備されつつある学問分野である。近年、ファジィ論理の一部であるファジィ推論の一つの応用として、ファジィエキスパートシステムが、制御や株式投資などに応用され、その有効性が実証されつつあり注目されている。ファジィ理論は、現在発展途上にあり、学問体系として確立されているとは言いがたい面もあるが、多くの実際の現場で役に立っている点を考慮すると、将来面白い有望な分野に成長する可能性が大きい。

ここでは、ファジィ理論についての深入りは避け、ファジィ理論の基礎の一つになっているファジィ集合の考えを簡単に振り返ってみる。

通常の集合では、ある集合Aにある要素xは、所属しているか所属していないかのいずれかである。これは、特性関数λ_Aを用いて、

$\lambda_A(x)=1$…xはAに所属している

$\lambda_A(x)=0$…xはAに所属していない

と表現できる。特性関数λ_Aは全体集合をUとするとUから{0、1}への写像

$\lambda_A: U \to \{0, 1\}$

を表している。図2（a）は初老及び老人の特性関数の例を示したものである。64歳は老人ではないが1歳年を取って65歳になると途端に老人になってしまうという不自然さがある。これに対して、集合Aに、ある要素xが所属している度合を0と1の中間の値を用いて表現しようという発想がファジィ集合の考え方である。例えば通常の特性関数の代わりに、メンバーシップ（所属度）関数μ_Aというものを用いて、例えば

$\mu_A(x)=0.8$…xがAに所属している度合は0.8

という表現を認めるものである。すなわち、メンバーシップ関数は、全体集合Uから閉区間[0、1]への写像

$\mu_A: U \to [0, 1]$

を表している。図2（b）は初老と老人とのメンバーシップ関数の例である。所属の度合が徐々に変わって行くので、従来の集合のような不自然さはない。ファジィ集合は、従来の集合の境界をあいまいにしたものである。これに対して、従来の集合はクリスプ集合と呼ばれることがある。

ファジィ理論は、ファジィ集合の考え方を基本として、研究対象にあいまいさの存在を認めた新しい理論である。ここでは、メンバーシップ関数を客観的に決めるということは余り重要視していない。客観的に決められないあいまいさの存在をはじめから認めているもので、メンバーシップ関数の形は使用者が主観的に自由に決めるとしている。ここにファジィ理論の良さがあり、これにより、言葉にかかわる関する主観的なあいまいさをメンバーシップ関数の形に吸収させてしまっている。ただし、逆に、ここが批判の集中するところでもある。例えば図2（a）のクリスプ集合では、初老と老人という概念の共通部分を見い出すことは困難であるが、ファジィ集合（b）では、両メンバーシップ関数の重なり具合いから、両者の概念が共通する度合なども数値として表すことができるのである。

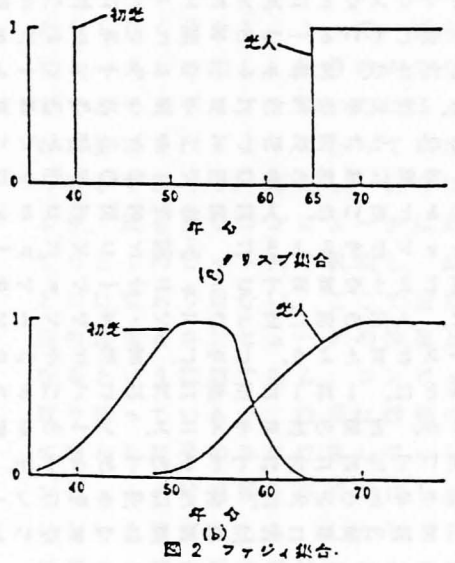


図2 ファジィ集合

5. ファジィ集合による意味の表現

従来の人工知能研究では、記号処理を主としていた。このため、例えば、"毛人"と"初毛"という二つの言葉がコンピュータに入力されても、記号レベルで見る限り両者は一致していないのでまったくの別物としてしか取り扱うことができなかった。ところが、図2（b）のファジィ集合による考え方を用いると、両者の概念の一致の度合、すなわち似ている度合などを表現することが可能である。集合は、通常、集合の名称とその特性関数という2つのものの混合物であると見なせる。ファジィ集合では、このことを特に強調していて、ファジィ集合の名称（ラベル）とそのメンバーシップ関数とを意識的に分けて使用している。よって、たとえば、初毛という記号としての言葉をファジィ集合の名称に、そしてその意味をメンバーシップ関数に対応させればファジィ集合は、情報としての言葉の一つのモデルになっていると考えられないであろうか。こう考えれば、きわめて単純であるが図2（b）のメンバーシップ関数をコンピュータに取り扱わせるということは、コンピュータに意味の処理を実行させていることに相当することにはならないだろうか。ただし、コンピュータ上に表現されたメンバーシップ関数は、あくまでも形式的に表現された確定的なものである。ファジィ理論で提唱しているあいまいさは、メンバーシップ関数を自由に設定してよい、その形を客観的に決める根拠はないという意味でのあいまいであって、一旦メンバーシップ関数として表現してしまえば、形式的で確定的であることはもちろんである。上の考え方は、ファジィ集合を用いれば、コンピュータに、極めて単純な形ではあるが、ある意味で、意味を形式的に取り扱わせているということができよう。このように考えるとき、ファジィ理論がなぜ役に立つのかという理由とその面白さと特徴の一端が明らかになるように思われる。

6. ファジィ理論に基づくヒューマンコミュニケーション

人間とコンピュータとは言葉で対話をする。この時、言葉はファジィ集合の名称に対応している。コンピュータはその言葉に対応する意味、すなわちファジィ集合のメンバーシップ関数をメモリから取り出して処理をする。これが前述したコンピュータが意味を形式的に処理をしていることに相当する。そして、結論として、あるメンバーシップ関数を得た場合には、そのメンバーシップ関数に対応した最も近い言葉を選び出してコンピュータは人間に返答をする。この時、その言葉を使用している人間の個性に基づく言葉のあいまいさは、メモリに蓄えられているメンバーシップ関数の形をその人なりに変更してもらうことにより吸収させる。このように考えると、上の人間とコンピュータとのコミュニケーションの構図は人間と人間とのコミュニケーションの構図と同じである。人間と人間とは言葉で会話をしている。しかし、理解はその言葉の意味を用いて各人がその人なりの解釈で行っているからである。これはヒューマンインターフェースの一つの基本的なモデルになっているのではないだろうか。

ファジィ制御などで実用になっているファジィエキスパートシステムでは、まず人間が知識に基づいてあいまいな言葉でルールの形でシステムを定式化する。そのルールに現れるあいまいな言葉をファジィ集合と解釈して、メンバーシップ関数を定義する。コンピュータはそのルールの前件部と入力とがどの位いの度合で一致しているかをメンバーシップ関数を用いて計算する。次に、その一致の度合に応じてルールの後件部を修正して出力として用いている。これも、前述のモデルの考え方に適合していると言えよう。

ファジィ理論は、コンピュータに形式的ではあるが（もちろんコンピュータは本質的に形式的にしかできないが）、しかも現在では極めて単純な形でしかないが、意味を取り扱うための一つの手法を提供していると言えよう。

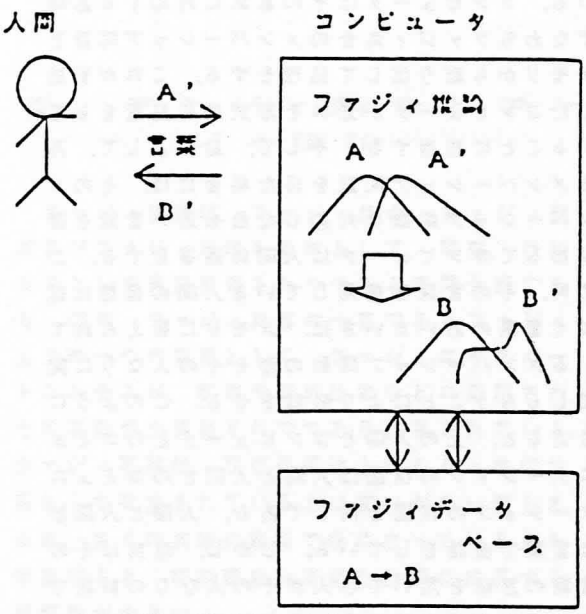


図3．ファジィ理論に基づくコンピュータとの対話

7．あとがき

ヒューマンコミュニケーションに関して、多くの考えるべき点があるが、人間の論理と機械の論理とは根本的に異なっているという認識は重要である。コンピュータの論理は記号レベルの論理であるのに対して、人間の論理は意味レベルの論理である。コンピュータのやっていることがよく分かるという人間側の立場に立った論理でマン・マシンインタフェースは構築されるべきである。高度情報化社会では今後、益々人間はコンピュータと付き合って行かざるを得ない。コンピュータは本来、人間の思考の代行の道具であり、うまく使って人間の能力を十分に発揮できる方向に使うべきである。このためには、人間と機械とが意味が分かるというレベルでのコミュニケーションが是非とも必要なのである。コンピュータは確かに記号だけの世界である。しかし、記号だけの世界にも、意味に近い記号表現と、まったく意味からは遠い記号表現とが存在する。ファジィ集合のメンバーシップ関数は非常に意味に近い記号表現と言えよう。本稿で紹介したように、ファジィ理論を用いると、コンピュータは形式的にではあるが意味レベルの処理をしているように考えられる。よって、従来のような記号レベルでの正確さを人間側に要求したりはしない。より柔軟な、人間中心、主観算重のマン・マシンインタフェースの一つのモデルになり得よう。

ファジィ理論は確率論と共に、ある種のあいまいさを取り扱うための数学的理論の一つである。そろそろ人間にとって本質的であるあいまいさについて、本気で取り扱うべき時期にきているのではないか。特に、機械と人間とが真に協調して生活していくためには、ヒューマンコミュニケーションにおいて、ファジィ理論の目指すところは極めて重要である。

参考文献：

（1）向殿：ファジィ理論が分かる本、ＨＢＪ出版局、1988

（2）中村（雄）他：ファジィー新しい知の展開、日刊工業、1989

第11回

ソフトウェア信頼性シンポジウム

発 表 者 募 集

主催：高信頼性ソフトウェア研究会
ソフトウェア技術者協会（ＳＥＡ）関西支部
電子情報通信学会ＦＴＳ研究専門委員会

このシンポジウムは，ソフトウェア開発の信頼性向上に関するさまざまなトピックスをめぐって，インフォーマルな（しかし，それだからこそ核心を突いた）議論を展開できるユニークな１つの場として，これまで５年間，毎年春と秋に開催されてきました．今年の春は，ソフトウェア・シンポジウム'90 の京都での開催と重なったために，ちょっとお休みをいただきましたが，再び秋から元のペースに戻ります．

ただし，会場をこれまでのホテルから大阪大学に移し，また，事前にそれぞれの発表の要旨を集めたプロシーディング（予稿集）も作って，少しシンポジウムとしての体裁を整えようと考えています．そこで，下記の要領で，論文または事例研究の発表希望者を募集します．ふるって御応募ください．

開催および発表募集要領

１．期日： 1990年 11月 1日（木）～ 2日（金）
２．会場： 大阪大学　基礎工学部　Σホール（大阪府豊中市）
３．定員： 100 名
４．費用（予定）： 会員（SEA または電子情報通信学会）20,000円　一般 30,000円
大学関係者 5,000円　学生 2,000円
５．運営委員会：
委員長：　鳥居宏次（大阪大学）
委　員：　天池学（カシオ）　飯沢恒（MCS）　臼井義美（JIP）　浦野義頼（KDD）　大場充（IBM）
岸田孝一（SRA）　松原友夫（日立 SK）　他
６．発表募集要領：

発表希望者は，裏面の応募用紙に必要事項を記入の上，1990年 8月末日（必着）までに，郵便または FAX で，運営委員長（阪大・鳥居）宛お送り下さい．

発表内容は，ソフトウェアの信頼性向上を目指した現実的努力の事例（成功例，失敗例いずれでも），信頼性の定量的把握のための試みや提案，いろいろな開発技法やツールの品質面での効用に関する経験的評価，プログラム・バグの分類や原因追跡など，何でも結構です．身近な経験にもとづく具体的な討論の材料を出してください．

なお，このシンポジウムはあくまでもインフォーマルな性質のものですから，形式の整った論文を書いていただく必要はありません．審査はアブストラクト（発表要旨）にもとづいて行い，予稿集にはそれを拡張した２～３ページのレジメを書いていただくだけで結構です．その意味で，どなたでもお気軽に御応募ください．

応募者多数の場合には，運営委員会で発表内容を検討した上，プログラム全体でのバランスを考慮して，採否を決めさせていただきます．審査結果の応募者への通知と，確定したプログラムにもとづく一般参加者の募集開始は，9月中旬を予定しています．

発表応募申込み先：

〒560　豊中市待兼山 1 - 1　大阪大学　基礎工学部　情報工学科
鳥　居　宏　次
TEL: 06 - 844 - 1151 (ex 4815)　FAX: 06 - 841 - 9741

第 11 回 ソフトウェア信頼性シンポジウム　発表応募用紙

氏　名: _____ (ふりがな) _____

年　齢: _____ (歳)　性　別: □ 男 □ 女　血液型: _____

種　別: □ SEA 会員　□ 電子情報通信学会会員　□ 一般　□ 学生

会社名: _____

部　門: _____ 役　職: _____

住　所: (〒　　) _____

TEL: (　) − (　) − (　　) 内線 (　　) FAX: (　) − (　) − (　　)

発表の題目:

_____

発表の要旨:

紙面が足りない場合は別紙を追加していただいてかまいません．また，関連する論文または技術資料などがありましたら，添付してください．

CALL FOR PARTICIPATION
International Workshop on UNIX-Based Software Development Environments

January 16-18, 1991
Hotel Grand Kempinski, Dallas, Texas, USA

Co-sponsored by:
USENIX Association (USA)
SIGMA Project (Japan)

Many software development environments have been described, built, or used which are intended to operate atop the UNIX system. The goal of this workshop is to share information on what these systems look like, what problems were solved by using UNIX and what problems were caused by it. We expect strong representation from the Japanese SIGMA workstation project, which defines a national software engineering environment that uses UNIX as its base, as well as American and European academic and industrial organization.

Participants will be selected by a program committee on the basis of submitted position papers. Attendance will be limited to 75 to encourage discussion. Meetings will include descriptions of important systems and presentations on particular technical points involving implementation and usage. Significant time will be set aside for panels and informal discussions of such topics.

Position papers will be evaluated by a program committee including researchers and practitioners from Europe, Japan, and the United States. Please send a 1-4 page position paper by **August 1, 1990** to one of the co-chairs:

Noboru Akima
Sigma Project
5th Akihabara Sanwa Bank Building
3-16-8 Soto-Kanda, Chiyoda-ku
Tokyo, Japan 101

Stuart Feldman
Bellcore
445 South Street
Morristown, NJ 07962-1910
USA

Electronic versions may be mailed to **sdeconf@bellcore.com**

Relevant topics that might be addressed in the position paper include:

UNIX is a registered trademark of AT&T

description of a significant system (by a designer or builder)

experience with using such a system
  novel tools or facilities offered by such a system
  evaluations of usage
    positive and negative experiences

experience with building such a system
  architectural considerations for a UNIX-based Software Development Environment
  advantages resulting from basing the system on UNIX
  problems (and solutions) encountered in designing and implementing such a system
    (e.g. file and database systems, networking and cooperation,
    scheduling and resource usage)

Workshop Format

The structure of the workshop will be decided after participants are selected. A likely agenda is:

Wed	Descriptions of SIGMA project
Wed	Descriptions of other large systems
Wed	Discussion of technical difficulties
Wed eve	Group reception and Birds of a Feather sessions
Thu morn	Panels on topics arising on Wednesday
Thu aft	Subgroups
Thu eve	Birds of a Feather
Fri morn	Subgroup reports
Fri aft	Panels, debate, wrapup

日本からは SEA および JUS を代表して，篠田陽一（東京工業大学）および松尾正敏（SRA）のお２人がプログラム委員会に入ることが予定されています．SEA は，御承知のように，創立以来毎年環境ワークショップを開催し，実践的な開発環境のあり方を考えて来ました．その意味で多くの会員の方々が積極的にポジション・ペーパを応募されることを期待します．

13th International Conference on SOFTWARE ENGINEERING

CALL FOR PAPERS

SYSTEM DESIGN: RESEARCH & PRACTICE

Austin, Texas May 13-16, 1991

CHAIR:
Laszlo A. Belady
ICSE13 Chair
MCC
P. O. Box 200195
Austin, Texas 78759 USA
belady@mcc.com
512/338-3504
FAX: 512/338-3899

PROGRAM CO-CHAIRS:
David Barstow
ICSE13 Program
Schlumberger Laboratory for Computer Science
P. O. Box 200015
Austin, Texas 78720-0015 USA
barstow@slcs.slb.com
512/331-3728
FAX: 512/331-3760

Koji Torii
ICSE13 Program
Dept. of Information & Computer Sciences
Osaka University
Machikaneyama 1-1
Toyonaka City
Osaka, JAPAN
torii%tori2.ics.osakau.ac.jp@relay.cs.net

TOOLS FAIR CHAIR:
Laurie or John Werth
Department of Computer Science
University of Texas at Austin
Austin, Texas 78712 USA
lwerth@cs.utexas.edu
jwerth@cs.utexas.edu
512/471-7316
FAX: 512/471-8885

TUTORIAL CHAIR:
Herb Krasner
Lockheed
Software Technology Center
O9610/B30E
2100 East St. Elmo
Austin, Texas 78744 USA
krasner@stc.lockheed.com
512/448-5738
FAX: 512/448-5728

LOCAL ARRANGEMENTS CHAIR:
Bryan Fugate
ICSE13 Local Arrangements
MCC
P.O. Box 200195
Austin, Texas 78759 USA
fugate@mcc.com
512/338-3330
FAX: 512/338-3899

ell into its third decade, the engineering of software is becoming the engineering of computerized applications. Increasingly, new systems arise through the adaptation and integration of existing applications, with software as the glue that holds the pieces together (in much the same way that CAD and CAM have been integrated as Computer Integrated Engineering). Creating these systems requires the expertise and cooperation of a wide array of specialists — specialists in the application domain, in real time performance, and in reliability. The software engineer becomes a system designer who must understand the application domain in order to select not only the system's software but also its hardware. Perhaps most importantly, the system designer must be able to work well in teams with others.

These complexities are giving rise to new, often interdisciplinary software engineering research subjects. Already, a wealth of experience and insight has emerged from research in such disciplines as computer supported cooperative work, distributed systems design, reverse engineering, integration technology, and computer aided education and training. And the more established research subjects — quantitative methods, languages and representations, project management, and others — are expanding in breadth and excitement. From this work, a new generation of tools and techniques for improving the precision, quality, and predictability of system building projects is arising. Without superb technology and continued research, companies and nations will lack the productivity to be competitive.

ICSE invites researchers and practitioners to share their recent ideas and experiences, particularly those which aim to improve the design of complex computer applications. The program committee will review all submissions for quality and for relevance to future work. Papers, panel proposals, and reports must be received in writing by September 14, 1990.

SUBMISSIONS: Eight (8) copies in English of papers, panel proposals, or experience reports should be submitted to the address below by September 14, 1990:

David Barstow
Schlumberger Laboratory for Computer Science
P. O. Box 200015
Austin, Texas 78720-0015

Papers should be limited to 6000 words, full-page figures being counted as 300 words. Each paper must include a short abstract, a list of keywords, and the lead author's address.

Panel proposals should include title, proposed chair, tentative panelists (including a short vita), a two or three paragraph description of the subject, format of the presentation, and rationale for the panel.

Experience reports should describe practical experience in some aspect of software engineering (using a tool or method, applying a metric, following a project management discipline, reusing work products, etc.). The contributor(s) should submit a 5 to 10-page written description of the experience and a one-page outline for a five-minute presentation.

TOOLS FAIR: A software tools fair will be held during the conference, so that attendees can see and learn about current experimental and commercial software tools. Those who want to exhibit or demonstrate a tool, and especially those interested in presenting a descriptive paper, should contact:

Laurie or John Werth
Department of Computer Science
University of Texas at Austin
Austin, Texas 78712

FURTHER INFORMATION: For further information and/or copy of the advance program when available, write either to ICSE13, MCC, P. O. Box 200195, Austin, Texas 78720 USA, or to ICSE13, IEEE Computer Society, 1730 Massachusetts Ave., NW, Washington, DC, 20036 USA.

IMPORTANT DATES: Submission deadline: 14 September 1990; Acceptance notification: 14 December 1990; Final versions due: 8 February 1991.

Sponsored by ACM Special Interest Group on Software Engineering
IEEE Computer Society Technical Committee on Software Engineering

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.
IEEE

ソフトウェア技術者協会
〒102 東京都千代田区隼町2-12 藤和半蔵門コーフビル505
TEL.03-234-9455　FAX.03-234-9454