SEAMAIL

Newsletter from Software Engineers Association

Volume 6, Number 2 1991

目　次

事務局から		1
1991 年度総会報告		2
新幹事名簿		4
SEA 今年の主なイベント		5
最近の幹事会での報告と討論		6
SEA はどこから来たのか？ SEA とは何者か？		
そして SEA はどこへ行くのか？	熊谷　章	13
熊谷さんの問いかけに答えて		
	筏井美枝子・臼井義美	16
	小林俊明・中谷多哉子・中山照章	17
	野村敏次・野村行憲	18
SEA への私のこだわり	渡邉雄一	
オブジェクト指向 CASE の動向	佐原伸・中谷多哉子・藤野晃延	21
Call for Papers/Participations		
SEA 秋の集中セミナー		40
1st International Conference on the Software Process		42
14th International Conference on Software Engineering		44
Nexus (NeXT ユーザ会の紹介)		45

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

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，1000余名を越えるメンバーを擁するにいたりました．法人賛助会員も約80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： 市川寛　落水浩一郎　岸田孝一　中野秀男　深瀬弘恭　松原友夫　山崎朝昭

幹事： 飯沢恒　筏井美枝子　上原憲二　臼井義美　奥山充　片山卓也　亀田繁　窪田芳夫　小林俊明
杉田義明　田中一夫　玉井哲雄　鳥居宏次　中谷多哉子　中山照章　野村敏次　野村行憲　浜野善吉
林香　平尾一浩　藤野晃延　二木厚吉　堀江進　本位田真一　盛田政敏　山崎利治　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)：野々下幸治
教育分科会(SIGEDU)：杉田義明　中園順三
ネットワーク分科会(SIGNET)：小林俊明
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：藤野晃延　北條政顕　野中哲　松下和隆
長野支部：市川寛　小林貞幸　佐藤千明　細野広水
名古屋支部：筏井美枝子　岩田康　鈴木智　平田淳史
九州支部：藤本良子　平尾一浩

賛助会員会社： ＣＳＫ教育事業部　ＮＴＴソフトウェア研究所　ＮＴＴ九州ネットワーク技術センタ　ＮＴＴ九州技術開発センタ
ＰＦＵ　ＳＲＡ　ＴＤＫ　アイ・エヌ・エス　アスキー　アトラス情報サービス　ウチダエスコ　エイ・エス・ティ
エスケーディ　エヌ・ティ・ティ・システム技術　オムロンソフトウェア　カシオ計算機　キャノン新川崎事業所
クレオ　ケーシーエス　コグノスジャパン　サン・ビルド印刷　シーアイシステムデザイン　システムラボムラタ
ジェーエムエーシステムズ　ジャステック　ジャストシステム　スインク　セイホーソフトウェア
セントラル・コンピュータ・サービス　ソフトウェアコントロール　ダイキン工業　テクノバ　テスク
テックシステムズ　ニコンシステム　ニッセイコンピュータ　ヒラタ・ソフトウェア・テクノロジー
ビーアイテイ　ファコム・ハイタック　フクダ電子　ムラタシステム　リコー　リコーシステム開発
リバテイーシステム　安川電気製作所　近畿日本ツーリスト　古河インフォメーション・テクノロジー
構造計画研究所　三菱電機カスタムＬＳＩ設計技術開発センター　三菱電機コントロールソフトウェア
三菱電機セミコンダクタソフトウェア　三菱電機メカトロニクスソフトウェア　三菱電機関西コンピュータシステム
三菱電機東部コンピュータシステム　三菱電機北伊丹製作所　松下ソフトリサーチ　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　千代田製作所　全日空システム企画　池上通信機　中央システム
辻システム計画事務所　東電ソフトウェア　東北ＳＲＡ　日本ＮＣＤ　日本エム・アイ・シー　日本システム
日本データスキル日本ユニシス　日本ユニシス・ソフトウェア　日本情報システムサービス　日本電気ソフトウェア
日立エンジニアリング　日立ビジネス機器　富士ゼロックス情報システム　富士写真フィルム　富士総研情報システム
富士通　富士通エフ・アイ・ピー　富士通ビー・エス・シー　明電舎（以上82社）

ＳＥＡＭＡＩＬ　Ｖｏｌ．６，Ｎｏ．２　平成３年８月２５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
TEL: 03-3356-1077　FAX: 03-3356-1072
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

☆

今年は，あちこちで火山が爆発したり，大洪水があったり，夏があまり夏らしくなかったり，ヤクルトが優勝争いに加わったりといった異常現象が続いたせいか，わが Seamail の「船便シンドローム」もきわめて深刻な状況に突入し，もう秋も近いというのに，これでようやく第２号にしかなりません．

☆☆

何人かの方から，お問い合わせやら催促やらお叱りをいただきましたが，編集部がもう長いこと壊滅状態にあるので，事務局としては，ただやきもきするだけで，手の打ちようがありません．まるで，野村監督のようにボヤキが止まらない心境です．

☆☆☆

せめてもの埋め草にと，先日ソフトウェア・シンポジウム'91 のプロシーディングスを全会員に遅らせていただきました．名古屋に行かれた方には２冊目になってしまいますが，SEA の宣伝その他の目的に御利用ください．

☆☆☆☆

さて，この号には，5 月に行われた総会の報告，新幹事の方たちの名簿に加えて，ここ何ヶ月の間，幹事会で議論されたこと（主として SEA のこれからの運営をどうして行くか？）を要約してのせました．また，その中で代表幹事の熊谷さんから，「そもそも SEA とは？」という浮世根問い的な問題提起がなされ，それについて何人かの幹事の方々からコメントをいただいたので，併せて掲載しました．

☆☆☆☆☆

そして，また，佐原・中谷・藤野の３人の方から，オブジェクト指向に関する力作のサーベイ・ペーパを寄稿していただきました．（こういう原稿が毎月どなたかから届けられれば，編集部なんかいなくても，定期的に Seamail が出るのですが.... みなさん，よろしくお願いいたします．）

☆☆☆☆☆☆

さあ，来月から，セミナー・ウィークを皮切りに，イベントの秋が始まります．また，ストレスに追いまくられて，やせるチャンスを逃がしそう？！（それはそうと，ヤクルトがんばって！）

☆☆☆☆☆☆☆

1991 年度総会報告

今年度 SEA 年次総会は，さる 5 月 30 日，5 月フォーラムの後，機械振興会館で開催された．出席者は 29 名，他に委任状提出者 160 名で，合計 189 名．定足数 98 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）が代表幹事の熊谷章さんから報告され，満場一致で承認された．

表１． 1990 年度　決算報告書（自 1990 年 4 月 1 日　至 1991 年 3 月 31 日）

貸　借　対　照　表（1991 年 3 月 31 日現在）

資産の部		負債及び正味財産の部	
流動資産		流動負債	
現預金	8,130,856	未払金	0
未収入金	599,000	正味財産	8,729,856
		（当期収支差額	721,110）
合　計	8,729,856	合　計	8,729,856

付　属　明　細　書

項　目	金　額	備　考
現預金		
普通預金	8,130,856	三菱銀行本店公務部
未収入金		
新入会費	40,000	4 名
更新会費	259,000	37 名
賛助会費	300,000	6 口

表１（続き）
収　支　計　算　書（自 1990 年 4 月 1 日　至　1991 年 3 月 31 日）

支出の部		収入の部	
人件費	536,540	新入会費	1,430,000
事務所費	5,872,231	更新会費	5,438,940
印刷費	4,928,910	賛助会費	4,450,000
通信費	2,907,354	イベント収入	5,013,258
会議費	3,504,346	雑収入	3,257,008
広報費	410,000		
消耗品費	167,373		
雑　費	541,342		
当期収支差額	721,110		
合　計	19,589,206	合　計	19,589,206

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）．

引き続き，前年度常任幹事会において検討された，新年度の役員候補者リスト（４ページ）が提示され，原案通り承認された．その後，代表幹事の熊谷章さんにより今年度予算案（表２）が提示され，これも異議なく可決された．また，今年度から賛助会費をこれまでの１口５万円から１０万円に値上げすること（細則の改訂）が提案され，やはり満場一致で可決された．

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２．1991 年度 予　算　書（自 1991 年 4 月 1 日　至 1992 年 3 月 31 日）

支出の部		収入の部	
人件費	3,900,000	新入会費	2,000,000
事務所費	4,200,000	更新会費	5,472,000
印刷費	6,000,000	賛助会費	7,000,000
通信費	3,600,000	イベント収入	4,000,000
会議費	360,000	雑収入	2,000,000
支部活動支援費	240,000		
消耗品費	360,000		
国際活動費	1,500,000		
雑　費	60,000		
当期収支差額	252,000		
合　計	20,472,000	合　計	20,472,000

	1991 年度	SEA 幹事会メンバー
代表幹事	熊谷　章	PFU & 富士通
常任幹事	市川　寛	電算
	落水　浩一郎	静岡大学
	岸田　孝一	SRA
	中野　秀男	大阪大学
	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
	山崎　朝昭	新日鉄情報通信システム
幹事	飯沢　恒	三菱電機東部コンピュータシステム
	筏井　美枝子	SRA
	上原　憲二	三菱電機
	臼井　義美	日本電子計算
	奥山　充	日立ビジネス機器
	片山　卓也	東京工業大学
	亀田　繁	ファコムハイタック
	窪田　芳夫	東京電力
	小林　俊明	ソフトウェア技術
	杉田　義明	日本 NCD
	田中　一夫	山一コンピュータセンター
	玉井　哲雄	筑波大学
	鳥居　宏次	大阪大学
	中谷　多哉子	富士ゼロックス情報システム
	中山　照章	富士通ソーシアルサイエンスラボラトリ
	野村　敏次	日本電子計算
	野村　行憲	岩手電子計算センター
	浜野　善吉	クレオ
	林　香	SRA
	平尾　一浩	ヒラタソフトウェアテクノロジー
	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
	堀江　進	日本電気ソフトウェア
	本位田　真一	東芝
	盛田　政敏	ケーシーエス
	山崎　利治	日本ユニシス
	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 1991 年の主要イベント

（実績と予定：8/14 現在）

1/7	SEA Forum「新春放談会」	東京：事務局会議室（参加者 26 名）
1/30-2/1	第 3 回 ソフトウェア プロセス ワークショップ	伊東：五景館（参加者 24 名）
2/5	ソフトウェア信頼性フォーラム	仙台：宮城第一ホテル（参加者 100 名）
2/8	SEA Forum「これからの中堅技術者教育を考える」	東京：機械振興会館（参加者 41 名）
3/14-15	春のセミナー ウィーク'91	東京：青年会議所会館（参加者延べ 228 名）
3/25-27	第 3 回 テクニカル マネジメント ワークショップ	直江津：ホテル センチュリー イカヤ（参加者 15 名）
4/12	SEA Forum「テクニカル・マネジメント」	東京：機械振興会館（参加者 30 名）
5/9-18	ICSE-13 研修ツアー	Austin, TX, USA
5/30	SEA Forum「ソフトウェア開発のノンテクニカルな側面」	東京：機械振興会館（参加者 30 名）
5/30	SEA 1991 年度総会	東京：機械振興会館
6/10	ソフトウェア シンポジウム '91 併設チュートリアル	名古屋：名古屋国際会議場（参加者延べ 168 名）
6/11-12	ソフトウェア シンポジウム '91	名古屋：名古屋国際会議場（参加者 362 名）
6/13-14	3rd International Symposium on Future Software Environment	彦根：彦根プリンスホテル（参加者 41 名）
6/24	SEA Forum「オブジェクト指向の実践」	東京：機械振興会館（参加者 65 名）
6/28	SEA 九州支部設立 3 周年記念 Forum	熊本（参加者 70 名）
7/25	SEA Forum「人間の知的活動支援システムを考える」	東京：機械振興会館（参加者 40 名）
8/29	特別 Forum in 仙台：「ソフトウェア開発の課題」	宮城（参加者募集中）
9/4-7	第 9 回 夏のプログラミング ワークショップ（若手の会）	盛岡（参加申込 40 名）
9/20	SEA Forum in 北海道	札幌（企画中）
9/24-26	秋のセミナー ウィーク'91	東京（参加者募集中）
10/4	SS'91 Follow-up Forum	名古屋（企画中）
10/21-22	1st International Conference on the Software Process	米国：Los Angels（参加者募集中）
10/24-26	第 5 回 教育ワークショップ	北陸（参加者募集中）
10/28-30	Beijing Internatinal CASE Symposium '91	中国：北京大学（参加者募集中）
10/?	SEA Forum（テーマ未定）	東京
11/7-8	第 12 回 ソフトウェア信頼性シンポジウム	仙台国際センター（発表者募集中）
11/?	SEA Forum（テーマ未定）	東京
11/?	SEA 特別 Forum in 北陸	金沢（企画中）
12/?	SEA Forum（テーマ未定）	東京

なお，来年のソフトウェア・シンポジウム'92 は，6 月 9-11 日，長野市での開催が予定されている（近く発表論文募集開始の予定）．

最近の幹事会での報告と討論

1. 新年度の財務状況

	予算	4月	5月	6月	7月
収入					
新入会費	166,667	70,000	60,000	70,000	110,000
更新会費	456,000	861,000	560,000	560,000	455,000
賛助会費	583,333	450,000	750,000	250,000	500,000
イベント収入	333,333	114,446	0	0	1,000,000
雑収入	166,667	0	21,000	0	602,268
合計	1,706,000	1,495,446	1,391,000	880,000	2,667,268
支出					
人件費	325,000	6,320	0	211,558	341,445
事務所費	350,000	365,676	272,576	274,842	491,966
印刷費	500,000	30,000	0	421,785	0
通信費	300,000	44,265	395,709	261,301	490,312
会議費	30,000	18,873	93,676	21,152	30,272
支部支援費	20,000	0	0	0	60,000
消耗品費	30,000	5,562	1,981	8,978	6,280
雑費	5,000	2,506	912	1,957	2,884
合計	1,685,000	473,202	848,834	1,201,573	1,423,159
収支	21,000	1,022,244	542,166	− 321,573	1,244,109

一応，予算と対比するとやや縮小状態の健全財政で推移しているが，このままではいけない．Seamail をきちんと出せば，それだけ印刷費・通信費がふえるし，発送の労力もかかる．やはり，前年度の会計監査で吉村先生に指摘されたように，会員数を現在の実質 1000 人の線から，２～３倍に増やして，イベント収入なしでも財政が安定するようにしなければならない．それが，今年度の幹事会に課せられた１つの課題であろう．

2. 各種イベントの報告

(1) SEA Forum

４月以降の月例 Forum（会場はいずれも機械振興会館）はすべて活況であった．内容的にもなかなかおもしろいという参加者の評判であった．問題は参加できない人々（特に地方の会員）のためにレポートをなんとかしなければならないということである．いずれ編集委員会で何とか手立てを考えてもらう必要がある．

昨年好評だった全国縦断 Forum は，今年も続けることになった．やり方は，これまで東京で毎月開いていた幹事会を隔月サイクル（偶数月）にし，その間の月（すなわち奇数月）に常任幹事会を地方で開催し，併せて，常任幹事を中心に forum を企画するというものである．

とりあえずは，９月（20 日）に北海道・札幌，11 月に北陸・金沢の企画が進行中．その後（来年）は，

1月に九州（福岡），3月に信州（松本），5月に東北（仙台），7月に四国（松山）の予定．

この他に各支部独自企画の Forum もいくつか．まずは，九州支部が設立3周年記念のイベントを 6 月末に実施．名古屋では，10 月上旬にソフト・シンポのフォローアップを企画中．長野では，92 年のシンポの PC ミーティングに連動して，何かイベントを考えようとしている．

(2) Software symposium '91

あらかじめ予想された会場の問題点（食事の場所がない）を除けば，すべて順調に終わった．参加者の地域別一覧は下表の通り：

地域	シンポジウム & チュトリアル		チュトリアルのみ		合計
	有料	無料	有料	無料	
北海道・東北	0	0	0	0	0
関東	127	44	2	1	174
東海	61	6	3	21	91
信越・北陸	6	1	0	0	7
関西	68	2	2	0	72
中国・四国	2	2	0	0	4
九州	9	1	0	0	10
その他	3	1	0	0	4
合計	276	57	7	22	362

シンポジウム有料申込者 276 名の種別内訳は以下の通り：

種別	人数	比率 (%)
SEA 正会員	79	28.6
SEA 賛助会員	68	24.6
協賛団体会員	55	19.9
一般	60	21.7
学校関係者	8	2.9
学生	7	2.5

シンポジウム有料申込者 276 名のうちチュトリアルも申し込んだ人は 59 名（約 19%）．
したがってチュトリアル受講者は 59+7+22=88 名（延べ 168 セッション）．

また，恒例の服部賞（旧ソフトウェア産業振興協会の故・服部正会長を記念した最優秀論文賞）は，越前孝（神戸日本電気ソフトウェア）および中小路久美代（SRA）の両氏が受賞した．

名古屋支部の活動は最近やや低迷しているが，今回のシンポジウム開催を機に，jus 東海支部と協力して，活性化の方向を探るべく検討中．とりあえず 10 月に特別 Forum を開く．

来年の Software symposium '92 は長野県長野市で 6 月 9 ～ 11 日に開催される予定．実行委員長：市川寛（電算），プログラム委員長：岸田孝一（SRA）・中野秀男（大阪大学），チュートリアル委員長：野村敏次（JIP）の各氏．いま PC メンバを選定中で，近く Call for Paper が出る予定．

(3) 3rd Int'l Symposium on Future Software Environment

Software Symposium に引き続いて，6 月 13 ～ 14 日に，彦根プリンス・ホテルで開催された．主催は SDA コンソーシアム，SEA は協賛．参加者 44 名．国際協同研究プロジェクト SDA のメン

バーに加えて，アメリカの著名なプロセス・プログラミング・プロジェクト Arcadia 主要メンバーも参加し，次のプログラムで，活発に討論が行われた．記録はすべて 8mm ビデオに収録してあり，いずれ Seamail 誌上で報告される予定．

6/13	AM	"Approaches to FSE"	
		Chair	William Riddle (Software Design & Analysis)
		Speakers	Leon Osterweil (UC Irvine)
			Hideyuki Nakashima (ETL)
			Gerhard Fischer (Univ of Colorado)
	PM	"Human Aspects in FSE"	
		Chair	John Sayler (Software Analytics)
		Speakers	Jonathan Grudin (Aarhus Univ)
			Dennis Heimbigner (Univ of Colorado)
			Tetsuo Tamai (Univ of Tsukuba)
6/14	AM	"Conceptual Issues on FSE"	
		Chair	Kokichi Futatsugi (ETL)
		Speakers	Takuya Katayama (TITECH)
			Akira Kumagai (PFU & Fujitsu)
			Brian Nejmeh (INSTEP)
	PM	"Supporting Technologies for FSE"	
		Chair	Lloyd Williams (Software Engineering Research)
		Speakers	David Notkin (Washington Univ)
			Rick Selby (UC Irvine)
			Takahira Yamaguchi (Shizuoka Univ)

3. 秋以降のイベントの準備状況

(1) 第 9 回　若手の会（9/3 ～ 6 in 盛岡）

定員をやや上回る参加申込みがあり，7 月 26 日（Y-Doc 7 月例会）の前に横浜で PC を開催し，開催要領の詳細を決めた．現在，PC メンバーが SEA の Unix マシン上のメイリング・リストを介して，今回のテーマである「ソフトウェア工学のテクノロジー・マップ」をどう扱うか？　それぞれのイメージするマップはいかなるものか？　等などについてネットワーク上で討論を続行中．

(2) SEA Forum in 北海道（9/20 in 札幌）

現在、企画の最終段階．8/22 の幹事会で詳細を決定して，案内を開始する．さて，これをきっかけに北海道支部立ち上げにもって行けるか？

(3) 秋のセミナー・ウィーク（9/24 ～ 26 in 東京）

7 月末にプログラムが決まり，現在参加者募集中．

(4) 1st International Conference on the Software Process

最近ソフトウェア工学コミュニティのなかでのホット・トピックスである「プロセス」に関する初めての国際会議．現在参加者募集中．PC に 3 人の会員が名を連ねており，SEA は協賛．

(5) 第 5 回　教育ワークショップ（10/24 ～ 26 in 北陸）

教育分科会の年間恒例行事．現在参加者募集中．

(6) 北京国際 CASE シンポジウム（10/28 ～ 39 in 北京）

7 月上旬に熊谷・岸田・玉井・山崎の 4 氏が北京大学での PC ミーティングに参加．詳細が決まり，

現在一般参加者募集中

(7) 第 12 回　ソフトウェア信頼性シンポジウム（11/7 ～ 8 in 仙台）

これまでずっと大阪で開催してきたが，今回は東北の杜の都・仙台に場所を移し，この夏にできる新しい国際会議場（仙台国際センター：９月オープン）を利用することになった．現在発表者募集中（締切りは 8 月末）．9 月中旬に仙台で PC を開き，詳細を決めて一般参加者の募集を行なう予定．

4. ソフトウェア・プロセス実態調査

かねてから，さまざまな国際会議などでの活動で SEA と関係の深かった W.E.Riddle 氏が主宰するアメリカの Non-Profit Organization: rMISE (Rocky Mountain Institute of software Engineering) が，ソフトウェア・プロセス（すなわちソフトウェア開発および保守工程）の実態に関する大掛かりなアンケート調査を欧米で実施する計画があり（５月に調査票を配布して，現在結果を回収中），SEA でもそれに呼応して同じ調査を日本で行い，比較・分析をしようという提案が，野村（敏）および岸田両幹事からあり，調査分科会（SIG-Survey）が 7 月中旬に発足した．

この SIG は，他の分科会と同様に，独立採算で活動するが，当初アンケート票の印刷・発送・集計などに，かなりの経費がかかるので，その費用は SEA 本体から借り入れる形でスタートし，調査結果の報告セミナーやレポートの販売で収益を上げて返済する．皮算用では，差引いくらかの収益を上げ，今後継続的に SEA として実施すべき各種の技術実態調査を引き続いて計画したいという意向である．

アンケートの日本語への翻訳は，分科会メンバーが手分けして 8 月上旬に終わり，現在印刷中．8 月末には，会員・非会員を含めて約 3000 通を発送する．回収期限は 9 月中旬．とりあえずの集計・分析結果は 10 月半ばにはまとまる予定．結果の概要は Seamail 誌上で報告される．

また，SEA のワークショップの中で最も古い歴史を持つ環境ワークショップが，現在「なかだるみ」の状態で，次回（第 6 回）の企画が中断しているので，この調査データにもとづき，もう一度原点に立ちかえって，「実践的ソフトウェア開発環境」のあるべき姿を討論したらどうか？　といったことも考えられている．

5. さまざまな検討・討論事項

(1) 議論の始まり

新幹事会メンバーが初めて顔を合わせた 6 月 25 日の幹事会で，これまで有名無実であった常任委員会の役割，幹事の責任分担，Seamail の立て直し，分科会の活性化，支部活動のあり方などについての自由な意見交換が行われた．時間切れで満足な結論にはいたらなかったため，本来隔月を予定していた幹事会を翌 7 月にも開くことになった．

(2) 常任幹事会

7 月 12 日，討論の叩き台を作るべく，常任幹事会が開かれ，次のようなことがらについて話し合った：

- SEA はどこから来たのか？ SEA とは何者か？そして SEA はどこへ行くのか？
  すなわち，既存の学会，業界団体，その他と SEA はどこが異なるのか？
- Seamail の定期発行体制をどうやって確立するか？
  たとえば，複数編集長制の導入．特集号スタイル etc.
- 年間行事の見直し：
  統廃合，新設（その基準），内容の変更，etc.

- 各委員会の役割:
  とにかく実質的に委員会をどう動かすか？
- 財政基盤の確立:
  ベースとしての個人会員をどう増やすか？
- 事務局問題:
  人手不足の解消，将来の事務所移転，設備の増強，etc.

その結果（結論というわけではない）として出てきたことは以下の通り:

(a) 現状における問題点

次にあげるようなことがらが，アトランダムに話題になった．

- SEA の活動（たとえば，フォーラムやセミナーやワークショップなどの話題）が，会員であるソフトウェア技術者の日常生活（御承知の通り泥臭い毎日）から少し遊離してはいないか？
- 地方支部の場合，会合を持つにしても，交通・トピックス・スピーカ探しなどの点で困難が多い．
- 一昨年に始めて参加者からは比較的好評だった技術交流会が中断しているのはなぜか？（SEA サイドの問題か？　それとも日本のアカデミック・コミュニティの性格が成せるわざか？）
- 一般に SEA の存在およびその活動の意義がまだ，社会的に十分認知されているとはいえない．したがって，会員が活動に参加しようとした場合，所属組織（会社・大学）からそのことがオーソライズされないことがある．
- 東京の分科会が一時はたくさんできたのに，その多くが休眠状態に陥ってしまったのはなぜか？（関西はよくやっているが，そのあたりの秘密は？）
- Seamail の定期刊行を何とかしよう．これだけイベントをやっているのだから，そのレポート体制を確立すればよいのでは？
- それにしても，アクティブな会員が足りないような気がする（単に頭数ではなく活動エネルギーの問題）．一般会員は SEA を従来の日本的な組織と同じように考えて，受け身でしか行動していないように見える．これを何とかしないと....

(b) 解決の方向づけ

上記のような問題点を解決するために，とりあえずのアクション・アイテムをいくつか列挙し，その具体化を次回以降の幹事会で議論してもらうことにした．

- 各種活動のプロセスを明示的に（たとえばマニュアル化）し，会員のだれもが SEA を利用して，自分のやりたい活動をキックオフできるようにする．
- 入会パンフレットその他の資料を整備し，SEA の存在をもっと一般に Visible にする．それによって社会的認知の度合いを高める．
- Seamail の編集は，複数（４～６人）の編集責任者を決め，年に１人何冊かの割り当て（スケジュールも）をする．とりあえず，岸田・熊谷・中野の３常任幹事が立候補．また，各支部や分科会のイベント，それに月例フォーラムやセミナーについては，きちんとレポータを決めて，Seamail の原稿を書いてもらう．
- 年中行事として決まっている各種のイベントは，それぞれに Steering Commitee を作って，長期的な計画をあらかじめ立てるようにしたい．

- 常任委員会は，担当の幹事および責任者を決め，それぞれのアクション･アイテムをきちんとこなして行けるようにしたい．
- 事務局の強化： とりあえず７月からパートタイマーの女性を１人採用する．会員管理，郵便発送など，事務処理の量がしだいに増えて来ているので，各種のイベントに必要な事務処理はなるべくその実行委員会の中で閉じた形で行い，事務局の負担を軽減する．また，現在は岸田前代表幹事の好意で，事務局人件費およびオフィス･スペースの提供など，任意団体で法人格のない SEA の欠陥を補ってもらっているが，長期的には何とかしないといけない．たとえば，賛助会員会社が持ち回りで負担するとか，法人化を図るとか．

(3) 臨時の幹事会

7月 23 日に事務局会議室で開かれ，常任幹事会からの叩き台をもとに，いろいろなことが話し合われた．主なことがらをアトランダムに列挙すると：

- Seamail に Problem Column のようなものを設けたらどうか？ みんながいま困っている問題をのせる．解決策まではいらない．議論のきっかけになればよい．たとえば，最近あちこちで，ネットワークのマネジメント（分散環境の管理をどうするか？）や，複雑化したソフトウェア･プロダクトやドキュメントのコンフィギュレーション管理についての，現実的な悩みを聞くことが多い．
- Columnmm を設けるのは簡単だが，過去の経験からすれば，原稿が集まるかしらという心配がある．アンケートを取ってみることはできる．
- SEA の活動がわれわれの日常とかけはなれているというのは，別に問題ではなく，それこそが SEA の存在意義なのではないか？
- SEA の利用価値の１つは，技術移転メカニズムとしてのそれだと思う．その場合，２つの側面がある．１つは，日常とはかけはなれた研究開発最前線の飛んだ話題にふれること，もう１つは，すぐ現場で役立つ技術の講習会（たとえば，X-window の使い方，C++ 入門）である．後者の性格を持つ活動がいままで，どちらかといえば欠けていたのでは？
- そうしたチュートリアルはたしかに可能だが，問題は，街のセミナー屋さんとの違いをどこでつけるかだろう．セミナー委員会で考えてもらおう．
- プログラミングという仕事は見た目よりずっと難しいので，そうしたチュートリアルは一見訳に立ちそうで，実はまったく無益であるように思う．SEA は真にプロフェッショナルなプログラマ集団（一種のギルドみたいなもの）を目指して，エトスをもっと養うようにすべきだ．その意味では，現在の会員数は少し多過ぎるともいえよう．
- たとえば，オブジェクト指向という話題が，フォーラムやセミナーあるいはシンポジウムで人気を集めているようだが，教育研修予算を握っている会社のアネージャたちには，まったく受けていない．そんな話を聞いて，いまやっているプロジェクトのどこで，どう具体的に役立つの？ という訳だ．
- 社会的認知を高めるという意味では，賛助会員会社向けの企画を考えてみたらどうか？ たとえば，採用問題，転職･退職問題，技術者教育など．いわば，，ソフトウェア産業の現状における問題点を論ずるというフォーラムまたはワークショップである．
- 日常の問題点は，ノンテクニカルな領域に属するものがほとんどである．たとえば，プロジェクト管理，標準化，定量的見積／分析，生産性など．SEA 発足以前の旧ソフト協技術委でのワークショップやセミナーは，こうしたテーマで人を集めていあっと思う．管理者の受けはよく，参加者は集めやすい．

－ その種のセミナーは街のセミナー屋さんがよくやっているが，羊頭狗肉で中身のないものが多い．ソフトウェア開発上のこうした問題は，病気のたとえれば水虫や風邪みたいなもの．絶対に完治しないし，なくならない．だから，薬が（効いても効かなくても）よく売れる．

といった形で，午後から夜にかけて延々と議論が続いた．具体的なアクション･アイテムに関する詳細は，さらに来月の幹事会で議論した上で，それぞれ担当の常任委員会にまかせることになった．

各常任委員会への幹事の割り振りの第１次案は次の通り（ただし，まだ幹事全員がアサインされていないし，当日欠席した人の了解は取れていない）:

総務（法人化，事務局，広報，予算など）:
熊谷，山崎（朝），市川，窪田，野村（敏），盛田，林．

技術研究（支部，分科会，ワークショップなど）:
中野，落水，深瀬，筏井，小林．

編集（とにかく Seamail）:
岸田，中野，熊谷，山崎（利），渡辺，中山．

セミナー（チュートリアルをどうするか，も含めて）:
杉田，亀田，浜野，堀江，田中，中谷，（人見），（横山）．

国際交流（新設）:
松原，岸田，玉井．

この問題に関する議論は，さらに 8 月以降の幹事会･常任幹事会でも続けられる．なお，代表幹事の熊谷さんから提起された「SEA はどこから来たのか？ SEA とは何者か？ SEA はどこへ行くのか？」については，熊谷さん自身に，問いかけの趣旨をパラフレーズした文章をお寄せいただいた．何人かの幹事が E-mail で寄せられたコメントとあわせて次ページ以降にのせてある．会員諸兄姉もぜひ，これらをお読みになった上で，コメントをお寄せいただきたい．

SEA はどこから来たのか？ SEA とは何者か？ そして SEA はどこへ行くのか？

熊 谷 章
（PFU & 富士通）

1. 序章

社会的な階層としてのソフトウェア技術者という集団が，いまの日本に存在するのだろうか？ ソフトウェア技術者とは，いったいどういう技術者なのだろうか？ 私の心のどこかに，いつも，漠としたこのような疑問がある．

ソフトウェア技術者は，コンピュータというハードウェアが発明された後，その派生物として世の中に登場してきた．われわれの心の内側には，われわれの心が狂気に向かわないようにするために，その働きを規制する「常識」なるものがたしかに存在している．それは，知らず識らずのうちにハードウェアがソフトウェアを規制しているのとよく似ている．人間は，いつもさまざまな物理的な束縛から逃れられずに思考し生活している．だから，時として，夢想的なアイデアに魅惑される．超現実的なイメージを，芸術と称して心から溺愛したりする．そこに，日常の身体や心を縛っている枠組みから開放された完全な自由を見出せるからである．

われわれが自由をなぜ望むか，については決定的な答えがある．人間は，本来自由に生まれてきたからである．このことについては，先日名古屋のソフトウェア・シンポジウムでお会いした斎藤信男氏(慶応大学)の次のことばが記憶に新しい：「東西ドイツの統合の TV ニュースで最も感動したのは，東ドイツの人々の嬉々とした笑顔だ．人間には何より自由が一番だ，とつくづく感じた」

2. SEA はどこから来たのか？

私の会員番号は 860358 だから，SEA に加入したのは設立後何ヶ月か経ってからである．設立当初のメンバの方々からみれば新参者であろう．設立にいたる経緯や思想，あるいは難産だったというそのプロセスの仔細を知らずに，SEA はどこから来たのかについて言及できる身分ではない．しかし，個人的な感想を述べる程度は許して貰えるだろう．

私には，SEA とは，旧来のさまざまな社会的なメカニズムの枠内での活動に厭きたらなかった人々が，より大きな自由を求めて作ったコミュニティだと感じられる．SEA 設立にいたるキーワードは，ソフト協・センター協・JISA・Σプロジェクト・JSD などであるが，これらはいずれも，ソフトウェア開発の実践的技術の交流を促進するための非学会的な活動の場であった．そうした既存の社会的メカニズムの中で活動していた（主としてソフトウェアハウスの）技術者たちが，より自由な夢を語るための新しい舞台を求めたのだ．

それは，中世のギルドに似た形の閉鎖的なプログラマ組合という風にも解釈でき，そこから多くの誤解が生まれたようである．いくつかのソフト会社で，社員に対して SEA 加入禁止の社長命令が出されたという話が，笑い話ではなく今でもときどき聞かれるのは，特別不思議なことではない．しかし，よく考えてみれば，SEA は労働組合のヴァリエーションにはなれないし，また，ならないことは明白である．現在の日本の社会構造からみて，個人加入の任意団体が社会的な圧力団体には決してなれるはずがない．特に，賃金や労働条件の改善などといった社会的・政治的な活動など到底不可能である．それに，SEA のメンバーもそんなことは微塵も考えていない．

私が SEA に加入した頃，ソフトウェアハウス以外に，メーカ，ユーザ，あるいは大学・研究所関係のメンバも少数ではあるが何人かいた．かれらもやはり，旧来の社会的活動の場に厭き足らずに，興味本位で SEA に入ったようだ．そして，SEA の活動内容の面白さと新鮮さにすっかり虜になってしまい，クチコミで周囲の人々を次々に SEA に引き入れてきたのである．SEA のエネルギー源の一つは，個人会員がベースであることだ．所属している機関とそこでの職位には依存せず，個人で何ができ何を発言できるかが，ここではもっとも重要なのである．したがって，教授・助教授・助手・社長・常務・部長・平社員が，同じ舞台で対等に議論できるのである．そのせいで，いま SEA には，従業員数人のソフトハウスから，一部上場の大企業にいたるさまざまな組織の技術者が混在している．この辺が SEA の SEA たる所以であろう．

こうして，あらためて SEA がどこから来たかを問えば，それは，形骸化された現在のソフトウェア産業構造の中から必然的に生まれてきたものであることがわかる．ツリー構造の意思決定機構がルール化された建前の社会では，決してなし得ないようなことがらを自らの手で研究し，開発し，実践しようとする飽くなき希望を持った人々の集団，それが SEA にとってもっともふさわしい形容であり，その誕生と発展の経緯をあらわしているといえよう．そんな風に私は理解している．

3. SEA とは何者か？

SEA は，一切の建前を外した本音の議論ができる日本で数少ない場の１つである．俗っぽいいい方をすれば，SEA とは，学会でもなく，政府筋にぶら下がった機関でもなく，ある筋のパトロン付きの協会でもなく，政府予算獲得のための会社などでもない．それは，同好の士の集まりという表現が最も適している１個の任意団体である．

一般に日本の社会における任意団体には，次のような特性がある．１つは，ガチガチに閉じた組織で，もっぱら内部のメンバだけの利益を追及するという傾向であり，もう１つは，特定の企業あるいは個人との強い癒着関係である．後者の特性は，実際にはその団体がある企業または個人の所有であるにもかかわらず，それを公にしないで，社会的に形を整えるために任意団体を装う際にあらわれる．私の知っている任意団体の多くは，この両方の特性をあわせ持っている．さて，それではわが SEA はいったいどうだろうか．

まず第一に SEA が閉じた組織か否かという点だが，御承知のように，しばしば SEA は会員サービスが悪いのではないか，という内部批判が出るほどに，開かれた活動を展開している．その意味では，単なる同好の士の趣味的な集まりというよりは，むしろ，巾広い社会的な活動を意図したボランティア集団と考えたほうが，SEA の特徴をよりよく表現しているといえよう．

次に，特定の会社や個人との癒着についてであるが，会員および賛助会員会社のそれぞれが，各自のできる範囲の中で，SEA の日常活動やさまざまなイベントに，ボランティアとしての「無償」の協力をしてくださっている．その例をあげればきりがないが，たとえば，前代表幹事であった岸田孝一氏の御好意で，事務所スペースの提供や事務局人件費の負担という形で，(株) SRA のお世話になっている．法人格のない任意団体の SEA では独自に事務所を借りたり人を雇ったりできないという弱点を，こうして補っていただいているのである．また，SEA の毎年のメイン・イベントであるソフトウェア・シンポジウムでは，実行スタッフの方々から，それぞれの所属組織のリソース（人・モノ・金）を必要に応じて提供してもらっている．商業ベースのイベントでないわれわれのシンポジウムが，採算的に十分以上に成り立ち，SEA 本体の財政にかなりの貢献がなしえているのは，実にこうした方々の協力のおかげである．

SEA が既存の類似組織と異なるポイントは，これらの協力が，経済性や私的利益の追求を第一義とする通常の社会原則とはまったく無関係に，ボランティア精神だけにもとづいて行われているということである．したがって，そこにはなんらの癒着関係もなく，SEA を私物化しようとかいったよこしまな論理も存在しない．その意味で，SEA は（哲学不在と世界各国から批判されている）日本の経済社会の中では，実に例外的なふしぎな団体である．

世の中には，そうした事情がまったく理解できず，SEA は一部の人間に私物化されているのではないか，といった悪口がときどき聞こえて来りする．そういう話は私は嫌いだ．何か，志が低すぎて哀しくなる．私が考えている SEA は，もっと個人が愉しめて，かつ，おもしろい場所なのだ．昔から「彼は在野でありながら一流の仕事をした」という表現がある．ここでいう在野とは，広い意味であって，単に官職についているかいないかをいっているのではない．むしろ，「恒産無くして恒心無し．恒産在りて恒心在り」という時の恒産に近い．近頃の金融証券疑惑に見られるように，いまの社会状況の中では，私は，上述の論語の表現は完全に逆転したと考えている．つまり，「恒産在りて恒心無し．恒産無くして恒心在り」である．感覚が鋭敏な若者が恒産を嫌っている傾向がよくわかる．SEA は，「恒産無くして恒心在り」が実践できる場である．

人は，よく初対面の人の品定めをする．世の中一般には，現在の肩書と学歴だけで判断する人が多いが，そのアプローチは SEA ではほとんど役に立たない．「恒心」をベースに社会と技術を議論し，考えて行く際には，会社でどれくらい偉いかとか，何年か前に数年間遊んだ学校がどこであったかなど，どうでもよい．その人が，いま，１個人として何をどのように考え，実際に何をなしうるかが，問題なのだ．問われているのは，個人の力と姿勢なのだ．

SEA は，その設立趣意書にある通り，ソフトウェア技術の実践的側面について，個人ベースで愉しみ，学習し，発見し，他者と交流する場である．そのアプローチは，科学的／工学的／社会的／文学的／社会科学的／芸術的／社会心理学的など，何

であっても構わない．ただ Keyword は，ソフトウェア技術であって，これは外さない．

最近の SEA の活動の中で主要なイベントを挙げれば，ソフトウェア・シンポジウム，環境ワークショップ，全国縦断フォラム，春秋のセミナー，月例フォラム，日中シンポジウム，教育ワークショップ，若手ワークショップ，テクニカル・マネージメント・ワークショップ，各支部活動，各分科会活動などである．何れも産・学・官のバラエティに富んだ階層からの参加をみている．

こう考えてくれば，SEA とは何者かがわかる．ERA モデルでお馴染みのように，あるエンテティは他のエンテティとの関係を明確にすれば，その意味がわかってくるのである．

ちなみに，SEA に関係するいくつかの団体の特性を拾ってみる．情報処理学会は，鈴木則久氏がどこかで指摘しているように，山車学会であり，皆が御輿を担いでいるようなものだ．日本ソフトウェア科学会は，情報処理学会が国内の大手コンピュータメーカの手に落ちた（？）ので，実質的なアカデミックな活動ができる場として作られたらしい．この学会も，まだ SEA と同じく任意団体であるが，閉じた集団という形から一歩踏み出すことができていない感じである．現実の産業界に対するインパクトが少ない嫌いがある．志が高いことはわかるが，思い余りてその体制いまだ社会に定着せず，といったところだろうか？

JISA は，歴史的には SEA と最も関係の深かった団体である．法人会員ベースの，ソフトウェアハウスと計算センターの業界団体である．通産省とも関係があり，公的なソフトウェア産業振興の一翼をになっている．法人ベースということがこの組織の正確を象徴的に表現している．SEA とは，対角線上にあり，補完関係の位置にあるはずである．SEA に最も近い団体に，日本 unix ユーザ会（jus）がある．物理的にも事務所がお隣どうしである．これも個人ベースの任意団体である．jus は，unix に特化した団体であり，unix ユーザに対していろいろなサービスを行っている．unix シンポジウムは，最近ビジネスにおける宣伝の場にもなっている．

4. SEA はどこへ行くのか？

人間にもまた社会にも「表」と「裏」が必要だ．

誰もがつねに「表」の世界を重要視している．それをどうするかで四苦八苦し，挙げ句の果てに泥沼にはまってしまう．そして，物事は形骸化し，つまらなくなり，構造と意味との乖離が始まり，退廃が進行し，残るのは，権威の発揚と権力表示と無意味なそしてムダな政策の実践だけになる．これは，人類の歴史が始まって以来変わらない事実である．大企業病，親方日の丸精神，日本社会に対する欧米からの批判，多くのコンピュータに関する不祥事件などは，こうした社会構造から発生した．いずれも「表」しか見ない日本民族の哀しさのあらわれである．

「表」社会，つまり建前の社会だけでは人は生きて行かれない．それは，技術，科学，芸術，社会生活，精神生活などあらゆる面で同じである．SEA は，こんな社会状況の中で，ソフトウェアの技術と科学を主テーマとして，「裏」の社会の構築を目指したい．そこでは，個人がベースであり，本音がキーとなる．すなわち，各個人が所属している機関からのあらゆる束縛を離れて，自由に伸び伸びとソフトウェア技術や科学を愉しむことができる，そして，あちこちの同胞たちと交流することによって，社会の構成要素である「個」が，さまざまな方向に励気されるであろう．

アナロジーとしては，次のような関係を夢見ている．つまり，中国における儒教と道教の関係である．儒教は「表」であり，形式であり，ルールであった．それは，時代の，つまり為政者の要請を吸収しながら，社会的な礼節とそれを実行するプロセスを表現した．無意味で耐えられないものであるが，本来「表」とはそうしたものである．その背後で，何千年にも渡る儒教世界を支えたものが道教であり，これは人間の個人の生き方を支えた．それは，個人をベースとした生き方の神髄と徳を説いて，深く人々の心の中に入り，心の拠り所となった．「裏」がしっかりしていれば，「表」が大きく変動しても人心は乱れない．

表裏一体と昔からいうが，現在の「表」社会の軽薄短小さは，まさに日本人の「裏」のなさというか，浅薄さの証明である．SEA の活動を介して，コンピュータ産業とソフトウェア産業の「裏」の世界を築き，その中でじっくりといろいろなものを熟成させたい，と感じている．それは，１企業の努力や１官庁の政策をもってしては達成できない仕事である．「裏」が豊饒で強くなれば，自然に「表」の社会も同じように変化するはずである．そんなところが SEA の目的だと，私は考えている．諸兄姉のご意見を乞う次第である．

熊谷さんの問いかけに答えて

筏井　美枝子（SRA・名古屋）：

ムムム，熊谷さんのエッセーを読んでいたら，釣られて，つい堅い文章になってしまいました．これまでSEAで大した活動をしない名前だけの会員だった私が，幹事になったとたん平気で以下の様なわかった風なことをいっている分けで，われながらいい加減なものです．このいい加減さが，まさにSEAの会員証明だったりして:-).

＜純粋な技術者集団としてのSEA＞

技術者と称するものにとって,"よいものを創る"というのは，永遠かつ不変の命題である．いつの世も，技術者は常に"よいもの"を追い求め，そこに生きがいを見い出す．時には，その願望が周囲の体制等とあい入れず，深く悩む技術者も少なくない．それは，技術者たるものの宿命ともいえる気がする．逆に言うと，他の何にも増して"よいものを創る"ことに執着した時，技術者は真の技術者たり得るのかも知れない．

純粋に"よいものを創る"ことを悲願とした時，より広くより深い洞察や知識が欲しくなる．同じ悲願を担い同じ宿命を背負った物達が集まり，私利私欲を離れ，熱い夢を語り慰め励まし合い，同胞を増やそうとするのは天然自然のことわりではなかろうか．利益とか効率だけを追求すれば質は落ちる．様々なものの質が落ちれば，社会そのものの質も落ちるだろう．現在，私たちはその恐れの多分にある世の中に生きている．とすれば，純粋に質を求める技術者集団が様々な分野で生まれ広がっていくことは，世の中の質の低下に歯止めをかけるだけでなく，質を向上させる唯一重要な意味を持つものかも知れない．

SEAはそんな集団であって欲しい．そしてそんな集団であり続けて欲しい．

臼井　義美（日本電子計算・大阪）

先月の幹事会でSEAはどこから来たのか？EAとは何か？SEAはどこへ行くのか？などという根源的な問いが発せられたことをe-mailで聞き及び，感涙にむせんでおります．

思えば，SEAは，SEの不満分子の集まりだとか，プログラマのなれの果てだとかの罵声を浴びせられながら，そうした中傷を全く気にしない厚顔のSEもしくはSEであった人達によって設立されました．そして，当時の女工哀史を地で行くようなSEやプログラマの駆け込み寺として，また，ソフトウェア開発環境の革新を提案したものの会社に受け入れられず矢折れ刀尽きてボロボロになった傷心のSEの救済の場として活動してきました．

このための活動としては，技術移転，教育，開発環境，ソフトウェアの信頼性，テクノストレスなどがあります．私自身はソフトウェアの信頼性を追い求める側より，しくじったSEを鞭でしばく方が向いていそうなので，システム監査の方に転向してしまいました．また，テクノストレスのパネル討論に出たこともありましたが，何がストレスなのかよく解らなくて困ってしまいました．

このようにSEAの活動の一つとしては，世の中の苦渋に堪え切れなくなったSEを救済することでしょう．また，情報一極集中国家における情報過疎地に棲息しているSEに，手ごろな情報流通機能を提供するという役割をも果たしています．

一方，技術者協会という位だから技術についても取り組んでいると思います．多分日頃はエンジニアリングとは程遠いプログラム製造業にいそしんでいるSEたちが，不満の吐け口として白熱した討論が展開することがあります．また，奈良時代の遣随使よろしく，欧米の研究成果を持ち帰って普及活動にあたっている人たちや，あわよくばテキトーな技術を利用して一儲けを企てるSEたちの情報交換の場ともなっています．

このため，SEAはCASE，プロセスプログラミング，オブジェクト指向など，ものになるかどうか解らないような新しい技術を，単なる研究ではなく大胆にも実践的に取り組んでいます．,br

さて，これからのSEAの進むべき道について一言申し述べます．この技術の変遷のはげしい時代にあって，ほとんどのSEは落ちこぼれて行きます．従ってSEのなれの果てを救済することはもちろん，誤ってこの世界に足を踏み込まないような正しい指導が必要であります．このためには技術だけを追い求めるのでなく，ソフトウェア文化を育成する活動が不可欠です．そのためには，よって立つべきアイデンティティを確立しなければなりません．

日本は東洋と西洋の思想の合流地点に位置しており，双方の文化を融合化できる独特な国です．SEAは西欧を中心として発達したソフトウェア技術にオリジナルな感性を加え，より洗練されたソフトウェアや開発環境を提供して行くべきでしょう．このためのSEAの活動は，中国との交流や，タオを説く代表幹事の貴いお姿に見られます．

私は最近いにしえの都である奈良に居を移しました．先日は友人のアトリエで東に若草山を臨み，南に興福寺の五重の塔を見ながら，先人たちの偉業を偲びつつビールをの

みかつ語りました．去年まで大阪の堂島に住んでいて，現代表幹事のクマさんと金類愛と人類愛を語り合った SEA 設立当初が懐かしく思い出されます．

技術がもの作りのためであり，労働が人月で量られ，それを金銭を基準として評価する時代は去りました．カネやない！ココロや！

もはや，物を利用したマネーゲームを楽しむ時代ではなく，人の感性を刺激し，コーフンさせて楽しむ時代なのです．SEA も真剣に人間の快楽を極め，コーフンするソフトウェアの研究に邁進すべきであります．それにしても奈良は心が休まる街ですね．

小林　俊明（ソフトウェア技術）

・「SEA とは何か」と問われたならば

やはり「組織の枠を越えた技術移転／技術交流の場」なのではないでしょうか？ SEA に参加することに意義を見いだすとすると，社内やコンピュータ雑誌では得られない技術情報を得られることと結論づけることができると私は思います．

技術情報を得るとは単に情報をハックするだけでなく，場に参加することでエンジニアとしての自分のポジションを自分の外側からもう一度見つめ直し，自分自身を再確認するメジャーも提供していると思います．

やはりそのような外界からの刺激があってエンジニアとして次はどのように振る舞うか自問自答し飛躍が生まれるのではないでしょうか？興味と刺激がなければやはり発展（真の技術移転）はないように思えます．

・「SEA はどこへいくのか」と問われたならば

私としては「組織の枠を越えた技術移転／技術交流の場」としてのスタンスは崩さないでいてほしいと思っています．技術交流を考える場合やはり双方向の交流が会員各位の間で活発化することが望まれると思います．さてどうしたらもっと活性化できるのでしょうか？

中谷　多哉子（富士ゼロックス情報システム）

やはり幹事になったからには，このことについて考えなければなりませんね．遅くなりましたが私の意見を．

SEA とは，日々の業務に追われ，業務を改善する方法がこの世に存在することも知らない優秀な技術者に一筋の光明を与える協会です．

困っていませんか，技術者の皆さん？　皆さんの仕事の効率を善くする方法を，SEA は紹介できます．

問題はありませんか？　もし皆さんが悩んでいることがあれば，共に考えてみましょう．同じ世界に生きる技術者として，皆さんの問題は私達の問題でもあるはずなのです．問題意識こそが，技術者を救う最初の一歩だと思います．SEA は問題意識を持ち，それを解決する方法を考え発表する協会です．

私をオブジェクト指向の巫女と呼ぶ人がいます．ううん，何か神様のご神託みたいですねえ．私はまだ加入して間もないので夢が大きいでしょうか．

SEA がこれから何処へいくかですか？　四谷からまた移転するんですか？　と，つまらない冗談はやめて．

これからも村の仲間や，山の仲間に機会を作ってあげましょう．仲間が増えたら，また次の方向を考えましょう．話しが変わりますが．千葉方面を啓蒙することがそろそろできるかも知れませんね．最近幕張あたりに移転する企業が増えていますから．YDOC の対岸で烽火を上げてもいい頃合ですよね．もうありますか？

ところで，今日は雷がなってとてもエキサイトしました．都庁に１００発位落ちたような騒ぎでしたよ．

(追伸)：なにせ SEA の幹事と言うとお兄様方がたくさんいらっしゃるので，緊張してしまいます．冗談も堅くなります．言いたいことも言えなくなります（これは嘘）．いぢめないでね．

中山　照章（富士通 SSL）

＜ SEA はどこから来たのか？＞

SEA 設立時のメンバーではないので，よく分からない．代わりに私自身が SEA に入会した動機を書く．

当時バリバリのプログラマ（？）だった私は，漠然と「会社の外のプログラマの集まりがあったらいいな」と思っていた．タコツボ掘りのようなプログラミングの仕事をやっていたので，掘れば掘るほど，どんどん深くなって世間との接触が少なくなり，酸欠状態になっていた．ここらで外の風に当たりたいと思って SEA に入会した．

また，一般に言われていた「プログラマから SE（システムエンジニア）へ」がダサいのに比べて，「プログラマからソフトウェア技術者へ」というのがカッコよく，SEA に入会したら自分も一人前のソフトウェア技術者になれるんではないかという甘い期待を持っていた．

＜ SEA とは何者か？＞

実際に入会してみたが，SEA Forum に参加しているうちは，会員価格で参加している単なる観客の域を出なかった．ところが分科会に参加すると，そこは別世界であった．色々なしがらみから開放されて自由な雰囲気で話し合える場である．ここにこそ，SEA の面白さがある．各種イベントの企画や運営をする方に回ると，さらに面白い．

幹事をやっていて感じたことは，SEA の色々な活動が人と人のネットワークで支えられているということである．そこで，＜ SEA とは何者か？＞に対する第一の答えは，次

のようになる．

－　SEAとはソフトウェア技術者のネットワークである．

SEAの活動に参加すると，日常の仕事からは得られない，いろいろな刺激を受けることができる．SEAの活動に参加すると楽しい．そこで，第二の答えは次のようになる．

－　SEAとはソフトウェア技術者のための遊園地である．

＜SEAはどこへ行くのか？＞

SEAを遊園地にしてしまったのだから，この先いき着く所はひとつしかない．さらに拡大させて，ソフトウェア技術者のための大レジャーランドにしなければならない．そのために，ますます，遊びのネットワークを拡大しなければならない．周りの人間をどんどん引きずり込んで，一緒に遊びましょう．

いませんか？　あなたの周りに．禁断の遊園地に興味はあるが遊びに毒されるのを怖がっている人が．そんな人にひとこと「SEAは面白いよ」って言って誘ってあげましょう．それがソフトウェア技術者の思いやりってもんですぜ．

野村　敏次（日本電子計算）

熊谷さんが常任幹事会で問題提起されたということですが，問題提起ということは，現在何等かの問題があり，将来にそれが影響するからこそ，問題として提起されたのだと思います．ただ，それが何であるか判らずに議論してもあまり生産的な話にはならないのではないでしょうか？

もし，何も前提がないのなら，過去の話をするのではなく，将来SEAはいかにあるべきか？　そのためには，今何をなすべきかを議論した方がよいのでは？そのためには，まず，代表幹事自らがSEAをどのようにしたいか，あるいはSEAはかくあるべきという考えをお話になり，それに関して幹事が意見を述べるのが筋であると思うのですが？

もし，皆の意見によってSEAは運営されているのであり，代表幹事の意見を聞くのはおかしいというのなら，代表幹事は誰がやっても同じでありまったく面白くない．代表幹事によってSEAがその特徴を出せる組織であって欲しいと願っています．熊さんらしいSEAというのが必要かと思います．

ところで，私自身SEAに対する将来の希望ですが，まず，自由な組織であって欲しい，大きな組織であって欲しい，ソフトウェア技術者がSEA会員であることに胸を張れる組織であって欲しい．そのためには，まず会員を増加させなければならない，組織として効率的な活動ができるように体制の見直しをしなければならない，等々．

何か書いていて，あなた任せで，我ながら情けない．自分が，何をなすべきか，何ができるかを，これを機会に考えることにしましょう．

野村　行憲（岩手電子計算センター）

「SEAはどこから来たのか？」って？

よくは知らないんだけど，噂では格式だけ高い情報処理学会と，不毛の情産協が協力してパンドラの箱というのを作ったんだけど，それを通産省とかいう人が，うっかり空けてしまったときに，最後に出てきたのが，SEAだったという話だよ．

だから，「SEAとは何か？」っていうと，そりゃあやっぱし，希望とかいうやつじゃない？

そんな話だったから，入っていたら自分の行くべき道を教えてくれるのかと思っていたんだけど，どうやらこれは甘い考えというやつだったんだよね．だってそりゃそうじゃない，ここは協会であって，教会ではないんだからね．

そんな訳で，「SEAはどこへ行くのか？」っていわれても，そんなの自分で見つけるしかないんじゃないの？「協祖様」とかいうのをそそのかして，自分の行きたい方向に向けてしまうのが，やっぱ一番いいんじゃないの？

ま，世の中そんなに甘くないって訳なんだよね．きっと．

私なりの SEA へのこだわり*

渡邉 雄一†

1991 年 8 月 1 日‡

1 SEA はどこから来たのか？

SEA の設立以前のことは，さすがに若くて[1]知らないが，その華々しい設立総会[2]から顔を突っ込んでいる一人としては，『そんなの知らないよ』とトボケ通すのもちょっと辛い[3]。まあ，でもこんな任意団体に自腹を切って[4]入っているからには，それなりに思うところがあるのでそれを書き殴ろう。

そもそも，私がキックオフの時から関わってこれたのも，ソフトウェアエンジニア[5]だけでなくユーザも含めて良いインフラストラクチャーを築き上げて行こうということが唱われていたからに他ならない。このスタンスは今でも変わっていない。たまたま，一生を棒に振るのが厭だったから今の会社に移ったが，それでも自分に出来る仕事といえば，利用者のための仕事のし易い環境作りのお手伝い[6]でしかない。そのような同志の語り合える場であることを期待し SEAMAIL の編集に，当初から[7]関わらせていただいた。ちょうどその年に，会社で使用している計算機のユーザ会の研究会に参加していて，自分なりにアクティブに活動しているにも関わらず思った反応が得られなくてストレスがたまっていたこともあって SEAMAIL に雑文を書いて憂さを晴らすのは，ちょっとした作家気分で痛快だった[8]。

ちょっと話しは横道にそれたが，僕としては**ユーザも交えた個人参加が原則 の ソフトウェアテクノロジを語る場 が真剣に求められていた** ということも どこから[9]ということの要因の 1 つとして書きとどめておきたい。

2 SEA は何者か？

今の SEA の本体が主催する活動は，有志が集まったことの 意義 を前面に押し出すために professional な集団[10]と捉え，そのための質 (先進性，実用性) を保った話題/議論にエネルギーを向けている。僕にはとても出来ないことである。だから賛成だ!。自分と同じレベルの人しかいないなら何ら勉強にならないからである。もちろん本来なら学会などでも同じであるはずだが，実際にはそこが大きな違いなのである。真の意味での SEA の存在を意義がここにある。志しある人には，手とり足とり教えてくれる[11]ということである。しかも，今までの幹事の努

---

*できそこない幹事の 1 人として仕方なしに雑文をしたためているが，こんなことを書いていて，よ〜〜っぽど暇な奴と誤解されるのが，と〜〜ってもこわい !?

†(株) アスキー OA 推進部

‡正確にはもう 0 時を回ったので 2 日

[1] 当時 26 歳だった。今はミソジを越えたが，何故か SEA の中ではまだ若い部類だ!

[2] 機械振興会舘の 1 室で行なわれたが，空腹だったので六本木のマクドナルドでハンバーガを買ってから行き，会場でカブリついていた記憶がある。かなり寒い日だった

[3] だからといってそんなに知っている訳ではない

[4] 私の現在勤めている会社は，対外活動などには非常に良心的で 学会費 はリーダの承認を得れば支給してくれる。だけどちょっと SEA にはこだわりがあるので，私は自腹を切っている

[5] SE と書いて，System Enginner ではなく Software Enginner と読ませていたのは印象深かった

[6] その 1 部に，アプリケーション開発があったりすると思っている

[7] kick–off 前に勧誘も兼ねたアンケートがあって，ボランティア活動をしないと SEA には入れないとと書いてあった。(これホント!) だから仕方なしに編集グループに丸をつけて出したら 今の泥沼にハマッタ!。でも後悔はしていない

[8] 最近は，自分の発言が物議をかもす年頃/立場になったので，いろいろな面で自重せねばと思い始めている

[9] 熊谷さんをはじめ多くの業界の人の気づかない 1 面だろう

[10] 業界のオジサン達からは，『技術崩れのヤクザ集団』と思われているらしい。ちなみに僕が以前勤めていた会社 [(株) 電力計算センターという] では，トップの方は技術的な展望が無いから，良くも悪くも何もそのようなことは理解していなかった (不感症とも言う) ようだが

[11] 例えば SEAMAIL にレポートを書くと言うことで，何と勉強させてもらったことかと思う

力で, それらが実に open に行なわれている[12]ということが嬉しい。

そして, そのような SEA 会合に参加していて思うのは, 志しの無い業界人によって自分達の仕事が『ずいぶんとつまらないものになっているなぁ』と感じることも少なくない。ユーザの中にいて仕事をした経験が長いから強く感じるのであるが, 素人の足元をみるようなプログラムが少なくないのである。他の業界の不祥事[13]をみるにつけ, この業界も早く自己浄化機構を組み込まないと大変なことになるのでは, と危惧している。Unix などの浸透で, 自然の駄目なプログラマは淘汰されてゆくのかと楽観視していたが, とんでもない!。未熟な者[14]は, 勉強しない限り駄目[15]なのだ!。詳細を別の機会にぜひとも詳細に書きたいと思っているが, 私のところでは昨年の 8 月後半から今年の 3 月まで仕事をしてもらった K 社[16]の システム開発室[17]で 僕の仕事を担当してくれた人達[18]の仕事内容には, 実に腹が立った[19]。

このようなことは, 普通のユーザが当たり前に経験しているのだと考えると実に寒気がしてくる。一方で, GNU に代表されるような良質の Free Software の普及など心強い面もあるが一部の PDS などのように, 無料だからバグがあっても良い, ということをしきりと強調するような傾向が少なくない[20]のは, やはり悲しむべきではなかろうか?。

で, SEA とは こんな当たり前[21]の事柄を, 時には激しく, 時には優しく, 語り合える人達が中心となって形成している組織[22] だということは間違いない。

3 SEA はどこへ行くのか ?

活発に活動している人の平均年齢が徐々に上がってきているとはいえ, 従来からの活動に問題はないので, 少なくともその方向性はそがれないだろう。例えは, 現在 SIGENV のように, 自分の出来る範囲[23]で, 自分の感じることに共鳴してくれる人/共鳴させてくれる人と, その活動を協力して進めてゆくことは, 従来と同じであろう。

ただ 人の考えは千差万別である。広い意味で人に危害を加えるようなソフトが, 激減する/させる ような時がきたとき[24]に, 自分がそれに値するソフトを作っている自信があるかと自問自答すれば, 答えは No である。そのような輩に SEA の行方など伺い知ることなど, 到底出来ない[25]。

ただ いずれにせよ その答は,

軽い気持ちで, 何らかの適当な機会に SEA の活動を楽しむこと をしないでは知り得ないだろう!

[12] 今までの各種のイベントに幹事の個性が強く表れていたのは事実だが, それと open に運営されているということを 混同/誤解してはいけない

[13] もちろん言わずと知れたバブル経済の崩壊のこと。証券業界の損失補点など 大蔵省の影が後ろにちらついて実に腹立たしい限りだ!

[14] もちろん私も含めてだが, それ以上にひどい奴がごろごろしている

[15] 少なくともお客の望むものを満たす義務はあるから, このような表現をしているだけで, その会社の人間が勉強しようといまいとこちらには被害が出ない限りどうでもよい

[16] あとでの人名は実名なので, さすがに会社名を明記するのは気が引ける

[17] この会社全体がこのようで無いとは思いたいが

[18] 宮里武幸室長, 大平直之さん, 飯野隆子さんの名前は特筆したい

[19] Unix をやっていて, まともに Makefile も書けないのだからしょうがない

[20] でも高額でバグの少なくないのはもっと許せない!

[21] もちろん業界のオジサンにとっては, どうでも良かったり, とんでもなかったりする

[22] SEA の偉いところは, 来る人拒まず で, 志しの高低に関わらず入会出来る (どこかの学会とちがって『正会員の推薦を要す』などということが無い) ので, SEA の会員だからといって先入観を持って接することは 危険である。少なくとの前に勤めていた会社には, そのような人がいた

[23] そろそろデートの時間も それなりに必要と考えるようになってきた

[24] これってやっぱり目標にしたいよなぁ〜

[25] 例え冗談にでも

オブジェクト指向CASEの動向

佐原伸(1) 中谷多哉子(2) 藤野晃延(3)

1.（株）SRA ソフトウェア工学研究所
2, 3. 富士ゼロックス情報システム（株） 企画部

要約

オブジェクト指向分析／設計（OOA/OOD）の手法とオブジェクト指向CASEツール（OO-CASE）を調査した。OOA/OODはすでに実用的なレベルに達し、構造化分析／設計（SA/SD）を包含しつつある。SA/SDからOOA/OODへ移行する条件は整った。OO-CASEは従来のCASEより統合的なツール・環境としての機能がまだ未熟であるが、かなりの機能を持ったものが出てきた。なお、多くの手法やOO-CASEがなお開発中である。

1.0 はじめに

プログラミングの世界は、オブジェクト指向プログラミング[Meyer 88bJ]によって急速に変わりつつある。オブジェクト指向プログラミング（OOP）を押し進める力となったものは、システムの巨大化・マルチメディア化・ユーザーインタフェースの複雑化・分散化などである。

従来からの構造化分析・構造化設計（SA/SD）[DeMarco 78J][Hatley 87J][Ward 86]や、ましてそのほかの古い手法では、これらのシステムを開発するのは困難であった。このため各種のオブジェクト指向言語が開発され、その言語による開発を支援するオブジェクト指向ライブラリーの開発も相次いだ。最近では、Apple社のMacintoshパーソナルコンピュータ上でで、MacApp[Apple 90][Wilson 90]と呼ばれるObject Pascal[Apple 91]とC++[Stroustrup 86J]のオブジェクトライブラリーが開発されている。同様の機能を持った、NeXT社のワークステーション上のNeXTStep[橋本 90] というObjective-C[Cox 86J]のオブジェクトライブラリーも開発されている。これらのマシン上では、オブジェクト指向パラダイムを使って、多くのアプリケーションプログラムやツールがすでに開発されている。

構造化プログラミングだけでは巨大なシステムの開発に支障をきたし、SA/SDとそれを支援するCASEツールが必要になったように、OOPだけでは大きなシステムを開発するのに困難が生じる。当然の帰結として、オブジェクト指向分析・オブジェクト指向設計手法と、それを支援するCASEツールが必要になってきた。

本稿では、最近になって多くの手法が出現してきたOOA/OOD手法と、それらを支援するCASEツールについて、その主なものを概説する。２章でオブジェクト指向CASEツールのベースとなる主要な３つの方法論について解説し、3.1節でそれらの方法論を支援する上流工程支援CASE（Upper CASE）ツールを解説する。また3.2節では、これらの方法論によらない下流工程支援 CASE（Lower CASE）についても紹介する。さらに3.3節では、オブジェクト指向による開発に欠かせない、ユーザーインタフェース構築支援のためのCASEツールを紹介する。3.4節では、オブジェクト指向を支援するビジュアルプログラミングツールについて述べる。

本稿の狙いは、オブジェクト指向パラダイムを使用して、実用的なシステムを開発する際に助けとなる手法とCASEツールを、分かりやすくかつ組織的な視点から捉え、類似性と相違点を明確にすることである。

Ada[DoD 83] を支援する方法論やCASEツールは、継承を支援していないこと、日本での重要性が低いことから、基本的には調査対象から除外した。

2.0 ベースとなる方法論

OOA/OODの方法論の多くは、ChenのEntity-Relationship（E-R）図[Chen 76]やSA/SDあるいはJSD法[Jackson 83J]の考え方を取り込んで、オブジェクト指向独自の記法や方法論を追加したものである。

Shlaer／Mellor[Shlaer 88J]やBoochの古い手法[Booch 83]あるいはHOOD[HOOD 89]などの初期のOOA/OOD手法は、後述する最新の手法に取り込まれ、より完全な形になっていったためここでは触れない。

2.1 評価の枠組

各手法を評価するための枠組として、以下の項目を選択した。

（1）オブジェクト指向の捉え方

オブジェクト指向は、ある意味では曖昧な概念であり、それぞれの手法によって微妙に定義が異なるため、その手法でどのようにオブジェクト指向を定義しているか、あるいは捉えているかを記述する。

（2）継承する方法論

その手法の元になった方法論と、継承した性質を記述する。

（3）対象とする言語

主として支援するオブジェクト指向言語あるいはその他の言語を示す。

（4）モデルと記法

その手法がどのようなモデルを採用しているかということと、そこで使われている記法について概略を記述する。

（5）手順

その手法でどのように分析・設計を行うかの概略を示す。

（6）特徴および問題点

その手法の特徴と問題点を述べる。

2.2 Boochの手法

Boochの手法[Booch 90]は、[Booch 83] に述べられているAda指向の方法を拡張し、OOD全体をカバーする方法論としてまとめたものである。Boochの手法は特に名前がないが、本稿では以下「Booch法」と呼ぶ。

Booch法は、オブジェクト指向を手続き指向・論理指向・ルール指向・制約指向などと同じ抽象化の１つの方法と捉えている。ただし、オブジェクト指向が一番適応範囲が広い方法論であるという認識である。

Boochはオブジェクト指向を以下の４つの主要な要素に分類し、どれが欠けてもオブジェクトモデルとは言えないと主張している。

（1）抽象性

抽象性は、あるオブジェクトを他のオブジェクトから区別する本質的な特徴を示し、見る人によって異なる、概念モデルの違いを明確にする。

（2）カプセル化

カプセル化は、オブジェクトの本質的特徴に関係ない詳細を隠す処理である。

（3）モジュール性

モジュール性はSA/SDと同じで、強度が高く結合度が少ない分割をされたシステムの特性である。

（4）階層

階層は抽象性の格付けあるいは順序付けである。重要な２つの階層として、クラスの階層（"kind of"階層）とオブジェクトの階層（"part of"階層）があり、継承は最も重要な"kind of"階層である。"kind of"階層は、他の方法論では一般化-特殊化構造あるいは"is-a関係"と捉えられている。多重継承も考えられる。また、モジュールとプロセスの階層もある。

Booch法は、自身の古い手法[Booch 83]・HOOD[HOOD 89]・OOSD[Wasserman 90]など、Adaの開発手法から強い影響を受けている。また、クラス間の関係の記述などはE-R図[Chen 76] から、オブジェクトの認識方法などはJSD法[Jackson 83J]の影響も受けている。

対象とする言語は特定していないが、主としてAda, C++, Object Pascalなどであり、Smalltalk[LaLonde 90], CLOS[井田 89]なども対象となっている。

Booch法はクラス図・オブジェクト図・モジュール図・プロセス図・状態遷移図・タイミング図の６つの図を使う。最初の２つは論理的な視点からシステムを記述するものであり、つぎの２つは物理的な視点からシステムを記述する。この４つはシステムの静的な意味を記述している。最後の２つは、システムの動的な意味を記述している。

Booch法で特徴的なクラス図を図2.2-1に示し解説する。

クラス図のオブジェクトは、図2.2-1で破線の雲型で示されるクラスと、それに影が付いたクラスユーティリティ、およびこの例では出てこない四角で表されるクラスカテゴリーからなる。

各オブジェクトは、関係を表す線で結ばれる。図2.2-1では白丸付きの２本線は、白丸が付いているクラスのインタフェース部（例えばObject Pascalのinterface unit）が、２本線で結ばれたもう一方のクラスのリソースを使うことを示す。黒丸の場合は、そのクラスのインプリメンテーション部（例えばObject Pascalのimplementation unit）がもう一方のクラスのリソースを使うことを示す。線の両側の数字は基数を表す。矢線は継承を表す。

従って図2.2-1は以下の意味のクラス構造を示す。Controller Utilitiesクラスユーティリティーのインタフェース部がEnvironmental Controllerクラスのリソースを使う。Environmental Controllerクラスのインプリメンテーション部はLightクラスのリソースを使い、

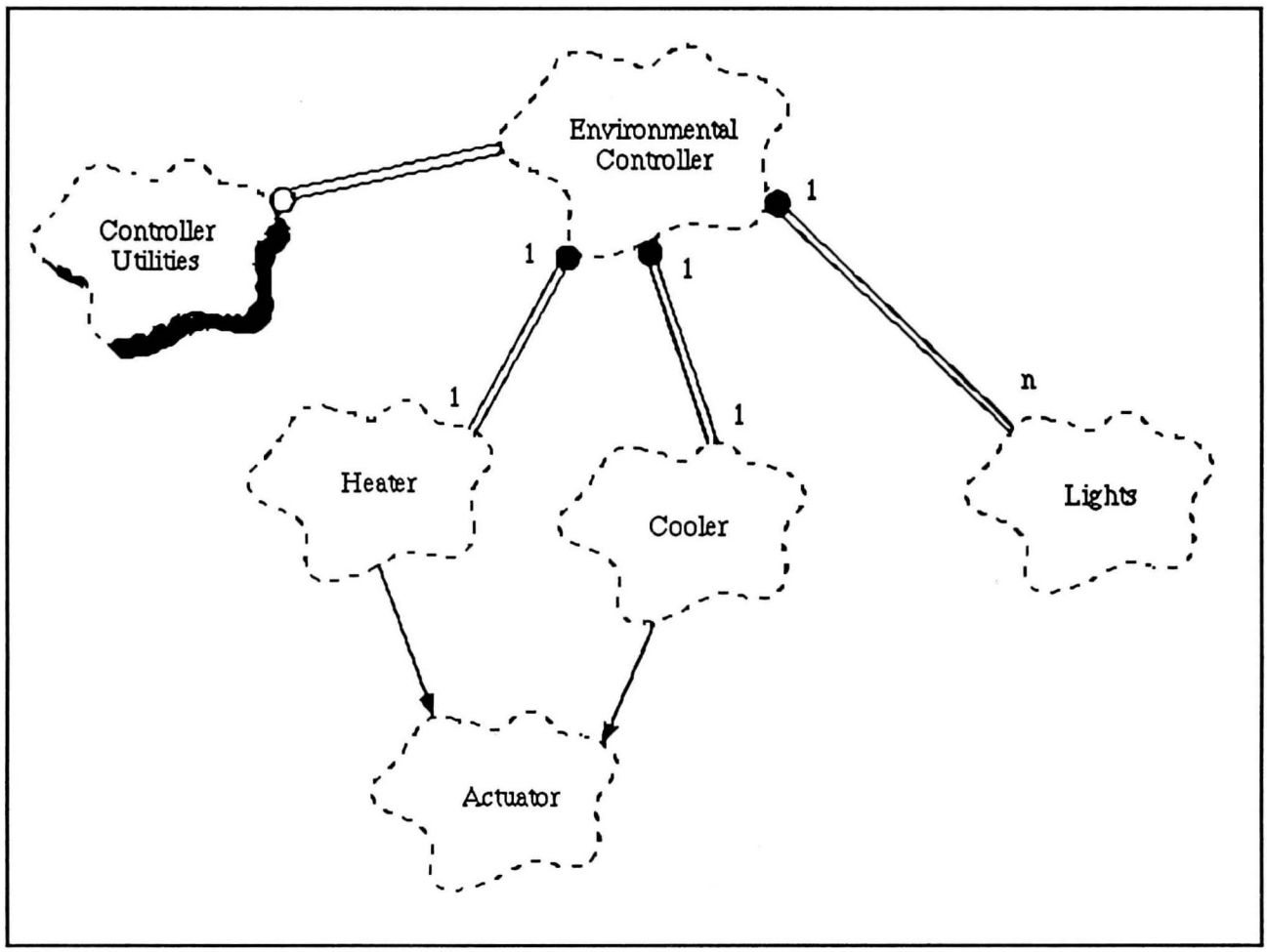

図2.2-1 Booch法のクラス図の例（[Booch 90]から引用）

Environmental Controller一つに対してLightがn個対応する。Environmental Controllerクラスのインプリメンテーション部は、HeaterクラスとCoolerクラスのリソースを使い、関係はそれぞれ1対1である。HeaterクラスとCoolerクラスは、それぞれActuatorクラスから継承する。

Booch法の手順は大きく以下の4つに分かれるが、特に詳細な手順やコツがあるわけではない。

（1）抽象度のレベルに応じてクラスとオブジェクトを見つける。

オブジェクトの見つけ方は、Shlaer／Mellorの方法やCoadの方法[Coad 91]を流用していて、問題を記述した文章あるいは構造化分析の結果から、役割・イベント・人・場所・物事・デバイスなどをオブジェクトの候補として洗い出してくるというものである。最初の分析対象はコンピュータシステムではなく、対象分野そのものである点はJSD法あたりの影響を受けている。

（2）これらのクラスとオブジェクトの意味を見つける。

（3）これらのクラスとオブジェクト間の関係を見つける。

（4）これらのクラスとオブジェクトを実現する。

Booch法の特徴は図の種類の多さと、図中に記述する項目の種類の多さだろう。ほとんどの言語で作成されたシステムの静的・動的な構造、あるいは論理的・物理的な構造を記述することができる。逆にこれがBooch法の欠点になっている。OOA/OODの段階で記述する必要のない詳細まで混在しているため、対象システムの領域や開発言語を特定した場合、不要な図や記号が多くなる。特にSmalltalkで作成されるシステムを記述する場合、かなりの記号と項目は記述する必要がなくなる。

また、OOA工程での指針が少なくOODに偏った手法であり、それも手順が十分に詳細化されているとは言いがたい。

クラス図においてオブジェクト間の結合（association）にあまり注意が払われていないといった欠点もあるが、Ada, C++, Object Pascalなどをベースとし

た開発には一応有効であろう。

2.3 OOSD

OOSD（Object-Oriented Structured Design）[Wasserman 90] は、オブジェクトと既存のモジュールの結合、開発言語からの独立、方法論からの独立を目指した手法である。

OOSDが支援する概念は、オブジェクト指向から

（1）抽象データ型
（2）クラス
（3）継承
（4）多相性（polymorphism）

などであり、Adaの

（5）パッケージ
（6）汎化パッケージ（generic package）

も取り込んでいる。また、SDの

（7）モジュール
（8）共通データ

などの概念も取り込んでいる。

OOSDは、SDの構造化チャート[Page-Jones 80]、Booch法のAdaのパッケージとタスクの記法、オブジェクト指向のクラス階層と継承などからアイデアを得ている。

対象言語は、C++, Objective-C, Eiffel[Meyer 88bJ] , Smalltalk, CLOS, Ada, C, Fortranなど、構造化機能を持っている言語すべてである。

OOSDは図としては1種類で構成される。表現できるオブジェクトは、（1）クラス（2）継承（3）汎化（4）動的束縛（5）同期（6）例外処理、などである。図2.3-1にOOSDの主な記法を示す。

OOSDは特定の方法論から独立であることが設計目標なので、特に手順はない。ユーザーはBooch法や他の方法論を用いて、図表記だけOOSDを使うことができる。

OOSDはSDの記法とオブジェクト指向の記法を融合しているため、既存のシステムを一部改良してオブジェクト指向を導入する時に便利である。また、方法論や言語から独立しているため適用範囲が広い。

一方、OOSDはOOAの手段を（ERE++として計画中ではあるが）現在何も持っていない。このため、OOA/OODで使用する記法と方法が一貫している、OMT（後述）などの方法論は難しい。

また、メソッドが多数あるオブジェクト（例えば普通のSmalltalkのオブジェクト）の場合、OOSDの記法で限られた画面や紙の上に記述しきれるかどうか疑問が残る。

C++, Object Pascal, Ada, Eiffelなどの開発には一応有効であろう。

2.4 Coad, Yourdonの手法

Coad, Yourdonの手法はOOAと自称しているが、一般的なOOAと混同するので、以後Coad法と呼ぶ。

Coad法は複雑さを軽減する原理を、以下のように分けている。

（1）抽象
　　手続きとデータの抽象
（2）カプセル化
（3）継承
（4）結合（association）
（5）メッセージによる通信
（6）組織化の普及手法
　　オブジェクトと属性、全体対部分、クラスとそ

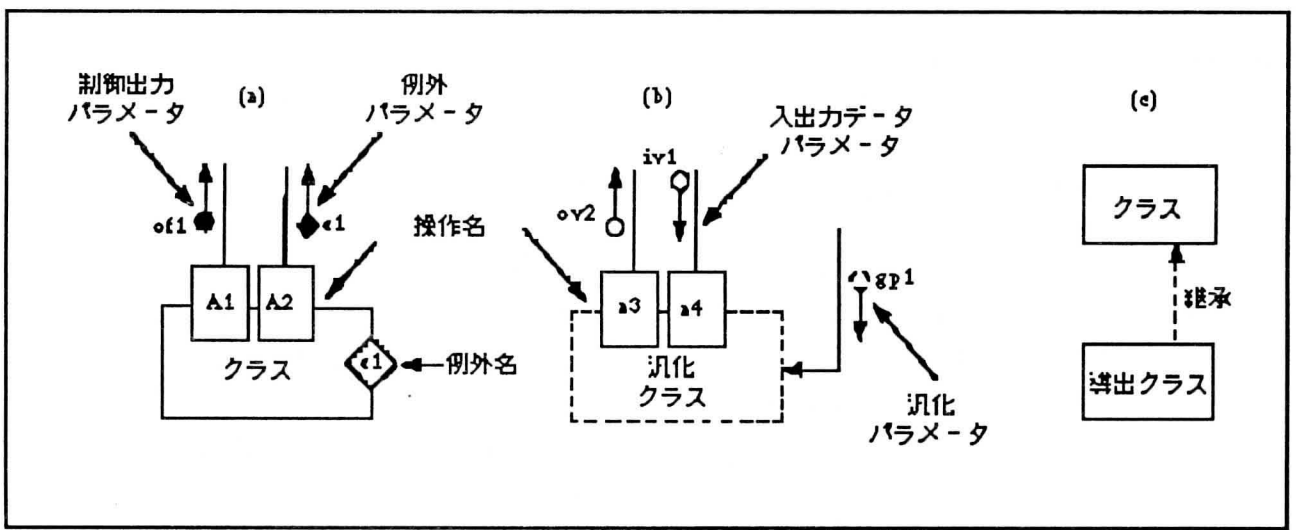

図2.3-1 OOSDの記法（[Wasserman 90]から引用）

のメンバーとそれらの間の区別

（7）スケール

そして、Coad法を構成する概念である

（1）クラスとオブジェクト

（2）一般化-特殊化構造（is-a関係）

（3）全体-部分構造（has-a関係）

（4）属性

（5）サービス（メソッド）

（6）インスタンス接続

（7）メッセージ接続

（8）サブジェクト

サブジェクトはオブジェクトのかたまりであり、スケールの問題を解決する。

で、上の複雑さを軽減する原理を支援していると主張する。

Coad法は、E-RモデルやShlaer／Mellorの方法と、OOPおよび知識ベースシステムの概念を組み合わせて発展させたものである。

Smalltalk, C++, Objective-C, Eiffelなどを直接の対象としている。他の言語の場合や、C++でもオブジェクト指向の機能を使わない場合は、OOA部分はそのまま使えるが、OOD部分はOOSDなどの付加的な方法論が必要になる。

Coad法は図2.4-1の記法で表すオブジェクト図とも呼ぶべき図と、状態遷移図とフローチャートに近いサービスチャートでモデルを構築する。

状態遷移図とサービスチャートは、オブジェクトの内部のサービスの定義に使用する。オブジェクト図で使わ

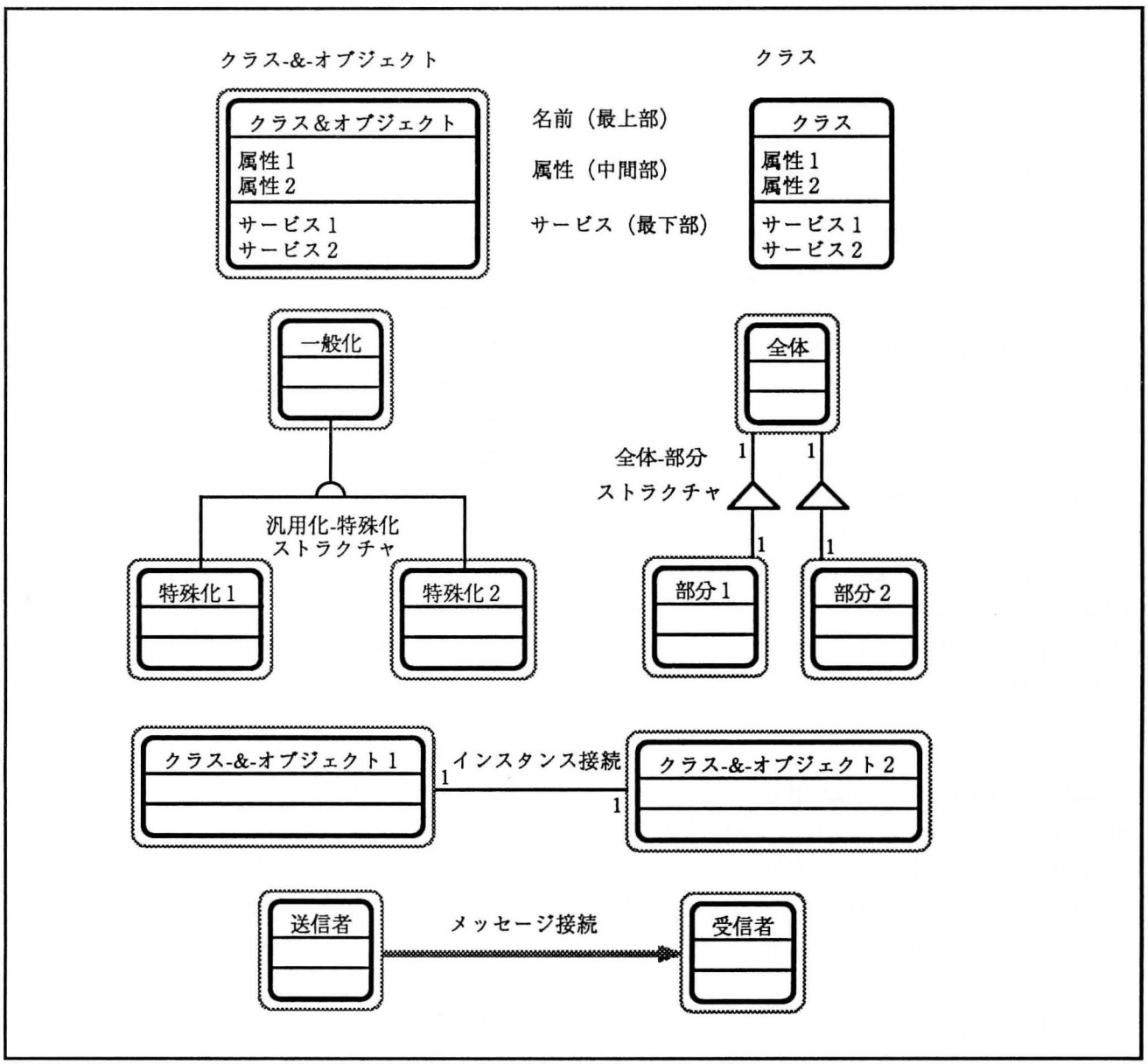

図2.4-1 Coard法の記法（[Object 91]より引用）モデル

れる記号は、2.4.1節で述べた概念をそのまま表している。

Coad法の手順は大きく以下の５つに分かれ、それぞれかなり詳細に標準の手順が決められている。

（1）オブジェクトの認識

どうやって名前を付けるか、どこを捜すか、何を捜すか、選択の基準は何か、不必要なオブジェクトは何か、といった指針がある。

（2）構造の認識

一般化-特殊化構造、全体-部分構造などを認識する。そのためのノウハウも提供されている。

（3）サブジェクトの認識

サブジェクトをいつ、どうやって選び、洗練するかの指針がある。

（4）属性の認識

属性の抽出、属性の配置、インスタンス接続の認識、属性とインスタンス接続の必要性のチェック、属性の制約の記述などの指針がある。

（5）サービスの認識

オブジェクトの状態の認識、状態遷移図の作成、サービスの抽出、メッセージ接続の認識、サービスの記述、サービスチャートの作成、ドキュメント化などの指針がある。

Coad法では、オブジェクトの構造と挙動の両方の分析方法が示されている。オブジェクトのかたまりとしてサブジェクトを置くことによって、概観の把握が容易になっている。また、記法が簡単な割に有用性が高い。

一方、分析方法は割合しっかりしているが、挙動面の分析はまだ弱く、設計の方法論は詳細とは言えない。Smalltalkなど、詳細設計にあまり留意する必要のない言語を使う場合有用であるが、C++, Object Pascalなどでは詳細設計に適した記法と方法論で補強する必要があろう。

2.5 Object Modeling Technique (OMT)

Object Modeling Technique（OMT）はGeneral Electric社の研究開発センターで開発された手法であり、ほとんどの種類のアプリケーションの開発に役だったと主張されている方法論である。現時点で最も完成された技法なので、多少詳しく解説する。

OMTでは、オブジェクト指向を以下のように捉えている。

（1）抽象

（2）カプセル化

（3）データと振舞いの結合

（4）共有

継承はいくつかのサブクラス間のデータと振舞いの共有であり、継承を使ったコードの共有はオブジェクト指向言語の最も有利な点であるが、他にも、将来のプロジェクトとの設計の共有や、再利用可能なライブラリーの構築が容易といった利点がある。

OMTは、SAからデータフローと状態遷移図を引き継ぎ、オブジェクトモデリングはE-R図やデータベース設計技法から引き継いでいる。オブジェクトの認識方法はJSD法から一部影響を受けている。

他のOOA/OOD技法のうち、特にShlaer／Mellorの手法やCoad法とは、オブジェクトモデリングの方法・記法が似ている。

対象とする言語は、ほとんどすべての言語である。オブジェクト指向言語のうちSmalltalk, Eiffel, C++ については、どのようにOMTモデルをプログラムに展開していくかのガイドがある。また、非オブジェクト指向言語のうちC, Ada, Fortranについても、どのようにOMTモデルをプログラムに展開していくかのガイドがある。もちろん非オブジェクト指向言語の場合、言語が支援しない機能はプログラマーの責任になるのだが。

OMTの場合特徴的なことは、OMTモデルをリレーショナルデータベースシステム（SQL言語相当のデータベース照会言語を含む）で実現する際の詳細なガイドがある点である。

OMTモデルは、オブジェクトモデル・動的モデル・機能モデルからなる。オブジェクトモデルは図2.5-1のようなオブジェクト図からなり、オブジェクトやクラスの静的構造を表す。動的モデルは状態遷移図で表し、オブジェクトの状態・動作を記述する。機能モデルはデータフロー図で表し、機能の詳細を表す。

OMTのオブジェクト図が表現できるオブジェクトは、以下のように豊富である。

（1）クラスとオブジェクト

クラスはCoad法と同じように内部を分割した箱で表す。図2.5-1ではPerson, Companyなどがクラスであり、クラスを表す箱の一番上の部分がクラス名、２番目が属性、３番目が操作を表す。

（2）属性

属性はクラスの記号の中央の部分に列記するが、データの型と初期値を記述することもでき

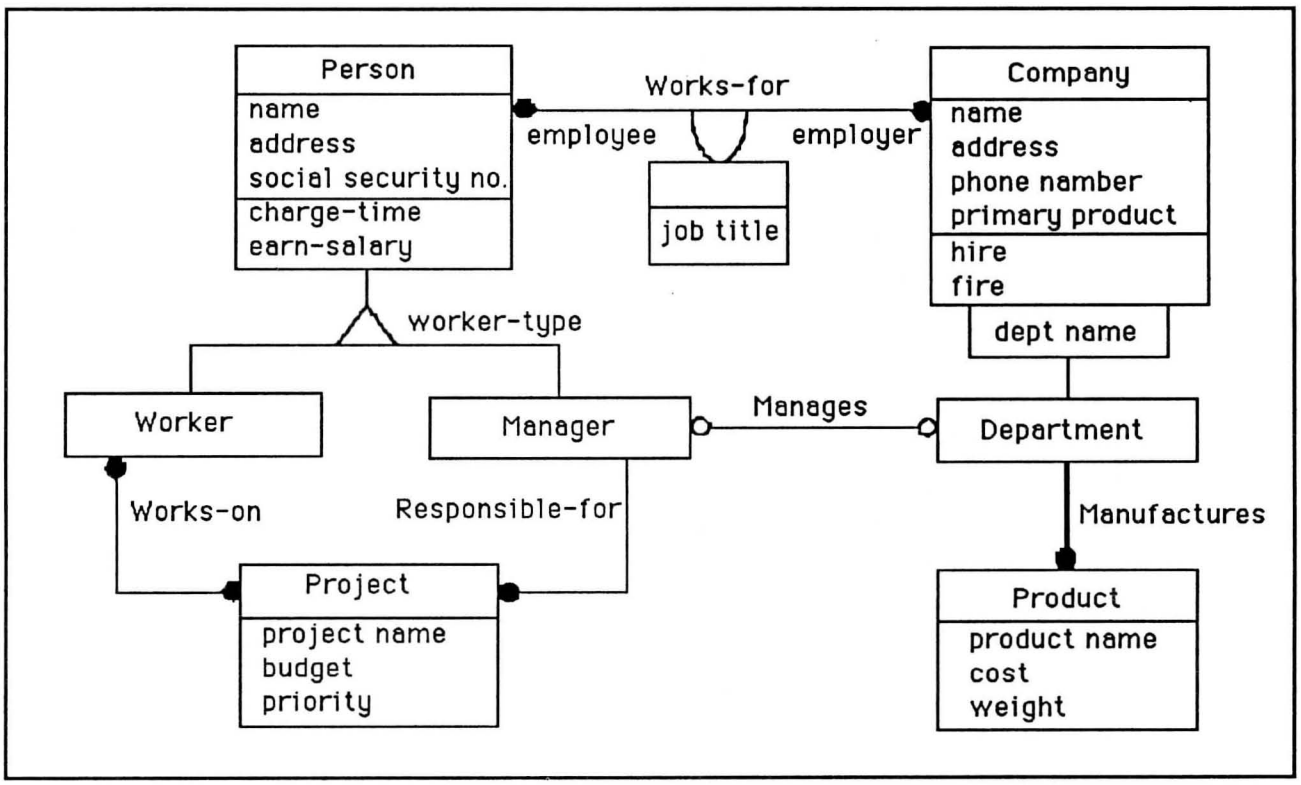

図2.5-1 OMTオブジェクト図の例（[Rumbaugh 90] から引用）

る。

（3）操作（operation）

操作はクラス記号の一番下の部分に列記するが、引数リストと返す値も記述できる。

（4）継承

継承は白抜きの三角形で表される。図2.5-1ではWorkerとManagerが、Personの性質を継承していることが示されている。多重継承を表すこともできる。

（5）結合（association）

結合はクラス間を結ぶ線分で表される。結合名は線分上に表され、役割名が線分の両端に表示される。図2.5-1ではPersonとCompanyを結ぶ結合は名前がWorks-forであり、役割名はPerson側がemployeeで、Company側がemployerである。

結合が1体1であれば線分だけで表されるが、図2.5-1のように両側に●あるいは○が付くと、それぞれ多数（0かそれ以上）あるいはオプション（0か1）であることを示す。図2.5-1では、Managerは1つのDepartmentを管理するか1つも管理せず、Departmentには0または1人のManagerが居て、Manager 1人は0またはそれ以上のプロジェクトに責任があることが記述されている。

条件付き結合（qualified association）は、結合を表す線分の片側に四角で条件子（qualifier）を記述する。図2.5-1では、CompanyとDepartmentがdept nameという条件子で結合されている。

リンク属性は、図2.5-1のWorks-for結合に付いている記号で表される。ここではjob titleというリンク属性が定義されている。リンク属性を拡張してクラスとして記述することもできる。

この他にも、集合体（Coad法の全体-部分構造と同じ）、三角関係（3つのオブジェクトの結合）などを表すことができる。

（6）インスタンス関係

（7）制約

オブジェクトの属性の制約、結合間の制約などを表すことができる。

OMTは以下の手順で分析・設計を行う。

（1）分析

ここではシステムが何をするべきかというモデルを作る。

(a) 問題記述文の最初の版の作成

(b) オブジェクトモデルの構築

オブジェクトのクラスの認識、データ辞

書の作成、クラス間の結合の追加、属性の追加、継承の導入、シナリオに基づくテストの繰り返し、クラスをグループ化してモジュールにする、といったことを行う。

(c) 動的モデルの作成

シナリオの作成、オブジェクト毎のイベントの認識とシナリオによるイベントのトレース、状態遷移図の作成などを行う。

(d) 機能モデルの作成

入出力の認識、データフロー図の作成、各機能の記述などを行う。

(e) 3つのモデルの検証と改良

機能モデルで見つかった主要な操作のオブジェクトモデルへの反映、シナリオによるテストなどを行う。

（2）システム設計

ここでは、システムの上位レベルの構造を決める。

(a) サブシステムへの分割

(b) 問題固有の並行性の認識

(c) サブシステムのプロセッサーとタスクへの割当

(d) データストアの実現方法の選択

(e) 共通リソースの認識とそれらをアクセスする制御機構の決定

(f) ソフトウェア制御の実現方法を選択する

プログラム内に状態を持つか、状態マシンを直接実現するか、並行タスクを使うかを選択する。

(g) 境界条件を考える

(h) トレードオフの優先順位を決める

（3）オブジェクト設計

ここでは、分析モデルをみがき上げ、特定の言語やデータベースシステムへの変換方法に依存しない、実現のための詳細を設計する。

(a) 他のモデルで見つけた操作のオブジェクトモデルへの反映

機能モデルのプロセスから操作を見つけるか、動的モデルのイベントから操作を見つける。

(b) 操作を実現するためのアルゴリズムの設計

操作のコストを最小にするアルゴリズムを選択する、アルゴリズムにあったデータ構造を選択する、必要であれば新たなクラスと操作を追加する、などを行う。

(c) データへアクセスするパスの最適化

アクセスコストを最小化し便利さを最大化するため冗長な結合を追加する、効率化のため計算順序を変える、複雑な計算をやり直さないため計算結果を保存する、などを行う。

(d) ソフトウェア制御を実現する。

(e) クラス構造を吟味して、継承を増やす。

継承を増やすためクラスと操作を再配置する、抽象クラスを抽出する、継承が意味的におかしいとき委譲（delegation）を使う、などを行う。

(f) 結合の実現方法を設計する。

結合を1方向だけ実現する、結合をオブジェクトとして実現する、結合の両側のクラスへ属性として追加することによって実現する、などを行う。

(g) オブジェクトの属性の表現方法を決める。

(h) クラスと結合をモジュールにまとめる。

（4）実現

プログラミング言語で実現するか、データベースシステムで実現するか、コンピュータ以外で実現するかを決める。次に、クラス・操作・結合を対応する各言語毎に指針にしたがって展開する。

OMTの特徴は、分析から設計まで一貫したオブジェクトモデルで記述でき、分析手順と設計手順と実現手順とが、かなり詳細に分かりやすく用意されていることである。オブジェクトモデルはBooch法よりは簡単で、Coad法より記述能力が高く、どちらの方法よりも結合に関しての記述力が高い。

OMTで使用するモデルは言語独立性が高く、利用可能な分野が広い。実際に種々の言語で、いろいろな分野のアプリケーションを作成した経験が感じられる。また、従来のSA/SD手法のE-R図をオブジェクト図に変えればOMTになるので、SA/SDからOMTに移る障害が少ない。

現時点で、もっとも完成されたOOA/OODの方法論である。

あえて問題点を捜せば、既存のライブラリーとオブジェクト指向のライブラリーを混合した場合、Booch法やOOSDのようにそれを直接的に表すことができない点だろう。既存のシステムを分析する場合にも、モジュールの呼び出し関係などを直接的に表せない。

3.0 CASEツール

オブジェクト指向を支援するCASEツールはまだ数が少なく、SA/SDを支援するCASEツールほど完成度は高くないが、ここ1年ほどの間に急速に増えている。ここではそれらのCASEツールを、方法論を支援するCASEツール、方法論とはあまり関係なくより言語に密着したLower CASE、オブジェクト指向のパラダイムでユーザーインタフェース構築を支援するCASEツール、オブジェクト指向ビジュアルプログラミングツールに分けて説明する。

3.1 方法論支援CASE

方法論を支援するCASEツールで完成しているものは少なく、開発中あるいはプロトタイプは完成しているが製品バージョンは開発中のものが多い。

3.1.1 OOSD

SA/SDを支援するCASE環境Software through Pictures [Wasserman 87] の上で動く、Interactive Development Environments社のOOSD支援ツールである。プロトタイプは完成しているが製品は開発中である。UNIXまたはVMS上で稼働し、Ada, C++, Eiffelの開発を支援する。

主な機能は以下のとおり。

（1）整合性チェック機能

名前の整合性、モジュールや操作のパラメータの整合性チェック、図上の各記号の接続の正当性チェックなどを行う。

（2）コード生成機能

OOSDの図からコードを生成する。言語毎の生成ルールは、データ辞書を通して設計チェックルール・生成ルーチンと統合化されていて、新しい言語に対応させることは比較的簡単である。また、生成されたコードはテキストファイルであり、パラメータファイルに指定されたコマンド（例えば文法チェッカーやコンパイラー）に渡すことができる。

（3）再利用機能

再利用ライブラリーとして検証された仕様とコードをデータ辞書に入れることができ、検索し複写し修正することができる。

（4）分かりやすさ支援機能

大規模なシステムの設計に利用できるよう、設計を階層化し、不必要な詳細を隠すことができる。

OOSDはSA/SDツールと連係しているため、分析フェーズではE-R図を使わなければならない。このため、ERE++というオブジェクト指向に対応したE-R図の拡張が計画されているが、詳細は不明である。

基本的に、オブジェクト指向の考え方を、SAのE-R図やデータフロー図でシミュレートし、モジュール設計段階からOOSDを使う形になるので、既存のCのライブラリーなどを活かし、一部C++などで開発する場合に有用であろう。

ユーザーインタフェースは、SA/SD機能との整合性を取るため、オブジェクト指向でなく機能指向なので、他のOO-CASEツールよりは使いにくい。

自身の開発にOOSDを使用しているかどうかは不明。

3.1.2 OOATool

Object International社の、Coad法を支援するツールである。Macintosh上で稼働する。支援する開発言語は特に限定されていないが、Smalltalkが一番向いている。図2.4-1はOOATool の出力例でもある。

Coad法を全面的に支援しているが、CASEツールとして付加された主な機能は、以下のとおり。

（1）評論機能

作成されたモデルを評価し、方法論の指針に反する部分や未定義部分を指摘する。

（2）フィルター機能

一つのモデルに複数の図を対応させることができ、各図はお互いの修正を反映することができる。さらに、フィルター機能によってモデルが見える部分を制限することができる。

（3）ドキュメント支援機能

オブジェクト毎に、ドキュメントのテンプレートを定義し、ドキュメントを作成することができる。

（4）戦略カード機能

Coad法の方法論を工程別に表示する。この工程や内容をユーザーが変えることも可能である。

OOATool はCoad法全体を支援するツールであり、Smalltalk/V[Digitalk 88]で開発されているため、Smalltalk/Vのエディターや環境を流用して高機能なツールとなっている。ただし、Smalltalk/Vの環境にはアクセスできない。

ユーザーインタフェースは良い方であり、オブジェクト指向ユーザーインタフェースを採用しているが、Macintoshのアプリケーションとしては良くない部類に入る。またツール実行中、しばしば（おそらく操作を間違えたときであるが）「Message not found」メッセージを出してツールから抜け出すことができなくなる。

自身の開発にCoad法を適用しているか不明。

3.1.3 ObjectCast

Objectworks/Smalltalkと後述する（3.2.1節）Blizzard-90[富士ゼロックス91]上で稼働する、富士ゼロックス情報システム社のCoad法を支援するツールである。現在プロトタイプが完成しているが、OMTに対応したツールも開発中である。図3.1.3-1にObjectCastのオブジェクト図の例を示す。

支援する開発言語は、オブジェクト指向言語全般であるが、現在はSmalltalk支援機能が一番豊富である。

Coad法を全面的に支援するとともに、各種技法を利用して、その拡張も行っている。以下に主な機能を説明する。

（1）マネージャー機能

Coad法のサブジェクトやオブジェクトをまとめる概念として「仕様書」を導入し、全体関係を把握し、オブジェクト図エディター（グラファーと呼ぶ）や次に述べるインベスティゲーター機能を起動するためのものである。

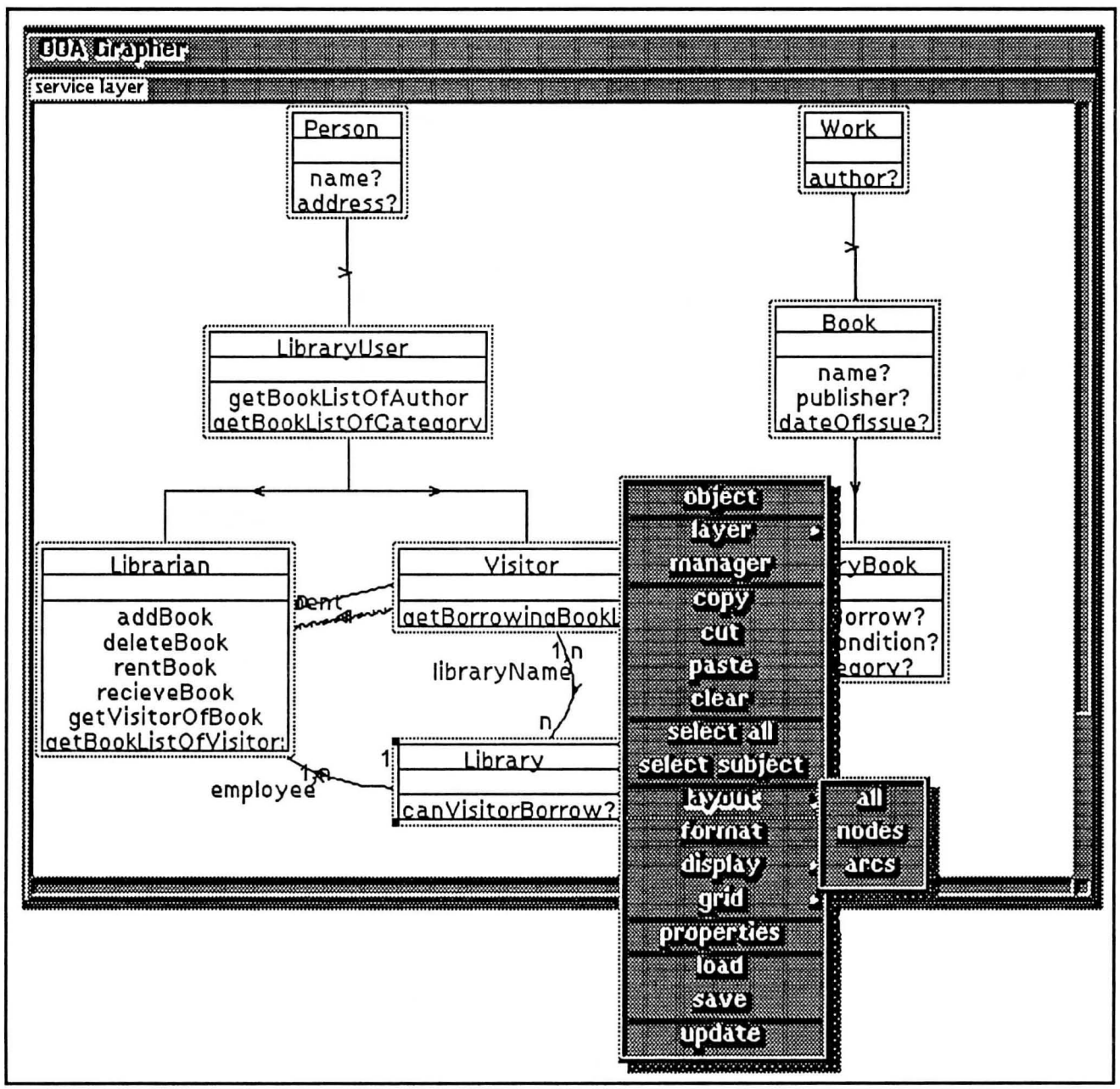

図3.1.3-1 ObjectCastのオブジェクト図の例

（2）インベスティゲーター機能

ObjectCastで扱うオブジェクトすべてについて、その情報を記述したり参照したりするための機能である。サービスインベスティゲーターには、入力条件・出力条件の記述機能もある。

（3）ハイパーテキスト機能[Conklin 87]

インベスティゲーター機能の注釈記述部分から、任意のSmalltalkの式を記述し、それを評価することによって、Smalltalk, Blizzard-90, ObjectCastの任意の機能を実行できる。

（4）状態遷移図

（5）ペトリネット図[Peterson 81J]

OOD段階で、タイミングを考察するツールとして導入された。

（6）データーフロー図

（7）フローチャート図

Coad法のサービスチャートは、実質的にフローチャートなので、サービスの詳細を記述するために導入された。

（8）コード生成機能

Smalltalkコードのテンプレート生成機能である。

ObjectCastはCoad法を支援するツールであるが、実際にOOA/OODを実践する中で、必要な図やツールが拡張されている。状態遷移図とデータフロー図を導入したことにより、記法を変えればOMTも支援できるため、OMTにも対応できるよう改造中である。

ObjectCastはObjectworks/SmalltalkとBlizzard-90上で稼働し、それらの環境も同時に利用できるため、OOA/OOD/OOPを一貫して支援するきわめて強力な環境になっている。

ユーザーインタフェースは基本的には良いが、各ツール間で多少の不整合が見られる。また整合性チェックのための機構は、現在開発中である。

自身の開発にCoad法を使用している。

3.1.4 その他

この他のツールとしては、以下のものがあるが、詳細は不明。

（1）Mack V System社のObjectMaker（多分開発中）

Booch法のツールで、Ada, C++, Cのコード生成とリバースエンジニアリングを行う。MS-Windos, X11, Mac上で稼働する。

（2）Easyspec社のObjectPlus

詳細不明。

（3）Cadre社のTeamWork（開発中）

バージョン4.0でShlaer／Mellorの方法を支援するE-R図と、Booch法のAda構造グラフを出す予定。UNIX 上で稼働。

（4）IPSYS社のCASET

Coad法を支援する。

（5）Popkin Software社のSystem Architect

E-R図と、AdaとC++のためのBooch法、Coad法を支援する。図の整合性とルール違反をチェックする。MS-Window上で稼働。

（6）GE社のOMTool

OMTを支援するCASEツールである。

方法論を支援するツールは、全体として図エディターのレベルにとどまっており、データ辞書を利用した整合性のチェック、ハイパーテキスト機能などは一部しか実現されていない。

3.2 Lower CASE

Lower CASEはソフトウェア開発の下流工程（詳細設計、プログラミング、デバッグ、保守の一部）を支援する。その性格上、方法論よりは言語に依存したものが多い。ここではその代表的なものを紹介する。

3.2.1 Objectworks/Smalltalk Release 4.0 & Blizzard-90

Objectworks/SmalltalkはParcPlace Systems社のSmalltalk支援環境であり、Blizzard-90は、Objectworks/Smalltalk上に構築された富士ゼロックス情報システム社のSmalltalk支援環境である。Blizzard-90は、本体のObjectworks/Smalltalkよりも多くのクラスとステップ数からなる巨大な支援環境と再利用可能なオブジェクトライブラリーの集合である。ユーザーインタフェースは、Objectworks/Smalltalkオリジナルのものも使用できるが、普通はすべてBlizzard-90流に改造されている。図3.2.1-1にBlizzard-90のブラウザーを示す。

Blizzard-90（Objectworks/Smalltalkの機能も含む）の主要な機能は以下のとおりである。

（1）新ユーザーインタフェース

従来のSmalltalkのユーザーインタフェースに、多数の新機能が追加された。

（2）アウトライン機能

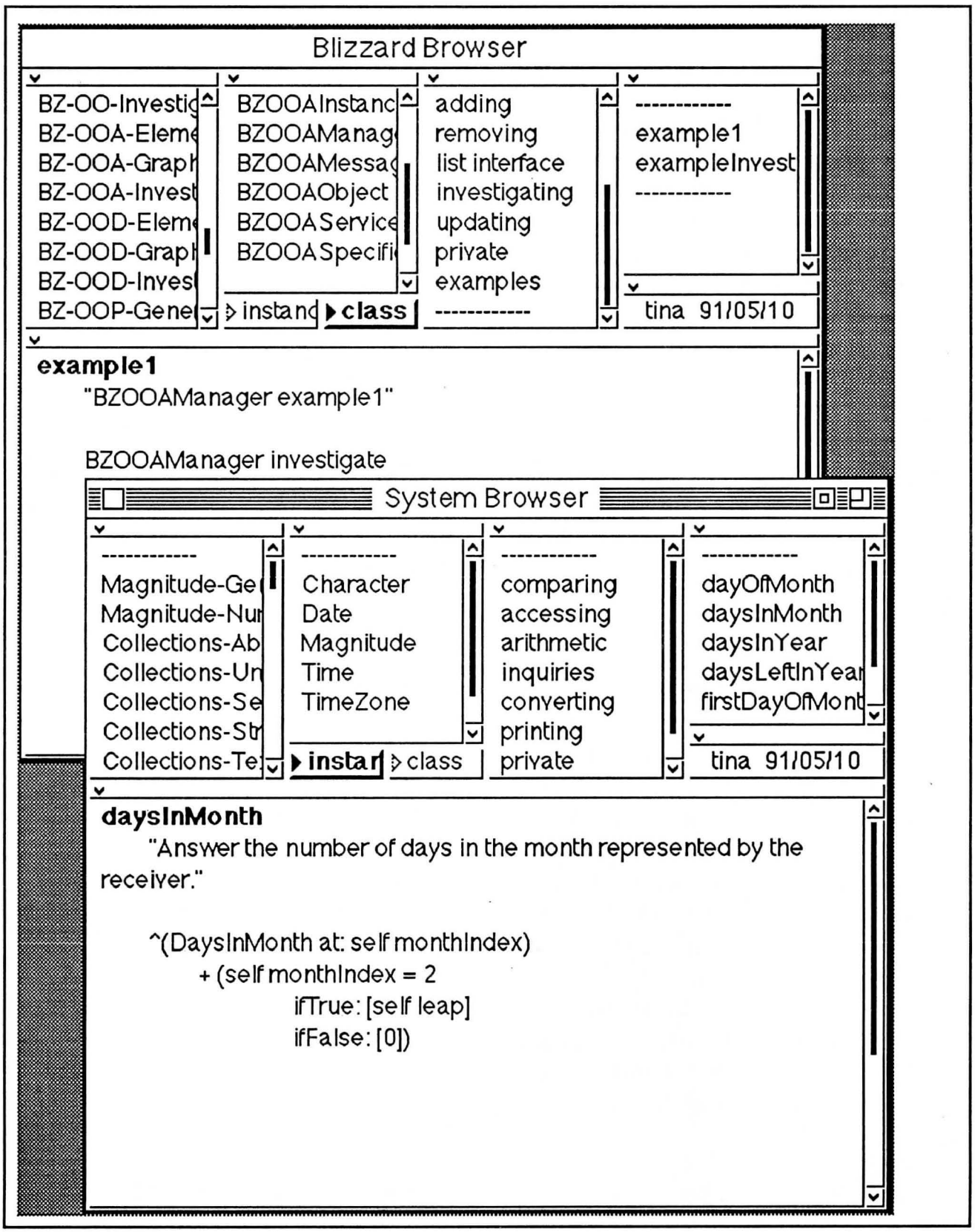

図3.2.1-1 Blizzard-90のブラウザー

ドキュメント作成や情報の整理になどに便利なアウトラインリストビューが追加された。

（3）各種の新ブラウザー

Blizzardブラウザー（図3.2.1-1）、クラス木ブラウザー、アウトラインブラウザー、アウトラインカテゴリーブラウザー、クラス階層ブラウザー

など、従来のSmalltalkより強力な多数のブラウザーがある。

（4）グラファ機能

図エディターの基本機能を持っていて、この機能を利用してクラス木ブラウザーや前述のObjectCastが実現されている。ノードやアークの整列機能も持っている。

（5）移植性

移植性のあるSmalltalkコードの作成が可能になった。

（6）動的コンパイル機能

（7）シンボリックデバッガー

（8）変更管理機能

各クラスおよびメソッド毎に変更履歴を管理できる。

（9）効率測定機能

（10）設計評価機能

特殊化-抽象化因子（HF）、組み立て-要素分解因子（RF）、多相化因子（PF）を計算して、ある程度の設計の評価ができる。

HF_A = クラスAのスーパークラス数 / (クラスAのスーパークラス数 + クラスAのサブクラス数 + 1)

RF_A = クラスAのトポロジカルソート順位 / 全クラス数

PF_A = システム内でユニークなクラスAのメソッド数 / システム全体のメソッドの種類の数

（11）ランタイムアプリケーション開発機能

必要のないクラスや開発環境を切り離してアプリケーションプログラムを作成することができる。

（12）3次元ライブラリー

CASEツールなどで利用するための3次元ライブラリーである。

（13）リレーショナルデータベース接続機能

オブジェクトやその他の情報を格納するための機能である。

全体として、Smalltalkの開発・保守支援環境として十分すぎるほどの機能を持っているが、チームによる協同作業を支援する機能が弱い。

3.2.2 Objectworks/C++

Objectworks/C++はParcPlace Systems社のC++2.0支援環境である。基本的に、Objectworks/Smalltalk的な支援環境をC++に与えるものである。Sunワークステーション上で稼働する。

主な機能は以下のとおり。

（1）ファイルシステムブラウザー機能

（2）ソースコードブラウザー機能

継承構造・クラス・関数の検索とアクセス、コンパイル、などを実現している。

（3）ソースコードデバッグ機能

（4）構成管理機能

（5）UNIXインターフェース機能

UNIXのファイルシステムへのアクセスや、UNIX端末機能、UNIXコマンドへのアクセス機能がある。

全体として、C++の開発・保守支援環境として強力な機能を持っているが、チームによる協同作業を支援する機能が弱い。

3.2.3 SoftBench

HP社のSoftBenchは、HP-UX上で稼働し、Fortran, C, C++.の開発を支援する。主な機能は以下のとおり。

（1）ブラウザー機能

継承構造・クラス・関数の検索とアクセス、コンパイルなどを実現している。

（2）構文検査と自動修正機能

言語規則と形式のガイドラインに反した構文を検査し、自動修正する。

（3）テンプレート生成機能

（4）プログラムデバッグ機能

（5）版管理と構成管理機能

（6）静的解析機能

クラス・メンバー関数・変数・関数の相互関係を解析し、照会することができる。

（7）カプセル化機能

Encapsulator機能と呼ばれ、Encapsulator記述言語を使用して、UNIXのコマンド行インタフェースを持つツールを、ツールのソースコードを変更することなく、SoftBenchに組み込むことができる。

全体として、大規模なシステムの開発を前提にし、UNIX上の既存ツールや他社製のツールを自らの環境内に取り込む機能を持った、Fortran, C, C++の開発環境であ

る。カプセル化機能を使って、OOSDなどOO-CASEツールを取り込む動きがある。

3.3 ユーザーインタフェース構築支援

OOA/OODだけでは、ユーザーインタフェースが有用なシステムを構築するのは困難である。OOA/OODによるモデル化と並行して、ユーザーインタフェースをあらかじめ設計しチェックする必要がある。このため、ユーザーインタフェース設計用のプロトタイピングツールなどが作られてきた。しかし、ここにきてプロトタイプだけでなく最終製品に拡張していくことも可能で、オブジェクト指向パラダイムにも対応したCASEツールが出現してきた。ここでは、その中でも高機能なWidgets/Vについて述べる。

3.3.1 Widgets/V

Widgets/VはAcumen Software社のユーザーインタフェース構築ツールであり、Macintosh上のSmalltalk/V環境に組み込まれて動く。Widgets/Vで作成されたユーザーインタフェースは、そのままアプリケーションプログラムへ発展させていくことができる。このためWidget/Vは、MVCモデルと対応するSmalltalk/VのMPDモデルは分かりにくいとして書き換え、ViewとControllerを一緒にしたWidgetsクラスを使用する。

主な機能は以下のとおりである。

（1）インタフェースエディター

Widgets/Vの中心機能であり、Macintoshのウィンドウやポップアップメニューやボタンなどからなるユーザーインタフェースを編集する。図3.3.1-1に画面例を示す。

このエディター上でユーザーは自由にユーザーインタフェース関連のオブジェクト（ボタン・フィールド・メニュー・画面など)を編集することができる。

（2）テスト機能

図3.3.1-1の左上のTest Itボタンをクリックすることにより、編集中のユーザーインタフェースを実行してテストしてみることができる。

（3）コード生成機能

インタフェースエディタから、ユーザーインタ

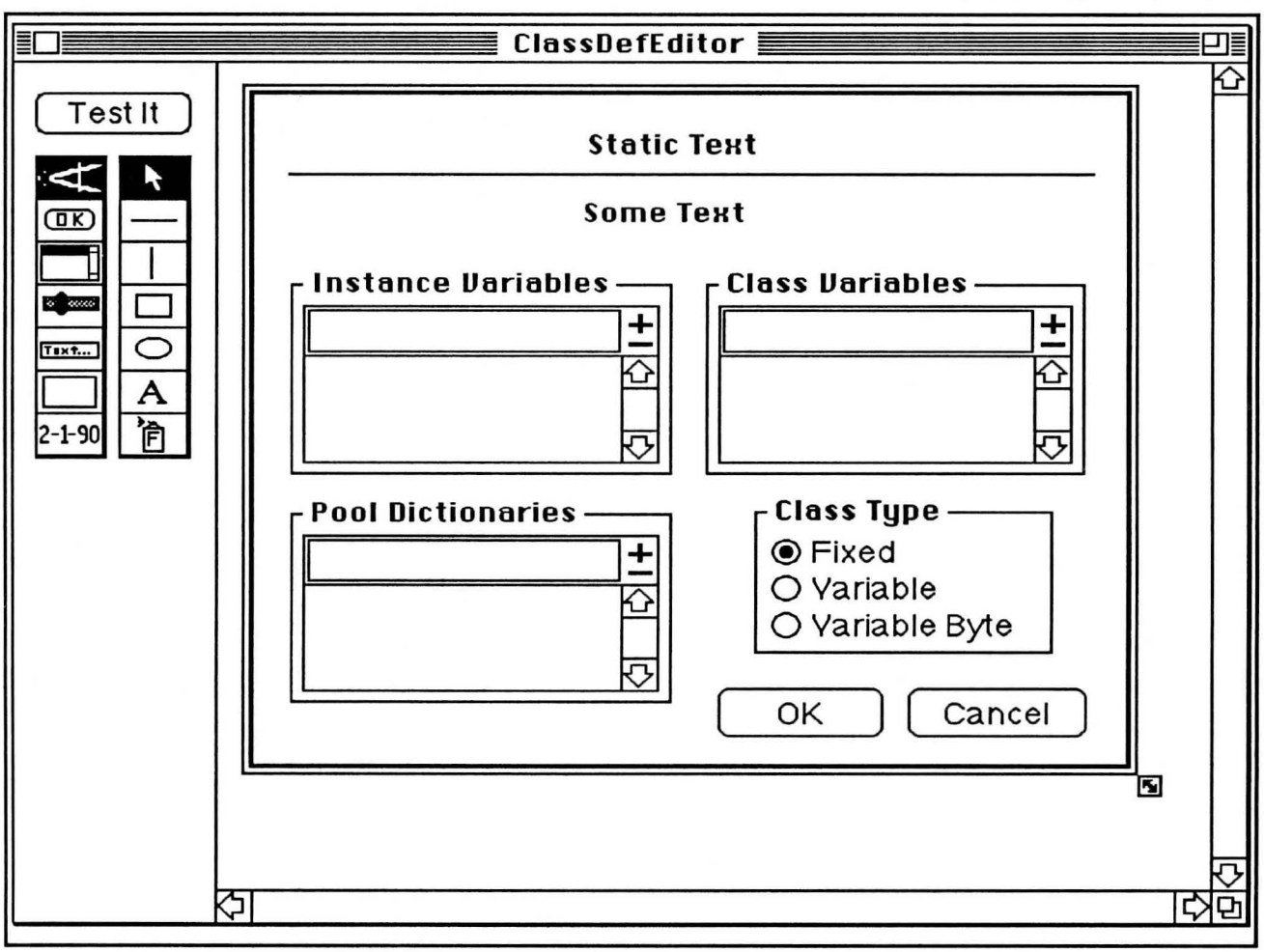

図3.3.1-1 Widgets/Vインタフェースエディター画面例

フェースに対応したSmalltalkコードを生成させ、ユーザーインタフェースの編集情報と一緒にライブラリーとして格納し、後で再利用することができる。

Widgets/Vはユーザーインタフェースを編集し、それをテストし、Smalltalkコードを生成し、それを再利用でき、ユーザーインタフェース以外のコードをSmalltalk/Vのブラウザーから作成することによって、最終的な製品まで作ることができる。

ユーザーインタフェース自体もオブジェクト指向になっており、分かりやすい。また、ベースにあるSmalltalk/V環境とも融合している。

現時点で考えられる、最良のオブジェクト指向ユーザーインタフェース構築支援CASEツールと言って良いだろう。

3.3.2 その他

この他のツールとしては、以下のものがある。

（1）Action

ExperTelligence社の、CLOS上のユーザーインタフェース構築ツールであり、Macintosh, PS/2, SUN SPARCstationなどのCLOSシステム上で稼働する。 基本的な発想と機能はWidgets/Vと同じである。

（2）ViewEdit

Apple社のアプリケーション用のユーザーインタフェース構築ツールである。Macintosh上で稼働する。Widgets/V とActionとは異なり、そのままアプリケーションへ発展させることはできず、ユーザーインタフェース構築リソースを作成するにとどまる。

（3）Interface Builder

NeXT社のアプリケーション用のユーザーインタフェース構築ツールである。NeXTワークステーション上で稼働する。ViewEditと同等の機能である。

（4）HyperCard2.0

Apple社のハイパーテキスト・ハイパーメディアを支援するオーサリングツールであるが、HyperTalk2.0[Winkler 90] というオブジェクト指向言語をベースに持ち、限定された機能ではあるがユーザーインタフェースの構築や簡単なアプリケーションの作成ができる。

3.4 ビジュアルプログラミングツール

従来のビジュアルプログラミングツール[萩谷 91]は、フローチャートや構造化チャートをビジュアル化したものが多かった。ここではOOD/OOPを支援するビジュアルプログラミングツールを紹介する。

3.4.1 Prograph

Prograhは、Macintosh上で稼働するThe Gunakara Sun Systems社のオブジェクト指向ビジュアルプログラミングツールである。

Prographはオブジェクト指向ビジュアルプログラミング環境であり、図形言語であり、アプリケーション生成ツールである。アプリケーションプログラムの作成までを、すべてPrograph内で行うことができる。カバーする開発工程は、 Objectworks /C++, SoftBenchなどとほぼ同じである。

主要な機能は以下のとおり。

（1）ブラウザー機能

クラスブラウザー、属性ブラウザー、メソッドブラウザーなどがあり、継承の記述、属性の記述、メソッドの記述が容易にできる。図3.4.1-1はクラスブラウザーの例である。

（2）図形言語編集機能

図3.4.1-2のような図形言語を編集し実行する機能がある。この図形言語は、オブジェクト内部のデータフローを記述していると思ってよい。

（3）図形言語インタープリト・デバッグ・コンパイル機能

図形言語編集ウィンドウから、言語のインタープリト、シンボリックデバッグ、コンパイルを行うことができる。

かなり分かりやすく強力なオブジェクト指向図形言語とその開発環境である。これからのプログラミングのひとつの方向を示していると思うが、専門家がObjectworks /C++, SoftBenchなどで開発するのに比べ、生産性が良いとは言えないだろう。ただ学習時間ははるかに少ないと思われるので、プログラマーの範囲を広げる効果はあるだろう。

4.0 問題点

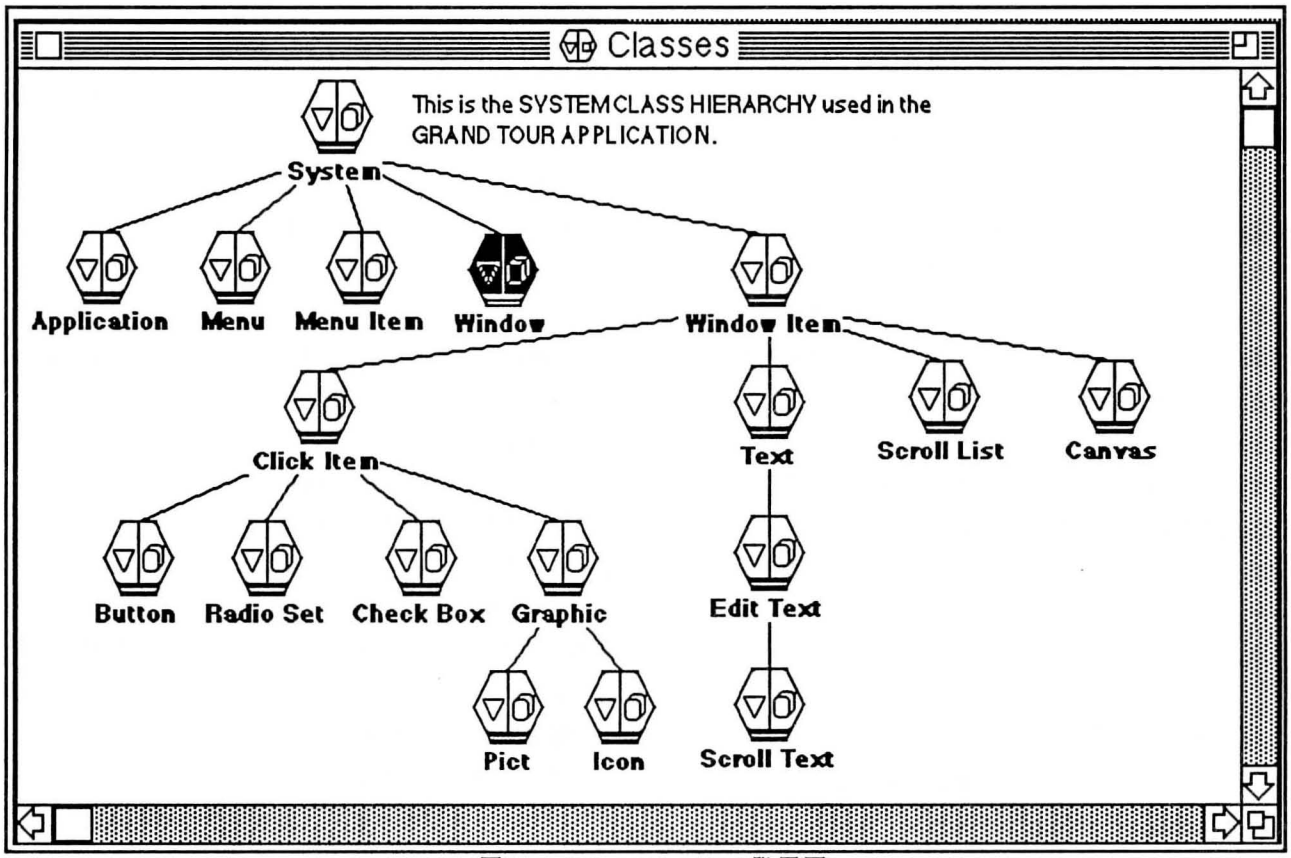

図3.4.1-1 Prographクラス階層図

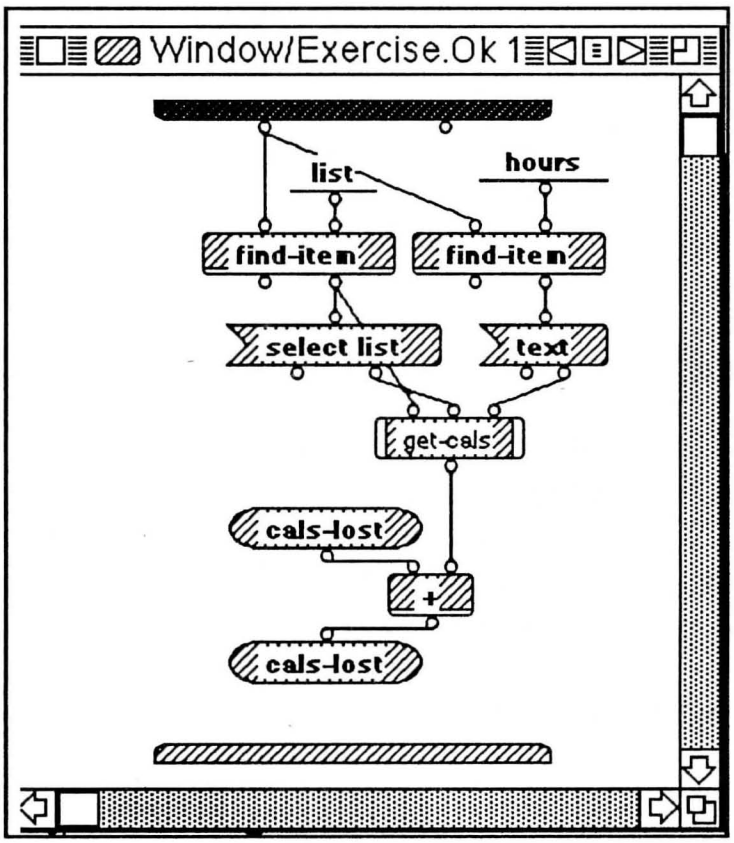

図3.4.1-2 Prograph図形言語

ここまでに見てきたOOA/OOD手法やそれを支援するCASEツールには、まだ多くの問題点がある。以下に主なものを示す。

4.1 手法と記法の混乱

現時点のOOA/OOD手法は、その概念にかなりの類似性があるにもかかわらず、その手法と記法の差が大きい。実際に、Booch法とCoad法とOMTとを比較する際、どの記号がどの記号に対応するかを調べるだけでかなりの労力が必要であった。

また、Booch法やOOSDのように、SA/SDとの一貫性を主張する方法論はSA/SDの記号とOODの記号を混合させているが、記号の数が多くなりすぎて分かりにくく、より詳細の設計に偏りすぎている感がある。整理が必要であろう。

手法とそれを支援するCASEツールのいずれも、プロトタイピングやユーザーインタフェース構築の手法を組み込んでいない。実際にはOOA/OODモデルだけでシステムを見通すことは難しく、プロトタイプやユーザーインタフェース構築とモデル化が並行して必要なことを考えれば、手法の拡張が望まれる。

4.2 オブジェクト・リポジトリーの欠如

SA/SDを支援するCASEツールでデータ辞書による支援がないものは稀であるが、OO-CASEでSA/SDでのデータ辞書にあたるオブジェクト・リポジトリーによる支援があるものは調査した範囲ではない。このため、モデルの整合性チェック機能や、モデル内のオブジェクトの検索機能が弱く、大きなシステムへの適用にはまだ困難がある。

4.3 フォーマル仕様の欠如

OMTでは意味的制約の記述ができ、Booch法でも前条件・後条件の設定はできるが、いずれもフォーマルな仕様記述言語は規定されていないため、対応するCASEツールで意味的エラーのチェックができない。このあたりの事情はSA/SDを支援するCASEでも同じだが、改善が望まれる。[Redmond-Pyle 91]で、フォーマル仕様をCoad法などの図表現と融合させる議論が行われている。

5.0 今後の方向

ここでは、OOA/OODの方法論とOO-CASEの進む方向を考えてみる。

5.1 モデル

OOA/OODのモデルとしては、OMTのようにオブジェクトモデル・動的モデル・機能モデルの３つでシステムを捉えていくのが最も自然だろう。Booch法の６つの図も、クラス図・オブジェクト図・モジュール図をオブジェクトモデル、状態遷移図・タイミング図を動的モデル、プロセス図を機能モデルと考えれば、この捉え方に一致する。Coad法も機能モデルが欠けている点以外はOMTにかなり類似している。

対象システムをオブジェクトモデル・動的モデル・機能モデルの３つで表す考え方は、結局SA/SDの情報モデル・動的モデル・機能モデルのうちの情報モデルをオブジェクトモデルに発展させただけであり、SA/SDとの概念的整合性もとれる。

現時点では、方法論としてOMTが最も精密であり合理的であるが、BoochもCoadも方法論の改善を図っており、今年中にも逆転があるかもしれない。いずれにせよ、この３つを中心としてオブジェクト指向のモデルと開発方法論が展開して行くであろう。

5.2 標準化

SA/SDを支援するCASEの発展は、SA/SDの記法の統一性によるところが大きい。オブジェクト指向の場合、各技法の記法の標準化はまだ無理だろうが、いずれ標準化を図らなければ、CASEツールの発展も制限されるだろう。

5.3 CASEに必要な機能

ここでは、OO-CASEで今後必要とされる機能を考える。

5.3.1 ユーザーインタフェース

現在のOO-CASE、特に方法論を支援するツールは、ユーザーインタフェースがまだ不十分である。特に以下の機能が望まれる。

（１） 手法のガイド

SA/SDでも明らかになったことだが、背後にある方法論を理解しないとCASEツールの効果はほとんどない。また、方法論の本を１冊読んだだけでCASEツールを使いこなす人間はそう多くない。従って、OOAToolにある戦略カード機能と評論機能のように、方法論をガイドする機能とチェックする機能が一体に組み込まれている必要がある。

OOAToolの場合、戦略カード機能は（変更は可能だが）単にテキストを表示するだけだが、今後はハイパーテキスト機能との連動が必要であろう。

（２） ハイパーテキスト

OOA/OODは基本的にボトムアップの方法論なので、ユーザーは情報の洪水の中を泳ぎまわることになる。OMTのモジュールやCoad法のサブジェクトといった段階的詳細化のための機能だけでは、適切な情報にたどり着くことはむずかしい。そこで、ハイパーテキスト機能で情報間の関連を保持する必要が出てくる。

ObjectCastでは、任意のオブジェクトの注釈にハイパーテキスト機能を持たせていて、しかもユーザー定義可能になっていた。このような機能がOO-CASEの任意の場所で提供される必要があろう。

5.3.2 自己記述性

SA/SDは、自身の開発手順をモデル化することができ、その能力をある程度証明できた。同様にOOA/OOD

は、自分自身のモデルと開発手順を記述することによって、その能力を証明できなければならない。

ObjectCastは、その開発にあたって各図のツール毎にCoad法を使用しているが、全体を統括したモデルはまだない。OMTは、オブジェクト図エディターに自分自身を適用してモデルを作成したようであるが、やはり全体を統括したモデルはない。

OO-CASEは（たとえSmalltalkを使用したとしても）巨大なツールになる。自分自身の分析と設計に、自分自身のモデル技法と手法を当てはめるのが何よりも大事であろう。

5.3 CSCW

OOSD, SoftBenchなどの下流工程を支援するOO-CASEは、ある程度、多人数による協同開発作業を支援してくれるが、OOATool, ObjectCastなどはまだ多人数による開発の支援機能が不十分である。いずれの場合も、CSCW（Computer-Supported Cooperative Work）[石井 91]的な、高度な協同作業支援機能を用意しているわけではない。

しかし、OO-CASEが対象にする主なタゲーットは、複雑なシステムを複数の人間で分析し設計していく、大規模システムの開発・保守であろう。この場合、複数の人間が、複数の版のモデルや仕様を協同で検討していくCSCW的な仕掛けが必須となろう。

6.0 おわりに

本稿では、OOA/OODの主な手法とOO-CASEを調査し、その概要と問題点を提示した。そこで明らかになったことのひとつは、SA/SDはもはやOOA/OODの一部として包含されてしまったということである。特にOMTはうまくSA/SDを取り込んでいる。もうひとつ明らかになったことは、SA/SDを支援するCASEツールに比べ、OO-CASEはまだ統合的なツールとしての機能が完成されていないことである。もっとも、Smalltalkをベースにして開発されたOOAToolやObjectCastの開発スピードを見ていると、その差も早晩解消されるだろう。それまででも、SA/SDを支援するCASEを使ってOOA/OODを実行してみる価値は十分にあるだろう。

なお、多くの手法やCASEツールが開発中であり、本稿で対象とした手法やCASEツールも調査の直前に発表されたものが多く、調査はまだ十分とは言えない。今後も調査を継続していかなければならないと考えている。

謝辞

同僚の酒匂寛氏には、[Rumbaugh 91] の重要性を指摘していただいたほか、種々の文献の入手に協力頂いた。青木淳氏(富士ゼロックス情報システム)には、各方法論の評価とCASEツールの入手に協力頂いた。土屋旨夫氏(横河ヒューレットパッカード)には、C++/SoftBench関連の文献の入手に協力していただいた。また同僚の佐藤圭氏には、Prographの入手に協力していただいた。いくつかのソフトウェアやOO-CASEの情報は、WIDEネットワーク[村井 90]を通して入手した。この場を借りて感謝したい。

参考文献

参照のしやすさを考えて、翻訳があるものは翻訳本の方を入れた。

[Acumen 91] Acumen Software. Widgets/V Mac Tutorial and Programming Guide. Acumen Software, 1991

[Apple 90] Apple Computer. MacApp 2.0 Tutorial. Apple Computer, 1990

[Apple 91] Apple Computer. Macintosh Programmer's Workshop Object Pascal v.3.1. Apple Computer, 1991

[Booch 83] Grady Booch. Software Engineering with Ada. Benjamin/Cummings, 1983

[Booch 90] Grady Booch. Object-Oriented Design With Applications. Benjamin/Cummings, 1990

[Chen 76] P.P. Chen. The Entity-Relationship Model-Toward a Unified View of Data. ACM Tranzaction on Database Systems. Vol. 1, No.1, pp. 9-36, March 1976

[Coad 91] Peter Coad, Edward Yourdon. Objet-Oriented Analysis Second Edition. Prentice Hall International, 1991

[Conklin 87] Jeff Conklin. Hypertext: An Introduction and Survey. IEEE Computer, Vol. 20, No.9, pp.17-41, 1987

[Cox 86J] Brad J Cox. 前川 守 監訳. オブジェクト指向のプログラミング ソフトウェア開発技法の進化. トッパン, 1988

[DeMarco 78J] Tom DeMarco. 高梨, 黒田 訳. 構造化分析とシステム仕様. 日経BP, 1986

[Digitalk 88] Digitalk Inc. Smalltalk/V Mac Tutorial and Programming Handbook. Digitalk Inc., 1988

[DoD 83] U.S. Dept. of Defense. The Ada Programming Language Reference Manual. ANSI/MIL-STD-1815A, February, 1983.

[Gunakara 90a] The Gunakara Sun Systems, Ltd. Prograph Tutorial. The Gunakara Sun Systems, Ltd, 1990

[Gunakara 90b] The Gunakara Sun Systems, Ltd. Prograph Reference. The Gunakara Sun Systems, Ltd, 1990

[Hatley 87J] Hatley, Derek, Pirbhai, Imtiaz A.. 立田種宏 監訳. リアルタイム・システムの構造化分析. 日経BP,1989

[HOOD 89] HOOD Working Group. HOOD Reference Manual, Issue 3.0. European Space Agency, 1989

[Jackson 83J] Michael A. Jackson. 大野, 山崎 監訳. システム開発JSD法. 共立出版, 1989

[LaLonde 90] Wilf R. LaLonde, John R. Pugh. Inside Smalltalk, Vlo1, 2. Prentice Hall, 1990

[Meyer 88bJ] Bertrand Meyer. 酒匂寛, 酒匂順子 訳. Object-Oriented Software Construction オブジェクト指向入門. アスキー, 1990

[Object 91] Object International, Inc.. OOATool for the Macintosh. Object International, Inc., 1991

[Page-Jones 80] Meilir Page-Jones. The Practical Guide to Structured System Design. Yourdon Press, 1980

[Peterson 81J] James L. Peterson. 市川, 小林 訳.ペトリネット入門 情報システムのモデル化. 共立出版, 1984

[Redmond-Pyle 91] David Redmond-Pyle. Can Formal Methods be User-Friendly?. American Programmer, Vol.4, No.5, May, 1991

[Rumbaugh 91] J. Runmbaugh, M. Blaha, W.Premerlani, F.Eddy, and W. Lorensen. Object-Oriented Modeling and Design. Prentice Hall, 1991

[Shlaer 88J] Sally Shlaer, Stephen J. Mellor. 本位田真一, 山口享 訳. オブジェクト指向システム分析 上流CASEのためのモデル化手法. 啓学出版, 1990

[Stroustrup 86J] B. Stroustrup. プログラミング言語C++. トッパン,1988

[Ward 86] Ward, Paul T., Mellor, Stephen J.. Structured Development for Real-Time Systems, Vol1,2,3. Youron Press, 1986

[Wasserman 87] Anthony I. Wasserman, Peter A. Pircher. A Graphical, Extensible Integrated Environment for Software Development. SIGPlan Notices, Vol.22, No.1, pp.131-142, Jan. 1987

[Wasserman 90] Anthony I. Wasserman, Peter A. Pircher, Robert J. Muller. The Object-Oriented Structured Design Notation for Software Design Representation. Computer, Vol. 24, No.3, 1990

[Winkler 90] Dan Winkler, Scott Knaster. HyperTalk 2.0 The Book. Bantam Books, 1990

[Wilson 90] David Wilson, Larry Rosenstein, Dan Shafer. Programming with MacApp. Addison-Weslye, 1990

[青木 90] 青木淳. オブジェクト指向ソフトウェア開発(OOA, OOD, OOP). SEA環境ワークショップ, 実践的ソフトウェア開発に関する集中討論資料, ソフトウェア技術者協会, 1990

[石井 91] 石井裕. コンピュータを用いたグループワーク支援の研究動向. コンピュータソフトウェア, Vol.8, No.2, pp.14-26, 岩波書店, March 1991

[井田 89] 井田昌之, 元吉文男, 大久保清貴, 他. Common Lisp オブジェクトシステム-CLOSとその周辺-. 共立出版, bit1989年1月号別冊, 1989

[萩谷 91] 萩谷昌己. 視覚的プログラミングと自動プログラミング. コンピュータソフトウェア, Vol.8, No.2, pp.27-39, 岩波書店, March 1991

[橋本 90] 橋本 正, 川端洋一. NextStepプログラミング入門. 共立出版, bit 1990年6月号～10月号

[富士ゼロックス 91] 富士ゼロックス情報システム. マルチクライアントプロジェクト「オブジェクト指向システム技術の実用に関するプロジェクト」成果報告書. 富士ゼロックス情報システム, 1991

[村井 90] 村井純. WIDEプロジェクト. bit, Vol.22, No.2, pp.4-12, 共立出版, 1990

SEA 秋の集中セミナー '91

参加者募集

主　催：　ソフトウェア技術者協会 (SEA)

恒例の SEA 秋の集中セミナーです．金融証券疑惑で騒がしい今日この頃ですが，1990 年代の前半は，いろいろな意味での激動が予感されています．今回は，そうした意味で，最近の SEA のさまざまなイベントの中で多くの関心を集めたトピックスを選び，産業界·学界の第一線で活躍されている会員の方々を中心に講師陣を編成して，8 セッションのプログラムを組んでみました．みなさんの積極的なお申込みをお待ちしています．

開　催　要　領

(1) 日　時：1991 年 9 月 24 日（火）から 26 日（木），時間帯は午前 9:30 - 12:30，午後 13:30 - 16:30

(2) 場　所：青年会議所会館（東京都千代田区平河町 2-14-3　TEL : 03-3234-5601）

(3) プログラム：

セッション	日	時間	テーマ	講師
A1	24 日	午後	定量的プロジェクト管理	堀江進（日本電気ソフトウェア）
A2	24 日	午後	新ピープルウエア	松原友夫（日立ソフトウェアエンジニアリング）
B1	25 日	午前	ソフトウェア·プロセスをめぐる最近の話題	岸田孝一（SRA）
B2	25 日	午前	オブジェクト指向入門	青木淳（富士ゼロックス情報システム）
C1	25 日	午後	マルチベンダー接続とセキュリティ	中野秀男（大阪大学）
C2	25 日	午後	CASE 技術の応用と課題	東陽テクニカ，SonyTektronix，SRA 司会：亀田繁（ファコムハイタック）
D1	26 日	午前	ソフトウェア設計方法論	山崎利治（日本ユニシス）
D2	26 日	午前	知識獲得支援のグループウエアから発想支援システム構築へ	國藤進（富士通研究所）

それぞれ半日単位で 1 セッションを構成します．時間帯は，午前 = 9:30 - 12:30, 午後 = 13:30 - 16:30.

(4) 参加費：　1 名 1 セッション当たり 15,000 円（SEA 会員），20,000 円（非会員），5,000 円（学生）.

(5) 申し込み及び参加方法：　所定の申し込み票にご記入の上，下記宛て郵便またはＦＡＸでお送り下さい．折り返し受講票，請求書をお送りします．定員は各セッションとも 30 名，先着順で締め切ります．なお，申込後のキャンセルはお断わりします．

〒160 東京都新宿区四谷 3-12 丸正ビル 5F

ソフトウエア技術者協会（SEA）事務局

TEL: ０３ - ３３５６ - １０７７　　FAX: ０３ - ３３５６ - １０７２

Ｅ－Ｍａｉｌ：sea@sea.or.jp (junet)

SEA 秋のセミナー'91 参加申込書（申込日：＿＿＿月＿＿＿日）

氏名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿　ふりがな：＿＿＿＿＿＿＿＿＿＿＿＿＿＿

種別（O印で指定）：*SEA* 会員（NO.＿＿＿＿＿＿＿＿＿＿）　非会員　学生

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

部門：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿　役職：＿＿＿＿＿＿＿＿

住所　：（〒　　）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

ＴＥＬ：＿＿＿＿＿＿－＿＿＿＿＿－＿＿＿＿＿＿＿＿（内線＿＿＿＿＿＿＿＿）

申込セッション：（O印をお付けください）：A1　A2　B1　B2　C1　C2　D1　D2

参加費：　申込総金額　　：＿＿＿＿＿＿＿＿円

内訳：申込セッション数：＿＿＿＿＿＿ x（15,000 円（会員），20,000 円（一般），2,000 円（学生））

SEA 秋の集中セミナー'91　プログラム

セッション A1（9月24日午後）

定量的プロジェクト管理

堀江進（日本電気ソフトウェア）

プロジェクトの定量的な把握や管理については，これまで，いろいろな概念や技法あるいは統計的な指標やモデルが提案されてきた．しかし，開発現場での応用は，いまだにそれほど進んでいないのが現状である．実際にプロジェクトの企画運営を成功させるためには，あらゆる側面で事態の定量的な把握と予測を行なうことが必要とされる．ここでは，講師自身が過去に実施してきた管理のための方策を，いくつかの実例を交えて解説する．

セッション A2（9月24日午後）

新･ピープルウエア

松原友夫（日立ソフトウェアエンジニアリング）

プロジェクト管理技術は，ソフトウェアの世界における永遠の課題の1つであるが，最近では，特にその人間的な側面が注目されている．また過去の経験にもとづくノウハウや生活の知恵の累積は，成功を目指すチームにとって必須のものである．ここでは Tom Demarco 氏の話題の書 Peopleware を中心として，ソフトウェア･プロジェクト管理への新しいアプローチを紹介するとともに，人間重視の観点に立ったこれからの管理のあるべき姿を考える．

セッション B1（9月25日午前）

ソフトウェア･プロセスをめぐる最近の話題

岸田孝一（ＳＲＡ）

従来のウォーターフォール型開発パラダイムの問題点を解決するために，その後さまざまなプロセス･モデルが提案されてきた．現在人気を集めている CASE ツールも，適切なプロセス･モデルなしには，有効な活用は期待できない．問題は，現実の開発プロセスをどのようにモデル化するか，そのモデルをどんな形で表現し，稼働させるかである．ここでは，最近の ICSE や ISPW での議論の内容を紹介し，これらの開発環境の動向を展望する．

セッション B2（9月25日午前）

オブジェクト指向入門

青木淳（富士ゼロックス情報システム）

オブジェクト指向型のソフトウェア開発は，現在最も多くの関心を集めている技法であろう．しかし，その原理的な考え方はともかくとして，現実の仕事に対して具体的にそれをどう応用したらよいかについては，まだよく理解されていない点が多い．ここでは，実際にオブジェクト指向言語を用いていくつかのプロトタイプ･システムの開発に従事した講師自身の経験を踏まえて，オブジェクト指向の概念，技法･ツールの概要や具体的な応用例を解説する．

セッション C1（9月25日午後）

マルチベンダー接続とセキュリティ

中野秀男（大阪大学）

ネットワーク技術の急速な発展によって，異機種コンピュータの接続が容易になり，ヘテロジニアスな環境構築がさまざまな場所で推進されつつある．しかし，実際のハード･ソフトの選定，接続に関するさまざまなトラブルの処理，運用に際してのセキュリティ管理など，解決すべき課題が多い．ここでは，講師自身の具体的な経験を例題として，異機種間ネットワーク接続技術の現状との課題を，システム管理者の立場から多角的に展望する．

セッション C2（9月25日午後）

CASE 技術の応用と課題

CASE ベンダー各社からの経験発表とパネル討論

司会：亀田繁（ファコムハイタック）

最近あちこちで CASE 採用の動きが目立っているが，その効率的活用のためには，背景にある技法のトレーニングや，既存のツールや環境との統合，分散環境下でのマネジメントなど，さまざまな面での検討が必要である．ここではベンダー各社が経験した実際の現場への適用事例を踏まえて，CASE 導入に際して起こりうるいくつかの問題点とその現実的な解決策を，参加者とともに考える．

セッション D1（9月26日午前）

ソフトウェア設計方法論

山崎利治（日本ユニシス）

ソフトウェア設計は，完成されたシステムの性能，信頼性，保守等々を左右するきわめて重要な作業である．この設計作業を支援するために，これまでさまざまな方法論や技法が提唱され，それぞれ一定の成果をあげてきた．ここでは，ジャクソン法をはじめ，各種のソフトウェア設計技法とその実践的な応用の推進を手がけてこれらた講師の豊富な経験をもとに，設計方法論の体系的展望と実務への応用上のポイントを解説していただく．

セッション D2（9月26日午前）

知識獲得支援のグループウエアから発想支援システムの構築へ

國藤進（富士通研究所）

コンピュータは人間の知的活動のうち，一体どこまでを支援できるのだろうか？　また，すべきなのだろうか？　それは情報処理の分野を越えたこれからのコンピュータ応用の中心的な話題である．その要素技術としての AI, KB, ニューロ，ファジィなどの理論はすでにある程度の成果上げてきつつある．ここでは，第5世代などで講師自身が進めてきたいくつかの知的支援システムのプロトタイプ開発実験の成果を解説していただき，今後の発展性を考える．

ADVANCE PROGRAM

1st International Conference on the Software Process

Holiday Inn Crowne Plaza, Redondo Beach, CA 21–22 October 1991

The software industry manufactures (and maintains and evolves) large, complex artifacts – software systems. We would like its products to be of high quality and reasonable cost, and to be delivered on time and within budget. One important way to achieve these objectives is to improve the manufacturing process – the software process. A software process is a set of steps for creating and evolving software systems that encompasses both technical and managerial concerns. In particular, it must address technical development, project management, data and configuration management, and quality assurance and control throughout the software life cycle. Historically, the software engineering community has focused on the products of software processes. Recent advances in the understanding of these processes, focusing on the activities involved in creating software products, present an opportunity to solve many of the problems underlying software creation and evolution.

Monday 21 October 1991

8.45am – 9.00am

Welcome and Opening Remarks – Mark Dowson, Conference Chair

9.00am – 10.30am

Panel 1: Why is Process Important?
Chair: M. Dowson, Software Design & Analysis
Panelists: W. S. Humphrey, Software Engineering Institute; M. M. Lehman, Imperial College of Science and Technology, London; L. J. Osterweil, University of California Irvine

11.00am – 12.30pm

Session 1: Using Process Modeling
M. Kellner, Software Engineering Institute, USA. *Software Process Modeling Support for Management Planning and Control*
X. Song and L. J. Osterweil, University of California Irvine, USA. *Comparing Design Methodologies Through Process Modeling*
H. Iida, T. Ogihara, K. Inoue and K. Torii, Osaka University, Japan. *Generating a Menu Oriented Navigation System from Formal Descriptions of Software Development Activity Sequence*

1.45pm – 3.00pm

Session 2: Experience Reports
D. J. Frailey, R. R. Bate, J. Crowley, and S. Hills, Texas Instruments, USA. *Modeling Information in a Software Process*
S. S. Sutton Jr., H. Ziv, D. Heimbigner, H.E. Yessayan, M. Maybee, L. J. Osterweil and X. Song, University of Colorado and University of California Irvine, USA. *Programming a Software Requirements Specification Process*
M. Saeki, T. Kaneko and M. Sakamoto, Tokyo Institute of Technology, Japan. *A Method for Software Process Modeling and Description using LOTOS*

3.30pm – 4.30pm

Session 3: Industrial Applications
H.T. Yeh, AT&T Bell Laboratories, USA. *Re–Engineering a Software Development Process for Fast Delivery – Approach & Experiences*
D. J. Frailey, R. R. Bate, J. Crowley and S. Hills, Texas Instruments, USA. *Defining a Corporate–wide Software Process*

4.30pm – 5.45pm

Panel 2: Process in Industry
Chair: M. Penedo, TRW
Panelists: TBD

6.30pm – 8.30pm

Conference Reception

Tuesday 22 October 1991

8.45am – 10.15am

Session 4: Process Centered Environments
R.F. Bruynooghe, J. M. Parker and J.S.Rowles, STC Technology, UK. *PSS: A System for Process Enactment*
C. Fernstrom and L. Ohlsson, CAP Gemini Innovation, France, and Utilia Consult, Sweden. *Integration Needs in Process Enacted Environments*
N. Belkhatir, J. Estublier and W. Melo, L.G.I. and CNRS, France. *Adele 2: A Support to Large Software Development Process*

10.45am – 12.15pm

Panel 3: Process Research Foci
Chair: W. Schaefer, University of Dortmund
I. Thomas, Hewlett–Packard. *Report from the 7th International Software Process Workshop (October 1991)*
D. Heimbigner, University of Colorado. *The Process Modeling Example Problem and its Solutions*
W. Schaefer, University of Dortmund. *European Process Work – Report from the 1st European Process Modeling Workshop (May 1991)*

1.30pm – 3.00pm

Session 5: Process Technology
P. Mi and W. Scacchi, University of Southern California, USA. *Modeling Articulation Work in Software Engineering Processes*
M. Suziki and K. Katayama, Tokyo Institute of Technology, Japan. *Meta–Operations in the Process Model HFSP for the Dynamics and Flexibility of Software Processes*
M. Gisi and G. Kaiser, Columbia University, USA. *Extending A Tool Integration Language*

3.30pm – 5.00pm

Panel 4: Future Process Directions
Chair: R. Balzer, USC–ISI
Panelists: TBD

1st International Conference on the Software Process

Prototype Systems Information

Non commercial information about prototype process–oriented systems will be distributed at the conference. Attendees may bring 100 – 200 copies of a one/two page description for distribution, or send them in advance to: M. Penedo, TRW, R2/2062, One Space Park, Redondo Beach CA 90278, USA. Descriptions should include system name, technical description, contact information for the system developers, availability of copies for experimentation, software/hardware platform, and references to additional technical information.

Program Committee

Bob Balzer (USA)
Gerhard Chroust (A)
Mark Dowson (USA)
Gail Kaiser (USA)
Kouichi Kishida (J)
Lee Osterweil (USA)
Dewayne Perry (USA)
Win Royce (USA)
Koji Torii (J)
Bill Curtis (USA)
Jean Claude Derniame (FR)
Watts Humphrey (USA)
Takuya Katayama (J)
Manny Lehman (UK)
Maria Penedo (USA)
Bill Riddle (USA)
Wilhelm Schaefer (D)
Colin Tully (UK)

Sponsors

Rocky Mountain Institute of Software Engineering (rMise), in cooperation with British Computer Society (BCS), Institute of Electrical Engineers (IEE), Software Engineers Association of Japan (SEA).

Cooperation requested from ACM and IEEE.

Travel Information

The Holiday Inn Crowne Plaza (Formerly the Sheraton) at Redondo Beach is 7 miles south of Los Angeles International Airport.

SuperShuttle provides on–request service between Los Angeles International Airport and the hotel for $9 each way. Call (215) 417 8888 or use the courtesy telephone in the luggage reclaim area. Advance reservations not needed.

United Airlines is the official carrier for ICSP1, and offers attendees 45% off full coach fare (with 7 day advance purchase) or 5% off any published fare. Discounts apply between 13 October and 25 October on domestic flights for attendees only. Call United on (800) 521 4041 or contact your travel agent quoting Plus ID 514KS.

For Further Information

Call Mark Dowson on +1 703 476 6250 or send email to icsp1@sda.com

Registration Form

Send a copy with your payment to:

ICSP1–Registration
rMise, PO Box 3521
Boulder CO 80303 USA

The conference fee covers admission to the technical sessions, a copy of the Proceedings, morning and afternoon refreshments, buffet lunches and (except students) the Monday evening (cash bar) Conference Reception.

Payment must be made by US dollar check payable to rMise. Sorry, foreign–currency checks, credit cards and POs **cannot** be accepted.

Cancellations received by 15 October will receive a refund less a $50. cancellation fee. **No** cancellations accepted after 15 October (but same–company personnel substitutions accepted).

Advance Registration Fee (received by 1 October):
US$300. Students US$100.

Late Registration Fee (received after 1 October):
US$350. Students US$120.

Payment enclosed: US$_____
Name: _____
Affiliation: _____
Address: _____
_____
_____
Telephone: _____

Accommodation Reservation Form

Send a copy direct to:

Holiday Inn Crowne Plaza, Ref: ICSP1
300 N. Harbor Drive
Redondo Beach CA 90277–2552 USA.

or telephone +1 213 318 8888 by 1 October to guarantee the rates below, which also apply immediately before and after the conference.

Rooms will only be held to 6pm on arrival date unless a one–night deposit (US$ check) or major credit card number is provided.

Name: _____
Organization: _____
Address: _____
_____
_____
Telephone: _____
Arrival date: _____ Departure Date: _____
Room (tax not included): Single ($89) __
Double ($95) __ Government rate ($78) __
Credit Card Name: _____
Card Number: _____
Expiry Date: _____
Signature: _____

ORGANIZING COMMITTEE:

GENERAL CHAIR
Mr. Tony Montgomery
Computer Science Department
R.M.I.T., GPO Box 2476V
Melbourne VIC 3001, Australia
email: aym@goanna.cs.rmit.oz.au
TEL: (61)-3-660-2943
FAX:(61)-3-662-1617

PROGRAM CO-CHAIRS
Dr. Lori A. Clarke
Department of Computer Science
University of Massachusetts
Amherst, MA 01003 USA
email: clarke@cs.umass.edu
TEL:(413) 545-1328
FAX:(413) 545-1249

Dr. Carlo Ghezzi
Politecnico di Milano
Dipartimento di Elettroncia
Piazza Leonardo da Vinci 32
20133 Milano, ITALY
email: relett24@imipoli.bitnet
internet: ghezzi@ipmelz.elet.polimi.it
TEL: (39)-2-2399-3529
FAX: (39)-2-2399-3411

TUTORIAL CO-CHAIRS
Dr. Dewayne E. Perry
AT&T Bell Laboratories
600 Mountain Ave., 3D-454
Murray Hill, NJ 07974 USA
email: dep@research.att.com
TEL:(201) 582-2529
FAX:(201) 582-5192

Dr. John Zeleznikow
Department of Computer Science
La Trobe University
Bundoora VIC 3083, AUSTRALIA
email: johnz@latcsl.lat.oz.au
TEL: (61)-3-479-1003
FAX: (61)-3-470-4915

TOOLS FAIR CHAIR
Mr. John Leaney
Electrical Engineering
University of Technology, Sydney
P.O. Box 123, Broadway
Sydney, NSW 2007, Australia
TEL:(61)-2-218-9320
FAX:(61)-281-5996

CALL FOR PAPERS

14th International Conference on Software Engineering

Melbourne, Australia, May 11-15, 1992

Sponsored by: ACM SIGSOFT, IEEE TC-SE, IEA, ACS, and IREE

The goals of ICSE are: 1) to present the most current and outstanding theoretical, experimental, and practical work being conducted in support of software engineering, and 2) to provide a forum to observe and discuss the state-of-the-art and state-of-the-practice of software engineering. ICSE will offer a variety of presentations: tutorials, refereed papers, panels, tools demonstrations, and abstracts.

REFEREED PAPERS are limited to 6000 words, with full-page figures being counted as 300 words. These papers should include a short abstract, list of keywords, and lead author's address. The content of these papers should not have been published before nor currently being considered for publication elsewhere. Refereed papers will be evaluated with respect to originality, significance, soundness, and clarity. Papers presenting research, experimental, or experience results are welcome. Research papers should indicate how the results contribute to software engineering practice. Experimental papers should describe the experimental method and interpret the results in terms of practice. Experience papers should describe practical experience with state-of-the-art software engineering methods or tools and, most importantly, assess the costs and benefits resulting from their use. All papers should indicate what is new and significant about the work presented and how it compares with related work. Accepted papers will appear in the conference proceedings. The program committee will select the best accepted papers to be considered for publication in a major journal.

PANEL PROPOSALS should include the title, a brief description, the panel chair and panelists (including short vitae). Proposals will be selected according to the substance and contributions to the conference program.

ABSTRACTS should be no more than 2 pages in length and may report anything from new results to experience. Abstracts will be selected on the basis of interest and overall contribution to the conference program. Copies of the abstracts will be distributed at the conference and authors will be given time to make a presentation about their work.
Eight (8) copies of Papers, Panel Proposals, and Abstracts should be submitted by **September 6, 1991** to program co-chair, Lori A. Clarke.

TUTORIAL PROPOSALS should describe important software engineering methods and tools and include a detailed description of a full day tutorial and the relevant experience of the tutor. Proposals will be selected according to their substance and contribution to the conference program. Eight copies of tutorial proposals should be submitted by **September 6, 1991** to the Tutorial Co-Chair, Dewayne E. Perry.

TOOLS FAIR PROPOSALS should represent current experimental and commercial state-of-the-art tools and include a brief description of the tool and the planned demonstration, indicating if a presentation is desired. Exhibits and presentations will be selected on the basis of their importance, relevance, and novelty. Tools Fair proposals should be submitted by **January 31, 1992** to the Tools Fair Chair, John Leaney.

FURTHER INFORMATION: For further information and/or a copy of the advance program when available, write ICSE14, IEEE Computer Society, 1730 Massachusetts Ave., NW. Washington, D.C., 20036 USA. Some travel funds may be available to assist a limited number of attendees. Priority for such funds will be given to authors of accepted papers.

IMPORTANT DATES:
Papers, Abstracts, and Panel Proposals submission deadline: **September 6, 1991,**
Acceptance notification **December 15, 1991,** Tools Fair proposals submission deadline:
January 31, 1992, Final versions of papers due: **February 1, 1992.**

PROGRAM COMMITTEE
W. Richards Adrion USA David Barstow USAVictor Basili USA Veronique Donzeau-Gouge FRANCE
Anne Duncan USA Jean-Pierre Finance FRANCE Kokichi Futatsugi JAPAN David Harel ISRAEL
Daniel Hoffman CANADA Susan Horwitz USA Mehdi Jazayeri ITALY Ross Jefferey AUSTRALIA
Gail Kaiser USA Takuya Katayama JAPAN Richard A. Kemmerer USA Reino Kurki-Suonio FINLAND
Bev Littlewood UK Carlos J.P. Lucena BRASIL Thomas Maibaum UK Chris D. Marlin AUSTRALIA
Carlo Montangero ITALY Leon J. Osterweil USA James M. Purtilo USA Gruia-Catalin Roman USA
Wilhelm Schafer GERMANY Sol M. Shatz USA Mary Lou Soffa USA Walter F. Tichy GERMANY
Koji Torii JAPAN Axel vanLamsweerde BELGIUM Anthony I. Wasserman USA Jim Welsh AUSTRALIA
David Wile USA Jeannette Wing USA Alexander L. Wolf USA Michal Young USA

***** SEA では例年通り会議参加の研修ツアーを計画しています *****

NeXT ユーザ会の紹介

NeXT ユーザ会は，1990 年 6 月に活動を開始し，1990 年 10 月 9 月月 26 日に正式に発足いたしました．

[目的] NeXT User Society (NeXus) とは，NeXT に代表される Computer 技術

- ·使い易い優れたユーザインタフェース
- ·統合されたネットワーク環境
- ·洗練されたウィンドウアプリケーション
- ·優れたオブジェクト指向開発環境など

に興味を持つ個人のために，会員同士が協力し情報交換を行なうための非営利の団体です．

[活動] またこのような新技術および文化の普及と発展を諮るために次に示す各種活動を行なっています．

- ·月例会 (第 4 水曜日) 開催 - (第 4 週の水曜日ではない)
- ·各種専門分科会活動
- · NeXus-Memo(月刊) の発行 - (原内容を保持する限り転載自由)
- · NeXT ユーザ会会報 (季刊) の発行 - (転載不可·非会員有償)
- ·会員内外との情報交換
- ·海外ユーザグループとの情報交換
- ·再配布可能ソフトウェアの配布
- ·ネットワークによるユーザ会活動の広報
  fj.sys.next, NeXT 会議室 (FCSALON)@Nifty, next.ws 会議 @mix
- · E-Mail サービス (mailing-list の運用)
  nexus-office@nexus.or.jp
- ·セミナー/シンポジウム開催

更に，ベンダーに対しての情報請求·改善提案·試用報告なども行ないユーザとして更に良い製品の改良·開発を支援していきたいと考えています．

[運営] ユーザ会の運営は，会員有志の自発的な協力と奉仕により行なわれています．また，その活動経費は会費と活動収入及び善意の寄付によって賄われています．

NeXT ユーザ会 (NeXus) は，全ての NeXT 及び NeXT 関連技術·文化に関心のある個人の為の非営利団体です．その活動は，会員有志の奉仕によって運営されています．NeXT 所有の有無は入会の条件ではありません．NeXT に興味を持つ方の月例会へのゲスト参加を歓迎します (参加費 1,000 円)．月例会参加者には，最新の NeXus-Memo を配布しています．

また最近の主な活動として，

- · Super ASCII 1991 年 6 月号 TEST LAB.「NeXT OS 2.0」(評価記事) の執筆
- ·キヤノンのプライベートショウ「NeXT コンピュータフェア '91」の参加

があります．

塩谷和範　　NeXT ユーザ会 代表幹事　fax:03-3351-0880
NeXT ユーザ会事務局　〒142 東京都品川区荏原 4-4-2　大橋医院気付
junet:nexus-office@nexus.or.jp, mix:shioya, Nifty:PFC02015(小澤)

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