SEAMAIL

Newsletter from Software Engineers Association

Volume 5, Number 4 1990

目　次

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，1650 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．

ＳＥＡＭＡＩＬ　Ｖｏｌ．５，Ｎｏ．４　　平成２年４月２７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）

〒１０２　東京都千代田区隼町２－１２　藤和半蔵門コープビル５０５

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

☆

4月18日23時20分，もうすでに門限に遅れている．今週はこれで2度目の深夜残業です．SEAMAILを順調に出し続けるのは大変です．しかし，なんとか月内に原稿を印刷所に入れることができました．

☆☆

今月号には，春のセミナーにおける松原さんの講義クイック・レポートと，1年前の盛岡でのSIGENVから青木さんの原稿をのせました．そして，この間行なったミニ・アンケートの集計でページをかせぎました．これで，手持ちの原稿はすべておしまいになりました．

☆☆☆

SIGEDUおよびSIGENVから，今年の活動計画をいただきました．他の分科会や支部の活動予定はどうなっていますか？　世話人の方々よろしくお願いします．

☆☆☆☆

久保さん（若手の会），杉田さん（教育），平尾さん＋篠田さん（環境），熊谷さん（テクニカルマネジメント），片山先生（プロセス），はやくワークショップのレポートをまとめてくださいネ．

☆☆☆☆☆

来月の総会で新しい役員がきまったら，いまほとんど壊滅状態にある編集スタッフを再編成していただきたいものです．ボランティア御希望の方，ふるって事務局まで御連絡ください．

☆☆☆☆☆☆

SEA 春の集中セミナー '90
Session C1

CASE 環境をめぐる標準化の動向

講　師: 松原　友夫〔日立ソフトウェア・エンジニアリング〕
報告者: 渡邉　雄一〔アスキー〕

0. はじめに

SEAの会員諸兄は「標準化」についてどの程度の関心をお持ちであろうか？

標準化というと，せいぜい「JISの規格ハンドブック」くらいしか思い浮かばなかったレポータ（渡邉）にとって，今回の講演はかなりショッキングであった．それは世界における日本の立場が，標準化のための地道な活動をわれわれが敬遠することによって，実は非常に危うくなることが痛感されたからである．このような危機感を私に感じさせたCASE 環境の標準化とは，一体どのような問題を含んでいるのであろうか？

1. いまシグマを考える

通産省が中心となって進めてきたΣプロジェクトに関しては，批判的意見が少なくない([３],[５])．ここでは，「標準化から見たΣプロジェクトの問題点」を，松原さん流の鋭い切り口で御披露いただいた（松原さん自身がΣプロジェクトの進むべき方向をどんな風に考えていたかの詳細は[１]を参照されたい)．

まず，Σプロジェクトは（当時そのことばが使われていたわけではないが）CASE 環境の標準化そのものと捉えるべきであった．このことについては，Σ本部ははっきりした態度をとらなかったが，諸外国からそのような評価を受けていたことは，各種の資料からはっきりとしている．また国内においても，業界が本当に期待していたものは，単に役立つツールや進んだ環境だけではなかったはずである．

Σのスタート時点には，「ソフトウェア工業化による生産性４倍増」などというスローガンがまことしやかに叫ばれたりしたが，現実にソフトウェア生産性の向上は，１企業内で試みても難しいのに，国がやればうまくなどと誰が思っていたであろうか？

ツールや環境は，誰でも自分でいじって自分用に使い易くモディファイしたいものである．本当に技術者が期待していたものは，日本の「共通プラットフォーム」としてのΣであった．それだけでもできていれば，たとえツール類が使いものにならなくても，標準化という観点からはそれなりに成果を上げたものとして評価できたのである．

なぜなら，もしそうしたプラットフォームができていたならば，必要なツールの類はその上で，自然発生的に開発され，自己増殖が始まっていただろうからである．国の政策に欲しい視点は，そのような技術的インフラストラクチャ（下部構造）作りを重視することであって，個々のツールや環境の整備は産業界が自らの力でやるべきものである．

また，Σでは，Unix をモデルとして，システム（いわゆるΣOS)の仕様を作成したようだが，それらは，実はΣ環境（OS ではない）の各種インタフェィスを決めることであり，今後日本で開発環境を考えるさいの事実上の規格となるはずのものであった．それには，Unix ではなく，むしろAPSE（Ada Progamming System Environment)におけるCAIS（Common APSE Interface Set）のほうを参照すべきであったと思われる．にもかかわらず，そのについての十分な配慮がなされないまま事態が推移してしまった．

このことは，JISA 会員に対するΣ説明会の席上での次のような質疑（質問者は松原さん自身）のすれちがいからも明らかである：

Σ：「現在，インタフェィス･スペックという名前の要求仕様書をまとめているところです」
Q：「インタフェイス･スペックは，システム要求仕様書の一部にはなり得るが，まったく性格が異なり，インタフェイス標準を決めるもので，Σの成否にかかわる重要な文書です．両者を混同してもらっては困ります」
Σ：「われわれは Unix 環境をモデルにして，Σシステムを作ろうとしているので，標準を作ろうとしているのではありません」
Q：「いや，シグマのモデルは Unix でなく ARSE ではありませんか？

国としてのオープンな環境を作るには，インタフェイス規格は必ず必要です．CAISがその例でしょう．シグマは事実上国の標準を作る仕事です．その積りでみんなが使える標準を作って下さい」

Σ:「！？」

Σプロジェクトが始まる半年ほど前に，プロジェクトのあり方を考えるべく開かれたあるワークショップで，「国のお金というものは，どの国でも10%有効に使えばいいほうです．少しでも有効に使うようにみなさん頑張って下さい．それにはおかしなものを作らないことです．すでに動いていて評価されているものを買ってきてシステムを組み，悪いところだけ直すのです．そうすれば役に立つシステムが作れます」という趣旨の発言を某大学教授がされたのだが，この予言はみごとに悪いほうに的中してしまったようである．

2. 標準化とは

2-1. 標準化の意味

標準ということばから，「ルールに従わされる」「統制」「制約」といったイメージを思い浮かべる人は少なくないだろう．だが，その先には一定の目的があり，そのためにみんなでルールを作ってそれを守るのが，真の標準化なのである．つまり標準化の目的は，利害関係者全員で共通の場を作り，それによって何らかの効用を生み出すことにある．その場の上で，各人が大きな自由を享受できる予感が先立つことが，望ましい標準化の条件である．

つまり，標準化によって，コミュニケーションの自由，交換の自由，利用の自由，選択の自由などが達成され，自由競争が促進されることとなる．

2-2. 標準化の効用

このような観点で標準化をとらえると，活動領域を提供する基盤としての「広いプラットフォーム(場)」が重要となる．それが広ければ広いほど効果は大きい．特に，情報(ソフトウェア)の分野は無形物が対象であるから，ハードウェアの分野より広域の場がもたらす効用は大きい．この意味では，現在行われている標準化の対象範囲がまだまだ狭いのは問題と言えよう．

また，まったく独立に作られた複数の標準は，かえって混乱を招き自由を損なうこととなり，好ましいことではない．たとえば，インドの鉄道や，ビデオテープのβとVHSなどがその例である．

さらに，局所的標準化は非関税障壁となりうるので，注意が必要である．最近の例では，ΣやTRONがそう見られており，ある意味で成功しなかったのが幸いかもしれない．

2-3. 国際標準

国や団体がローカルな規格を作り，それが非関税障壁になるといった問題をなくすために，GATTスタンダード・コード(貿易の技術的な障害に関する協定)が1979年に締結された．その基本原則は次の3つである：

(1) 国の規格の適用が国際貿易の障害とならないようにすること．

(2) 国際規格を国の規格の基礎とすること．JISはこの原則に従っており，特に「情報」の部門では，ISOの案が決まってからそれをJISに持ってくるため，JISの規格作りの実態は翻訳作業(！)になってしまっている．

(3) 国際標準化機関が国際規格を立案する場合は，各国は十分な役割を果たすこと．

このようなことから，国際プロジェクトでは規格を守らねばならないことが常識となっている．また最近では，特にEC諸国の統合化の動きと併せて，ヨーロッパが標準化作業に熱心である点は注目すべきである．

2-4. 成長と衰退

さて，標準化は共通の場であると述べたが，その場には「成長する場」と「衰退する場」とがある．すなわち，場のライフサイクルが考えられなければならない．たとえば，ビデオテックスの世界では，フランスのMinitelは成長する場だが，日本のCaptainは衰退する場だろうし，ビデオテープでいえば，VHSは成長，βは衰退と判定できる．日本語ワープロのテキスト・フォーマットの場合は，MS-DOSの場が成長し，JISフォーマットは衰退している．開発環境の分野では，APSEは今後も成長を続けるだろうし，シグマはもうすでに衰退が予想される．

成長する場の条件としては，

(1)ねずみ算メカニズム：「味をしめた人が他人にすすめる」というボトムアップで自己増殖的な力，これはその場に魅力がなければダメである(例：Unix)．

(2) 規格開発の参加基盤が大きい：(例：OSI, OSF, UI, CAIS, PCTE)

(3) 既存の場の利用：(例：IBMのOS)

(4) 強制力：(例：アメリカにおけるDoDやNIST)

といったことが挙げられる．

2-5. 標準化のプロセス

標準化の作業プロセスも，これまでのような De Fact の追認という形から，まず規格開発を行なうという方向になり，一昔前と比較して，かなり変わってきたといえる．すなわち，今後は，規格を先に決めて製品はあとから作るというものが多くなるであろう．たとえば，８ミリビデオや OSI がそのいい例である．このようなことから，ますます，標準化の活動に参加することが重要になる．同時に，標準化は，以前にもまして息の長い地味な仕事になってきた．

特に，国際的な場では，特定の人間が継続的に関連会議に出席し，具体案を提示することが，その国の発言力を大きく左右するようになってきた．このことは，「社内でその職務を担当することになった人間が（標準化の）委員を引き継ぐことが当たり前」な日本の社会メカニズムには馴染まず，また，具体的に会合に参加するさいの地理的なハンディキャップも少なくないが，今後とも努力を重ねていく必要がある．

そして，そのような国際会議において重要な心構え（標準化の精神）とは，

（1）話合い　－　競争相手と
（2）強調　　－　主張と妥協，積極的な貢献
（3）参画　　－　みんなで一緒に考える
（4）忍耐　　－　続けることが大事

の４点である．

また最終的に，標準化案の決定は投票によって行われるが，これは１国１票である．この決定メカニズムは EC に有利であり（アメリカがどんなに大国でも，たった１票しか権利がない），また実際の市場の大きさなどからも，EC の動向には今後注意が必要である．

2-6. ソフトウェアの標準化

この節の締め括りとして，ソフトウェアの分野における標準化の動向について一言しておこう．

これまで，各企業の内部では，ドキュメンテーション，ライフサイクルにおけるプロセス定義，作業基準，見積基準などが主な関心事であった．また，国際的には，図記号やその使用法，ドキュメンテーション，品質の評価，品質保証契約条項などが主に検討されてきた．

そのような標準化の関心の対象が，ソフトウェアの複雑化と CASE の出現によって，いまや大きく変わろうとしている．実際，巨大化し階層化されたソフトウェア・システムの構造は，自然に，人々に対して，ざまざまなインタフェィスの標準化問題を意識させるようになってきた．具体的には：

－ OS とアプリケーション
－モジュール間
－共通のレポジトリ
－外部システムとの接続
－ツール間
－ツールと人間

などである．

そして，CASE はそうしたインタフェイス・ルールの厳密化をもたらすのである．

3. CASE の定義

前節で見てきたように，ソフトウェア標準化の関心が CASE に向いてきたことと関連して，その作業を進めていく上では，当然ながら CASE の定義をする必要がある．たとえば，IEEE ソフトウェア工学用語集では，

「ソフトウェア工学のプロセスを支援するためにコンピュータを使用すること．ソフトウェアの設計，要求定義の追跡，コード生成，試験，文書生成，およびその他のソフトウェア工学的作業のためのソフトウェア・ツールの応用を含む」

と説明されている．

しかし，だれでも容易にわかるように，これが完全なものと到底いえない（たとえばこの説明では，CASE の適用範囲が規定されていない点など，注意が必要といえよう）．CASE 関連の標準化を進めていく上で，「まだ CASE そのものが定義できていないから議論が前に進まない」ということはないので，個々の要素の標準化が熱心に進められてはいるが，それと並行して，やはり CASE そのものの定義も，併せて検討されている．

現状では，特に各種のツールのうちどこまでを CASE に含めるかは，人によってさまざまである．しかし，多くの定義には，

（1）統合型（開発工程の上流＋下流）のツール
および
（2）ツール・サブセットまたは単体ツール

の両方が含まれている．特定の業務分析などを対象としたもの（たとえば，インダストリアル・ダイナミクス用ツール Stella）や，超上流工程支援ツール（いわゆる CAST: Computer Aided System Theory）までを含めるかどうかについては，意見が分かれるところである．

CAST の例としては，ペトリネットや FSM（Finite

State Machine) on WS などがあげられる．これらのツールの開発や利用に関しては，どちらかといえばヨーロッパのほうが熱心である．いずれにせよ，いろいろな領域で CASE が検討され始めていることは事実である．

4. CASE の利用段階と市場の形成

4-1. CASE の利用段階

CASE の標準化への関心の高さは，CASE の利用の段階（技術レベル）に強く関連していると思われる．それについてはあとで第６節で説明するが，そのための前提として CASE の利用の段階を説明するのにうってつけの資料［２］があるので，そこでの分類を紹介しておこう．

一般に CASE の導入や理解は，以下のような段階を経て進む：

Trigger Point-1:

何か新しい手法またはツールを使う必要がありそうだ．

Stage-1: CASE への目覚め

よくわからないのにツールを買うが，その 80～90% は使われないままに終わる．

Trigger Point-2:

ツールはどうやら役に立つらしい，ソフトウェア開発の改善に必要なものらしい．

Stage-2: 失望

使ってはみたけれど，うまく行かなかった．どうもこちらの理解が足りなかったらしい．いくつかのツールを使ってみたが，どうやらこれらを何らかの形で統合化することが必要のようだ．

Trigger Point-3:

どうやら，方法論とツールは分けて考えないといけないらしい．

Stage-3: とにかくもう一度やってみよう

仕事に合ったツールを探しはじめる．１つ１つのツールがみんな違うことがわかり，選択・評価の重要性に気づく．そこで，専任者を割り当て，パイロット・プロジェクトを設定し，CASE の導入と適用を試みる．

Trigger Point-4:

もっと本格的に，資源（人とお金）を割当てなければ，うまくいかないということに気づく．

Stage-4: 本腰を入れる

小規模なパイロット実験ではなく，主要なプロジェクト開発のために，特定の CASE ツールおよび方法論を本格的に導入する．

プロジェクトを厳格な管理下に置き，その状況をモニターする．

CASE の活用をうまくやるために，さらに教育や指導に力をいれ，ツールを追加購入する．

Trigger Point-5:

いくつかのプロジェクトを CASE を使って実施してみて，１つの方法論だけに頼っていたので，はうまくいかないことに気づく．

Stage-5: 成熟期

１つ１つのプロジェクト固有の要求にあった複数の方法論を組み合わせて，ツールを採用するようになる．また，そのほうが CASE の応用を効果的にすることがわかる．

4-2. CASE 利用の初期段階について

まず最初のステージは，CASE への目覚めである．何らかのきっかけで，「CASE には効き目があるらしい」と気づくわけである．その要因をを断片的に拾ってみると，以下のようなことが考えられる：

(1) CASE の利用効果が定量的に数値で示された論文や資料からの刺激．現実の効果は，CASE への慣れの問題があるので，まず品質に表れ，次に生産性がついてくる

(2) メソドロジー，テクニック，標準の確実な実施．今までは，ハードウェア資源の制約もあって，プログラミング工程にのみしか適用できなかったツールを，パソコンやワークステーションの普及という状況の変化にともなって，ライフサイクル全体に適用したい．

(3) 非可逆的な改善．失敗を繰り返していつも振出しに戻るような無駄な努力に終わらないよう，段階的に改善をして行きたい．

(4) 開発と管理の統合．プロダクトも，またそれを開発する工程の管理数値も，いずれも情報なので，それらを統合することは比較的容易である．またその統合の効果も大きい．このことは，欧米では盛んに言われている．ただし，日本では，へたをすると監視社会のの落とし穴にはまる危険性もある．

(5) 改善へのフィードバックが容易.
(6) 可視性の向上. マネジメントにとっては, 作業の管理がしやすくなる. また, 作業者自身にとっては, 自律性が働く.
(7) 強制感のない標準化. 一定の方法論やツールを用いて作業を進めていくため, おのずと作業内容が標準化されてくる.

4-3. CASE 市場の形成と問題点

最近, 特にアメリカでは, ソフトウェア・ハウスのマーケティング意欲を刺激するに十分な CASE 市場が形成されつつあり, 急成長している. 現在のところ年率実に35%もの伸びを示しており, 1993 年には, その規模はおそらく 1500 億円にも達するだろうと予測されている.

しかし, 次から次へと新しいプロダクトが生まれている一方で, それらがビジネス上の採算ラインにはなかなか達していないのも事実であり, CASE ベンダは乱立から淘汰の時代へ向かいつつあるというのが現状である.

さて, こうして CASE の利用人口が確実に増加してくると同時に, いろいろな問題もクローズアップされてきた. それらのうち, 主要ものを箇条書きにすると, 以下の通りである:

(1) ツールのミスマッチ

ツールが対象システムと合わなかったり, プロジェクト環境と合わない(たとえば, APSE との連携をいかにはかるか?). アプリケーションの開発規模が大きいと使えない (この問題はかなり深刻! 何のための CASE か?). 作業がうまく流れない. 開発現場の実情に合わせようとしても, うまくカストマイズできない. かゆいところに手が届かない (機能が大雑把でまだ足りない).

(2) 使うための前提条件が未整備

導入する側が, ライフサイクルを変えられない. 開発のプロセス・モデルが規定されていない (会社の中で, まだ標準的なライフサイクル・モデルが決まっていない場合には, それを規定する必要がある). 作業分割のルールが決まっていない (アメリカでは分業化が進んでいるため, このことは大きな問題である). 利用者側に方法論の理解が足りない. ワークステーションの台数が技術者の人数に比較して足りない. 資源 (お金, 人) の割当てが十分でない.

(3) 拒否反応, 恐怖感

プログラマの仕事がなくなってしまうという危機感が出てきた. 開発工程がある意味でガラス張りになってしまうので,「監視社会」が形成される恐れがある (プライバシーの尊重は絶対必要!).

(4) 実際の開発プロジェクトで使ってみると....

上流から下流へのツールがうまくつながらない. 複数の方法論やツールが必要となるが, これらをうまく統合できない. リポジトリが共用できず, 再利用が局所にとどまる. 異なるベンダのツールは特にうまくつながらない. 同じベンダのものでも, 1つの部分は良くても他がダメなことも少なくない.

(5) ニーズへの対応

つまり現在の CASE は環境として使いたいというニーズに対応できていない. CASE をのせる商用のオープンな環境が強く求められている. 同時に, CASE 環境アーキテクチャの未熟さも大きくクローズアップされてきた.

4-4. CASE 利用の現状

ところで, CASE の利用は現在どのような段階にあるのだろうか? 国別に状況を分析してみよう.

まずアメリカだが, 一般的にいえば, 上述の Stage-3 から 4 へ向かいつつあるようだ. DoD 関連の組織は Stage-4 から 5 へ, その他の民間企業は Stage-2 から 3 へといった開きはあるが, 民間でも, 先進的な企業はすでに Stage-4 に到達しているように見える.

一方, ヨーロッパ各国はといえば, EC 統合をにらんで数年前から始められた Esprit や Eureka などの国際協力プロジェクトの影響もあって, 急速に Stage-4 へ向かいつつあると見るのが適当であろう. 特に, ヨーロッパでは, フォーマリズム (形式的言語) 中心のアプローチが盛んであり, したがって上流 CASE への関心が高い. また,「高度なプロフェッショナルだけにソフトウェアを作る資格がある」というエリート主義的な考え方が支配的であるために, ツールも高度なプロ向けのものが多いというのが特徴である (これに対してアメリカ製 CASE の多くは大衆向けプロダクトだといえよう).

このような国際状勢の中で, 残念ながら, 日本はまだ Stage-1 から 2 の段階にいるところが中心だといわざるを得ない. Σの影響は Stage-1 の会社数をわずかに増やした程度にすぎず, むしろ今後 Stage-2 で止まってしまうおそれが多分にあるようで, 懸念される. 資本力のある企業の中には, Stage-3 から 4 へ進んでいるところもあるが, おおむね自社内で閉じたクローズドシステムが

多いというのが，実情である．

5. CASE 標準化への関心

CASE 利用がさらに成熟化し，Stage-5 に進む上で予想される大きな壁は，オープン・プラットフォームの欠如であるといえる．それを打破するためには，CASE 関連技術の標準化が絶対に必要であるが，この問題に関する関心の高さは，現在の CASE 利用段階と強い相関がある．現状では，Stage-4 段階にある CASE ユーザが，特に標準化の必要性を痛感しているようである．

アメリカの状況を見ると，オープン CASE 環境の必要性は，すでにコンセンサスになっているといってよい．昨年 NIST (National Institute of Standard Technology) が主催するワークショップに参加したが，そこでの討論には標準化への熱気さえ感じられた．

ヨーロッパでは，まだベンダー主導で先端技術の追求に多くの関心が払われているが，EC 統合に向けて，環境におけるツール・インタフェイスの欧州標準案である PCTE (Portable Common Tool Environment) への関心は，急速に高まりつつある．これに対して，日本では CASE 標準化に関心を持つ人はまだごく少数でしかない．特に，公的機関における関心が低く，アメリカとは逆の状況である．

6. NIST 環境ワークショップの話題から

前述の NIST が主催するワークショップは，ISEE (Integrated Software Engineering Environment) と名づけられており，すでに２回開かれている．こうしたワークショップを舞台として，CASE の標準化は動き出している．

このワークショップに集まった人々の顔触れをみると，CASE ユーザ (NASA, Air Force, Navy などの軍関係者および一般ユーザ)，CASE ベンダ (メインフレーマ，ソフトウェア・ハウス)，中立機関 (大学，研究所) そして行政機関 (NIST 他) と，実に多彩である．特に，第１回には上院財務委員会の Joe Humphreys 議員も参加している．出席者のほとんどはアメリカ人だが，外国からの参加者としては，日電の寺本さん (第１回のみ)，松原 (第２回のみ)，Ian Thomas (PCTE の推進役だった英国人だが，いまはアメリカで仕事をしている) などがいる．

第２回の会合 (1989.12.5-6) では，前回の討論結果に対する宿題をこなしながら，インタフェィス標準のガイドラインを設定することを目標に，ソフトウェア (工学的) 開発支援環境の要求仕様とアーキテクチャ・モデル (SEAM:Software Environment Architectural Model) に焦点をあてて，討議が行なわれた．実際には (a) ISEE のための要求仕様，および (b) 概念的なフレームワーク／アーキテクチャ／ライフサイクル・メタモデル，という２つの分科会討論 (３つ予定されていたが，集まりの悪かった１つは中止) が行われた．

さて，ここでの討論の話題から CASE 標準化が関係することがらを拾ってみよう．

(1) システムサービス：OS の標準化．

(2) リポジトリ：CASE 固有のリポジトリ要求，ツール，再利用部品，知識ルール．DBMS に関しては，すでに SQL[FIPS127] や IRDS[ANSI 3.138] などがある．

(3) データインタチェンジ：プロダクトデータや文書，グラフィクスなど．これらについては，
プロダクトデータ→ NBSIR 88-3813
文書→ FIPS152,ISO/IS 8613 (ODA,ODIF)
グラフィックス→ GKS,FIPS128
などの標準案がある．

(4) 言語：VHDL，フォーマル言語，設計言語，マルチ言語のインターオラビリティなど．特に最後の項目は今後の大きな課題である．

(5) 図記号：データフロー図，機能構造図，E－R 図などに関する標準．ISO/IEC JTC1/SC7/WG1 がそれにあたる．

(6)ネットワーク・サービス：OSI, FIPS 146 (GOSIP)などのデータ通信まわり，および IEEE P1003.8 で扱われているファイル・マネジメント関係．

(7)ユーザ・インタフェィス：ウィンドゥ・システム，つまり X-Window (X11R4) など．

(8) プロセス：プロセスモデルやライフサイクル (IEEE P1074)．

(9) メトリクス：ソフトウェア品質メトリクス (IEEE P1061 SC7/WG3)，およびソフトウェア生産性メトリクス (IEEE P1054)．

(10) 評価：CASE ツールの評価 (IEEE P1209)．

(11) 管理：プロジェクト管理 (IEEE Std 1058) やソフトウェア構成管理 (IEEE Std 828,1042)．

このように CASE ツールの標準化で扱わねばならない領域は，OSI などと比較しても遥かに広く，とても１

つの機関で手に負えるようなものではない．したがって，その作業を進めるにあたっては，壮大なグローバル･プロジェクトを起こす必要があるといえよう．現在，その方向へ向かってのリーダシップをとっているのは，NISTと EC であり，いわば欧米連合の形での国際プロジェクトが形成されつつある．そこでは，世界中でやっている標準化活動(DoD Std, IEEE Std, EC Std. など)のマップを作って，相互に連絡をとりながら進めようとしている．

具体的には，これまでヨーロッパが進めてきた PCTE と，アメリカ(DoD) が進めてきた CAIS とを統合化し，オープンな CASE 環境のプラットフォームとして，両者の名前を足して２で割った（！） PCIS: Prtable Commmon Interface Set なるものを 90 年代半ばまでには制定するという方針が決まっている．

また，ツール･インターコネクション [IEEE P1175] に関する 247 件の関連規格を調査した上で，インタフェースを表現する STL (Standard Text Language) をまとめた規格草案ができている．

そして，なぜか日本はこうした動きに参加できないでいるというのが，悲しい現状である．

7. 日本も積極的な貢献を！

私(松原)自身がこれまでやってきたソフトウェア標準化に関する努力の多くは CASE 標準化に結びつくものであった．これまで世界各国(特に日本)で標準化にかかわってきた人たちも，心のどこかで虚しさを感じていたとというのが，正直な実態であった．そこへ，思いがけない(？) CASE ブームによって，具体的な目標があらわれた．

だから，みんなで得意なところ，ニーズの強いところを分担しあって，連絡を取りながら活動を進めて行こうという気運が高まっている．ある意味では，ソフトウェアの標準化が CASE の標準化を加速しつつあるといえる．

そのような状況の中で，世界の中の日本が抱える問題点／課題は，以下の通りである：

(1) 国際標準化に対する日本からの貢献は，なぜかいままでところゼロ．日本は１人だけはずれている感じた．

(2) シグマ会社は OS の標準化にだけ参加しようとしているが，それはわずか一部にすぎない．

(3) CASE の標準化はこれから本格化するので，過去のいきさつは忘れて，これから積極的に貢献すればよい．

(4) 日本もいろいろなことがらに対して，積極的に手を上げることを提案したい（公の立場の人の積極的参加の必要性）．

また，ソフトウェア開発の現場でも，それぞれ，いまからオープン環境を整備するための条件整備をやっておく必要があるだろう．特に，CASE の利用に関しては，欧米と日本とのギャップが大きいので，効果のありそうなもの(たとえば，プロセスモデル，作業分割，管理手法など)を積極的に取り入れて行く必要もある．また，日本の特殊条件(たとえば，日本語の扱いなど)を考慮したプロセス･モデルを作るなど独自の活動も当然求められる．

【参考資料】

[１] 松原: シグマ･プロジェクトへの建設的提案, Seamail,Vol.1,No.3 (pp.12-16), 1986

[２] Cary T. Hughes, Jon D. Clark: The Stages of CASE Usage,DATAMATION,Feb.1, (pp.41-44) 1990,

[３] 田口: ２５０億円と５年をかけた国家プロジェクトの失敗, 日経コンピュータ, (pp.72-100) 1990.2.12

[４]たとえば情報処理(情報処理学会誌)の巻末に「情報技術標準化のページ」が設けられている．

[５] 失敗に終りそうなΣ計画: インターフェース, Vol.1,No.4 (pp.310), CQ出版, 1990.4

オブジェクト指向プログラミングのアニメーション

～ SIGENV Workshop in 盛岡 ～

富士ゼロックス情報システム株式会社(FXIS)

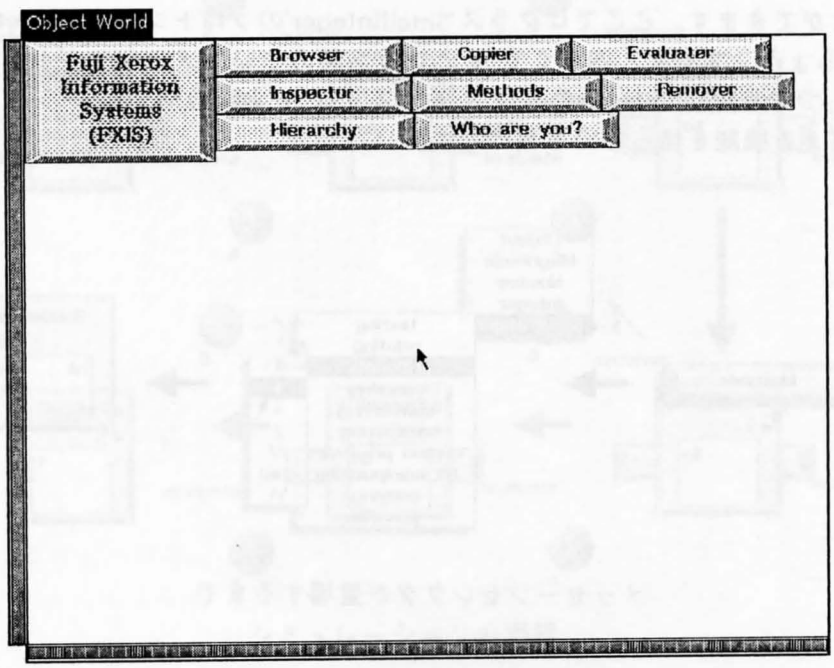
青木　淳　　阿部和広

Smalltalk-80のオブジェクト指向計算モデルシミュレータであるARK-likeを用いて、シミュレーションを行う過程について説明します。このシミュレーションは視覚的なアニメーションによって進行しますから、これによって「Smalltalkって何なの?」という疑問に対する直感的な理解を得ることができます。

その前に、Smalltalk-80の動作環境について説明します。Smalltalk-80は現在、富士ゼロックス、Sun、Apollo、HP、SONYの各ワークステーション上のUNIXおよびApple Macintosh上のMacOSで稼働しています。また、80386プロセッサを持つIBM-PC上のMS-DOSにも移植されています。また、現在のSmalltalk-80のバージョンには2.2、2.3、2.4があり、ARK-likeはSmalltalk-80で記述されたアプリケーションであるため、いずれのワークステーション、バージョンでも動作可能です。これは、Smalltalk-80の高いポータビリティを証明しています。

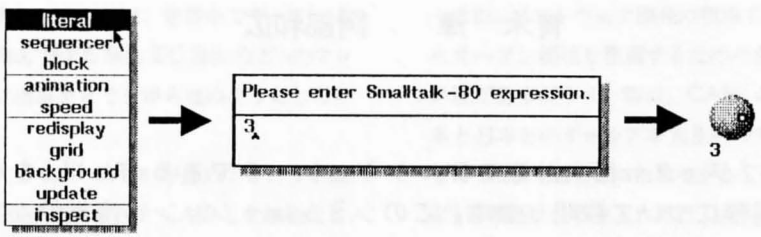
1. オブジェクトたちの世界

では、ARK-likeについて実際に見ていきましょう。下の図にARK-likeの初期画面を示します。



ARK-likeの初期画面

ARK-likeのビューのラベルには"Object World"と書かれています。このラベルの名前の通り、このビューの中に様々なオブジェクトが登場し様々な役割を演じます。例えば、'3'というオブジェクトを登場させてみます。マウスとメニューを用いて3を入力すると、まさにオブジェクト(物体)となった3がビューの中に現れます。



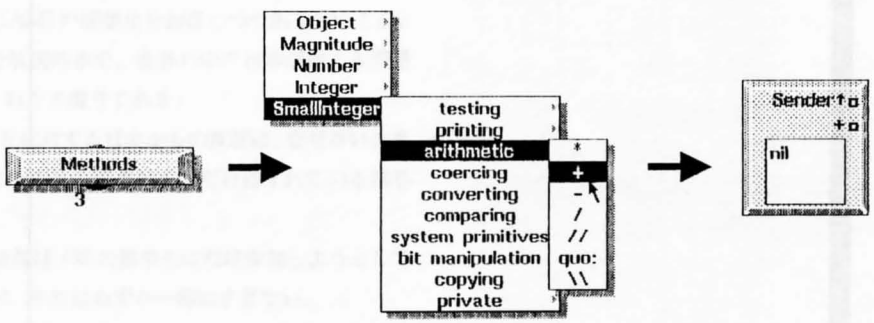
オブジェクトの登場のさせ方

オブジェクトは陰影のついた球で表され、下に付いているラベルには"3"と書かれています。この球＝オブジェクトはそれ自身がデータを持っており、かつそのデータに対する処理機能も持っています。例えば、ここに登場した'3'というオブジェクトは、内部状態が3であり、その他に演算や比較などの様々な機能を持っています。例えば足し算とか掛け算がそうです。

では、本当にそうであるかを調べ、さらには、その機能を起動するトリガであるメッセージセレクタを取り出してみます。

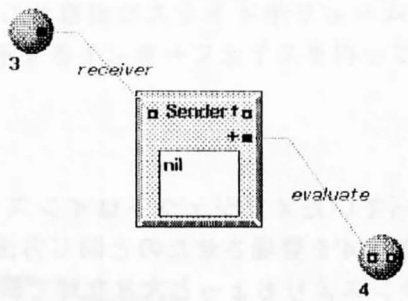
まず、'3'の上にインタラクタ(interacter)のひとつ'Methods'を乗せます。すると、'3'のクラス階層を表すメニューが現れます。このメニューから'3'はSmallIntegerのインスタンスであること、そして、'3'はSmallIntegerの持つ機能のみではなく、そのスーパークラスたちの機能を継承(インヘリタンス)していることがわかります。さらに、このメニューは階層化されており、例えばあるクラスを選択すると、そのクラスに含まれるプロトコルのリストが表示され、その中からプロトコルを選択すると、インスタンスメソッドのリストが表示されます。

そのメソッドのリストの中から使いたい機能を選択することによって、メッセージセレクタを取り出すことができます。ここではクラス'SmallInteger'のプロトコル'arithmetic'のメソッド'+'を選択しましょう。すると、四角くて'+'という名前のついたメッセージセレクタがビューに登場します。つまりSmallIntegerのインスタンス'3'は'+'というメッセージセレクタを持つメッセージに答える機能を持っているわけです。



メッセージセレクタが登場するまで

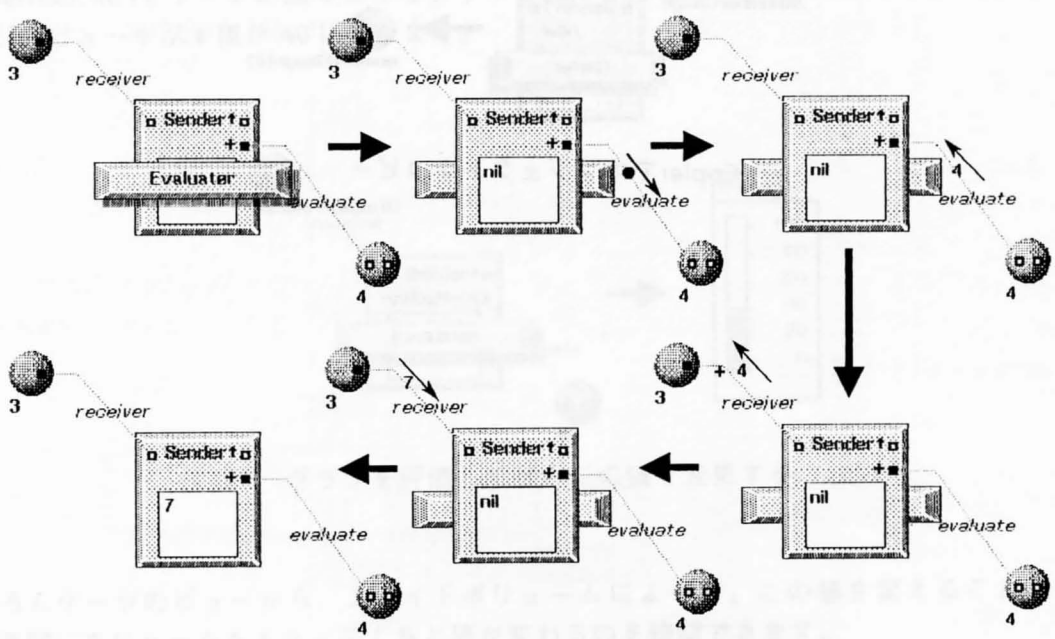
次に'4'を'3'と同じようにして作ってやります。これで、必要な役者がそろいましたから、2つのオブジェクトをメッセージセレクタを介してリンクで接続しグラフ化します。'Object World'の世界ではこの接続されたグラフ自体がSmalltalk-80のメッセージ式を表すことになります。このグラフはユーザが'3'に'+ 4'というメッセージを伝送することを表わします。



メッセージ式'3 + 4'を表すグラフ

では、このグラフを評価してみましょう。インタラクタのひとつである'Evaluater'をメッセージセレクタの上に持っていって乗せてやると評価シミュレーションが開始されます。

シミュレーションを開始すると、まず黒い小さな丸がメッセージセレクタから飛び出し、リンクに沿って、最初に'3'に、次に'4'に向かって移動していきます。これは住所の確認、つまりリンクにつながれているオブジェクトの所在を確認しているわけです。それを受けて'4'はメッセージ送信を開始し、'4'がリンクに沿って、メッセージセレクタへと移動していきます。メッセージセレクタに着いた'4'は'+'と合体しメッセージ'+ 4'となって'3'に向かって移動していきます。'+ 4'を受け取った'3'は演算を行い'7'と答えます。'7'はリンクに沿ってメッセージセレクタに戻って行き、メッセージセレクタの箱(テンポラリ変数領域)に格納されます。このような形で計算が進むのがSmalltalk-80の計算過程です。

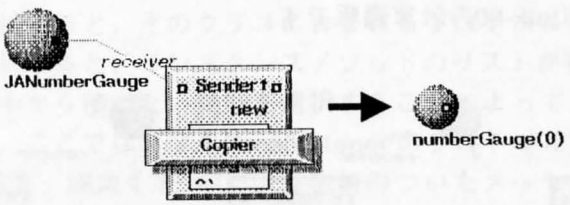


シミュレーションの過程

従来の手続き型言語の場合はあくまで'+'が主役であって、それに'3'と'4'をパラメータ(わき役)として与えてやることにより、その手続き'+'からの値をもらうというのが計算のやり方でした。しかし、Smalltalk-80の場合はこれと違って、オブジェクト(主役)である'3'自体が機能を持っており、'+ 4'というメッセージに対して'3'が答える、という方法をとっています。これをSmalltalk-80の式で表記した場合、単に'3 + 4'となり、この計算を行うプロセスが分かりづらいのですが、このシミュレーションを行うことによって大変分かり易くなっています。

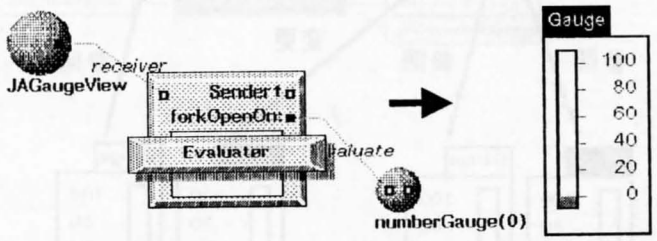
2. 絵によるプログラム

先程のシミュレーションで扱っていたオブジェクトはインスタンスでした。次にクラスをビュー上に登場させてみます。'3'や'4'を登場させたのと同じ方法でクラス'JANumberGauge'を登場させます。クラスはインスタンスよりちょっと大きな球で表されています。先程と同じように「何ができるの?」と聞いてみましょう。'JANumberGauge'に'Methods'を乗せてやると、例によって、そのオブジェクトが持っている機能をリストにしたメニューを答えてくれます。ここでは、JANumberGaugeのインスタンスを生成したいと思いますから、クラス'JANumberGauge class'のプロトコル'instance creation'のクラスメソッド'new'を選択します。すると、メッセージセレクタ'new'が登場します。次に'JANumberGauge'と'new'をリンクで接続し、'new'に'Evaluater'を上から乗せて評価してやります。すると、シミュレーションが開始され、0という値を持った'JANumberGauge'のインスタンス'numberGauge(0)'が返ってきて、メッセージセレクタの箱に格納されます。この箱はテンポラリな変数領域ですから、インタラクタのひとつである'Copier'を'new'の上に置くと箱に格納されている'numberGauge(0)'のコピーが1つ取り出せます。



Copierでオブジェクトをコピー

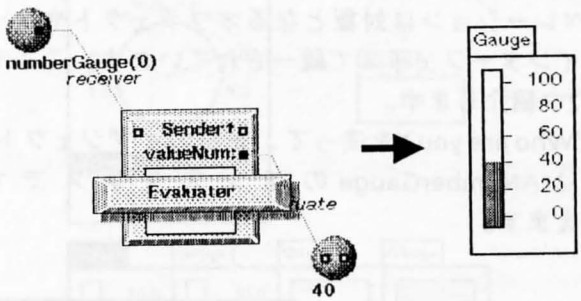
この'numberGauge(0)'をモデルとした独立したビューを新たに生成するために、'JANumberGauge'と組になっているビューのクラスである'JAGaugeView'を登場させます。'Methods'を用いて、新たなプロセスを作ってビューを開くためのメッセージセレクタ'forkOpenOn:'を登場させ、'JAGaugeView'と'numberGauge(0)'をリンクで接続します。次に'Evaluater'を'forkOpenOn:'の上に持っていって乗せてやるとシミュレーションが開始され、'Gauge'というラベルの付いた新たなビューが生成され表示されます。これがnumberGauge(0)をモデルとしたビューであり、これ自体がスライドボリュームによって自由に値を変更したり、バーの長さによって値を表示するインターフェイスを持っています。



グラフを評価してビューを開く

ここまででお分かりのように、ARK-likeの世界では絵を描くことが、即ちプログラミングになっています。また、Smalltalk-80におけるプログラムの実行はメッセージ式のテキストをセレクトして'do it'することですが、ARK-likeでは、'Evaluater'をメッセージセレクタに重ねることに他なりません。

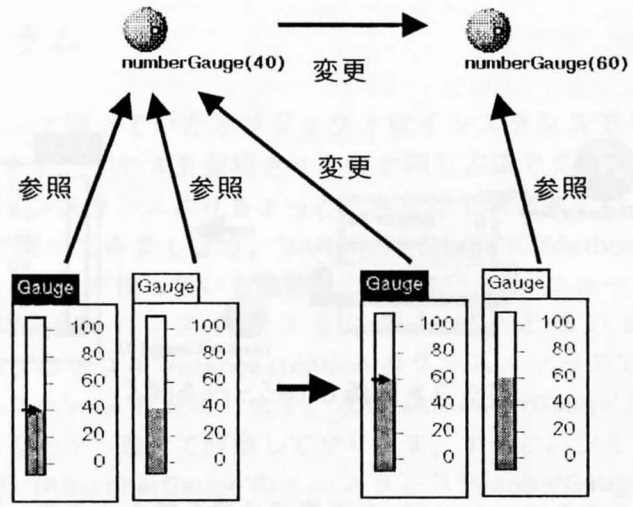
次に'numberGauge(0)'の保持する値にアクセスしてみます。現在は0がゲージの値として入ってますが、値を40に設定するには、'40'が値をセットしてくれというメッセージ('valueNum: 40')をゲージに送るようなグラフを描きシミュレーションします。そのとたんにゲージのビューが示す値が'40'になります。



グラフを評価してゲージの値を変更する

もちろんゲージのビューから、スライドボリュームによって、この値を変えることもできます。実際にボリュームをドラッグすると値が変わるのを確認できます。

今度は、もう1枚ゲージのビューを作ります('forkOpenOn:'をもう一度評価する)。そうすると先程のゲージのビューと同じ値を示します。これはこの2つのビューが同じモデルの保持する値を見ているからです。ですから、ゲージの値を例えば60に設定し直すと2つのビューには同じ60が表示されます。



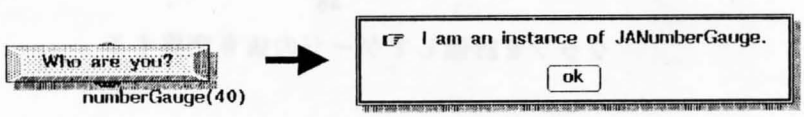
ゲージの連動

ここまでで、Smalltalk-80の実行過程とオブジェクトの計算モデル自身をシミューレートしたことになります。このように、ARK-likeは視覚的にSmalltalk-80のオブジェクトを管理できます。

3. インタラクタとゲージのいろいろ

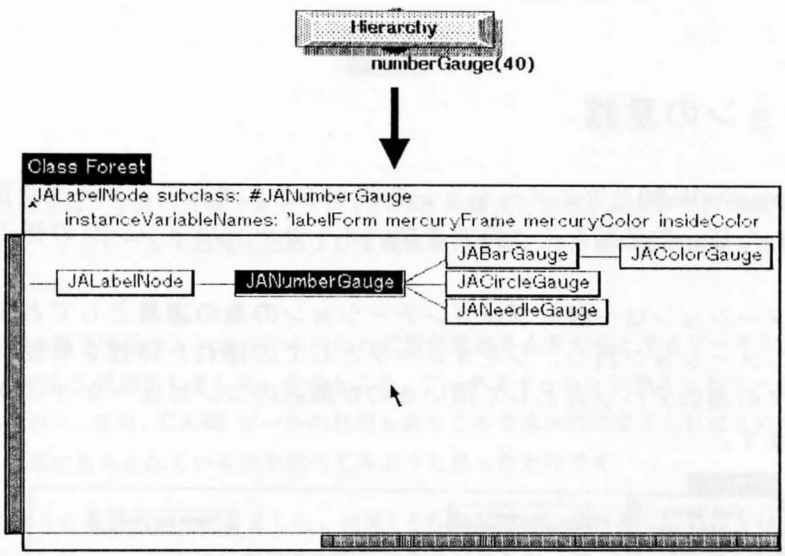
ARK-likeの基本的なオペレーションは対象となるオブジェクトやメッセージセレクタにインタラクタを重ねると言うインターフェイスで統一されています。ここではユーティリティとして便利なインタラクタを2つ紹介します。

例えば、インタラクタ'Who are you?'を使って、先程のオブジェクト'numberGauge(60)'に聞いてみると、「私はJANumberGaugeのインスタンスです。(I am an instance of JANumberGauge.)」と答えます。



オブジェクトに「君は誰だい?」と問いかける

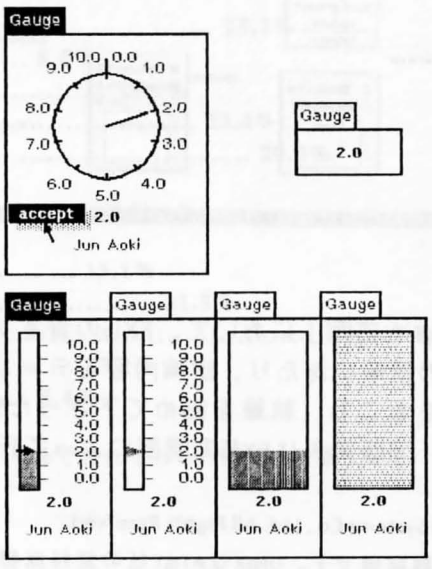
また、インタラクタ'Hierarchy'を使って聞いてやりますと、親からの機能の継承関係をグラフィカルなツリーで示したクラス階層ブラウザが開きます。



クラス階層ブラウザを開く

このツリーを見てみると、JANumberGaugeの親戚にはJABarGaugeとかJACircleGauge、JANeedleGauge、JAColorGauge等があることがわかります。これらのクラスに対して、クラス階層ブラウザからクラスブラウザ等を開き、実際のソースコードを参照、変更することができます。ソースコードを変更すると、そのオブジェクトの性質やふるまいはとたんに変わってしまうことになります。

最後に、このJANumberGaugeの親戚たちをモデルとするビューたちを、次の図に示します。これらは、みんな同じ値を参照しているために、みんな連動して動きます。



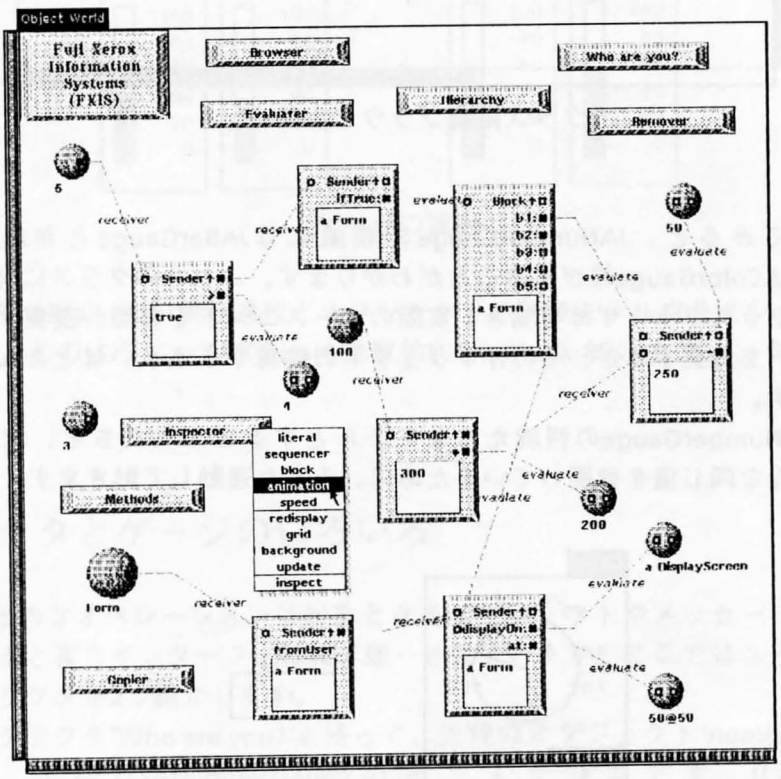
いろいろなゲージ

右下のJAColorGaugeはゲージの中に示される色の濃淡によって値を示しています。右上のJADiditalGaugeは数値によって値を示ます。例えばこれの値を6に変えてやると、すべてのゲージが指す値が6に変わります。これらのゲージを入力装置としたり、出力装置としたりすることによって、実行過程をさらにビジュアルに見せることができます。

4. アニメーションの意義

以上のように、Smalltalk-80とアニメーションを使うことによって、データが推移していく様子をつぶさに観察したり、本来見ることのできないメッセージパッシングの様子を見ることもできます。

このようなアニメーションは一般にプレゼンテーションの為の道具としてとえらえることが多いのですが、アニメーション持つ、シミュレータとしての優れた特性を有効に用いることによって、プログラマの為のデバッガとして用いるのが最近のコンピュータ･アニメーションの考え方であると言えます。



5. 謝辞

この原稿は「SIGENV Workshop in 盛岡」において、FXISの青木が行った15分間の講演内容をもとに加筆したものです。原稿の執筆にあたり、講演内容をテープ録音して下さった電力計算センターの渡辺氏、そのテープをおこし、執筆を勧めて下さったSRAの塩谷氏に紙面を借りて感謝いたします。また、再構成、図版挿入はFXISの阿部によって行われました。

ソフトウェア開発工程および支援環境に関する ミニ・アンケートの集計結果

編集部

1. はじめに

SEAMAIL 編集部では，２月の月例フォーラム（CASE の導入と活用）および３月の集中セミナーに参加された方々を対象に，ソフトウェア開発工程および支援環境の現状と技術者の意識調査を目的としたミニ・アンケートを実施しました．

質問表は，２年前に第２回日中シンポジウムのために野村敏次さんを中心とするワーキング・グループが使われたものを，簡略化して再利用しました．このところ，ワークステーションやネットワークの普及がかなりのスピードで進んでおり，また，CASE ツールの利用もあちこちで具体的に考えられはじめたので，そうしたことの影響がどんな風にあらわれているかを調べてみようと思ったわけです．

全部で 107 人の方々にお答えいただきました．結果（それほど詳しい分析をしたわけではなく，ほとんど単純集計をしただけですが）は，以下に示す通りです．去年のソフトウェア・シンポジウムで報告された野村さんの調査レポート（SEAMAIL Vol.4, No.4-5 に再録）と比較してみると，興味深いと思います．

2. 集計結果

Q1. あなたのソフトウェア開発経験年数は？

1 - 5年	32.7%
5 - 10年	40.2%
11 - 15年	14.0%
16年以上	13.1%

Q2. 最近あなたが（直接または間接に）その開発または管理を担当されたソフトウェアの種類は？（複数ある場合は主なもの１つ）

事務計算	25.2%
科学技術計算	5.6%
プロセス制御	13.1%
システムソフト	23.4%
その他	29.0%

Q3. そのソフトウェア開発での，あなたの役割は？

上級管理者	13.1%
直接の管理者	21.5%
設計者	43.9%
プログラマ	13.1%
その他	8.4%

Q4. そのソフトウェア開発には，全体でどれくらいの期間と工数（人月）がかかりましたか？

集計省略．

Q5. そのソフトウェアの規模は？

集計省略.

Q6. それぞれの開発工程におよそどの程度の工数がかかっていますか？ 大体の比率を教えてください.

要求分析／定義	..... 11.9%
基本設計	..... 13.3%
詳細設計	..... 17.7%
プログラミング／デバッグ	..... 24.7%
テスティング	..... 20.9%
ドキュメンテーション	..... 11.6%

Q7. そのソフトウェア開発で,トラブルが多く発生したのはどの工程ですか？ 問題の多い順に番号を振ってください.

要求分析／定義

1位	..... 21.1%
2位	..... 13.7%
3位	..... 8.4%
4位	..... 8.4%
5位	..... 20.0%
6位	..... 28.4%

基本設計

1位	..... 8.4%
2位	..... 16.8%
3位	..... 16.8%
4位	..... 32.6%
5位	..... 15.8%
6位	..... 9.5%

詳細設計

1位	..... 9.5%
2位	..... 12.6%
3位	..... 28.4%
4位	..... 15.8%
5位	..... 22.1%
6位	..... 11.6%

プログラミング／デバッグ

1位	..... 21.1%
2位	..... 29.5%
3位	..... 11.6%
4位	..... 16.8%
5位	..... 11.6%
6位	..... 9.5%

テスティング

1位 **40.0%**
2位 **26.3%**
3位 **14.7%**
4位 **5.3%**
5位 **10.5%**
6位 ... **3.2%**

ドキュメンテーション

1位 . **1.1%**
2位 .. **2.1%**
3位 **15.8%**
4位 **9.5%**
5位 **9.5%**
6位 **62.1%**

Q8. そのソフトウェア開発で発生したトラブルのなかで，一番困ったことは何でしたか？ 簡単に説明してください．

- 仕様もれの見積りができない（開発の人間のスキルにかなり依存する）．
- 詳細設計以降の作業を個人に分散したこと．その統括者（または総合テスト者）がシステムの内容を十分理解していなかったために，完成されたシステムに穴があいてしまった．
- ユーザの要求がスムーズに出てこない．ユーザ側のシステム化に対する認識不足と，ユーザ側の意見を統括する責任者の不在等の理由から，要求項目が全部出きらないうちに，基本設計に進まざるを得なかった．その結果，本番開始後にたくさんの改善要求が出てきた．
- 要求分析が不十分であったため，基本となるモジュールに機能変更が多発し，テスティング時間が予想をはるかに越えててしまった．
- 開発環境上で正しく動作(シミュレーション)していたものが，システム・テスト段階でトラブルを起こした場合，検証の方法がない．ハードウェアとソフトウェアの切り分け，自コンポーネントと他コンポーネントとの切り分け，特に，インタフェース上関連ある他コンポーネントの詳細なスペックがよくわからないことが多く，それを開発したグループとの調整がうまくいかない．
- バグが収束しない（バグが取れない）．
- サブシステム間で仕様があわなくなる．
- 開発目標が狂う．
- 他メーカ機器とのインターフェースがうまく取れなかった．原因を分析してみると，該当インタフェースについての知識不足から，仕様の取り決めが不十分であったことがわかった．
- 委員会による指導の不適切さとコミュニケーションの不足．
- 基本設計において具備すべき機能が洗い出されておらず，要求定義に後戻りすることがあった．
- 工程を急ぐあまり単体テストが不十分で，結局生産性を下げた．
- 試験運用の段階で，ユーザの現場サイドから種々の仕様変更が発生した．ユーザサイドでの仕様のレビューが原因だと考えられるが，開発サイドとしては，ユーザ内部の要求の取りまとめの段階からサポートしてやらなければいけないのかなと反省しています．
- システム機能を多数の設計者に理解させること（技術者の能力レベルのバラツキ）．
- 仕様変更による修正時のやり直しテスト範囲の特定．
- 要求定義が固まらない．
- スキル不足のために再コーディング．
- ユーザの要求がなかなか固まらず，何回も仕様書を書き直すことになった．
- ソフトウェア完成後の要求仕様とのくい違い．
- 分析から設計へなかなか移行できない（技法がない）．
- ハード/OS及びアプリ（これを担当）の同時開発のため，仕様が不明確であり，また仕様変更が多発する．
- OJTだったので，何から何まで初めてだった．ソフトウェア開発は環境から受ける制約が非常に大きいと思う．CASEには大いに期待している．

- 要求内容が確定的でなく，定義した内容に自信が持てなかった．
- システムおよびツールのバグ．
- 仕様が不明確．
- 大規模な仕様変更．
- ドキュメントの不備．
- 見積りの不備．
- 工程管理．
- デーモン･プロセスと一般ユーティリティ間でのタイミングによる競合．
- システム･ソフトウェアのバグ．
- 頻繁な仕様変更．変更の必要な個所があちらこちらに及ぶことによるバグの発生．
- 既存環境（この場合は X11）上での性能予測が困難だった．
- 相手先が海外の企業であり，遠隔地であるため，情報の授受がうまく行かず，導入時になって，仕様の違い等が指摘された．
- 設計思想の統一と徹底（漢字が出ない！）．
- 仕様が後々まで明確にならなかった（仕様変更によるバッグログ）．原因は，発注者と受注者（私）とのあいだで，仕様に対する責任が明確になっていなかったこと．
- 外注化（設計工程以下）した結果でき上がったソフトがまるで駄目で，基本設計からレビューし直すことになった．
- オブジェクト指向型プログラム言語を用いる場合のクラス構成およびメソット定義の基準．
- システム全体について知っているものがいない．群盲像を撫でるが如し．
- デバッグ環境の悪さ．
- 各プログラム開発者のスキルの差．
- コミュニケーション．
- ユーザがいったようにできていない．
- 納期が間に合わない．
- 利益がない．
- 数年後のメンテナンスに対応できない．
- 要求分析に起因する手戻り作業．
- ユーザ要求がうまく定義できない．システムとして動作し始めてから，初めて問題が表面化する．ユーザ要求定義時に要求自信に誤りがあっても，発見できない場合がある．そのために，あとで多大な手もどりが発生する．
- 上流工程のトラブルが下流工程に大きく拡がったこと．
- 予定通りのレスポンスが得られなかったこと．
- ユーザーの要求がよく変わったこと．
- システムの規模が大きいため，全体が見わたせない．要するに，どこで何をやっているのかよく分からない．
- ターゲット･マシンの基本ソフトにバグが多かった．プログラマが，そのマシンに対して十分な知識を持っていなかった．制御対象機器にハードウェアのバグが多く，総合テストに長くかかった．
- 開発言語の学習と並行していたので，ある機能を実現するのにどうコーディングしてよいか，わからないことがあった．
- マルチプロセスのデバッガがなかったこと．
- エンドユーザからの要求条件を収集している最中に，要求自体が変化してしまった．
- 要求定義の不確定．
- できあがったものが，ユーザが思っていたものとは違っていた．詳細設計での意識の相違があった．
- ユーザ自身の要求が不徹底であったこと．その不明確を予めチェックできるほどの対ユーザ･スキルがわれわれになかったこと．
- 要求仕様が固まらない．
- 詳細設計で明確にされていない部分が，プログラム開発時に大きな問題となった．
- 仕様決定（要求仕様）のあいまいさ．
- 対応できかねる仕様変更要求．
- オンライン回線トラブル．
- ソフトウェアの完了判断をどうやって定量的に把握するか．
- 仕様変更と障害との切分け．

- 新しい開発環境だったので，慣れるまでに時間がかかった古い開発環境にもとづく設計手順で初めの頃は作業したが，うまくいかなかった．
- 要求を整理する手段がない．ワープロよりも，原始的なカードのほうが便利だったりして…．ハイパーカードを使わないのがまずいのかなぁ?
- 新しいツールや言語（パッケージ）の限界が見えるまでが，やっぱり苦労する．
- 利用者にとって必要不可欠な機能がまったく抜けていることが，テスト工程の中盤になってわかり，その機能を盛り込むことになった．
- VM の下で異なるユーザ間で VSAM ファイルを共有するような要求定義を満足させるための工夫（アイデア）を出すのがむずましかった．
- システム･ソフトウェアのダウン．
- OS のシステム･ダウン．
- Smalltalk（オブジェクト指向）の理解度の差から生じるプログラム品質のバラツキ．
- テストする人材がいなかった．
- 量産後のバグ発覚．
- 量産後のユーザ仕様変更要求．
- OOD における要求定義書の作成方法．オブジェクトの抽出など，OOD の設計作業を進めていくそれぞれの工程で，試行錯誤を何度もくりかえした．
- 基本設計書および詳細設計書であいまいな所を残したまま，プログラミングに進んだところ，その部分がテスト時にトラブったため，設計（インターフェイス）の見直しを行なわざるを得なかった．
- コーディング不良．後のメンテナンスのことを考えると，不用なコードや，もっと簡潔にできるのにわざわざ難しくしている部分があった．
- 開発中（というよりメンテナンス時）に，各サブシステムの機能的独立性が低く，エラーが発生し易かった．
- Unix カーネルをマルチ CPU 対応にするため，オリジナルのカーネルを完全に把握する必要があったが，不十分なまま開発に入った．
- プログラミング～テストで見積り規模が２倍になった．
- 利用者に役立つソフトとしての提供（使いやすさ/機能の十分性）．
- 開発人員が確保できず，何ヶ月かの単位でメンバーが替わった．さらに，ほとんどが開発対象システムの内容について素人（新人）の集団を使ってプログラミングを行なわざるを得なかった．
- 基本設計のレビューが甘く，αリリース後の設計変更．
- ユーザ要求に例外処理のもれがあった．
- ユーザ側とメーカ側とのあいだで，仕様に関して見解の相違があった．同様なシステムは何回が手掛けており，過去の経験にもとづいて解釈したが，ユーザ独自の見解を確認することをおこたったため，精度上のトラブルが発生．これがテスト段階で発覚したので，やむなく，設計方針（思想）に目をつぶって改修した．
- 要求定義では，より詳細に記述すべきと思う．性能等品質に係わるものすべてが記述できればと思う．
- プロジェクト間の意思の疎通が欠けているため，同じ機能をもつプログラムがいくつもできてしまった．
- 要求機能のあいまいさ，プロジェクト後半での仕様変更．
- ユーザ要件が明確でなく，ソフトウェアとして何が達成されればプロジェクトは成功といえるのかが，はっきりしなかった．
- 外注が３社参画し，その間の立場の違いや利害の調整に苦労した．
- 問題が要求として明確にできなかったこと．
- 要求分析または定義の点で，開発内容（作業量）と予算の関係をユーザに理解していただくのに困った．開発工数で一番ウエイトの重いテステイングのフェーズを軽くできるような支援環境を利用することにより，開発内容と予算とのギャップを埋めたいと考えている．
- 詳細設計の段階でプログラム構造が確定できず，実際の開発に最も適したモジュール構成がつかめなかった．
- 開発（コーディングおよび単体テスト）完了後にも仕様修正が多く，フェーズが混乱した．
- ドキュメントの解釈の相違によるモジュール間インタフェースの不良により，システム･テスト段階で大幅な詳細計仕様の変更を余儀なくされた．

Q9. 今後，類似ののシステム開発において，体系的な（またはソフトウェア工学的な）方法論を利用することによって，各工程の改善はどの程度可能でしょうか？ 下の表の該当欄にチェックしてください．

要求分析／定義

大いに	31.8%
いくらか	30.8%
あまり	15.0%
ほとんど	6.5%
不明	0.9%

基本設計

大いに	30.8%
いくらか	33.6%
あまり	12.2%
ほとんど	5.6%
不明	2.8%

詳細設計

大いに	18.7%
いくらか	41.1%
あまり	17.8%
ほとんど	3.7%
不明	3.7%

プログラミング／デバッグ

大いに	20.6%
いくらか	36.5%
あまり	17.8%
ほとんど	5.6%
不明	4.7%

テスティング

大いに	16.8%
いくらか	42.1%
あまり	15.9%
ほとんど	5.6%
不明	4.7%

ドキュメンテーション

大いに	22.4%
いくらか	42.1%
あまり	10.3%
ほとんど	6.5%
不明	3.7%

Q10. 同じく，適当な開発支援ツール（いわゆる CASE）を利用することによって，各工程の改善はどの程度可能でしょうか？　下の表の該当欄にチェックしてください．

要求分析／定義

大いに	30.8%
いくらか	31.8%
あまり	12.2%
ほとんど	9.4%
不明	1.9%

基本設計

大いに	29.9%
いくらか	34.6%
あまり	13.1%
ほとんど	.5.6%
不明	2.8%

詳細設計

大いに	25.2%
いくらか	33.6%
あまり	17.8%
ほとんど	4.7%
不明	4.7%

プログラミング／デバッグ

大いに	32.7%
いくらか	28.0%
あまり	13.1%
ほとんど	7.5%
不明	4.7%

テスティング

大いに	25.2%
いくらか	42.1%
あまり	11.2%
ほとんど	4.7%
不明	2.8%

ドキュメンテーション

大いに	35.5%
いくらか	33.6%
あまり	8.4%
ほとんど	2.8%
不明	5.6%

Q11. 次に，ソフトウェア開発技術や環境の改善に関連する一般的なことがらについての御意見をおうかがいします．

Q11-1. 下の表は，ソフトウェア開発の現状における問題点をランダムにリストアップしたものです．これらのうちで，あなたがもっとも問題だと思われるものを**３つだけ選んでください．**

(c) 開発要員が質・量ともに不足している	40.0%
(h) 設計の方法論や技法が体系化されていない	31.0%
(b) 開発工数や費用の見積りがいいかげんである	26.0%
(e) 技術者教育が十分になされていない	24.0%
(d) 管理者教育が十分になされていない	11.0%
(k) 中間成果物のレビューが不完全である	10.0%
(o) 品質の評価や管理のための技術が整備されていない	9.0%
(a) オフィス・スペースなど物理的な作業環境が悪い	8.0%
(i) 必要な開発支援ツールがそろっていない	7.0%
(j) 端末やワークステーションの台数が足りない	7.0%
(p) プロジェクト開始時の計画が不十分である	7.0%
(l) テストや検証がきちんとなされていない	6.0%
(m) ドキュメントのコンピュータ化が遅れている	4.0%
(f) 研究開発成果の産業界への技術移転が遅れている	3.0%
(q) プロジェクトの工程管理がきちんとなされていない	3.0%
(r) 保守のことを考慮した開発が行われていない	2.0%
(g) 構造的プログラミング技法が普及していない	1.0%
(s) 要員のスキル管理がきちんとなされていない	1.0%
(n) コンピュータ・ネットワークの整備が遅れている	0.0%
(t) 要求仕様があいまいである	0.0%

Q11-2. あなたは，ソフトウェア開発生産性の向上の要因として何が重要だと思われますか？　下の表にリストアップされた項目のなかから，あなたが特に重要だと思われるもの，また，あまり重要ではないと思われるものを，**それぞれ３つずつ選んでください．**

Q11-2a. 生産性向上要因として重要である

(a) 一貫した方法論にもとづく統合的支援環境の整備	51.0%
(c) 開発要員の質的レベルの向上	50.0%
(r) 要求分析・定義を支援するツールの整備・拡充	33.7%
(t) レビュー等による品質管理の徹底	23.5%
(o) テスト支援ツールの整備・拡充	17.4%
(j) 設計支援ツールの整備・拡充	15.3%
(s) プログラミング支援ツールの整備・拡充	14.3%
(m) 中堅技術者教育の充実	13.3%
(p) ドキュメントの完全なコンピュータ化	11.2%
(d) 開発要員のスキル・インベントリの拡充	10.2%
(f) 開発用ハードウェア・リソースの拡充	9.2%
(b) オフィス・スペースなど物理的作業環境の改善	8.2%
(g) 管理者教育の充実	8.2%
(h) 工程管理の改善	7.1%
(n) 定量的プロジェクト管理の確立	6.1%
(q) 見積り技術の改善	6.1%
(k) 新入社員教育の充実	5.1%
(l) 端末やワークステーションの大量導入	5.1%
(i) コンピュータ・ネットワークの整備・拡充	2.0%
(e) 開発要員の量的な不足の解消	1.0%

Q11-2b. 生産性向上要因としてあまり重要でない

(e) 開発要員の量的な不足の解消 43.9%
(q) 見積り技術の改善 33.7%
(l) 端末やワークステーションの大量導入 23.5%
(b) オフィス・スペースなど物理的作業環境の改善 21.4%
(h) 工程管理の改善 20.4%
(k) 新入社員教育の充実 19.4%
(p) ドキュメントの完全なコンピュータ化 17.4%
(n) 定量的プロジェクト管理の確立 16.3%
(g) 管理者教育の充実 13.3%
(f) 開発用ハードウェア・リソースの拡充 10.2
(i) コンピュータ・ネットワークの整備・拡充 10.2%
(d) 開発要員のスキル・インベントリの拡充 8.2%
(a) 一貫した方法論にもとづく統合的支援環境の整備 6.1%
(t) レビュー等による品質管理の徹底 6.1%
(m) 中堅技術者教育の充実 5.1%
(s) プログラミング支援ツールの整備・拡充 ... 3.1%
(r) 要求分析・定義を支援するツールの整備・拡充 ... 3.1%
(j) 設計支援ツールの整備・拡充 .. 2.0%
(o) テスト支援ツールの整備・拡充 . 1.0%
(c) 開発要員の質的レベルの向上 0.0

Q11-2c. プラス・ファクタ（[重要である] − [重要でない] >ゼロ）

(c) 開発要員の質的レベルの向上 50.0
(a) 一貫した方法論にもとづく統合的支援環境の整備 44.9%
(r) 要求分析・定義を支援するツールの整備・拡充 30.6%
(t) レビュー等による品質管理の徹底 17.4%
(o) テスト支援ツールの整備・拡充 16.4%
(j) 設計支援ツールの整備・拡充 13.3%
(m) 中堅技術者教育の充実 13.3%
(s) プログラミング支援ツールの整備・拡充 11.3%
(d) 開発要員のスキル・インベントリの拡充 10.2%

Q11-2d. マイナス・ファクタ（[重要である] − [重要でない] >ゼロ）

(f) 開発用ハードウェア・リソースの拡充 . − 1.0%
(g) 管理者教育の充実 ..... − 5.1%
(p) ドキュメントの完全なコンピュータ化 ..... − 6.2%
(i) コンピュータ・ネットワークの整備・拡充 ..... − 8.2%
(n) 定量的プロジェクト管理の確立 ..... − 10.2%
(b) オフィス・スペースなど物理的作業環境の改善 ..... − 13.2%
(h) 工程管理の改善 ..... − 13.3%
(k) 新入社員教育の充実 ..... − 14.3%
(l) 端末やワークステーションの大量導入 ..... − 18.4%
(q) 見積り技術の改善 ..... − 28.6%
(e) 開発要員の量的な不足の解消 ..... − 42.9%

Q11-3. これまで，ソフトウェア開発の改善に関しては，さまざまな新しいアイデアや方法論が提案され，またそれにもとづくツールが試作／開発されてきました．下の表にリストアップされた方法論やツールの実用化／普及の程度や開発現場での有用性について，あなた自身の評価をお聞かせください

Q11-3a. 実用化／普及について

1. 新しい高水準言語（Ada など）

すでに普及している 6.5%
普及はこれから 38.3%
まだ研究開発段階にある 18.7%
わからない 15.9%
よく知らない 20.6%

2. COCOMO 見積り技法

すでに普及している 3.7%
普及はこれから 13.1%
まだ研究開発段階にある 11.2%
わからない 17.8%
よく知らない 54.2%

3. オブジェクト指向プログラミング

すでに普及している 10.3%
普及はこれから 40.2%
まだ研究開発段階にある 29.0%
わからない 9.4%
よく知らない 11.2%

4. 構造化分析・設計

すでに普及している 36.5%
普及はこれから 44.9%
まだ研究開発段階にある 6.5%
わからない . 0.9%
よく知らない 11.2%

5. アプリケーション・ジェネレータ

すでに普及している 9.4%
普及はこれから 18.7%
まだ研究開発段階にある 26.2%
わからない 15.0%
よく知らない 30.8%

6. 実行可能型（Operational）仕様

すでに普及している .. 1.9%
普及はこれから .. 1.9%
まだ研究開発段階にある 15.9%
わからない 18.7%
よく知らない 61.7%

7. ジャクソン法（JSP & JSD）

すでに普及している 20.6%
普及はこれから 27.1%
まだ研究開発段階にある 5.6%
わからない 14.0%
よく知らない 32.7%

8. ソフトウェア開発用エキスパート・システム

すでに普及している . 0.9%
普及はこれから 11.2%
まだ研究開発段階にある 58.9%
わからない 10.3%
よく知らない 18.7%

9. ハイパーテキスト

すでに普及している 9.4%
普及はこれから 33.6%
まだ研究開発段階にある 17.8%
わからない 14.0%
よく知らない 25.2%

10. UNIX

すでに普及している 79.4%
普及はこれから 8.4%
まだ研究開発段階にある .. 1.9%
わからない 0.0%
よく知らない 10.3%

11. 第４世代言語

すでに普及している 34.6%
普及はこれから 29.9%
まだ研究開発段階にある 10.2%
わからない 8.4%
よく知らない 16.8%

12. テスト・カバレージ分析

すでに普及している 23.4%
普及はこれから 23.4%
まだ研究開発段階にある 10.3%
わからない 16.8
よく知らない .. 2.0%

13. ラピッド・プロトタイピング

すでに普及している 7.5%
普及はこれから 21.5%
まだ研究開発段階にある 29.0%
わからない 8.4%
よく知らない 33.6%

14. ソフトウェア信頼性推定モデル

すでに普及している .. 1.9%
普及はこれから 14.0%
まだ研究開発段階にある 27.1%
わからない 15.0%
よく知らない 42.1%

15. UpperCASE ツール

すでに普及している 0.0%
普及はこれから 41.1%
まだ研究開発段階にある 33.6%
わからない 5.6%
よく知らない 19.6%

16. LowerCASE ツール

すでに普及している 17.8%
普及はこれから 34.6%
まだ研究開発段階にある 21.5%
わからない 7.5%
よく知らない 18.7%

17. Σシステム

すでに普及している . 0.9%
普及はこれから 22.4%
まだ研究開発段階にある 28.0%
わからない 28.0%
よく知らない 20.6%

有用性の評価について

1. 新しい高水準言語（Ada など）

きわめて有用だと思う 11.2%
ある程度は有用であろう 43.9%
あまり有用ではない 7.5%
わからない 15.9%
よく知らない 21.5%

2. COCOMO 見積り技法

きわめて有用だと思う ... 2.8%
ある程度は有用であろう 20.6%
あまり有用ではない 4.7%
わからない 17.8%
よく知らない 54.2%

3. オブジェクト指向プログラミング

きわめて有用だと思う 43.9%
ある程度は有用であろう 28.0%
あまり有用ではない ... 2.8%
わからない 12.2%
よく知らない 13.1%

4. 構造化分析・設計

きわめて有用だと思う 40.2%
ある程度は有用であろう 40.2%
あまり有用ではない 6.5%
わからない .. 1.9%
よく知らない 11.2%

5. アプリケーション・ジェネレータ

きわめて有用だと思う 12.2%
ある程度は有用であろう 35.5%
あまり有用ではない 7.5%
わからない 14.0%
よく知らない 30.8%

6. 実行可能型 (Operational) 仕様

きわめて有用だと思う 4.7%
ある程度は有用であろう 10.3%
あまり有用ではない 3.7%
わからない 21.5%
よく知らない 59.8%

7. ジャクソン法 (JSP & JSD)

きわめて有用だと思う 3.7%
ある程度は有用であろう 41.1%
あまり有用ではない 10.3%
わからない 13.1%
よく知らない 31.8%

8. ソフトウェア開発用エキスパート・システム

きわめて有用だと思う 9.4%
ある程度は有用であろう 33.6%
あまり有用ではない 14.0%
わからない 21.5%
よく知らない 21.5%

9. ハイパーテキスト

きわめて有用だと思う 23.4%
ある程度は有用であろう 31.8%
あまり有用ではない 4.7%
わからない 14.0%
よく知らない 26.2%

10. UNIX

きわめて有用だと思う 43.0%
ある程度は有用であろう 35.5%
あまり有用ではない 8.4%
わからない ... 2.8%
よく知らない 10.3%

11. 第4世代言語

きわめて有用だと思う 19.6%
ある程度は有用であろう 42.1%
あまり有用ではない 11.2%
わからない 10.3%
よく知らない 16.8%

12. テスト・カバレージ分析

きわめて有用だと思う 13.1%
ある程度は有用であろう 30.8%
あまり有用ではない 12.2%
わからない 17.8%
よく知らない 26.2%

13. ラピッド・プロトタイピング

きわめて有用だと思う 20.6%
ある程度は有用であろう 33.6%
あまり有用ではない ... 2.8%
わからない 10.3%
よく知らない 32.7%

14. ソフトウェア信頼性推定モデル

きわめて有用だと思う 4.7%
ある程度は有用であろう 25.2%
あまり有用ではない 10.3%
わからない 17.8%
よく知らない 42.1%

15. UpperCASE ツール

きわめて有用だと思う 27.1%
ある程度は有用であろう 42.1%
あまり有用ではない ... 2.8%
わからない 7.5%
よく知らない 20.6%

16. LowerCASE ツール

きわめて有用だと思う 16.8%
ある程度は有用であろう 51.4%
あまり有用ではない 5.6%
わからない 6.5%
よく知らない 19.6%

17. Σシステム

きわめて有用だと思う . 0.9%
ある程度は有用であろう 13.1%
あまり有用ではない 44.9%
わからない 20.6%
よく知らない 20.6%

SIGENV 1990年度の活動予定について

渡邉 雄一
アスキー

1. 昨年の活動概要と反省から

環境分科会(SIGENV)では，一昨年から昨年にかけて「プロトタイピング」をテーマに一連の討論会を開催してきました．その一応の成果は SEAMAIL の特集号(Vol.4,No.2-3)として以前に発表したとおりです．しかし，この活動も段々と停滞してきました．

その原因の1つは，もともと世話人が多忙で SEA の活動に時間が十分割けないような状態であったにもかかわらず，事例収集と地方会員との交流を意図したフォーラムを開催する(しかも合計3回も)などという無謀な行動に走ってしまったことにあります(もちろんフォーラムそれ自体はいずれも成功でしたが....)．また，プロトタイピングを支援する環境やツールとして，際立ったものが現れなかったことも，現場の技術者という立場に立つわれわれの興味を弱める一因ではありました．

しかし，メンバ個々人の知識整理の時間として，SIGENV での討論は極めて貴重なものであったと考えています．そして，最近の CASE ツールの充実を見ていると，あらためてプロトタイピングを論ずる機会が，また近いうちにやってきそうに思われます．いずれ適当なタイミングをはかって(再び·しぶとく·地道に)このテーマに挑戦する場を設けたいと思っています．実際に，アメリカでは興味深い資料が2つ出ました．

[1] COMPUTER,IEEE (1989.5) (Rapid Prototyping の特集号です)

[2] R.P.Gabriel (editor), R.M.Balser et al.: Draft Report on Requirements for a Common Prototyping S ystem, SIGPLAN NOTICES, Vol.24,No.3 (1989.3)

特に[2]は，まだドラフトの段階ですが，昨年の OOPSLA89 でパネルに立った Balser は，これを示していろいろとオブジェクト指向の問題点やプロトタイピングとの融合の難しさを述べたそうです．このあたりのフォローも ENV の中で，輪講形式で進めて行きたいところです(出典から想像できるでしょうが，がっちり書かれている物なので，英語の苦手な私など一苦労なのです)．

ところで，「今までの活動の中で欠けていた要素(話題)は何だろうか？」と，みんなで考えてみたところ，いくつか意見が上がりました．

(1) 言語の話題がない

たしかに言語の話題はほとんどありませんでした．しかしこれを話題の中心にするにはかなりの覚悟が必要で，そこまでは踏み込めなかったというのが，偽らざる本音です．

(2) システムの事例が少ない

某国家プロジェクトの(悪)影響もあってか，国内では，昔ほどに自分たちのアイディアにもとづいたツール試作の試みや，システム開発の話題が少ないように思います．また，プロトタイピング手法によるアプリケーション開発の事例研究についても，われわれ(SIGENV)の考えるプロトタイピングの観点をまず明確にし，その立場からさまざまな事例の良し悪しを論ずるというよりは，結局のところ，ただケース·バイ·ケースで個々の事例を考えることしかできていないのではないか，という意見もありました．

(3) 派手な活動は必要ない

フォーラムなどは確かに SEA の存在を宣伝する効果を持っています．実際 ENV で主催した地方のフォーラムでは，いずれも新規会員の獲得に成功していまず．しかし，ある意味で宣伝を兼ねたような催しは，やはり SEA 本体の企画するイベントで十分な気もします．分科会は，そのメンバの合意の上で好きに活動するという本来の趣旨に立ち戻り，今年度は特に人集めに苦労する企画は避けて，じっくりメンバが納得いく運営を心掛けたいと思っています．

2. 具体的活動予定 ―オブジェクト指向―

時には言語の問題も扱えて，でもやっぱり ENV だから環境の話題を中心にしたいし，などと欲張ったことを満たしてくれる話題として，今年度は OOA(Object Oriented Analisys)を取り上げることにしました．その目論見の一

端を披露すると，

(1) 言語の話題には事欠かない

Smalltalk80 や C++ など，話題は十分過ぎるほどあります．

(2) システムの事例も豊富にある

特にそれぞれの言語の支援環境は注目すべきでしょう．また，OODB の適用事例などは，きちんとフォローをしていきたいと思っています．また，海外の展示会などで比較的容易に入手できるデモプログラムも，それなりに説得力があって楽しめますし....

(3) 勉強会を進めていくのに適当な書物が揃ってきた

SIG のレベルでやる勉強会ですから，あまり飛んだ話題や内容のものは辛すぎます．しかし，ある程度，現実の泥臭い話題からは逃避したいものです．この絶妙のバランス感覚（？）を満たしてくれるような資料を選ぶのに困らないほど，OOL 関係の書籍が本屋の店頭を賑わせてくれています．

分科会の活動は，昨年度と同様に「勉強会」形式を中心にします．また，せっかくみんな多忙な中を集まってくるのですから，できるだけ有意義な時間を過ごせるように，勉強のペースは早くなく遅くなく進めて行けるよう努力するつもりですので，興味を持たれている方は，ぜひ「在宅会員」としてでも結構ですから，参加していただければと思います．おおよそのスケジュールは以下の通りです：

まず，これから７月までは，OOA の勉強会と情報交換会をやります．

この時期は，夏合宿に向けての準備期間として，OOA の勉強会と OOL に関する情報交換を目的として開催することにしました．材料は，

［３］ Bertrand Meyer : Object-oriented Software Construction, Prentice Hall (ISBN 0-13-629031-0)

です．

この本は（あのパリのエッフェル塔にちなんで名付けられた）Eiffel という静的な型付けを行うオブジェクト指向言語でシステムを記述するとともに，オブジェクト指向分析，設計について論じた本です．Eiffel についての日本語文献としては，雑誌論文ですが，

［４］ Bertrand Meyer : Eiffel 環境, UNIX MAGAZINE,Vol.3,No.12,pp99-107 (1988.12)

があります．

また，文献［３］に関しては，

［５］ 高田広章: 書評, 情報処理, Vol.31,No.2 pp.280-281 (1990.2)

が出ましたので，それも参考にしていただけると幸いです．

８月は夏休みで休会です（９月の宿題を各自こなすので大変だろうから！？）．

９月は，OOA の集中勉強会を，２泊３日程度の合宿形式で実施します．この合宿は，首都圏のメンバも，近畿圏のメンバも共に参加しやすい場所として，名古屋近郊の保養施設で開催することを検討しています．参加の条件は次の２つです：

(1) 次の文献［６］から１～２編の論文を事前に読んでくること．

［６］ Won Kim,Frederic H. Lockovsky : Object-Oriented Concepts,Database, and Applications, Addison-Wesley (ISBN 0-201-14410-7)

みんなが同じ論文を読んできてもあまり意味がないので，５月の連休明けにはその分担を決める予定です．

(2) ほどほどに酒が呑めること

せっかくみんな仕事のやり繰りをして，集まるのですから，美味い酒と肴を楽しみたいものです

今回は，自分たちのための勉強会としての合宿をするので，フォーラムやセミナーとは趣旨が異なります．よって特に宣伝活動はしないつもりです．

１０月以降は未定ですが，OOPSLA'90 に参加された方をゲストとしてお招きして，その様子を話していただこうかなぁなどと考えていますが....

3. 終わりにかえて

いろいろと予定を書きましたが，世話人の力不足のために，一歩間違えばどこかで空中分解する危険性を多分に秘めている企画です．とにかく，できるところまで進めてみたいと思っています．また，あくまで勉強会としての目標を最初から下げないために，横文字の本を材料に掲げていますが，適当なものがあればそちらに鞍替えする可能性も十分にあります．OOA を一緒に勉強したいという方はぜひ分科会を覗きに来て下さい．

SIGEDU 1990年度の活動について

中園 順三（富士通 BSC）
杉田 義明（SRA）

1989年度最後のSIGEDUの会合を，この３月２０日に開催しました．もともとこのミーティングは，１年の活動を振り返ってあれこれと反省しながら，次年度の計画を立案するために設けられていたものです．常連のメンバーに混じって初めての方も２人ほど参加されて，合計１２名の方々でいろいろと話を交わしました．以下はこのミーティングの議事録であり，昨年の活動を踏まえて設定した今年度の計画概要です．

１．1989年度の活動記録：

１）４月１２日（水）at: SRA
テーマ:「採用の効果的な活動」
講師: 平本 巌（電力計算センター）
参加者: １４名

２）５月１７日（水）at: SRA
テーマ:「SEの資質と教」
講師: 山口 圭一（SRA）
参加者: １８名

３）６月２８日（水）at: 日本電子専門学校
テーマ:「学校内でSE教育は可能か」
講師: 河村 一樹，沖山 豊（日本電子専門学校）
参加者: １５名

４）７月２６日（水）at: SRA
テーマ:「SI技術」
講師: 杉田 義明（SRA）
参加者: １０名

５）８月３１日（木)ー９月２日（土）
場所: 熊本県阿蘇山麓
テーマ:「ソフトウエア設計教育」
参加者: ３４名

６）１０月１８日（水）at: シーイーシー
テーマ:「効果的なOJT」
講師: 大浦 洋一（シーイーシー）
参加者: ９名

７）１１月１５日（水）at: SRA
テーマ:「採用時の業者テストと入社後の評価」
講師: 鴇田 昭夫（川鉄システム開発）
参加者: １７名

８）１２月１３日（水）at: SRA
テーマ:「プレゼンテーション技術教育の実践」
講師: 篠崎 直二郎（日本電気ソフトウエア）
参加者: １１名

９）１月１７日（水）at: SRA
テーマ:「知的財産権と教育」
講師: 中山 照章（富士通 SSL）
参加者: ５名

１０）２月２１日（水）at: シーアンドシーコンピュータ学院
テーマ:「CALの実践」
講師: 古川 則雄（シーアンドシーコンピュータ学院）
参加者: １２名

１１）３月２０日（火）at: SRA
テーマ:「1990年度SIGEDU活動計画」
参加者: １２名

１年間の活動を振り返って眺めてみると，いろいろと反省したり，感動したりとなかなか楽しかった．その主なものは以下の通り：

- 新入社員教育に関する討論の場が欲しかった（特にヒューマンウエアの部分）．
- 分科会の後の飲み会の情報が有効であった．
- 教育現場の見学会など具体的な内容を取り上げて有効であった．
- 専門学校における実施の例やOJTの実体験が印象に残っている．
- ソフトウエア業界の社会でのあり方も教育にかかっているのではないか．
- ワークショップで普段聞けない話がありよかったが，運営にもうひと工夫が必要ではないか（特に常連でない人への配慮が必要）．

・活動が活発ではあるが，内容がいまいち．メンバーが固定化すると内輪だけになり，新鮮さに欠ける．
・もっと外部と交流を．
・外部からの話題提供者を積極的に呼ぼう．
・ドキュメント化がなされていない，他の SEA メンバーへもっとアピールして，参加を呼びかけよう．
・発表をやりっぱなしにしないで，何らかの形でまとめよう．

2．1990 年度の活動方針：

以上のような話をしながら，今年度の活動を検討した結果，すばらしい方針ができあがりました．

1）毎月の運営（講師の設定，レポートの作成，その他）は月当番が責任をもって行う
2）話題提供者は常に外部から招待し，まとめを必ず行う
3）まとめの結果は SEAMAIL に掲載する
4）講師の謝礼は基本５，０００円として最大１０，０００円，交通費は別に調整する
5）テーマは教育だけにこだわらない
6）月例会の活動日と当番を次のように決めました．

４月２５日（水）篠崎
５月２３日（水）杉田
６月２７日（水）平本
７月２５日（水）河村
９月２６日（水）川辺
１０月２４日（水）山口
１１月２８日（水）鴇田
１２月１９日（水）中園
１月２３日（水）大浦
２月２７日（水）中山
３月２７日（水）（年度総会）

3．ワークショップに関して：

恒例のワークショップに参加することを，心待ちにしておられる方がおられることと思いますが，今年は何と韓国で実施することを計画しています．SIGEDU のアクティブなメンバーの李さんの強力なお薦めがあって，彼の地元で開催することの詳細を詰めています．

計画の内容が明らかになった段階で，SEA のみなさんにお知らせしますが，もし韓国と何らかのビジネス情報や関連の有力者をご存知の方がおられたら，中園さん（今回のワークショップの実行委員長），君島さん（プログラム委員長）に連絡をくださるようお願いします．

4．月例会の案内について：

今まで月例会を，FAX メールだけにて案内してきましたが，この度，世話役の中園さんの作業を分担する意味からも，さまざまな媒体にてお知らせすることにしました．案内状は中園さんが作って，その原文をもとにしてはがき（山口さん），Junet（山口さん），NIFTY（中山さん），FAX メール（中園さん）で発送します．また年に２回ほど会員名簿の整理を行い，どんな媒体で案内を欲しいか，メンバーに選択していただきます．

SEA 会員の方で月例会の個別案内を望まれる方は，それぞれ希望される媒体を選んで，下記の担当窓口宛に連絡先を知らせていただければ，５月以降は自動的に案内をお届けいたします．

1）はがき：山口（TEL: 03-239-5473）
2）Junet: 山口（yamaguch@sra.co.jp）
3）NIFTY: 中山（SGC00372）
4）FAX メール：中園（TEL: 03-501-4161）

SEA Forum May '90

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 —

主催：ソフトウェア技術者協会

ひところのAIブームが去って，90年代の前半はどうやらファジィやニューロなどの新しい計算モデルとそれを具体化したマシンが，話題の一角を占有しそうです．

ファジィは，「あいまい理論」という何となく訳の解らないユーモラスなネイミングが受けているようです．また，最近のテレビCMでも証明されているように，理論よりも実践の面で実績があるというのも凄いところです．今回キーノートをお願いした向殿先生は，ファジィが始まった当初からの「あいまいさ」にまつわる数々のエピソードをお持ちのようです．今回は，そうした楽しい話題を含めて，ファジィ理論の起源から今日の流行に至るまでの経緯をお話しいただき，あわせてファジィ・コンピュータの今後についての予想をうかがいたいと思います．

それに続くパネルでは，ファジィ，ニューロ，並列マルチプロセッサといった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，会場のみなさんからの御意見も交えて，夢多い討論を展開したいと考えています．ありふれた日常生活にうんざりしているあなた，何か新しい知的な刺激を求めているあなた，ぜひこの機会に自分をリフレッシュしてください．

なお，当日の夜には，引き続き同じ会場でSEAの新年度総会も開かれますので，なるべく多数の方々の御参加をお待ちします．

開 催 要 領

（1）日　時：　1990年 5月 15日（火）　13:30 ～ 17:00

（2）会　場：　機械振興会館（東京・港区）　地下3階 研修2号室

（3）プログラム

基調講演：ファジィ理論とその応用　向殿政男（明治大学）（13:30 ～ 14:30）

パネル：ファジィ，ニューロ，etc.（15:00 ～ 17:00）

コーディネータ：　藤野晃延（FXIS）

スピーカ（予定）：　宇佐美仁英（富士通）　中野秀男（大阪大学）

向殿政男（明治大学）　他

（4）参加費：　3000円（SEA会員）　6000円（一般）　1000円（学生）

（5）申し込みおよび参加方法：

下の申込書に必要事項をご記入の上，郵便，FAXまたはE-Mailでお送り下さい．折り返し受講ハガキをお送りします．受講料は，**当日現金で受付にてお支払下さい**．ただし，定員（70名）になり次第，締め切らせていただきますので，あしからずご了承下さい．なお，申込後のキャンセルはお断わりします．

申込先：　〒　102　東京都千代田区隼町 2 - 12 藤和半蔵門コープビル 505

ソフトウェア技術者協会・フォーラム係

TEL: 03 - 234 - 9455　　FAX: 03 - 234 - 9454

E-Mail: sea@jus.or.jp (JUNET)

.....

SEA Forum May '90 **参加申込書**（申込日：＿＿月＿＿日）

氏名（ふりがな）：＿＿＿＿＿＿＿＿＿＿＿＿＿＿（　　）

種別（いずれかにチェック）：☐ *SEA* 会員（*NO.* ＿＿＿＿＿＿）　☐ 一般　☐ 学生

参加費：＿＿＿＿＿円

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

部門：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿　役職：＿＿＿＿＿＿＿＿＿＿＿＿

住所：（〒　　）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

TEL：＿＿＿＿ - ＿＿＿＿ - ＿＿＿＿＿＿（内線 ＿＿＿＿＿）

FAX：＿＿＿＿ - ＿＿＿＿ - ＿＿＿＿＿＿

SIGMAN スペシャル（Part-2）

ソフトウェア標準化の国際動向とその対策

参　加　者　募　集

主催：SEA 名古屋支部 & SIGMAN

ソフトウェアに関する標準化は，国際標準化機構（ISO）で規格化が進んでいますが，その動向は，ソフトウェア生産の多国籍化にともなって，われわれ技術者も決して無関心ではいられないような状況になりつつあります．

このような状勢を踏まえ，SIGMAN では，ISO の規格化に活躍しておられる加藤さん（凸版印刷），松原さん（日立ソフトエンジニアリグ）をお招きし，表記のテーマについて徹底的に議論する場を提供したいと考えました．

また，今回は，日常の被害者意識（?!）を心中から一掃し，気分をリフレッシュすることをねらって，東京を離れ明治村のふもと犬山でミーティングを持つすることになりました．

みなさんの積極的なご参加をお待ちしております．

開　催　要　領

期　日： 1990 年 5 月 25 日（金）～ 26 日（土）

会　場： 愛知県犬山市　レイクサイド入鹿（TEL: 0568 - 67 - 3811）

定　員： 20 名．

費　用： 10,000 円（SEA 会員）　13,000円（一般）　現地集合・現地解散とし，期間中の宿泊費および資料代を含みます．

支払は当日会場でお願いいたします．

実行委員： 岩田　康（SRA）　斎藤末広（テスク）　平田淳史（SRA）

プログラム委員： 天池　学（カシオ）　大久保功（ＹＨＰ）　加藤重郎（三谷コンピュータ）
坂本啓司（オムロン）　野々下幸治（富士ファコム）

プログラム（予定）：

5 月 25 日（金）

11:30	受付
12:00	自己紹介＆昼食
13:00	第 1 回　スペシャルセッションの報告　野々下幸治（富士ファコム）
13:30	講演　品質保証関連標準　　加藤重信（凸版印刷）
15:30	討論会　　司会：大久保，坂本
19:00	発表，Ｑ＆Ａ　　司会：天池

5 月 26 日（土）

08:30	講演　ソフト全般関連　松原友夫（日立ソフトエンジニアリング）
10:30	討論会　　司会：大久保，坂本
13:30	発表，Ｑ＆Ａ　　司会：天池
15:00	解散

申込方法： 今回も，比較的小人数で，建前でなく本音で討論をしたいと考えています．したがって，オフレコになります．参加を希望される方は，「国際標準」に対して御自身が持っている考えや問題点の指摘をポジションペーパの形にまとめて，下記の宛先に，E-Mail または FAX または郵便で送付して下さい．折り返し，詳細な参加案内をお送りします．

ポジションペーパ送付先：

野々下　幸治
〒151　東京都渋谷区代々木 4 丁目 30 番 3 号（新宿コヤマビル）
富士ファコム制御
TEL: 03 - 374 - 1711 (ext 3712)
FAX: 03 - 374 - 3113
Junet: nonosita@ffcnt.ffc.fujitsu.co.jp
Nifty: PFC01561

ソフトウェア・シンポジウム '90

および

併設チュートリアル

開催予告および参加者募集

1990 年 6 月 6 - 8 日　◇　於：京都リサーチパーク（KRP）

主　催：　ソフトウェア技術者協会 (SEA)
日本ソフトウェア科学会 (JSSST)
情報サービス産業協会 (JISA)
Rocky Mountain Institute of Software Engineering (rMISE)

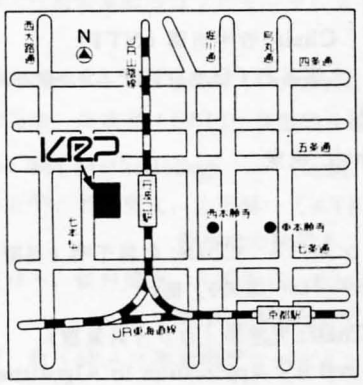
第 10 回目を迎える今年のソフトウェア・シンポジウム '90 は，会場をこれまでの東京から初めて関西エリア（京都）に移して，装いを新たに開催されることになりました．

主催団体も，これまでのソフトウェア技術者協会（SEA）および情報サービス産業協会（JISA）の他に，日本ソフトウェア科学会（JSSST）および Rocky Mountain Institute of Software Engineering（rMISE）の２団体が加わり，産学および国際間の交流という香りがちょっぴり強くなりました．

とはいえ，ソフトウェア技術の実践的な側面に関する発表や討論を行い，参加者が相互に情報や知識を交換する場としてのシンポジウムの基本的性格は，まったく変わりありません．

なお，例年通り，併設チュートリアル（前日に半日×６セッション）およびツール展示（商品ではなく研究開発途上のプロタイプのデモおよびプレゼンテーション）も行われます．多くの方々の参加をお待ちします．

京都リサーチパーク（KRP）
京都市下京区中堂寺南町 17　TEL 075 - 322 - 5348



■交通のご案内

JR丹波口駅より
丹波口駅から西へ徒歩５分

近鉄・JR京都駅より
● 市バス乗り場
C6 …… 73「洛西バスターミナル」行き
…… 75「双ヶ丘」行き
● 京都バス乗り場 1 …… 81「大覚寺」行き
…… 84「御室仁和寺」行き

阪急・四条大宮駅より
● 市バス 32「梅津車庫」行き
66「五条車庫」行き

京阪・五条駅より
五条駅から河原町五条バス停まで
徒歩３分
● 市バス 43「久世橋東詰」行き
80「梅津車庫」行き
※上記のバス（所要時間約15分）で、五条千本下車、七本松通りを南に約50ｍ東側。

シンポジウム・スタッフ

実行委員長	盛田　政敏（ケーシーエス）	熊谷　　章（PFU & 富士通）	松原　友夫（日立 SK）
プログラム委員長	鳥居　宏次（大阪大学）	亀井　　朗（ASTEM）	宮本　　勲（Hawaii 大学）
	臼井　義美（日本電子計算）	中野　秀男（大阪大学）	山田　　淳（東芝）
プログラム委員	阿草　清滋（名古屋大学）	竹中　市郎（NTT）	山田　　茂（広島大学）
ツール展示委員長	中野　秀男（大阪大学）	竹中　豊文（ATR）	M. Dowson（sda）
	天池　　学（カシオ計算機）	田中　克己（神戸大学）	B. Nejmeh（INSTEP）
	大西　　淳（京都大学）	玉井　哲雄（筑波大学）	W. Riddle（rMISE）
	大場　　充（日本 IBM）	林　　香（SRA）	J. Sayler（Michigan 大学）
	落水浩一郎（静岡大学）	原田　賢一（慶応大学）	L. Williams（SER）
	片山　卓也（東京工業大学）	藤野　晃延（FXIS）	R. Zhang（復旦大学）
	菊野　　亨（大阪大学）	二木　厚吉（電総研）	

ソフトウェア・シンポジウム'90 プログラム

6月 7日 (木)

09:30 − 10:00　**受付**

10:00 − 12:00　**Opening Session** − Chair: 臼井義美 (JIP)

基調講演: Prof. Gerhard Fischer (Colorado 大)

Cooperative Knowledge-Based Design Environments for the Design, Use and Maintenance of Software

12:00 − 13:00　＜昼食＞

13:00 − 14:30　**Session 1A: Requirement Analysis** − Chair: 飯沢恒 (三菱電機東部 CS)

河合和久, 小山雅庸, 大岩元 (豊橋技科大) 他: カード操作ツール KJ エディタの評価実験

織田健, 片山卓也 (東工大) 他: データ構造の段階的詳細化による仕様作成法

山村吉信 (日本 IBM): Mutable Specifications: An Experiment

Session 1B: Formal Approach − Chair: 二木厚吉 (電総研)

澤田寿実 (SRA), 二木厚吉 (電総研): An Inplementation of Order-Sorted Term Rewriting

今泉貴史, 篠田陽一, 片山卓也 (東工大): 属性文法によるコンフィグレーションコントロールシステムの記述と実現

奥井順 (名工大), 藤井護, 谷口健一 (大阪大): Two Algebraic Specifications of a Process Control Program and Their Relations

Session 1C: Software Process − Chair: 落水浩一郎 (静岡大)

伊野誠 (SRA): オブジェクト指向の開発プロセス・モデル化と支援環境

太田剛, 山口高平, 落水浩一郎 (静岡大): ソフトウェアプロセスの動的特性に対応するための一機構

荻原剛志, 井上克郎, 鳥居宏次 (大阪大): オブジェクト指向によるソフトウェアプロセスの記述モデル

Session 1D (Panel): これからの CASE − Chair: 竹中市郎 (NTT)

Panelist: 江沢智 (日商エレクトロニクス), 大篭高之 (日本システム), 桜井麻里 (SRA), 竹田尚彦 (豊橋技科大)

Session 1E: Tool Presentation #1 − Chair: 未定

14:30 − 15:00　＜休憩＞

15:00 − 16:30　**Session 2A (Panel): これからのメインテナンス** − Chair: 佐藤千明 (長野県協同電算)

Panelist: 宮本勲 (Hawaii 大), 盛田政敏 (ケーシーエス), 他.

Session 2B: Testing and Debugging − Chair: 天池学 (カシオ計算機)

濱利行 (日本 IBM): Prolog Type System and it's Application to Algorithmic Debugging

Nie Shixue, Zhang Ran (復旦大): New Debugging Tool Based on Dynamic Slicing Concept

大場充 (日本 IBM): Optimum Test Case Generation for Programs without Loops

Session 2C: Software Development Environment − Chair: 藤野晃延 (FXIS)

島健一 (ATR): 判断履歴を用いた設計文書の理解について

山田宏之, 手塚慶一 (愛媛大): Software Modification Support System Using Interaction

小野田崇 (電力中研): M-cube: マルチメディア・コンサルテーション・システム開発支援環境

Session 2D: Objest-oriented Approach − Chair: 河田亨 (シャープ)

青木淳 (FXIS): オブジェクト指向プログラミング環境におけるインテグレーション過程

塩谷和範 (SRA): オブジェクト指向プログラミングによる CASE ツールの実現

野口和彦, 伊藤光恭 (NTT): 通信制御ソフトウェアにおけるオブジェクト指向設計法

Session 2E: Tool Presentation #2 − Chair: 未定

16:30 − 17:30　＜休憩＞

17:30 − 20:00　＜情報交換パーティ＞

ソフトウェア・シンポジウム'90 プログラム

6月 8日（金）

09:30 − 11:00 **Session 3A: Database** − Chair: 田中克巳（神戸大）

C.F.Liu, Z.L.Gong, M.Yu, W.J.He（捷成信息工程公司）: EASYDB - A DBMS Prototype for Managing Hierarchically Organized Data

加藤康人（PFU）: ソフトウェア要求定義・分析におけるオブジェクトマネージメントシステム

酒匂寛（SRA）, L.G.Williams（Software Engineering Research）: ソフトウェア DB のためのデータモデルの提案ならびに試作システムについて

Session 3B (Panel): 中国における CASE ツールの開発 − Chair: 熊谷章（PFU & 富士通）

Panelist: Cai Linxi(華東計算機研究所), Hao Kegang(西北大), Ju Dehua(華東化工学院), Zhu Sanyuan (上海ソフトウェア・センタ)

Session 3C: Software Quality Techniques − Chair: 菊野亨（大阪大）

大寺浩志，山田茂（岡山理大）: テスト空間に基づくソフトウェアの信頼性評価に関する考察

毛利幸雄（日本ユニシス），菊野亨，鳥居宏次（大阪大）: フォールト修正文に基づくフォールト混入工程の分析

楠本真二，松本健一，菊野亨，鳥居宏次（大阪大）: プログラム開発におけるチーム性能の実験的評価

Session 3D: Knowledge Base − Chair: 玉井哲雄（筑波大）

平井一人（JIP）: 知識処理における部分情報問題とその処理方法 − 自然言語処理を例として

桝本人司，知念徹，石元繁一，大熊清司（三菱電機コントロールソフトウェア）: エキスパートシステム開発における知識獲得プロセスに関する一考察

Session 3E: Tool Presentation #3 − Chair: 未定

11:00 − 11:30 **Session 4A: Reuse** − Chair: 阿草清慈（名古屋大）

高橋章（三菱電機関西コンピュータシステム）: ひな形プログラム管理システムによるソフトウェアの再利用

中小路久美代，G.Fischer（Colorado 大）: Reusability in Domain-oriented Design Environments

石塚正令，鐘友良（FXIS）: 代数的仕様記述における再利用について

Session 4B: Methodology − Chair: 亀井朗（ASTEM）

橋本正明，竹中豊文，山下紘一（ATR）: モデルの組合せと設計情報の寿命に着目したソフトウェア自動作成の枠組み

宮尾淳一，浜村博康，八田浩一，若林真一，吉田典可（広島大）: リアルタイムソフトウェア設計システム VD/RCS −高信頼性と高速化

新田　稔（SRA）: 関数型プログラミング言語の Software in the Large への適用

Session 4C: (Panel): これからのソフトウェア開発 − Chair: 大場充（日本 IBM）

Panelist: 四野見秀明（日本 IBM），田中和弘（オムロン），戸沢昭（日立 SK），西山聡（三菱総研）

Session 4D: Hypertext − Chair: 野村行憲（岩手電子計算センター）

塩見彰睦，竹田尚彦，河合和久，大岩元（豊橋技科大）: ソフトウェア基本設計のための図式エディタシステム

馬場始三，中野秀男（大阪大）: ハイパーテキスト／メディアシステムにおける自動配置について

熊谷章（PFU & 富士通）: 研究開発のためのコンピュータによる支援環境

Session 4E: Tool Presentation #4 − Chair: 未定

13:00 − 14:00 ＜昼食＞

14:00 − 16:00 **Closing Panel: 情報化時代における世界の中の日本** − Chair: 鳥居宏次（大阪大）

基調講演： 脇田寿（大阪大）

Panelist: G.Fischer（Colorado 大），片山卓也（東工大），岸田孝一（SRA），深瀬弘恭（アスキー），松原友夫（日立 SK），宮本勲（Hawaii 大）

ソフトウェア・シンポジウム'90

および

併設チュートリアル

参加申込み要領

下の申込書に必要事項をご記入の上，郵便または FAX で，次の宛先までお送り下さい．折り返し受付票および請求書をお送りいたします．

〒 102 東京都千代田区隼町 2 - 12 - 505　ソフトウェア技術者協会　シンポジウム係

TEL:　03 - 234 - 9455　　FAX:　03 - 234 - 9454

	会　員	一　般	学校関係者	学　生
シンポジウム（２日間，パーティ参加費を含む）	40,000 円	50,000 円	20,000 円	4,000 円
チュートリアル（シンポジウム参加者）*	10,000 円	15,000 円	5,000 円	2,000 円
チュートリアル（シンポジウム不参加者）*	15,000 円	20,000 円	10,000 円	4,000 円

***ただしチュートリアルの料金は１セッション当たりの料金です．**

併設チュートリアル（シンポジウムの前日６月６日（水）に同じ会場で開催）

セッション	時間帯	テ　ー　マ	講　　師
A1	午前	最新オブジェクト指向データベース	田中克己（神戸大学）
A2	午前	プログラム開発におけるフォーマルアプローチ	谷口健一（大阪大学）
A3	午前	ソフトウェア設計法最前線	松本吉弘（京都大学）
P1	午後	CASE ツールの導入と活用	佐原　伸（SRA）
P2	午後	システム開発を成功させる戦略的アプローチ	岡田正志（NES）
P3	午後	90 年代のソフトウェア・メインテナンス支援環境	宮本勲（ハワイ大学）

***ただし，午前は 9:30 ～ 10:30，午後は 13:30 ～ 16:30.**

.....

ソフトウェア・シンポジウム'90 参加申込書（申込日：＿＿＿月＿＿＿日）

氏名（ふりがな）：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿（　　）

種別（いずれかにチェック）：☐ 会員（☐ SEA　☐ JISA　☐ JSSST）　☐ 一般　☐ 学校関係者　☐ 学生

申込：　☐ ソフトウェアシンポジウム

☐ 併設チュートリアル（A1　A2　A3　P1　P2　P3）　希望セッションを○で囲む．

参加費：＿＿＿＿＿＿＿＿＿＿＿＿円

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

部門：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

役職：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

住所：（〒　　）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

TEL:　＿＿＿＿ - ＿＿＿＿ - ＿＿＿＿＿＿（内線 ＿＿＿＿＿）

FAX:　＿＿＿＿ - ＿＿＿＿ - ＿＿＿＿＿＿

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

Well 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

sea
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