SEAMAIL

Newsletter from Software Engineers Association

Volume 8, Number 10 February, 1994

目　次

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

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

その主な活動は，機関誌SEAMAILの発行，支部および研究分科会の運営，セミナー／ワークショップ／シンポジウムなどのイベントの開催，および内外の関係諸団体との交流です．発足当初約200人にすぎなかった会員数もその後飛躍的に増加し，現在，北は北海道から南は沖縄まで，900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50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，東北の各地区で設立されており，その他の地域でも設立準備をしています．分科会は，東京，関西，名古屋で，それぞれいくつかが活動しており，その他の支部でも，月例会やフォーラムが定期的に開催されています．

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

ＳＥＡＭＡＩＬ　Ｖｏｌ．８，Ｎｏ．１０　　１９９４年２月２８日発行
編集人　岸田孝一
発行人　ソフトウェア技術者協会（ＳＥＡ）
〒１６０　東京都新宿区四谷３－１２　丸正ビル５Ｆ
TEL: 03-3356-1077　FAX: 03-3356-1072
印刷所　サンビルド印刷株式会社　　〒１６２　東京都新宿区築地町８番地
定価　５００円（禁無断転載）

編集部から

☆

政治改革や減税をめぐるゴタゴタも，時ならぬ大雪も，すべてSEAMAILが毎月発行されているせいだという噂もありますが，ともかくVol.8, No.10をおとどけします．

☆☆

今月は，中野先生から，去年のTICSの旅行記が出てきたので．編集部でそれにあわせて基調講演の資料を整理し，TICS報告特集を組んでみました．

☆☆☆

TICSまわりでは，参加者の１人である伊藤昌夫さんが，いま，あのときのポジションをリライト中で，そのうち力作論文が出てくるものと期待しています．また，前･PFU上海総経理の熊谷さんが，BICS'91からUICS'92を経てTICS'93にいたるこれまでのCASEシンポジウムin中国の歴史をまとめてくださっているはずです．いつごろ完成するかはわかりませんが....?

☆☆☆☆

この号には，また，昨年鳥取大学へ移られた山田先生からは，前々号の業界リストラ論義に対応するメッセージを，コロラドの中小路さんからはAmerican Programmer誌に載せたエッセイの日本語版を寄稿していただきました．American Programmerといえば，これまで同誌に執筆した日本人のほとんどはSEAのメンバ(青木，大場,，玉井，中小路，中野，etc)で，SEAはあたかもアメリカのソフトウェア界に向けてのVoice of Japanであるかのような感じがします．

☆☆☆☆☆

と，ちょうど編集が終った所へ，大場さんが事務局へ．昨年の秋に京阪奈で行なわれた国際ワークショップのレポートをわざわざとどけてくださったのですが，これは，すでにご案内した来月の国内ワークショップ向けに用意された「品質データ収集･分析に関するアンケート案」と一緒に，来月号にまわすことにします．乞う，御期待!

☆☆☆☆☆☆

TICS'93 総括報告

岸田 孝一
(SRA)

北京 (BICS'91), ウルムチ (TICS'92) に続く第3回目の国際CASEシンポジウムin中国は, 山東省·泰安の山東鉱業大学および御座賓館において開催された. テーマは,「21世紀におけるソフトウェア開発パラダイム」. プログラムは次の通りであった.

10月13日(水)
午後 基調講演 (1)
·楊芙清 (北京大学)
Jade Bird System
· D. Bjorner(国連大学)
PRaCoSy プロジェクト
· L. Williams(SER)
オブジェクト指向を越えて
夜 レセプション
10月14日(木)
午前 基調講演 (2)
·中小路久美代 (SRA)
将来のCASE環境の機能的要件
· L. Belady(MERL)
来るべき時代のソフトウェア技術
グループ討論
(A) コンセプト·クリエータの視点
(B) システム·アーキテクトの視点
(C) システム·インプリメンタの視点
(D) ユーザおよびマネージャの視点
午後 グループ討論 (つづき)
10月15日(金)
終日 曲阜観光
10月16日(土)
午前 特別講演
· D. Bjorner(国連大学)
システム要求分析と形式的技法
クロージング·パネル

参加者は, 37名 (内訳は, 中国20, 日本13, 欧米3). 経済状勢のせいもあって, 過去2回よりは小規模な会議となった.

第1～2日の5つの基調講演については, それぞれ, OHPの内容要約またはポジション·ペーパを以下に収録した.

グループ討論は, 各グループとも, 日中各2～3人のスピーカのプレゼンテーションのあと, それぞれのテーマについての討論を行った.

私自身は, Bグループに参加した. 中国側の発表は, 技術的にナイーブすぎる感じであった. 伊藤さん (MASC) の発表は, やや抽象的ではあったが, 内容がControversialで興味深かった. 峰尾さん (ユニシス) の発表は, 形式仕様RAISE応用例題のわかりやすい説明で, Bjorner教授も感心するできばえであった.

第3日の曲阜観光は, 戸外での自由討論を通じて参加者相互の親睦を深めるとともに, われわれ自身の未来を考えるにあたって, 古代の大思想家の足跡に思いをめぐらそうという意図で計画された. 幸い好天に恵まれ, 楽しい一日であった.

最終日は, まず, Bjorner教授から, 形式的仕様化技法の現実問題への応用アプローチについて, 熱のこもった特別講演があり, 引き続いて, 私が司会役となって, この3日間での討論を総括するパネルを行った.

翌日の日曜日, 全員で聖地·泰山に登り, 今後のソフトウェア技術の発展を祈った.

Software Technology for the Coming Indterdisciplinary World

Les A. Belady
(Mitsubishi Electric Research Laboratory)

以下は，Belady 氏の基調講演の OHP を編集部で要約したものである．

1. ある予言

故 Alan Perlis 教授 (Yale 大学) が，かつて，次のように述べた:

> — われわれは，いま，コンピュータ時代の始まりに位置しているのだということを，決して忘れてはならない．近い将来，時代はまさに爆発的に変化するであろう．コンピュータの物理的形態も，いろいろ変わるかもしれない．しかし，「計算」の抽象的概念についてのわれわれの認識は，時の流れとともに深まり，また，われわれはそれによりいっそう依存するようになるだろう．もはや，コンピュータのない人間生活は考えられない．だからといって，もちろん，機械を偶像的に崇拝するなどということがあってはならない．われわれは，機械を飼いならし，よりよく働かせるようにしなければならない．未来社会では，コンピュータは，人間活動のあらゆる局面において，何らかの役割をになうことになるであろう．

まさしく現在，ビジネスや文化や，その他もろもろの要素が複雑にからみあい，情報技術に対する期待もそれだけ高まりつつある．コンピュータをより速く安く小さくするだけでは不十分である．利用技術の上での革新が求められているのだ．そうした新しいアプリケーションには，ふつう，多くの相異なる技術が組み合わせて利用されなければならない．そして，それらの技術も，また，アプリケーションの環境も，目まぐるしく変化する．かくして，再び，ソフトウェア技術者たちの肩に，重い負担が課せられるのである．

このプレゼンテーションでは，数年前にアメリカで行われた 2 つの産業政策ワークショップでの討論内容を紹介し，それを踏まえて，情報技術の研究開発に関する私のビジョンを述べる．

2. アメリカの国際競争力

1989 年 12 月と 1991 年 1 月に，National Research Council の主催で，「アメリカのコンピュータ産業の国際競争力を強化するには?」をテーマに，2 回のワークショップが開かれた．

2.1. 1989 年のワークショップ

政策アジェンダの定義を目標に，(1) ハードウェア，(2) ソフトウェア，(3) システム，(4) ビジネスおよび経済，の 4 つのパネルが行われた．結論として，アメリカのハードウェア産業は弱い，ソフトウェアは強い，しかし，将来もっとも望みがあるのはシステム·インテグレーション (SI) だということになった．

SI のノウハウを支援するためには，次のようなことが必要であると認識された:

- 最適な設計を容易にするための標準化．完全に標準化された環境においても，SI はやはり複雑な仕事である!)
- ソフトウェア·エンジニアリングの教育と再訓練．これからの教育においては，実プロジェクトでの経験，ハードウェアおよびコミュニケーション技術への習熟，そして，すくなくとも 1 つの特定アプリケーション·ドメインの知識が必要である．
- コンピュータによって支援されたシステム設計プロセス．当然ながら，これは CSCW を含んでいる．

当面のコンピュータ·アプリケーションの動向は，各種の装置とアプリケーションとを統合する傾向に向かいつつある．これは，旧世代のコンピュータが，時間的にも空間的にも孤立したかたちで使われていたのとは，好対照である．その結果，ヘテロジニアスな分散ネットワーク·システムが主流になる．そうしたハード/ソフトのコンポーネントを一緒にまとめる「接着剤としてのソフトウェア (Software Glue)」の役割が

重要になる．ソフトウェア産業は，独立した産業カテゴリではなく，あらゆるプロダクトあらゆる人間活動の一部分になる．の役割が重要になる．ソフトウェア産業は，独立した産業カテゴリではなく，あらゆるプロダクトあらゆる人間活動の一部分だと考えたほうがよい．

統合的なネットワーク･アプリケーションは，当然，個々のユーザの要求に合わせてカストマイズされる必要がある．したがって，ソフトウェアへのニーズは，次の2つに分極化する:

A型－純粋なソフトウェア部品の「製造」
B型－それを材料とするシステム設計

カストマイズはバラエティの増加，複雑さの増大を意味する．したがって，将来のソフトウェア･エンジニアは，アプリケーション･システム･デザイナとしての役割を担うことが要求される．

A型のソフトウェアはどこでも製造でき，完成したものは，既成部品として，だれでも利用できる．しかし，B型のソフトウェアの開発には，アプリケーションに対する深い理解と，カストマとの共同作業を必要とし，したがって現地でなければ行えない．SIの成功には，対象の複雑さをマスターしなければならず，それには多額の投資を必要とするが，結果としては大きな収益が期待できる．

将来におけるアメリカのコンピュータ産業の強みは，この分野において求められるべきであろうというのが，ワークショップの結論であった．

2.2. 1991年のワークショップ

システム･インテグレーションをテーマに，産･官･学の代表者が集まって，(1) アプリケーション，(2) コミュニケーションとネッテワーク，および (3) 支援技術，に関して，3つのパネルが行われた．

SIの需要を喚起する新しい経済的ニーズとしては，次のような諸要因が考えられる:

- 経済成長において，労働力プラス情報 (ノウハウ) の果たす役割が増大しつつある．
- 分散化と組み合わされた規模の経済学．
- 60～70年代に個別に自動化された「情報の島々」のあいだのコミュニケーションをコンピュータで支援する必要性．
- ネットワーク分野での新ビジネスの機会 (たとえば，制御ソフトウェアにおけるノベルやロータスの躍進)

いくつかの興味深い数字がある:

- 現在のSIビジネスの規模は，アメリカだけで約170億ドル．年間成長率13%．EDSやアーサ･アンダセンなどの大会社をはじめとして，すでに1600社が従事している．
- すでに，7500万台のPCがアメリカ全土に普及しているが，ホワイトカラーの生産性はあまり向上していない．
- 現在，アメリカのコンピュータ･ネットワークの半分は，個々の企業の内部に存在している．
- 2015年までに，日本の全人口は光ファイバで結合されるだろう．
- いま連邦政府が計画している高性能コンピューティング計画 (HPCP) の予算の14%は，マルチ･ギガビットの全米研究教育ネットワーク (NREN) の開発にあてられる予定．
- 2000年までに，アメリカのディジタル技術関連産業は，全産業の1/3を占めることになるだろう．

経済現象として，すでに，次のような力点のシフトが起こりつつある:

パフォーマンス: 裸のCPU＋メモリから，広帯域ネットワークへ．

労働集約性: 研究および概念形成から，具体化 (複雑なシステムのほんとうのエンジニアリング) や，オペレーション (人間の介入は不可避) へ．

生産性: 個人作業の効率化から，複数の人間の協調へ．

システム･アーキテクチャ: 単純な自動化から，人間･機械間のコミュニケーションへ，(特に人間サイドの) プロセスの再設計へ．静的なシステムから，進化するシステムへ．スタンドアロンから，システム間の連携へ．費用対効果から，安全性･強靭性･予測可能性へ．

市場競争力: 省力化･原価削減から，環境条件の急激な変化に対応するために，ビジネス経験 (ノウハウ) の収集･加工･普及･実践へ．

データ: 処理･搬送･表示から，分析･学習へ．

政府: 統制からリーダシップへ．産業刺激を意図した規制/標準化．先端的実験分野の開拓．教育·訓練の促進．

3. 新たな研究開発の機会

3.1. 協調型問題解決

システムの設計やエンジニアリングだけが多数のエキスパートを必要としているわけではない．すべての人間活動が複合領域 (interdisciplinary) なかたちになってきているのである．

今日われわれが直面している諸問題は規模がきわめて大きく，多面的であるため，複数の視点からの分析が必要とされる．また，技術·資本·市場が地球規模で分散しつつあるので，多言語·多文化的なチーム編成が，必然的に要求される．

しかしながら，そうしたチームが共同作業をするさいに，コミュニケーションの難問が生じる．これは，情報技術にとっては，絶好のアプリケーション分野である．すなわち，多種·多彩なエージェント (人間および機械) たちの間での，適切なパートナシップの確立をどうするか，ということである．

そこには，もちろん，いくつかの制約条件が存在する．

まず，過去の遺産として残っている社会的インフラをどう底上げするか，である．ネットワークに関していえば，CATV や電話回線をそのまま使うか，それとも新しい線を張るか，という問題がある．また，既存のソフトウェア遺産(負債?) およびデータベースをどうするかも，大きな問題である．

政府関連では，お互いに調整の取れていない各種の規制や標準を整備しなおす必要がある．

ソフトウェアの生産性や品質もいまのままでよいわけではない．また，要員訓練についても，対象システムの性質が個々に異なっているがゆえの困難がある．そして，一般にいって，人間は，革新や変化に対して抵抗を示しがちである．

パフォーマンス面では，プロトコルの複雑さを何とかしなければならない．そして，ギガビットの情報洪水をどうやってバッファリングするかも，大きな問題であろう．

3.2. 人的資産の構築と更新

教育は，開発途上国のみならず，先進工業国においても，緊急の課題である．そのリクワイヤメントは，第 1 に複合領域的であること，すなわち孤立した専門知識ではなく，システム化された知識の習得が必要とされる．そして，第 2 にフレキシブルであること，つまり知識の学び方を学ぶ必要がある．

現有の要員の再訓練問題も大きな課題である．ノウハウの絶えざる変化によって，すべてのワークフォースをつねに訓練しなおすことが必要とされる．教育，再訓練，そして実作業は，3 つのフェイズ分けされた活動ではなく，お互いに入れ子関係になっている．すなわち，実際の問題解決を通じての学習が要求されるのである．

このことは，コンピュータによる教育·訓練の支援だけをとりあげてみても，今後きわめて大きなビジネス·チャンスが開かれていることを意味している．

4. まとめ

1990 年代はディジタル·コミュニケーションの時代である．

ネットワークは，実験や革新のための新しいプラットフォームを提供する．コミュニケーションは通勤や移動のロスを排除することで，仕事の環境改善に寄与する．

ISDN およびディジタル·スイッチングを介した広帯域光ファイバへの移行や，インターネットのコマーシャル化もすでに進行中である．しかし，コンピュータに比較して，コミュニケーションはより複雑であり，より複合領域的である．ネットワーク上に，データや処理パワーをどううまく配置するかが，新しい課題になる．

ネットワークをベースとする新しい時代のアプリケーションのために，システムの「接着剤」としての B 型のソフトウェアを開発するには，コンピュータ (ソフトウェア) に関する専門知識だけでは不十分である．ほとんどすべての問題は，単なるソフトウェア技術の範囲を越えて，複合領域的な，多文化のチームによって解決されるであろう．

ネットワーク化されたコンピュータは，作業·学習·訓練の新しいやり方に対する要求条件を表しており，またビジネス·チャンスを暗示している．そうした機会を生かすために必要な技術の移転は，もしかすると，新技術の創造よりもむずかしいかも知れない．

PRaCoSy Project

An Example of UNU/IIST Training and Joint R&D

Dines Bjorner
(United Nations University)

以下は，Bjorner教授の基調講演のOHPを編集部で要約したものである．

1. UNU/IIST とは?

UNU(United Nations University 国連大学) は，本部を東京·青山に置く国連組織であり，世界各地に設けられたテーマ別の研究·訓練センタの連合体である．

IIST (International Institute of Software Technology: 国際ソフトウェア技術研究所) は，ポルトガル，中国，およびマカオ政府の基金により，1992年7月に，マカオに開設された．その主目的は，ソフトウェアに関する技術移転，すなわち開発途上国におけるソフトウェア技術ニーズに応えて，必要な基礎的·先進的教育訓練ならびに研究開発設備を提供することである．

現在の年間予算規模は1200万USドル，スタッフは，専任研究者5人，訓練生10人，事務員6人である．3年後の1996年には，予算·組織とも3倍程度に拡大することを予定している．

2. 活動

UNU/IISTは，次のようなカテゴリの活動を行っている:

(1) 先進的共同研究開発

現在進行中のものは:

MacShubert (オーストリアの支援による教育技術開発)
MaGICS (マカオ政府のための汎用情報処理システム)
PRaCoSy (中国の鉄道制御システム)

その他に以下のプロジェクトが計画中である:

DiMaCS (Dosaster relief and recovery Management Computer System)
GaDIMS (Geo- and Demographic Information Management System forEnvironment and Sustainable Development)
HEMIS (Higher Education Management Information System)
MoTras (Mongol/Manchu Traditional Script computerization)
SaM2I3 (Small and Medium-size Manufacturing industry Information Infra-Structure)
WHeC2S (WHO Health Care Computing System)

(2) セミナーおよび教育ワークショップ

これまでに次のセミナーを実施した:

(1) Peking, China, Oct 92
(2) Pune, India, Nov 92
(3) AIT, Bangkok, Apr 93
(4) Hanoi, Vietnam, May 93

また，現在計画中のものは:

(5) Pyongyang, DPRK, Oct 1993
(6) Macau, Jan 94

[編集部注] 最後のものは，すでにSEAMAILでも広報したが，フォーマルなプログラミング方法論に関する2週間の集中セミナーであり，日本からは荒木啓二郎 (奈良先端大)，佐原伸 (SRA) の2人が参加した．

(3) 共同研究

途上国の大学·研究機関との共同研究である．現在進行中/計画中のものは:

(1) DeTfoRS (Design Techniques for Real-time Systems)
(2) ProCoS (Provably Correct Systems)

(4) イベント

UNESCOの支援により，この11月に北京で，途上

国の大学におけるソフトウェア教育カリキュラムをテーマにしたワークショップ UCiST (University Curricula in Software Technology) を計画中である．このワークショップは，開催場所を中央アジア，南アメリカに移して，3 回連続で開催の予定．

[編集部注] 北京のワークショップには，日本から，二木厚吉 (北陸先端大)，土居範久 (慶応大)，岸田孝一 (SRA) の 3 人が参加した．

(5) 情報およびソフトウェアの配布

全世界の学術雑誌，研究レポートをライヴラリに収集し，要求に応じて，途上国の研究者に配布する．

また，UNU/IIST はすでにインタネットに接続しているので，全世界のフリー･ソフトウェアを，ネットワークを介して収集し，MT その他適当な媒体で途上国の研究者に配布する．

3. PRaCoSy プロジェクト

PRaCoSy は PRC Railway Computing system の略称である．

中国は世界最大の鉄道システムを保有しているが，その制御は決して進んでいるとはいえない．中国鉄道省は，現行設備の更新および新鉄道の建設に，ドル換算で 18 億の投資を行いつつあり，また，今後 3 年間に世界銀行から4.2 億ドルの借款を予定している．近い将来において，中国鉄道省は，世界最大の鉄道関連ソフトウェアのユーザになるであろう．現在は，主要なソフトウェアはすべて海外から購入している．

このプロジェクトの目的は，中国鉄道省コンピュータ･センタ (北京･上海･成都) が，自力で必要なソフトウェアを開発し，先進的な新しい統合ソフトウェアの研究開発を行い，ゆくゆくはそうした鉄道制御関連ソフトウェアを他国に輸出できるようになることである．

プロジェクトは，1992 年 10 月にスタートし，現在 5 人のエンジニアが，UNU/IIST において，先進的なソフトウェア仕様化･設計技術の訓練を受け，成都－武漢間 600KM の鉄道における列車制御システムの開発を目的として，作業を行っている．

すでに，これまで 1 年間をかけて，「鉄道とは何か?」についての，インフォーマルな記述および，Z や VDM などの仕様記述言語を用いたフォーマル･モデルの構築を，ほぼ完了した．今後の予定としては，1994 年から 95 年までの 27ヶ月で，(1) 試作開発，(2) デモンストレーション，および (3) 技術移転を行なうことになっている．その期間に，さらに 20 人ほどの技術者の訓練が予定されている．

PRaCoSy は，当面，UNU/IIST の「旗艦」プロジェクトであり，私自身を含めて，すべてのスタッフが全力を傾注している．また，その成果は，世界中からかなりの注目を集めつつある．

[編集部注] Bjorner 教授は，来る 3 月 7 ～ 9 日に，東京･青山の国連大学ホールで開催予定の ACM 日本支部結成記念イベントに，基調講演者の 1 人として来日される．9 日に予定されているその講演では，これからのソフトウェア開発およびソフトウェア産業のあるべき姿を論じられるそうだが，予稿を見ると，PRaCoSy の技術的詳細についても，例題として，ある程度言及されるものと思われる．興味を持たれた方は，ぜひ参加･聴講をおすすめしたい．

Jade Bird System

楊　芙清
(北京大学)

以下は，楊教授の基調講演の OHP を編集部で要約したものである．

1. Jade Bird 1

これは，第7次5ヶ年計画の一環として行われた統合的ソフトウェア･エンジニアリング環境構築を目的とする国家プロジェクトであった．プロジェクト名 Jade Bird (青鳥) は高名な詩人の作品から採られている．

中国全土から，10ヶ所以上の大学･研究機関が開発に参加し，最後のツール･インテグレーションは，環境のデータベース開発を担当した北京大学において行われた．試作開発は 1991 年に終了し，BICS-91 に参加された方々には，デモをお目にかけた．その後，商品化のための開発作業が行われた．

システムの特徴は，オブジェクト指向の概念および技法を採用して，オープン･メカニズムによる環境の統合を目指しているということである．環境データベース BETA-87 の上に，各種の開発支援ツールを統合し，カストマイズ可能な標準ユーザ･インタフェイスを提供している．

ハードウェア･プラットフォームとしては，Sun (Unix), 386/486PC (Unix/Xenix), Portable PC (DOS) を用いている．

2. Jade Bird 2

引き続き，第8次5ヶ年計画の中心的プロジェクトの1つとして，Jade Bird 2 がスタートした．これには，やはり全国20ヶ所以上の大学･研究機関が参加している．

プロジェクトの目標は，中国ソフトウェア産業のための開発基盤を整備し，国家的な技術標準へのリファレンスを提供し，統一的でオープンなソフトウェア開発用プラットフォームを構築することである．

計画されている主要な活動項目は次の通り:

(1) 開発ガイダンスのための標準/仕様の提供

- OO 開発の標準仕様
- OOA ドキュメント標準と例題
- JB2 インタフェイスの設計仕様
- 伝統的ツールの設計仕様
- JB2 ツールの構造と開発基準
- JB2 の機能的仕様
- CASE C++ 言語のリファレンス

(2) 統合のメカニズム

- OMS (オブジェクト管理システム)
- CASE C++ 言語
- CASE インタフェイスとクラス･ライブラリ
- メッセージ･サーバ
- 統合制御言語

(3) 伝統的ツール

- 13 種のライフサイクル･ツール
- DB アプリケーション･ジェネレータ
- ネットワーク･サービス用のツール

(4) アプリケーション開発ツール

- GIS
- リアルタイム･モニタリング･システム

(5) OO ツール

(6) 先進的技法/ツール

- 統合メカニズムの研究
- 分散システム開発用ツール･セット
- 各種の自動化ツール
- 各種の知的ツール
- MMI 生成ツール

Beyond Object-Oriented

— Software Methods in the 21st Century —

Lloyd G. Williams
(Software Engineering Research)

1. Introduction

As the use of computers becomes more pervasive in business, industry, and every day life in the twenty-first century, issues of software quality and productivity will become even more significant than they are today. Software systems of the twenty- first century will also be considerably larger and more complex than those produced now. Meeting the demands of software development in the future will require software methods that are qualitatively different from those currently in use.

Object-oriented development has proven useful as a programming technology and several methods for object-oriented analysis and design have appeared. Object-oriented techniques, however, provide only a partial solution to the problems of software quality and productivity. To adequately address these problems, future software methods will have to go beyond current object-oriented techniques.

Our view is that object-oriented development is an "enabling technology." Object-oriented techniques are necessary for addressing these problems but they are not sufficient. This paper discusses the use of domain engineering and formal methods to augment object-oriented development. The use of these techniques will make it possible to take advantage of the benefits of object- oriented development while overcoming the limitations.

2. Object-Oriented Development

Object-oriented development has contributed to our ability to design and implement software systems in several ways. One significant contribution has been to reduce the conceptual distance between the way that a problem is framed and the way that it is solved. With object-oriented development, software structures directly model problem domain concepts, resulting in systems whose structures more nearly correspond to those of the problem domain. Another important contribution is enhanced reusability of software components through encapsulation and inheritance.

2.1. Limitations of Object-Oriented Development

While object-oriented techniques have proven useful on smaller projects, they have not always been as successful when applied to larger systems. Part of the problem is that object-oriented development does not, in itself, provide a sufficiently powerful set of abstractions. Another aspect of the problem is the informality of current object-oriented methods.

2.1.1. Higher-Level Abstractions

Constructing complex software systems from individual classes and objects is similar to trying to write a non-trivial program in assembly language — the tools are simply not powerful enough. High-level languages provide a higher and more powerful level of abstraction than assembly language. Similarly, higher-level abstractions are necessary in order to apply object-oriented techniques to large, complex systems.

The need for higher-level abstractions is especially apparent when trying to construct systems from reusable components. Two different problems arise here. The first is the size of the component to be reused. The second is a trade-off between generality and power.

The reuse of small components (i.e., individual classes) does not provide significant productivity enhancements. The effort required to locate, comprehend, and modify a component to reuse it may not be much less than that required to develop it.

There is also a trade-off between power and generality in reuse [Biggerstaff and Richter, 1987] . Components that are very general may be reusable in many different applications, but they do not provide enough power to make their (re)use worthwhile. Abstract data types are very general, but reusing them does not provide substantial leverage. Larger components (such as frameworks)

are domain-specific and, therefore, less general. Reusing them, however, provides large gains in productivity.

2.1.2. Informality

Object-oriented analysis and design methods are based on construction of models of the system under development. These models are based on intuitive abstractions and convenient, easy to understand graphical notations. Unfortunately, these notations are almost always informally defined.

This lack of precision means that the models constructed using these object-oriented techniques cannot be effectively analyzed. It is only possible, at best, to perform limited syntactic analysis of specification or design models. With current object- orient modeling techniques, it is not possible to determine whether a model is complete and consistent, exhibits the correct behavior, or is free from deadlock.

2.2. Beyond Object-Oriented Development

The above observations argue for an approach that goes beyond object-oriented development as it is currently defined and used. This approach should focus on the abstractions that are relevant to a particular application domain and emphasize rigorous definition of object-oriented models.

3. Domain Analysis

Systems analysis has traditionally focused on individual systems in isolation. each new problem has spawned its own project in which the cycle of analysis, specification, design, and implementation is repeated. This type of approach focuses on the differences between individual systems from a given problem area or application domain and ignores the commonalty that exists among them. It also ignores the potential for reuse of software products across different applications from within the domain.

Here, the term domain refers to a particular problem area, such as data management or spacecraft navigation. A domain is characterized both by its subject matter, or content, and by the types of problems that are addressed within that subject matter. The domains of airline flight reservations and air traffic control, for example, address similar subject matter (e.g., aircraft and flights) but consider very different kinds of problems (e.g., seat reservations versus movement of aircraft through an airspace). From the point of view of software development, a domain is defined by a collection of (actual and potential) software products that share common characteristics and address a common set of problems.

A domain analysis therefore, is an investigation of a particular problem area or application domain whose purpose is to discover and exploit the common characteristics of a family of related applications within that domain. Domain analysis addresses two significant problems of today's software methods: the "thin spread of application domain knowledge," and software reuse.

3.1. The Thin Spread of Application Domain Knowledge

In their landmark study of the software design process for large systems, Curtis, Krasner, and Iscoe [Curtis, et al., 1988] identified the "thin spread of application domain knowledge" as one of the most significant problems facing today's software development organizations. They noted that accurate problem domain knowledge is critical to the success of a project. This knowledge, however, is difficult to obtain. The problem is compounded by the fact that what domain knowledge does exist is spread thinly and unevenly across project personnel. In the twenty-first century, problem domain knowledge will be even more critical to the success of projects and more difficult to obtain.

Domain analysis addressed this problem by focusing on domains rather than individual applications. Domain models capture domain knowledge that is then available to all members of a project. This knowledge can also be used in training new project members.

3.2. Software Reuse

Domain analysis also makes it possible to identify components that are reusable across a family of similar, or related, applications. Domain analysis also makes it possible to identify a domain- specific architecture for the domain. The domain-specific architecture captures knowledge about the interconnection between classes within the application domain.

Frameworks [Deutsch, 1983] , [Johnson and Foote,

1988] , a well- known concept from object-oriented programming, represent domain- specific architectures. A framework may contain both abstract and concrete classes. The concrete subclasses of an abstract class capture knowledge about variations among different systems within the domain. In some cases, the framework can serve as the basis for a domain-specific software environment that makes it possible for users to design their own systems by selecting from among the possible concrete subclasses of the abstract classes that make up the framework.

3.3. Current Research

Our current research is aimed at developing methods for defining domain-specific architectures and incorporating them into domain- specific design environments. We are focusing on domain separation [Ward and Williams, 1990a] , [Ward and Williams, 1990b] . Domain separation makes use of the observation that any completed application contains components from several different domains. Some components will belong to the primary, or dominant, domain. Others will be from subordinate, or supporting, domains. Domain separation provides a means of simplifying domain models and enhancing reuse. Our work is concerned with formalizing the relationships between domains and between components from different domains. A preliminary view of this work is presented in [Williams, 1991] .

4. Formal Methods

The use of mathematically formal software development methods can help to eliminate design errors in software products. The elimination of such errors is essential to improving quality and productivity in the software industry. It is also essential to maintaining safety in software-based systems that are used in critical applications such as avionics or the nuclear power industry.

4.1. Limitations of Formal Methods

While formal methods have the potential for dramatically improving the quality of software, they have not been widely accepted within the software industry. Some of the most significant reasons for this have to do with the difficulty of using formal methods for developing large complex systems. These include:

- Difficulties experienced by both developers and application domain experts in understanding and using formal notations. Even developers who are experienced with such notations can have difficulty in using them properly [Leveson, et al., 1992] .
- The lack of "methods" (i.e., particular sequences of actions or tasks to be performed) for problem analysis and specification development.
- The lack of adequate support for structuring large, complex, formal specifications to aid in their understanding.
- Difficulty in achieving an appropriate level of abstraction when constructing formal models [Hayes, 1985] .

These problems are especially significant in developing large, complex software systems where understanding the problem and validating the specification are critical issues. The use of formal methods in conjunction with object-oriented development offers a significant opportunity for overcoming these problems.

4.2. Current Research

Our work is aimed at integrating formal methods and object- oriented development. Formal methods can serve as a basis for constructing rigorous, analyzable models of software systems and verifying their refinement into code. Object-oriented development can assist in producing models that are well-structured and easy to understand. Using object-oriented methods as an interface to formal methods can also help to overcome problems with the useability of formal methods.

The integration of formal methods and object-oriented development can also help to improve the definition and use of object-oriented methods. As noted above, current object-oriented methods employ intuitive concepts and informal notations whose semantics are poorly defined. Formal definition of these concepts and notations will help to make object-oriented models more rigorous and enable the construction of automated tools to analyze object-oriented specifications and designs.

The use of formal methods with object-oriented development must, however, be approached with care. Much of the attraction of object-oriented techniques is due to their use of familiar, intuitive abstractions and

straightforward, congenial graphical notations. Our approach is to use formal techniques to define and refine the abstractions used to construct object-oriented models. These abstractions are then mapped to a convenient set of graphical notations drawn, where possible, from those used by existing object-oriented methods.

We have completed work on definition of inheritance of behaviors
[Williams, 1992] . We have also completed a preliminary formal definition of static object-oriented models using an extended version of Rumbaugh's notation [Rumbaugh, et al., 1991] and constructs from Z [Williams, 1993] . Current work is aimed at providing a formal definition of communicating finite-state machines that is compatible with object-oriented concepts and adding real-time capabilities to object-oriented methods.

5. Conclusions

Coping with the increased size and complexity of future software systems will require software methods that go beyond those currently available. While object-oriented development has contributed substantially to improving software quality and productivity, object-oriented techniques, as currently defined, are inadequate for large, complex systems. This paper has discussed the use of domain analysis and formal methods to augment object-oriented technology. Domain analysis addresses the problem of capturing and representing application domain knowledge as well as the problem of reuse. Formal methods address the lack of rigor in current object-oriented methods and help to produce analyzable models.

6. References

[Biggerstaff and Richter, 1987] T.Biggerstaff and C.Richter, "Reusability Framework, Assessment, and Directions," IEEE Software, vol.4, no.2, pp.41-49, 1987.

[Curtis, et al., 1988] B.Curtis, H.Krasner, and N.Iscoe, "A Field Study of the Software Design Process for Large Systems," Communications of the ACM, vol.31, no.11, pp.1268-1287, 1988.

[Deutsch, 1983] L.P.Deutsch, "Reusability in the Smalltalk-80 Programming System," Proceedings of the Workshop on Reusability in Programming, Newport, RI, 1983, pp.72-76.

[Hayes, 1985] I.J.Hayes, "Applying Formal Specifications to Software Development in Industry," IEEE Transactions on Software Engineering, vol.SE-11, no.2, pp.169-178, 1985.

[Johnson and Foote, 1988] R.E.Johnson and B.Foote, "Designing Reusable Classes," Journal of Object-Oriented Programming, vol.1, no.2, pp.22-35, 1988.

[Leveson, et al., 1992] N.G.Leveson, M.P.E.Heimdahl, H.Hildreth, and J.D.Reese, "Requirements Specification for Process-Control Systems," Technical Report No.92-106, Department of Information and Computer Science, University of California, Irvine, CA, November, 1992.

[Rumbaugh, et al., 1991] J.Rumbaugh, M.Blaha, W.Premerlani, F.Eddy, and W.Lorensen, Object-Oriented Modeling and Design, Englewood Cliffs, NJ, Prentice Hall, 1991.

[Ward and Williams, 1990a] P.T.Ward and L.G.Williams, "Domain Analysis: An Example," Technical Report No.SERM-013-90, Software Engineering Research, Boulder, CO, May, 1990.

[Ward and Williams, 1990b] P.T.Ward and L.G.Williams, "Domain Modeling," Technical Report No.SERM-012-90, Software Engineering Research, Boulder, CO, February, 1990.

[Williams, 1993] L.G.Williams, "Toward a Rigorous Definition of Class Diagrams," Technical Report No.SERM-018-93, Software Engineering Research, Boulder, CO, April, 1993.

[Williams, 1992] L.G.Williams, "Inheritance of Behaviors in Object-Oriented Modeling," Technical Report No.SERM-016-92, Software Engineering Research, Boulder, CO, July, 1992.

[Williams, 1991] L.G.Williams, "Modeling Multiple Domains," Workshop on Domain Modeling for Software Engineering, Participant Proceedings, Thirteenth International Conference on Software Engineering, Austin, TX, May, 1991.

Functional Requirements for Future CASE Environments

Kumiyo Nakakoji
(SRA, Boulder & University of Colorado)

1. Introduction

As "downsizing" of computer systems (i.e., transition from centralized main-frame computer systems to distributed workstation-based systems) has been widely recognized [Shimoda 92], demands for small-scale task-oriented software development will increase. In this position paper, I claim that CASE tools for such software development must provide a communication medium among system developers and end-users with an emphasis on conceptual coordination rather than on physical coordination. We are interested in creating a shared understanding among stakeholders that allows both clients and software developers to contribute their respective knowledge to the task. The CASE architecture that I propose in this paper integrates: (1) tools and technologies that allow end-users to store and access informal information about their tasks in the domain and about a system to be developed, (2) tools and technologies that allow system developers to develop the new system that supports the users' tasks, and (3) tools and technologies that allow the stakeholders to assess and evaluate the partially developed software artifact.

2. Problems

Traditional CASE approaches pay little attention to conceptual coordination among stakeholders. Existing CASE tools only support system developers to develop a new system according to the system requirements that are assumed to contain complete requirements of end-users. However, empirical evidence has shown that it is impossible to have complete specifications because requirements fluctuate over time and conflict with each other [Curtis, Krasner, Iscoe 88]. In producing customized task-oriented software development, asking end-users to produce a complete specification is not feasible because of (1) communication breakdowns between end-users and software developers [Greenbaum, Kyng 91; Fischer, Nakakoji, Ostwald 93], and (2) the ill-defined and open-ended nature of software design [Fischer et al. 92]. Current tools do not take into account that software development is a design task, which requires collaboration among different stakeholders [Fischer, Nakakoji, Ostwald 93], and is inherently ill-structured and open-ended [Nakakoji 93]. The current approaches are object-centered, because their methods and tools are organized in terms of software objects to be developed, and not human-centered.

2.1. Communication Breakdowns between End-Users and Software Developers

System requirements originate from end-users, who are not familiar with available computational resources and terms used in describing available options to implement a system. Software developers, who have to bridge the great transformation distance between a generic programming language substrate and high-level abstract domain concepts, are expected to understand the domain, which is at best described by end-users using their specialized domain language.

Initially, there is a symmetry of ignorance between software developers and end-users, in which 1) the understanding required to solve the design problem is distributed, and 2) there is no common language that allows the stakeholders to communicate their understanding to each other [Greenbaum, Kyng 91]. Thus, not only the problem of the thin spread of application domain knowledge among software developers [Curtis, Krasner, Iscoe 88] but also the problem of little understanding about computer technology among end-users [Ehn 88] cause communication breakdowns between end-users and software developers.

Traditional CASE approaches require software developers to be able to understand tasks of end-users represented in a situation model and to map them into a system model that a computer system can manipulate [Fischer, Henninger, Redmiles 91]. My claim is that future CASE tools must also be used to allow end-users

to learn about available options with regard to computer technologies. This mutual education processes enable the construction of a shared "language of doing" that allows end-users and software developers to collaboratively build the knowledge required to solve the design problem [Fischer, Nakakoji, Ostwald 93].

2.2. Ill-Defined and Open-Ended Nature of Software Design

Software development is a type of design problem. Theoretical foundations provide ample evidence that design problems are ill-defined where one cannot specify a design problem completely before starting to solve it [Rittel 84; Simon 81]. Understanding what the problem is plays a major part in design activities. Starting with a vague and incomplete problem requirement, designers sketch out a partial solution. By seeing the partial solution, designers identify portions of the problem that have not yet been understood, gain an understanding of the problem, and then refine the solution [Snodgrass, Coyne 90]. By iterating this process, understanding of the problem gradually emerges. This process characterizes design as a reflective conversation [Schoen 83] with the materials of design construction.

Design problems are open-ended in that the knowledge necessary for a design can never be completely articulated a priori. It is neither possible to identify all the relevant knowledge for the design nor to formalize it for generating a design that fits to varieties of changing needs of design tasks. It is neither possible for end-users to articulate all the relevant information about the domain nor for software developers to articulate all the relevant information about available computational technologies because some knowledge always remains tacit. People know more than they can tell [Polanyi 66], and they tend to give inaccurate descriptions of what they know, and they tend to be unaware of what they know [Schoen 92].

3. Limitations of Current CASE Approaches

Because of the aforementioned problems, "design disasters" occur, in which the implementation is correct with respect to a specification that is wrong. Although the waterfall-type development model is still widespread, mainly due to managerial concerns, there has been a shift of emphasis from "downstream" activities (transformation of formal specifications into implementations) to "upstream" activities (development of specifications in order to decide precisely what to build) [Belady 85]. Future CASE must integrate both upstream and downstream activities by developing shared understanding for all stakeholders in software development [Fischer et al. 92].

Existing CASE approaches rely on the model based on written documents, formalized languages and prototypes, all of which are produced by software developers. For identifying system requirements, software developers have to analyze the task domain, and produce explicitly stated system requirement specifications. Since it is impossible for end-users to articulate all the relevant knowledge about their problem in the first place [Polanyi 66], and it is impossible for software developers to completely articulate all of their understanding about the problem using symbol systems [Winograd, Flores 86], such specifications can never be complete.

Thus, such CASE tools at best support communication as one way feedback from system developers to end-users. As discussed above, software development is an ill-defined design problem, where one needs to coevolve both problem specification (i.e., system requirements) and solution construction (i.e., final products). A partially constructed solution developed in a CASE environment may trigger end-users to reframe their ideas about what the new system should be, but the current approaches do not allow them to change the requirements, or even if they do, the modifications are not recorded within the CASE environment. Most importantly, such system requirements in end-users' mind are typically informal and vague, while the existing CASE tools often requires the system requirement specification to be explicitly and formally stated with a full commitment.

4. Approach

In their article, Kensing and Munk-Madsen [1993] suggest to model software development processes in terms not only of work products (i.e., documents, design and code), but also of the "knowledge" developed as results by the people involved. They suggest to model the knowledge in two dimensions. One is in terms of three categories about tools, technologies and results,

Table 1: Functional Requirements for Future CASE Environments

	Domain	Design	Implementation
Informal Representations	Descriptions of Users' Task	Design Rationale	Prototype Systems
Formal Representations	Domain Model	Design Description	Final Systems

consisting of (1) users' present work, (2) technological options, and (3) a new system. The other is in terms of levels of abstraction and specificity, such as (a) abstract knowledge and (b) concrete experience.

Based on the model, I propose functional requirements for a future CASE environment that include six types of representations necessary in software development, consisting of domain, design, and implementation, both formally and informally stated (see Table 1).

Thus, future CASE environments must support and integrate all of the six representations so that system developers and end-users can freely explore the representations relevant to their task at hand. Techniques and tools have been explored and are available that support each of these aspects but they have not been fully integrated as a CASE environment. Existing CASE environments support formal representations and prototypes, while existing design rationale recording systems such as gIBIS [Conklin, Begeman 88], SIBYL [Lee 90], and DesignRationale [MacLean, Young, Moran 89] are isolated from those CASE tools. Informal techniques to understand users' task domain, such as observations and interviewing users, have been found effective [Ehn 88; Greenbaum, Kyng 91], but no system support is provided as a tool in a CASE environment.

In a CASE environment that fulfills these requirements, end-users have access to informal representations such as informally stated users' task domain, design rationale, and prototype systems. End-users may actually input informal representation of the domain for themselves. Software developers map informally stated tasks into a domain model using techniques such as the object-oriented analyses technique [Prieto-Diaz, Arango 91], produce design document using various object-oriented design techniques and notations [Meyer 88; Booch 91; Monarchi, Puhr 92; Korson, McGregor 90], and implement the system [Stefik, Bobrow 86]. While designing, system developers may record design rationale [Lee 90; Conklin, Begeman 88], which is accessible by end-users, and partially constructed programs can be used as prototypes [Boehm, Gray Seewaldt 84; Bodker, Gronbaek 91]. Design rationale and prototypes can be understood by end-users and may trigger them to gain a deeper understanding of their problem and domain. Based on this feedback, the software developers reframe their domain models, design, and implementations. Thus, software developers and end-users coevolve system requirements and final products while gaining shared understanding about the problem domain and recording the process of increasing understanding.

5. Challenge: System Building Efforts

Given existing technologies that individually support each functionality, challenge is how to integrate those tools and mechanisms not only at data interface levels but at a level that provides conceptual coherence.

We propose to use a hypermedia system to integrate different types of representations, consisting of nodes, each of which stores a representation or a link to a representation, and links among the nodes. Studies have shown that structure supports people to understand complex information spaces [Kawakita 84; Anderson 83]. People understand a representation by seeing how the representation is connected to others [Greeno 89; Norman 93]. Thus, if heterogeneous representations (e.g., domain models, design descriptions, design rationale, and source code) are stored in a CASE environment and interconnected, then both end-users and system developers will understand how the representations are used, applied, modified, and evolved by traversing links among the representations.

Since building CASE environments is yet another

software development, we wanted to implement our ideas and evaluate it in order to assess the suggested approach. In our research group, the EVA (Evolving-Artifact-Approach) approach has been proposed and applied to build the EVASERVICE system [Ostwald, Burns, Morch 92], which supports NYNEX (a local telephone company for New York region) service order representatives to perform the complex task of service provisioning.

In the EVA approach, descriptive representations of application domain concepts provide a context for understanding system requirements, and functional representations are used to guide implementation of system functionality. The representations facilitate the mutual education process between end-users and software developers necessary for achieving a deep knowledge of the application domain, for capturing design rationale, and for representing domain knowledge. Both descriptive objects and functional objects serve as a final software artifacts when completed [Ostwald 93; Fischer, Nakakoji, Ostwald 93].

The EVA system is built on top of CONCORDIA, a hypermedia interface on SYMBOLICS, as a substrate. The textual description of their domain (i.e., the service provisioning operations) was first encoded by EVA. Then, while system developers and end-users (i.e., service representatives) browsed the hypermedia space together by using EVA, the end-users articulated which kind of functionality was necessary for a portion of task description in the hypermedia. The system developers implemented a set of small functionalities in CLOS, some of which were accompanied with informal annotations (such as natural language text, or graphics) as design rationale, and each functionality was linked to a node in EVA that described the portion of the original task descriptions. Thus, function calls were associated with graphical objects, so that clicking on the object would execute the function. Embedding the functional representations in descriptive representations provided a context for understanding the functionality. The outcome was the fully annotated chunks of functionalities which were connected through the hypermedia substrate.

Applying the EVA approach to building EVASERVICE, descriptive and functional representations served two purposes: (1) they elicit knowledge about the task domain from end-users that is often tacit and therefore not expressible in abstract situations, and (2) they communicated the intentions of system developers to the end-users. In both cases, the explicit representations have been found crucial.

6. Summary

The CASE architecture discussed in this position paper is for small-scale, task-oriented, customized software projects, and may not be suitable for developing software where system requirements can be fixed and stated formally, or where end-users are not available to system developers, such as contract-based large scale software development, or commercial packaged software development. However, envisioning a "Software Development Paradigm in the 21st Century," I have confidence that the demand for small task-oriented software will increase. Computers should be viewed as not only computational tools that supports people to do a task, but also as media that facilitates human-human communications and collaborations. This position paper does not propose innovative technologies to be used in future CASE tools; rather, I claim that we have to integrate those existing technologies for supporting our work as software developers by providing structures among heterogeneous representations using hypermedia technologies.

References

[Anderson 83] J.R.Anderson, *The Architecture of Cognition,* Harvard University Press, Cambridge, MA, 1983.

[Belady 85] L.Belady, *MCC: Planning the Revolution in Software,* IEEE Software, Nov. 1985, pp.68-73.

[Bodker, Gronbaek 91] S.Bodker, K.Gronbaek,*Cooperative Prototyping: Users and Designers in Mutual Activity,* International Journal of Man-Machine Studies, Vol.34, Mar. 1991, pp.453-478.

[Boehm, Gray Seewaldt 84] B.W.Boehm, T.E.Gray, T.Seewaldt, *Prototyping vs. Specifying: A Multi-Project Experiment,* Proceedings of the 7th ICSE, pp.473-484.

[Booch 91] G.Booch, *Object Oriented Design with Applications,* The Benjamin/Cummings Publishing Company, 1991.

[Conklin, Begeman 88] J.Conklin, M.Begeman, *gIBIS: A Hypertext Tool for Exploratory Policy Discussion,* Transactions of Office Information Systems, Vol.6, No.4, October

1988, pp.303-331.

[Curtis, Krasner, Iscoe 88] B.Curtis, H.Krasner, N.Iscoe, *A Field Study of the Software Design Process for Large Systems,* CACM, Vol.31, No.11, November 1988, pp.1268-1287.

[Ehn 88] P.Ehn, *Work-Oriented Design of Computer Artifacts,* Almquist & Wiksell International, Stockholm, Sweden, 1988.

[Fischer et al.92] G.Fischer, A.Girgensohn, K.Nakakoji, D.Redmiles, *Supporting Software Designers with Integrated, Domain-Oriented Design Environments,* Transactions on Software Engineering, Vol.18, No.6, 1992, pp.511-522.

[Fischer, Henninger, Redmiles 91] G.Fischer, S.R.Henninger, D.F.Redmiles, *Cognitive Tools for Locating and Comprehending Software Objects for Reuse,* Proceedings of the 13th ICSE, 1991, pp.318-328.

[Fischer, Nakakoji, Ostwald 93] G.Fischer, K.Nakakoji, J.Ostwald, *Facilitating Collborative Design Through Representations of Context and Intent,* Working Notes of the AAAI 1993 Workshop on AI in Collaborative Design, July 1993.

[Greenbaum, Kyng 91] J.Greenbaum, M.Kyng (eds.), *Design at Work: Cooperative Design of Computer Systems,* Lawrence Erlbaum Associates, 1991.

[Greeno 89] J.G.Greeno, *Situations, Mental Models, and Generative Knowledge,* in D.Klahr, K.Kotovsky (eds.), *Complex Information Processing: The Impact of Herbert Simon,* Lawrence Erlbaum Associates, 1989, pp.285-318, ch.11.

[Kawakita 84] J.Kawakita, *Hassouhou (Abduction - How to Develop Creativity),* Chuuou Kouron, 1984, (in Japanese).

[Kensing, Munk-Madsen 93] F.Kensing, A.Munk-Madsen, *PD: Structure in the Toolbox,* CACM, Vol.36, No.4, June 1993.

[Korson, McGregor 90] T.Korson, J.McGregor, *Understanding Object-Oriented: A Unifying Paradigm,* CACM, Vol.33, No.9, September 1990, pp.40-60.

[Lee 90] J.Lee, *SIBYL: A Tool for Managing Group Decision Rationale,* Proceedings of the Conference on CSCW, 1990, pp.79-92.

[MacLean, Young, Moran 89] A.MacLean, R.Young, T.Moran, *Design Rationale: The Argument Behind the Artifact, Human Factors in Computing Systems,* CHI'89 Conference Proceedings, 1989, pp.247-252.

[Meyer 88] B.Meyer, *Object-Oriented Software Construction,* Prentice Hall, 1988.

[Monarchi, Puhr 92] D.E.Monarchi, G.I.Puhr, *A Research Typology for Object-Oriented Analysis and Design,* CACM, Vol.35, No.9, September 1992.

[Nakakoji 93] K.Nakakoji, *Increasing Shared Understanding of a Design Task between Designers and Design Environments: The Role of a Specification Component,* Unpublished Ph.D.Dissertation, Department of Computer Science, University of Colorado, 1993, Also available as TechReport CU-CS-651-93.

[Norman 93] D.A.Norman, *Things That Make Us Smart,* Addison-Wesley, 1993.

[Ostwald 93] J.Ostwald, *Evolving Artifact Approach to Information Intensive System Design,* Dissertation Proposal, Department of Computer Science, University of Colorado, at Boulder, July 1993.

[Ostwald, Burns, Morch 92] J.Ostwald, B.Burns, A.Morch, *The Evolving Artifact Approach to System Building,* Working Notes of the AAAI 1992 Workshop on Design Rationale Capture and Use, 1992, pp.207-214.

[Polanyi 66] M.Polanyi, *The Tacit Dimension,* Doubleday, 1966.

[Prieto-Diaz, Arango 91] R.Prieto-Diaz, G.Arango, *Domain Analysis and Software Systems Modeling,* IEEE Computer Society Press, 1991.

[Rittel 84] H.W.J.Rittel, *Second-Generation Design Methods,* in N.Cross (ed.), *Developments in Design Methodology,* John Wiley & Sons, 1984, pp.317-327.

[Schoen 83] D.A.Schoen, *The Reflective Practitioner: How Professionals Think in Action,* Basic Books, 1983.

[Schoen 92] D.A.Schoen, *Designing as Reflective Conversation with the Materials of a Design Situation,* Knowledge-Based Systems Journal, Vol.5, No.1, 1992, pp.3-14.

[Shimoda 92] H.Shimoda (ed.), *Revolution of Computer Usage: All About Downsizing,* Yomiuri Shinbun-sya, 1992, (in Japanese).

[Simon 81] H.A.Simon, *The Sciences of the Artificial,* The MIT Press, 1981.

[Snodgrass, Coyne 90] A.S.Snodgrass, R.Coyne, *Is Designing Hermeneutical?,* Technical Report, Department of Architectural and Design Science, University of Sydney, Australia, 1990.

[Stefik, Bobrow 86] M.J.Stefik, D.G.Bobrow, *Object-Oriented Programming: Themes and Variations,* AI Magazine, Vol.6, No.4, Winter 1986.

[Winograd, Flores 86] T.Winograd, F.Flores, *Understanding Computers and Cognition: A New Foundation for Design,* Ablex Publishing Corporation, 1986.

泰安紀行

中野 秀男
(大阪大学)

10月12日(火)

今日から10日間の中国旅行である．例によって前日に「正露丸」や風邪薬を揃え，明け方まで大学で10日間の仕事の段取りをすませて，一眠りしてから大阪空港へ．今回は，成田ではなく，大阪発なのでちょっと楽である．

坂本さん(オムロン)，名古屋から新幹線の伊藤さん(MASC)，アメリカから数日前に来阪していたL.williamsさん(SER)と空港で待ち合わせ．羽田から国内線で飛んで来た山崎プログラム委員長(前·日本ユニシス)たち本隊とは出発ゲートで合流して上海へ．昼過ぎに上海着．

熊谷総経理(PFU上海)，張然先生のお出迎えを受ける．前日に「下海」したSRA Boulderの中小路女史の顔も見える．済南行きの国内便が夕方発で，かなり時間があるので，とりあえず新錦江飯店へ．荷物はPFU上海の若手社員が空港で張り番をしてくれるとのこと(この国際空港には一時預けの設備がないらしい)．ごくろうさま．

マカオからすでに上海入リしていた国連大学のD. Bjorner夫妻とホテルで合流．山崎さんたちに誘われてタクシーでPortman Hotelに行き，さっそくショッピング·モールをうろついて，おすすめの不老長寿(安眠)枕を買う(家に持って帰ったが，残念ながらあまり使う暇がなく，枕だけが今バルコニーで寝ている)．

夕刻，上海空港に戻り，夕食(当然中華)．アメリカから明日到着予定のL.Beladyさん(MERL)を迎えるために残留の岸田さん(SRA)を残して，空路，山東省の省都·済南へ．到着前に平衡感覚を失い吐く．疲れと飲みすぎのせいだろう．済南着は夜．出迎えのマイクロバスが若干道に迷ったが(運転手が地元の人でなく，泰安から迎えにきているため)，なんとか宿舎の南効賓館に到着．そのまま就寝．

10月13日(水)

済南から泰安へマイクロバスで移動．出発前にホテルの庭園をバスで見学．とても歩けないくらい広い．済南は湖，泉で有名とか．泰安への道路は有料，結構高速(?)で泰安·御座賓館に到着．平屋または2階建で10～20室程度の建物がいくつもあり，グループ毎の宿泊には便利そう．昼食後にシンポジウム会場の山東鉱業大学に移動．いつものことながら昼食には(夕食は当然だが)ビールが出る．ここのビールは飲める．今年はピッチャーに入ったビールがあった．

午後，いよいよTICS'93の開幕．ホストである山東鉱業大学の呉哲輝先生，実行委員長の居徳華先生(華東理工大学)，プログラム委員長の山崎利治さんの挨拶に続いて，3人のスピーカによる基調講演．

楊芙清(北京大学):
- 国家プロジェクト: "Jade Bird(青鳥) System 1 & 2"

Dines Bjorner(国連大学):
- PRaCoSyプロジェクト(Formal Approachによる鉄道制御)

Lloyd Williams (SER):
- オブジェクト指向を越えて(21世紀のソフトウェア開発技法)

夕方，御座賓館に戻ってレセプション．SEA代表幹事として最初に挨拶．英語を用意したのだが，日本語でもよいとのことで，張先生には中国語に，中小路さんには英語で通訳してもらう．結局，英語で用意したことしかしゃべれず残念．レセプション後連れだって街へ．ビールと醤々麺と餃子．ホテルに帰って風呂に入ろうとしたが，湯の汚れがいくらたっても取れないので，しかたなくシャワー．あとは恒例のBear's Bar(熊谷さんの部屋)でナイトキャップを飲んで就寝．

10月14日(木)

朝，上海から泰安まで15時間の夜行寝台列車で，Belady，岸田の両氏が到着．

シンポジウム2日目．午前の最初のセッションは，昨日と同じ山東鉱業大学で2人の基調講演．

中小路久美代(SRA):
- 未来のCASE環境の機能的仕様

Les Belady(MERL):
- 来るべきインターディシプリナリな世界にお

けるソフトウェア技術

その後，場所を御座賓館に変えてグループ討論．グループは次の4つであった：

(A) Concept Creator's View
(B) System Architect's View
(C) System Implementor's View
(D) User's and Manager's View

私が参加したのはグループAであったが，予定していたメンバの何人かが欠席のため，リーダの熊谷さんを含めて5名という小人数．午前·午後に分けてそれぞれがプレゼンを行い，質疑·討論．3時過ぎに終って，あとはグループ全員で市内見物をするはずが，結局熊谷さんと2人だけになる．ホテルの隣の岱廟をゆっくり見学．

ここは，歴代の皇帝が泰山参詣の際に宿泊した所である (われわれが泊まっているホテルの名前も，御座所の御座に由来する)．今回は「地球の歩き方：中国篇」の関係部分を拡大コピーして持ってきたので，見学のポイントは比較的よくわかった．中国3大建築物のうち2つを今回見ることになっている (この岱廟と明日行く予定の孔廟)．すでに一昨年北京で，そのうちの第一番札所·故宮を見ているので，規模では驚かない．しかし2500～5000年の歴史の重さには，いつもながら驚かされる．ちょうどいま，岩波文庫の「春秋左氏伝」を読んでいるのだが，その歴史の舞台に来ているかと思うと感無量．現実に斉·魯の国が自分の足の下にあるというのは，不思議な感じである．

岱廟の表門を出たあとは，市内をブラブラしてホテルに帰る．熊谷さんとは共通な事項や友人がいるだけに，散策途上の話題はつきない．今日は，昼·夕とも食堂に入るのは，私と熊さんとが一番．よほどお酒が好きだと給仕係の小姐に思われているのだろうか，席につくとすぐビールが来る．夕食の時は，早すぎたので2人で乾杯．ここまでは，「日本から導入された近代技術が中国銘酒をダメにした」という山崎さんのアドバイスにしたがって，中国酒は紹興酒以外飲んでいない．夕食後は，久しぶりに仕事をしたあと，Bear's Barで午前2時まで．

10月15日 (金)

今日はシンポジウム参加者全員で，孔子の故郷 (儒教愛好家の岸田さんにいわせると「ソフトウェア工学原理発祥の地」) である曲阜へエクスカーション．例によってバスの最後尾に座ってバウンドを楽しむ．昨年，トルファンに行った時は，頭を天井に打った人もいたが，今回は主要道路を通ったせいか，工事中の場所を除けば，道路事情は比較的よい．

午前中は，案内書を片手に孔子廟，孔府と回る．孔府の後楼は孔子の居所だとか．第2次大戦後，直系の子孫が台湾に逃げるまで77代続いたそうだ．孔子が生きていたBC400年ごろは，日本史ではただ縄文·弥生時代としか習わなかった．中国の歴史には恐れいる．

門前の屋台店で山崎さんが人民帽，坂本·岸田両氏がレーニン帽を買い，共産主義の亡霊がうろつき始める．伊藤さん，中小路さんたちは，廟内郵便局の土産品販売部でハンコを彫って貰ったようだ．

昼飯の後の一休みは，例によって熊谷·山崎組が近くの屋台で「豆腐鍋＋ビール」しているので，誘われて参加．そのせいか，有料公厠 (公衆トイレ) に初めて入る．使用料の2角 (日本円で4円) は高いのか安いのか (!?)．路上の店で，見かけは大根のようだが，中が赤くて竹刀がバラバラになったように切ってある果物 (または野菜) をたくさん売っている．杉田さんが買ったのを一切れもらって食してみたが，やっぱり大根．

次なる見学スポット孔林は，孔子の墓がある所で，孔子が亡くなったあと弟子たちが3年，一番弟子の子貢は6年間墓守をしたそうである．お墓の横で書の実演販売をしているおじさんたちがいて，当然のことながらすごく達筆．

泰安への帰路のバスでは最初から最後まで睡眠．夕食まで1時間程寝る．大学に直通の電話を入れるが，計算機室には人がいなくてかからず．例年私が長期出張をすると必ず計算機がつぶれるのだが，夜に電話を入れたところでは問題はないようだ．昨年は，ウルムチに着いてすぐに計算サーバが落ちて，1ヶ月程隣接のjunetサイトに迷惑をかけた．

夕食後は山東鉱業大学の学生さん3人と山崎·熊谷·中野組の日中友好囲碁大会．三子から四子ぐらい腕前が違う感じで，全員惨敗．9時ごろ囲碁大会が終ったので，のんびり風呂に入る．その後，北京大学の揚先生が早く帰ったために空いたスイート·ルームへ．もはや持ち込んだ酒が切れてきたので，3人ほどが街へビールと餃子を買いに行く．私は杉田さんと囲碁．今日は12時におとなしく寝る．どのチャンネルを回し

ても，サッカーの日本対サウジアラビア戦はやっていない....

10月16日(土)

朝早くから目がさめていたが，7時前にうとうとしたら7時20分．今日は朝食を抜こうかと思ったが，少しは食べることにして先に身支度をしておく．初日のオープニングと今日のクロージングは，ブレザーにネクタイ着用である．ワイシャツは2枚しか持ってきていないのでこれで終り．

今日のセッションは，まず，Bjorner先生のレクチャ．オープニングで喋り足りなかった分の補足ということで，いま，中国と共同でマカオの国連大学研究所で進めている中国の鉄道制御システム整備プロジェクト(PRaCoSy)を例題として，システムの要求分析･定義にフォーマル･スペックをどう応用すべきかについて，熱弁を振われた．私自身，昔はグラフ理論屋だったので，けっこう面白く聞いたが，司会役の山崎さんが途中で何度も後ろを見ていたのは，居眠りしていた人が多かったのかな．私は代表幹事の立場上，今回のシンポジウムのキーノートはすべて最前列で聞いていたが，結果は4勝3敗であった．

最後の締括りは，岸田さんの司会で「21世紀におけるCASEの問題点と解決策」というパネル．フロアからの質問も活発で，けっこう盛り上がった．ところで，このパネルで松原さんがISO-9000周りの議論を紹介したのだが，それをフロアで聞いていたBjorner先生が，セッション終了までの短い時間に，開発途上国を対象とする技術標準化ワークショップの計画メモをまとめあげ，演壇から降りて来た松原さんをつかまえてさっそく相談をはじめた早業には，脱帽!

御座賓館に戻っての昼食の食堂は，昨日まで行われていた重要会議が終わったためか，新しい主食堂に変わる．ここのほうが感じがいい．夏に山崎･熊谷両氏が下見に来た時はまだできていなかったそうだ．予定では，基調講演などすべてのセッションをここ御座賓館でやるつもりだったのだが，全員収容可能な会場がまだ工事中だとのこと．

昼食後，これから北朝鮮へ向かうというBjorner夫妻，それにBelady，中小路，清水(ユニシス北京)のみなさんは，明朝一番の飛行機で北京へ飛ぶため，バスで済南に向かった．

田舎町らしく，泰安でのおすすめはリンゴ，薩摩芋，豆腐など．とくにリンゴはおいしくて，ふだん果物ナイフなど持ったことがない私でも，部屋のリンゴを今日までに2つも食べてしまった．食事に関しては，今回は野菜を多くとることを心がけた．

昼食後，まとめのPCミーティングの終了を待って，日本人ほぼ全員揃って，革命烈士記念碑と晋照寺(Temple of Universal Illumination)を目指して散歩に出る．

いつもは何気なく使っているお寺の名前も，ただの固有名詞と考えれば1つのシンボルにすぎないが，普照(あまねく照らす)をそのまま英語に直して'Universal Illumination'などと書かれると，頭ではそうかと思うが，何か変な感じがする．今回のシンポジウムの議論でも文化(Culture)の問題が出たが，東洋と西洋は違うし，日本と中国はやはり違う．東京から見れば関西と一括されるが，京都と大阪と神戸とは，私から見ると明らかに異なる文化圏である．歴史や街並などが生活以外にも影響しているのだと思う．

中小路さんが最後のパネルのために"Rigorous"というコトバの反対語を前日の夜考えていて，われわれにも意見を求められたのだが，西欧的な考え方の"Rigorous"の対極にある東洋的な"あいまいな"や"ばくぜんとした"あるいは"非論理的な"といった単語は，和英辞書を引けば何か出て来るだろうが，はたして西欧圏でそれぞれの民族が考えている単語の意味するところと，われわれの感じるニュアンスとは違うだろう．Bjorner先生は"Sloppy"でどうか，と会場から答えていたが，辞書を引いてみると悪い意味らしい．

晋照寺には，馮玉祥将軍の記念館があり，関係する史料がたくさん展示されていた．この人は，日本では「赤い馬賊」と呼ばれていた軍閥の頭領で，中国共産党と親しく，一時この寺に篭って晴耕雨読の生活を楽しんだらしい．

中国でいろいろな街を訪れると，堯･舜時代から毛沢東にいたるまで，由緒正しい史跡名勝がどこにでもあって，いつもながら感心する．そうした環境の中で，中国の人びとの生活は，ほんとうにゆっくりしたペースで進歩(このコトバの意味も難しいが)しているように思われる．

帰途，お酒を補給する．山崎･岸田両氏があれこれ相談して，一応無難そうに見える白酒を1本買い求め，さっそく夕食のテーブルに供する．私も4杯程乾杯したが，夜，胃が少しおかしくなった．岸田さんいわく

「2時間ほど我慢すればよい」. それでは酒ではないような気がするが....

10月17日(日)

今日は, ネズミを求めて(!)朝から泰山に登る. 中腹の中天門までバスで行って, ロープウェイで山頂近くの南天門まで行く予定らしい. しかし, 中国政府の要人か外国の賓客かが来ているらしく, ときどきマイクロバスが止められる. 山の入口で入山料の支払等で少しもたもたしたが, 予定どおり中天門へ. そうした小停止の間でも, 坂本さんは店屋を見つけて探索に出かける. 今回のメンバのなかでは一番元気で一番買いまくったようで, みんなから「バイヤー・サカモト」と仇名されていた.

泰山はもともと中国の皇帝が天下を我が手中におさめたことを天帝に報告するため, 岱廟で封禅の議を行ない, そのあと泰山に(駕篭で)登ったらしい. 孔子も登ったとか? 岱廟にあった展示説明によれば, この儀礼は夏時代あるいはそれ以前にまで戻るらしいが, 真偽のほどはわからない.

中天門のロープウェイ乗り場に皆でならんだが, 恐ろしい長蛇の列. そういえば, 今日は日曜日だ. 2時間以上は待つらしいと聞いて, 私も含めて10人以上が南天門まで歩いて登ることになった. 中国の方を除けば, 杉田, 松原, 坂本, Williamsの各氏と私の5人であった. 道がほとんど石段だったため, 歩行のペースをつかむことができず苦労した.

1時間15分で南天門へ, 途中で5分程度の休憩をとっただけだった. 休憩をとらなかった杉田さんはさらに早く, 途中ビデオ撮影等をしながらゆっくり登った坂本さんが2時間ぐらいであった. 松原さんが登ったのには拍手.

泰山は泰安の町から登ると垂直で1500m, 石段は何と7千数百段. 北京にある日本ユニシスと中国の合弁会社から参加された清水さんは, われわれより早く泰安に来られて, シンポジウムの前に麓から徒歩で登り, 山頂で御来光を拝まれたそうだが, さすがに足が張ったといっていた.

話では, 山頂はかなり寒いとのことだったので, Tシャツの上に厚手の長袖シャツを着て, さらにジャンパーの上下といういでたちで, ナップザックには予備のトレーナーを入れるという完全装備を用意したが, 幸い晴天で, むしろ汗ばむくらいの陽気で, 登りでは途中からTシャツだけになってしまた. 南天門直前の最後の石段はまさに胸突き八丁の急傾斜. その入り口は昇仙門という名前で, ここを越えれば仙人だそうだが, 仙人になるにはまだ少し早いような気がする.

南天門で, 熊谷さんらロープウェイ組の到着を1時間程待って合流し, 山頂近くのホテルで昼食. いつもながら若干名はすぐにビール. もう少し冷えていれば最高だとは贅沢な意見である.

山頂を1時間程度散策. 何年か前の環境ワークショップ(@ 釧路)のとき, 岸田さんがモジュール・エンキャプシュレーションの原理書だ(?)と推薦してみんなに読ませた「封神演義」(翻訳本は講談社文庫・全3巻)に登場する仙人・碧天霞元君を祭ってある神社(でいいのかな?)では, 中国式に膝をつき拝礼をしてみたが, どうやれば正統なのか, 中国の人のやり方もバラバラでよくわからん.

下りも中天門まで歩いて降りる. 今度は岸田, 山崎, 熊谷, 伊藤, 張先生等も参加. 下りは約1時間. 山崎, 熊谷組は例によって途中の屋台でビールを買って飲んでいるという情報があったので, 一心不乱に駆け下りた岸田、杉田, 伊藤, 私の4人は, 中天門商店街のレストラン前の路上テーブルで, 山東風の兎肉料理をつまみに, のんびりビールを1杯.

店の小姐たちの話すコトバは, 素人の耳にも少しなまりがあるように聞こえた. われわれの国籍がわからないらしいので, 筆談で日本人だと教える. 熊さんたちはきっと1時間以上遅れるだろうとたかをくくっていたのだが, 案外早く降りて来たそうで, 中国の学生さんが呼びに来る.

夕食のあと日本側のPCメンバと代表幹事の私に, ローカル・アレンジメント担当の呉先生(山東鉱業大学)が記念品を持って来てくださる. 神亀を形どった飾り墨と山東銘酒のセット. 今日ロープウェイで2時間以上待たせたことはまことに申し訳ないと, 最初から最後まで謝っておられた. 聞けば, 大学の事務局(外事弁公室)から正式に書類を出しておけば, VIPとして優先的に乗れたのだそうだ. また, ある学生の父親が中天門のロープウェイ乗場に勤めているので, そこに一声かけるべきであった(この辺はいかにも中国的!)などと, 今日のバスに随行した助手の人を叱っておられたが, 私としては, 当初のもくろみ通り自分の足で石段を登れたので, 結果的にはよかった. 地元の学生たちはもう20回も登ったとか.

10月18日(月)

朝食後，マイクロバスで済南へ．岸田，Williams，熊谷の3人によれば，路の両側に低い山並みが続く川沿いの風景は，コロラド州ボールダーの周辺によく似ているそうだ．道路脇にはところどころ，中国式ドライブイン・レストランが店を並べていて，何年か後に来てみたらどうなっているだろうと想像すると楽しい．

途中，道路工事などもあって混んでいたのと，済南東駅から列車で北京へ帰る中国の方々を降ろしたりで，南効賓館到着は正午近くであった．月曜日には上海行きの飛行機が飛ばないので，今日はここにまた1泊しなければならない．バイキング形式の昼食の後は済南市内見学となる．

山東省博物館は移転準備のため残念ながら休館中であったが，地元出身の画家の旧居を利用した絵画館を見たあと，市内にいくつもある有名な泉の1つを囲む公園に行くと，ここにも小さな美術館があり，民族芸術のお面が安かったので買い求める．その後は大明湖のほとりを散策．

車で30分程の距離に黄河があり，歩いて渡れるそうなのだが，先に帰った中小路さんからの報告によると，いまは橋が工事中で駄目だとのこと．中国の大河は日本とスケールが違うので楽しみにしていたのだが，残念．

夕食後，ホテルのマイクロバスを借りて，夜の街を冷やかしに行く．翌日，山東大学を訪問することが急に決まり，もう洗濯ずみワイシャツの在庫が切れたので，1枚15元(300円)の一番地味そうな柄のを，夜店で買う．しかし，ホテルに帰って，明るい電灯の下で見てみれば，やはりハデであった．毎年中国で買う衣料品は，他の人は知らないが，私の場合はわりあい長持ちしている．着るものに無頓着ということもあるかもしれないが，今回のワイシャツはどうだろう?

子供のお土産(その時は自分用だった)に，昨年西安空港で買いそこねた中国語で時を告げる腕時計を45元で購入．翌日，山東大学購買部で55元でもう1個購入．ホテルの売店では103元であった．

火鍋の屋台で，どんな具があるのか覗いてみると，セミの幼虫(抜け殻ではなく中身がはいっている)や，蛾のさなぎ，ヤゴ(とんぼの幼虫)，冬虫夏草など，ゲテものがたくさん並んでいる．まあ，グラグラ煮た鍋に入れて食べるのだから，なんとか食べられるかもしれないと思う．

岸田さんが，ウイロウの元祖のようなお菓子を何種類か買う．ココナッツ味とか，栗風味，胡麻味など，ホテルへ帰って食べてみたが，けっこういける味であった．

10月19日(火)

朝，山東大学計算機系を訪問．私と山崎さんはVIPとかで大学から差し向けられた高級車に乗り，他の人はマイクロバスで行く．

いつものように，VAXとその端末，スタンドアロンのWS，同じくスタンドアロンのPC群を見る．デモは，これもおなじみの織物用CAD．1学年のクラスのサイズは200人だそうである．RS232C系だけがネットワークのようで淋しい．計算中心(センター)には富士通のM340があり，富士通関係者もいて感心する．

昼食は，済南の街中で，来年のシンポジウム開催予定地の昆明スタイル・ラーメンを食べる．夕方の飛行機で上海着．シンポジウム実行委員長の居先生のグループが待っておられて，新錦江ホテルにチェックインした後，旧錦江ホテルで晩餐．華東理工大学の若い先生と席が隣りになったので，研究内容等について話をした．

ホテルに戻ると，例によって夜の街へくりだす相談が始まったが，私は坂本さんと一緒にもう1日滞在することになっているので，パス．熊谷さんの自宅(ホテルの1室)でマッカラン18年を飲みながら読書．みなさんは，24時間営業の美食街へ行き，去年見つけておいた四川料理屋で，羊肉火鍋を食べたそうだ．半分寝ながら本を読んでいたら，1時頃みなさん元気に御帰館．それから少しおしゃべりして，寝たのは2時ぐらいだったような気がする．

10月20日(水)

4階の中華レストランで朝食のつもりが，そこはお昼からとかで，3階で飲茶．坂本さんはオムロン上海へ仕事に出かける．他の人たちは午後帰国するので最後のショッピング．私は，何本か日本に電話を入れたあと部屋でのんびり．

よく考えると，今回のツアーは，夜寝る以外のんびりしたのはこの時が初めてだけだったような気がする．10日もあると最低1日は休養が欲しい．本来休

養日だったはずの曲阜や泰山が結構ヘビーだったので，今回は疲れを感じたのだろう．

11時半にフロントで全員集合して空港へ．みなさんを見送ったあと，私と熊さんは，そのまま総経理専用車ボルボで別のホテルへ行き昼食．熊さんが運転手に迎えにくる時間の指定を1時間まちがえたので，その間に展覧中心へお土産を買いに行く．

展覧中心では，人民元が使えるようになったので路上の両替屋さんをが探したが，こんな時に限って見つからず，買い物が終って出た所で見つかる．夜の食事代のこともあるので，2万円を人民元に替える．日々レートが違うようで，お互いに電卓を出しあって交渉．

そのあとPFU上海のオフィスを訪問．教員養成学院の一部を借りているそうで，入口の構えや建物のたたずまい，途中の廊下の雰囲気は，ソフトハウスらしからぬ感じがしたが，オフィスの内装はきわめていい感じになっている．社員教育の最中とかで，途中割り込んで15分程スピーチ．「私は総経理の友人の一人です．といってもお酒の友達です」と自己紹介して，受ける．

夕食は坂本さんと合流して，オムロン上海のオフィスがあるビルの1階のレストランで食べる．オムロン上海社長の包叔平さんが，奥さんに市場で上海カニを買って来させ，それを持込みで料理してもらって，1人あたり雄雌1匹づついただく．美味．

疲労のせいか，腹具合があまりよくないので，熊さんちで，お酒ではなく，日本茶と梅ぼしで午前2時ぐらいまでおしゃべり．部屋に戻ってお土産を整理し，荷物のパッキング．岩波文庫の春秋左氏伝(上)をやっと読み終って寝る．

10月21日(木)

朝，7時半に熊さんとホテルの43階の回転食堂で洋式バイキング．しかし，おかゆはやっぱりかかせない．華東理工大学の若い先生がロビーに迎えに来られ，私1人で大学訪問．この大学の計算機科学科の教授が，2国間交流で，ちょうど入れ違いに訪日され，いま私の研究室に滞在しておられるので，その意味では交換訪問のようなものである．CS関係の授業や計算センターを見せてもらったあと，SUNの上で動くCASEツールなどをデモしてもらった．途中でマウスが動かなくなりデモは中断．泰山を汚したタタリか!?

この大学は以前，華東化工学院という名前だったので，半分以上が化学関連の学科らしいが，アメリカのメリーランド大学あたりとも交流があり，CASEツールに関して，かなりまともに研究している印象を受けた．中国では，先進諸国における企業の研究所や開発部門の役割を大学の研究室がやっていると聞いていたし，私もそう理解している．

図書館を見学したあと，ゲスト用の会館で昼食を取り，記念撮影をしてから，空港まで送って貰う．そのために，帰国までネクタイ着用になってしまった．空港では，手持ちのお金(150元ぐらいしか残っていなかった)で，お土産を買い足してJALに乗る．

大阪空港では，坂本家のみなさんが御主人を出迎えておられたので挨拶する．家に帰ってさっそくログインすると496通のメール．2時間ほどかけてざっと流し読みしているあいだに，途中でまた10通ぐらいとどく．急ぎの用は1通だけだったので，その返事だけを書き，翌日の第1限の演習もあるので，1時には寝る．明日からは，またいつもの生活が始まる．

ソフトウェア信頼性の評価･計測に対する認識は十分か?

山田 茂
(鳥取大学)

労働集約型の手工業的ソフトウェア生産環境に対し，人材の不足とあり方が問われて，ソフトウェア･クライシスという課題が提起されて以来，どのように実質的な問題解決がなされてきたのだろうか?

この対策としては，人材の育成と効果的な生産技術の開発により，科学的管理法の下でソフトウェアの品質と生産性の向上を図り，さらにソフトウェアの生産自動化を実現して行けばよいことは，明らかである．また，品質が生産性向上に寄与する最大の原因であることもわかっている．

したがって，経営者の理解の下で，ソフトウェアの品質管理 (SWQC) 部門を設置して，生産プロセスの全工程でバグが潜入しないようにし，また潜入したバグは早期検出するように組織的に管理して行く総合的品質管理 (TQC) を推進して行かなければならない．

ソフトウェア品質の可視化のためには，定量的尺度とその計測法を用いる必要がある．生産組織内の関係者が，ソフトウェア品質の実現プロセスと作業結果を数値的に把握するには，「当たり前品質」としての品質特性である信頼性を計測･評価することがもっとも重要である．このために，バグの発生･検出過程を抽象化したさまざまなソフトウェア信頼性モデルが研究され，適用技術を含めてその妥当性が精力的に議論されている．

このモデルは，バグの発生･検出現象が不確定事象であるため，確率･統計論により取り扱われることが多い．このため，SWQC 担当の管理者あるいは上級 SE でさえも，数学的信頼性アレルギーを持ち，「モデルの仮定が非現実的だ」，「あれはわれわれの環境には合わない別世界のもの」などと有効性を無視して，逆にアラ探しまでやってしまう．

もちろん，完全なモデルなど存在しないのであるから，モデル研究者の知識を大いに利用して当該環境に合うように適宜改良･改善し，従来のソフトウェア品質文化にもとづいて独自のソフトウェア信頼性文化を創出し，温故知新を図ればよいのである．

バブル経済が崩壊したあと，景気後退とも重なって，ユーザの情報化投資の抑制･選別が進み，情報産業はマイナス成長を余儀なくされている．企業体質の強化と情報技術の世代交代をはじめとする構造変化が進むなか，いまのうちに原価低減に直結するソフトウェア信頼性技術の実践に取り組む必要があるのではないだろうか?

(電気通信普及財団機関誌「TAF クォータリー」Vol.33, 1994 年 1 月)

コンピュータ･システム･デザインと文化

中小路 久美代
(SRA, Boulder Lab. & University of Colorado at Boulder)

1. システム･デザインと文化

本稿は，American Programmer 誌の October 1993 号に掲載された記事をもとに，日本の方々に向けて直したものです．

アメリカでは，昨今，コンピュータ･システム･デザインと文化の関わりについて，大きな関心がそそがれています．まず１つは，開発プロジェクトの要員が異なる国々にまたがる場合，もう１つは，異なる国(特に日本やアジア諸国)に向けてシステムを商品化する場合です．

日本でも，まだまだその割合はアメリカに比べれば低いものの，近い将来，異文化にまたがるシステム･デザインが必要になることは明らかですし，現時点においても，社風や役職による”文化”の違いから来るいろいろな問題に直面されている開発者の方々も少なくないはずです．

本稿では，主として，Human-Computer Interaction まわりのデザインにおける文化的配慮について言及しますが，システム設計や商品化に向けても深く関わってくる問題です．

開発プロジェクトが異なる文化的背景を持った人たちで構成される場合，単一文化の人間だけで構成されたプロジェクトでは予期しえないような，いろいろな問題が表面化します．問題は，単に言葉の壁にとどまらず，社会的背景に対する”共通の理解”がないことも，相互のコミュニケーションの妨さまたげとなります．

異なった文化に向けてシステムを開発する場合には，単にその文化で用いられている言語のインタフェイスをかぶせればよい，といったものではないことは，みなさんも経験済みではないかと思います．たとえば，ほとんどのアメリカ製のワープロ･ソフトは，タイプライタ文化の発達した西洋人向けにデザインされたものであり，タブやマージンなどといった概念は，一般の日本人には聞き慣れない表現でした．

本稿では，異なった文化に向けてシステムをデザインする場合の問題について焦点をあてますが，これらの問題は，異なった文化圏で開発されたシステムを導入する場合に生じる問題点だともいえます．

たとえば，アメリカ製の CSCW(Computer Supported Cooperative Work) ツールをそのまま日本の根回し社会に導入しても，期待したような効果は得られないでしょう．意思決定の過程を緻密に記録するメカニズムを持っていたところで，肝心の意思決定の過程は「酒の席」で話し合われるわけですから，そうした会議支援ツールは役に立たないのです．

もう１つの例をあげましょう．

ある会議用ツールがヨーロッパで開発されました．そのツールを用いると，会議中に匿名で意見を述べることが可能になり，かなりの成果(より活発な意見の交換と生産的な合意)が得られたと報告されました．

ところが，そのツールがアメリカで実験的に使用されたところ，ユーザからはよく受け入れられませんでした．匿名になることで，人々は会議にたいして非常に消極的になり，会議が進行しなくなってしまったのです．自己主張がすべてというアメリカ文化においては，意見を述べることで会議に対する貢献度が評価され，したがって，匿名では意見を述べる意味がなくなってしまったからです．

このように，文化的背景は，システム導入の成功･不成功を大きく左右します．ところが，文化といった大局的な背景や状況は，概して見落とされがちで，システムの要求仕様記述に言及されることはまれですし，また言及されたところで，それを具体的にどう表現し，対処すべきかといった分野は，まだあまり研究されていません．

アメリカには，コンピュータ･システムと人的要因との関わりを研究する CHI (Human Factors in Computer Systems) という学会があります．その学会で，1992 ～ 93 と 2 年続けて，"Cross-Cultural Considerations in Human Computer Interaction" というテーマで，ワークショップが催されました．1992 年のテーマは，

「異文化の人々と共同作業をするとはどういうことか?」で，参加者はいろいろなゲームを通じて，それを身を持って体験しました．1993年のテーマは，「異文化に向けてシステム開発をするとはどういうことか?」で，文化を理解するための手法などが話し合われました．

本稿では，この1993年のワークショップで話し合われた内容を，簡単に紹介します．1992年の内容は，Kellogg and Thomas [1993] (SIGCHI Bulletin Vol.25, No.2, pp.40-45, 1993) に紹介されています．

2. 文化的要因とアーキテクチャ

従来の Human-Computer Interaction デザインは，(1) タスク要因，および (2) 認知要因，という2つの視点から考えられてきました．

タスク要因は，そのシステムを用いてユーザが行うタスクを考慮します．認知要因は，人間の基本的な性質を考慮する視点で，たとえば，1秒間に何回キーをたたくことができるかとか，画面上での色の見分け方，といった視点からシステムを解析します．

異文化に向けてシステム設計を行うさいには，これに加えて，第3の要因を考える必要があります．すなわち，言語，社会的要素，規範，価値感といった文化の要因です．

コンピュータシステムの使われ方は，国によってかなり異なることが認識されています．たとえば，

- 日本では，trial-and-error の手法は，時間の無駄であり，あまりよい方法ではないが，アメリカでは，システム理解を助ける方法として，とても有益だととらえられている．
- システム購入の際に，日本ではアフターケアの有無が大きな影響を与えるが，アメリカでは値段が最も重要な要因である．
- ドイツでは，緻密なデータベース·クエリを記述する方法が好まれるのに対し，イタリアやスペインなどのラテン系の人々は，ハイパーテキストなどのブラウザを用いて情報検索する方法を好む．

このように，万国共通に受け入れられるインタフェイスなどというものは存在しないことがわかります．ですから，異文化に向けてシステムを商品化する際には，既存のシステムをその文化に向けて修正するよりも，むしろ，システムそのものを，最初から文化の違いを考慮しながら設計することが，成功の秘訣となります．理想的には，開発者がその文化を理解し，システム設計の段階から，文化的要因を要求仕様として把握することが望ましいのですが，では，いったいどこまでが文化に関わったインタフェイスで，どこまでからが文化に関わりなく使いやすいインタフェイスであると認識することができるのでしょうか？

前述のワークショップでは，これらの課題に答えるべく，KJ法を用いて活発な意見が交換され，最終的に文化を考慮したシステム設計のアーキテクチャが提案されました．このアーキテクチャは，(1) 中心レイヤ，(2) 文化と独立したレイヤ，そして (3) 文化に依存したレイヤ，という3つのレイヤ (階層) から構成されています．

(1) 中心レイヤは，人間としてユーザの認知的視点を支援します．このレイヤに属するデータ表現や知識表現は，言語に関わりなく，人間の知覚システムに関するものです．

(2) 文化から独立したレイヤは，中心レイヤから一歩進んで知覚よりもより抽象度の高い表現形態を用いますが，文化には依らない機能から構成されるレイヤです．具体的な技術としては，

(2-1) 標準化された表現の利用．たとえば道路標識は，人間とのインタフェイスにうまく万国共通の標準を用いた成功例といえます．赤地に白の水平線が進入禁止を表わすという暗黙の了解が，多くの文化ででき上がっています．たとえ，初めてその文化に接した人々でも，その機能を推測することができます．

(2-2) メタファを用いることで，言語に頼りがちなインタフェイス·デザインを改善することができます．昨今のマルチメディア技術によって，なお一層，メタファを用いやすくなりました．一方で，メタファに頼り切ってしまうことで生じる問題もあります．たとえば，ごみ箱のアイコンがよい例です．国によっては，いったんゴミ箱に捨てたものは，二度と取り戻せませんが，他の国では，いったん捨てても，また取り戻すことが可能です．したがって，人々はその国々によって，そのアイコンにたいして，異なる機能を連想してしまうわけです．

(2-3) ユーザの学習性も，忘れてはいけない要因です．

たとえ標準の存在しないドメインでも，一貫した表現法を用い続けることで，それを標準化することができます．たとえば，初めは意味のわからないアイコンでも，次第にその機能がユーザに覚えられて，理解されることになります．

(2-4) このレイヤをデザインするにあたって大切なことは，常にユーザがシステム･デザイナの予期した通りにシステムを理解しているかどうかです．そこで，システムの振る舞いのフィードバックをユーザに与えることが，異文化に向けてシステムを設計する際には，特に大切になります．

(3) 文化に依存したレイヤを設計する仕事は，その文化の特徴を理解することから始まります．

(3-1) システム設計者は，そのシステムのユーザの文化の特徴をまず理解しなければなりません．そのためには，ユーザと直接関わりあい，アンケートや調査用紙を用いて要求仕様を認識することはもとより，その文化ですでに成功を収めているシステムを探し出し，それを解析することも，有意義な情報源となります．

(3-2) ユーザの側から各々の文化をシステムに明示するためのメカニズム，たとえばアンケートやテンプレートなども，必要です．その情報を用いて，システムが自動的にシステムの振る舞いをその文化に併せることができます．たとえば，ユーザが，スペイン語文化圏であると明記すれば，システムは，メニュー表示やエラーメッセージなどをスペイン語で表示するのです．システムの側からは，ユーザの振る舞いをモニタして，どのようなコマンドを多用するかなどの情報から，ユーザ･モデルを作り出し，それを用いて，その振る舞いを変更することも可能です．

(3-3) それぞれの文化に向けて設計されたシステムは，それぞれの文化で，テストすなわち評価されなければなりません．異文化に向けて設計されたシステムでは，通常の Human-Computer Interaction デザイン以上に，予期せぬ問題が発生する可能性が大です．たとえば，輸出先のアラブの国で，ラディアル･タイアの溝のパターンがアラーと読めてしまったためにタイアをすべて回収した (!) という事例は，まだ記憶に新しいところです．

(3-4) ヘルプ･メッセージやオンライン･マニュアルは，個々の文化に併せて記述し直すべきです．言語によって異なる文の言い回しがあったり，論理の展開の仕方に違いがあったりするので，単に翻訳するのでは，意味のとおらないことやわかりにくいことが多々あります．たとえば日本語では，キーとなる文章は段落の最後に位置することが多いのですが，英語ではほとんどの場合，段落の最初に位置しています．したがって，斜め読みをする際に，どの文章を拾い読みしてゆくかに違いが出てくるわけです．また，用語なども，言語や文化によって，同じ言葉をまったく異なる意味で用いている場合なども少なくなく，翻訳の際には注意が必要です．

3. 文化を表わすモデル

最後に，文化をシステマティックに特徴づける方法として，以下の要因からなる Trompenaar's Model を紹介します．文化を表わすモデルは，社会･人類学や経済学などの分野で研究されていますが，このモデルは，その中でも，英国のEast London Business School の David Gee 氏から，前述のワークショップにおいて紹介されたモデルです．これらの要因が文化を表わすために十分なモデルだというわけではありませんが，異なる文化を理解する上で注意すべき点として，興味深いものです．

＊時間の知覚/認識

文化によって，時間とか時の流れという概念の捉え方に違いがあり，それが業務を行う上で違いとなって現われます．たとえば，事を同時に並列してこなしてゆくことを好む文化もあれば，ひとつずつ，逐次処理してゆくことを好む文化もあります．また，人々の辛抱強さも国々によって異なり，それがどれだけ敏速なサービスを重視するかという点に反映されます．敏速さが何よりのサービスという国もあれば，愛想の良さなど他の要因の方がずっと重要という国もあります．

＊個人 versus 全体

文化によって，個人の能力を最重要視するか，それとも全体としてのパフォーマンスの方を重視するかに違いが現われ，これは CSCW などのツールを設計する上で重要なキーとなります．特に，この個人対全体の違いは，国によっての違いはもとより，ワークステーションとメインフレームのユーザの違いなどにも，しばしば見られるものです．

＊手続き的　versus　宣言的

西洋文化圏では，一般に，宣言的もしくは断言的に記述された説明が好まれるのに対し，東洋やロシア文化圏では，手続き的に記述された説名文が好まれるといわれています．これらの違いは，マニュアルやヘルプメッセージの書き方，また商品化にさいしての，CM のデザインなどにも反映されるべきです．

＊普遍的　versus　特殊化

普遍性や単一性を非常に重視し，あらゆる機能に対して同一のインタフェイスを要求する文化もあれば，特殊化することに意義を見い出し，個々の機能がそれぞれに見合ったインタフェイスを持つべきであるとする文化もあります．

＊内的制御　versus　外的制御

あくまで病源を絶とうとする西洋医学の方法にも見られるように，えてして西洋文化圏では，外界の環境を制御することを好みます．それに対して東洋では，自然と和合して治療させることを目指す東洋医学のアプローチに見られるように，外界との調和を重視します．このような違いは，コンピュータ・システムとのインタラクション設計において，能動的にシステムの制御を行えるようにするか，それとも受動的にシステムと対話できるようにするか，という違いに現われてきます．

＊会話における間の取り方

言語によって，人々の会話の際の間隔の取り方にかなりの差があることが報告されています．英語や独語においては，間をおかずに話し手が交替します．ところが，アジア語圏では，会話のあいだに少しの間が空きます．一方，ラテン語においては，一人が話し終わらないうちに次の人が話し始めるというように，会話が重なることがしばしばあります．これらの違いを考慮することは，システムとユーザのインタラクションを会話としてデザインする場合に，重要なポイントとなります．

4. おわりに

この文章を読まれて，「だからどうした?」と思われた方，あるいは，「これはかなりの独断と偏見ではないか?」と感じられた方も，少なくないと思います．この原稿は，冒頭に記したように，もともとアメリカ人に向けて，システム・デザインにおいて文化の違いをどうとらえるかを述べたものを，ほとんどそのまま翻訳したものです．

ある意味でとても一方的な内容にも関わらず，それが American Programmer 誌に掲載されると，何人かの読者の方々から反響をいただきました．文化の違いなどというとてつもない難題を，方程式か何かのように魔法のごとく解決できると考えている人たちがいることが，とてもアメリカ的だと感じました．また，人種のるつぼと言われているこのアメリカで，世界には自分たちと異なるものの考え方をする人間が存在するという事実に今さらながら気づいたという人が少なくないというのも，われわれ日本人にとってはある種のカルチャーショックかもしれません．

しかし，ソフトウェアエンジニアとして振りかえってみて，私たちも日ごろ，知らず知らずのうちに文化の壁と格闘しているのではないでしょうか．たとえば，どうしてユーザがあんなことをできないのかわからないとか，新しいプロジェクトに配属されて勝手の違いにわけがわからないとか，感じられたことはありませんか．大切なことは，そのような "breakdown(故障，行き詰まり)" が生じたときに，自分自身や他人を責めるのではなく，まず文化の違いを認識して，次に，その違いをなくそうとするのではなく，それに対して建設的に対処して行くことだと思います．

「文化とはある特性を持った人々の集まりの個性である」といった人がいます．ユーザの文化，マネジャーの文化，メインフレームの文化，COBOL プログラマの文化，Unix の文化，私たちの周りには，ありとあらゆる単位の文化が存在します．文化は，人びとの築き上げた知識と経験のかたまりです．コンピュータ・システムは，これらの大切な文化を活かしながら，その上に新たな文化を作り出すものです．システム設計者・開発者としては，その責任の重要性を改めて認識すべき時ではないでしょうか．

Preliminary Call For Papers
The First Asia-Pacific Software Engineering Conference (APSEC)
December 7 - 9, 1994, Tokyo, Japan

SPONSOR
The Special Interest Group on Software Engineering
of the Information Processing Society of Japan (IPSJ/SIGSE)

The first Asia-Pacific Software Engineering Conference (APSEC) is organized to bring together researchers and practitioners in software engineering from the international community, mainly from Asia-Pacific countries.

In the software engineering arena, the North American countries and the European countries have their own well-established international conferences which have been providing good occasions for researchers and practitioners to exchange their ideas. However, these conferences have been usually held far away from the Asia-Pacific countries. This geographic condition is one of the major reasons to limit the number of papers accepted and the number of people attending these conferences. This inevitably led to an accumulation of cries such as "Why we don't have our own conference in the Asia-Pacific region!" which has finally encouraged us to establish a new conference.

APSEC has originally grown out of the joint activities of the two SIGSE's, namely the Special Interest Group of the Information Processing Society of Japan (IPSJ/SIGSE) and the Special Interest Group of the Korea Information and Science Society (KISS/SIGSE), in organizing the Joint Conference on Software Engineering (JCSE) twice. The first one was successfully held in Seoul, Korea in March 1992 and the second one was successfully held in Fukuoka, Japan in November 1993, which have received twice as many papers as the first conference from several Asian countries. By seeing this. we understood that there are growing demands in the Asia-Pacific countries to have our own conference on software engineering. This led us to decide to terminate the JCSE after the Fukuoka conference and then to start a new conference, i.e., APSEC. We believe that APSEC will give the researchers and practitioners in that area a greater opportunity to express and exchange their ideas among people in neighboring countries, and thus to expedite leveling up of their research and development.

THEMES

The conference will address the following principal themes:

Requirements Engineering, Specification & Design, Testing, Maintenance, CASE tools, Software Metrics, Software Process, Reuse, Reverse Engineering, Object Orientation, CSCW, Distributed Development, Information System Development, Project Management, Quality Assurance, Education, and the other topics relevant to Software Engineering Field

INSTRUCTIONS TO AUTHORS

APSEC Program Committee solicits original technical papers and proposals for panel discussions. All contributions will be reviewed and evaluated based on originality, technical quality, and relevance to Software Engineering.

- *TECHNICAL PAPERS* should be no longer than 6000 words. All papers should include a separate cover sheet which provides the following information : the title, authors' names, postal and electronic mail addresses, telephone and fax numbers, a 200 words abstract, a list of keywords. Experience papers or practical papers are also welcome to be submitted. Submitted papers should be written in English and clarify what is new and significant about the presented work. Accepted papers will be published in the conference proceedings.

- *PANEL PROPOSALS* should describe the title and a brief description (including an aim and a goal). The panel chair and panelists can be included in the proposal if the author has a plan. They should be no longer than two pages.

Six (6) copies of TECHNICAL PAPERS and PANEL PROPOSALS should be sent by **April 30, 1994** to either of program co-chairs, not both. :

From Asia, Africa, and Europe
Motoshi Saeki
Dept. of Electrical & Electronic Eng.
Tokyo Institute of Technology
Ookayama 2-12-1, Meguro-ku
Tokyo 152, Japan
TEL. +81-3-3726-1111 (Ext.2192)
FAX +81-3-3729-1399
E-mail saeki@cs.titech.ac.jp

From Australia, Oceania and America
Roger Duke
Dept. of Computer Science
University of Queensland
Brisbane, 4072, Australia
TEL. +61-7-365-2097
FAX +61-7-365-1999
E-mail rduke@cs.uq.oz.au

IMPORTANT DATES

Submission Deadline :	30 April 1994
Acceptance Notification :	15 August 1994
Camera-ready Copy Due :	30 September 1994

GENERAL CONFERENCE COMMITTEE

Conference Chair	Sadahiro Isoda (NTT)
Program Co-Chairs	Motoshi Saeki (Tokyo Institute of Technology)
	Roger Duke (University of Queensland)
Local Arrangements	Yoshiaki Fukazawa (Waseda University)
Treasurer	Mikio Aoyama (Fujitsu)
Publicity	Kazuhito Ohmaki (Electrotechnical Laboratory)

STEERING COMMITTEE MEMBERS

Ken-ichi Harada (Keio University, Japan)
Sadahiro Isoda (NTT, Japan)
Yong Rae Kwon (KAIST, Korea)
Lin-shan Lee (Academia Sinica, Taiwan)
Danny Poo (National University of Singapore, Singapore)
Karl Reed (La Trobe University, Australia)
Vincent Shen (The Hong Kong University of Science & Technology, Hong Kong)
Chi Su Wu (Seoul National University, Korea)

PROGRAM COMMITTEE MEMBERS (Tentative)

Tsuneo Ajisaka (Kyoto Univ., Japan)
Paul Bailes (Univ. of Queensland, Australia)
Donghae Chi (ETRI, Korea)
Chyan-Goei Chung (National Chiao Tung Univ., Taiwan)
Masahito Hirakawa (Hiroshima Univ., Japan)
Shinichi Honiden (Toshiba, Japan)
Hisayuki Horai (Fujitsu, Japan)
Stan Jarzabek (National Univ. of Singapore)
Kyo-Chul Kang (Pohang Inst. of Sci. & Tech., Korea)
Jyh-Sheng Ke (Inst. for Information Industry, Taiwan)
Moon Hae Kim (Kon-Kuk Univ., Korea)
Yue-Sun Kuo (Academia Sinica, Taiwan)
Kazushi Kuse (IBM Japan)
Don Lawrynuik (National Univ. of Singapore)
Hing-Yan Lee (Info. Tech. Inst., Singapore)
Keung Hae Lee (Hankuk Aviation Univ., Korea)
John Leaney (Univ. of Tech., Sydney, Australia)
Chris Marlin (Flinders Univ., Australia)
Morio Nagata (Keio Univ., Japan)
Kazuhito Ohmaki (ETL, Japan)
Young Chul Shim (Hongik Univ., Korea)
Katsuyasu Toyama (NTT, Japan)
T.H. Tse (Univ. of Hong Kong)
Seung-Min Yang (Soongsil Univ., Korea)
K.T. Yung (Hong Kong Productivity Council)

CALL FOR PAPERS

ICSM '94

International Conference on Software Maintenance and Tools Fair
Victoria, British Columbia, Canada
September 19 - 23, 1994

Sponsored by IEEE Computer Society's Technical Committee on Software Engineering

General Chair:
Lee J. White
Dept. of Comp. Eng. & Science
CASE Western Reserve Univ.
511 Crawford Hall
Cleveland, Ohio 44106 USA
Voice: 216-368-2802
Fax: 216-368-2801
E-mail: leew@alpha.ces.cwru.edu

Program Co-Chairs
Hausi A. Müller
Dept. of Computer Science
University of Victoria
P.O. Box 3055
Victoria, B.C. V8W 3P6 Canada
Voice: 604-721-7630
Fax: 604-721-7292
E-mail: hausi@csr.uvic.ca

Mari Georges
Cap Gemini Innovation
86/90, rue Thiers
92513 - Boulogne-Billancourt
France
Voice: 33-1-49-10-53-98
Fax: 33-1-49-10-06-15
E-mail: mari@capsogeti.fr

Tutorials Chair
Shawn Bohner
The MITRE Corporation
611 West Poplar Road
Sterling, VA 20164-4733 USA
Voice: 703-883-7354
Fax: 703-883-6991
E-mail: bohner@mitre.org

Tools Fair Chair
Daniel Hoffman
Dept. of Computer Science
University of Victoria
P.O. Box 3055
Victoria, B.C. V8W 3P6 Canada
Voice: 604-721-7222
Fax: 604-721-7292
E-mail: dhoffman@csr.uvic.ca

Program Committee
R. Arnold (USA) K. Bennett (UK)
S. Bohner (USA) G. Caldiera (USA)
F. Calliss (USA) M. Capretz (J)
D. Card (USA) E. Chikofsky (USA)
R. Ciampoli (I) A.. Cimitile (I)
F. Coallier (CAN) J. Cross II (USA)
R. Donnelly (USA) P. Feiler (USA)
J. Foster (UK) M. Gentleman (CAN)
M.J. Harrold (USA) D. Hoffman (CAN)
R. Keller (CAN) M. Kellner (USA)
T. Khoshgoftaar (USA) P. Kuvaja (FIN)
S.-C. Lim (SIN) N. Madhavji (CAN)
L. Mancini (I) E. Merlo (CAN)
J. Munson (USA) J. Mylopoulos (CAN)
J. Ning (USA) P. Oman (USA)
M. Orgun (AU) V. Rajlich (USA)
D. Robson (UK) N. Schneidewind (USA)
J. Slonim (CAN) P. Sorenson (CAN)
W. Strigel (CAN) A. Symonds (USA)
T. Tamai (J) L. White (USA)
H. Zuse (D) N. Zvegintzov (USA)

4th Reengineering Forum
For more information about the 4th Reengineering Forum please contact: Elliot Chikofsky (e.chikofsky@compmail.com).

The 1994 Reengineering and Maintenance Week in Victoria, British Columbia, Canada will include ICSM '94 (formerly called CSM) and the 4th Reengineering Forum (formerly called Reverse Engineering Forum).

The Software Maintenance Conference is the world's premier forum for the presentation and discussion of state-of-the-art developments in the field of software maintenance. The theme of ICSM '94 is "**Build for Software Evolution.**" Software maintenance includes enhancing, adapting, and correcting software. Software evolution includes capturing, preserving, and extending knowledge about software systems evolving over long periods of time. ICSM '94 will provide a forum for discussing software maintenance and evolution through refereed papers, experience reports, panel sessions, exhibits, and informal meetings. ICSM '94 will present the most important practical, experimental, and theoretical work currently conducted to support software maintenance and evolution. The participants will include practitioners and researchers from industry, academia, and government.

Major Topics include, but are not limited to: designing for ease of maintenance and evolution; living with legacy or heritage systems; software migration and conversion; achieving and maintaining fitness for use; reverse engineering and design recovery; design decision capture and exploitation; annotating and documenting software systems; program understanding and comprehension; recognizing patterns in existing software systems, reengineering and restructuring; business reengineering; standards and methodologies; process and lifecycle models; empirical maintenance studies; maintenance and maintainability metrics; formal methods applied to maintenance and evolution. change and impact analysis; knowledge-based system maintenance; verification and testing methods for maintenance; software maintenance and evolution tools and architectures: software maintenance education.

Submissions. Submit 5 copies of papers, experience reports, or panel proposals in English to one of the Program Co-Chairs by **March 25, 1994**. Notification of acceptance or rejection of all submissions will be made by **June 10, 1994**. The final papers will be due **July 29, 1994**. Submissions will be refereed by an international program committee for their applicability and relevance to the software maintenance community. For **Papers** submit 2000-5000 words, which have not been previously published and are not under consideration for publication elsewhere. Please include on the first page the title, all authors' names, complete contact information of the lead author (address, phone/fax numbers, e-mail), a short abstract of no more than 250 words, a list of descriptive keywords, and a specification of the type of submission (Research/ Experience). **Research Papers** will be evaluated for originality, significance, soundness, and clarity. Methodological papers should show how the results presented contribute to maintenance practice. Papers on systems should concentrate on technical and architectural issues. Experimental papers should describe the experimental methods used and interpret the results in terms of practice. **Experience Reports** might include experiences with software maintenance technologies or processes (e.g., using a tool or a method, applying a metric, following a project management or software process discipline). Emphasize outcomes, insights gained, and lessons learned. **Panel proposals** should include a title, the proposed chair, the tentative panelists, a brief description of the panel session subject, and a supporting rationale. Panelists should have agreed to participate before the submission of the proposal. ICSM **Tutorials** should address maintenance topics that are of particular interest to practitioners. Proposals for full-day or half-day tutorials should include a detailed outline of the material, a description of past experience with the tutorial, an assessment of the materials' maturity, and credentials of the instructor. Submit 5 copies to the Tutorials Chair.

Exhibits. A tools fair proposal should include a 1-2 page description of the tool or environment, its applicability to software maintenance, an outline of the proposed demonstration, the amount of exhibit space required, hardware platform required, and any other requirements. Clearly identify the submission as "Research Prototype" or "Vendor Tool." **Submit 3 copies by May 31, 1994** to the Tools Fair Chair.

Conference Location. The 1994 Reengineering and Maintenance Week will be held downtown in the Victoria Conference Centre. Victoria is one of the top leisure destinations of the world. Mountains and sky, islands and sea, hospitality and high technology–all come together in the city of flowers and gardens. The mild climate in Victoria is due to warm Pacific currents which give British Columbia's capital a definitely Mediterranean climate. Conference delegates and their families have the opportunity to experience a holiday they will remember for years to come. Some of the world-famous attractions in Victoria and on Vancouver Island include Butchart Gardens,Craigdarroch Castle, City tours on Double Decker buses, University of Victoria, High Tea at the Empress Hotel, Royal BC Museum, salmon BBQ, whale watching tours, San Juan and Gulf Islands, and Long Beach National Park.

Transportation. Victoria's prime location, on the southernmost tip of Vancouver Island, midway between Europe and Asia, provides easy access from almost anywhere in the world. Victoria's International Airport is connected to the Vancouver or Seattle-Tacoma International airports by over 40 flights daily offered by several airlines. The 20 minute flight across Puget Sound or the Gulf Islands is absolutely spectacular. There are direct flights offered by a variety of carriers from many cities in the US, Europe, Asia, Australia, and Canada to Vancouver and Seattle. Victoria can also be reached from Vancouver and Seattle by float planes, ferries, and catamarans.

4th Reengineering Forum

19-21 September 1994
Victoria, BC, Canada

Call for Presentations:
"Reengineering in Practice"

The Rengineering Forum is a combined industry/research review of the state of the art and the state of the practice in reengineering of business processes, software and systems and in reverse engineering of the same. It is a meeting place for key people in the reengineering field: developers, researchers, and leading-edge users.

Presentations are invited on all aspects of reengineering in practice:

- **business process reengineering**
- **IT transformation to better support the enterprise**
- **systems reengineering**
- **software reengineering**
- **reverse engineering**
- **program understanding**
- **related topics...**

Our objective is to form a broad perspective snapshot of the present state of the field. Technical and user experience presentations are sought; sales pitches are discouraged. Blue sky and "what I really need is..." presentations are welcome.

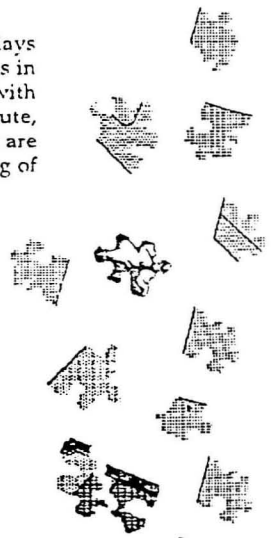
The Forum includes presentations by experienced users, tools developers, and industrial and academic researchers, describing the current state and directions of products, prototypes, and approaches. As a principal technical meeting of the reengineering field, a major focus of the Forum is to allow ample opportunity for one-on-one and group discussion among presenters and attendees. Interaction is paramount. The Forum is held as much to be a meeting place for the speakers to compare notes on the future of the field as it is for attendees to learn about the state-of-the-art and new developments. It incorporates and expands upon the Reverse Engineering Forum, established in 1990. The Forum has been successful as a key resource for software professionals who are looking for, or developing, solutions for coping with existing system assets.

This year, the Forum joins with the 11th International Conf. on Software Maintenance (ICSM'94) of the IEEE Computer Society to form the 1994 Reengineering and Maintenance Week. This brings together industry and research communities with similar interests throughout the week of September 19-23, including exhibits, tutorials, joint sessions, working meetings, and workshops.

Presentation and panel session proposals are due by April 30, 1994. Submit a 1-2 page description to:
Reengineering Forum, P.O. Box 400, Burlington, MA 01803 USA; email reengineer@computer.org; fax 617-272-8464.

The Forum also welcomes proposals for Discussion Round Table sessions. These are extended, focused working discussions on technical topics. Round Table topics have included: reengineering economics models; development of a DoD reengineering strategy; revisiting the IEEE Software taxonomy of reverse engineering and reengineering; the role of standards in maintenance and reengineering.

The Forum uses a "rolling admissions" policy for accepting presentations. Some speaker slots are kept open until days before the meeting to enable previously undiscovered work and new results/developments to be presented. This results in a high-quality, timely, and very current program for Forum attendees. Speaker foils are published as the proceedings with as late a deadline as logistics will allow. Specific dates/time for all speakers are assigned at the last possible minute, maximizing speaker participation in the entire program and retaining flexibility for the best possible program. Speakers are encouraged to be full participants in the Forum, and not just dropping in to give a presentation. In general, as a meeting of the field, speakers will pay an attendee fee.



General Chair: Elliot Chikofsky, Northeastern University, Industrial Engineering & Information Systems,
75 Lexington Street, Burlington, MA 01803 USA
617-272-0049; fax 617-272-8464; email e.chikofsky@computer.org

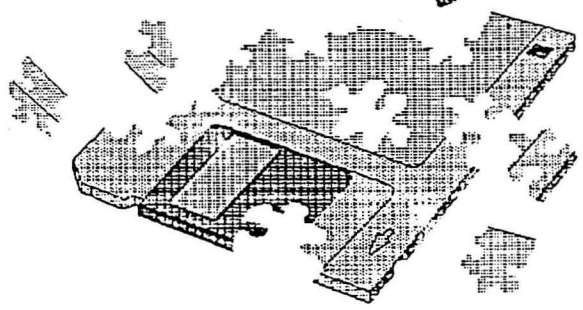
Exhibits: Judith Marx Golub, Software Management News,
B-10 Ste 237, 4546 El Camino Real, Los Gatos, CA 94022 USA
415-969-5522; fax 415-969-5949; email j.golub@computer.org

Tutorials: Shawn Bohner, The MITRE Corporation,
611 West Poplar Road, Sterling, VA 20164-4733 USA
703-883-7354,; fax 703-883-6991; email bohner@mitre.org

ICSM: Lee J. White, Case Western Reserve University, Dept. of Computer Engr & Science,
511 Crawford Hall, Cleveland, OH 44106 USA
216-368-2802; fax 216-368-2801; email leew@alpha.ces.cwru.edu
or IEEE Computer Society, Conferences Offfice, 202-371-1013

4th Reengineering Forum is sponsored by
the non-profit Reverse Engineering Forum Inc.

In cooperation with:
International Workshop on CASE (IWCASE)
Software Engineering Institute (SEI) / Carnegie-Mellon Univ.
Software Management Association
Software Management News



TOOLS USA '94

Technology of Object-Oriented Languages and Systems

FOURTEENTH INTERNATIONAL CONFERENCE AND EXHIBITION

Santa Barbara, California, August 1-5, 1994

CONFERENCE ANNOUNCEMENT AND CALL FOR PAPERS

Program Chair: Raimund Ege, Florida International University • Panel and Workshop Chair: Madhu Singh, Bellcore • Conference Chair: Bertrand Meyer, ISE

TOOLS is the major international conference and exhibition devoted to the applications of object-oriented technology. TOOLS USA '94, the fourteenth TOOLS International Conference and Exhibition, returns to Santa Barbara for the fourth time and continues the commitment to excellence demonstrated by earlier conferences in Europe, the Pacific and the U.S. Standards established for TOOLS by earlier conferences in the series include:

- Emphasis on the practice of object-oriented technology and its application in industrial environments, complementing the more academically-oriented perspective of traditional conferences.
- Solid technical program, based on a strict refereeing process.
- Balanced coverage of the wealth of approaches, trends and variants in the object-oriented community.
- Combination of tutorials by recognized experts, up-to-date exhibitions of products and services, invited talks by innovators in the field, relevant panel discussions, and technical papers targeted to industry practitioners.

As with previous conferences, the TOOLS 14 Proceedings will be published by Prentice-Hall and made available worldwide.

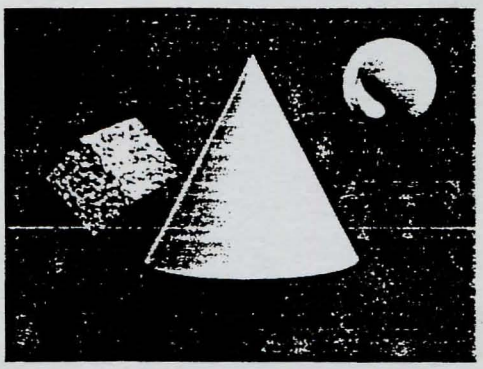
SUBMITTING A PAPER

TOOLS USA '94 is now soliciting papers on all aspects of object-oriented technology. All submitted papers will be refereed and judged not only according to standards of technical quality but also on their usefulness to practitioners and applied researchers. A non-exhaustive list of suggested topics includes:

- Reports of experiences with O-O technology
- Migration toward O-O software development
- O-O user interfaces, persistence and databases
- Class management and evolution
- Empirical and field studies
- Production, distribution and revenue collection of industrial-quality reusable components

TOOLS 14 will feature a special track on issues relating to the migration from conventional to object-oriented programming. This track will include workshop(s), tutorials, panel discussions, and featured speakers. Technical papers are expressly sought in this area.

Papers should be range of 10 to 20 double-spaced pages. Six copies of each submission should be sent to:



TOOLS USA '94
Attn: Dr. Raimund Ege
School of Computer Science
Florida International University
Miami, FL 33199 USA
Phone: (305) 348-3381, Fax: (305) 348-3549
E-mail: ege@scs.fiu.edu

Alternatively, you may contact ege@scs.fiu.edu about our experimental e-mail submission procedure. General information is available by e-mail from tools-info@scs.fiu.edu. "Guidelines for Authors" are available via anonymous ftp from tools.fiu.edu in file pub/tools-authors-guide.

Proposals for panels and workshops are also solicited and should be sent to the Panel and Workshop Chair:

Madhu S. Singh
Bellcore, RRC 1A 111
444 Hoes Lane
Piscataway, NJ 08854
Phone: (908) 699-4366 Fax: (908) 336-2830
E-mail: msingh@bcr.cc.bellcore.com

Tutorial proposals should be sent to the conference organization committee at the address listed below.

IMPORTANT DATES

All submissions must be received by March 18, 1994 to be considered for inclusion in the conference. Submissions should be in English. Notification of acceptance will be mailed by April 30, 1994. Final manuscripts will be due June 1, 1994.

INTERNATIONAL OBJECT-ORIENTED WEEK

Friday, August 5 has been set aside for independently organized events, such as User Group meetings or standardization committees. The TOOLS USA '94 organizers will help coordinate such events if they fall within the scope of object-oriented techniques, and will include the announcements in the final TOOLS program. If you are interested in setting up such a meeting, please contact the conference chair for details at tools@tools, com.

Conference Organization Committee:
TOOLS USA '94
P.O. Box 6863
Santa Barbara, CA 93160
Phone: (805) 685-1006, Fax: (805) 685-6869
E-mail: tools@tools.com

For further information complete the coupon below and e-mail it to the Conference Organization Committee.

❑ I intend to register for TOOLS USA 94. Please send me the conference program.

❑ I intend to submit a paper with the following title:

_____

This will be ❑ a full paper ❑ an extended abstract

❑ Please send me the Guidelines for *"Prospective TOOLS Authors."*

❑ My company is interested in exhibiting. Please send me the *"Call for Exhibitors"* application.

TOOLS USA '94
P.O. Box 6863
Santa Barbara, CA 93160

Phone: (805) 685-1006
Fax: (805) 685-6869
E-mail: tools@tools.com

Name _____

Address _____

City, State, Zip _____

Telephone & Fax _____

E-mail: _____

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