SEAMAIL

Newsletter from Software Engineers Association

Volume 6, Number 6·7 1991

目　次

編集部から		1
プログラミングの自由のための連盟からの2つのアピール		
ソフトウェアの特許に反対しよう		2
ユーザ・インタフェイス著作権に反対しよう		14
OOPSLA'91 に参加して	塩谷 和範	22
私のプロジェクト・マネジメント論	勝田 祐輔	30
教育システム工学の提案	君島 浩	30
ソフトウェア・プロセスをめぐって		
ソフトウェア・プロセスの形式的体系	鯵坂 恒夫	46
PDL とそのシステムを開発して思うこと	井上 克郎	48
ソフトウェア・プロセスの記録と記述	上原 憲二	50
ポジション・ペーパー	臼井 義美	51
現場のソフトウェア・プロセスにもっと関心を	久保 宏志	53
ポジション・ペーパー	田中 浩之	55
新職人時代（居職考）	新保 修	56
江南春紀行	熊谷 章	58
Call for Papers		
5th Symposium on Software Development Environments		70
ソフトウェア・シンポジウム'92		71
Conference on Software Maintenance - 1992		73

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，北は海道から南は沖縄まで，1000余名を越えるメンバーを擁するにいたりました．法人賛助会員も約80社を数えます．支部は，東京以に，関西，横浜，長野，名古屋，九州の各地区で設立されており，その他の地域でも設立準備をしています．分科会は，東京，関西，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，あなのお力を貸してください．

代表幹事： 熊谷章

常任幹事： 市川寛　落水浩一郎　岸田孝一　中野秀男　深瀬弘恭　松原友夫　山崎朝昭

幹事： 飯沢恒　筏井美枝子　上原憲二　臼井義美　奥山充　片山卓也　亀田繁　窪田芳夫　小林俊明
杉田義明　田中一夫　玉井哲雄　鳥居宏次　中谷多哉子　中山照章　野村敏次　野村行憲　浜野善吉
林香　平尾一浩　藤野晃延　二木厚吉　堀江進　本位田真一　盛田政敏　山崎利治　渡邊雄一

会計監事： 辻淳二　吉村成弘

分科会世話人 環境分科会(SIGENV)：田中慎一郎　渡邊雄一
管理分科会(SIGMAN)：野々下幸治
教育分科会(SIGEDU)：杉田義明　中園順三
ネットワーク分科会(SIGNET)：小林俊明
法的保護分科会(SIGSLP)：能登末之
調査分科会(SIGSVRVEY)：岸田孝一　野村敏次

支部世話人 関西支部：臼井義美　中野秀男　盛田政敏
横浜支部：藤野晃延　北條政顕　野中哲　松下和隆
長野支部：市川寛　佐藤千明　細野広水
名古屋支部：筏井美枝子　岩田康　鈴木智　平田淳史
九州支部：平尾一浩　藤本良子

賛助会員会社： ＮＴＴソフトウェア研究所　ＮＴＴ九州ネットワーク技術センタ　ＮＴＴ九州技術開発センタ
ＰＦＵ　ＳＲＡ　ＴＤＫ　アスキー　ウチダエスコ　エイ･エス･ティ　エスケーディ
エヌ･ティ･ティ･システム技術　オムロンソフトウェア　カシオ計算機　キャノン新川崎事業所
クレオ　ケーシーエス　コグノスジャパン　サン･ビルド印刷　シーアイシステムデザイン　システムラボムラタ
ジェーエムエーシステムズ　ジャステック　ジャストシステム　スインク　セイホーソフトウェア
セントラル･コンピュータ･サービス　ソフトウェアコントロール　ダイキン工業　テクノバ　テスク
テックシステムズ　ニコンシステム　ニッセイコンピュータ　ヒラタ･ソフトウェア･テクノロジー
ビーアイテイ　ファコム･ハイタック　フクダ電子　ムラタシステム　リコー　リコーシステム開発
リバテイーシステム　安川電機　近畿日本ツーリスト　古河インフォメーション･テクノロジー
構造計画研究所　三菱電機カスタムＬＳＩ設計技術開発センター
三菱電機セミコンダクタソフトウェア　三菱電機メカトロニクスソフトウェア　三菱電機関西コンピュータシステ
三菱電機東部コンピュータシステム　三菱電機北伊丹製作所　松下ソフトリサーチ　新日鉄情報通信システム
新日本製鉄エレクトロニクス研究所　千代田製作所　池上通信機　中央システム　辻システム計画事務所
東芝アドバンストシステム　東電ソフトウェア　東北コンピュータ･サービス　ＳＲＡ東北　日本ＮＣＤ
日本エム･アイ･シー　日本データスキル　日本ユニシス　日本ユニシス･ソフトウェア　日本情報システムサービス
日本電気ソフトウェア　日立エンジニアリング　日立ビジネス機器　富士ゼロックス情報システム　富士写真フィル
富士通　富士通エフ･アイ･ピー　富士通ビー･エス･シー　（以上76社）

ＳＥＡＭＡＩＬ　Ｖｏｌ．６，Ｎｏ．６～７　１９９２年１月３０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　１０００円（禁無断転載）

編集部から

☆

明けましておめでとうございます．依然として編集部で孤軍奮闘（というほどまじめではありませんが，この年末年始は原稿整理に追われていました）の岸田孝一です．

☆☆

当初の予定では，昨年（いやもう一昨年）の若手の会のレポートをまとめるはずだったのですが，あまりにボリュームが多く，時間がかかりそうなので，別の手持ち原稿を１冊にまとめるほうを先にしました．

☆☆☆

巻頭に載せた２つのアピールは，SEA 会員の何人かもきっとその Emacs や C Compiler のお世話になっている GNU の Richard Stallman 氏からのものです．この LPF（プログラミングぼ自由のための連盟）については，１年以上前にそれが結成されたとき，紹介記事が bit 誌に出たので，御存じの方も多いと思います．このアピールの古い版は，去年ある雑誌に翻訳・掲載されていますが，ちょっとみたところ日本語になっていないようなので，改訂版を Richard から送ってもらって，翻訳しなおしました．

☆☆☆☆

このアピールを読んで，SEA 会員のみなさんの御意見はいかがでしょうか？　翻訳者としての素朴な感想は，「アメリカ社会がずいぶん病んでいるな」ということです．しかし，あらためて周囲をみまわしてみると，日本のコンピュータ業界も，かなりその病気に感染しかかっているように思われます．瞳の中に＄や〒のマークがちらついている人種とは，あまりつきあわないほうがよさそうです．

☆☆☆☆☆

塩谷さん，勝田さん，君島さん，新保さん，熊谷さん，原稿をどうもありがとうございました．他の会員の方々も，ぜひ，技術論文，随筆，アピール，その他何でも結構です．郵便，FAX，または E-Mail で事務局までお送りください．

☆☆☆☆☆☆

昨年，SEAMAIL の船便状況が極限に達したため，事務局の中島さん宛に，数多くの問い合わせや不平が寄せられたそうです．これらの電話または手紙に対する編集部からのお答えは，こうです：「みなさん，そんな心配をする前に，たとえ１ページでもいいですから，原稿を書いて送ってください！」

☆☆☆☆☆☆☆

SEA は，既存の学会その他の組織とちがうボランティア・ベースのプロフェッショナル・ソサイエティであり，この SEAMAIL は会員からの自主的な投稿から成り立っている（べき）ニュースレターです．そのへんのところを，どうかお忘れなく．われわれにとって役に立つ情報は，決して空から降ってくるのではなく，われわれ自身の手で作り出され，交換されるものなのです．

☆☆☆☆☆☆☆☆

ソフトウェアの特許に反対しよう

— Against Software Patents —

1991 年 2 月 28 日

プログラミングの自由のための連盟
The League for Programming Freedom (LPF)

ソフトウェアの特許は，アメリカのコンピュータ産業を荒廃させる危険を含んでいる．過去１０年間に承認されたいくつかの特許が，いまや，たとえば Lotus Development Corporation のようなソフトウェア会社が独自に開発したプログラムを販売するのを妨害するために用いられている．このような状況が続けば，いずれ新しい企業はソフトウェアのマーケットから完全に締め出されてしまうだろう．ほとんどの大きなプログラムは，それを動かすために何十という特許を取らなければならなくなるが，そんなことは実際上不可能である．この問題を解決する方法は１つしかない．それは，ソフトウェアの特許を止めさせることである．

特許制度とコンピュータ・プログラム

アメリカ合衆国憲法を作った人たちが特許という制度を制定したのは，発明家たちに，かれらの発明を一般大衆と共有することを促すためであった．発明の秘密を明らかにする代わりに，発明者には，その特許を１７年間独占する権利が与えられる．特許権者は，他人にその発明を使用する許可を与えることができる．あるいはまた，その使用を拒否することもできる．後で，別の人間が同じ技法を独立に（再）発明したとしても，それを使う権利は与えられない．

特許は，特定のシステムに対しては適用されない．そうではなくて，いろいろなシステムの構築に使われる個々の技法，あるいはそれらのシステムが提供する個々の機能に適用される．いったん，ある技法または機能に特許が与えられると，その権利を持つ人間の許可なしに，特定のシステムの中でそれを使うことができなくなる．たとえ，それが違うやり方で具体化されていてもである．コンピュータ・プログラムは，ふつう，数多くの技法を利用し，数多くの機能を提供するものであるから，一度に多くの特許を侵害する可能性がある．

最近まで，特許の概念はソフトウェアの分野には適用されなかった．ソフトウェア開発者たちは，個々のプログラムに著作権を主張したり，それらを業務上の機密（トレード・シークレット）として扱うことで，かれらの権利を守ってきた．著作権は，その由来からして，特定のプログラムの具体化の詳細をカバーするものであり，プログラムの機能や，使われている一般的な手法には適用されない．そして，業務上の機密という概念は，その定義からいって，機密を知らない人間が行なう開発作業を妨げるものではありえない．

こうした枠組みの上で，ソフトウェア開発は，個々の開発活動に対する何等の禁止事項もなく，きわめて高い収益性を持つビジネスとなり，かなりの額の投資を享受してきた．しかし，いまやこの図式はもう成り立たない．１９８０年代初期にアメリカ政府の方針が変り，その結果，特許申請の氾濫をもたらした．いまや数多くの特許がすでに承認され，現在もその数は加速度的にふえつつある．

多くのプログラマたちは，まだこの変化に気がつかず，その影響の大きさを正しく認識していない．今日，いくつかの訴訟事件が始まったばかりのところである．

特許の不条理性

特許庁および裁判所は，これまで，コンピュータ・ソフトウェアの扱いにはたいへん苦労してきた．特許庁は，

最近まで，コンピュータ·サイエンスの学位取得者を審査官として採用することを拒んできた．ソフトウェア業界と比較できるような給与を支払うこともできなかった．したがって，ほとんどの特許審査官は，ソフトウェアの特許申請を評価する能力を持っていない．つまり，個々の申請が，周知の技法または明白な技法（どちらも却下の理由になる）にもとづくものであるかどうか，を判断することができない．

一般的に使われているソフトウェア開発技法の多くが，コンピュータ·サイエンスの文献に出ていないということで，かれらの仕事はいっそうむずかしくなっている．ある種の技法はわざわざ出版するにはあまりにも明白であり，また他の技法は一般性に欠けている．そして，ある種の技法は，プログラマたちにとって公然の秘密である．

コンピュータ·サイエンスを勉強した人間ならだれでも，広範囲な状況に一般化できるような技法が数多く存在することを知っている．しかし，特許庁は，ある技法の個々の使い方が，それぞれ新たな特許の候補になると考えているかのように見える．たとえば，Apple 社は Hypercard プログラムが特許番号 4736308 違反だと訴えられている．その特許は，２つ以上の文字列の一部分を一緒にスクリーン上に表示する技法（つまり，複数のサブウィンドウのスクロール）に関するものである．スクロールもサブウィンドウも，ともに周知の技法であるが，しかし，それらを組み合わせて使うことは，いまや違法なのである．

特許庁によってある特許が承認されると，必然的にそれは法律的に有効になる．申請がなされる以前に長年使われて来た周知の技法であっても，いったんそれが特許として承認されると，連邦裁判所はそれを支持することになる．法律上の問題が発生した時点で，ある技法が周知のものであることを証明するのは，かなりむずかしい．

たとえば，スクリーン上にカーソルを表示するために排他的論理和を使う技法は，周知であり，また明白でもある（その利点は，同じ操作をもう一度行うことにより，スクリーン上の他のデータを損なうことなしに，カーソルを消去できることである）．この技法は，ほんの２～３行でプログラムすることができる．頭のよい高校生ならだれでも思いつくことである．しかし，それは特許番号 4197590 として承認されており，その特許申請がなされる少なくとも５年以上も前から広く使われていたにもかかわらず，２回の裁判で特許権者の権利が支持された．そして，この特許を持っている Cadtrak 社は，大きなコンピュータ·メーカ数社から何百万ドルもの使用料を得ている．

イギリスでは，従来からある普通のグラフィクス技法（エアブラッシング，ステンシリング，２つの画面を第３の画面の制御のもとに合成することなどを含む）に対して特許権が認められ，その分野の開拓者たちが，それらの技法はかれらが何年も前に開発したものだと証言したにもかかわらず，最近裁判所で特許権者の権利が支持された（それに相当するアメリカの特許は，4633416 号および 4602286 号である．これらはまだ裁判で試されてはいないが，近くそうなるであろう）．

スプレッドシート·プログラムの主な開発者たちは，いま，「自然な順序の再計算」（スプレッドシートの各項目を固定した順序で計算しなおすのではなく，ユーザが行った変更によって影響を受けるものだけを再計算する技法）に関して認められた特許 4398249 号の脅威を感じている．いまのところは Lotus 社だけが訴えられている．しかし，この訴訟がもし原告側の勝訴に終われば，他のスプレッドシート·ベンダたちの希望はほとんど消え失せてしまうだろう．われわれ LPF は，裁判でこの特許を打ち負かせる可能性を持つより先行的な技術を発見したが，しかしわれわれが勝訴できるかどうかは，まだ確実ではない．

プログラマがたまたま特許になっている技法を使ってしまい，その結果訴えられることを防ぐ手段は存在しない．既存のプログラムのどれかを取り上げて，そのパフォーマンスを改善しようとしたりすれば，すでに承認ずみまたは近く承認されるであろういくつかの特許に触れることになる．

いずれは特許庁がソフトウェアをもっとよく理解するようになるとしても，連邦議会または最高裁判所が介入して，これらのばかげた特許の無効を宣言しないがぎり，現在かれらが犯しつつある誤ちが，次の世紀までわれわれを追いかけて来るだろう．

しかし，問題はこれですべてではない．コンピュータ・プログラミングは，特許制度が適用されてきた他の分野とは，根本的に異なっている．特許制度は，もしがソフトウェアに対して「狙い通りに」働いたとしてもなお，それが本来振興しようとしている産業を阻害することになるのである．

「明白」とは何か？

特許制度のもとでは，明白であると判断されたものが特許として承認されたり，裁判で支持されることはない．しかしながら，この制度は，「明白 (Obvious)」ということばを，コンピュータ・プログラマたちにはとても考えられないような仕方で解釈するのである．これまで他の分野で使われてきた明白さの基準は，ソフトウェアに対しては適切でない．

特許審査官や裁判官は，ほんの小さな漸進的な変化でさえもが新しい特許に値すると考える習慣を持っている．たとえば，有名な Polaroid 対 Kodak の裁判では，フィルム中の乳剤の層の数と順番が問題になった．Kodak が使っていた技法と Polaroid が説明した技法との差異が，特許を終了させた．裁判所はこれらの差異は明白なものではないと判定したのだ．

コンピュータ・サイエンティストたちは，プログラミングという手段に手慣れており，いろいろな問題を素早く解決できる．かれらは，解法原理を１つの問題から他の問題へと一般化する習練を積んでいる．そうした一般化の１つに，ある手続きを繰り返したり，分割したりするということがある．プログラマたちはこれを明白なことだと考えている．しかし，特許庁は，上述の複数の文字列のスクローリングに特許を与えたとき，そうは考えなかったのである．

このようなケースは，誤ちであるとは考えられない．特許制度は，まさに当初考えられた通りに働いている．しかし，ソフトウェアに関しては，とんでもない結果を生み出すのである．

わざわざ出版するには明白すぎるもので特許を取ること

ときには，あまりにもそれが明白であるがために，だれもそれについて論文を書こうなどと考えもしなかったような，決して新しくはない技法を特許として申請することもできる．

たとえば，MIT が開発した X Window System を無料で配布しているコンピュータ会社は，現在，「バックアップ記憶 (backing store)」に関する特許 4555775 号を侵害するという理由で AT&T から訴えられる恐れがある．この技法は，複数のプログラムがそれぞれ固有のウィンドウを持ち，それらが互いにオーバラップすることが可能なウィンドウ・システムで用いられる．それは，一時的に他のウィンドウの影に隠された部分をオフ・スクリーンのメモリにセーブしておき，邪魔をしていたウィンドウが消えたときに，素早くそれをスクリーン上に戻すための仕掛けである．

初期のウィンドウ・システムは，複数のプログラムを同時に走らせることができないようなコンピュータで開発された．これらのマシンの記憶容量は小さく，ウィンドウの内容をセーブすることは，明らかに貴重なメモリー空間の浪費でしかなかった．後になって大型の多重処理コンピュータが登場し，バックアップ記憶の技法を利用してそれぞれのプログラムに固有のウィンドウを持たせることが一般的になってきた．そうした技法の組み合わせは当然のことであった．

バックアップ記憶の技法は，AT&T が特許申請をするより以前に，MIT の Lisp Machine System プロジェクトで用いられた（偶然にも Lisp Machine もまた，多重処理を支援していた）．Lisp Machine の開発者たちは，その時点では，バックアップ記憶については何の論文も発表しなかった．それはあまりに明白だと考えられたからである．ただ，プログラマーズ・マニュアルの中で，そのオン・オフをどうするかの説明に関連して言及されているだけである．

しかし，そのマニュアルが出版されたのは，AT&T が特許を申請した１週間後であった．すなわち，特許を打ち負かすための先行技術として考慮されるには遅すぎた．こうして AT&T の特許が有効となった．いずれ MIT は，自分が AT&T より先に使っていた技法を使い続けることを禁じられることになるかもしれない．

そして，その結果として，X Window がフリー・ソフトウェアであるという理解のもとに，それを MIT から受取っていた何十社かの企業と数十万人のユーザが，訴追を受ける危険にさらされることになる（かれらはまた，Cadtrak 社の排他的論理和の特許にも脅かされている）．X Window System プロジェクトは，すべての開発者が自由に使えるウィンドウ・システムの開発を意図していた．この公共的なサービスという目標の達成が，いまや特許制度によって妨害されているのである．

ソフトウェアはなぜ違うのか？

ソフトウェア・システムの設計は，同じ部品数のハードウェア・システムの場合と比較して，はるかにやさしい．たとえば，１０万個の部品からなるプログラム（サイズは５万ライン程度）は，優秀なプログラマが２人いれば，１年で作れるだろう．必要な設備費用は１万ドル未満である．それ以外にかかるコストはプログラマの生活費だけであり，投資額の総計は１０万ドル未満ですむだろう．もし，その開発が大きな企業の中で商業ベースに行われたとしても，開発コストはせいぜい２倍以内であろう．それと対照的に，１０万個の部品から成るふつうの自動車を設計するには，かなり大がかりなチームを必要とし，何千万ドルもの費用がかかる．

また，ソフトウェアは製造コストもきわめて安い．１万ドル以下の値段で手に入るふつうのワークステーションで簡単にコピーすればよいからである．一方，複雑なハードウェア・システムを製造するには，しばしば何千万ドルもかけて工場を建設しなければならない．

なぜこうなのか？　ハードウェア・システムは現実の部品を使って設計されなければならない．それらの部品はコストも異なる．性能にも制約がある．温度や振動，または湿度に敏感であったりする．雑音を発生するかもしれないし，またエネルギーを消費する．一時的に，あるいは永久に故障することもある．物理的にも適切な場所に組み込まれなければならず，故障のさいには交換できるようになっていなければならない．

さらに，ハードウェアの設計においては，それぞれの部品が他の多くの部品の挙動に影響を与える場合が多い．このことが，ハードウェア設計の仕事をきわめて複雑なものにしている．設計が終わるころになって，基本的な数学モデの誤りが明らかになることもある．

コンピュータ・プログラムは，それと対照的に，想像上の数学的な構成要素から組み立てられる．個々の部品の挙動は，近似的にモデル化されるのではなく，抽象的規則によって明確に定義されている．if 文の後に while 文が続く場合，if 文が while 文のエネルギーの一部を吸収してしまい，出力がおかしくなるとか，while 文に余計な圧力がかかって故障するなどといった余分な心配をする必要はまったくない．

しかし，単純な部品から組立られているにもかかわらず，コンピュータ・プログラムは信じられないくらい複雑である．１０万個の部品から成るプログラムは，設計が容易であるにもかかわらず，自動車と同じ程度の複雑さをそなえている．

プログラムを書いたり，製品化したり，販売したりするのに要する費用は，自動車の場合に比べてかなり安いが，特許問題を処理するコストは決して小さくならない．部品の数が同じであれば，平均的にいって，ほぼ同数の特許の対象となる技法が含まれているからである．

訴訟がもたらす危険

現行の特許制度のもとでは，法を守ろうとするソフトウェア開発者は，プログラムがどの特許を侵害しているか判断して，それぞれの特許所有者からその特許の使用許諾（ライセンス）を貰うための交渉を行なわなければならない．ライセンス料がとんでもなく高額になることもあるだろうし，市場での競争相手が特許を持っている場合には，使わせて貰えないかもしれない．個々のライセンス料がたとえ「妥当」なものであっても，それらの金額を合計すると，プロジェクトが実行不可能な額になってしまうかもしれない．だからといって，既存の特許をまったく使わずに開発を進める道を見出すことはほとんど不可能である．

特許制度における最大の危険は，開発者が製品を市場に出した後で，その製品がいくつかの特許を侵害していることがわかるというケースである．その結果発生する裁判とそれに要する弁護士費用は，中規模の会社を破産させるに十分な額になるであろう．

さらに困った問題は，ソフトウェア開発者がこの危険を回避する実際的な方法が存在しないことだ．あるシステムがどの特許を侵害しているかを発見する有効な方法が，まったくないのである．特許検索という方法がないわけではないが，検索それ自体があまり信頼できないし，ソフトウェア・プロジェクトに適用するには費用が高すぎる．

特許検索はあまりに高価である

１０万個の構成要素から成るシステムは，おそらく，すでに特許化されている数百の技法を利用しているであろう．それぞれの特許検索には，何千ドルかの費用がかかる．したがって，特許侵害の危険を避けるための検索費用の総額は簡単に１００万ドルを越えてしまう．これはプログラム開発費をはるかに上回る金額である．

費用はこれだけではすまない．特許申請のための書類は，弁護士によって弁護士のために書かれたものである．プログラマがそれを読んだとしても，かれのプログラムがその特許を侵害しているとはとても考えられないであろう．しかし，連邦裁判所はの判断は違うのである．したがって，いまやプログラム開発のあらゆる段階に，特許関係の法律的専門家が関係することが必要になってくる．

しかし，もしそうした手を打ったとしても，それはただ，後で訴えられる危険を少なくしただけに過ぎず，危険を完全に取り除いたわけではない．やはり，起こり得る訴訟に備えてあらかじめ資金を用意しておくことが必要である．

会社が，あるハードウェア・システムの設計に数百万ドルのお金を使い，さらにその製造に数千万ドルの投資を計画している場合なら，特許問題の処理に必要な１～２万ドルの経費は許容範囲内である．しかし，きわめてコストの安いプログラミング・プロジェクトの場合には，それだけの余分な経費はとても負担できない．個人や小さな企業の場合はなおさらである．ソフトウェアの特許は，ベンチャー企業家を市場から締め出す働きをする．

特許検索は信頼性が低い

もし，開発者が特許検索の費用を負担できるとしても，それは既存特許の使用を回避するための手段としては，

はなはだ信頼性が低い．特許検索は，現在審査中の特許申請について，何も教えてくれないからである（それらは特許庁のなかに機密扱いで保持されている）．ソフトウェアの特許が認められるのには，平均して数年を要するから，これは深刻な問題である．ある特許が申請された後で，別の開発者が大きなプログラムの設計を開始し，特許が認められる前に製品を出荷することもあり得る．そして，後になってから，そのプログラムの配布が法律違反であることを知るのである．

たとえば，広く使われているパブリック・ドメインのデータ圧縮プログラム compress は，IEEE Computer 誌所収のある論文に書かれたアルゴリズムにしたがっている（このアルゴリズムは PKZIP その他のマイクロコンピュータ用のポピュラーなプログラムの中でも使われている）．これらのプログラムの開発者およびユーザは，いま，原論文の著者の１人に特許 4558302 号が発行されたことを知って，当惑している．Unisys 社は，現在，このアルゴリズムの使用料を要求しようとしている．compress プログラムは依然としてパブリック・ドメインにあるが，それを使うことは訴訟の危険を冒していることを意味する．

特許庁は，ソフトウェアの特許を分類するための実際的な枠組みを持っていない．特許は，ほとんどの場合，「鉄を鋼鉄に変換する」というように，最終的な結果にしたがって分類される．しかし，現在，多くのソフトウェア特許は，プログラムの中での使い方が，そのプログラムの最終目的とは全く独立しているようなアルゴリズムに適用されている．したがって，たとえば，人間の音声を分析するプログラムが，高速フーリエ変換のスピードアップに関する特許を侵害する恐れもあれば，抽象代数のプログラムが（大きな数の掛け算にさいして）同じ危険を持つことになる．しかし，このような可能性を検索すべき特許のカテゴリ分けはほとんど不可能であろう．

すでに特許化されたソフトウェア技法のリストを作成し，それをメインテナンスするのは容易ではないか，あるいは単にそのリストをどこかに記憶しておけばいいのではないか，といわれるかもしれない．しかし，そうしたリストの管理はほとんど不可能に近い．実際，１９８９年にこの分野を専門とする弁護士たちのグループが編集したリストには，このペーパで引用したいくつかの特許が漏れている．

あいまいな特許

あなたが何かの発明を考えたときには，おそらく，短いことばでその特徴を表現しようとするだろう．たとえば，「固定の湾曲した翼を持った空を飛ぶ機械」とか，「マイクロフォンとスピーカを使った電気的な通信装置」といったぐあいである．しかし，ソフトエア特許の多くは，複雑で詳細なプロセスを扱っており，とても簡単にいいわらわすことはできない．また，かなり多くの特許は，それ自体が複雑なよく知られたプロセスのスピードアップや変形に関するものである．

これらの特許の多くは明白でもなく，華麗でもない．それらは「あいまい」である．有能なソフトウェア設計者なら，１つのプロジェクトの進行中に，そうした改良案をいくつも「再発明」するであろう．しかしながら，ソフトウェア技法を改良する道はたくさんあるので，２つのプロジェクトにおける改良案が，特定のものに一致する可能性は低い．

たとえば，IBM は，最適化コンパイラが行うべき周知の計算処理に関して，複雑ではあるが巧妙な高速化技法の特許をいくつか持っている（特許番号 4656583 その他）．これは，たとえば，レジスタの色分けや，すでに評価ずみの式の計算などを含んだものである．

特許はまた，すでに広く使われている技法の組み合わせに対しても認められる．１つの例は，IBM の 474240 号特許である．それは，shared copy-on-write segments に関するものである．この技法は，いくつかのプログラムが，１つのファイル中の情報を表わすメモリの同じ部分を共有できるようにするものである．もしプログラムのどれかが，そのファイルのあるページに書き込みを行うと，すべてのプログラム中のそのページは，新しい

コピーで置き換えられる．そうすることによって，それらのプログラムは，ファイルを共有するのではなく，そのページを互いに共有し続けるわけである．

「セグメントの共有」および「書き込みにもとづくコピー」の概念は，１９６０年代から使われてきたものである．この特定の組み合わせは，たしかに特別な機能を提供する新しいアイデアかもしれない．しかし，それは決して発明ではありない．にもかかわらず，特許庁はそれを特許に値するものと判定した．この特許は，いまや，新しいオペレーティング・システムを開発しようとするすべての人々が考慮に入れなければならないものとなってしまった．

こうしたあいまいな特許は，地雷のようなものである．すべてのソフトウェア開発者は，そうした特許をあらかじめ発見し回避するよりも，これらの技法を再発明する可能性のほうが大きい．そして，特許法違反で訴えられるのである．これらの特許のどれか特定の１つに行き当たる可能性は小さいが，しかし特許の件数が非常に多いので，地雷を１つも踏まずに遠くまで進むことはできない．あらゆる基本的ソフトウェア技法には数多くの変形があり，また数多くの組み合わせがあり得る．アメリカ特許庁は，これまでに，少なくとも２千件の特許を認めている．EDS 社がまとめたリストによれば，１９８９年だけで７百件以上ある．このペースはさらに加速するものと予想される．１０年以内に，プログラマたちはもはや盲滅法に行軍する以外になくなるだろう．そして，かれらが地雷を踏まずに生き残れる確率はほとんどない．

特許のライセンスにも問題がある

大きなソフトウェア会社の多くは，自社でいくつもの特許を取得することによって，この問題を解決しようと考えている．つまり，主要な特許の多くを保有している他の大企業とクロスライセンスを結ぶことにより，これまでと同じように，自由に仕事を続けて行こうというわけである．

Microsoft や Apple あるいは IBM といった大企業は，この方法でビジネスをられるかもしれない．しかし，後発の新しい会社は締め出されてしまう．将来，自分自身の特許を１つも持たずにソフトウェア・ビジネスを始めようとする企業は，これらの巨人が決めた特許使用料を（それがいかに高額であろうと）支払うことを強いられる．最近起こった Borland 社および Santa Cruz Operation 社に対する Lotus 社の訴訟事件は，このメカニズムがどのように働くかを示すよい例である（この訴訟そのものは，特許というよりは，拡張された著作権の概念に関係するものであるが...）．

特許の独占的な権利を獲得し，他人を訴訟で脅かすことだけを商売にしているような会社を相手にした場合には，たとえ巨大企業といえども，クロスライセンスによって身を守ることは不可能である．たとえば，ニューヨークに本拠を置く Refac Technology Development Corporation という会社がある．この会社は，前に述べた「自然な順序での再計算」に関する特許権所有者の代理人である．Refac 社は，その社名とは裏腹に，訴訟を起こす以外，何らの技術開発も行っていない．つまり，同社は，クロスライセンス協約に参加する理由をまったく持っていない．「排他的論理和」に関する特許所有者の Cadtrak も，同じような訴訟会社である．

Refac 社は，主要なスプレッドシート・プログラムのベンダに対して，売上げの５パーセントを特許使用料として支払うよう要求している．もし，未来のプログラム商品が，このような特許を２０件侵害したとすると，ライセンス料の合計は販売価格の１００％を越えてしまう．コンピュータ・プログラムの複雑さと，多くの特許の適用範囲の広さを考えれば，これは決して非現実的な仮定ではない（実際には，ほんの数件の特許侵害を犯しただけで，プログラム商品の利益はすべて失われてしまう）．

基本的な疑問

合衆国憲法によれば，特許制度の目的は，「科学および有用な技術の進歩を促進する」ことである．したがって，

ここでの基本的な疑問点は，もともとソフトウェアの進歩を奨励する方法として考えられたソフトウェアの特許が，はたしてほんとうにそういう働きをするのか？　それとも反対に進歩を遅らせてしまうのか？　ということである．

ここまで，われわれは，特許がいろいろな形で通常のソフトウェア開発をむずかしくする事情を説明してきた．しかし，特許がもともと意図していた利点とは，いったい何だったのだろうか？　より多くの発明を生みだし，それらの発明の成果をより多く一般に公開させることではなかったのか？　ソフトウェアの世界で，そのことは実際にどの程度起こり得るのだろうか？

ソフトウェア特許制度から社会が得られる利益はきわめて少ない．というのは，ソフトウェアの世界においては，すでに特許制度が始まる以前からたくさんの発明がなされ，それらは，だれもが利用できるように，いろいろな雑誌に発表されていたからである．発明の習慣はあまりにも盛んであり，そのため，同じ発明がしばしば繰り返しなされることもあった．

独立の再発明は日常茶飯事

特許は完全に独占的である．すべての人間は，特許になったプロセスを使うことを禁じられる．たとえ独立にそれを再発明した場合でもである．この方針は，暗黙の内に，発明がきわめてまれな出来事であり，貴重なものであることを仮定している．そういう状況のもとでのみ，特許制度は有益なのである．

ソフトウェアの分野は，常に再発明が行われる世界である．だれかがいったように，プログラマたちは，他の分野の研究者が１年間に生み出すよりも数多くの「発明」を，毎週のように行い，結果を使い捨てている．大規模なソフトウェア･システムの設計が比較的容易であることから，たくさんの人間がこの分野で仕事をしている．それらのプログラマたちは，１本のプログラムを開発する過程で，数多くの問題を解決して行く．これらの解決法は，他のプログラマが同様の問題に対処するさい，何度も繰り返し再発明される．

このように，独立の再発明が一般的であることは，特許の通常の目的を否定するものである．特許は発明を促進し，なかんずく，その内容の一般への開示を奨励するための制度である．もし，ある種の技法が何度も再発明されるものであるなら，さらに多くの人々がそれを発明するよう奨励する必要はまったくない．そうした発明者のだれかが（何らかの意義を感じて）それを出版しようとしているなら，その技法の自由な利用を禁止するという対価を払ってまで，わざわざ特定の発明者に出版を奨励するいわれはない．

発明重視の弊害

日米両国産業の比較分析を行っている多くのアナリストたちは，日本が高品質の製品を製造することに成功した原因を，かれらが，世の目を引く発明よりも，漸進的な改良や，ユーザにとって便利な機能や品質を重視したからだとしている．

ソフトウェアの場合，開発が成功するかどうかは，まず第一に，細部を正しくすることにかかっている．そして，有用なソフトウェア･システムの開発を行なう場合，作業の大部分は製品の細部に関わっており，発明は，仕事の中では比較的に重要性が低い部分である．

ソフトウェアの特許というアイデアは，製品よりもむしろ発明を重視するというアメリカ人の誤った先入観の１例である．そして，具体的な個々の特許はこの視点の誤りを助長し，本来ならもっとよいソフトウェア製品を生み出すはずの開発作業を邪魔しているのである．

技術革新に対する妨害

ソフトウェアの特許は，開発に携わるプログラマの人数を減少させることによって，実際に技術革新を妨げる．ソフトウェアにおける技術革新の多くは，開発作業の過程でさまざまな問題を解決しているプログラマたちによってもたらされるのであって，何かを発明して特許を取ることを目的にしたプロジェクトから出てくるわけではない．いいかえれば，技術革新はソフトウェア開発の副産物なのである．

特許が開発をより困難なものにし，開発プロジェクトの数を減らすことになれば，開発の副産物すなわち新しい技法が生み出される数もまた減小する．

特許はほんとうに有益か？

ソフトウェアの特許制度は，一般に社会全体にとって有害であるが，われわれは決して，個々のソフトウェア特許のすべてが有害だと主張するものではない．慎重に研究すれば，(大部分の場合を除外するような)特定の狭い条件下でなら，ソフトウェアの特許を認めることが有益である場合が発見されるだろう．

それでもなお，現時点で行うべき正しい行為は，もっと大きな損害が生じる前に，可能な限りすみやかに，すべてのソフトウェアの特許を廃止することである．慎重な研究はその後で実施すればよい．

弁護士を除けば，ソフトウェアの特許制度を緊急に必要としている人間などいないことは明かである．特許制度が導入される以前のソフトウェア産業には，特許によって解決できたような問題など，何も存在していなかった．発明が足りなかったわけでも，投資が不足していたわけでもなかった．

ソフトウェア特許の全廃は，理想的な解決策ではないかもしれない．しかし，ほぼそれに近いし，大きな改善をもたらす．その単純さは，人々が詳細な議論をすることによる大きな遅れを避けるのに役立つ．

もし，ある例外的な場合にソフトウェア特許性の有益性が示されたら，そのときには，必要なら再び法律を変えることができる．現在の破滅的な状況をその日まで引きずって行く理由は，まったく存在しない．

ソフトウェア特許に対する法的な疑問

驚くべきことではあるが，特許法を拡張してソフトウェアに適用したことには，いまだに法律的に疑問がある．この拡張は，１９８１年に最高裁判所が行ったある判決（Diamond 対 Deihr 事件）の極端な解釈にもとづくものである [*].

---

[*]"Legally Speaking", Communications of the ACM, August 1990 参照．

伝統的に，特許が認められるプロセスの種類は，(鉄を鋼鉄に変えるといった)物質変換のプロセスに限られていた．われわれがプロセスだと考える他の多くの活動，たとえばビジネス上の手法，データの解析，「精神的プロセス」などは，対象からまったく除外されていた．これは，「主題」原則 ("Subject matter" doctorine) と呼ばれるものである．

Diamond 対 Deihr 事件は，特許庁によって，この原則をくつがえすものだとして解釈されたが，裁判所はこれを明確に却下することをしなかった．このケースは，ゴムの矯正（すなわち物質の変換）プロセスに関するものであった．係争点は，そのプロセスの中でコンピュータ・プログラムを使うことが特許に値するか否かということであった．そして，裁判所は，それは特許に値すると裁定した．特許庁はこのきわめて狭い条件つきの判定

を，ソフトウェア技法に対して無制限に特許を与えることへの青信号だと解釈した．そして，よく知られた通常の活動を行うさいのソフトウェアの利用にまで拡大して適用できると考えたのである．

多くの弁護士たちは，口では，特許の新しい境界線は，何十年かの時間をかけて，費用のかかる一連の裁判を通じて決定すべきだといいながらも，この変化を歓迎した．こうした一連の動きは，たしかに弁護士にとっては好都合であるが，ソフトウェア開発者およびユーザにとっては，はなはだ迷惑である．

ソフトウェア特許を廃止する１つの方法

われわれは，特許の対象領域からソフトウェアを除外する１つの法律を提案したい．その法律は，いかなる特許であろうと，それがソフトウェアとして具体化された場合には適用されず，ただ，設計のむずかしいハードウェア形式での具体化のみに適用されるということを規定するものである．この方法の利点は，特許の審査にさいして，その申請をハードウェアとソフトウェアに分類する必要がなくなることである．

この目的のために，どうやってソフトウェアを定義するのか？　つまり，どこで線を引くのか？　が問題になる．この法律の目的からすれば，ソフトウェアの定義は，ソフトウェアの特許が特に有害であるような特性にもとづいて行われるべきである．すなわち：

- ソフトウェアは，想像上の堅固な数学的構成要素から組み立てられ，各構成要素からの出力は，入力側の構成要素のいかんによる影響は受けない．

  想像上の数学的構成要素は，抽象的な規則によって定義され，したがってそれらが故障することは，その定義からして起こり得ない．これらの構成要素から組み立てられるシステムの挙動は，同様に，各構成要素に抽象的な規則を順次適用して行くことによって定義される．
- ソフトウェアは，容易に安い費用でコピーできる．

この基準にしたがえば，素数を計算するプログラムは１つのソフトウェアである．しかし，同じ計算を実行するために特別に設計された機械的装置は，ソフトウェアではない．機械部品には摩擦があり，相互の動作に干渉し合い，故障することもあり，動作可能な機械を作るには，てまをかけて物理的に組み立てなければならないからである．

どんなソフトウェアも，それを動かすには，ハードウェアのプラットホームを必要とする．ソフトウェアは，ある計画にしたがって，ハードウェアの機能をある組み合わせで操作する．われわれの提案は，このような形で機能を組み合わせることは，特許の侵害を絶対に起こさないということである．もし，ハードウェア自身が特許を侵害していなければ，プログラムの制御の下で，あるやり方でそれを使うことは，特許の侵害になるはずはない．実際，プログラムとはプログラマの精神の延長であって，プログラマの代理人としてハードウェアを制御しているのである．

通常，ここでいうハードウェアとは，汎用コンピュータであり，特定のアプリケーションを想定して作られた機械ではない．そのようなハードウェアは，コンピュータの製造に関する特許を除けば，いかなる特許も侵害するはずがない．われわれの提案は，ユーザがあるプログラムを汎用コンピュータ上で走らせる場合に，コンピュータの製造に関する特許以外の，いかなる特許も適用しないことを意味している．

伝統的なハードウェアとソフトウェアの区別には，いくつかの複雑な特性が含まれており，それらはお互いに連携して働く可能性を持っている．ゲートアレイやシリコン･コンパイラなどの新技術の登場により，その区別はあいまいになってしまう．そうした新技術は，ハードウェア的な特性とソウトウェア的な特性とを組み合わせたものだからである．しかし，これらの新技術のほとんどは，特許審査という目的のためには，上述の判定基準

を用いて，少しのあいまいさもなしに，ハードウェアまたはソフトウェアのいずれかに分類することができる．いくらかは灰色の領域が残るかもしれないが，しかしそれらは比較的小さく，通常のソフトウェア開発に対して特許制度が引き起こす問題を解決する障害には，必ずしもならない．これらの領域は，最終的には，ハードウェア，ソフトウェア，あるいはその中間の何物かとして扱われることになるであろう．

あなたにできること

ソフトウェア特許を廃止する運動を助ける１つの道は，われわれの組織 LPF (the League for Programming Freedom) に参加することである．LPF は，ソフトウェア特許およびとインタフェイス著作権に反対するプログラマならびにユーザの草の根組織である（LPF は，個々のプログラムの著作権には反対していない）．個人会員の年会費は，企業の従業員は 42 ドル，学生 10.5 ドル，その他は 21 ドルである．アクティブな活動家はもちろん大歓迎だが，自分の時間を割いて貢献することができない会員も歓迎する．

LPF への連絡は，下記の宛先まで，電話または E-Mail または郵便でお願いしたい：

League for Programming Freedom
1 Kendall Square #143
P.O.Box 9171
Cambridge, MA 02139
U.S.A
TEL: (617) 243 - 4091
E-Mail: league@prep.ai.mit.ed

アメリカ国内では，もう１つの援助手段として，連邦議会に手紙を出すという方法がある．あなた自身の州から選出された議員宛に出してもよいが，もっと効果的なのは，この問題を審議している小委員会宛に出すことである．

House Subcommittee on Intellectual Property
2137 Rayburn Bldg
Washington, DC 20515
U.S.A

Senate Subcommittee on Patents, Trademarks and Copyrights
United States Senate
Washington, DC 20510
U.S.A

各州選出の議員には (202) 225-3121 で電話がかけられる．あるいは次の住所宛に手紙を出せばよい．

Senator So and So
United States Senate
Washington, DC 20510
U.S.A

Representative Such and Such
House of Representatives
Washington,DC 20515
U.S.A

１つ１つの特許との闘い

ソフトウェア特許の全廃に成功するまでの間，われわれは，個々のソフトウェア特許を覆すことに努力しなければならない．それにはかなりのお金がかり，にもかかわらず問題のごく一部しか解決できないが，しかし，何

もしないよりはましである．

法廷で特許を覆すには，先行的な技術を見つけ出すことが必要になる．それはやさしい仕事ではない．LPFは，ソフトウェア特許訴訟の被告側を援助するために，この種の情報の交換所の役割をはたそうと考えている．それがうまく行くかどうかは，あなたの援助いかんにかかっている．もし，あなたがいずれかのソフトウェア特許に関する先行的な技術について御存じなら，どうかその情報を上記のLPF事務局までお送りいただきたい．

もし，あなたがソフトウェア関係の仕事をしているなら，特許申請への協力を拒否することで，ソフトウェア特許の防止に個人的に協力できる．その詳細は，それぞれの状況に応じて異なるであろう．

結論

特許制度の適用範囲からソフトウェアを除外することは，恐ろしく費用のかかる特許検索や，既存の特許を回避するための無駄な労力，あるいは避けがたい訴追の危険などから，ソウトウェア開発者を守るものである．

もし，状況が何も変わらなかったとすれば，現在は有効で創造的な活動も，実行不可能なほど費用のかかるものになる．たとえば，歩道の敷石のそれぞれに所有者がおり，歩行者がそれを踏むたびにライセンスを要求されるものと仮定しよう．この制度の下で，1ブロックの街路を歩くのにどれだけの交渉が必要であるかを想像してほしい．ソフトウェアの特許が存続した場合のプログラミングは，ほぼそれと同じようなものになる．これまでコンピュータ革命の原動力であった創造性の閃きや個人主義は消滅してしまうのである．

ユーザ・インタフェイスの著作権に反対しよう

— Against User Interface Copyright —

1991 年 10 月 20 日

プログラミングの自由のための連盟
The League for Programming Freedom (LPF)

１９９０年６月，Lotus 社は，Lotus 1-2-3 で使われているのと同じキーストローク・コマンドにしたがうスプレッドシートを具体化した小さな会社 (Paperback Software) に対する著作権侵害の訴訟に勝訴した．Paperback 社は 1-2-3 からコードをコピーしたことによって告訴されたのではない．ただ，互換性のあるユーザ・コマンドをサポートしたことに対して訴えられたのである．このような模倣は，近年，裁判所が著作権法の適用範囲を拡大し，われわれが予想もしなかったような判決を次々に下すまでは，ごくあたりまえの行為であった．

それから１週間もしないうちに，Lotus 社は，1-2-3 とほんの少ししか似ていないコマンド言語を持つスプレッドシート Quatro に関して，Borland 社を告訴した．こうしたキーストローク・シーケンスの類似性，および 1-2-3 をエミュレートするようにインタフェイスをカスタマイズできる機能は，著作権の侵害だとするに十分だというのが，その主張である．

もっと恐るべき事実がある．Apple Computer 社は，Microsoft および Hewlett Packard の両社を，Macintosh システムと部分的に似たディスプレイを持つウィンドウ・システムを具体化したとして告発し，一方，Xerox 社は，昔の Xerox Star システムからいくつかの一般的なコンセプトを引き出し，それを Macintosh システムとして具体化したことで，Apple 社を告発した．これらの訴訟において，原告側は，Lotus 事件の判決を拡張し，かなり広範囲なユーザ・インタフェイスに関する著作権を確立することを狙っている．Xerox 社の訴訟は技術的な理由で却下されたが，もし同社が勝訴したとすれば，Apple 社の訴訟以上に広範囲な独占をもたらしたことであろう．

そして Ashton-Tate 社は，Fox Software 社を，dBase で使われているのと同じプログラミング言語を受けつけるデータベース・プログラムを具体化したことで告発した．このいささか特異な訴訟は，１９９１年に Ashton-Tate 社を買収した Borland 社によって取り下げられたが，しかし，プログラミング言語が著作権化される可能性はまだ残っている．Adobe 社は，まだだれに対する訴えも起こしてはいないが，Postscript 言語に対する著作権が自分のものだと主張している．また，Wolfram Research 社は Mathematica 言語の著作権を主張し，Univercity of California に対して，訴訟をちらつかせている．もし，あるプログラミング言語の著作権が認められたとすれば，長年にわたってその言語でプログラムを書いてきたユーザに対する衝撃は，まさに破壊的であろう．

このペーパは，主として特定のユーザ・インタフェイスの著作権を扱っているが，ほとんどの議論は，より広範囲な他の独占問題に関しても適用できよう．

ユーザ・インタフェイスとは何か？

ユーザ・インタフェイスとは，ある機械を操作するためにあなたが学習しなければならないもの，いいかえれば，その機械にあなたの意志を伝えるための言語である．タイプライタのユーザ・インタフェイスはキーの配列である．自動車のユーザ・インタフェイスには，方向転換のためのハンドル，スピードを上げ下げするためのペ

ダル，方向指示のためのレバーなどが含まれる．

その機械がコンピュータ・プログラムである場合，インタフェイスは，コンピュータ自身のそれ（キーボード，スクリーン，マウス）に，そのプログラム特有のものを加えた形になる．これらはふつう，コマンドやメニュー，プログラミング言語，データのスクリーン上での表示形式などである．

ユーザ・インタフェイスの著作権とは，その使い方が，政府の管理の下で，だれかの手で独占されることを意味する．タイプライタの例でいえば，各メーカはそれぞれ異なったキー配列を使うことを強制されることになる．

著作権の目的

アメリカ合衆国憲法は，著作権の目的を「科学および有用な技術の進歩を促進する」ことだと述べている．そこには，著作権化された作品の利用者を傷つけることによって著作権保有者を豊かにしようなどという意図など，毛頭存在しない．

最高裁判所は，Fox Film 対 Doyal 事件の判決の中で，その理由を次のように述べている：「合衆国の基本的な関心および（著作権の）独占認可の基本的な目的は，著作権所有者の労働から社会によって引き出される一般的な利益にある」と．

いいかえれば，著作権は政府の管理下における一種の独占であり，それはきわめて重大な形で公共の自由を束縛するので，それが社会にもたらす利益が社会にとってのコストを大幅に上回るときだけ，はじめて正当化されるのである．

個人の自由を追求する精神は，当然のことながら，いかなる形の独占に対しても反対の立場を取る．最高裁判所の判例あるいは自由主義の原則にしたがえば，基本的な質問は次の通りである： すなわち，ユーザ・インタフェイスの著作権は，社会に対してどんな価値をもたらすのか？ そして，そのためにわれわれはどれだけの対価を支払わなければならないのか？

第１の反対理由：いま以上の開発意欲は必要でない

Star，Macintosh システム，1-2-3，dBase などの開発者たちは，もしインタフェイスの著作権が存在しなければ，それらの製品を開発するのに十分な開発意欲（インセンティブ）は得られなかったろうと主張している．この表明が誤りであることは，かれら自身の行動によって証明できる．

1986 年以前には，だれもユーザ・インタフェイスの著作権の話など聞いたことがなかった．コンピュータ産業は，ユーザ・インタフェイスの模倣があたりまえの合法的な行為であるという仕組みの中で発展してきた．この仕組みの中で，現在の訴訟の原告たちはかれらの製品を開発する決定を下した．つまり，現実にそのような意志決定の場面に直面したとき，かれらは，自分たちが「十分な開発意欲」を持っていると判断したのである．

競争相手の会社がこれらのインタフェイスを模倣することが自由であったとしても，それは，オリジナルな製品が成功し，投資に対する大きな見返りを生み出すことの妨げにはならない．実際に，かれらは大きな成功をおさめ，それらのユーザ・インタフェイスは「事実上の標準」としての地位を確立している（Xerox Star だけは，類似製品が存在しなかったにもかかわらず，マーケティングの拙さのために失敗した）．

たとえ，インタフェイスの著作権が，開発意欲をよりいっそう増進したとしても，それは，ユーザ・インタフェイスに対して，これ以上の改良をもたらすものではない．いったん酒瓶を飲み干してしまった後は，いくら吸い

込んでも，もう何も出てきはしない．現在のコンピュータ業界における開発意欲はきわめて大きく，開発する価値のあるアイデアを持つ人間すべてを動機づけるのに十分である．公共の負担によって，より以上の開発意欲を導入することは，ただ開発コストを増加させるだけに終わるだろう．

第２の反対理由：それは小さな会社を保護しない

ユーザ･インタフェイス著作権の提唱者たちは，これによって，小さな会社が大企業の手で市場から追い出されることが防止できると主張している．しかし，周囲の状況をよく見てみよう．現在係争中のインタフェイス著作権の原告席に立っているのは，いずれも著名な大企業ばかりである．著作権は，そのインタフェイスが実質的な標準になったときに，破壊的な力を持つ．しかし，弱小な会社の製品は，ふつう，ユーザの数も少なく，そのインタフェイスはあまり人に知られていない．このような状況のもとでは，ユーザ･インタフェイスの著作権は，それほど小さな会社の助けにはならない．

たとえば，１万人の顧客を持つ小さな会社があるとしよう．ある大企業が，その商品には，潜在的に百万人規模の市場があり，ただしその市場は小さな会社にはとうていカバーできないと考えたとする．その場合，その大企業は，類似の商品を開発し，持ち前の営業力を利用して，潜在的な市場を占有しようと努力するだろう．

ユーザ･インタフェイスの著作権は，決してこの状況を変えることはできない．著作権によって，大企業がもし，これまでの商品と互換性のないインタフェイスを開発することを余儀なくされたとしても，そのことは，大多数の潜在的な顧客にはほとんど何の影響も与えない．かれらは，他のインタフェイスについて聞いたこともないからである．結局，これらのユーザは大企業の製品を買うことになるだろう．

それ以上悪いことに，大企業の製品が実質的な標準の地位を獲得したとすると，ユーザ･インタフェイス著作権は，小さな会社に対して，さらに不利に働く．そうすれば，新しい顧客は，大企業の製品を選択するもう１つの理由を持つことになる．生き残るために，小さな会社はこの標準に対する互換性を提供しなければならないが，ユーザ･インタフェイス著作権はそれを許さないのである．

小さな会社が成功するのは，独占的な手段に頼る代わりに，身軽さ，低いオーバヘッド，危険を冒す意欲など，かれら独自の利点を生かすことによってである．

第３の反対理由：インタフェイスの多様性は望ましくない

著作権制度は，多様性を促すべく考えられたものである．制度の細部は，この目標を達成するように働く．小説や歌など，伝統的な著作権の領域では，多様性が第一の目標である．読者は，いままでに読んだことのない小説を読みたいと望んでいる．

しかし，多様性はインタフェイス設計の目標ではない．どんな種類の機械のユーザも，使いやすさのために，ユーザ･インタフェイスの統一性を求めている．こうして，自動車のダッシュボード記号が統一され，それによって，運転免許を持つドライバならだれでも，余分な講習を受けることなく，どの車でも運転できるようになっている．インタフェイスの非互換性は，どうしてもそうすべき価値があるときにだけ支払うべき代償であって，決して利点ではない．

すぐれたインタフェイスを考え出すことは容易ではない．しかし，単にいままでのものと異なるインタフェイスを考えるのは簡単である．インタフェイスの著作権は，そういった意味での「インタフェイス開発」を促進するだろう．その結果，無意味な非互換性が氾濫することになる．

第４の反対理由：意味のある競争が減少する

インタフェイス著作権が支配する環境下では，すでに声価が確立した製品に対して互換性のある競争がなくなってしまう．ユーザにとってみれば，違うブランドの製品に乗り換えるためには，もういちど訓練をやりなおさなければならない．

しかし，ユーザは，たとえそれによってかなり大幅な改善がもたらされる場合にも，訓練のやり直しをしたがらないものである．たとえば，何十年か前に考案された Dvorak キーボードのキー配列は，標準の "QWERTY" 配列と比較して，より速くより正確にタイプできるものであるが，現在ほとんど一般には使われていない．新人のタイピストたちでさえ Dvorak を練習しようとしない．かれら（かの女ら）は，世の中でふつうに使われているタイプライタのキー配列を学習することを望むのである．

このように，消費者の側で余分な努力を必要とするような代替製品は，実質的な競争相手とはなりえない．確立されたインタフェイスの独占は，実際上，それによってアクセスできる機能の独占をもたらす．商品の価格は上昇し，技術進歩は停滞する．メーカにとっては幸運な状況であるが，社会一般にとってはよいことではない．

第５の反対理由：非互換性はなくならない．

もし，自動車のハンドルに５０年間のインタフェイス著作権が与えられていたら，どうなったであろうか？その期限はつい最近まで続いていたはずである．その著作権の有効期間中，方向転換のためにジョイスティック，レバー，ペダルなど，さまざまなインタフェイスを持つ自動車が登場したことであろう．われわれユーザは，運転を覚えるのにブランドを選ばなくてはならず，また車種の変更は容易なことではなかったであろう．

著作権の有効期限が切れれば，メーカは，現存するインタフェイスの中で一番よいものに切り換えることができるようになる．しかしながら，もし Ford の車は丸ハンドル操作され，GM の車はペダルで操作されていたとすれば，どちらのメーカにとっても，これまでの古い顧客を見捨ててインタフェイスを切り変えるのは，容易なことではあるまい．単一のインタフェイスへの統合には，おそらく何十年かの時間を必要とするであろう．

第６の反対理由：ユーザ側の投資は開発者側より大きい

現在の訴訟事件の原告たちは，それらのユーザ・インタフェイスが，かれらの側における巨額な投資をあらわすものだと主張している．

実際には，コンピュータ・プログラムのユーザ・インタフェイス設計に使われる労力は，プログラム自体を設計する費用に比較すれば，ふつう少ない．ユーザ・インタフェイスに多額の投資をしているのは，それを使うための訓練を行っているユーザである．ユーザは，1-2-3 の使い方を学ぶために，Lotus 社がプログラム全体を開発するのに使った以上の時間と金を消費している（ Lotus がインタフェイスの開発に全体のコストのうちどれだけを使ったかは知らないが）．

したがって，もし，投資額の大きさが所有権を正当化するものであるならば，ユーザこそが所有者になるべきである．だれが（市場で）それを使ってよいかを決定する権利は，ユーザに与えられるべきである．（１９８９年１月中旬の）Infoworld によれば，コンピュータ・ユーザたちは，一般に，ユーザ・インタフェイスの著作権は有害だろうと考えている．

第７の反対理由：ソフトウェアの共有に対する差別

ユーザ・インタフェイスの著作権は，フリーウェアやシェアウェア，あるいはパブリック・ドメイン・ソフト

ウェアといった形での，ソフトウェアの自由な再配布を差別するものである．

所有権がはっきりしているプログラムの場合，そのインタフェイスをライセンスの対象にすることは可能である**かもしれない**．もし所有者にその意志があれば，これらのライセンスには，ふつう，一定の料金をコピーごとに支払うことが要求されるだろう．自由に再配布可能なプログラムの場合には，その料金の金を集める方法はまったく存在しない．その結果，かなり多くの種類のユーザ·インタフェイスが，そうした所有権のないソフトウェアでは利用できなくなってしまい，しかも，その範囲はどんどん拡大するであろう．

これらの自由に再配布可能なプログラムの作者たちは，社会に対して，それらのプログラムを共有し，その動作を自由に研究し，ときにはそれを変更する権利を寄付しようと考えているのである．これは，一種の社会的な奉仕である．それは，技術革新ほど一般的なものではない．しかし，ある種類の寄付行為を制約し，別の種類の技術革新だけを促進するのは，あまり意味のあることではないだろう．

第８の反対理由：著作権は脅迫の道具になり得る

インタフェイス著作権の適用範囲があまりにあいまいで，かつ潜在的に広いので，すべてのプログラマが，だれかに訴えられる可能性がある．ほとんどのプログラムはインタフェイスを必要としている．そして，そのインタフェイスを設計するには，いつかどこかで見たアイデアを土台にする以外，方法はない．現在ふつうに使われているものと類似性を持たないインタフェイスを，しかも実用的な形で考え出せるのは，偉大な天才だけであろう．したがって，ほとんどすべてのプログラミング·プロジェクトは，インタフェイス侵害の危険にさらされることになる．

「国の守りには百万ドルかけてもよいが，貢ぎ物には１セントも出すな」という外交原則は，今日のビジネスの世界ではまったく尊重されていない．顧客や投資家は，通常，訴追の標的になった企業を敬遠する．最終的には裁判に勝つかもしれないが，それには長い時間がかかるし，もしかしたらそれ以前に破産してしまうかもしれない．したがって，ロイヤリティを支払うか，それとも告訴されるかの選択を迫られた場合，たとえ勝訴の可能性があったとしても，たいていの企業はお金を払う方を選ぶのである．

こうした企業行動のパターンはよく知られているので，ある種の人間はそれを利用して，抗議文を送ったり，勝てそうもない訴訟で脅かしたりする．インタフェイス著作権が何らかの形で存在する限り，この形の脅迫行為が，より広範囲にわたって展開されるだろう．

第９の反対理由：有用な技術革新が阻害される

インタフェイス技術は時間をかけて除々に発展して行くという性質を持っているので，実際上，インタフェイス著作権は技術進歩を遅らせる働きをする．

十分細部まで完成されたインタフェイスが，孤独な巨匠の全身全霊を込めた傑作として作り出されることは，あまりない．すぐれたインタフェイスは，過去の試みから何かを学びとりながら，いくつかの相異なるグループの手で，繰り返し具体化が行われた結果として生まれてくるのである．たとえば，Macintosh のインタフェイスは，以前に Xerox や SRI で，そしてそれより前に SAIL（スタンフォード人工知能研究所）で試行されたアイデアにもとづいている．Xerox Star もまた SRI および SAIL からインタフェイスのアイデアを引き継いでいる．1-2-3 のインタフェイスは Visicalc その他のスプレッドシートのそれを自分なりに改良したものであり，dBase の場合は，ジェット推進研究所で開発されたプログラムに起源を持つ．

こうした進化の過程は，交響曲や小説，映画などのそれとは異なっており，むしろ民族芸術の発生に類似して

いる．われわれが奨励すべき進歩は，ほとんどの場合，すでに他のだれかが作り出した成果物に対する小さな部分的変更である．もし，それぞれのインタフェイスに著作権が認定されると，そうしたアイデアを具体化するのはむずかしくなる．たとえ，改良の対象になったインタフェイスの著作権所有者がライセンスを与えてくれるとしても，そこに至る手続きのわずらわしさと費用を考えると，よほどの人間でないかぎり，嫌気がさしてしまうだろう．

ユーザは，プログラムを使いやすく，また速くするような小さな漸進的変更を歓迎する．このことは，変更が上位互換であること，または，周知のインタフェイスのほんの一部分にしか影響しないことを意味している．そのようにして，コンピュータのキーボード上には，ふつうのタイプライタにはないファンクションキーや矢印キー，抹消キー，コントロール・キーなどが追加されたのである．しかし，もともとの文字キーの配列は変っていない．

著作権法では，このような部分的な変更は許されない．もし，新しいインタフェイスの主要な部分が，著作権化されたインタフェイスと同じであれば，その新しいインタフェイスは違法である．

第１０の反対理由：開発者もインタフェイス著作権を望んでいない

１９８９年の ACM Conference on Computer-Human Interaction において，Emory School of Law の Samuelson 教授は，ユーザ・インタフェイスの著作権問題に関する「模擬裁判」を提案し，会議の参加者（ユーザ・インタフェイスの研究者および開発者）に対して，この問題に関する即席アンケートを行った．

回答者の圧倒的多数が，ユーザ・インタフェイス著作権のすべての側面に対して反対の立場を表明した．その賛否の比率は，ある側面に関しては反対が４に対して賛成はわずか１であった．ユーザ・インタフェイス著作権がこの分野の研究にとって，有害かそれとも助けになるかという質問に対する回答は，「有害」を１，「助けになる」を５とする５段階評価で，平均１．５という結果であった [*].

---

[*] 全体の結果については CACM 誌 May 1990 を参照されたい．

ユーザ・インタフェイス著作権を提唱する人々は，それが，ユーザ・インタフェイス開発者の地位をより安全なものにし，より高い収入をもたらすだろうと，甘いことばで語っている．しかし，この調査の結果からすれば，想定された受益者たち自身が，「余計なお世話はやめてほしい」といっているのである．

あなたはほんとうにユーザ・インタフェイスの著作権を望んでいるのか？

ビジネスにとって，顧客を「封じ込める」ことは，一時的には有利である．しかし，独占的なオペレーティング・システムのベンダたちがすでに痛い目を見たように，そうした営業戦略は，結局のところ顧客からの反発を生み，オリの中からの脱走を考えさせるように仕向けることになる．長期的には失敗に通じる．

つまり，ユーザ・インタフェイスの著作権を許可することによって，われわれの社会は，ビジネス・コミュニティにおける反生産的な考え方を助長することになる．どのビジネスもそうした誘惑に抵抗できるわけではない．われわれは，そうした誘惑の芽を摘みとるべきだと考える．

結論

ユーザ・インタフェイスの独占は，ユーザにとっても役立たないし，また「科学および有用な技術の進歩を促進

する」ことにもならない．ユーザ・インタフェイスは，数年前まで議論の余地もなくそうであったように，すべての人間の共通の財産であるべきである．

あなたにできること

- ユーザ・インタフェイス訴訟の原告である Xerox, Lotus, Apple, Ashtontate などの会社と通常のビジネスをしないこと．代わりに，競争相手の製品を買うこと．これらの会社の株を売ってしまうこと．これらの会社のコンピュータ・システムではなく，ほかのコンピュータ・システムのために新しいソフトウェアを開発し，既存のアプリケーションをこれらの会社のシステムから別のシステムへ移植すること．

- これらの会社で働いたり，下請けの契約をしないこと．

- われわれの組織 LPF (the League for Programming Freedom) に参加すること．LPF は，ソフトウェア特許およびインタフェイス著作権に反対するプログラマならびにユーザの草の根組織である（LPF は，個々のプログラムの著作権には反対していない）．個人会員の年会費は，企業の従業員は 42 ドル，学生 10.5 ドル，その他は 21 ドルである．アクティブな活動家はもちろん大歓迎だが，自分の時間を割いて貢献することができない会員も歓迎する．

  LPF への連絡は，下記の宛先まで，電話または E-Mail または郵便でお願いしたい：

  League for Programming Freedom
  1 Kendall Square #143
  P.O.Box 9171
  Cambridge, MA 02139
  U.S.A
  TEL: (617) 243 - 4091
  E-Mail: league@prep.ai.mit.ed

- この論文のコピーを友人や同僚，顧客に配布すること．

- あなたがアメリカに住んでいるなら，あなたの地域選出の連邦議員および連邦議会の次の小委員会宛に手紙を書くこと．

  House Subcommittee on Intellectual Property
  2137 Rayburn Bldg
  Washington, DC 20515

  Senate Subcommittee on Patents, Trademarks and Copyrights
  United States Senate
  Washington, DC 20510

- ヨーロッパでは，EC がすでに 1 つの法令 (directive) を採用した．この法令は，ごく自然に解釈すれば，あらゆる種類のインタフェイス（プログラミング言語さえも含む）の著作権を認めるものである．EC への加盟を現在検討中の諸国もまた，この法令の影響を受ける危険がある．

  この法令を，別のもっと無害な形で解釈することも不可能ではないが，個々の EC 加盟諸国の政府が確固とした方針を打ち出さないかぎり，裁判官たちがそうした解釈に味方するとは思えない．各国政府の説得には，プログラマおよびユーザからの政治的圧力が必要である．

  この問題について働いているロビイストによれば，ヨーロッパの議員のほとんどは，コンピュータのことをよく知らず，インタフェイスの著作権がどのくらい有害であるかを理解することができない．したがって，

ヨーロッパのプログラマがなすべきことは，自分たちの選良を教育することである．

ひとつのアイデアは，あなたの議員に 1-2-3 の基本を教えることから始めることだ．コマンド言語の学習にどれだけの労力が必要かを理解してもらったあとで，実はまだ全体の十分の一しか教えていないのだと告げるのである．このやり方で，確実にわれわれの主張のポイントは伝えられるはずである．

政治的な有効性を達成するには組織が必要である．われわれの連盟 LPF は，現在，フィンランド，ドイツ，イギリス，オランダ，ノルウェイ，スイスに存在する(イギリスでは Edinburgh Computing and Social Responsibility Organization もまたこの問題を取り扱っている)．あなたの国の LPF に関する情報，または新しい組織を結成するためのアドバイスについては，アメリカの LPF まで問い合わせられたい．

編集部からアンケートのお願い

以上２つの LPF からのアピールを読んで，あなた自身はどのような感想をお持ちですか？　次の簡単なアンケートにお答えください(回答は，FAX: 03-3356-1072, または E-MAIL: sea@sea.or.jp, または郵便で SEA 事務局宛に．締切は一応 1992 年 2 月末日までとします)．

Q1. あなたは，これらのアピールに示された LPF の意見に賛同しますか？反対ですか？

(1) ソフトウェア特許権問題について

□全面的に賛同する　□部分的に賛同する　□賛同しない　□どちらともいえない

(2) ユーザ･インタフェイス著作権問題について

□全面的に賛同する　□部分的に賛同する　□賛同しない　□どちらともいえない

Q2. あなた自身またはあなたが働いている会社が，過去または現在，ソフトウェアの法的権利に関するトラブルに巻き込まれたことがありますか？

□ YES　□ NO　□知らない

Q3. 日本におけるソフトウェア特許やユーザ･インタフェイス著作権問題が，法律上どのように取り扱われているか，御存じですか？

□よく知っている　□いくらか知っている　□ほとんど知らない　□全然知らない

Q4. SEA は LPF からのこれらのアピールに答えて何らかの活動を開始すべきでしょうか？

□そうすべきだ　□もう少し状況を検討してから　□ SEA はそんな活動はすべきでない

Q5. その他，御意見またはコメントがあればお書きください (スペースが足りなければ別の紙を継ぎ足してください)．

OOPSLA'91 に参加して

塩谷 和範

(SRA)

昨年の 10 月，OOPSLA'91 に参加したさいの印象を，SEA Forum November で使った OHP 原稿をもとに，簡単に報告する．

会議のフルネームは Conference on Object-Oriented Programming Systems,Languages, and Applications，日本語に訳せば「オブジェクト指向プログラミング・システム・言語・アーキテクチャについての会議」ということになろうか？　今度で第 6 回目になる．

会議の規模その他を数字で示せば，以下の通りである：

場所	Phoenix Arizona, USA
期間	1991 年 10 月 6 日（日）から 11 日（金）までの 6 日間
公開参加者	およそ 1,700 人（内，日本人は 70 人ほど）
出展組織数	63 社/組織（出版関係など 10 社/組織）
デモ	16 件/2 日
Tutorial 数	19 件/2 日
Workshops 数	11 件/2 日
セッション数	12 件/3 日（2 セッションづつ並行に実施）
パネル数	7 件/3 日
方法論に関する並列宇宙	3 件

最初の２日がチュートリアルとワークショップ，続く３日が会議で，これと並行してデモと展示が行われた．

[全体の印象]

今年の会議参加者総数は約 1,700 人，内日本人は 70 人ほどだった．これに展示関係者および展示のみの参加者などを入れるとおよそ 2,000 人と言ったところか．昨年は公式発表があったのですが今年は公式発表なしだった．(昨年より賑っているように感じたので，もっと多いと思う．昨年に比べてチュートリアルの数 (19) とワークショップの数 (11) は同数だったが，昨年中止になったデモ (16) が復活したぶん活発になっていると思う)

[OOPSLA の変遷]

今年の OOPSLA の印象は一言でいえば，OO 分析・設計用 CASE ツールの発表だった．今までにも，いくつかツールが開発され，販売もされていたが，それらは用途が限定されていたり，単発のツールばかりだった．ようやく今年になって，プログラミング支援ツールでは無く，分析設計用の統合化ツールが現れはじめたという印象を持った．

前々回の OOPSLA'89(New Orleans) は，OO 定着の年 (OO 布教が軌道に乗り社会的に認知されはじめた年) であった．そして，前回の OOPSLA-ECOOP'90(Ottawa) が，C++ の年 (C++ 信者が他を圧倒した年) であり，分析設計段階からの OO 手法による一貫したプログラム開発の必要性 (Seamless OO Development) が説かれた年だった．

今年は，やっとプログラマーのみならず分析設計屋さんにまで OO 教が浸透しはじめた，あるいは，浸透する用意ができてきた年ということになるのだろうか？ 次には，OO 教不信心者でも OO の恩恵を被ることができるようになる兆しが現われてきた．(OO CASE tools)

[並列宇宙 (parallel universes)]

今回異色だったのは,"parallel universes"と称する 4 種類の方法論 (Methodology) に関する特別セッションが，その方法論を提唱する専門家の手によって並行して行われ，少しづつ覗き見たり，あるいはじっくり聞いたりすることができた点である．

4 つの方法論とは，

Peter Coad の OOA/OOD 手法，
Tony Wasserman の OOSD 手法，
Elite Systems 社の Norm Kerth の手法，
Wayland Systems 社の Steven Weiss の手法 (キャンセルされた)

である．

さらに，この後のパネルで，まとめが発表され比較することを意図していた (OOPSLA 事務局用のシステムを設計すると言う共通の課題が出されていた)．

残念ながらこの後のパネルでは，事務局側が意図したような方法論の比較はできなかった．3 人の報告の仕方に統一性が無く，経過のみの報告で最終的にどうデザインしたのかが報告されなかった (方法論の説明に時間がかかり最後までできなかったようですが)．会場からも失望の声が上がっていた．

[ツール展示とデモから]

[Rational Rose]

米国 Rational 社の Grady Booch が提唱する設計手法を支援する CASE ツール，X11(OSF/Motif) 上で動作し，ユニークな点は，ドキュメンターション支援のために，ハイパーテキスト機能を持った FrameMaker コンパチの DTP ツールをシステムに組み込んである．

[Demeter System]

米国 Northeastern 大学の Karl Lieberherr 教授らが開発しているオブジェクト指向開発用の CASE ツール特徴は，Demeter の法則に基づいて，クラス構造を分析し最適化し，最終的にソースコード (現在は C＋＋) を出力する統合システムである．

Demeter ソースコードをコンパイルし，実行し，テストする環境まで用意されている．デモでは，X11 Motif 上で動作するグラフィックエディタがあり，これを使って Class Dictionary Graph を作成できた．

[ObjectMaker]

米国 Mark V Systems 社の製品．統合化されてはいないが，万能の CSAE ツール．なんと 26 もの設計手法を支援するグラフィックエディターを提供している (Coad, Booch, GE-Rumbaugh らの手法および旧来の構造化手法もサポート)．

これらの手法をメタ･エディタによりカスタマイズできる点が優れている．さらに，Ada, C, C＋＋ のコードを出力することができ，逆にソースコードから，図表 (ダイアグラム) を作成する (Reverse

Engineering) ことができると主張している．ただし，各種ダイアグラム間の関係を分析したりすることはできないようだ．X11 Motif, Macintosh, MS-Windows 用の各バージョンがある．

[C++]

今年もまた沢山の C++ 用ツールが出ていた．C++ の環境も段々整備されてきて，やっと Smalltalk の域に達しはじめたようだ．一例として，既に C の開発ツールとして有名な Saber-C (Saber Software, Inc.) の C++ 版 Saver-C++ は，ブラウザー，ソースレベルデバッグ機能に加えて実行時のエラー検出ができるというのが売りでした．PPS(ParcPlace System, Inc.) の ObjectworksC++ とならんで，現在一番優れた C++ 開発環境ではないだろうか?

[PPS Smalltalk]

PPS の最新版の Objectworkslltalk Release 4(R4) は，そのマシン標準のウィンドウシステムを使うように大幅に機能を変更したため，身軽になり総合的な実行速度も向上した．不評だった仮想キーボードによる日本語入力機能が，R4 では，プラットフォームが提供する日本語変換機能により直接入力できるため，使い慣れた日本語 FEP を使うことができる．また，やっと本格的なカラーサポートが提供された．

PPS 版の特徴は，でき上がった (あるいは開発中の) イメージ (Smalltalk の中間コード) を別のプラットフォームに持って行って実行させることができる点である．

クラスライブラリも充実しているのだが，最大の不満は，価格が高いことだ．PPS のブースでは，R4 により作成したサードパーティー製の各種アプリケーションのデモが行われていた．ObjectworksC++ は，PPS Smalltalk で書かれているそうである．各種 Unix ワークステーション X-Window 版，Macintosh 版，MS-Windows 版の VM(Virtual Machine: インタプリタ + 実行環境) がある．

[Smalltalk-V]

米国 Digitalk 社の Smalltalk-V は，パーソナルコンピュータ上で，実用的な Smalltalk プログラムを作成することを目標としていて，Mac 版，DOS 版，MS-Windows 版，OS/2 PM 版がある．これの特徴は，価格が安いこと，バイナリを作り (PPS 版は，実行環境として VM が必要) 自由に配布販売できることである．

Smalltalk-V は，当初からアセンブリコーデイングができ，必要に応じて高速化ができた (PPS 版も今はできる)．これによりマシンの特性を十分に発揮できる．

欠点は，PPS の様にマシン間の互換性が無いことである．ソースレベルの互換性もない．当然 PPS 版との互換性もない．

しかし，安価で高速化できることなどにより計測機用制御システム (Tektronix 社の LAN プロトコルアナライザ)，ペン操作 PC(Momenta 社の製品) などの作成に利用され，展示実演されていた．

[Eiffel]

今回も，Eiffel が頑張っていて半日のチュートリアルがあった．Interactive Software Engineering, Inc. のブースでも Eiffel と ArchiIText というシンタックスドリブンのプログラミング/ドキュメンテーション用構造化エディタを展示していた．

ArchiIText は LDL と呼ぶメタ言語をよりその構造を指定するもので，Eiffel, Pascal, Ada, Fortran,

C のの基本文法が提供されている．

[BETA]

北欧のノルウェイからは，BETA という Simula (最初の OO 言語といわれている) の後継言語とその開発システムが展示されていた．やはり半日のチュートリアルがあり，少ないながらも言語フリークが集まっていたようだ．

[Objective-C]

The Stepstone Co. の Objective-C もあいかわらず頑張っていた．NeXT 社と共同で開発した GNU 版 Objective-C も近々 FSF (Free Software Foundation) から提供されるそうだ．NeXT 上では，Interface-Builder と NeXTstep 用のクラスライブラリ (AppKit) が使えるが，他の環境用には，Stepstone 社が提供する ICpac というクラスライブラリを使用することにより，GUI を含め簡単にプログラム開発が行える．(日本では横河 DCL が取り扱っている．)

[Modula-3]

Modula-3 は，今回初めて展示された言語で DEC 社と Olivetti 社により開発された．Modula-3 は Pascal 系の言語で，大規模ソフトウェア開発用言語として，読みやすく，使いやすくかつ高品質のオブジェクト指向言語として設計されている．

また，C++ と違い自動的ガベージコレクション・内部 Threads による並列実行の機能などを有している．多くの Unix WS に移植され X11-R4 インターフェースも提供されている．まだ PC 版は無い．やはり半日のチュートリアルがあった．

[CLOS]

CLOS も Allegro Common Lisp 開発環境 (Allegro Composer) と供に Franz Inc. が展示していた．米国では，Lisp の人気は根強いものがある．CLOS も半日のチュートリアルがあった．

[その他]

寂しいのは，いつも展示があった Actor という MS-Window 環境での OO 言語が展示されていなかったこと．(Whitewater Group) Actor は，Forth と Smalltalk のアイデアを発展させたユニークな，純粋なオブジェクト指向言語．C++ の勢いに負けてしまったのかな?

[PenOS]

Momenta 社のペンベースノートブック PC は，Smalltalk-V をプラットフォームとして，Pen-based computing を実現していた．

Go 社の PenPoint OS と同様に，ペンによるゼスチャー (ペンの動き) により，手書き文字の認識と，各種操作 (図形オブジェクトの移動，メニューの呼び出し，選択など) ができ，コンピュータアレルギーの専門家および一般ユーザなどにとって，使いやすい環境を実現していた．(基本的にはマウスをペンに置き換えただけの Pen-Windows よりも軽快で使いやすそうでした．やはりジェスチャー操作を取り入れてはいますが．でも，MS-DOS, MS-Windows をこれだけ普及させたマイクロソフト社の力は侮れません．)

Go 社の PenPoint OS を実装した，試作機のデモもあった．こちらは，システム手帳メタファーを GUI に取り入れていて，手帳感覚で使用できる．ペンによるゼスチャーで操作する．

Go 社の試作機を試用した．残念ながら，手書き文字認識 (現在は，アルファベットのみ) は，アメリカ式の書き順でないと正しく認識できない．私の悪筆のせいも多分にあるが，正しくきれいに書いたつもりでも，認識されない文字があった．ローマ字と書き順が違う．それを指摘したら，欧州各国とも若干書き順が違うため正しく認識されない．各国語用の認識オプションを提供する予定だと言っていた．

[OO-OS/OO-Framework]

Momenta Pen PC は，Smalltalk-V をプラットフォームにし，その上に MADE(The Momenta Application Development Environment) というオブジェクト指向の開発実行環境 (Framework) を作り，Momenta Pen PC 上で動作するプログラムの開発環境を提供している．

また，Go 社の PenPoint OS は C により作成されているが，専用のオブジェクト指向マルチタスク OS を実現していて，250 のクラスを提供している．

PenPoint OS 上のアプリケーションは，Smalltalk 上のアプリケーションなどと同様に PenPoint OS のサブクラスとして作成され，オブジェクトとして実行される．また Go 社の試作機では，持って歩くことを想定しているため，ハードディスク，フロッピーディスクなどの壊れやすい機械部分を内蔵せず，すべて RAM 上で動作する．(実際，デモではいきなり手から落すことから始まった．デモ担当は，これが一番楽しいと言っていた.)

そのため，常時使わないアプリケーションのシステムへの組み込み，取りはずし，消去などの機能を備えている．

プログラム・データの登録・保存はどうするのだろうと言う疑問がわくが，この試作機では外付けの FDD を試用するか，通信機能によりファイル転送する．既に，AppleTalk(LocalTalk) インタフェースを用意していて，レーザライタに出力したり，ファイルサーバにアクセスしたりできるようになるそうだ．もちろん，IC カードなどを使うこともできるようになるのだろう．

赤外線リンク・無線リンクなども開発中だそうだ．Go 社は，PenPoint OS をライセンスしているだけなので，いろいろなハード構成の PenOS PC が発売されるだろう．

[OODBMS]

今回も OODB が多数出展されていた．今回は，いずれも GUI に特徴をだし，手軽に GUI を構築できることを売り物にしていた．

特に，ITASCA は以前 MCC で開発されていた ORION の後継で，分散環境に対応している(ほとんどがそうだが)．また，C, C++, Lisp から直接 DBMS を操作でき，Fortran コードの動的なローディングもできる．さらに，スキーマエディタを通して動的スキーマ変更ができ使いやすい．貰った資料に代表的な OODBMS 5 製品との比較があり，その優位性を強調していた．

O2 は，フランスの会社の新製品で，O2Engine, O2C, O2Look, O2Tools, O2Query から構成され，それぞれデータベース，4GL，GUI ツール，プログラミング環境，SQL 型問い合わせ機能を提供する総合的な OODBMS である．

[その他展示から]

Macintosh 上で動く TurboCASE 4.0 がなかなか良さそうだった．これは，賞を貰ったツールとかで，Multi-View で Object 間の関係とかインスタンス変数，Method など親クラスから継承されている状態が

明示的に表示でき便利そうだった.

[Panel および Session]

先ごろ，日本でセミナーを行った Ivar Jacobsen の ObjectOry の発表と展示があった．Reverse Engineering により既存の大型システムを，段階的に改良して行くための方法論と CASE ツールを発表していた.

いつも展示・デモなどを中心に見てくるので，聞かなかったせっしょんが結構あります．今回は時間がないので次回機会があれば報告します.

[資料入手先など]

Modula-3:

バージョン 1.6 のソースコード及びドキュメントが
gatekeeper.dec.com (pub/Dec/Modula-3) から ftp で入手できる.
(サポートなし)

BETA:

pre-release 版を ftp するために，support@mjolner.dk に問い合わせる.
Mjolner Informatics ApS
Science Park Aarhus, Gustav Wiedsvej 10, DK-8000 Aarhus C, DENMARK
Fax: +45-86-20-12-22, phone: +45-86-20-20-00, E-mail: mjolner@mjolner.dk

Demeter:

X 対応前のデモ版（Sun3/4 用バイナリ）が同大学から ftp で入手できる.
(129.10.8.150 pub/demeter)

TurboCASE 4.0:

デモ版（MAC 用バイナリ）が下記住所から $15 で入手できる.
StructSoft, Inc.
5416 156th Ave.S.E., Bellevue, WA 98006, U.S.A.
Fax: +1-206-644-7714, phone: +1-206-644-9834

Rational ROSE:

Rational.
3320 Scott Boulevard, Santa Clara, CA 95054-3197, U.S.A.
phone: +1-408-496-3700

PenPoint OS:

PenPoint OS の入門書："The Power of PenPoint" Addison-Wesley

ITASCA:

ITASCA Systems, Inc.
2850 Metro Drive, Suite 300, Minneapolis, Minesotta 55425, U.S.A.
Fax: +1-612-851-3157, phone: +1-612-851-3155

O2:

O2 Technology.
7, rue du Parc de Clagny, 78000 Versailles, France
Fax: +33-1-30-84-77-90, phone: +33-1-30-84-77-77

ACM の連絡先:

Association for Computing Machinery
11 West 42nd Street, New York, NY 10036, U.S.A.
phone: +1-212-869-7440, Fax: +1-212-944-1318

[参考文献]

[OOD: Booch] Grady Booch.
Object Oriented Design with Applications.
Benjamin/Cummings, Redwood City, California, 1991

[OOA: Coad/Youdon] Peter Coad and Edward Youdon.
Object-Oriented Analysis (second edition).
Youdon Press, 1990,1991.

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

[OO-CASE: ObjectOry] Ivar Jacobsen, Fredrik Lindstrom.
Re-engineering of old systems to an Object-Oriented architecture.
In OOPSLA'91 Proceedings, pages 340-350, Oct 1991 ACM.

Ivar Jacobsen.
Object-Oriented Software Engineering.
ACM Press, Addison Wesley, 1992

[OO-CASE: Demeter] Karl J. Lieberherr, Paul Bergstein, and Ignacio Silva-Lepe.
From Objects to classes: algorithms for optimal object-oriented design.
Software Engineering Journal, 1991. (Accepted for publication)

[OOD: Responsibility Driven Design] R.Wirfs-Brock, B.Wilkerson, L.Wiener.
Designing Object-Oriented Software.
Prentice Hall, 1990.

[OOD & Eiffel] Bertrand Meyer.
Object-Oriented Software Construction.
Prentice Hall, 1988.

Bertrand Meyer 著，二木厚吉監訳，酒匂寛，酒匂順子訳
オブジェクト指向入門
ASCII 出版，1990

[C++] B.Stroustrup.

The C++ Programming Language. Second Edition
Addison Wesley, 1986,1991.

[Objective-C] Lewis J.Pinson, Richard S.Wiener.
Objective-C Object-Oriented Programming Techniques.
Addison Wesley, 1986.

[Modula-3] Greg Nelson.
System Programming with Modula-3.
Prentice Hall, 1992.

[CLOS] Jo A.Lawless, Molly M.Miller.
Understanding Common LISP Object System.
Prentice Hall, 1991.

[参考までに]

OOPSLA は，アメリカを中心とした最も古い一番活発なオブジェクト指向の会議です．Oregon 州の Portland で第 1 回が行われ，今年が 6 回目です．通常毎年 10 月初旬に開催されてます．

これに対して，ECOOP (European Conference on Object-Oriented Programming) という欧州を中心とした理論中心の学会もあります．昨年は，OOPSLA-ECOOP'90 という合同の会議がカナダの首都オタワで開かれました．また，最近では Tools-{USA,Europe,Pacific} という名前から想像すると実用面に主眼をおいた会議（？）も米国，欧州，太平洋諸国のそれぞれで開かれています．

これらの中で OOPSLA は，理論と実践のバランスがとれ，展示も多く参加者も多いため最も参加する価値のある会議だと思います．

私のプロジェクト・マネジメント論

— On the Guiding Principles for Project Managers —

勝田 祐輔
（日本ユニシス）

1. 序論

システム開発のための種々の手法・ツールが案出され，かつ提唱されてきている．

ところで，システム開発において，所期の目的を達成させるためには，個々の手法・ツールが使えるという単なる技術知識だけでよいものだろうか？

個々の手法・ツールがいかに優れていても，それらを「使う側の問題」があって，それ如何ではせっかくの手法も有効とはならない場合がある．

技術知識は必要であるが決して十分ではない．その手法の考え方を理解していること，その手法の限界を認識していること，問題に対して正しく適用できること，変に応じて用いる能力といったものがあってこそ，優れた技術も真に活かすことが可能となるのではないだろうか？ [*1]

本稿はプロジェクト・マネジメントの手法の詳細を論ずることを目的としたものではない．

私は本稿において，プロジェクト・マネジャーが，どんな考え方で，どのような点に留意しながらマネジメントを行なうべきかについて，なにか普遍的ないわばプロジェクト・マネジャーの指導原理のようなものを示すことができたらと思う．

私達の周囲で日頃発生している問題を眺めると，それらの殆んどのものは，テクニカルな問題として起こっている．私達の周囲のオペレーションの多くがテクニカルなものを対象として行なわれていることからして，この事は自然な現象と言える．

しかしながら，少し注意してみると，現象としてはテクニカルな問題としてあらわれている事柄も，そのよって来る源，従って真の問題のありかは別の所にあることが指摘できることも少なくない．こうした観察から，本稿において，プロジェクト・マネジャーの指導原理というべきものを論じることにした次第である [*2]．

第一部では，プロジェクト・マネジャーが基づくべき考え方を，『孫子』の思想を基にして論じ，第二部においては，第一部の万全主義ともいうべき考え方に基づき，具体的・実践的な定式化として提唱している．

2. 本論

[第一部]

『孫子』は中国の最古でかつ最もすぐれた兵書であると言われている．数年前，私はあるリアルタイム・システム開発のプロジェクト・マネジャーを命ぜられた．この時のケースは，それ以前私が携わったいくつかのケースとは別の難しい条件を含んでいるものであった．特にユーザーとの関連において，この成功は極めて重要事とされていたのであった．その条件下で，一刻を争う本番開始，しかも絶対安全な開始が至上命令であった．この仕事をどのように進めてゆくべきか私が日夜思案していた時，ふと手にしたのがこの『孫子』であった．読み進めるにつれて，私がいかなる考え方に基づきこの仕事を進めてゆくべきか確信を持つに至った．

私はこのリアルタイム・システム開発を戦争と捉えた．速やかにして安全な本番開始がすなわち勝利であった．いかにして，あやうげない勝利を得るか，これが私のテーマであった．

戦争における将軍の役割は，「人的資源と物的資源を，ダイナミックな組織体に結集させて顧客への満足と部下へは高いモラルと喜びを与えつつ，組織体の目的を達成させる行為」というマネジメントの定義 [*3] そのものである．

春秋時代（紀元前 8 世紀～前 5 世紀）から戦国時代（紀元前 5 世紀～前 3 世紀）における戦争の規模は，史記によると将兵数万の規模から数十万に達する規模のものがあった

[*1] このことは，科学技術が高度に発達しつつある現代において，使う側の哲学が一層必要とされていることに類似していると思う．

[*2] 個々の手法は自ら限界があるものである．原理というものは広く応用のできるべきものと考え，本稿でもそれをめざしている．

[*3] アメリカ経営者協会：L.A. アプリー「マネジメントの本質」より

[*4]. その人的資源，数百里から数千里の行軍をささえる兵糧，武器，戦車，輜車等の物的資源は厖大である．これらすべてを指揮統率し，兵の鋭気を保持し，緻密で変に応じた戦略をたて，味方を勝利に導くという将軍の役割は，マネジメントの最たるものと言える．しかも敗戦に至った場合，すなわち国家存亡，国民の死活にかかわる結果の重大さは，他の比ではないであろう．

『孫子』は，このような熾烈な実体験と，精密な現実観察に基づいて，その上に深い思索を加えて産まれたものといわれる [*5].

そこには，極めて科学的な態度，合理主義，そして徹底した現実主義が貫かれていて，単なる戦争技術以上のものがある．

さて以下この第一部では，プロジェクト·マネジメントの考え方と，『孫子』の思想との類似性に着目して，プロジェクト·マネジメントにおける『孫子』の思想の活用について述べる．

(1) 計画について

計画立案の重要性は，誰しもが認めるところであると思う．しかし，観念的にはそう理解していても，真にその重要性を知る人は，自身で体験した人，すなわち自分で計画を立ててその結果，よい計画の恩恵を実際に味わった人ではないかと思う．

システム開発のプロセスとは，漠然としている未知の時間空間を秩序だててゆく明確化のプロセスである．従って計画立案の時点で，何を目標にしてどのように進めてゆくべきか，途中のプロセスを仔細に考察する．他人（参謀）の知恵も十分に活用する．そして計画ができれば，その計画をたよりにして，未来にむかって進んでゆくのである．もし計画を軽視して，ないしは形だけの計画で，事がうまく成就した例があるとするなら，それはたまたま幸運に恵まれたのであるか，あるいはその目標がとるに足らぬものであったかのいずれかであろう．しかし幸運はそういつも続くものでない．プロジェクト·マネジャーは，華々しくはなくとも目的はつねに達せさせることが要請されるのである [*6].

結果の如何が重要であればある程，計画段階において十分に考察することが必要である．

『孫子』においては，その冒頭の一節で，事を起こす前に，よくよく熟慮するべきことを説いている [*7]. 不十分な計画で軽々しく事を起こすというのは，孫子のとらないところであった．

(2) 作戦

戦車千台，数十万の将兵といかぬまでも，一旦プロジェクトを発足させ，事を起こしたからには，思惑と違って長びき遅延することは避けなければならない．遅延し長びいてゆけば費用も大きくなってゆき [*8]，プロジェクト·メンバーの士気にもかかわる．

まずくともすばやくやるというのはあるが，うまくて長びくという例はまだないと，孫子は教えている [*9].

システム開発において，完成遅延により，システムの価値の減少，甚だしくは開発自体意味のないものになってしまうといった場合もでてくる．

現場の要求は，例えば次のようなものである．「できるだけ早く新システムの恩恵をこうむりたい．現状ではもう我慢ならない．一刻も早く完成せよ」

こうしたものであるから，このいわゆる拙速主義を，スケジュール作成，その実行段階において，プロジェクト·マネジャーは念頭に置いているべきである．すなわち，計画は時間をかけて熟慮して，実行はすばやくというわけである．

(3) やさしい道を考えること

プロジェクトの目的は，プロセスを見せることではないのであるから，同じ目的を達するにしても，やさしい道を考えること．

『孫子』は，戦わずして勝つことの要道を述べている [*10].

[*4] 『孫子』は，春秋時代の後期ないし戦国時代に成立したといわれている．

[*5] なお，かのナポレオンが『孫子』を座右の書としていたことは，有名なこととされている（徳間書店：『孫子·呉子』）．また，第一次欧州大戦をひきおこしたドイツ皇帝ウィルヘルム二世は，敗れたのち『孫子』を知り，「20年前にこの書物を読んでいたら....」と述懐したという（同）．『孫子』では，将の役割を特に重視する立場で貫かれている．

[*6] いわゆる「百戦して殆うからず」（謀攻篇第三，後述）．なお，このために本質的に重要と思える方法については，第二部にて述べる．

[*7] 「兵とは国の大事なり．死生の地，存亡の道，察せざるべからざるなり....」（計篇第一）．ここでも将の役割の重要性を示唆している．

[*8] 例えば，マンパワー，M/Cタイム，機器，消耗品等数名·数ヶ月を考えると，すぐ数百万になる．

[*9] 「兵は拙速なるを聞くも未だ巧久なるをみざるなり．それ兵々しくして国の利する者は未だこれ有らざるなり」（作戦篇第二）．

この言葉は，問題をやさしい方法で解決するという技術的な面での意思決定においてしかりであるが，システム開発途上，他部門との折衝においても，いたずらに無用の摩擦を起こさないで，円満に協力をとりつけるという点でも示唆するものと思う．

ただし，ここでいう他部門とは，なにも敵でもないし，屈せさせるというようなことではないのであって，プロジェクト・マネジャーの真剣な取り組みや熱意が根底にあって，しかも相手の立場を考えた上でのことでなければ，真の協力はあり得ようわけはない．

(4) 彼れを知り己れを知ること [*11]

システム開発において，彼れとはこれから為し遂げようとする仕事の質・量であり，起こるであろういくつものハプニングである．未確認のソフトウェア・エレメントの発生するかも知れぬバグ，ハードウェアのトラブル，機器確保・M/C タイムにかかわる問題，端末／回線にまつわる種々の問題，ユーザー開発部門の体制・能力，オペレーション部門の体制・質および開発部門との関連，ならびにエンド・ユーザーにからむ問題等々のような物的・人的な問題が考えられる．

己れとは何か？ プロジェクト・マネジャーの力量も含めて，プロジェクト・メンバーのスキル・パワー，周囲の協力体制等である．己れの実情を正しく把握し，やれることとやれないこととをはっきり認識して計画を立案し，それに基づいて推進してゆくことがプロジェクト・マネジャーの基本動作である．明らかに実現不可能なスケジュール，裏付けのない線引き（線を引いただけ）があるとすれば，「戦う毎に必ず殆うし」ということになると思う．

(5) 主導性

戦争において主導権を確保することの重要性を『孫子』は述べている [*12]．

対ユーザー，対エンド・ユーザーとの関連において，さらにプロジェクトへの協力体制の中においても，対スタッフとの関連においても主導性を発揮することは肝要である．

受動的立場をとらないこと，先読みしたオペレーションは，情勢を正しく把握し，迅速に正しく処置してゆくことによって得られるものであろう．

(6) 十分な備え

プロジェクト発足からその終りまでの間に，予期せぬ問題にぶつからないなどということは，殆んど考えられない．例えば，システム・セットアップ，端末の作動確認，単体テスト，結合テスト，ラッシュ・テスト，リカバリ・テスト，総合テスト，本番稼働直後，それぞれの段階で大なり小なり問題に出合うものだ．

このように，問題は必ず起きるものと考えるべきである．問題が起こらないことを願うのではなく，いつどんな問題が起ころうと，それに備えておくべきなのである [*13]．すなわちプロジェクト・マネジャーは，このあとどんな問題が発生しうるか，何が手薄になっているか [*14]，どんな手が考えられまた手を打っておくべきかと常に注意しているべきである [*15]．さらに，本番稼働後，新たな問題を発見し，それの影響・対処に憂慮することを思えば，稼働前に一つでも多くの問題を発見すること，問題を発見し得たことを歓迎すべきである．

(7) 本番直前

本番を直前に控えて，プロジェクト・メンバー全員が張り詰めた気持ちでいる時，このような時再び次々と問題が発生してくる場合がある．ここが勝負どころ，自分自身にも他のメンバーにも最も厳しい考え方で臨むべきという，緩むべきでない時が，システム開発のプロセスの中であるものである．こうした時のために，体力を蓄えておき，そして一挙に（急速に）収斂させるのである．これは，『孫子』における，勢の考え方に通じるものと思う [*16]．

さて，プロジェクト・マネジャーの最後の課題は，いつ本番を決行するかということである．プロジェクト・マネジャーは，様々のことを想定して，種々考えあぐむことであろう．ここまで命をかけてきた彼の最後の決戦（仕上げ）で

[*10]「百戦百勝は善の善なる者に非ず．戦わずして人の兵を屈するは善の善なる者なり」（謀攻篇第三）．

[*11]「彼れを知りて己れを知れば百戦して殆うからず．彼れを知らずして己れを知れば一勝一負す．彼れを知らず己れを知らざれば戦う毎に必ず殆うし」（謀攻篇第三）．

[*12]「善く戦う者は人を致して人に致されず」（虚実篇第六）．

[*13]「その来たらざるを恃むことなく，吾れの以って待つあるを恃みとす」（九変篇第八）．

[*14]「手薄なところが討ち取られる」この命題も真理である．

[*15] 例えば：体制の敷き方，連絡ルート，二重化の対策等．

[*16]「善く人を戦わしむるの勢い，円石を千仞の山に転ずるが如くなる者は勢なり」（勢篇第五）．

ある．むざむざ失敗することはできない．

本番直前のプロジェクト·マネジャーの指導原理として，『孫子』の言葉を挙げておく．

「勝兵は先ず勝ちて而る後に戦いを求め，敗兵は先ず戦いて而る後に勝を求む」[*17]

このようにして，プロジェクト·マネジャーは勝利すべきである．

第一部のしめくくりとして，プロジェクト·マネジャーの信念ともいうべきものについて述べる．

プロジェクト·マネジャーが，真に責任を負うのであれば，変なことに遠慮すべきでない．自分の信念に基づいてやれないでいて，どうして責任を負えようかということである [*18]．真に熟慮した上では，このぐらいの気魄でもって，プロジェクトを統率してゆくべきである．

第二部

ここで述べるプロジェクト·マネジャーのより実務的な原則は，私の過去の実践結果を整理して，最も重要で基本的であると思われる要素を選び出ししたものである．従って，それらは皆現実の生々しいオペレーションに対応したものである．

それでは，現実に課せられた問題に対して，どのように計画を立て，実行してゆくべきであるか？　私はここに，8つの基本的な原理ともいうべき手順を提唱し，それらを名付けて，Guiding Principles (GP-1 から GP-8) と呼ぶこととする．

個々の原則は簡単なものである．それら 8 つを統一的に用いることにより効果を発揮することができるものである．『孫子』は，次のように言っている．「勝は知るべし，而して為すべからず」[*19]．すなわち，勝つことを知ることはできても，それを必ずなしとげるわけにはいかない，と．勝つことは難しい．それには，いろいろの要因が微妙にからみあっている．

ところで私が述べる原理が，勝に至る要道であることを示すには，どうすればよいだろうか？　証明なしで，天からの啓示として与えるのは預言者のすることである．では，「私はこの原理に則って完全な成功を収めました」と言って，一，ニの実例を与えることで証明となるだろうか？　勝に至った微妙な因果関係を立証することは難しいし，これでもって一般性を主張できるとも思えない．これに反して敗北の要因は立証し易い．

この原則を否定すれば，兵法に照らして，勝が少なくともあぶなかしことは示せる．また証明はできなくとも，個々の原則について，理由·根拠を示し，一般的な説得力を持たせるように努めることはできる．私はこのような方法をとることにした．

しかしながら，この原則は「必要かつ十分」なものであろうか？　私はすべてのケースにおける，「必要かつ十分」な原則があろうとは思わない．私は「必要」な主要な原則をうち立てたいと思っている．そして，個々のケースにおいて「必要」を補足し，より「十分」に近づけて実施することは，原則を応用するプロジェクト·マネジャーの役割であると考える．

さて，計画立案のステップは，AMA のテキストで述べられている通り [*20]，次の基本ステップに従うことになる．

1. 目的の定義 [*21]
2. 作業手順の決定
   - What (どれだけの作業)
   - How
   - Where
   - When (いつまでに)
3. 責任の分担

以上のような前置きをした上で，さらに計画立案における「目的の定義」がされたとして，第二部の主題である GP-1 から GP-8 について述べる．それらは，すでに述べた「明確化のプロセス」ということから示唆できるものである．

「明確化のプロセス」の観点から考えられる主要な事項

[*17] 形篇第四．

[*18] 司馬遷『史記』列伝の部の孫武伝に，「臣既に已に命を受けて将と為る．将，軍に在りては君名も受けざる所在り」とある [参考文献 (3)]．

[*19] 形篇第四．

[*20] アメリカ経営協会：P.M ストークス「計画立案」．

[*21] テキストでは「目的の明確化」と訳されているが，「明確化」という言葉より「定義」とした方が日本語としてより明確な印象を受けるので，ここでは「定義」とした．
目的設定を誤らないことが，計画立案における最初の留意点である．また目的があいまいなままでスタートしないこと．例えば，会議一つ開くにしても，途中，一体この会議は何を目的にしているかわからないといった事を経験したことがあるが，これも冒頭でその会議の主旨を説明し，何がどこまで審議できれば，その会議の目的が達せられたことになるかを出席者全員に理解させてから始めるべきである．

は，概ね次の項目で網羅できると考える．

(1) 全体の仕事
(2) 戦うための布陣（体制）
(3) 必要な物とその手配
(4) スケジュールと担当者
(5) 道標とチェック･ポイント
(6) 進捗の管理
(7) 問題点と対処策
(8) 計画の提示とそのオーソライズ (Authorize)

私が主張する General Principles とは，これら普通に考えて当然と思える事柄に対応している．それらを断片的に採用するのではなく，全体の考え方を理解して系統的に応用することにより，確実に効果を出すことができると考えている．

[GP-1] プロジェクト･マネジャーのアサインから始まって，プロジェクトの終りまでを慮り，あらゆるアクティビティを想定して，それらの前後関係･併行可能性を考慮しつつ，ネットワークの図にする．

[詳細説明]: PERT の手法でよく知られたこの図は，計画立案時において，全体の作業を見極め，それらの相互関係･順序関係を明らかにし，プロジェクト･マネジャーのアプローチの仕方を決めてゆくために用いる．このネットワーク図をよく検討し吟味することにより，プロジェクトの進行過程がより具体的なイメージとして捉えることができる．吟味を加え終ったならば，アクティビティを表す○印の中には，日付を入れないで，連番を入れておく．このネットワーク図は，最後まで不動のものとするべき（その程度に吟味すべき）ものである．

いま述べたのは，この後 PERT 手法を使ってゆくというわけでなく，この図を根底に据えて，作業量の見積もり，スケジュールの作成，M/C タイムや機器の手配，要員のアサイン，教育計画，回線申請，用紙の発注，マイルストーンの設定，チェックポイントの設定等々をプロジェクト･マネジャーが考えるために，そのために用いるのが目的である [*22].

次に個々の作業に作業量（人･日ないし N 時間× M 回）を考察して [*23]，アクティビティ間を結ぶ線の上に書きこんでゆく．こうして線の長短をアジャストして，ネットワーク図を完成させる．

根拠／理由: なすべき仕事の全体をよく知らないで，または整理しないで，突き進むのは，「彼れを知らない」やり方である．全体を整理しないでとにかく作業を始めるようでは，モレやヌケが出ても不思議な事ではない．さらに，作業量の大きさを定量的に把握する事は，より客観的に「彼れを知る」ことになる．定量的把握の大切なことは，すでに『孫子』で述べられている [*24].

このように，すべての作業，順序，クリティカル･パス等をわかり易い形に整理することは，よりよく「彼れを知る」ための基本的作業であると言える．

[GP-2] 上記ネットワーク図をもとにして，目的を達成させるためにとるべき体制（布陣）について考え，責任体制図の形にまとめる．

[詳細説明] プロジェクトの目的を達成するためには，通常いろいろな部門が関連するものである．そうした関連部門には，何を期待するかをはっきりさせる．そうして，期待したことを相手（関連部門）に，確かに受け止めてもらわなければならない．

さらに，関連部門とのレポーティング，すなわち誰が，どこに，どういうことをレポートするかもはっきりさせる．これら責任分担を定義した全体体制図を完成させる．

根拠／理由: 目的を達成させる上で，プロジェクト／関連部門の役割／責任が不明確で，依頼したことが守られなければ，たちまち支障をきたすことは明白である．

また複数の人間，複数の部門が Synchronize してゆく上で，情報の流れがうまくいっていないことが，トラブルの原因になっている例は多い．従って最初にレ

[*22] なお各担当者に全体計画と個々の作業の位置づけを理解させるためにも使うことができる．

[*23] これまでの実績値を使う．私のもとの上司に指摘されたが，「経験の蓄積」が重要である．従って，プロジェクトが終了し，次のプロジェクトをよりよく成功させたければ，実績を整理しておく必要がある．これらデータの貴重さは，（科学工学や機械工学におけるデータの貴重さと同様で---これらの分野では極めて重視されよく整理されている－実際に使って恩恵を受けないとわからないかも知れない．プロジェクト･マネジメントの分野では，まだ十分に系統的には整理されてはいない．これだけで一つの論文のテーマとなるものであろう．

[*24] 「兵法は一に曰く度，二に曰く量，三に曰く数，四に曰く称，五に曰く勝」（形篇第四）．

ポーティングについて決めておくことが，つまらない無用のトラブル発生を防ぐことになる．

図にすること [*25] の効用は，先ず表現しやすいこと，直観的で理解しやすいこと，簡潔なこと，表現上のあいまいさが少なくなること等が挙げられる．

[GP-3] 必要機器，設備，M/C タイム，テスト用データや本番，データの準備等，プロジェクトの目的達成に必要なものの準備を，誰が，いつまでに，何を，どれだけ (min,max) と詳細に具体的に整理し，責任分担表にまとめる．

[詳細説明] 機器，その他どれが欠けても計画遂行上障害となるのであるから，このような事柄を（その手配に）余裕がある間に，手落ちのないようにしておくことは，極めて重要である．

根拠／理由： 戦いにおける戦車，兵器，その他の物資の調達・補給は，死命を制する最重要時の一つであって，『孫子』においてもいろいろと論じられている（その結果「拙速主義」が主張されているのである）．

我々の場合においても，これなしでは戦えないのである．このように兵法になぞらえて考えれば，物の確保に，Severe に意を用いることの重要性はより明白となるであろう．

[GP-4] GP-1 の図をもとにして，アクティビティ，作業量 [*26]，開始日，終了日を盛りこんだスケジュール表を作成する（スケジュール表作成と小マイルストーン設定）．

[詳細説明] 個々のアクティビティに対して，プロジェクト・マネジャーが担当者のアサインを考える時，たいていの場合スキル・ギャップの問題にぶつかるものである．必要なスキルを持った人間をプロジェクト・メンバーとして，すべて集められるということは，プロジェクト・マネジャーの努力にもかかわらず，まずあり得ないことである [*27]．必要とするスキルなしにプロジェクトを成功させることは，無理な話である．そういうわけで，プロジェクト・マネジャーが先ず最初に苦心することは，どのようにして短期間の内に，プロジェクト・メンバーに必要なスキルを習得させるかという事になる．これは初期の重要な事柄の一つである．このためには，プロジェクト・メンバーの各人について，よりよく知っておく必要があり，スキル習得には最も効果的な，時には思い切った方法を案出すべきである [*28]．スケジュール表には当然，このための期間も盛り込むべきである．所詮，実現ベースでスケジュールをひくのであって，明らかに不可能な線をひいてもしょうがないのである．

しかしながら，最初にひいたスケジュールは，現実にはいろいろ差異がでる．あるものは遅れ，あるものは進みといった具合である．すべてがそうであると，完成予定日は誠に危なっかしいものとなる．であるから，全体を睨んで，細部はいろいろ変化しても，これだけは，この日にはぜひ終了していなければならないという，いわば不動点を設定してゆくのである．これらの不動点を，プロジェクト内の努力目標とする．これがスリップしそうになった時，どういう策をこうじるかは，スケジュールをひく時に考えておくべきである [*29]．

スケジュール表は，不動点以外は，途中書き替えられていってもよいものであると思う．ここでいう不動点は，いわゆる「小マイルストーン」と呼ぶことができよう．

根拠／理由： GP-1 から GP-3 に比してこの GP-4 では，特に「己れを知る」ことが要請される．スケジュール表は彼我を知らないとできないからである．

各担当者が何をいつまでになしとげるかを先ず定義し，プロジェクト全体としては，小マイルストーンを定義する．これも時間的に節目をつける明確化のプロセスである．これなしでは，始めての長い道程をゆくのに，地図も持たずに，定刻に着くことを願うようなものであって，急がないといけないのか普通でよいのか誠に心もとないことになるといったような類である．

[*25] これは私の以前の上司に指摘されたことである．

[*26] 標準的な値からさらにその担当者ならどのくらいかかるか，個人差による修正 (±) を加えた値で表わす作業の難易度等も書き込んでおくとよい．

[*27] むしろプロジェクト・マネジャーは，スキル・ギャップが必ずあるとの前提に立つべきである．現実はこの通りであった．

[*28] これをおろそかにすると後々影響が大きくなってゆく．しかしダラダラやっても仕方がないので，拙速主義でゆくべき．

[*29] 例えば，多少の余裕を見込む等々いろいろ考えられる．

また，ここで強調しているスキル·ギャップの問題は，武器をよく扱えずに戦争することの無謀さに比せば，納得できるものと思う．スキル·ギャップを冷静に把握すれば，その結果いろいろと対策を講じることができるのだ．

[GP-5] プロジェクト内の作業および関連部門への依頼事項に関し，チェックポイントを設定する．

[詳細説明] プロジェクト·マネジャーおよび上位マネジメントは，このチェックポイントをおさえてゆくことになる．設定した各チェックポイントには，番号，日付，期待結果，責任部所を記入し，一覧表にしておく．チェックポイントで，期待した結果が満足されていなければ（それが無視できる程度でないならば），Extra Efforts を払わなければならない [*30]．さもないとスケジュールは遅延する．もしチェックポイントを無事通過しないで，スキップすると，計画した品質を得ることができなくなり，後にトラブルと混乱を引き起こすことになる．

プロジェクト内の作業に関するチェックポイントは，GP-4 に対応させて大マイルストーンと呼ぶことができる．チェックポイントにおける報告は，単なるレポートによる場合と，関連部門の参集を依頼して報告しなければならない場合もあるであろう．

根拠／理由： すべて計画通りに進むというのは難しいものであるから，途中の軌道修正，重大な上位マネジメントの Decision が必要になることもあるし，関連部門へ改めて要請を出さなければならないことも出てこよう．

あらかじめ節目を設けておいて，計画的に節目節目でチェックして，進捗の確認·問題点の把握を行ない，もし対策を講じるにしても手遅れにならないように，というのがこの考え方である．手遅れになると，うつ手が限られてくる，費用が余分にかかる，犠牲が大きくなる，使った金が活きない.... 等良いことは少ない [*31]．

[GP-6] プロジェクト内の進捗管理，担当者がかかえている問題の内，プロジェクト·マネジャーがフォローすべき問題点の把握，およびそれらのコントロールのための手段を設定する．

[詳細説明] 1 ～ 2 週間単位が適当であろう [*32]．ただし，初めて一緒に仕事をするような場合は，はじめの内はより木目細かくチェックしてゆくことが必要なこともある．進捗のチェックは，個人差を理解して，それに応じて行なうべきであって，例えば，うまく進行していないことをどうしても報告できない性格の人もいないわけではないということを指摘しておく．こうしたケースでは，担当者の報告だけで済ましていると，いよいよという直前になって事態に気付くということになる [*33]．従って進捗管理において，プロジェクト·マネジャーは個人差を考え，実際に仕事を（自分で）確かめることを基本とすべきである．さらに付け加えると，一旦担当者に割り当てた仕事は，何とかその担当者が自分で成し遂げるようにもっていってやるよう忍耐と苦心をするべきである．

根拠／理由： プロジェクト発足の最初にこうした手段を設定し，全員にその手段による報告の義務を負わせ，報告も重要な任務であることを認識させる [*34]．これは，現実の正しい把握，認識がプロジェクトを制御していく上でのベースとなるからである．『孫子』行軍篇にも，現実観察の方法が細々と説かれており，その重要性を示唆している．

[GP-7] 既にわかっている問題点や，起こるかも知れない問題を設定し，それらの問題点についていくつかの対策を検討する．そしてそれらを整理して問題点と解決案として表にまとめる．

[詳細説明] （複数の）解決案について，それぞれの長所·短所を，おとすことなくできるだけ客観的·具体的に書く [*35]．総合的には，どの解決策をとるべきか，プロジェクト·マネジャーとしての見解は打ち出しておく．

根拠／理由： これも「彼れを知る」ための努力であり，早め

[*30] プロジェクト·マネジャーの努力目標は，チェックポイントをほぼ満足のゆく状態で通過することである．Extra Efforts には，上位マネジメントの Decision が必要になる場合もあろう．

[*31] いわゆる「消火型オペレーション」と「防火型オペレーション」の差．

[*32] 私は週単位で実施してきた．

[*33] 私が駆け出しのマネジャーであった頃の失敗．以下は，それによって得た教訓．

[*34] 通常，報告は，紙に書いたものをもとにする．それにより，コミュニケーションの正確さを期す．

[*35] 誰がどのように困るのか現実ベースで書く．

に問題点の全貌を明らかにして，それに備えることになる．『孫子』虚実篇に曰わく，「凡そ先きに戦地に処りて敵を待つ者はいつし（＝楽である），後れて戦地に処りて戦いにおもむく者は労す」

[GP-8] GP-2 から GP-7 の内容を盛り込んで計画案とし [*36]，上位マネジメント，関連部所，Technical People に対して提示する．

[詳細説明] 提示の方法として，これらの人々に集まってもらうことが最も望ましい．

このプロジェクト・レビュー・ミーティングで，プロジェクトの計画全体を，十分に理解してもらう．その上で各部所に対する依頼事項について協力を要請し，その結果上位マネジメント，関連部門共々，各々の責任の分担を確たるものとする．

そして，こうした人達に細部の Presentation を行なうことにより，自分では気付かなかった新たな問題の指摘を受けたり，よりうまい解決策やヒントが得られるかも知れない．思わぬ誤りの指摘を受けたり，根拠となるデータ類の修正やその他の情報の入手等々（特にプロジェクト・マネジメントには経験の積み重ねが重要であるといわれているが），こうしたミーティングによりプロジェクト・マネジャー自身の経験を補うことに努めるべきである．十の指摘の内一つでも有効となればすばらしいことであろう．このミーティングにおいては，プロジェクト・マネジャーの熱意と謙虚さが必要である．

こうした主旨であるから，関連部門以外にも，経験の深い人にぜひ出席してもらうよう努めるべきである．

こうして，このミーティングにおいて責任分担を確たるものとし，プロジェクト計画の Authorize を得るのである [*37]．なお議事録は必ず配布する．

根拠／理由： 所詮人は自分の限られた空間で考え行動するものである．しかるに，新しいプロジェクトというものは，大なり小なり未経験のものを含んでいるものだ（そうでないようなプロジェクトは逆に問題にならないだろう）．そういった意味で，大きな見落しをしたくなければ，自分の空間を広げることを考えなければならない．

こうした基本的なステップを，（めんどうでも）着実に踏んでゆくことにより，自分では気付かないような微細な徴候から，問題を発見する可能性が出てくるのである．しかし，こうしてさえ発見できないかも知れない．従って，このような努力をしようとしなければ，問題を早めに発見できる可能性は小さなものになるといえる．

もう一つ，「関連部門への確実な責任の分担と，上位マネジメントの明確な Authorize を得る」のがねらいであって，このような事が不確実では，全員の共通認識によって，「積水を千仞の谿に決する」ことは不可能である．

GP-8 は，彼れを知り己れを知り，味方を不敗の態勢にもってゆく上で，決定的なステップであると言えよう．

以上が本稿の主題である "Guiding Principles" である．本稿は，考え方を述べることを目的としているのであり，「GP-1 から GP-8 をそのままの形で実行すること」よりは，「GP-1 ～ GP-8 の考え方を応用すること」を主張するものである．

3. 結論

プロジェクト・マネジャーは，課せられた目的を，期日通りに確実に達成させる為に，いかなる考え方に基づくべきか？ 散慢なやり方では，到底不可能であることは御同意頂けると思う．

本稿第一部の思想，第二部の原理ならびに付録 1 に流れる共通の考え方，私の主張したいことは，次のようになる．

それは，「要所をおさえる」ということである [*38]．六頭立ての馬車の手綱をさばく御者の如く，プロジェクト・マネジャーはどの手綱も完全に掌握しているべきである．掌握

[*36] 計画案としては，この他に，システム運営方法，開発費用，運営費用，導入計画，システムの価値評価，技術的裏付け資料等々が含まれていなければならないが，これらを論じることは本稿の主題ではないので，ここでは触れない．

[*37] もし多くの指摘を受けて，計画案を再度練り直さなければならないことになっても，それはそれでレビュー・ミーティングの目的は達したわけであり，もう一度やり直せばよい．悲観しないこと．

[*38] 自分のところの作業はもとより，必要なソフトウェアが遅れても困る，ハードウェアが間に合わなくても困る，そのための手続きがされてないのも困る，大事なファイルを壊されても困る……「すべてを掌握し，要所をしかとおさえてゆく」これが本稿第一部，第二部を貫いている考え方である．

できておらず，ゆるんだ手綱があるようでは，そうした散慢なやり方では，確実な成功はおぼつかない．

「何が要所であるか」を見抜き，そしてその「要所をおさえる」こと，これ如何ではせっかくの各種技法も有効とはならない，と序論で指摘した「使う側の問題」の意味はここにあり，個々の技術と相まって，それら技術を有効にはたらかせるためのかなめであると結論するものである．

そうして，この「要所をおさえる」ことは，「彼れを知り，己れを知る」ことによって可能となるのであって，かくの如き科学的ともいえる命題を与えている．この２千年以上も昔に成る『孫子』に，学ぶことが多くあると思う次第である．

4. あとがき

本稿においては，プロジェクト･マネジャーと部下との関連については，直接には論じていない [*39]．

プロジェクト･マネジメントにおけるこの重要な要素については，多くの論文，特に AMA のテキストに言い尽くされており [*40]，私にはさらに加えなければならない理由がない．

この拙稿はずいぶん以前にまとめたもので，内輪ではよくコピーされて読まれていたようです．たまたま，ある方にお見せしたところ，SEAMAIL に掲載しては，ということになりました．

内容を更新して載せるのが本来なのですが，その余裕も能力もない次第です．ご容赦下さい．

付録　1. ５つの留意事項
　　2. Guiding Principles 概念図
　　3. AMA テキスト「動機づけ」要約

[参考文献]

(1)『孫子』金谷 治 訳注（岩波書店）
(2)『孫子･呉子』村山孚 訳（徳間書店）
(3)『孫子』天野 鎮雄 訳注（講談社）
『孫子･呉子』鎮雄 訳注（明治書院：新釈漢文大系）
(4) AMA マルチメディア教育システム
『管理者養成コース』（フィルム台本）
アメリカ経営者協会（国際経営センター）

[*39] 間接的には，いくつか触れたつもりである．

[*40] R.A. マッケンジー「コミュニケーション」および J.J. クリビン「動機づけ」．なお，付録 3 にこの簡単な要約を示すので参照されたい．

付録 1.
５つの留意事項
（プロジェクト推進にあたって意識した具体的事項）

留意事項１： 各作業に最も期待することは何かを，担当者に対して作業指示書や仕様書等によって，はっきりと指示する [*41]．
例：「実行スピードを最も速くせよ」
「バグ発生をゼロにせよ」
「わかり易い構造にしてメインテナンスをしやすくせよ」等
この指定が有るか無いかだけで，出来上がりの品質が大きく左右される．

留意事項２： 曖昧さを無くす．そういうものは放置せず，時期を失せずフォローする．現在時間が無くて検討できないならば，いつ検討するかだけでもはっきりさせる．
次に，あまりいい話は，そのベールを剥ぎ取り，その実体を見定めるべく努めること．

留意事項３： 誰れでもが同じように認識できるように客観的な表現をする．例えば，具体的な数値で示す．
例：「大量の変更」－>「20000 ステップのうち約 2 割，4,000 ステップの規模の変更」

留意事項４： ラッシュ･テスト，リカバリ･テスト，総合テスト等，線だけ引いてあるものを見かけることがあるが，テストの数ヶ月前には，内容を具体的に定義しておく [*42]
例えば：どんなプログラムとデータで，どんな端末を使って，どんな担当割で，どんなテスト項目を，どんなフェイズ分けで，何時間を何回，といった事柄を整理しておく．
一本の線だけでは，誠に心もとない．

留意事項５： ユーザー･プログラムが収束してき，さらに

[*41] 'The Psychology of Improved Programming Performance' (Gerald M.Weinberg) の中で述べられている実験と結果の考察が示唆に富む．
この考え方は，「個々の作業についても，プロジェクト･マネジャーは明確な目標を設定するべきである」ということで，いろいろと応用ができる．

[*42] 実際，本番の二ヶ月前にやって私も安心できた．

ソフトウェアやハードウェアの問題が無くなってくると，問題が発生する要因は，プロシデュア，その運用，人の面に絞られてくる．そして，何もしないとこうした問題が顕在化してくるものである．

従って，プロジェクト・マネジャーは，こうした面で問題を発生させないための仕組みを設定すべきである．

実際に私がやったことは，どんな問題が発生しそうか私自身がとくと考えて，全部書き上げ，これを開発部門，オペレータ部門，エンド・ユーザーに対する窓口を担当する人等，関係者に提示し，対策を検討させ，対策・役割を決定した等である．本番前にこうした事柄を水も漏らさぬ程シビアに検討し，実施方法を決めた．私としては，問題を発生させないための「仕掛け」をセットしたつもりであって，効果は大きかった [*43]．責任者としてギリギリのところに追いつめられると，いろんなことに考えが巡らせられるものであると思う [*44]．

なお，この時の資料の抜粋を次にに示しておく．

リアル稼働時に発生の可能性がある問題を，「未然に」防ぐための措置がどう考慮されているか？

（検討のためのチェック・リスト）

Q1. 翌日使用するログ・テープは，いつ，誰が，どこに，何本準備するか？

Q2. ログ・テープに使用する MT の品質基準は？

Q3. ログ・テープをセットするテープ・ユニットは任意のユニットか？

Q4. そのユニットのヘッド・クリーニングは毎朝行なうか？

Q5. 使用中のログ・テープをオペレータが誤ってリワインドさせたりすることのないような措置をあらかじめ講じるか？

Q6. リアル終了後のログ・テープの管理は決まっているか？誰が，どこへ，いつまで，どのように保管するか？（レーベルの内容，リング等）

Q7. 翌日使用する Initiation Set Up Card の保管責任者は誰か？ そして，副管理者は誰か？ 保管場所はどこへ？ ディスク・ファイルにも持つか？

Q8. Initiation Set Up Card の変更についてのルールは？
- 複数の人がさわることはないか？
- 変更は前日××時までとするか？
- 変更した時，リストをとって保存するか？
（××月××日 リアル稼働環境）

Q9. Real Initiation User Program，サブルーチン等の共通モジュールは誰がメインテナンスするか？

Q10. Real Application Program の Maintenance は，誰が行なうか？ 正しくメインテナンスされたことを誰が確認するか？ 必ず作動確認を行なうか？ それとも省略するか？

Q11. 正しくメインテナンスされたプログラムをオペレーション部門へ引き継ぐ時のルールは？ どういうことで引き継いだことになるか？ オペレーション部門は，どういう基準を満たしている時，引継ぐことにするのか？

（途中省略）

Q24. 最後に，あやうくセーフになるか？ 確実にセーフになるか？ どちらを選びますか？

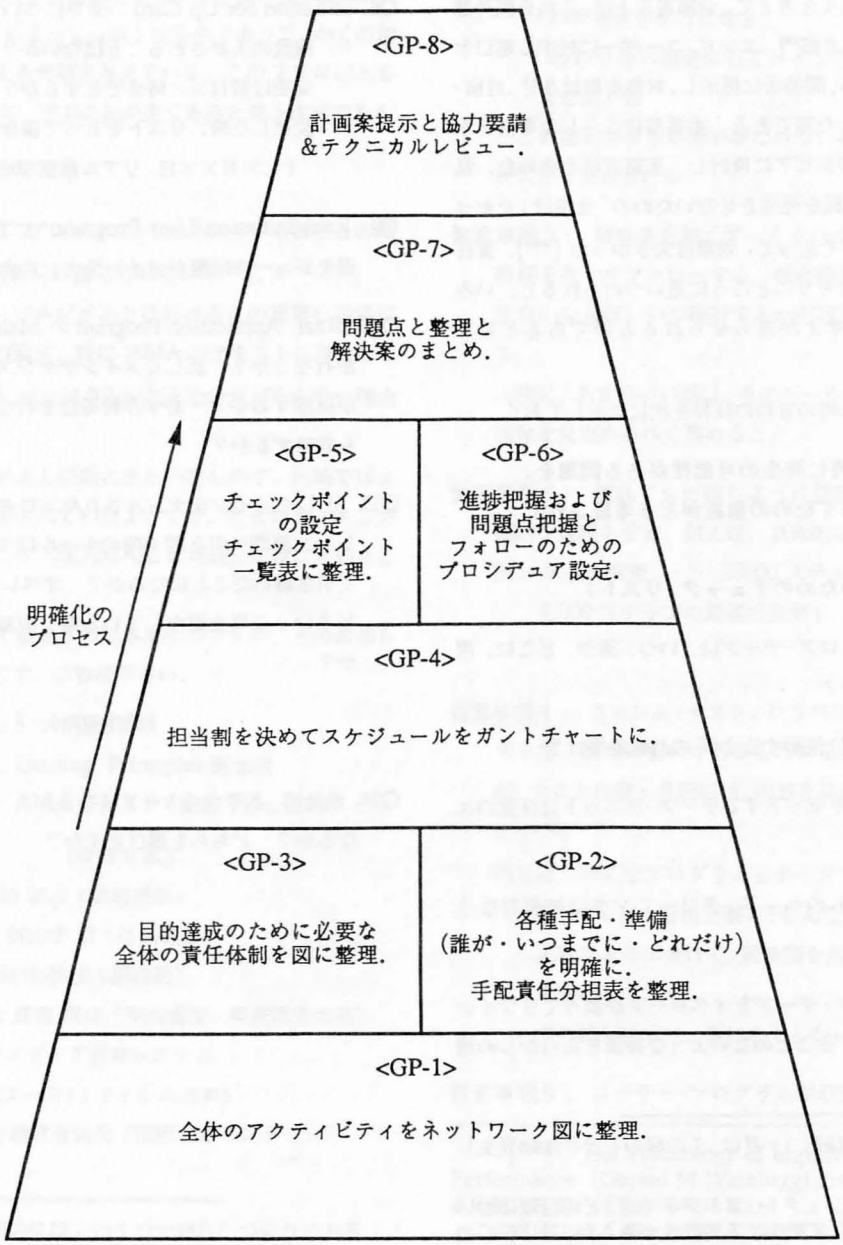
[*43] 稼働後約 1ヶ月は，この種のトラブルは発生しなかった．

[*44] プロジェクト・マネジャーは，どの段階においても，何が問題として発生する可能性があるか，常に細心の注意を払っているべきである．担当者は，割り当てられた作業をこなしているのであって，プロジェクト・マネジャーまで担当と同じような意識でいては，いくらでも問題は発生してしまう．

付録 2:

Guiding Principles 概念図

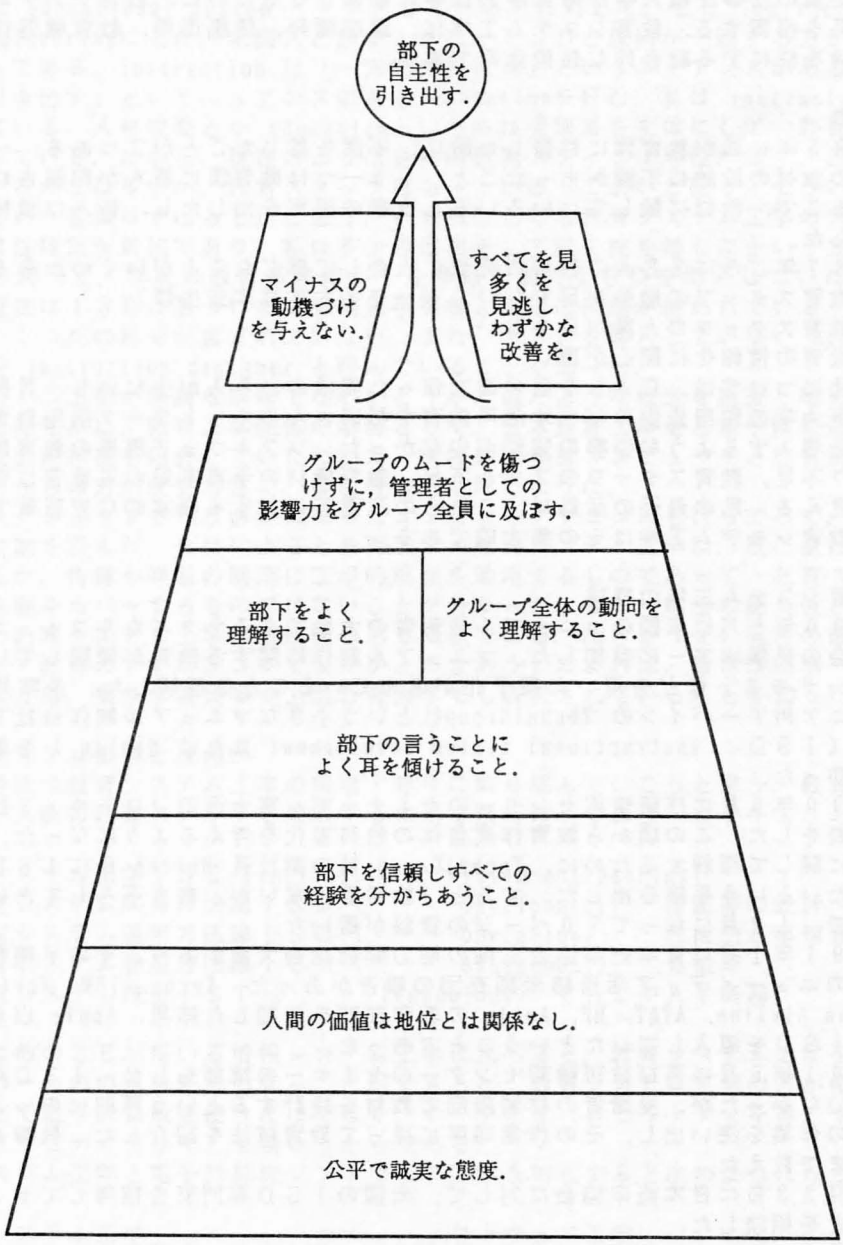
本論第二部で論じているプロジェクト・マネジャーの 8 つの指導原理について，図案化して以下に示す．



付録 3:

AMA テキスト

J.J クリビン「動機づけ」の要点図式 [*45]



[*45] 本稿が，AMA のテキストに目を通されていない不特定の方に読まれるかも知れないことを想定し，動機づけについて，私の理解した範囲で，要点を図で示す.

教育システム工学の提案

君島　浩
（富士通株式会社）

はじめに

教育要員の仕事を職人芸から工学的仕事に転換させるために，教育システム工学という技術体系を提案する．教育システム工学は，講座開発，講座運用，教育機器技術，教育管理方法論を柱にする総合的な技術体系である．

1．背景

１９８５年に私が教育課に移籍した時に，不満を感じたことが二つある．一つは新入社員教育の教材の設計に不満があったこと，もう一つは教育課に新人が配属されないことである．そこで，教材に関してはいろいろと改善の提案を出したし，新人は積極的にもらうようにした．

１９８７年ごろにＳＥＡの教育研究会に入会して感じたことがいくつかある．

- 教育スタッフの勉強が足りない．（ハワイ大学宮本先生談）
- 教育スタッフの人数が足りない．
- 教育の情報化に関心が薄い．

情報化については，ＣＡＬを自分達で作った実績のある人が少ないし，月例会で発表された長野大学の和田先生のビデオ活用教育や杉田さんのネットワーク活用教育について，それらを導入するような姿勢の質問が少なかった．ソフトウェア産業の教育部門は，教育スタッフ不足，教育スタッフのスキル不足，教育機材の予算不足の三重苦に苦しんでいるように見える．私の最近の活動は，これらの三重苦に対する根本的な克服策であるように思う．教育システム工学はその集大成である．

2．教育システム工学の経緯

１９９０年５月に米国のマニュアル執筆者の大会であるテクニカルコミュニケーション国際大会の見学ツアーに参加した．マニュアル制作に関する教育が徹底していることと，ハイパーテキストなどを用いた電子化が進んでいることを理解した．企業見学ではカリフォルニア州アーバインの TechniGroup社という小さなマニュアル制作会社で，教育開発方法論（ＩＳＤ：instructional system development または design ）を適用していることを知った．

１９９０年９月に技研情報センターのセミナーで，富士通のソフトウェア技術者教育の事例発表をした．この頃から教育作業自体の教科書化を考えるようになった．そこで，ＩＳＤに関して理解するために，TechniGroup 社の副社長 Herron 氏にＩＳＤの教科書を入手したいという手紙を出した．すると「教科書はないが，書き下ろしてあげよう」ということで，１２月になって７０ページの資料が届いた．

１９９１年１月に日本能率協会主催の能力開発総合大会があり，その予稿集に１９９０年８月のニューメディア活用訪米調査団の報告があった．Xerox，IBM，Northern Telecom，American Airline，AT&T，HP，Apple の教育部門を訪問した結果，Apple 以外のすべての企業がＩＳＤを導入していたということであった．

１９９１年５月に再び技研情報センターのセミナーの講師をした．ＩＳＤの詳細は勉強できていなかったが，受講者の作業指向で教材を設計するという原則にのっとって，教育担当者の作業を洗い出し，その作業順序に従って教育技法を紹介した．教育スタッフの採用活動まで教えた．

１０月２３日に日本能率協会に対して，米国のＩＳＤ専門家を招待してセミナーを開催することを相談した．

１０月２４日からの教育ワークショップでは，富山大学の山西先生から教育情報学のカリキュラムの話をうかがった．このような専攻が日本でも設置されていたということに感心した．ただし，教育情報学は教育学と電子計算機学が柱であり，企業内教育担当者養成のためにはＩＳＤ関係とＡＶ関係が不足していると感じた．

１０月２９日にソフト・リサーチ・センターにＩＳＤを中心とする教育技法の講義をすることを持ちかけ，了承された．しかし，セミナーの全体設計がすっきりしないな，という感じを抱いたまま，いろいろ文献を読んでみた．１９９１年９月には中央情報教育研究所の佐藤文博氏が教育システムエンジニア（ＥＳＥ）の構想を提案している．ＩＳＤとＡＶをカバーしている．これなら企業内教育担当者養成は大丈夫だ．しかし，何かが欠けている．前に述べた企業内教育の三重苦，①教育スタッフの技量不足と③教育の情報化の

不足は改善できるが，②の教育スタッフの人数不足を改善できない．そこでソフトウェア工学の考えを加味したらどうかとひらめいた．ソフトウェア工学には工数見積などを含むプロジェクト管理方法論が含まれる．教育についても，これだけのページ数の教材を開発するから，人を何人くれ，というように定量的に人数を調達すればよい．この三本柱をそろえれば，教育の三重苦を克服するノウハウがすべてそろう．

3．教育システム工学の位置付け

技術体系の名称は，佐藤氏の命名を尊重して教育システム工学とした．ただし，英語名は佐藤氏は educational system ・・・としているのに対して，私は instructional system engineeringにした．米国人と話すときに，ＩＳＤと最初が一致する方が便利だと思ったからである．instruction は「一方的に教える」というニュアンスがあるので，日本人は「引き出す」というニュアンスのある educationを好む．私は instructionでもよいと思っている．人材開発とか educationというのは受講者を主体にして，教育スタッフがへり下っている．だから，現在のように企業内教育スタッフは地位が低く，三重苦にあまんじているのではないか．教育スタッフはもっといばってよいと思う．そういう意味で「先生」という言葉はすばらしいと思う．いずれにしても教育システム工学の大部分を提案したのは佐藤氏が最初であり，私は多少の追加をして総合性を増したという立場になる．

その後の調べで，弊社の教育事業部が１０年ほど前にＩＳＤを導入していたことが分かった．現在はＩＳＤに基づいた講座開発手引書と各種の用紙が使われている．ただし，ＩＳＤのいくつかの部分が含まれていない．また，ＩＳＤを導入していない Apple社でも教育要員を instruction designer と呼んでいることを知った．

教育システム工学が陳腐な提案でないかどうか，似たような概念を調査し始めた．富山大学の山西先生には，教育工学関係の学会として，日本教育工学会と電子情報通信学会教育工学研究会の存在を紹介していただいた．情報処理学会コンピュータと教育研究会に比べると，こちらの方が歴史も古く，参加者の層も厚いようだ．

教育ワークショップで知り合いになったコンピュータサロンにも行ってみて，教育工学に関する文献を読んだ．それによると教育工学や教育システム工学は，既に使われている用語であるが，指導や学習の局面に工学的概念を適用するものであって，教育スタッフのすべての作業をカバーするものではないことが分かった．カバー範囲の総合性という意味で，新しい教育システム工学は新規性のある概念である．㈱ぎょうせい発行の後藤和彦他編「メディア教育のすすめ④コンピュータとつきあう」によれば，教育情報学・教育システム工学などは，東京学芸大学や上越教育大学でも研究されていることを知った．

4．教育システム工学とは何か

私は，今後は教育システム工学の提唱・普及に取り組んでいこうと思う．教育システム工学とは一人前の教育スタッフを養成するに足る技術体系であり，カリキュラムである．

教育システム工学ＩＳＥ（instructional system engineering）
- 教育システム開発方法論ＩＳＤ（・・・ development）　講座等の設計・制作
- 教育システム運用方法論ＩＳＯ（・・・ operation）　教授法と事務管理
- 教育システム管理方法論ＩＳＭ（・・・ management）　見積等
- 教育システム技術ＩＳＴ（・・・ technology）　ＣＡＬやＡＶ機器

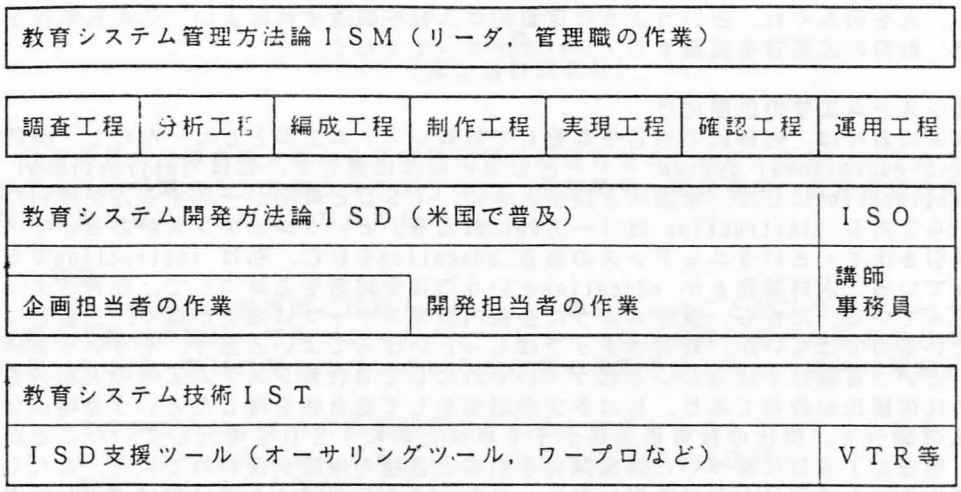
電子計算機のＳＥが用いる情報システム工学に比べると，教育システム工学を情報システム工学の一つの応用分野とも見ることができるが，教育システム工学の方が幅が広い．ＣＡＬなどで電子計算機を使うだけでなく，オーディオ（カセットデッキ等）やビジュアル（ＯＨＰやビデオデッキ）も用いるからである．

教育システム工学と電子計算機ソフトウェア工学を対比すると次のようになる．

教育システム工学	⇐⇒	ソフトウェア工学
教育システム開発方法論		ソフトウェア開発方法論（ＳＡ／ＳＤ等）
教育システム運用方法論		ソフトウェア保守方法論
教育システム管理方法論		プロジェクト管理方法論（見積等）
教育システム技術		開発支援ツール（ＣＡＳＥ等）

教育システム工学はどんな職種でも教育する一般的方法論である．システム工学の業種・業務知識に対応する分野定石も教育システム工学に対して存在する．ＳＥ教育，ソフトウェア技術者教育，自動車販売員教育，自動者保守員教育，スチュワーデス教育，原子力発電所操作員教育などが，教育システム工学の応用分野の例である．教育システム工学の一般的方法論に加えて，これらの分野定石も教育要員には必要である．

教育システム工学の体系と作業工程の関係を次に示す．工程用語は検討中であり，今後変わるかも知れない．



企画担当者と開発担当者の作業は同じ工程を用いるが，規模が違う．複数講座群を設計するのが企画作業であり，単一講座を設計するのが開発作業である．

教育要員には次のようなキャリアパスを設定して，それぞれに必要な技法の教育を行う．これによって，経験者に頼らずに，新入社員を教育要員として育成することができる．

ジュニア要員	実現工程・確認工程・運用工程	（主に講師）
シニア要員	調査・分析・編成・制作	（主に講座設計者）
リーダ	調査・分析・編成	（主にカリキュラム編成）
管理職		

5．ＩＳＤの概要

ＩＳＤの特徴は分析工程に現れる．ここでは受講者の仕事（job ）を作業（task）に分解する．講師の持っている理論の構造によるのではなく，受講者の作業の構造に従って講座を設計しようとするのである．講座によって得た知識を，どんな作業に適用するのかで，講座を設計する．つまり作業指向の設計をする．

例：　ＵＮＩＸ講座は，ＵＮＩＸ知識を得た受講者が職場に戻って，ＵＮＩＸ上の応用プログラムを開発するとしたら，どんな作業項目があるかを洗い出す．

例：　コンパイラ技法講座は，受講者が職場に戻って，基本設計，機能設計，構成設計，詳細設計，テストの作業をするであろうことを想定して，設計する．例えば，コンパイラのテスト技法を教えたりする．

例：　新しいソフトウェア製品の企画をＯＨＰで説明する時に，聞き手である事業部長がどんな判断・行動するのかを分析して，ＯＨＰのストーリーを設計する．

例：　教育システム工学を教育要員に教育する時には，教育要員の作業項目を洗い出す．例えば，教育部長が大学を訪問して教育要員の採用活動をするという作業項目を洗い出す．

これらの作業指向の分析は，マニュアルの分野では既に日本でも普及している．米国では２０以上の大学がマニュアルライタ向けのテクニカルコミュニケーション学科を設立した．日本でもマニュアルライタの教科書が出版され，団体・学会が設立されつつある．ただし，ＩＳＤや教育システム工学の教科書や学科が，マニュアルライタの教科書や学科ほど急速に増えるかどうか心配である．企業内教育部門には外部講師に頼って，自社で教科書を開発しないところが多いからである．

マニュアルライタの技法とＩＳＤの違いは次のとおりである．

- ＩＳＤではすべてを教科書に記載するのではない．作業項目によってはＯＪＴにまかせる．
- ＩＳＤはメディアが多様である．紙資料，ＯＨＰ，スライド，カセットテープ，ビデオテープ，ＣＤ－ＲＯＭ資料，ＣＤ－ＲＯＭ型ＣＡＬ，フロッピー型ＣＡＬなど．
- ＩＳＤではＩＳＤ支援ツールが発達している．ＣＡＬのオーサリングツールなどである．

一般に，マニュアルライタの技法よりＩＳＤの方が幅が広い．しかし，マニュアルもビデオテープマニュアル，ＣＤ－ＲＯＭマニュアルなどが登場してきているので，次第に近いものになりつつある．電子計算機ソフトウェアもＣＤ－ＲＯＭを用いたり，ビデオデッキを制御したりしつつあるので，教育システム工学とソフトウェア工学・情報システム工学も近いものになりつつある．

６．教育システム管理方法論

教育システム工学のもう一つの目玉は，教育システム管理方法論である．これはソフトウェア工学の工数等の見積などに対する概念である．以下のような尺度がある．

- 教材の量／開発工数
- 教育要員数／社員数
- 受講人日／人年

従来の教育については，教育の最終的な効果（受講部門の生産性）などばかり議論されて，教育にどれだけの投資をしているかの指標が議論されなかった．これからは「ＩＢＭ社は社員数の３％を教育要員にしている」，「アンダーセン・コンサルティング社は７％を教育要員にしている」などというように，教育投資について意識するようにしたい．ちなみに日本の企業では，教育要員投資は少なくて，１％以下であることが多いようである．

７．教育システム技術

教育システム技術とは，ＡＶＣＣＭと略称される教育用の機器である．

Ａ：	audio	カセットデッキなど
Ｖ：	video	ビデオデッキ，ＯＨＰなど
Ｃ：	computer	ＣＡＬなど
Ｃ：	communication	衛星通信，パソコン通信など
Ｍ：	machine	自動車，フライトシミュレータ，消火器訓練用火皿など

これについては詳しくは説明しないが，ＯＨＰなどでは文字フォントや行間隔など出版業者の知識が必要であるし，ＡＶについては放送局等の知識が必要であり，非常に奥深い．

８．まとめ

教育システム工学の位置付けや発想の経緯を紹介した．今後は２月のソフト・リサーチ・センターのセミナーで企業内教育担当者に対して詳細な説明をする．また３月の電子情報通信学会春季大会で発表して，教育工学の権威の方々に意見をうかがう予定である．

＜参考文献＞

1) 後藤他編，「メディア教育のすすめ④コンピュータとつきあう」，㈱ぎょうせい，１９８７．
2) 富士通社内資料，「INSTRUCTIONAL SYSTEM DESIGN METHOD」，TechniGroup，１９９０
3) 「企業内教育におけるニューメディア活用訪米調査団90」，日本能率協会，１９９０
4) 柚口，「編集工学入門」，アスキー出版局，１９９１．
5) 清水，「日米における教育ニューメディア活用の現状と将来」，第10回記念能力開発総合大会，日本能率協会，１９９１
6) 佐藤他，「教育を企画・推進する新しい人材像とその育成課題」，コンピュータと教育研究会予稿集，情報処理学会，１９９１

ソフトウェア・プロセスをめぐって

－ 第２回 JSPW のポジション・ペーパ拾遺 －

いささか古くなったが，一昨年の２月に，伊豆・五景館で行われた第２回ソフトウェア・プロセス・ワークショップ (JSPW) の予稿集の中から，いくつかのポジション・ペーパを選んで再録する．本来であれば，もっと早く，討論記録などを含めた完全なレポートの形にまとめるべきものであったが，ワークショップ関係者と Seamail 編集部の両方が多忙であったために，遺憾ながらその時機が失われてしまった．

しかし，これらのポジション・ペーパの内容はいまでもかなり新鮮な技術的内容を含んでおり，再録の価値が十分にあるというのが，編集部の判断である．それは，裏を返せば，一部の研究者の方々の努力にもかかわらず，ソフトウェア・プロセス（特にそのモデル化）の重要性についての産業界の認識が，いまだに深まっていないことを意味する．

JSPW の第３回は，昨年の２月に同じ会場で，文部省・科研費の大型プロジェクトの研究者たちとのジョイント・ミーティングという形で開催された．今年の第４回は，すでに参加者募集パンフレットで御案内の通り，やはり２月に，今度は場所を浜松に移して開催される予定になっている．

ソフトウェアプロセスの形式的体系
ISPW5 での議論を踏まえて

鯵坂 恒夫
（京都大学）

1. ソフトウェアの形式的体系

ソフトウエア工学ないし科学には，機械工学や電気電子工学などと比べ，その基礎となる理論的体系がないとよくいわれる．しかし，実はないのではなく，理論と実際との距離が他と比べるとかなり遠いというのが真実である．機械工学や電気電子工学の基礎となる力学や電磁気学は，成熟した解析学に基づいて構築されており，日常の現象を直接的に説明する．一方，計算機ソフトウェアの基礎となる離散数学や数学基礎論などは，数学のなかでも古典的解析学に比べて随分新しくて未成熟な面があり，またとくに数学基礎論は数学の数学というメタな性格から哲学的な色彩を帯びていて，実際とはやや距離がある．これは例えば相対論的量子力学が今のところ実学からかけはなれているのと似ている．

しかし，やや実際から遠いところにあるとはいえ，Hoare の意味論や Church, Turing, Kleene といった人々の与えた形式的体系は，ソフトウェアの立脚する本質としていろいろな示唆を与え，またある種の安心感を与えている．今はまだ黎明期の混沌にあるソフトウェアプロセスの研究であるが，実際の開発・生産プロセスの観測に基づく研究の一方で，基礎理論や形式的体系の整備，できれば実際になるだけ近いものが必要である．ソフトウェアプロセス研究への視線が Osterweil によるプロセスプログラムの提唱により集中の度合を高めたことからも，プロセスの形式化への方向がうかがえる．

以下，ISPW5 における議論を踏まえ，そこで話題になったことがらがどのように形式化できるか考察してみる．本文中 [ ] で示した参照は，すべて ISPW5 の position paper, presentation または discussion である．

2. プログラムの意味とプロセスの意味

さて，「ソフトウェアプロセスもまたソフトウェアである」というテーゼからすると，ソフトウェアプロセスの形式的体系はソフトウェアの形式的体系と基本的には同じである，ということになる．事実，ソフトウェアプロセスに関わる研究者の多くが，プロセス記述言語の定義に興味を向けているという観察があり，しかも，その一部は，プロセス記述言語も従来のプログラム言語（プロダクト記述言語）と同じ枠組みでよいと主張しているという [Curtis,Balzer]．プログラム言語には形式言語理論という堅固な基礎理論がついており，これは計算可能性に関する一連の理論にも連なることから，形式的体系としては心強い．

しかし，形式言語理論が美しくかつ現実的であるのは構文論であるからであって，意味論となると途端に理論と実際との遊離が激しくなる．先にも引用した Hoare の公理的意味論は，プログラムの制御とデータのミクロな論理を組み上げていかなければならないため，大規模で複雑なソフトウェアの意味を与えるのは不可能に近い．これは素粒子論によって日常的な物理現象を記述することに意味がないのと同様である．表示的意味論になるとさらに集合論的な抽象化が著しく，現実から遠い牙城となる．一方，操作的意味論は，意味を与える過程がやや形式性に乏しいが，最も現実的かつ直感的である．この意味での「意味」をもう少し

追求してみよう．

ソフトウェアに関わる意味には次の 3 つのレベルがある．

データの意味： これは日常使う「言葉の意味」という意味に近く，該ソフトウェアの対象世界における概念集合に写像される．

プログラムの意味： 入力データに対応する出力データによって与えられる．

言語の意味： コンパイラというプログラムの意味がこれにあたり，ソース言語に対応するオブジェクト言語によって与えられる．

プログラムの意味と言語の意味は，ともに入力に対応する出力によって意味が与えられるという点で，type free の体系（集合 D と (D → D) を区別しない）ならば区別する必要はないが，実世界のデータを扱う現実的ソフトウェアを対象とする限り，これらは階の異なるものとして区別しなければならない（このあたりが理論と実際の遊離が始まるきっかけのひとつである）．

この線にそってプロセスの意味を考えると，入力アーティファクトに対応する出力アーティファクト（プロダクト）によってその意味が与えられる．一方，プロセス記述言語の意味であるが，おそらくプロセス記述言語に対しては，コンパイラのような翻訳·変換による意味づけはできず，プロセス記述と入力アーティファクトから得られる出力アーティファクトによって意味を与える．インタプリタ式の解釈·実行による意味づけになる．いずれにせよ，アーティファクト間の関係を明確にすることが，プロセスの意味を明らかにすることにつながる．なお，ここでアーティファクトとは，ソフトウェアプロセスに現れる仕様書等のドキュメントやプログラム，およびその内部の構成要素を取り出して指すこともある [Cheatham]．現在，筆者の所属する研究室では，アーティファクトとその相互関係の管理を中心として，オブジェクトモデルによるソフトウェアエンジニアリング·データベースの研究，開発を進めている [Matsumoto-Ajisaka]．

プロセスの意味，アーティファクトやプログラムの意味は最終的にはデータの意味，すなわちアプリケーションドメインにおける概念に連なり，この概念集合を明確にしておくことがプロセスを定義する上でも重要である．これはプロセス定義がアプリケーションに独立ではありえない所以である [Curtis, Wileden]．

3. プランと人間

プロセス記述言語はプロダクト記述言語と同じ枠組みではない．プロセスプログラムとは誤称であって，プロセスはプランされ，モニタされるものである，という主張もある [Balzer, Dowson]．ではプランとは何か．先に述べた一連の「意味」は多分に関数的，プロダクト指向であって，時間の流れが欠如している．プランとは，PERT で示されるような時間の流れを強調したプロセスの記述である．collaboration/cooperation/coordination もここに含まれ，プロセス指向のプロセス記述である．筆者の属する研究室ですすめている Unit Workload Network の研究，開発もこれにあたる [Matsumoto-Ajisaka]．

collaboration/... が出てくると必然的に，形式化には最も手強い人間が現れる．人間はとりあえずプロセッサとして形式化しておくしかないが，不確実な性能属性を具えたプロセッサであり，ill-behaviour の源泉である．これを形式化するためには，不連続性，非単調性，矛盾性を扱うモデルを研究する必要があるが，[Katayama]，より現実的な線としては，これまでのプロダクト記述でも用いられてきた技術，例えば Jackson 法に現れる back tracking (roll-back), CLU でとりあげられていた例外処理のメカニズム，実時間システムにおける同期の手法などが使えそうである．

4. プロセスの階層とオブジェクトモデル

プロセスにはその実働化の主体により個人，チーム，プロジェクト，組織等という階層がある [Curtis]．プロセスモデルを構成するオブジェクト空間も，この階層に対応した型階層があり，各階層に適したオブジェクト管理があるはずである．プロセスプログラムあるいはプランを格納する動的オブジェクトの型階層がまさにこれにあたる．一方，アーティファクトやプロダクトを保存する静的オブジェクトにも，要求仕様記述の標準的な章だてやプログラムモジュールの構成に見られるような階層構造があって，これと動的オブジェクトの階層との対応も検討しなければならない．ソフトウェアプロセスの扱うデータモデルとしてオブジェクトモデルを採用することについては，あまり異論のない前提のようである．しかし，このオブジェクトモデルは，最近の爆発的な流行とは裏腹に，形式的体系としては今のところかなり弱い．形式化や正規化には滅法強いデータベース·コミュニティが，このところ怒涛の勢いでオブジェクトモデルになだれこんできている．persistency や型階層などを含め，是非われわれのコミュニティの手でオブジェクトモデルの形式的体系を確立したいものである．

5. フォンノイマンの悪魔

すでに言及したように，プログラム·プロダクトはデータから見れば 1 階上の存在であり，プロセスはプロダクトより一階上にある．さらにもう一階上を考えるとプロセス定義プロセスがあって，ここにプロセスのポリシーなるものが登場する．このポリシーをプロセス定義に反映させ，プロセス実働化において強制する方法として，

(1) プロセスモデル定義プロセスにおいて ad hoc に
(2) プロセスモデルのテンプレートとして
(3) "policy statement"として独立に
(4) プロセス実働時の meta-rule として
(5) プロセスモデル·ジェネレータへの"polich parameter"として

などが考えられているが [Dowson]，前に述べたプロセスの意味にそって考えると，ポリシーとはプロセス記述に対するアサーション記述であって，上の 5 つのなかでは (3) にあたる．プログラムの公理的意味論を検証に用いる際の論理式に対応し，プロセスステップ間の制約条件や生成されたプロダクトに対する制約条件を与える．

上の 5 つの内，(4) や (5) は例によって attractive であるが，危険である．ジェネレータ·ジェネレータの複数の階層によって，だまし絵のような自動機関を錯覚させる．エネルギー保存則と同じように，情報量保存則があるはずであって，stored program による自己適用を駆使したフォンノイマンの悪魔のいたずらにはまってはならない．プロセスプログラムもプログラムであるから，プロセス生成プロセスの無限の階層はない，と Osterweil は指摘しているが，これでは前にも少し触れた type free の世界に陥ってしまい，実世界のデータを扱う現実的ソフトウェアとその開発プロセスを按配しようとするときには，おそらく使えない手段ではないかと思われる．結局 (λx.xx) (λx.xx) のような sploge に行き着くのではないだろうか．

しかし，ソフトウェアプロセスには dynamism が要請され，実働中のプロセスの変更や定義の追加が許され，その変更の波及，伝播が追跡されなければならない．また，プロセス·パスの多重性や非決定性もありうる．そのため，リフレクションのような高階性のメカニズムはどうしても必要となろう．各階の型モデルとそれらの間の関係を明確にし，かつ高階の機能がうまく働くような形式的体系が，プロセスモデルとして求められているものである．

ソフトウェア開発過程記述用言語 PDL とそのシステムを開発して思うこと

井上 克郎
（大阪大学）

1. はじめに

われわれは，いままでにソフトウェア開発過程記述用言語 PDL (Process Description Language) を設計し，PDL で記述された開発プロセスを「実行」するためのインタープリタを作成してきた．ここでいう実行という意味は，プロセスの進行にともなってあらかじめ定義されたツールが起動され，必要なオブジェクトが用意されることを意味する．PDL を用いて種々の開発過程を記述してみた．たとえば JSD (Jackson System Development), JSP (Jackson Structured Programming), モジュール化した C プログラム開発過程，TeX のドキュメントの開発過程等の記述を行ない，実際に実行してみた．また，これらの PDL の記述を容易にするための，一種の拡張言語 SPDL(Structured PDL) を定義し，そのグラフィックエディターや PDL への変換のトランスレーター等を作成し，容易に必要とするスクリプトを作るためのいろいろな試みを行なってきた．

これらの仕事の成果報告などでよく問われる質問は，「そういうふうにあらかじめ決められたスクリプトに従って開発作業を行なうことは本当に可能か？ もし可能ならそれは，使いやすいか？」というものである．これに対する答えは，「イエス．そして，使いやすさは，そのスクリプトが開発者のレベルに合っているならば非常に作業効率の向上に役立つが，そうでなければ，細かいひっかかりが全体の作業効率の向上を阻害してしまう場合がある」ということである．

以下もう少し具体的に議論を進めてみる．

2. 初心者にとって

まず，初心者とはここでは開発環境を自分で構築する能力のない者をいう．開発作業の概念は理解しているが，それを具体的にどのような開発環境に反映させて仕事を進めて行くかがわからない．そういう初心者にとっては，自動的にツールが起動されて必要なオブジェクトが用意されること自体非常にありがたいことであろう．比較的こと細かにメッセージを開発者に与えて指示を行ない，開発者を導いて行なってくれるような環境が望ましい．

我々の例では，JSD や JSP の PDL 記述がそれに近いように思う．これらの記述を実行すると，次は何を行なうかは，そのときのいろいろな条件で天下り的に決められており，開発者は，本質的な開発作業に専念できる．

一方，これらの記述は，自分で環境を構築できる人間にとってはうるさいおせっかいになってしまう．はい，この条件はなりたっていますか，はい，次はこれをしなさい，ツールは自動的にこのモードでたちあげておきますからね.... 等など．実際，本当の始めの段階では，決められた通りの作業で満足していた初心者も，やがてすぐに決められた方法以外の手順で作業を行なうことを希望するようになってくる場合が多いであろう．

3. 経験者にとって

作業環境を自在に操れるような経験豊富な開発者にとっては，細かい作業手順の指示やおしきせのツールの起動などは不必要であろう．しかし，だからといってすべてのプロセス記述が役立たないわけではない．いちいちツールを起動するコマンドより，決まり切った作業を自動的に行なってくれる機能，必要なオブジェクトを用意してくれる機能は必要である．例えば熟練者にとっても Unix の make の機能は，重要であろう．

ここで，特に重要と思われるのは，そのスクリプトにあらかじめ作業の進行をある程度，またはかなりその熟練した開発者に選択できる機能が必須であろう．「次に行なう作業は，今の状態ではこれこれできますが，いったいどれにしますか？」的なアドバイスを与えられるなら，このプロセス記述は，有益であろう．

我々が作成したスクリプトの中で，TeX のドキュメント作成過程のものは，ツールの一つとしてメニュー表示，入力を行なうプログラムを，各プロセスが終了した段階で起動するようにしている．その際，メニューの各項目に対応した条件がそれぞれあって，それらの条件が満たされるものだけをメニューで選択可能とし，それ以外は選択できないようにしている．これによって，たとえば，必要なファイルがないのにプレビューしようとすることが避けることができる．このスクリプトは我々の研究室内で実際に利用されており，まずまずの評価を得ている．

4. おわりに

いままで述べてきたように，開発過程を記述し，それを実行するということは，それなりのターゲットをしぼり，適切なプロセスの切り分け，ツールの選択，条件判定などを十分考えたうえで記述を書けばそれなりの利用価値はあろう．いいかげんに，ちょこちょこっと自分勝手な解釈で開発過程を書いて実行してもあまり，有益なものは得られないと思う．また，スクリプトの適合 (Adaptation) が最近注目を集めているが，その機能は必要不可欠なものであろうが，それに頼っていいかげんなスクリプトの修正を行なってはだめであろう．

5. ワークショップを終えて

以上は，ワークショップを行なう前に書いたポジションペーパーである．以下，ワークショップに参加しての感想をいくつか述べてみたいと思う．

(1) プロセスを記述するための目的をどこにおくかという議論が活発にされた．これは，初めに岸田氏 (SRA) が述べた，みんながハッピーになるような環境という抽象的な目的に対していくつかの反対意見が出された．そのような抽象的な目的の議論をしてもあまり面白くない．もう少し目的を割り切って，少しテクニカルな議論を進めた方が良いという意見である．今までのある程度標準的なベースを決めて，その上でテクニカルにいろいろなモデリングやフォーマリゼーションを比較する方が，現実的と思われる．この種のメタな議論をするかしないかあらかじめ主催者側ではっきりしておく方が，時間の無駄を省けて良いかも知れない．

(2) 大槻氏 (日立) が最後に言っていたが，テクニカルなブレークスルーは，無かったように思われる．皆それぞれの方法でフォーマリゼーションを行なっているが，画期的に新たな発見，発明はなかった．しかし，そのなかでも，皆，地道にそれぞれの方法を進めており，それなりの進歩はあるように見える．

(3) 佐伯氏 (東工大) の研究は面白いと思った．前提やその結果の適用などにやや疑問点はあるが，形式的にいろいろな開発過程を表現する上での新たな分類法に用いることができるかもしれないと思った．

ソフトウェアプロセスの記録と記述

上原 憲二
（三菱電機）

Osterweil のプロセスプログラミング [1] は，ソフトウェアプロセスの研究の各方面に大きな影響を及ぼした．その一つは，「プロセスプログラミング」という言葉から直接的に「計算機支援」，「自動化」等が連想される事から起こっていると思われる．この連想から大きな期待が生まれ，逆に反発も生まれた．期待に基づいて多くの研究，実験が行なわれ，その結果に基づいて，またプロセスプログラミングに対する賛否両論がかわされている．ここで，結論を急ぎすぎてはいないかという危惧がある．何が計算機支援されるのか，何が自動化されるのか，それは面白いか，というような議論を，プロセスプログラミングの本来の目標を置き忘れたまま，行ってはいないだろうか．

本来の目標は，人間の知的活動としてのソフトウェアプロセスを解明することである．そのために，プロセスプログラミングはソフトウェアプロセスをプログラミングできるものとしてとらえることを提案している．この提案によりソフトウェアプロセスと言語が結びついた．そこで，既存のプログラミングパラダイム，言語またはその拡張を用いて多くの実験が行なわれた．これらはあくまで問題を詳細化，明確化する第１歩であり，今後更に具体的，実際的なソフトウェア開発の場で実験することが重要である．現時点で結論をだすものではなく，ソフトウェアプロセスをプログラミングするためのパラダイム，言語とはどのようなものか探求することが必要である．

では，ソフトウェアプロセスのためのパラダイム，言語の探求はどうしたらよいだろうか．ここではソフトウェアプロセスの記録と記述の繰り返しによる方法について述べる．

記録は，ある枠組みに従って，実際のソフトウェア開発のプロセスの記録をとることによって，プロセスの構造を明らかにする手掛りを得ることである．記述は，得られた手掛りを抽象化することによってプロセスの構造を抽出し，それを記述するためのパラダイムや言語を定め，それで元のソフトウェア開発が属する分野のソフトウェアプロセスを記述することである．次に，この記述に基づく計算機支援や自動化の有効性によって，得られた構造や記述用の言語を評価し，洗練化する．次の記録に際しては，記録の枠組みを，得られた構造に係る作業が明示的に記録できるよう拡張する．最初の記録の枠組みは，問題，生産物，知識，作業等の関係を表す基本的な構造に従うもの [2] で，詳細は自然言語で書くものから始める．繰り返すに従って枠組みは，ソフトウェアプロセスのより詳細な部分を形式的に表す能力をもつようにする．

最後に，ソフトウェアプロセスの記録と記述において注意すべきことは，分野の特徴をよくふまえた上で記録，記述を行なうことである．分野の特徴をとらえていないプロセス記述は表面的であり，詳細なプロセスまではとらえられないであろう．したがって，その記述に基づく計算機支援，自動化の有効性は低いであろう．

参考文献

[1] L.Osterweil, Software Processes are Software Too, Proceedings of the 9th International Conference on Software Engineering, Mar.-Apr. 1987.

[2] 中島他，PPK 法: ソフトウェア設計プロセスの記録と分析の手法，情報処理学会ソフトウェア工学研究会 67-2, 1989 年 7 月．

ポジションペーパー

臼井 義美
（日本電子計算）

ソフトウェアプロセスの論議は，開発現場にいるものにとっていろいろな示唆を与えてくれる．しかし個人的にはこれまでの多くのソフトウェアとそれほど違った物とは考えていない．開発プロセスに人間的な要素が多いことは事実であるが，従来の経理システムや人事システムなども人間の介在を前提としてシステムを利用している．要は対象となる業務のうちどのプロセスを機械で実行するかという問題であり，機械を利用することが効果的である部分，もしくは方法の発見であると思う．

1. プロセスは人間主体

コンピュータ・システムの本来の目的は，人間の作業を片っ端から機械に置き換えることではなく，人間の活動に役立つように利用することであろう．そのためには，対象領域の作業で機械でなければできない，または機械で十分な作業を抽出して，機械と人間の最適な作業分担を行なう必要がある（というより，そのほうが幸せになれる）．

卑近な例をあげれば，小説の本を作るとき，すべてを機械化することにどれほどの意味があるだろうか？　小説を書くプロセスの多くは記述できないであろうし，製本するプロセスは既に機械化されているだろう．思考や創造，直感的な判断などの人間系に依存するプロセスの機械化は難しいだろうし，すべきでないと思う．この分野で機械のできることは，人間の思考や創造力を増幅し，未踏領域へ踏み込ませる勇気を与えるような支援環境を提供することであって，それらの活動プロセスでさえ思考の対象となる場合がある．

従って，ソフトウェア・プロセスにおいても，人間の活動を活発にするための支援部分と，人間系の活動から排除して自動化すべき部分を分離して考えたい．その時，開発プロセスのすべてを一律にプロセス記述言語で表現し，その中で人間の活動を規定するという考え方にはどうも抵抗がある．むしろ，利用者である人間が主導権を握って目標に到達できるような環境を提供すべきであると考える．例えば，自由な思考を繰り広げられる環境，引き続き実行すべきプロセスの効果的な選択，思考の成果を自然に次のプロセスに引き渡すインターフェース，要求に近づくようにガイドするナビゲータ・システム，目標となる要求を逐次変更できるような管理システムなどである．

2. 有用性の再認識

ソフトウェア・プロセスの機械化を考えるに当たって，これまで開発に携わってきた経験から思い付く素朴な疑問について考えてみたい．その一つは，従来の開発手順書などに示されたプロセスは，手作業を中心とした作業手順を前提としたものであり，機械を利用した場合の最適プロセスではないということである．ソフトウェア・プロセスにおいても，従来行なっていた方法をそのまま機械化するのでなく，何らかの革新が必要であると思う．

第１回のプロセス・ワークショップでも提起した話題であるが，少し例をあげて考えてみたい．

「ある人と話をしたいから会いに行く」ためのプロセスとして，昔は歩くか篭や馬に乗るという手段を取ったが，そのプロセスをいくら分析しても自動車，鉄道による解決策は出てこない．また，地上を移動するという制約を無視しなければ，飛行機を利用するという解決方法は得られない．これらの「行く」ための手段の発展は，従来の方法の延長線でなくある種の飛躍が必要である．さらに，直接の目的は話をしたいことなのであるから，移動という手段を無視すれば，電話や電子メール，電子会議システムによる解決策がある．これは電気という全く新しい手段を利用したプロセスの革新であるとも言える．

もう一つは，目的のプロダクトは本質的に問題の解決に役立つのかということである．ソフトウェアが量産できるようになれば，中途半端なクズ・プログラムの山を築くことになりかねないだけに，非常に気になる．

極端な例だが，我々が最終的に音楽に求めているものは聞いて心地好くなりたいのであって，楽譜を作りたいのではないということである．楽譜が必要なのは人間が楽器を演奏するという，従来の人間系を介在させたプロセスを前提としている．シンセサイザを利用してコンピュータが直接人間の音感に刺激を与えるのであれば，利用者の快感度数をフィードバックして，適切な快感を与える音を発生すればいいわけである．従って，従来のプロセスとは全く異なる方法ではるかに効果的に希望する目的を達成できる．

同様に銀行の窓口システムを例にとると，従来の窓口業務をそのままプロセスで機械化するより，窓口を無くしてATMに切り替えるほうが有効であり，POSやホームバンキングに切り替えるとさらに機械化の効果が大きい．もし，ネットワーク上で受発注や資金決済を行なってしまえば，

銀行というものの存在自体を考え直さねばならないような，本質的なテーマに直面する．

これらの例のように，目的とする業務の本質をもっと真剣に見つめ，本当に必要なものを最も適切に開発するのでなければ，無駄なソフトウェアのためのプロセスについて必死で議論していることになりかねない．我々はプログラムを作ることを目的としているのではなく，ある目的を達成するのにコンピュータ・システムを有効に利用したいのである．逆に言えば，コンピュータを使ってはいけないところを知っておく必要があるかもしれない．もっとも，有効性を論議するのは開発技術の論議と関係ないということかも知れないが，作っては見たけれど後は知らないという技術者の態度も無責任であるように思える．

創造的な部分は人間にまかせ，そうでない部分は機械によってできるだけ合理化すべきであろうが，創造的でない人間もいるわけで，ルーチンワークともいう部分をすべて機械で置き換えた場合，多くの人間にとって本当に求めている結末かどうかは疑問の残るところである．いずれにしろ，本当に求めているものを最も有効な方法で実現しているのかという，有用性工学というような分野があってもいいように思う．

3. 実世界と目的を結ぶプロセス

対象となる世界を忠実に表現し，目的の結果を直接得ることが究極のシステムと言える．そのためにはプロセスよりも求める結果を明確に記述する技術が必要である．先程の音楽の例で見れば，曲を作る方法や楽譜を記述する方法をプロダクトに求めるのでなく，何が気持ちいいのかを記述してそのプロセスを求めるのである．(記述の必要は無いかも知れない) ただし，結果は唯一のものでなく，利用者や時間などによって様々に変化する．このように利用者が求めているものは常に変化しており，それを適切に支援できるプロセスが欲しい．

この場合，求めるプロセスとは現実世界と希望する世界を結合するもので，求めるものを作るプログラムを作るためのプロセスを作ることではない．いわばダイレクト・モデルとでも言えばいいのか．オスターワイルのプロセスプログラミングは，現在のプログラミングの内包する問題を，同じ問題を含むメタなプログラミングに置き換えただけのような気がする．

前回，プロセス・モデルのダイナミクスについての論議があったが，それはソフトウェア・プロセスのためというよりも，プロダクトが本来持っていなければならない特性なのではないだろうか？　同様にプロセス・プログラムの並列実行についても，一般のプロダクトに必要が無いわけではなく，技術的，コスト的に無理だったからではないか．

4. ガンダムとジャムセッション

プロセスについてもう少し考えてみたい．ある場所に到達するプロセスを支援するものに車がある．車を製造するプロセスを支援するのが工場でありロボットである．ロボットを製造するプロセスを支援するのはロボットである (以後，無限ループに入る)．では，ロボットがロボットを作るのでなく，直接車を作ることはできないか？　もしくはロボットが車としての役割を果たせば車を作るロボットは不用ではないか？　そうすると車が車を作ることと同じではないのか？　本当は必要に応じてあるときは車になりあるときは洗濯機になる，場合によっては同時に実行するというのが理想である．ハードウェアでは物資を利用するだけにこういう物を実現するのは難しいが，ソフトウェアでは，資源としてのデータと目的の結果を得るためのプロセスによって，比較的容易に実現可能ではないだろうか．マッキントッシュの「ジャムセッション」などはハードウェアによっていたプロセスをソフトウェアによって実現した好例といえる．

ソフトウェア・プロセスを考えることは，今まで気づかなかったソフトウェアの本質を探ってみたい欲望に駆られる．また，人間とソフトウェアのより良い関係について考えさせられる．今回も，様々な分野の方々との興味の尽きない議論を期待している．

現場のソフトウエア·プロセスに もっと関心を持ってほしい

久保 宏志
（富士通）

昨年の秋，「富士通におけるソフトウエア品質保証の実際」と題する書物を私の監修で出版した．この本をどんな考えでつくったか，できたのがどんなものであったか，できたものをどう活用してほしいかを書いてポジション·ステートメントにかえる．ソフトウエア·プロセス·ワークショップのためのものであるから，ソフトウエア·プロセスとの関わりの観点からのステートメントである．

ソフトウエア·プロセスとその進化について，一つの想定をもっている．それを提示することから始める．

ソフトウエア·プロセスはソフトウエア開発環境とソフトウエア技術者との相互作用である．ソフトウエア開発環境はコンピュータ·ネットワークの上に，必要に応じてつくられる．組織ごとにあるいはプロジェクトごとにつくられる．その意味でソフトウエア開発環境は個別的なものであるが，それらに共通的に一つのアーキテクチャをもつことができる．そればかりか，環境の構築に使われる環境素材は一定の汎用性を備えることができる．

ソフトウエア開発環境の進化は，共通のアーキテクチャの上に蓄積される環境素材が量質ともに充実されて厚い層をなすことであるととらえる．

ソフトウエア·プロセスの進化は，環境素材の進化の成果を取り込んで素材を環境に統合するテクノロジーと，統合環境との相互作用のテクノロジーを進化させることであるととらえることができる．

素材の進化にせよ，統合のテクノロジーや相互作用のテクノロジーの進化にせよ，それらの進化は，相互作用の実践から浮かび上がってくる問題を解決しようとしてなされる研究開発に負うことが多いであろう．現場で行われているソフトウエア·プロセスの実践はいわば，ソフトウエア·プロセスの進化の源泉である．

このような想定のもとに，ソフトウエア·プロセスの進化に寄与するための一つの営みとして，冒頭に掲げた本を著そうとした．書こうとしたのは現場で行われているソフトウエア·茄鴻 Z スの実際である．そういう思いが次の目次をつくらせた．つくったのは一昨年の９月である．

「富士通に於けるソフトウェア品質保証の実際」
の企画書私案

緒言
目次
第１部　品質保証の実際の論じ方
(1) 各部とも IEEE の Software Metrics Standards Draft の枠組みで書く．
(2) 品質を説明するときに使う図表は QC 7 つ道具に限定する．
(3) 各部ともライフサイクルの全体を論ずる．
(4) いわゆる品質管理屋的視点ではなく，設計者の視点で論ずる．各章とも，最重要要求品質が何であるかから説き始める．要求品質を作り込むための設計技術を他人が評価可能な形で提示する．提示した設計技術の自己評価を示す．設計技術にイノベーションをもたらすための着眼を示す．イノベーションに向けての研究開発の最前線を紹介する．富士通で行われている研究開発を，その中に位置付ける．参考文献を多数示す．
第２部　システム·エンジリアリング·サービスの品質保証の実際
第３部　情報システムの品質保証の実際
第４部　カスタム·ソフトウェアの品質保証の実際
第５部　業種向け汎用パッケージ·ソフトウェアの品質保証の実際
第６部　システム·ソフトウェアの品質保証の実際
第７部　ユーザ·マニュアルの品質保証の実際
結語
文献

ここには私の思いがこめられている．プロセス·テクノロジーの進化について書かせようとしている．プロセス·テクノロジーを書くのに一定の枠組みを用い，標準的なツールを使って書かせようとしている．

ところが実際に書かれたものは，ここで構想したものからは大幅に後退している．第１部は，いわば第２部以下を書くための方法論のためを企図したが，そういうものにはしえなかった．ソフトウエア·プロセスの中にサービス·プロセスも含めて，それのプロセス·テクノロジーの章を独立させていたが，これは割愛せざるをえなかった．これが大きな後退であるが，各部の記事についても後退がある．次

を指摘できる．

技術のアセスメントが含まれていない．技術開発の契機を示すことができていない．バリデーションからのフィードバックによってプロセス･テクノロジーをコントロールすることが行われているはずであり，そのコントロール･テクノロジーもイノベーションを繰り返しているはずであるが，それの実際も示すことができていない．こういうプロセス･テクノロジーを実践していますという叙述にとどまっている．プロセス･テクノロジーの定量的評価，比較評価の試みに寄与することはできなかったことになる．

後退に後退を重ねたが，ソフトウエア･プロセスの進化の様を書くことにはかなりこだわった．第５部の最初の原稿ができた段階で，それをレビューしたときにつぎの文章を書いている．

第２章の品質要件の各項と，第３章の品質保証の技術を対応させる．問題と答えの関係を明確にして読者の読みやすさに資する．読者に著者との対話を促す．問題と答えの関係を明示することに全般的に注意したい．この注意は，第５部だけに対するものではない．なぜ，その答えが適当であるかの説明を付して，読者が批判的に技術を検討し評価できるようにしたい．これが趣旨である．

まだ技術と呼べる水準に達していなくても隠すことはない．属人的な技術に止まっているのか，他人に伝達可能な技術に達しているのか，技術の実践を支援するツールがあるのかないのか，そこいらを分からせたい．技術開発の問題がどこにあるかを分からせたい．その到達水準が低いと観察し問題を理解すれば，読者は技術開発への挑戦心を抱くであろう．

更に研究したい人に対する研究ガイドも忘れないようにしたい．

ここに述べた考え方は，全般に適用したい．第５部だけではない．

環境との相互作用の形でプロセス･テクノロジーを示すようなことは企画の段階でも落としている．当然，プロセス･テクノロジーの成果を共通的な環境アーキテクチャの上に蓄積していくようなことも最初から落としている．

現場のソフトウエア･プロセスを書物に著そうとして行った一つの営みの，これは現実である．ソフトウエア･プロセスをプログラミングしようとして挑戦が行われている一方での現実である．二つの距離は遠いと言わざるをえない．

ありていに言って，プロセス･テクノロジを使って行われているソフトウエア･プロセスを書いてみようとする試みも決して多くない．このことも問題にしたい現実である．

現場のソフトウエア･プロセスの実際を書くことがもっともっと行われてよい．ソフトウエア･プロセスを書く試みに対して，より伝達に有効な書き方をするにはどうすればよいか，の観点から建設的な批判が寄せられることを望みたい．書かれたソフトウエア･プロセスを，プロセス･テクノロジーの成果を活用して進化させていくことについても提言が寄せられることを望みたい．

ソフトウエア･プロセスに言語表現を与える．ソフトウエア･プロセスを理解するためにこれを行う．理解を共有するためにこれを行う．既に考案されているプロセス･テクノロジの移転に機会を与えるためにこれを行う．新しいプロセス･テクノロジの発明を必要としている問題を顕在化させるためにこれを行う．進化したプロセス･テクノロジーを社会の資産として蓄積していくことを可能にするための環境の標準アーキテクチャを生み出すためにこれを行う．

プロセス･プログラミングが産業界からも関心をもって迎えられたのは，このような展開への兆候を感じたからではないのか．

現実のソフトウエア･プロセスの論議は何をめぐって行われているのであろうか．ミクロな問題に関心が移って，箱庭研究になってしまう傾向にはないか．

「富士通におけるソフトウエア品質保証の実際」なる書物を監修した立場からポジションを綴ってみた．

ポジションペーパー

田中 浩之
（新日鉄情報通信システム）

1. はじめに

プロセス，プログラミングについては，今回のワークショップに参加するまでは，話として耳にしていた程度であり，一体何をしようとしているのか皆目見当がつかない状態で参加した．以下に，このワークショップで自分が感じたことを述べる．

2. プロセス-プログラミングについて気がついた点

このワークショップに参加してプロセス・プログラミングについて気がついたことを２点述べる．

(1) 開発プロセスの記述法と新しい開発技法の開発

今回の話の中で，開発プロセスの記述法について多くの人が述べており，ソフトウェア開発プロセスの可視化という意味で重要な課題の一つであると思う．これらの研究によって今後，ソフトウェア技術者間のコミュニケーションがより良くはかれるようになり，ノウハウの蓄積や再利用が効率的に行なわれるようになることを期待している．

それと同時に期待していたのが，何か新しい開発技法の提案が多くあるのではないかということである．これについては特に全面に出して主張しているものはなかったようであるが，現場の開発に携わっている我々が，研究をしている方に一番期待しているのは，新しい開発技法の提案である．開発プロセスの技術法というのは新しい開発技法が提案されればそれに見合ったものが必要になってくるものではないだろうか．今後もこのようなワークショップを開くときには，もっと開発技法の提案を表面に出したものが多くてもいいように思う．

(2) 開発プロセスの形式的モデル化の困難さ

私自身はどう実現したらよいか，なかなか考えが浮かばないが，形式的モデル化までできれば静的解析等も可能となり，開発プロセスの定量的な測定も可能となるなど，非常に利益がある．形式的モデル化の課題となっているのは，並列プロセスや人間を介在したシステムをいかに記述するかということである．これに関しては，ソフトウェア開発プロセスに固有の問題ではないと思う．

この形式化が困難な問題について，他の分野ではどのように解決してきたかというと，一つには人工知能の分野で行なわれてきたようなエキスパートシステムの方法がある．他には，現実を大胆に近似したモデルを作るやり方もある（OR などはこのやり方で成功している）．今回の話の中で出てきた社会科学的なアプローチ方法，等々，色々なアプローチ方法があり，形式化できないからといって悲観的になる必要もないと思った．

ただ，やはり形式的モデル化できるメリットは大きいので，たとえ適用分野が限られているモデルでも価値があると思う．汎用的でなくてもいいので，研究をしている方には，形式的モデル化を精力的に研究してもらいたい．

3. どの開発工程で議論しているか

プロセス開発，システム開発には，さまざまな工程とさまざまな成果物がある．今回のワークショップでの議論は，それらのうちのすべて，あるいは一部に関するものであったと思うが，いったいどの範囲までを対象にしているのかを明確にしておいた方が，議論に混乱が少なくなるのではなかろうか．例えば，「開発工程の要求仕様の記述を述べている」等々，のようにである．

4. ソフトウェア-データベース

（プロセス開発におけるデータ中心設計）

データがプロセスより安定なものであるならば，プロセス開発においてもデータ中心設計ができないかと考えるのは自然だと思う．それが，今回の「プロセス vs プロダクト」の議論だったのだろう．

開発プロセスについては，開発現場でも標準化等の形で整理されてきているが，開発プロセスで生じる仕様書やメモ等の成果物をどのように蓄積するかということは大きな課題として残っている．今回は，その点についての議論が少なかったように思う．

従来のウォーターフォール・モデルでもよいから，あるデータベース設計手法を用いて実際にソフトウェア開発のデータベースを示したらどうであろうか．それによって，プロセスのみからのアプローチよりもプロダクトが果たす役割がより明確になると思う．

新職人時代（居職考）

新保　修
（アセム・シー　有限会社　NIFTY-SERV PFC00121）

私がソフトウェア技術者となって、はや１６年になる。学校で習ったCOBOLが役に立って何とか御飯を食べてきた。しかし、ここ５年程で環境が激変してしまった。

対象となるコンピュータも大型汎用機やオフコンからパソコンやＷＳに変わった。言語もCOBOLからC言語[1]に変わった。システム開発の手法も目まぐるしく変わった。これらは、すべて私自身が経験したことだ。私の体験したことは、世の中の動きのごく一部に過ぎない。しかし、思えば遠くに来たものだ。紙テープ[2]の時代が懐かしいのは、年寄りの繰り言になってしまうだろう。

ところで、ＴＶの「大岡越前守」を思い出せば、長屋には大抵独りや二人の職人が居て、なにやら一身に働いていた。その職人について話してみよう。

皆さんは、「居職」という言葉を御存知だろうか。実は私もつい先日知ったのだが、住居と仕事場が同じ所にあることを「居職」と言うらしい。「居職」を行える人は当然職人である。近所の畳屋さんや桶屋さんを思い出してほしい。皆、住む所と仕事場が一緒の所にある。（いまどき、そんなの見たことないって。残念、少なくとも私の少年時代には東京の下町にだってあったのである。近所の子供達と一緒に木屑を貰いにいったものだ。勿論、夏休みや冬休みの工作の宿題の材料のためにである。）

ソフトウェア技術者といっても様々である。ＳＡ[3]、ＳＥ[4]，ＰＧ[5]といった分類やもっと細かな分類もある。あるいは、企業のＥＤＰ部門や大学等の研究機関に所属している人。コンピュータ・メーカーに所属している人。私のように流れ流れて小さなソフトウェア・ハウスに所属しているものもいる。

しかし、企業に帰属していると真のソフトウェア技術者足りえないのではないかと最近考えるようになってきた。なぜなら、企業の利益追求のためにやりたくない仕事もやらなければならない。それだけではなく、仕事の完成度も思うようにいかなくても仕事として済んでしまうことが多々あるのだ。（皆さんも、そういう経験ありませんか）

私の考える「職人」は、必要以上の仕事をせずに誇り高く暮らす人々のことである。納得のいかない仕事ばかりするのではなく。少しくらい儲からなくても、納得のいく仕事をする人々のことでもある。（左甚五郎もきっとそうに違いない。すくなくも、落語や浪曲の左甚五郎はそうである。）

企業に帰属しなければ、「ソフトウェア技術者」は「職人」になれると私は考えている。

最近、「エコロジー」という言葉がもてはやされているが、私流に「地球にやさしい」生き方のことだと理解している。つまり、必要最小限の暮らしで良いのではないかということだ。なにも、六本木に洒落たオフィスがなくても良いではないか。

ところで、最近「居職」の条件が整ってきたと思われる。（一昔前に在宅勤務は夢物語であった。）　何故なら、パソコンが安価になってきたし、パソコン通信も当たり前になってきた。この文章も自宅のパソコンで書いている位である。

しかし、結局の所様々な原因で「居職」は難しいのが現実である。だいたいにおいて個人の資格では仕事を発注してくれる企業は皆無である[6]。仕様については打ち合わせが欠かせない。だいたいが要求仕様さえはっきりしない仕事がある。勿論、そうした場合はお客様の所で汗をかいて見せるしかない。

SEAMAILには様々なソフトウェア技術に対する解決策が寄せられている。「居職」に対する問題点も、そうした中から前向きに進むと信じている。私の理想は、弁護士や税理士のように「ソフトウェア技術者」が認知されることである。知的財産の権利を本当に認められる世の中が来るならば未来は明るいであろう。

「居職」によって妻や子の信頼を取り戻そう。おとっちゃんの一所懸命働く姿を見せることにより可能である[7]。老人介護の問題もさしせまった問題である。これも「居職」によりある程度解決可能である。

私は、「ソフトウェア技術者」による「新職人時代」が来ると信じている。

---

[1] あろうことか、Ｃ言語やＸ－ＷＩＮＤＯＷの講師をおこなった事もある。SEAMAILの読者でＣ言語の勉強会をやりませんか。COBOLしか経験の無いあなた、連絡下さい。

[2] 紙テープ、紙カード、バッチ処理。懐かしい！！　絶句！！

[3] システム・アナリスト　何故かメーカーの営業もこう呼ばれている。

[4] システム・エンジニア　最近は猫も杓子もこう呼ばれる。

[5] 御存知プログラマー

[6] 私が現在の所に転職した一つの理由である。

[7] まず何より嫁さんを貰わぬことにはそうなる訳がない。

江南春紀行

熊谷 章

(PFU & 富士通)

1991年4月21日から僅か4日間であるが，文字通り"江南の春"を愉快な仲間達と訪れた．これはその時の遊記である．

4月21日（天気 晴）

朝早く逗留宿である南京東路20号にある和平飯店から上海駅に向けて出発した．上海からの目的地は南京である．時は春，日は朝，朝は7時，行く手は南京，連れはエキセントリックな米国人と日本人，それに中国人，条件は揃い過ぎるくらい満たされている．私の荷物鞄は，昨日外文書店で買求めた中国に関する本で重すぎてポーターたちの厄介物になっている．

上海駅から超特急列車に乗る．汽車には，軟座と硬座があり，予約していた席が硬座であったため軟座に変更しようと張先生が例の粘り腰で交渉したが満座でうまくいかない．汽車の内装はまるで日本の新幹線に似ている．一列5席で，2,3の席配列である．2階建ての構造になっており，効率よくたくさんの人々を運べるかを主テーマにしていることが一目瞭然だ．"日本の新幹線の悪しき影響が出ている"という声．実に様々な人々が乗り込んで来る．老若男女混在である．会社員風，田舎のおばさん，ブローカー風,..... 硬座であるせいか，外国人はあまり見かけない．

やがて，汽車は動きだした．蘇州，無錫を経由しての約4時間の旅である．向かいの席には山崎利治氏，岸田孝一氏がおり，私の隣には林香氏がいる．同じ列の反対側にはDr. David Notkin & Catherine Tuttleの夫妻と玉井哲雄氏がいる．次の列には，Dr. John Sayler, 伊野誠氏，佐藤圭氏，張然先生，そしてもう一人の張（育超）先生がいる．

上海駅を出て間もなく車窓が田園風景に変わる．見渡す限りの平野で遮るものは何もない．そのゆったりとした平面を実に見事な配色が為されている．一面の緑と一面の黄色が織りなすように様々な文様を描きだしている．緑が麦畑で，黄色が菜の花畑である．線路沿いには，決まって湖沼があり，線路に沿って樹林が植えられている．芽生いたばかりの樹林の新緑を透して見る緑と黄色が織りなす絨毯模様はなかなかのものだ．切れることのない湖沼では，アヒルや水牛がおり，中には魚を網でとっている人や子供らが群れをなして遊んでいる風景が見える．田畑を耕しているのは機械ではなく人であった．昔懐かしい鍬を使って一人で，大勢で働く姿が見れた．目を住居の方に移すと，万元戸らしい家が見える．二階建てが普通で，中には三階建てのものもある．張先生の説明によれば，農家では分家が許されていないので，子供が一緒の世帯に住むと二階建てになり，孫も一緒に住むと三階建てにするという．この気が遠くなるほど広い土地で，その政策を聞くと驚いてしまう．いかにも現代中国らしい面白い現象である．

処々に川があり，その河原もほとんど菜の花に覆われている．川を挟んだ両側が段々状になっており，河岸段丘のように見える．その段々を緑と黄色が流れているようだ．こうしてみると，全風景がある意味ではartificialである．日本では，何処かにnatureが存在し，artificialとnatureが共存している．が，ここ中国では一切がartificialで覆われている．これと同じ説明を本で読んだことがある．確か，ユリイカという雑誌だった．日本文化と中国文化の根本的な相違は，自然に接する態度で象徴的に表わされている，とあった．前者は，自然に融け込むような文化であるが，後者は一旦は自然を徹底的に破壊し人工的なものにした上で，そこに文化を築くという内容だった．こういう光景を目の当りにすると得心がいく．

水が切れることがない，のがよい．日本は，自らを秋津島瑞穂国と呼んできたが，これほど水が豊かではない．これらの湖沼もほとんどartificialである．水路，動物性蛋白質の供給源，植物の生命源，美的景観等の多目的な環境として，湖沼は正に中国5千年の知恵である．それらは環境に溶け込んで美しい田園風景の中心となっている．酒を飲みながら風景を見ていたら，知らず識らずの内に，次の詩が頭に浮かんできた．

水を渡りまた水を渡る
花を看また花を看る
.....
覚えず君が家に至る

途中の部分を思い出せないが，覚えず君が家に至るというくだりが手にとるように分かる．これは，高校時代に漢文

で習った漢詩の一つである．二十数年前の記憶が何故急に想起できるのだろうか．とても不思議な気がする．次に，向かいの岸田氏が"江南の春"を記述し皆に披露した．

千里鶯啼緑映紅
山村水郭酒旗風
南朝八百八十寺
多少楼台烟雨中

これは，杜牧の有名な作である．流石，彼の記憶力は天下一品である．時速 70KM 位の特急列車は，当に"江南の春"の気分の中を走る．覚えず，南京に至る，とまではいかない．

車中では，自然に酒盛りが始まる．上海ビールを張先生から戴いて飲む．次に，中国で発明されたという椰子の実ジュースにウィスキーを混ぜて飲む．そして，ついにはウィスキーをストレートで飲む．主な飲み手は，言わずと知れた山崎氏と熊谷である．話題は尽きない．例えば，線路沿いにある樹木名は何か．私は，カラマツ系統の木だ，と主張する．見れば，すぐ分かる（Seeing is proving）と馬鹿の一つ覚えを繰り返す．山崎氏は，納得がいかず他の木だと主張する．張先生に解を求めるがよく分からないという．次の話題は，ここらは水田地帯なのに，何故水田が全然ないのか．この問いには，明解な答えがあった．ここらは，二毛作で麦を収穫した後に水稲を植えるのだそうだ．いまは，麦の時期である．また，山崎氏，岸田氏，玉井氏の出会いの話，山崎氏の西域旅行の話等数え上げればきりがない．

蘇州の近くでは，斜塔で有名な虎丘を遠くに見ることができた．上海を出てから 2 時間位で列車は無錫に停車した．車内は禁煙なので，喫煙家は颯爽とホームへ駆け降り一服．私は，もう一人の張さんから無錫の名物が豚肉の煮込みだと教えられたので，酒のつまみに最適だと考え 10 元位でビニールパック入りの豚肉を買う．私は，この豚肉のお陰で大変苦しい思いをすることになったが，このときはまだ何も知る由がない．

無錫を出ても相変わらずの風景が続く．少し変化があったのは進行方向に向かって左側に小山が見えるようになったことだ．酔いで意識が朦朧とする中エキセントリック集団の会話は続く．途中から玉井氏も会話に加わった．また，昨日外文書店で購入した中国詩 101 選（対英文訳付）を読んだりした．今回のワークショップで，私の発表したテーマが lyric expression に言及していたため，"101 Chines Lyrics"の本を見付けて嬉しくなり即座に買ったのだ．中国詩を英語で読むと分かり易い．平易なボキャブラリを使っているからだろうか．中国通で語学に堪能な岸田氏も次のように言っている．難解であると言われている中国の古文も英語の翻訳されたものを読めば，なるほどと分かるものが多い．これは，日本漢学者たちが本来はそう難しくないことでも，日本語に翻訳する時に殊更難しい表現をとったことが原因しているのではないか．これから，新解釈に基づいた中国文の日本語への翻訳が必要だ，と．という話題を肴に，先に求めた豚肉を食べ酒盛りは続く．この" 101 Chines Lyrics"に収められている詩は，ほとんど日本人に馴染み深いものではない．英訳をしたのは，Foreign Language Institude in Beijing の Chu Dagao 教授である．参考までに，私が気に入った詩とその英文対訳を紹介する．

臨江仙（蘇軾）

夜飲東坡醒復醉
帰来彷彿三更
家童鼻息已雷鳴
敲門都不応
椅杖聴江聲

長恨此身非我有
何時忘卻営営
夜闌風静穀紋平
小舟従此逝
江海寄餘生

After Drinking (Su Shi)

Tonight I drank at East Slope,
-- sobered once, drunk again.
On my return, It seems to be the third watch;
My boy-servant already snoring laud,
No one answers my knocking.
Leaning upon my stick,
I listen to the sound of the river.

Often I regret that my life is not my own;
When can I become oblivious of worldly affairs?
The night deep, wind silent, waves calm --
O that I might go in a small boat,
To spend the rest of my life
on the rivers and seas!

我々の席の反対側に陣取っている Notkin 夫妻には一種

独特の雰囲気がある．Davidは，モダンアメリカンの代表的な学者である．ジョークが好きなナイスガイである．後ろ髪を結わえて尻尾のようにしていることを私に自慢気に見せてくれた．彼の連れであるCatherineは，風情が落着いていて，学者風でlyricに溢れた素敵な婦人である．Davidに言わせれば，彼女は"independent person"だそうだ．眼差しが柔らかくロマンチストに見える．日本語ができるらしいが，ほとんど話さない．都市建築家であり，一人でスケッチしたりメモを取っていることが多い．二人とも，アジアに興味を抱いているようである．日本人，中国人，アメリカ人の団体旅行は実に愉快である．これが，交通であり，コミュニケーションである，と感じる．人生とは旅であり邂逅である，と言った故人がいたが，正に邂逅の妙味を堪能しながら汽車は目的地南京に向かう．

漸く南京に着いた．毎日の深酒に今日の昼酒でもう這う這うの体である．それに，ここではポーターがいないのであの重い荷物鞄を自分で運ばねばならない．一気に痩せる思いがした．駅前には，大きな篠懸の木があった．英語で，篠懸の木を何と呼ぶか，話題になった．Catherineが，多分"plane tree"というと教えてくれた．

江蘇計算機開発中心から出迎えの人が2人来ていた．一人は，小柄な人のよさそうな男性で名前を唐学勇といい，もう一人はすっと背の高い夏目雅子風の美人の若い女性であった．彼等の案内で南京の宿，南京飯店に向かう．

南京市内は，今までいったことがある中国の街とは様子が違っている．一口で言えば農村都市という感じである．一番近い雰囲気を持っているのが杭州だと思った．宿までの道中，中国の夏目雅子，陶金梅さんと一緒だった．次の日に予定されているワークショップの観衆が約150名集めっていると聞いた．一緒にいた張先生が，それは凄い，上海ではそんなに沢山の人を集めることをとてもできない，と言った．南京では人々の計算機に関する興味がとても強いらしい．

南京飯店は，古風なヨーロッパ風の建物である．晩餐までの小一時間を一人で市内散策に費やした．南京飯店が面している中山北路を玄武湖の方へ歩いた．この中山北路が見事に美しい．日本風に言えば，4車線で，往路，復路共に2車線である．一番外側が，自転車専用路で，中の2車線が自動車専用である．そして，一番外側に広い舗道がある．歩道と自転車道，自転車道と車道の間には樹齢200 - 300年になろうとする篠懸の街路樹がある．だから，4筋の鈴掛の街路樹が続いていて，その周囲に6本の道が続いている．総ての道はこの街路樹の緑のアーケードで覆われている．夕方の6時頃なので，彼方此方に明かりが燈り始めた．往来の自動車と自転車の量は，上海のような凄さはないが，繁華街であることを示し喧噪である．路を歩いていると，上空から黄緑色の柔らかいものが降ってくる．はらはらと降りてくる．折からの灯りに照らされて，それは幻想的な状況を醸し出している．淡く哀しきものの降るなり，― それらは，まるで初夏の夕闇の中を蜉蝣が飛び交っているようだ．奇麗だ．抒情に身体が包まれる．抒情は水路だ，安らぎだ．Lyricは Beliefだ．

そんな中で，客寄せの姐さんたちに会う．多分，個人経営の小さな酒家の呼び込みである．舗道に椅子を並べて座っており，客人らしき通行人が来れば何やら話かけて誘う．歳の頃は20前後の若き姐さんで，中国でこのような光景に出会ったことがなかったので面白かった．また，随所でバトミントンを興じている姿を見掛けた．街角には，水果を売っている屋台があった．林檎，みかん，なし，西瓜などを並べている．店番にいるのは大抵は老夫婦である．彼等の顔は日に焼けてシワが多いが，その表情がとても穏やかである．つい，日本の老人ホームに入っている小奇麗な植物のような表情をした老人たちと比較してしまう．中国の老人たちは今を生きているが，日本の老人たちは過去に生きている，そんな気がしてならない．そして，私はいまからはみ出ている異邦人である．

中山北路の大きな辻で amazingな人物に出会った．その人は，80歳位のおじいさんで，街角で椅子に坐り，左手に掌大のラジオを持ち，そのラジオを耳にぴったりくっつけて何かを聴いていた．傍によってみたら，それは何かのスポーツ中継だと分かった．中国語が不明であるが，多分ボクシング中継だと推測できた．そのおじいさんの隣には，体重計が置いてあった．体重計には，車がついており，運搬できるようになっていた．私は，漸く呑み込めた．彼は，体重を計る商売をこの街角でしているのだ．誰も客はいないし，また来そうにないが，彼は一向にそんなことを気にせず一心にラジオを聴いている．顔は三船敏郎に似ている．ラジオを聴き，遠くを見つめている．その表情を眺めて，何処かで見たことがあると思った．そうだ，上野動物園で見たいまは亡きボンゴに似ているのだ．少しも動かずじっと遠くを見つめて動かないボンゴを私は好きだった．美しく沈黙に生きるボンゴに近寄り難い気高さを感じていた．この体重計のおじいさんにも同じものを見た．

そのすぐ脇では，新聞紙を広げた上で切手の売買が行われていた．そういえば，30 数年前切手収集が日本でも流行った．10 数人の人が数枚の切手を囲んで何やら話し合っていた．反対側の辻には，ラーメン屋，煮卵を売っている人，野菜を売っている人達が大道で商売をしていた．そこらをブラブラした後，中山北路を引き返したら，先ほどのおじいさんが帰り支度をしていた．朝にやって来て，一日ラジオを聴いて，夜になれば体重計を引き引き家路に着くのであろう．それは，一幅の人生だ．私が，日本に帰り，この駄文を書いているいまでも，あの南京の街角にはあのおじいさんが，生ある限り，体重計を横に左手にラジオを抱えて坐っていることだろう．彼は，私がそのことを知っているとは知らない．だが，私は彼のことを知っている．この他愛のないことが何故か私を昂奮させてくれる．東京と南京という異なった空間に，共時的な関係が張られたような気分になる．知っている，とはこういうことなのかも知れない．あの名前も知らないおじいさんを核として，私の南京の思いは広がる．一切の南京の思いは，体重計のおじいさんをシンボルとして豊饒になる．これも抒情のなせる技か．

この日の晩餐を皆で南京飯店で採った．例によってビールと紹興酒を飲んだ．菜を沢山食べたがあまり記憶にない．最後に食べたのが，麻婆豆腐と炒飯である．久し振りの米だったので，麻婆豆腐をたくさん掛けて炒飯を 2 皿食べた．

食事の後で，数人で夜の中山北路を散歩した．丁度先ほど独りで散歩した場所まで行き引き返した．途中で路上で売っているゆで卵を買う．早速，誰かが食べたがその瞬間奇声をあげた．これは，卵ではない．殻の中で雛に孵っているぞ．食べると，ふにゃっという感触があり，噛むと口に中に骨が残るらしい．騙されたね，という話をした．しかし，あとで聞けば南京の人はこの雛入りゆで卵が好物だという．身体によい食べ物で妊婦などはこれをよく食べるらしい．因にこの奇妙な卵を平気な顔でバリバリ食べたのは佐藤圭氏だけである．彼は，中国のあちこちで完全に中国人と間違えられた人物である．飛行機の中でも，討論会の会場でも中国人は彼に中国語で話かけるのだ．それで，彼が全部食べる理由もうなづけるというものだ．

南京飯店に戻ってから，ホテルの土産屋を少しからかった．そこで，Dr. John Sayler が Four Roses を買った．それを彼の部屋で飲もうと約束をして部屋に帰った．一息入れてから飲みにいこうとベッドに横になった．これから生まれて初めての苦しみが始まったのだった．

小一時間もしない内に胸の辺りが苦しくなってきた．その内あっという間に気分が悪くなり，トイレに駆け込み，吐いた．喉元を食べた物が逆流するとき，食べた時と同じ味がした．麻婆豆腐の辣油，炒飯，無錫の豚肉の味が主だった．中でも無錫の豚肉の脂の味が頭に響いた．あれが引き金かと思った．雑多な物を吐きながら，得体が知れない何かをも一緒に吐き出している気分になった．

これで，何かが確実に変わる，そんな信念らしきものが身内に湧いて来た．南京の妖怪な魔力が私に作用し，腐れきった現代文明に犯された身体から悪い癌を吐き出させているようだ．吐きながら，これで確実に変われるという安堵があった．しかし，どこからどこへ，どのように変わるというのだろう．苦しみのあまり，目から涙が出始めた．心の中で，誰かが吐け，吐けと叫んでいるようだ．冷蔵庫からミネラルウオータを取出し，やっとのことで正露丸を数粒飲んで横になった．

今度は，腹痛がし始めた．少ししたら，二度目の嘔吐が襲ってきた．こんどのは本格的な嘔吐で胃液まで全部吐いた気がした．これで，すっかり別の人間になれた，と思った．今度ははっきりと，正露丸の味覚を味わえた．吐瀉物とトイレットペーパのためトイレが詰まってしまった．いまは，下痢していないが下痢が始まったらどうしようと，苦しい中でそんなことが頭を掠めた．

腹痛があまりひどいので，ブスコパンを 2 錠飲んだ．横になると，下になった部分が焼けるように痛い．身を屈めて丸くなり，痛さを堪えながらじっと回復を待つ．蛹から昆虫が生まれるような気分である．変身の時が来たのだ．それにしても，南京の魔力はどこからやってきたのだろうか．

4 月 22 日（天気　晴）

朝方，下痢を催したので，自力でトイレの詰まりを直し用を済ます．嘔吐に比べれば下痢の症状は大したことがなかった．それにしても，腹痛のため一晩のたうち廻ったのは生まれて初めてである．ほとんど眠っていない．朝 5 時ころから明るくなり始めた．ホテルの庭園をみたら，従業員らしい人々が歩いていた．鳥の囀りが聴こえてきた．昨年ブダペストで聴いたリゴーという鳥の啼き声によく似ている．そういえば，ここ南京とブダペストは陸続きで繋がっているアジア大陸だと実感した．自然が美しい豊かな農業国という形容は，南京とブダペストにピッタリである．

そんなことを考えながら，うとうとしていたら岸田氏から電話があり，OHP 用紙とペンを貸して欲しいという．昨夜来腹痛でひどい目にあったと言ったら，林香氏も同じ現象である，という．さては全員かと思ったが，病気に陥ったのはこの 2 人だけであった．我々 2 人を残して，皆は講演とパネル討論へと出掛けた．

私は，大分調子がよくなってきたので，時折林氏に電話して，容体を聞いたが彼は一向に駄目らしい．ブスコバンを持っていって上げたりした．仕方がないので，部屋で独り横になりながら，上海外文書店で求めた "China's Minority Nationalities"という本を読んだ．中国には，56 に及ぶ民族がいること，その民族の分布，人口，習慣，言葉，民族の特徴，顔写真等が分かって興味深い本であった．全人口の 93.3% 近くが漢人であることに，今更ながら驚いた．因に，人口の多い順に民族名を 5 つ挙げると次のリストができる．Han (漢),Zhuang, Hui, Uygur, Yi. Zhuang が全人口の 1% を超えているが，後は皆 0.5% 前後である．清王朝を支配した満州族は，僅か 0.43% しかいないことに改めて驚く．写真や説明を読み，私の興味はやはり西域の民族に集中した．例を挙げれば，チベット，ウイグル，カザク，キルギス，ウズベク，タタール等の民族である．彼らの写真を見ていると，何か得体の知れないものを感じてしまう．

11 時近く，ドアをノックする者がいた．誰かと思いドアを開ければ，張先生が"具合はいかが？"と流暢な日本語で入って来た．その後ろには，カメラを構えた岸田氏がおり，更にその後ろに颯爽としたビジネスレディがアタッシュケースを持って立っていた．中国の医者を連れてきたのだった．

林氏は既に看て貰い，消化不良という診断だったらしい．彼は，少し熱があるらしい．その女医は，女優佐藤友美によく似ていた．彼女は，まるで汚らしいものを見るような目つきでハスに構えて私を見た．私は，張先生に，もう直りつつあるから診察は不要である，と訴えたが，診察してもらった方がよい，と押し切られた．一瞥の後，彼女は体温計を私の舌の下に入れた．中国では，舌の下で体温を計る．最初，舌の上で計ろうとしたら，下にするように指示された．幸い，私は熱が高くない．薬を呉る，というから，即座に私は言った．西洋の薬なら要らない，東洋の薬なら呑むと．漢方がベースであるという黄色の錠剤を貰った．素直にこういう発言をするとは，昨夜の嘔吐で私は似而非東洋人から真の東洋人に生まれ変わったのかも知れない．

張先生の勧めで，お粥と林檎とみかんをルームサービスでとって貰った．お粥の菜には，なんと懐かしい"チョロゲ"の味噌漬が出て来た．2-3 センチの大きさで瓢箪を捩じったような形をした植物の根である．昔，田舎の柿木の下で，茸きに似た葉っぱを掘起しこの"チョロゲ"を取ったことやそれをやはり味噌漬にして食べたことを思い出した．ルーツはやはり江南にあるかと＿ チョロゲをかじり粥をすすり，秋田の田舎で暮している年老いた父母を思いやる．いまでは，田舎で"チョロゲ"を栽培している家はないだろうと想像する．

たらちねの父母の顔するチョロゲ哉

りんごもみかんも見てくれはよくないがとても美味い．りんごは私のこぶしよりやや小さく，青い．日本の早生りんごとインドりんごの合いの子の味がした．みかんは，ポンカンに似ており，味はポンカンとオレンジの両方の味がして，とてもジュウシーであった．チョロゲ，りんご，みかんを食べて少し休んだら，東洋人として再生した私はすっかり回復した．こうしては，いられない．街に出よう，という意識が強くなり，林氏を誘うが再生とは程遠い状況にいる彼はまだベッドでのたうち廻っている．独りでカメラを持って出掛けた．時刻は午後 4 時頃であった．

街に出るとそこは初夏の陽射しが輝いていた．南京飯店は由緒正しい古いホテルらしく，手入れが行き届いていた．庭園にはきれいな花が咲いていた．ホテルの入り口近くには，自転車置場があり，そこには車庫番のお爺さんがひねもすのたりしている．昨日歩いた中山北路を南に少し歩く．そして，最初の三叉路を左に玄武湖の方へ曲がる．この路は，街路樹が極めて小さかったりないところがあるので新しいようだ．広さも 2 車線くらいの普通の通りである．名前を新模范馬路というらしい．舗道と車道とは段違いになっており，分かれている．

汗ばむ程の陽気である．私の出で立ちは，白のジーパン，青のチェックのカラーシャツ，紺ブレザーそれにハンガリー製の Rockport の靴である．日本を出る時に出ていた醜いお腹が真の東洋人のなったお蔭でなくなりスマートになった，積もりでいる．路上で沢山の南京の人々にあった．自転車修理工達，建具を作るためにカンナを引いている大工達，徒党を組んで下校する学童達，リヤカーで荷物を運んでいる人，駄菓子やアイスクリームを売っている店番のお婆さん，工場から出入りする男女，＿ 素顔の南京に会えた．皆日焼けしていて健康である．人民服も結構多いが，シャ

ツとズボンがほとんどである．私は，一つ一つを懐しむような思いで見た．それらの光景は，まさしく私が 30 数年前に日本の秋田で経験した光景に酷似している．庭で飼われている鶏，自転車とリヤカーの生活，大工仕事をじっと見つめている老若男女，およそきれいとは外交辞令でもいえない汚れた家屋，埃っぽい田舎じみた匂い，.... まるで，昨日のことのようだ．

今日は不思議に異邦人の感覚がない．傍を女子小学生が歩いている．全員赤いネッカチーフを首に巻いているのが洒落ていてとても可愛い．その児らは手をつなぎながらアイスキャンデーを美味しそうにしゃぶっている．その脇では，中学生と思しき 3 人連れが自転車に乗って鬼ごっこをやっている．この中学生が乗っている自転車は随分とよい品だと一目でわかる．働いている人らが乗っている自転車より遥かに高価そうだ．南京にも親バカがいる．街並は，工場や住居があるが店がほとんどない．野卑た田舎街の感が強い．

15 分くらい歩いたら玄武湖の周囲を巡っている中央路に出合った．この道は広く 4 車線あり，路の両側には樹齢 200 年位の鈴掛の街路樹が立ち並んでいる．自動車の往来が頻繁で主要道路だとすぐに分かる．舗道の幅が 6 メートルくらいの立派な途だ．この路沿いには，いろいろな店が立ち並んでいる．この通りは日本にないエキゾチックな雰囲気を醸し出している．明るい高い樹，白壁の続く途，小さな店の並び，人通りのない静かさなどがその原因であろう．その中をぷらりぷらりと気の赴くままに逍遥する．

やがて，前方に賑々しい様子が見えた．玄武湖の入口の玄武門に違いないと思ったが，近づいて見たら南京天覧中心であった．とてつもなく広い場所に見るからに権威を誇っているような厳しい建物だ．午後 4 時半過ぎであったが既に南京天覧中心は閉まっていた．しかし，皆塀を乗り越えて中に入って写真を取っていた．天覧中心の門の前には，屋台風の出店がある．お茶，麺，西瓜，コーラ，ファンタ，饅頭，アイスクリームを売っている．40 分も歩いたので何か飲みたいが，昨夜の痛みを思い起こし我慢する．ここらが意気地のないところだ．少し，向こうで黒山の人だかりがあり大いに賑っている．行って見たら銃で風船を割る大道ゲームである．20 個ほどの風船を布に留めておき，3 メートルほどの距離から空気銃のような銃で打つのである．玉は丸くなく針金でできた針のようなものだった．景品は何もなく，ただお金を払って風船を割るだけであるが皆物凄く楽しんでいる．私には，何がそんなに面白いのかよく分からなかった．

魅惑的な西瓜売りの脇を通り抜けて城壁に至る．中国の都市の象徴が城壁である．ここの城壁は西安のそれに比べると規模が小さい．高さ約 10 メートル，幅約 8 メートル位か？ 5 角ほどの入場料を支払い玄武湖公園に入る．黒い玄武門を抜けるとそこには明るい湖面が広がっていた．湖岸に腰を下ろし休んだ．湖岸沿いに若芽が芽生いている美しい柳の並木がある．水と新緑が見事に調和している．一連の樹影が水面に写し出されている．ふと，周囲を見れば近郊からやってきたような観光客がいる．少し遠くには彼等のバスが停まっている．そのバスからは，中国の歌謡が流れ出ている．ゆったりとしたおおらかな安堵感を感じた．

玄武湖には 4 つの島がある．杯（中国語では木へんではなく玉へん）洲，梁洲，翠洲，菱洲である．実は，杯洲の中にはもう一つ桜洲という島がある．池の中に池を作り，その中にまた島を作ったのである．中国の人はこのような入れ子が大好きだ．ALGOL のスコープのような概念が好きでリカーシブな造形をよく造る．杭州の西湖には多重になった池がある．彼らは artificial を心から悦ぶようである．

反対側をみれば，丘の上に五重の塔が見える．それは，城壁の上の方に聳え立っている．塔も中国人の趣味である．近くには雨花石を売っている店があった．雨花石はこの地方の名産である．サイケデリックな極彩色をした丸い石である．大きさは，碁石くらいから子供の掌くらいまでのものが多い．値段は一個数百元から一山数十元にいたるまで幅が広い．この石を水を容器に入れて色々な角度からながめ微妙に変化するその紋様を楽しむらしい．どこか世間離れしていて，哲学的で，無為の香りがして私好みであるが買い求めなかった．ピッタリ来るものが無かったのだ．それに触ることができないのが玉としては寂しい．ここから南京飯店へ戻ることも考えたが，真の東洋人である身となってはそんな情けないことはできない．玄武湖をグルリと廻って帰ることにした．

湖畔には，遊歩道が完備されていた．その周りにはいくつものベンチも用意されている．ベンチは，ほとんど男女のカップルで占領されている．若いのもあれば壮年のものもある．湖畔の途を独り歩く．その内空いているベンチを見付け坐る．隣のベンチには若いカップルが食べ物と飲み物を持って来て寛いでいる．時々彼等を眺めながら，自分の来し方行く末を思う．ベンチは大きな樹で覆われており，

その梢が湖面まで伸びている．舗道とベンチは皆石でできている．なかなか見事な風景である．

ふいに背後からある会話が聞こえてきた．見ると壮年の二人連れがある植物を指して何やら話しているのだ．それに引き釣られて周囲を見渡せば，そこの植物の生態系は日本のそれに酷似している，ことに気付いた．草本，雑草，潅木，樹などいずれも馴染み深いものばかりである．中でも，いま花が咲いている海棠，桐，花蘇芳が印象的だった．

海棠，鎌倉，街道，木瓜，バラ，仏教，寺，出家，弟子，庫裡，木の目，残り陽，日向の匂い,....

桐の花，北原白秋，ハイカラー，梅，黄金色，抒情，水郷，柳川，立花亭，禁断，異国，童謡,....

花蘇芳，桜，初春，陽炎，田舎，庭，藤，河，河原，尾花，青春,....

という連想を楽しむ．

ベンチのすぐ下の草を見ればこれも覚えがある．日本でも垣根のしたによく植えられもので，深緑で長さ 15cm ほどのテカテカ光る幅 5mm ほどの草である．これは青い実を付ける．小学生のころこの青い実をよく探したことを思い出した．一体私は植物を好きだ．一匹の小さな昆虫ほど宇宙の精密さを表現しているものはない，とどこかで読んだ．この点は植物も同じだ．通勤途中で道端のスミレ，タンポポ，ハナミズキやトガゲに時間を費やし遅刻したことが幾らあったことか．これらのものに星辰の動きを見るのは古来からの常套手段である．

何かの拍子で，竹西寛子氏の木賊と人の情をテーマにしたエッセイを思い出す．木賊も懐かしい草である．田舎の裏庭の隅にあった．筒状で表面がギザギザしており，地面から垂直にすっと生えている．見た目に涼しい．しかし，その根はいくら切っても切れないほど地下深く縦横に伸びている．人間の業の深さと木賊の根の張り方の関係を描いた心に残るものであった．この木賊を採って来てヤスリ代わりに鉛筆のシンを削ったがうまくいかなかったことがあった．その時木賊の根の凄さを知ったのだった．

そんな思いに耽っているうちに隣のカップルがいなくなった．私も動物園の方へ向かうことにした．途中太鼓橋を渡る．動物園の前には，生きた植込みで作った巨大なドラゴンが飾られていた．このような人をぎょっとさせるデコレーションも中国人の愛するところである．動物園の入口の傍には水鳥の園がありたくさんの色とりどりの鳥が戯れていた．客はほとんどいないようだ．

植込みの中の小途を歩いていたら，50cm ほどの釣り上げられたと思しき魚と寛ぎ中の若いカップルに出合った．魚は死んでいるようだった．大抵の人が目を向けるが持って帰ったり，池に放ったりする人はいない．植込みの陰の若いカップルは人目を偲んで逢引している風情でなかなかよろしい．長い堤のような湖の中の路を抜けて城壁の一部である解放門に至る．城壁の上に登る入口を少し探したが見当たらない．

門をくぐり坂道を下る．煉瓦でできた中国特有の住居が続き，幾らか多目の人々と会う．線香の匂いがし何処からかお経と鐘の音が聞こえてくる．見るとここは鴐鳴寺というお寺である．門前の店はもうすでに閉まっていた．やはり，中国の大衆の間には仏教が根付いているのか，と思う．3 年前に，無錫の太湖湖畔の公園内にあったお寺を思い出した．詣でた人々が全員でお経を合唱しており，その声が近くの竹林に響いていたのが忘れられない．この時，何かいまの日本に欠けているものがここにはある，と実感したものだった．その感じが南京にもあったのだ．

鴐鳴寺路を抜けて北京東路に出る．この路は先ほどの中央路と同じ広さであるが，街路樹が針葉樹である．見たところカラマツ系だと思った．唐松と書くからきっと当たっているに違いない．道路沿いに色々な店がある．西洋レストラン，酒家，デスコなどが見受けられた．当然，ラーメン屋や雑貨屋もあった．麺一杯一元もしない．路上には，靴修理屋や自転車修理屋がいた．多くは女性だったのが印象的だ．路が丁度真西に向かっているので沈みつつある太陽が街路樹の真ん中に見える．赤い夕陽が差し込んで来て，北京東路の街路樹に囲まれた通りを美しく飾っている．空気が少しもやっていて不思議な国を訪れている感じだ．

人通りは多くはない．一分間歩いて数人に合う位である．周囲の建物は立派なものが多く多分政府関係のものだろう，と推測できる．工芸美術大楼，南京工学院，市財政局，市税務書，江蘇電視台などがあった．やがて，喧噪な五叉路に出た．流石にここらは人出が凄い．北京東路，中央路，中山路，北京西路，中山北路が交差している場所である．バスターミナルにもなっており，大勢の人々が行き交っている．タクシーもあった．ここでは，バイクにリヤカーの小型を付けたのがタクシーである．オープンカーなので，気持ちがよさそうだ．一瞬，乗ろうと考えたが，止めた．

その角には大きな看板があり，電信大楼と書いてあった

気がする．路上では大道商人が沢山いる．日が暮れたので薄暗闇であるが，路上に敷いた新聞紙の上にいろいろなものをおいて商売している．雨花石，日常雑貨品，パンツなどである．中でも驚いたのは，破れた新聞紙の切れはしに靴を一足置いて売っている小母さんである．きっと亭主の靴を売りに来ているに違いない．何かの事情であの靴が売れなければ困るのだろう．ここではなんでも売れる，と思った．

中山北路はすぐに見つかった．四列の鈴掛の街路樹が目印なのだ．デパート，水菓店，酒家，麺屋などを横目にみて一路中山北路を上る．何と言ってもこの路はきれいだ．昨夕の感激が蘇ってくる．木樹の上からはやはり薄緑色の若芽が燦々と舞落ちてくる．自転車と車は頻繁に走っているが人通りが少なくなってきた．前方に，白壁が 100m 以上続く通りが見えた．さっきの落日に勝とも劣らない風景である．暫し佇みその風景に茫然とする．暫く歩いてその塀の門に至り，学校だと分かる．

その門を少し，過ぎたところで面白い光景に出合った．それは絶え間なく流れている自転車の群れの中で，ハンドルの上に本を広げて読みながら運転している青年の姿である．ほぼ熟読に近く周囲に全然注意を払っている気配がない．周りの人々は迷惑そうに避けて走っているが，さほど問題にはしていない．流石，自転車と勤勉の国中国であると唸った．さしずめ，中国版二宮金次郎といったところか．後で，この光景を見たと張先生に話したらその反応が凄かった．"見ましたか！！　実は，私はまだ見たことがないので一度みたいと思っていました．それは凄い．""上海ではそんなことはできません．多分，北京にもいないでしょう．"この件で，自転車が中国の人の手足にまで進化していることがよく分かる．

次にもっと驚く人物に出合った．それは，云南路と中山北路が出合う辺りだが，両足の膝から下がない小母さんが膝に丸いゴム靴を付けて市中を歩いているのだ．最初見た時，誰かが転倒しているのかと錯覚した程である．農業をやっているらしい出で立ちである．頭は日本で言う頬被りをしており，日焼けしていた．周囲の雰囲気からそこはダウンタウンであることが一目瞭然である．両足切断，それはどんな事件があったのだろう．日本でもそのような人はいるだろうが，このような市中を膝靴をつけて堂々とあるいている人がいるとは聞いたことがない．何というたくましさだろう，と感心してしまった．それに比べ，飲み過ぎ食べ過ぎでベッドで唸っているとはなんとだらしないことか．お前はアマちゃんだ！世間識らずだ！軟弱だ！似而非技術者だ！似而非ヒューマニストだ！臆病な自尊心の塊だ！現実の前では一溜まりももないだろう！といった声が聞こえてくる．しかし，待て．俺は昨夜生まれ変わったのだ．身体の何処かに拠り所となる核が微かにあることが分かる．そこを中心に生きて行けるのだ．

やがて，昨夕あのお爺さんに出会った辻にやってきた．今日は，遅いのでその場所におじいさんはいなかった．すぐ傍では，相変わらず，切手の売買は行われていた．ここからは勝手知ったる路である．例の呼び込みがいる酒家の前を通って我が投宿中の南京飯店に着く．時刻は 8 時頃である．

早速，ベッドで這い回っているであろう林香氏を訪ねた．熱は下がったがまだ胃腸の調子がいま一のようだ．軟弱にも薬に頼って回復しようとしている．全部吐いてすっきりすればよいのだ．ここ南京には，魔性の力が働いているのだ．それを堪能しない手はないのだ．といっても彼には通用しない．他の連中は今晩は江蘇計算機中心主催の歓迎レセプションで帰りが遅いので，二人で晩餐をホテル内の西餐で取ることにした．

客は誰もいなかった．ウェイターに注文するが要領を得ない．英語が駄目らしい．スパゲッティを注文する．その外に果物が欲しいと言うが，彼の返事はメニューの中のとんでもないところを指している．これはだめだ，と諦める．そこへ，レセプション帰りの面々がやって来た．岸田孝一氏，張然氏，それにもう一人の張氏である．様子からして，相当酩酊していることが分かる．張先生に，果物がないといったが本当にないのかチェックして貰ったら，ウェイターはあるという．中国語ならいいのかと思っていたら，少しして何もないという返事に変わった．どうしょうもない奴だと思った．そしたら，もう一人の張氏が外の水果屋までいって買求めて来るという．"不要，不要"といったが彼等は一度言い出したら実行してしまう．スパゲッティを食べている内に，彼は洋梨を 4 個買って来た．西瓜は，"無要"らしい．

林氏の部屋に戻り，岸田氏からその日の様子を伺う．しかし，さっきからどうも様子が変である．酔っ払っているのは確かであるが，単なる酔い方ではない．彼も南京の魔性に取り憑かれたのか，とさえ思った．ミネラルウオータは飲むは，さっき戴いた洋梨を食べるは，身体が燃えるから散歩に行こうとかのたまうのだ．聞けば，歓迎レセプショ

ンでの中国版夏目雅子こと陶さんの凄さにやられたということだった．

八頭身で色白で痩身の彼女が名うての酒豪だという．彼女に日本陣が乾杯を挑まれ，最後まで闘えたのが岸田氏と山崎氏だという．彼女は，立て続けに白酒を 6 - 8 杯乾杯するのだそうだ．例の中国式乾杯である．中国版夏目雅子と白酒の乾杯が原因らしい．次の日の朝，山崎氏から次の台詞を聞いた．"昨夜は，夢か現か定かではないが，美女と酒に悩まされて寝ていたか起きていたかよく覚えていない"その内，岸田氏は林氏の部屋のベッドの上で熟睡に入ってしまった．二人で無理矢理起こし，各自部屋に帰る．風呂に入り，洋梨を一個食べる．残る一個を冷蔵庫に入れ，朝食にする積もりだ．

4月 23日（天気　晴）

朝食にお茶と昨夜の洋梨を採って集合時刻の午前 7 時にロビーに行った．そこには，話題になった陶さんが既に来ていた．見れば，二日酔いの雰囲気もなく普通の爽やかな顔をしている．赤と黒，スタンダールで装いを決めている．彼女は，寡黙にして多くを語らない．物憂いに富んだ表情でうつむいている時が多い．この人が，白酒を続け様に乾杯するとはとても想像できない．

この日は，南京大学，太平天国天王府，夫子廟 - 秦淮（貢院街），中山陵，明孝陵，雨花台，南京長江大橋を訪問する予定である．

南京大学では，シンガポールの第 12 会 ICSE で，岸田孝一氏のキーノートを聞いたという教授が我々を迎えてくれた．南京大学は，とても明るい感じの大学で，多くの外国人学生を見受けられた．アメリカ人が多いようだ．一通りの説明を受けた後，コンピュータ関係の研究室を廻った．刺繍のデザインをコンピュータで支援するシステムを見た．スキャナーからデザインの原型を読み込ませて，そのイメージをベースとしてベクトルで紋様を決めていくシステムである．例として，南京大学のロゴを読み込ませ，T シャツのマークを作成するところを見せてくれた．パソコンで実現している実用システムということだ．次は，日中翻訳システムを見た．これは，SUN-WS を使って作成していた．簡単な日本語を入力すれば，それに対応した中国語が出力されてくるものである．対話的なのがよい．Prolog で作成すれば，簡単な翻訳を実現できるが，ここのシステムはどの程度本格的なのか，聞いたがその詳細を忘れてしまった．それから，もう一箇所ネットワークのゲートウェイか何かのハードウェアに関する研究も見せて貰った．ほかの大学の研究に比較すれば，実用的で面白いアプローチをしていると思った．

これは，あとで上海オムロンソフトの包さんに聞いたことだが，南京大学は中国で，コンピュータを専門とした教授を出したのが初めての大学だそうだ．ほかの大学のコンピュータ系の教授は，数学や電気等のコンピュータ以外の専門から移った人がほとんどという．それだけ，コンピュータに関する熱意が南京では強いという証明らしい．そういう包さんも南京大学卒業だという．

次いで，太平天国天王府へ行った．ここは，かの有名な太平天国の本拠地があったところであり，また辛亥革命で孫文が清朝を倒し中華民国をつくった，中国民主主義革命の本拠地でもある．いまは，公園になっており多くの観光客で賑わっていた．孫文は民衆に圧倒的人気があるようだ．ここの庭園は，典型的中国の様式である．孫文が使った部屋や彼の書いた綱領などを見る．三民主義のなんたるかをかの地にて見て，少し心が踊る．国内諸民族の平等と外国からの独立を意味する民族主義，民主制の実現である民権主義，それに平均地権と資本節制を説く民生主義．日本の政治家に見られない，スケールの大きさを感じた．そういえば，孫文は医学を専攻したが，フィジカルを対象とする医者にはならず，メンタルを対象とする思想家になったのである．彼は，ほとんどを日本，アメリカ，イギリス，アジア等の外国で暮らし，ずっと革命活動を続けた．イギリスで，清朝大使に逮捕されてから，釈放になるまでの件はまさに劇的の一語に尽きる．この事件は，如何に国際的な友人関係が重要であるかを物語っている．彼は，イギリス人であった彼の師と世界の世論によって釈放されたのである．それは，ついこの前起きたソ連のクーデターとゴルバチョフの軟禁とその釈放事件と同じである．記念館の前の孫文の胸像の周りは記念写真をとる民衆でごった返していた．中国では，孫文を"国父"と形容するという．分かるような気がする．

太平天国の記念館も少し見た．こちらには，悲哀に満ちた物語がたくさんありそうだ．私の少ない知識では，太平天国陣営は洪秀全を筆頭に内部から腐敗し崩壊していったという記憶がある．何となく，現代の新興宗教に似た雰囲気があるように感じる．神にお伺いを立てるあたりがとても面白い．そして，そのインチキな神の声が絶対的な価値を持ち全員がそれに従うところが妙である．太平天国は文字通りの意味とは裏腹に，地獄のような終焉をした，という

ところが皮肉である．この公園の入口に，実に見事な花を見た．それは，数メートルの高さの木であり，紫陽花のような形の花を付けている．その花がたくさん付いている様は，，まるで淡い乳白色のボールが鈴なりになっているようだ．多分，大手鞠という名前である．明るい緑色の葉の繁った中で乳白色のまんまるい花の群れが初夏の日差しを浴びて光っている情景は記憶に残っている．それらの木木は，太平天国も辛亥革命も視続けてきたのだろうか．だから，あのようにまるく清らかに咲いているのだろう．

午後は，貢院街に行った．近くの飯店で昼食を取った．主食の皿よりも点心が多いのに驚いた．餃子，小豆の汁粉，蓮の澱粉で作った餅，焼売，杏仁豆腐，などいくら食べてもあとからあとから出てきた．この店の名前を忘れたが，後で地図を見たら得月楼かと推測される．かの夏目雅子こと陶さんが肉片やら点心やらをほおばっている姿はどこか落ち着かない．蓮の粉でできたねっとりした餅が好物であるらしい．それは，張先生も同じであった．途中，甘い小豆汁粉らしいものがでた．これには流石に閉口して手をつけなかったが，甘党を自称する岸田氏が数人前，ペロリと平らげたのに一同感心した．

貢院街界隈にはあの有名な孔子廟がある．今まで，数回中国を訪れているが，不思議なことにこの孔子廟を見たことがなかった．外見は立派な建造物で日本の寺社と同じ造りである．中には，孔子像や論語等がたくさんあるだろうと予想して行ってみたら，がらんどうで何もなかった．流石に拍子抜けしてしまった．どういうわけか，孔子廟の入ってすぐの左手に，お釈迦様の大きな右手が飾られていた．それが，妙に印象的である．そこを出て，昔の花街を抜けて貢院へ行く．貢院は，科挙の試験場である．一週間くらい弁当を持参してこの試験場に詰めて試験を受けたのだという．受かっても落ちてもその後の人生を送るためには花街が必要らしい．実際の試験を受けた建物が観光客用に置いてあった．それは，人間が一人漸く入れる位の広さしかない個室である．網走で見た独房にそっくりである．当時を忍ぶ極彩色の人形が個室の中に飾られていた．展示場には，科挙の試験問題，合格を知らせる号外用御輿，試験を受かって気がふれている人などがあった．ああ，人生である．

農村風景の中車を走らせ中山陵へ行く．あちらこちらから雲古の匂いがしてくる．道路も誇りっぽい．道行く人々，風景が心に和む．芥川竜之助は，雨のぬかるんだ日に南京にやって来て惨めな気持ちになったと書いてあったが，その気持ちが分かりそうだ．しかし，彼は間もなく舟に乗り，南京市内を旅遊し，途中美人に会いそれからすっかり南京が気に入ったらしい．昔から江蘇辺りは美人の産地である．そして，男は本当に現金なものだ．中山陵は，山中にあり，そこへ到る道は美しい．陽が差していて明るい初夏の感じである．市内から 20 分くらいで着いたと思う．中山陵の広大さと壮麗さには瞠目させられた．歩いて登ってみてその凄さに改めて驚いた．青いカワラと白い石，それに緑の樹木に覆われて，それらの配色がとてもきれいだ．それらは，南国の初夏を演出している．まだ四月だというのに，もう蝉が鳴いていた．その声を聞いて張先生がとても興味深い句を教えてくれた．

蝉噪林愈静
鳥鳴山更幽

詩の韻を踏んでいる．この句をみて松尾芭蕉の感覚は此処ら辺から輸入されたのかと思った．

静かさや岩にしみいる蝉の声
行く春や鳥啼き魚の目は涙

私は直観的にこれらの句を連想した．ここ中山陵は，山間にあり，周囲が森林に囲まれている．私は，日本の奈良山を想起していた．豊かな緑，美しい自然，豊穣な田畑，暖かな気候，美しい人々 _ 江南地域はすべてのものに恵まれているように思える．しかし，この地方の人々は気性が激しく中国を代表する反逆者や革命家のほとんどを江南地域から出しているらしい．その関連が分からない．

近くにある明孝陵へ移動する．こちらの方は美しくきれいな中山陵とは対照的に朽ち果てた山中というところにあった．入口に観光客目当てに駱駝がいたのに驚いた．この駱駝はシルクロードを通ってやってきたのだろうか．その哀しい目が印象的である．明孝陵の背後には栗林が広がっており，そこには鹿が餌付けされていた．栗林全体が陵であり，我々が登ったのが陵の単なる門であると聞いたようにも思える．

この門の上から楓のような木の梢を通して山崎さんが座っている姿が見えた．三国連太郎に似た端正な容貌から何かが漂っている．お洒落で，意気で，スマートで，沈着で，謙虚で，かつどこからか溢れるパッションが感じられる．インテリゲンチャとは，彼のような人を指すのだろう，と思った．少なくとも，小生の身近には存在していない人種であった．この山崎さんと玉井さんの組合せは軽妙にして

洒脱で面白い．しかし，この明孝陵は北京郊外にある明 13 陵と異なり内部は一切不明である．途中に堂があり，その中に明の初代皇帝，朱元璋こと洪武帝の掛け軸が飾られていた．いかにも悪人といった風貌のそれはいろいろなことを想像させてくれた．目が吊り上がり，顎が三日月のように上にしゃくれあがった，一度見たら忘れない顔である．流石，皇帝に着いたあと，嘗ての僚友や仲間達，それに兄弟親戚一同を皆殺しにした面構えである，と得心する．明のこの大虐殺は歴史のなかでも有名である．この絵一枚しか朱元璋に関係したものがなかった．

この堂の周りには，おびただしい数の鳥篭があり，多くの種類の鳥が飼われていた．明孝陵に至る参道には動物の石像が並んでいるという．今回見ていないが，明 13 陵と同じような動物がいるのだろう．そして，どうしてこのようにたくさんの動物が登場するのだろうか，と考えた．中野美代子著の"仙界とポルノグラフィ"の中の"天子の動物園"を思い出した．天子の愛玩が動物だったのだ，と推測した．また，駱駝のいる入口に戻った．ここ明孝陵は，朽ち果てた廃墟の雰囲気があり，周りの自然と融け合っている．さっき見た中山陵とは対照的である．

ここから，有名だという長江大橋を渡りにいった．このドライブがひどかった．橋は，往復路ともに 2 車線ずつあるが渋滞でちっとも進まない．われわれ一行の車は 2 台あったが，1 台は遣り手の運ちゃんで物凄い運転術を披露しながら視界から消えた．小生が乗っている方の運ちゃんは，小心者ですっかり微々っている．必要がないのにやたらとクラクションを鳴らすが，形ばかりでちっともアクセルを踏まない．途中でよっぽど代わってやりたくなった．橋の中央ほどにこの橋を造った偉大な人民の石像が例のスローガンの形で飾られていた．それを見て，ブタペストのチェーン橋やロンドンブリッジを思い出した．長江大橋は難儀をして渡っただけでなんの変哲もなかった．南京市街と反対の地面について少し休んだ．

そこは畑の中で畦道があった．先行部隊は 30 分くらい前に着いていたという．Catherine が野草の花を摘んで手に持っていたのが印象的であった．Notkin がそれらの花を髪に飾っていた記憶がある．その風情が何かホッとさせてくれた．後で分かったことだが，Catherine はそれらの草花をスケッチしその周りにたくさんのメモをとっていたのである．彼女は，都市デザイナーであるらしいが，小生の見たところ自然科学者のように感じられた．大学時代に中国語を専攻し，いまは日本語も話すことができるらしい．しかし，誤解されることを嫌ってあまり話さないという．不思議な人である．帰りは比較的簡単に橋を渡ることができ，夕闇に南京飯店に着いた．

この日は，南京最後の晩餐で日本側がホストする宴である．南京飯店で宴が始まった．あの例の麻婆豆腐を食べたところである．早速席に着く．今日は，中国版夏目雅子こと陶さんと乾杯を競う積もりである．席順は，小生の右隣が玉井さん，その隣が伊野さん，次が陶さんである．先ず，酒選びから始まった．矛 (本当は草冠の下にこの字を書くのだが，JIS 第二水準にないのでこれにした) 台酒（マオタイチュー），汾酒（フェンチュー），四風酒（シーファンチュー），南京産の白酒のなにがしかが候補に上った．私は，即座にマオタイチューがいいと答えた．誰かが，四風酒がよいという．陶さんがやはりためらわずマオタイがよいと答えた．さすがに，酒を見る目がある．一本目は，マオタイに決まった．軟弱な人々のために紹興酒（シャオシンチュー）を一瓶とった．ビールと中国産のワインも頼んだ．用意万端整った．適当な菜を頼む．菜を決める役割は，岸田氏と張先生である．私は，昨夜の嘔吐を思い出しながら食を抑え気味にする．ビールと紹興酒で乾杯した後歓談と菜を食べた．あちらこちらでさまざまな話が起きていた．英語，中国語，日本語みんな交じっている．

それは，突然やってきた．陶さんがいきなり立ち上がり,"Would you stand up please!"とか言った．中国式乾杯儀式の始まりの合図である．彼女は，当然マオタイの杯を持っている．"来たな！"と内心期するものがあった．"乾杯（カンペイ)"と叫び一気に杯を干すのが礼儀である．杯を干したあと，酒が残っていないことを示すために皆の前で杯を逆さにして見せるのだ．その得意顔がなかなかである．一杯目はほとんど全員が呑んだ．二杯目からは有志しか呑まなくなった．岸田氏，山崎氏，伊野氏それに私が日本勢で，中国は陶さんと唐さんである．四杯目を空けたらマオタイが空になったので，御当地の白酒にした．やはり，味が格段に違うと感じた．この酒の名前を忘れてしまった．一気に五杯目を空けた．喉元から胃にいたる経路で酒が燃えるのがわかる．昨夜の病が気になるが，今宵は不思議に酒が美味しく感じられ素直に入ってゆく．少し，めまいがする．

呑み終わって座ると，向こうの席にいる陶さんが流し目で誘いをかけてくる．

"まだ，呑めるの！そこのおじさん！"

"まだ，私と勝負する気？"

といっているようである．"人生，意気に感ず"を座右の銘にしている私がここで引き下がるはずがない．杯を持って勝負を挑む陶さんに私は快く立ち向かう．周りからやんややんやの歓声が挙がる．みんな酔っている．みんな狂い出している．私は，その挑発的な誘いに乗って乾杯を何度も繰り返した．七，八杯目を乾杯したとき彼女が少しひるむのが分かった．その乾杯競争は，結局私が一杯多く呑んで終わった．最後は，白酒がなくなり，紹興酒で乾杯したのだった．多分，彼女は遠来の客である日本人に勝ちを譲ってくれたのだろう．闘いが終わり，私は好敵手陶さんと握手をした．細くて白いその手はひんやりとして，しっとりとしていた．その感触は，内気で寡黙な彼女の性格によく似ているように感じた．彼女のイメージを，"極道の妻たち"を演じた夏目雅子に知らず識らずの内に重ねている，自分に気が付いた．

南京最後の晩餐と宴はこのようにして華やかに終わった．飯店の出口で皆に挨拶をし部屋に戻った．酔いが身体中を駆け巡る．頭の中では，相変わらず中国の夏目雅子が妖艶な容子で私を乾杯に導く．東洋人に生まれ変わった私はもう酔わない．私には，もう吐くものがないのだ．自然そのものなのだ．南京の風土と一体化できたのだ．そんな気分になった．そして，昨夜吐きながら変身したとき考えていた問いの答えが分かった．それは，南京の魔力の来し方行く末である．

この夜私は久し振りに田舎の父母に電話をした．酔いのため何を話したか憶えていないが，"真の東洋人になれてよかったねえ"といわれた気がする．また，"お前は本当に幸せだ"，とも言われた．それは，夢の中の出来事だったかも知れない．意識が次第に薄れていった．朝四時ころ目を覚ました．パジャマを着てベッドに横になっている自分を見付ける．カーテンの外はまだ薄暗い．窓をあけて涼風を誘い込む．春の匂いが風に運ばれてやって来る．次第に空は明るくなり，鳥たちがささやき始めた．今朝で，この南京とはおさらばである．その心境は，次の芭蕉の句である．

行く春や鳥啼き魚の目は涙

4月24日（天気　晴）

朝６時ころ南京飯店のロビーに集合である．行ってみたら，昨夜の深酒をした陶さんが既に来て座っている．けろりとした風情である．朝の挨拶をした．普通と変わりない様子である．確か，山崎さんが彼女に問いかけて，ぜひ日本に一度おいで下さい，招待します，と話していた．

ホテルを出て駅に向かった．ホームで今回のホストをしてくれた江蘇省の計算センターの唐さんと陶さんに，オールドパーとセブンスターをプレゼントした．陶さんが，"Thank you"といって受け取った．その目には，昨夜の晩餐で乾杯を挑発した色がなかった．今日は，白いブルゾンに黒いスカートである．初夏の爽さを漂わせていた．私は素敵な彼女の笑顔を見ながら，どうして陶さんの歯が白くないのか，と一生懸命に考えていた．やがて，南京に来た時と同じ電車の軟座に席をとった．ホームで見送りをしてくれている唐さんと陶さんにてを振り別れを告げた．上海に向かう列車の中でも言わずとしれた酒盛りが始まった．見渡す限りの黄色の海原を，江南の春の中を，エキセントリックな一団を乗せた列車が走り続けた．再会！南京！

私は中国の裏にある抒情に出会った旅のように感じていた．そして，もう一度あの中山路の鈴掛の街路樹の中を新芽がハラハラと降り注ぐ中を歩きたい，と思っていた．体重計のおじいさん！乾杯美人の陶さん！再会！

Call for Papers

ACM SIGSOFT '92: Fifth Symposium on Software Development Environments*

Washington D.C., USA, Dec 9–11, 1992

General Chair: Ian Thomas (USA), thomas@sda.com
Program Chair: Herbert Weber (FRG), hweber@udo.informatik.uni–dortmund.de

Symposium Theme: Evaluation and Integration

Experience gained with the use of current generation software development environments (SDEs) has led to the identification of a number of shortcomings. Independently developed tools incorporated into an environment do not provide the expected levels of support for users' activities and development processes. A number of factors influence the degree to which it will be possible to put tools together to form an integrated environment supporting user development processes. One of them is a better understanding of both the tools and the frameworks of environments. Others are better evaluation and integration techniques.

Evaluation techniques will be needed to determine the suitability of a tool's function and performance, and to determine the effectiveness of its support for the users' activities and processes. Integration techniques are needed to support the inter–connection of tools using communication mechanisms, to allow for data inter–operability, and for the coordination of the work of many software developers in accordance with a software process. New, flexible architectures and frameworks are needed to promote effective use of tools and environments for an organization. Authors are encouraged to emphasize issues relating to evaluation and integration when discussing one or more of the following subjects:

- Conceptual Models of Environments
- Environment Frameworks and Architectures
- Environment Integration Strategies
- Database support for Environments
- User Interfaces for Environments
- Distributed Environments
- Environment Generators
- Process Support Facilities
- Language Support Facilities
- Security Concerns
- Environment Deployment Issues
- Environments and Issues of Scale

Of particular interest are:

- Results of Empirical Studies
- Results of Comparative Analyses

Authors should explain what is new and significant about the work presented and must adequately compare the work with similar work or similar systems. Authors must list the key ways in which their contribution is unique or important. Papers will be reviewed by the Program Committee.

Information and Instructions for Authors

Please send **six** copies of your full paper, including an abstract, to the Program Chair. Papers are limited to 6000 words.

Prof. Herbert Weber, Chair 5SDE
Fachbereich Informatik
University of Dortmund
Baroperstrasse 301
4600 DORTMUND 50, Germany

Submission Deadline: April 21, 1992

Acceptance Notification: July 1, 1992

Camera–ready paper due: September 1, 1992

Authors of accepted papers will be requested to sign an ACM copyright release form. The best papers will be candidates for planned special issues of appropriate refereed journals. Proceedings will be distributed at the Symposium, or may be purchased from ACM.

Demonstrations

Proposals for demonstrations of prototype or production quality environments and tools having novel capabilities related to the scope of the symposium will be accepted by the program committee. **Six** copies of a short abstract (3 to 5 pages) describing the environment or tool, what is new and significant about the environment or tool, experience with the environment or tool, its current status and availability, and its hardware/software requirements should be submitted by April 21, 1992 to the program chair.

* Sponsored by ACM SIGSOFT

論 文 募 集

ソフトウェア・シンポジウム '92

1992 年 6 月 9 日 (火) - 11 日 (木)
於： 長野県農協ビル・アクティーホール (長野市)

主 催
ソフトウェア技術者協会 (SEA)
協 賛 (予定)
日本ソフトウェア科学会 (JSST)　情報処理学会 (IPSJ)
情報サービス産業協会 (JISA)　長野県情報サービス振興協会 (NISA)

基調テーマ： ネットワーク時代における分散開発環境

メーカ・ソフトハウス・計算センタ・エンドユーザ・大学・研究所などさまざまな仕事場で活躍している技術者および研究者が一堂に会し，ソフトウェア技術の実践的側面に関する経験や知識を交流するユニークな場として運営されてきたこのソフトウェア・シンポジウムも，早いものでもう 12 回目を数えるにいたりました．ワークステーションやネットワークの活用も本格化し，開発技術／環境も，またアプリケーションの形態も，大きく様変わりしつつあります．

そこで，今回の論文募集にあたっては，そうしたダイナミックな技術変化の方向性を探るべく，以下に示すような問題点および技術分野をセッション・テーマの候補として選び，討論の焦点を絞ろうと考えています．バラエティに富む具体的な経験にもとづく実践レポートや，革新的なアイデアの提案など，みなさんからの積極的な応募をお待ちします．

募集論文テーマ

プログラム委員会としては，次のような問題点および技術分野をセッション・テーマの候補として考えています：

問題点：	分散システムの構築と運用	ヘテロ・ネットワーク	SW の資産化と再利用	CASE の導入と活用
	新技術の評価・導入・改良	AP ノウハウの蓄積と活用	ユーザとの折衝	チームリーダの育成
技術分野：	開発環境の構築と管理	オブジェクト指向	CSCW	ネットワーク管理
	環境の基盤技術	AI と SW 工学	技術者教育	システムの進化
	SW 開発プロセス	SW データベース	品質の定量化	SW 技術の社会的側面

応募要領

応募論文 (Full Paper または Extended Abstract, ワープロまたはコンピュータ出力で A4 判数ページ) を **5 部コピー**して，**1992 年 1 月 31 日**までに，2 人のプログラム委員長のいずれか宛に郵便でお送りください．あるいは，junet にアクセス可能な方は，E-Mail で応募されても結構です (この場合はもちろん 5 部コピーの必要はありません．フォーマットは roff. TeX, Plain Text 何でもかまいません)．

応募論文は，プログラム委員会で内容の審査を行い，結果を 1992 年 2 月下旬までに，応募者全員にお知らせします．審査を通過した方々には，最終論文 (予稿集にのせるためのカメラ・レディ原稿) を1992 年 4 月中旬までに仕上げていただくことになります．その他，パネル討論のテーマや招待講演，チュートリアル，ツール展示などに関してのアイデアも，どしどし提案してください．

シンポジウム・スタッフ

＜実行委員長＞	市川寛 (電算)	西条経治 (長野県中小企業連合会)	
＜プログラム委員長＞	岸田孝一 (SRA)	玉井哲雄 (筑波大)	堀江進 (NES)
	中野秀男 (大阪大)	手塚正義 (金沢工大)	松原友夫 (Office Peopleware)
＜チュートリアル委員長＞	野村敏次 (JIP)	中谷多哉子 (FXIS)	松村好高 (SRA)
＜ツール展示委員長＞	丸山徳 (電算)	布川博士 (東北大)	松本健一 (大阪大)
＜プログラム委員＞	荒木啓二郎 (九州大)	野呂昌満 (南山大)	山口高平 (静岡大)
	飯沢恒 (三菱東部 CS)	平岡信之 (長野大)	山崎利治 (日本ユニシス)
	筏井美枝子 (SRA)	平野広美 (ENICOM)	山田茂 (広島大)
	海尻賢二 (信州大)	平山伸一 (KCS)	好川哲人 (ASTEM)
	亀田繁 (FHL)	深瀬弘恭 (アスキー)	我妻克彦 (東北 CS)
	熊谷章 (PFU & 富士通)	二木厚吉 (電総研)	

***** 応募の宛先および応募用のカバー・シートは裏面にあります *****

ソフトウェア・シンポジウム '92

論文応募カバー・シート

氏　名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿（ふりがな）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

会社（大学）：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

部門（学部）：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿役　職：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

住　所：（〒　　）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

TEL: [　　－　　－　　内線　　] FAX: [　　－　　－　　]

E-Mail: ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

論文タイトル（予定):＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

共同執筆者（もしいれば):＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

Abstract：（論文の要旨を10行以内でまとめてください）

＿＿

＿＿

＿＿

＿＿

＿＿

＿＿

＿＿

＿＿

＿＿

問題点と分野：　あなたの論文は以下の問題点や技術分野のどれに関連していますか？　あてはまるものにチェックしてください。

問題点：

□分散システムの構築	□ヘテロ・ネットワーク	□SWの資産化と再利用	□CASEの導入と活用
□新技術の評価・導入・改良	□APノウハウの蓄積と活用	□ユーザとの折衝	□チームリーダの育成
□その他（　　）			

技術分野：

□開発環境の構築と管理	□オブジェクト指向	□CSCW	□ネットワーク管理
□環境の基盤技術	□AIとSW工学	□技術者教育	□システムの進化
□SW開発プロセス	□SWデータベース	□品質の定量化	□SW技術の社会的側面
□その他（　　）			

カテゴリ：　あなたの論文は，次の2つのカテゴリのうち，いずれに属しますか？（いずれかにチェック）

□ Idea Paper:　新しい概念／技法／ツールなどの提案.

□ Experience Paper:　実際の開発／試行の経験にもとづく報告.

---------- 送　付　先 ----------

〒565 吹田市山田丘２－１
大阪大学　工学部　通信工学科
中野 秀男
TEL 06-877-5111 ext.4773　FAX 06-875-0506
E-MAIL: nakano@oucom.osaka-u.ac.jp

または　〒160 新宿区四谷３－１２　丸正ビル５F
(株)SRA ソフトウェア工学研究所
岸田　孝一
TEL 03-3357-1507　FAX 03-3351-0880
E-MAIL: k2@sra.co.jp

********** 応募締め切り：　**1992年1月31日（金）** **********

CALL FOR PAPERS, TUTORIALS, AND PANEL SESSION PROPOSALS

CONFERENCE ON SOFTWARE MAINTENANCE – 1992

ORLANDO, FLORIDA

November 9 – 12, 1992

The Conference on Software Maintenance – 1992 (CSM-92) will gather software managers, developers, maintainers, and researchers to discuss new solutions to the continuing challenges of software maintenance and software maintainability.

Theme: Improving the Processes of Software Maintenance

CSM-92 will focus on the processes employed in performing software maintenance – as a means of understanding, defining, improving, managing, and supporting the practices of software maintenance. CSM-92 seeks papers and other contributions that will advance the state-of-the-art or state-of-the-practice of software maintenance, and help foster increased interaction between the research and practitioner communities.

SUGGESTED TOPICS

Papers, Experience Reports, Panel Session Proposals, and Tutorial Proposals related to, but not limited to, the following topics are invited:

• Software Maintenance Process Modeling or Assessment • Successful Maintenance Processes • Software Maintenance Methodologies • Software Evolution, Migration, and Conversion • Reengineering, Reverse Engineering, and Restructuring • Standards for Software Maintenance • Empirical Maintenance Studies • Software Maintenance and Maintainability Metrics • Software Maintenance Management • Software Maintenance Education • Program Understanding • Achieving Feedback from Maintenance to Development • Software Maintenance Tools • Impact Analysis • Developing Maintainable Systems • Documenting for Software Maintenance • Recertification of Modified Software

SCOPE AND PURPOSE

Software Maintenance includes the enhancement, restructuring, and correction of software. CSM-92 will provide a forum for the discussion of software maintenance through refereed papers, experience reports, panel sessions, tutorials, and informal meetings. CSM-92 will present the most important current practical, experimental, and theoretical work being conducted in support of software maintenance. CSM-92 will entertain papers and other contributions in the areas of Basic Research, Experiences with Technology Application, and Empirical Research. The conference is open to all who are involved with or have an interest in software maintenance: practitioners and researchers, from industry, government, and academia.

SUBMISSION INFORMATION

Five (5) copies of papers, experience reports, or panel proposals should be submitted in English to the program chair at the address listed. The author of each accepted contribution will be expected to present it at CSM-92.

Papers should be 1000-5000 words in length. Papers must not have been previously published or be under current consideration for publication elsewhere. The first page must include title, all authors' names, lead author's address and telephone number, and abstract (250 word maximum).

Experience Reports are intended to summarize experiences with the practical application of software maintenance technology or processes (e.g. using a tool or method, applying a metric, following a project management discipline, etc.). The contributor(s) should submit a 4-6 page written description of the experience, highlighting the outcomes and lessons learned. The submission should be identified by the keyword "Experience" on the top of the first page, and otherwise follow the first page instructions given for papers. The evaluation criteria for these submissions will be based on the applicability and relevance of the results to the software maintenance community.

Panel Proposals should include title, proposed chair, tentative panelists (including short vitae), a brief description of the subject of the panel session, and supporting rationale. Panelists must have agreed to participate prior to submission of the proposal.

Tutorial Sessions have been a significant part of past CSM conferences. Proposals are solicited for both full-length, all-day tutorials and also for mini-tutorials of two to four hours in length. Direct five (5) copies of proposals to the tutorials chair.

IMPORTANT DATES: Submission deadline: April 1, 1992;
Acceptance notification: July 15, 1992; Final version due: September 1, 1992

CONFERENCE COMMITTEE

GENERAL CHAIR
Vaclav Rajlich
Department of Computer Science
Wayne State University
Detroit, MI 48202
TEL: (313) 577-5423; FAX: (313) 577-6868
email: vtr@cs.wayne.edu

PROGRAM CHAIR
Marc Kellner
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
TEL: (412) 268-7721; FAX: (412) 268-5758
email: mik@sei.cmu.edu

PUBLICITY CHAIR
Dave Card
Computer Sciences Corporation
Systems Sciences Division
4061 Powder Mill Road
Calverton, MD 20705
TEL: (301) 572-3815; FAX: (301) 572-7502
email: dc@sei.cmu.edu

TUTORIALS CHAIR
Frank Calliss
Department of Computer Science and Engineering
Arizona State University
Tempe, AZ 85287-5406
TEL: (602) 965-3190; FAX: (602) 965-2751
email: calliss@asuvax.eas.asu.edu

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