SEAMAIL

Newsletter from Software Engineers Association

Volume 9, Number 6-7 November, 1994

目　次

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

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

その主な活動は，機関誌SEAMAILの発行，支部および研究分科会の運営，セミナー／ワークショップ／シンポジウムなどのイベントの開催，および内外の関係諸団体との交流です．発足当初約200人にすぎなかった会員数もその後飛躍的に増加し，現在，北は北海道から南は沖縄まで，800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40社を数えます．支部は，東京以外に，関西，横浜，長野，名古屋，九州，東北の各地区で設立されており，その他の地域でも設立準備をしています．分科会は，東京，関西，名古屋で，それぞれいくつかが活動しており，その他の支部でも，月例会やフォーラムが定期的に開催されています．

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

ＳＥＡＭＡＩＬ　Ｖｏｌ．９，Ｎｏ．６－７　　１９９４年１１月２６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　１，０００円（禁無断転載）

『ソフトウェア・シンポジウム特集号』の発行にあたって

渡邉 雄一 *
(SS94 プログラム委員長)

SEA 恒例のイベントであるソフトウェア・シンポジウムも, 今年で 14 回目となった。函館という北の地での開催ということで, 参加者の減少が少からず心配されたが 御蔭様で 100 余名の参加者を得て無事 成功裡に終わった。関係者を代表して厚く御礼申し上げたい。

今回, 何故か私がプログラム委員長の 1 人にかつぎ出された訳だが, その理由は 単に「事務局の手が少ないので, 雑務経験が多少あって力仕事も出来そうな 30 代の SEA の幹事の 1 人」ということであったようだ。そのような訳で, プログラムの構成や技術的な内容に関しては玉井先生 (東京大学) にすべて面倒をみて頂いた。では, その他の仕事が出来たかというとこれも疑問で, 事務的なことでは相変わらず SEA 事務局の中島嬢の手を煩わせ, 進捗の管理や各種準備などについて経験豊富な他の PC の指導を仰いで, という体たらくであった。その替わりというにはおこがましいが, このような多少なりともまとまった形の記録を残すことも意義があろうかと思って取り組んでみた次第である。だが, このような記録が SEAMAIL に掲載されるは, これが初めてでは無い。最近では, 市川さん (電算) が長野で開催された SS92 のパネルディスカッションの膨大な記録を SEAMAIL Vol. 7, No. 3 に, しかも独力で残されている。只々 感服するばかりである。今回はそのような過去の記録も参考にしながら, OOPSLA といった他の活発なシンポジウムの補遺集も横目で睨みながら取り組んでみた次第である。とは言え, 遅々として捗らないテープ起こしが なんとか終わって まとめの作業に入ると, 結局は 御発表頂いたパネリスト諸氏及びフロアから発言して下さった方々の手を煩わせて文章の校正をして頂くという, 万事が全てこの始末である。ご協力頂いた方々のお名前は誌面の関係で割愛させて頂くが, 感謝申し上げる。

さて, 余計な説明はこれくらいにして, 以下に本編の構成を簡単に紹介して序に代えさせて頂く。
まず 今回 最優秀論文賞を授賞された 田中 敏文さん (オムロン) に感想を寄せて頂いたので, 最初にそれを配置した。続いて, 簡単だが今回シンポジウムの論文の傾向や発表の様子などを振り返ってみる。
そして, 今回の特集の中心には シンポジウムでの 2 つのパネルセッション

イブニング・パネル Software Lemmingineering !?
クロージング・パネル これからのソフトウェア・プロセス・マネージメント

の記録を据えた。とくに, イブニングパネルの方はその中でも紹介されたアンケートの集計を含む総括と, SS のあと川崎で開催された討論会の報告を 塩谷さん (SRA) が寄稿して下さった。今回の特集の目的の 1 つは, 地理的な制約から参加出来なかった会員諸兄への情報の補完と考えているが, その面でも非常に有益な資料と言えよう。もっとも パネルディスカッションに関して言えば, 本来は, きちんと要点を整理し箇条書にまとめて収録するのがスジであろうが私の力量ではとてもそこまで出来ず, このような記録という冗長な形になった。その分, 少しでも会場の雰囲気がお伝え出来れば, というのは言い逃れだが, 何卒 御容赦頂きたい。

なお, 当初は『補遺集』として企画されたこの号が『シンポジウム特集』となったのは, 例年以上にハイペースで準備が始まっている来年の SS95 の関係者から, 熱いメッセージが多数寄せられているからにほかならない。是非とも奮って積極的な論文投稿/参加を会員諸兄にお願いする次第である。

* (株) アスキー OA システム部

ソフトウェアシンポジウム '94 最優秀論文賞を授賞して

田中 敏文 *

"内容があまりにも泥臭かったので, 最初 (発表前) はどうかと思っていたのですが, 発表で結構ウケていたので, ひょっとしたらと思ってました。"

函館で, 最優秀論文の発表と表彰があり, その直後に PC 委員長のアスキーの渡邊さんからマイクを差し向けられ, 最初に私の口から出た言葉です。後で同僚から "ひょっとしたらもらえるかも, なんて言うかー?" と言われてしまいました。

でも, 発表中の開発現場の混乱状況の説明に対する聴衆の笑い交じりの反応もさることながら, 発表後に, 多くの方々が歩み寄って来られて, 質問やら "面白かった" と言う感想やらを頂いて, なんとなく手応えを感じていました。中でも, 今シンポジウムのパネルで仙人と紹介された某氏から, 面白いというか手厳しいコメントを頂きました。

"オムロンさんほどの会社でも, まだあんなところで苦しんでいるの? うちの OS グループでは 1970 年代に解決している問題だよ。ハー。でもまー, うちの会社だってまだまだの所もあるけどね。ハー。"

自分自身で自分の成果を分析するのは恥ずかしくてくすぐったいものがあるのですが, 今あらためて, 何故 最優秀論文賞という栄誉が頂けたのかをじっくり考えてみますと,

(a) 現場の混乱したプロセスを図や数字で示し, 多くの現場で苦労した経験のある人の共感を得た。

(b) 理論的には目新しくもない技術を, 現場に導入・定着させるためにプロセスの図式化や数値化 (メトリクス化) をし, 現場の人間を説得し巻き込んでいる。

(c) その効果を, 数値でもって証明している。

などが挙げられると思います。さらに, これらの要因をもっと突き詰めると, 次のようなことが見えてきます。

(1) 何故 共感を得たのか?
→ まさに皆こういうレベルの低いところで苦しんでいるから

(2) 何故 現場の人間を説得し巻き込めたのか?
→ 改善スタッフとして, 決して腰の退けた活動ではなかったし, また逆に技術の単なる押しつけでもなかった。現場に入り込んでいっしょに問題点を見つけるという姿勢で, 今回の改善に取り組んだから。

(3) 何故 効果を数値化できたのか?
→ 改善プロセスの中に効果が測定できるような仕組みを組み込んだから。

これらに関する技術的な内容については発表の中で触れていましたが, スタッフとしての姿勢の問題として (1)〜(3) に対してもう少し突っ込むと, 次のようなことが言えると思います。すなわち,

"発表ネタとして脚光を浴びる多くのテーマは, 比較的レベルの高い組織 (プロセス成熟度で言うとレベル 3 以上) に対して効果のある高度かつ先進的な技術を取り上げたものなのですが, 実は多くの組織 (たぶん 9 割以上の組織) では当たり前と思われる技術の移転がなかなかできていなくて, 発表したような泥臭い問題で苦しんでいるわけです。今回発表した内容は, この部分に対し目をそむけることなく, 結果に対する責任を負う立場でプロジェクトに参画し, 成果を挙げたスタッフとしての改善活動事例なのです。"

ここまで, 自分で書いてしまうと, 読んでる皆さんはさぞかしうんざりなさることでしょうが, 実は先述の仙人さんのコメントを, 読み返して下さい。。。。。。どうです, 同じことをおっしゃっていたということがお分かりになるでしょう。ハー[1]。

これで私の独断と偏見に満ちた授賞の言葉は終りたいと思いますが, 最後に, 発表の準備のために, 唯一 函館山にガスのかかっていなかったシンポジウム初日の夜に, 夜景を見ることが出来なかった怨み[2]を申し上げておきます。

* オムロン (株) EFTS 統轄事業部 開発センタ 第 2 開発室

[1] 仙人さん, 勝手に解釈してごめんなさい。
[2] 誰に対するものでもないのですが

ソフトウェア・シンポジウム'94を振り返って

渡邊 雄一 *
(SS94 プログラム委員長)

0. はじめに

ソフトウェアシンポジウム'94を簡単に振り返ってみる。立場上, 一個人としての感想を述べると言う無責任なことでは済まされないようだが, かといって大層なことを書けるわけでもない。また 誌面の関係もあるので, 何から何まで網羅して こと細かに, というわけにもいかない。まあ, ともかく滅多に出来ないプログラム委員長という立場から, その目に映ったことを, 思いつくままに記そう。

1. 初日 (6月15日)

ちょっと汗ばむような日射しの函館空港に降り立って 五稜郭の辺りを散策してから市電に乗って会場近くの停留所まで行き, そこから徒歩で5分弱の港に面したレンガ作りの倉庫街にある会場 (金森ホール) に辿り着く。しかしさすが (?) 北海道, 風がそよぐと暑くもなく寒くもなく調度良い。

1.1 オープニング, 招待講演

型通りのオープニングで順調な滑り出し。続いて玉井先生の司会で, 棟上先生の講演。これも本当は詳細にレポートすべきなのだろうが, 講演の内容は 論文集に書いて下さった内容[1]に沿ったものであったので, テープ起こしのボランティアの手が足りないこともあって割愛させて頂くことにした。[2] 特に, 棟上先生の話は, 一部の限られた所では同様の講演などもされているようだが, このように開かれた場で, 喋って頂くことは少く, その意味で良かったと思っている。

1.2 セッション1 : CASE

初日に組まれたペーパセッションはこの1つだけである。いずれも興味があったが ちょっとした進行のもたつきもあって, 時間不足になって 議論が十分にされなかったことと, 最後に発表に立った伊藤さん (MASC) にしわ寄せが行ってしまったのが残念。でも, その分, 夜の情報交換パーティ[3]で盛り上がっていたようだ。まあ, シンポジウムの最初のセッションというのは誰でも緊張するのだろうが。
このセッションでは, 小室さん (日立ソフト) の, 論文に書かれていない適用例の具体的な部分の話が興味深かった。

2. 第2日 (6月16日)

昨日と同じく良い天気。北海道というところは日本の一番北に位置しているとともに東に振れていることもあって, 朝が早い。そのせいか, 町全体が動き出すのがとても早いように感じた。今回のSSは, 実質この1日に詰め込まれていて, 集中出来る反面, 緊張を維持出来るか ちょっと不安である。資料を持ってセッションの開始 (9:30) に遅れないように歩いて15分ほどの会場に向かう。

2.1 セッション2 : プロセス

このセッションには, いずれも水準以上の, そしてまたそれぞれに個性的なものが集まった。青山さん (富士通) のものは, 経験報告の書き方の1つのお手本とでも言えるだろう。自分でも何人かの人に論文の投稿をお願いしたが, 「色々とまずいことがあって書きにくい。特に状況をある程度説明しないと読者に伝わらないであろうし, それではそれを書いてしまうと 機密の保持などとの絡みもあって」というにことを少からず言われた。氏の場合には, 自社で中心になってやっておられることで, このような制約は 下請けの者と比べれば, はるかに少ないであろうが…。しかし, それを差し引いても話のディフォルメの仕方というようなことも含めて的確に読者に読ませる素晴らしいものだと思う。

一方, 田中さん (オムロン) ほか のものも, 青山さんのとは違った意味で経験報告の威力を示した一級のものと言える。特に, ある箇所に意図して摘要した改善が予想しなかった所に波及し, それが全体としてより良い効果を得たというデータを適宜示しているのは, 会場から少なからず共感の意志表明があった。また当然ながら, 今回の論文の中で扱われていること

* (株) アスキー OA システム部

[1] 今回は, この講演も2つのパネルセッションも 全てあらかじめ予定されていた方々には, お忙しいにも関らず論文集のために書いて下さった収録することが出来た。

[2] 講演の後の質疑応答が面白かったとの声も聞かれたが, マイクできちんと拾えていないので今から記録として整理するのは大変な労力が必要となる.

[3] 隣の棟にあるビアホールで行なわれた.

の土台になっているもの前には, 最低でも1年の表には出ないデータの蓄積があろうと思う。このような問題にアプローチしている現場の実務者には, 大変貴重なデータが盛られている。また, 発表もとても素晴らしかった。

2.2 セッション3: 品質・見積り

ここにも高水準のものが集まった。しかしどうも私は個人的にはこの分野が苦手だ。といって, 他に得意な分野がある訳でもないが …。その理由の1つは, ある程度の規模のプロジェクトに参加した経験が無いので, そこでの品質管理の問題が皮膚感覚として今一つ理解しきれていないことにあると思っている。

三宅さん (NTT) ほか の論文は, かなり大きな規模と高品質を要求されている立場での, ソフトウェア作りにおける提案である。論文も発表も立派であった。特にご本人は, 1つの提案なので討論をされたいとのことで, 発表も簡潔にまとめられたが, 裁ききれないほどの質問や意見が飛び交うという所までは残念ながら至らなかった。思うに, そのようなことを議論するほどの品質を求めずに漫然と仕事をしているソフトウェアの現場が少なくないのでなかろうか?。

一方, 高橋さん (電力中研) ほか の論文は, 彼が長いこと取り組んでいるテーマの1つの中間発表的な色彩が強いものと言える。それは, Cobol 系の言語の FPA による実践的な適用に答を携えてきた, とでも表現すれば良かろうか。論文としては経験報告の色彩が強いとの声もあったが, 表現の的確さなど書き方は一級品。

こんな2つの論文に負けじと出てきたのが, 岡田さん (富士ゼロックス) のもの。この論文は, 失敗をきちんとまとめて下さったことである。質疑応答では, いくつか的外れな意見も飛び出したが, 現場の苦労を立派にまとめている点で, 敬服する。但し, 分析のアプローチがこれで本当に適当だったのかとか, さらにこれだけのデータでもっと別の分析がや予測が出来たのでは, との声も一部にあった。いずれも, 今後への成果の期待へのエールと受けとって欲しい。

2.3 セッション4: 仕様

経験報告として出てきた2つは, ともにマイコン制御分野からのものであった。そのためか, 会場から出た質問/コメントは発表そのものの周辺を捕らえたものが少なからずあったのが印象深い。査読段階では出されなかった無かった視点での意見交換が一番なされていたように思うからである。特に, 組み込みソフトというのは, モジュールサイズや応答速度といった, やったものでないとわからない難しさが潜んでいるからだろう。また, 今までどちらかと言うと表舞台に登場しなかった分野からの論文投稿や参加があったことは喜ばしい。

しかし, その分野に造詣のある者でも, 個々に使用されている用語のニュアンスの解釈には 少からず違いがあったようだ。仕様の話は, 真面目にやると本当に時間がかかる。発表者らが, 厳密にある種の代表的な手法を使ってのことならばいざ知らず, 大抵はそれらを現場に併せて修正摘要したり, また新たな提案だったりするわけで, それを短い説明の中でどの程度伝えられるのか …。

一方, 技術論文としては形式仕様記述に関する2点だった。今回の SS での発表の中で元気のあったのはいずれも, 数少い女性の方の発表だった。しかしそれでも 形式的な手法の発表は (今回の SS に限らず) 会場が重苦しくなる。今回もその傾向はみられた。厳密な議論や証明は 予稿集にきちんと盛り込んで, 発表は 観点の筋道を説明することを中心にした方が良いだろうと思うのは素人の浅はかさな了見だろうか?

2.4 セッション5: 人間的要因

最後のセッションのタイトルはちょっとこじつけとも言えなくないが, 今までのセッションの枠組みでは捕らえきれない領域のもの。この中では, 今回採録された唯一の英語の論文である Reeves さん (SRA) ほか のものが高い評価を受けていた。ただ残念なのは, 発表された共著者の中小路さん (コロラド大) が久々の日本語で緊張されたのか, やや導入部の説明が冗長だったので, 肝心の後半の説明が駆け足気味になったことだろう。

2.5 イブニング・パネル

塩谷さんの総括を参照して頂こう。このイブニング・セッションは, 今回の SS での初めての試みであった。パネルの出来としては, まぁまぁではなかったかと思っている。

が, セッションが終わってから 繰り出した先で店に入って一息つくと, もうラストオーダだと言われたのは参った。函館の町の時間の流れをもっと調べてプログラムを編成すべきだったと後悔することしきりである。

3. 最終日 (6月17日)

特に天気が悪いとは思えないが, よくよく回りを眺めると山で会うのとは違った不思議な感覚を覚える霧に覆われている。そのため, 空港に飛行機が降りら

れず千歳空港に行っているという。ということは帰れない?。なんて心配をよそに最終日の唯一のプログラムがスタートする。

3.1 クロージングパネル

このパネルは, 当初は ISO9000-3 などの規格の周辺に関与/精通している数名を招いて'一発触発'の議論になるかとも思われた。しかし, 残念ながら, 丁度同じ時期にカナダで開催された国際会議と重なったために調整がつかなかった。そのため当初の目論見とは視点を変えたパネルになった。テーマの設定及び人選に関しては, 司会をお願いした大場先生 (広島市立大) に すっかりお世話になった。

会場からの反応は, 兼子先生と桑名さんという普段のSEA のイベントではお目にかかれない 2 人の発言が予測出来ないことと, パネラそれらの話に説得力があって, それをフォローするのに精一杯といった感が無いではなかった。しかし, ちょっと長いかなぁ〜と考えて設定した時間が短く感ぜられる, 充実したパネルだったと思う。

4. 全体を振り返って

今回のシンポジウムの特徴の 1 つに "シングルトラック" のプログラム編成が上げられる。結論から言うと, とっても良かったと思っている。1 番のメリットは, 聴きたいセッションがバッティングすることなく, 自分の聴きたいものをすべて出席できるからである。まあ, 普段だったら聴かないようなセッションに参加するというのも, 年に 1 回くらいはあっても良いなぁという感じである。もっともそれも, 厳しい査読を経て採録論文ばかりだからなのかもしれない。今回, 初めて "全論文のみ" で審査という英断をされた玉井先生に対して, また, 忙しい中で投稿して下さった各位に感謝する。

さて, 全体を通して聴いてみて遅蒔きながら解ったことは, 「みんな同じようなことで悩んでいて, それぞれ特徴あるアプローチで問題解決を試みているのだなぁ」ということである。そういうことも含めて, 技術者が交流する場として 特定の狭いテーマで開催されるのとは別の, しかも査読のされたペーパを中心に進行してゆく会議は貴重なんだと改めて思った。そのような訳で, 逆に気になったのが, 経験論文における題材の表現の仕方である。題材としておもしろいのに, 分析のドフォルメの仕方で (極端に言えば) そのでの成果が結果として物足りないといったことも出てくる。他人に訴えるということは難しいものだと改めて感じた。

しかし, 最優秀論文に選ばれたものを改めて読み返してみると, SS のための経験報告のネタは みんなの足元にころがっていることが良く分かる。そして, ネタはきっかけにしか過ぎず, そのネタを育て成果に結び付けるよう努力せねばならないことも。

そんなことを考えながら 読んだ論文について言及しておくと, (まあ投稿されたものを読んだというのが初めての経験だったこともあるのであろうが) 書き急いだものが少くなかった。それが文章としてこなれていないという程度ならまだしも, 文章として推敲してから他人に見せるものにするのは, 最低の礼儀かと思う。査読を依頼された方は, 行間を読みとるべく対するわけであるのだから。逆に, 例え部分的に至らない点があったとしても, 査読者からのコメントをもとに より良い論文に改訂されてくればなんら問題ない。そのような論文がカメラレディ原稿を届けて頂いたときは, 委員として苦労した甲斐があったかなぁと喜ばしかった。

5. おわりに

ある雑誌の広告に, 秋山 仁さん (数学者) のエッセイが載っていた。

> 日本人は高度経済成長の頃から子供達の教育に関しても, 合理性と効率性を追求した結果, 平均水準の高い能力を育てることには成功しました。ユネスコの調査でも日本の中学生の計算能力, 正答率等は世界中のトップです。
>
> けれども, 日本が「これは, こう解く」と決めて, 天下り的に知識を丸暗記するマニュアル人間から脱却して自分の目で見, 心で不思議を感じる感性を養わなくては, これからの世界のリーダーにはなれないと思います。

本号の後半に見られるように 来年の SS95 は 積極的かつ順調に準備が進んでいる。目先の技術も仕事のためには必要だが, それ以上に 技術者の交流を通じて "感性を磨く場" として, SS が来年以降も SEA のメインイベントとしてさらなる成長を遂げてくれるだろうことを願って, 拙い回想録の筆を置く。

Software Lemmingineering !? *

パネリスト： 長崎 祥 † 熊谷 章 ‡ 伊藤 昌夫 § ゲスト： Gerhard Fischer ¶
司会： 塩谷 和範 ‖

司会 (塩谷) まず テーマの Software Lemmingineering ですが, IEEE Software の昨年 9 月号に出ていた, 4 ページのコラムで, 筆者は Alan Davis です [1]。元のペーパを読んでらっしゃらない方も多いと思いますが, 大ざっぱに, 表[1]があれば, 一応, 概要だけは分かっていただけると思います。この表の Davis さんの評価が絶対だと主張するつもりはありません。単なる 1 つのネタにしたいと思います。

で, 今さらながら, 私の解釈を簡単に説明します。Davis さん自身は, この表以外に [1] の中で説明しているのですが, 例えば (表では) 上から, 構造化, OO ... と分類されていますが, この分類そのものも 1 つの視点から切った分類でないのは, お分かりだと思います。そういった意味で, 何が危険度 (risk) なのかとか, 何が引合い度 (pay-off) なのかとかは, Davis さんのペーパでも厳密に定義していません。

また, 昨日お願いしたアンケートには「あなた/私は, どれくらいレミングに近いのかなぁ?」という意味あいで「レミング度」という分類を付けてみました。約 40 人の方からご回答頂きました。細かい集計は終っていませんが, 相対的に 皆さんのレミング度は低かったように思います。

当然ですけれども, アンケート項目にある「構造化」というのは昔の話で「構造化なんて当り前の時代からやっているので私は知らない」と書いておられる方もいらっしゃいます。つまり, 当り前になっているので, それを自分のレミング度が高いと書いた方もいますし, 逆に 当り前になっているので書けないと記入した方もありました。

あまり私だけが喋ってもしょうがないので, 会場に, Gerhard Fischer さんが, コロラドからいらっしゃっています。AI 畑といっていいのですよね? まず最初に, 彼の視点から Software Lemmingineering について, 意見を聞いてみたいと思います。しかし, パネルは日本語でなされるわけです。それを逐次通訳するのはパネリストの顔ぶれをみても不可能じゃないかと思いますので, フィッシャーさんは, 最初にコメントを頂いた時点で取り敢えず解放してしてあげることにしましょう。次に, 壇上に並んでいるパネラの方々に, 御自分の主張を展開していただきます。そのあとは, 会場のみなさんの御意見を, たっぷりと時間をとって聞きたいと思います。ただ, 司会者の私が言うのもなんですが, 個人的に, この話はまとまらないんじゃないかと思っているんですが, とにかく始めましょう。それではフィッシャーさん, コメントをお願いします。

フィッシャー[2] (コロラド大) こちらに来てから, 日本語が出来なくて恐縮しています。日本に来て, 困る度に「今度こそ, 日本語を勉強するぞ」と固い決意で, ボールダーに帰るのですが, 日々の雑事に追われて, また ボールダーで 日本語を喋らなければいけない必要性は非常に低いので, ついついサボッています。

まず, 最初に, この Davis の記事 [1] に関してコメントさせて頂きます。まず, 初めにこの記事を読んだときに思い出したのが, Fred Brooks が 4〜5 年前に書いた「ソフトウェア工学に, 銀の弾丸は無い (No Silver Bullet)」[2] でした。で, この Brooks の記事に書いてあったことは, ソフトウェア工学というのは, 非常に難しくて, どんなメソッドを用いようが, どんなツールを用いようが, 絶対にソフトウェアエンジニアリングが簡単になるということは無い, ということでした。

もう 1 冊, 本を紹介させて頂きます。トーマス・クーンというアメリカ人の書いた「科学革命の構造 (The Structure of Scientific Revolutions)」という本の中で, 人類の歴史を振り返ると, 人々は何かパラダイムを作り上げて, そのパラダイムに従うわけですが, 時間が経つうちに, そのパラダイムが変ってゆくとが書いてあります。ですから, この「レミンジニアリング」

---

* 1994 年 6 月 16 日 18:30 〜 20:50
† (株) ビジョン・コーポレーション
‡ (株)PFU
§ MHI エアロスペースシステムズ (株)
¶ Professor, Department of Computer Science and Institute of Cognitive Science, Director of Center for Lifelong Learning and Design (L3D), University of Colorado at Boulder
‖ (株)SRA
[1] [編集部注] SS94 論文集 pp.175 あるいは それを補足した本号の塩谷氏の記事の 表.1 レミングの軌跡 を指す
[2] 通訳は, フィッシャーさんの所で研究されている, 中小路 久美代 さんにお願いした

という傾向は, ソフトウェア工学に限らず, あらゆる人類の歴史に於てみられる傾向なわけです。

ですから, 問題となるのは, 他の人がするから自分もするというのが, 「宗教」なのか, それとも「科学」的な前提に基づいたものなのか?, そのどちらなのかということを考えることだと思います。その観点から, ソフトウェア工学の歴史を振り返ってみますと, 確かにある種, 宗教的な傾向がある。例えば, ある偉い先生が“あるメソッドを使うと, またあるパラダイムを使うと, ソフトウェア工学の問題が解決します”と言うことによって, みんなそれに飛びついてしまい, そのパラダイムを使ってきてしまったということです。

例えば, 1 つの宗教的傾向の例が, プロセス・プログラミングにみられます。7〜8 年前に, ICSE で言われ始めたプロセス・プログラミングというパラダイムに沢山の人が賛同して, コミュニティーを作り上げました。そのコミュニティーの内側の人達は, ある種, 宗教的な信仰者になってしまって, 一方で そのコミュニティに属していない人達が, そのプロセス・プログラミングの是非に対して科学的に論争しようとすると, 宗教的なパワーに押されて, なかなかその科学的論議が出来ませんでした。

先にも, 申し上げましたように, この傾向は, ソフトウェア工学に限らず, 他の科学分野でも言えることです。 例えば, 私が属してきました AI の分野では, 10 年程前に, エキスパートシステムが現れて, 「何か難しい問題があったら, 全てエキスパートシステムを作りましょう」と, みんな信じてきた訳ですが, 今になって, “これこれ こうして こうなる” と, 完全によく分かっているドメインでやっと作れるくらいで, 他の実際に難しい部分の問題では, エキスパート・システムは完全に失敗しているわけです。

こういうことを踏まえて“じゃあ私達がどういうことを信じているか?” ということについて, 少しお話したいと思います。そして最後に, Alan Davis 氏の書かれましたレミングジニアリングの記事について触れたいと思います。

この表[3]は, 過去に渡って 4 つの角度から見た計算機環境の変化についてまとめた表です。言語, ユーザ, メタファ, どこに物理的にコンピュータが存在するのか?, ということの変化をまとめたものです。この線より上が 今まで歴史的に起こってきた進化, それから今日の状況を表しています。この線から下の部分が, これから起こるであろう新しい方向性ですとか, 面白くなってゆくのではないかと現時点で思われている分野です。

例えば, この右端の列は, コンピュータが“どこに在ったか?” という変遷です。まず, メインフレームがあって, ワークステーションがあって, ラップトップに変って, そして今日, ペン・ベースのパーム・トップ・コンピュータなども出てきています。例えば, アメリカですと, 将来どうなるかというと, ここにも, そこにも, というように点在する (ubiquitous computing) のです。結局, コンピュータが見えなくなってしまって という状況に今後なるであろうと思われています。

また, 一番左はじのドメイン指向型 (domain oriented) の言語環境ですが, これはコンピュータそのものが, 業務の後ろ側に隠れてしまい, コンピュータを使っているという感じではなくなり, 単に業務を遂行しているというふうに変ってくると思われます。

これまで, 私どもの研究室では, ドメイン指向型の設計環境というものをずっと研究してまいりました。ここに 1 つのスクリーンの例を持ってまいりました。何人かの人達は「台所設計」というのを永い間してきましたので, それを期待されるかもしれませんが, ごめんなさい。これはネットワークデザインです。1990 年に, 京都でのソフトウェア・シンポジウムにお招き頂いて, その際に 詳しく, このドメイン指向型の設計環境のアイディアについては述べさせて頂きました [3, 4] ので, ここでは, 簡単に説明します。この設計環境では, コンピュータネットワークを設計します。この左下の部分に カタログという部分があり, 既に設計したものを貯めておくライブラリのようなものです。 これを使うことによって, 例えば再利用が可能です。このアプローチを用いて, いくつかのドメインに構築してきましたが, それを改めて解析して, どういうアプローチで我々が環境を作ったかのプロセスを解析した結果が, ここにお見せしているプロセスモデルです。

“種を蒔いて, それが徐々に進化していって, もう 1 回種を蒔き直す。” というプロセスです。それについて簡単にご説明申し上げます。

皆さん, もうご承知のことかと思いますが, ソフトウェア開発にかかる, 一番のコストというのは, 実は, 新規に開発する部分ではなくて, ソフトウェアをリリースした後に, ずっと断続的に生じる改善とか保守という作業にかかるコストの方が大きなわけです。このシステムを解析するに従って, 多くのソフトウェア・システムの開発プロセスに応用することが出来ることが分かりました。それでは UNIX を例にとっ

[3] フィッシャー氏が用意した OHP は、残念ながら編集作業に間に合わなかった。

て 簡単に説明します。

UNIX の例では 種蒔き (seeding) のプロセスは, UNIX を開発した AT&T の人と (UNIX を作って出来たアプリケーションのユーザではなくて) UNIX 自体のユーザである我々研究者との共同作業で UNIX システムが出来上がって来ました。過去, 15 年以上にわたって USENIX なるユーザコミュニティが形成され, その中で UNIX を使ってゆくうちに, 徐々にツール群とかコマンド等を構築してきました。
再種蒔き (re-seeding) のプロセスでは, usenix などのコミュニティで構築されたツールのどの部分を次のバージョンの UNIX のカーネルとして装備するか? といったことを検討するという re-seeding のプロセスがサポートされます。

この, UNIX の場合は ユーザがコンピュータのエキスパート/プロフェッショナルでしたから UNIX ユーザ自身が自分でプログラミングをして UNIX の環境を徐々に進化させてゆくことが出来たわけです。しかしこのモデルを実際に他のアプリケーションドメインに応用した場合には, そのドメインのエキスパートは コンピュータのエキスパートではありません。この場合, このプロセスが徐々に進化する (evolutional growth) 過程で, エンドユーザプログラミングですとか, エンドユーザコンピューティングといった手法が非常に重要になります。

Alan Davis の記事に戻りますが, 正直申し上げてこの記事はあまり大したことは無いと思います。Fred Brooks の書いた「銀の弾丸は無い」の記事の方が, もっと深く問題を追求して広い範囲で語っていたのですが, この Alan Davis の記事は, まあまあかなぁという感じです。

例えば, この表を見て頂くと判りますように, まずこの low/mid/high という危険度 (risk) と見返り (benefit) ですが, 好き勝手に付けてあるなぁと思いますし, 第一, 誰にとって危険で誰にとって見返りがあるのかが, そもそも明確でないのです。一番賛同しがたい点は, 記事全体が "機械から見たソフトウェア工学" という形で書かれているように思えることです。私達の経験から, ソフトウェア工学というのは "人本位" であるべきだと思います。

私が 3〜4 年前に出席した会合で, 玉井先生 (東京大学) と野村さん (JIP) だったと思いますが, 何がソフトウェアエンジニアリングの生産性を上げるのか, どういうプロジェクトが成功したかという調査がありましたが, そのときの結論では, 一番キーとなる要素は, プログラミング言語でも, プロセスでもマシンでもなく, 質の高いソフトウェアエンジニアを見つけることだ, という結論でした。両方の調査で, そのような同じ結論が出ていました。

青山 (富士通) 何故, 産業界は簡単にプロセスプログラミングなどのように, 小さなコミュニティを追っていってしまったのでしょうか?

フィッシャー クーンが言っているように, 人間は 誰か他の人がしているパラダイムをどんどん追いかけてゆく, という性質があります。例えば, あるコミュニティに属していることが, 何か知的な家庭 (intelligent home) に属しているように感じられ, そのコミュニティに属していないと, まるでホームレスのような感じられてしまいます。ちょうど小さい子供が両親のすることをずっと真似してゆくように, ボスとか前任者があるパラダイムに従っていると, その通りにしていないと不安に感じてしまうということです。人間は, そういうものだと思います。

青山 科学と工学の違いは, 科学では, 定理を打立てて それを証明するということがありますが, 工学の中には, そのような正しい答とか間違った答とかを証明する手段が無いので, レミング効果が生じ易いのではないしょうか?

フィッシャー ハーバート・サイモンが書いた The Sciences of Artificial という本があるのですが, それを例にとりまして説明します。
自然科学とコンピュータなどを使った The Sciences of Artificial とでは, 設計 (design) という違いがあって, 設計に於ては, 正しいとか間違っているといった証明はありません。したがって, レミング効果について, 哲学者のカール・ポッパーの言葉を引用します。
"例え自然科学であっても, 科学の発達というのは間違いを排除することで発展している。アインシュタインがニュートンを越えたのは, その間違いを発見したからだ。"

藤野 (富士ゼロックス情報システム) ポッパーやアインシュタインが登場したので, ここで ソシュールを出したいと思います。

結局 今, フィッシャーさんが "パラダイム" として言及した部分というのは, すでにソシュールが言及していて "言語共同体というのが形成出来ない限りは, そこで新しいムーブメントは無いのだ" ということを, 彼は随分前に言っています。したがって, ある言語自体が, 自然に思えるようになるのと同様に, あるパラダイムが自然に思えるようになるには, そのパラダイムを信じるという, 言語共同体が出来ないとしょうがない訳です。最近では, その言語共同体というの

は, ここ10年くらいの間に生物学的な見地からの研究で Autopoiesis という考え方が出てきているのですが …

フィッシャー　確かに, 仰るように 共通の話題に興味を持つ人々が共同体を形成する際には, 共通の言語を開発し, 概念の共有化を図ります。しかし, 問題は, 我々の経験を越えるような大きな変革に対してどう対処していけばよいのか, 追従していけるのかということです。

私が1974年のIFIPコンファレンスに出席したとき, "1980年に FORTRAN は生き残っているか?" というパネルがあり, 誰もが 「そんなことあるわけがない」という結論が出ました。そこで興味深いのは, トップクラスのサイエンティスト達が, 頭を寄せ合って考えて, 存続を否定する結論を出したにも関わらず, 皆さん御存知のように FORTRAN90 まで出現しています。つまり「どうしてその時, そのトップクラスのサイエンティスト達が, それを予言出来なかったのか?」ということを考えることが大切なのではないかと思います。

司会　それぞれの分野の専門家をパネラ方々に御招きしていますので, 各分野の立場から, コメントしていただきます。では, 長崎さんからお願い致します。

長崎　私は, この春まで大学院生でした。まだ就職して間もないので, プログラムを開発してきたプログラマの人達とは少し意見が違うというか, 見方が理想的な部分に走ってしまい, 現実を見ていないということになるかもしれませんけれども, 御容赦頂きたいと思います。

私に与えられたテーマは, オブジェクト指向についてです。私は, 大学の時に, Intelligent Pad というシステムの研究/開発に携わっていました。多分, そういうことからオブジェクト指向のテーマを頂いたのだと思います。

まずこの表[4]を見た時にですね, オブジェクト指向の他に "再利用" という部分がありますが, オブジェクト指向の中の有益な/ありがたいということの1つに再利用があるのではないかなぁ, ということを思いました。特に Intelligent Pad は, もともと画面上に見える視覚的なオブジェクトをユーザが (画面上で) 重ね合わせて貼ってゆくということによって, 新しい Pad を作り出す。つまり, プログラムあるいは, ツールを作り出してゆく。こういう考えに基づいて, その出来上がった Pad を, また別なものに再利用するということを, 大学に於て研究している間は, (再利用は) 当り前のように考えていました。ですから, この表の中の, 再利用は, オブジェクト指向の中に取り込まれてよいのではないか?と思いました。

したがって, オブジェクト指向を取り入れた時に, まず開発コストが劇的に下がるのではないか?。これは当然, 部品の再利用ということ, オブジェクトの再利用ということに絡んできているんだろうと思います。しかし, ただオブジェクト指向を取り入れたからといって, すぐに再利用性が高くなって開発コストが下がるか?というと, 答は多分 "No" であろう。

私は, Smalltalk という言語を使っていたのですけれども, その言語に於て, 例えばオブジェクトの色々なクラスを使って Pad などを作ってゆきますが, 再利用可能なクラスを作ってゆくというのは, 最初は中々一度で再利用可能なものを作るのは非常に難しく, まず作ってそれを使ってそのサブクラスを作った時に, インタフェースが適当ではないとか, 機能の分割がうまくいっていないとかの理由で, 再利用が旨くできない。そこで, 「もう1度クラスを作り直して」ということを何度か繰り返さないと, 本当に再利用性の高いクラスというかオブジェクトは出来てこない, ということを感じました。再利用可能なオブジェクトがあれば, 開発コストは (多分) 劇的に下がると思いますが, まずそれを作るのが難しい, ということがあると思います。これは Smalltalk に限らないと思います。

また, 再利用可能なクラス/オブジェクトがあり, それを沢山, 色々な種類を揃えれば揃えるほど, 全く新しく作る部分は減ってきます。けれども, 部品が増えてくると, 今度は, その必要な部品をどのように探すのか? ということがきっと問題になってくると思います。この部品を探してくるというのは, 今までのデータベースなどに, ほうりこんでおけばよいのか? データベースには, オブジェクト指向のデータベースとか色々ありますが, まあ "キーワード" なり "クラス" なりで引いてくることになると思います。実際には, (データベース中に) 色々な部品が蓄積されてきて, またその部品を豊富にする為に, 他人が作ったクラスやオブジェクトを貰ってきたり, 自分の作ったものをあげたりしてゆくと, 段々, その, 統一された考え方で, キーワード付けなどを行うことが難しくなってくるのではないかと思います。

そうなってくると, キーワードからの検索は段々困難になってきて, 必要な部品が, きっとどこかにあるのだろうけれども, それが見つからないということになってくると思います。つまり, 雑多な部品が在る

[4] Davis の掲げた表を指す.

ときに, その中から本当に自分が必要とするものをどうやって見つけてくるか ということが十分に支援されていないと, 再利用性を高めて, 必要な部品とか新しいツールなりプログラムなりを, 簡単に組み合せによって作り出すというのは難しいのではないかと考えています。

ちなみに, Intelligent Pad というものでは, どういう方法を考えているかというと, まずカタログ集を作ってパラパラめくって, それから見つける。あるいは Browsing によって捜し出す。

その次には, Hyper-Link などの構造を使って目的のものを捜し出す。これはあらかじめ その Pad を開発した人なりが link 構造を使って関連性のある部品を登録しておくわけです。

その他に, Intelligent Pad では, Pad という視覚的なオブジェクトの重ね合わせた構造で機能が決ってきますので, その貼り合わせの構造をキーとして, 構造から検索することが出来ないか?という研究が今行われています。

但し, この方法, 唯これだけあれば, どんな場合すべての場合をカバーするとは当然思えないわけで, その他にも別の何か方法があるのではないかと考えられています。

私の視点としては, このようにオブジェクト指向と再利用性というのは, 密接に絡んでいると思います。そのためには再利用出来る部品をまずどうやって作るのかという問題もありますが, それをどうやって見つけ出してくるのか? ということも, しっかり考えておかないといけないのではないかと, 当り前かもしれませんが, 思っています。

司会 どうもありがとうございました。後で, 会場からの声も含めて, まとめてコメント頂きたいと思います。では, 次は熊谷さん, お願いします。

熊谷 今晩は。えっ, かなり会場が硬いので, 柔らかくしたいと思います。フィッシャーが結構柔らかい話をしてくれたと私は思っています。

一応, 自己紹介代りに, 最近思っていることは, 世の中には, 色々な制度があって, 例えば有名な話では, 病院が無ければ病気という状態は生じていない, ということがあって, 私はこれは正しいと思うんですよね。制度が如何に我々を害しているかというのは, その"病気"という1つの状態をとってみてもよく分かると思います。全世界から病院を無くせば, 我々, 全人類からは, 病気が全部無くなる。本当ですか?(笑)

それぐらいがあってですね〜, 私が最近感じているのは, **制度を如何に越えて, 自然な行為をするのか?** というのがコンピュータ関係のエンジニアには一番重要だと思っています。そのためにはですね, 最近私がやっている方法はですね, **課題を まあ, 少なくとも一晩寝かせておく。** 寝かせておくと 私の頭の中ではですね, **意識と無意識が協調する**んですよね。協調して色々なことを ある課題に対して次の日に目覚めた時に解を出しておく, という何と言いますか, 頭で考えるのじゃぁなくて, 体で物事を考えて解を出すと。

だから, ここらへんをコンピュータとすごくスムーズな形でやるのが重要じゃないかと思っていますけれども。えっと, アンケートについて私の思ったのは, リスク (risk) というのは, 私は低いとかいうものではなくて, 0(ゼロ) だと思うのですよね。例えば, 色々な道具があってそれを使うか使わないかは, 本人の責任なわけね。「道具が悪いから」なんていうのは, なんたる自己主張の無さ, というか情けない生き方であってですね, 私はどういうメソドロジーでも道具でもリスクは多分 0 だと思います。

それと, あとはですねぇ, 思ったのは, 私はプロトタイプですけれども, pay-off の度とか lemming の度というのは, やはり ∞ (無限大) だと思います。私が好きなのは タオイストですから, タオというのは, あれですよね, 色々な物事を 2 つで考えるという。きしくも, これは西洋の考え方と一緒なんですよね。色々な物事があったら二項分類して, その差を比べてやりましょうというのは, ほとんど同じなんですけれども, タオの凄い所はですねぇ, 2 つに分けるでしょう。分けて, それは本当はそれは, その 2 つは通じているんだという所が凄いところですね。

西洋はずっと分かれっぱなしですね。だから離婚率は高いですね。東洋の方では, 2 つに分けるのは, まあよいのですけれども, 分けたあとが全然違うんですよね。その極まると, 他に移っちゃうと。

流行を追うということは, とっても良いことでですね, 流行を全然知らない人は話になりませんよね。その意味で, Alan Davis みたいに「流行を追うな」などというのは, 全然ノッていない過去の人の話で, ここに挙げられた項目は, 全部やらなくちゃいけないと思います。だから, その, 各立場で全部流行を追ってですねえ, 中に C とか追ってもしょうがないやつとかプロセスとか追いようのないやつとか, こういうのがありますけれども, そうでないやつは, 私は徹底的に追うべきで, まあ, 後はこういうカテゴリがおかしいから評価のしようが無いやつは, 私は 0 だと思いますけれども, re-use とか Comquats とかは当たり前の話であってね, 議論する必要が無いような気がし

ます。

で, ポジションペーパに, 私は, この場で答えるということを幾つか書いてあってですね, 一応, 私はミーハですから一応全部をやっているのですよね。一応 Structured は最初からやりました。これは Assembler でもやったし C でも FORTRAN でも全てでやれることは全部やりました。これは, 私は普通でよいなぁと思っています。Structured は, 究めるとカオス (chaos) になるんですよ。

それから OOP とか OOA とかは, これはもう 8 年くらい前になりますかねぇ, 自分の会社に Smalltalk とか移植したりとかを, やってですね, うまくいきませんでした。(笑)。私は, そういう意味でレミングでしたね, 完全に。断崖絶壁に走っていっちゃったという感じで。仲間の何人かを一緒に連れて海に入ったことは確かです。

それから, プロセスは随分試しましたけれど, こうした Alan の話す通り, 今, コメントのしようが無いですね。

C は, 当然, ほとんどの人が使っているから, 特にコメントは不要ですね。

あと, プロトタイピングはですねぇ, これはツールを作って売ったんですけれども, 確か今まで 2〜3 年になりますかねぇ, うまくいきませんでした。

それから CASE はですね, 先ほど 中小路さんの hyper-media と methodology とがどういう関係があるかなぁと, ちょっと思ったんですけれども, 質問すると失礼かなぁと思って控えたんですけれども。これは, hyper-media + methodology という形でやっていて, これはまだあまり売っていませんので, これはまだ海にはつっこんでいないという状態です。

それでですね, 私のペーパの所にですね, あの不易と流行を話さないと, 話にならないよ, というのを書いてありますので, 一応, 不易と流行を書いて見ました。80 年代と 90 年代と 2000 年代の不易という変らないものと, 流行 次々と変ってゆくもの。で, 私の老子の考えに基づくと, これとこれとは密接に関係していて, 本当は相通じている理論なんですけれども, ここでちょっと話したいんですけれども, 時間ですので, もしあとで時間があれば話してみたいと思います。

それから最後に, 私の立場としてプロトタイピングというのが, 結構あっちこっちで引用されていましたけれど, で, Alan Davis はですね, 「quick and dirty だから, もう今のプロトタイピングは, もう駄目よ」と書いていますよね。で, ちょっと私, 非常にその, 嫌いと言うかおかしいなぁと思っていまして。プロトタイピングはですねぇ, incremental にスケルトンからどんどんやってゆけばよいという考えもありますが, 私の考えだとプロトタイピングはですねぇ, 分析とか設計そのものをやる artifact というか道具だと思っているんですよね 1。それを incremental していって最後にその現場で使おうなんていう, ある意味で意識が低いというか, セコイ考えで物事にあたっては駄目だ。もともとプロトタイプは私の考えではですよ, スケルトン型となんとか型であるというのは, 私は全然意味がなくて, 色々な人が沢山いるのでですね, 物を分析したり設計したりするための道具としてプロトタイピングの技術を持ってきて, それに対する artifact を作ってですね, これを見ててですね, 生きる勇気を養おうというのが, 正しいというか, タオというか, 自然な生き方として見て, より良いと思っています。ですから Alan Davis がですねぇ, これを使って, こう, みんな楽しむ訳ですよね。それ, だから "quick and dirty" でいいんですよね。とても良いと思います。

で, 別の人が言っていますが, プロトタイピングというのは CASE の一種で, 上流工程だろうと思います。で, 結論としてはですね, Alan Davis は "全然 物を分かっていない!, 彼の小論は, 全然議論するに値しない" というのが 私の意見です。どうもありがとうございます。

司会 じゃあ, 伊藤さん お願いします。最後の突っ込みを期待していますので ••• 。

伊藤 え〜, 一応, 打ち合せでは, 役割分担というのが出来ていまして, 熊谷さんがボケて, 私が一応突っ込むということになっていましたので, そういう風に振舞いたいと思いますが, これはけっして私の本意ではないんですが, role (役割) として まあそうしたい。で, 今, 熊谷さんのお話の中で, Alan Davis は分かっていないなんていう話が出てきましたが, え〜, 熊谷さんも分かっていない訳ですね。実はですね, パネラは自分の担当分の統計は自分でとるという約束になっていまして, 私の方はこのようにとっています。

それで, これが今回の皆さんのアンケート結果です。で, 今回 奇数/偶数 という話が電中研の高橋さんの発表中で出てきましたが, 高い/中間/低い というのがありましたが, ここでは 2/1/0 と点数付けした結果がこれです。え〜, return と書いてあるのは pay-off のところですかね。そうすると CASE に関わる所は どういうふうになっているかというと,

risk	0.806
return	0.710
レミング度	0.759

これは 高橋さん, どういうことなんでしょうね?。やはり, 悲観的ということなんでしょうかね?。よく分からないですね。え〜, 1 つ言えるのはですね, 平均が 1 ですから CASE というのは関心が低いと。

で, CASE の方はですね, ちょっとこれは色々な人がいて, まとめきれなかったんですが ••• 。大きく 4 項目ありまして, 最初の 2 つというのが生産性に関わる話, あとの 2 つというのは品質に関わる話です。Davis の話は, CASE と言った場合, 生産性の向上と品質の向上というのが主要なポイントなわけですね。それが最初は ちょっと生産性が上がる。ここすごかったですね。 次は, 劇的に生産性が向上する, と書いてありましたね。それからドキュメント品質を向上させる。それから品質を向上させる。

それで, 皆さんの項目毎に書かれた人を見ると, この生産性という所は risky だという回答が多かったです。risky だけれど, 当然のことですが何せ劇的な生産性ですから 当然劇的な high-return があるはずなんですね。ところが何を勘違いされたか, この劇的な生産性は low-risk なのにも関わらず low-return だと書かれた方がいて, それはちょっと理解出来なかったんですが ••• 。まあ, こういう結果になっています。

これを見てですね, この会に参加された皆さんが, 如何にすばらしいソフトウェア技術者かということが判ってしまったわけですね。何故判ったかという証明を次にしたいと思いますが ••• 。

えっと, 次に示すグラフというのは IEEE の SPECTRUM に載っていた表 (表.2) なんですが, 3 年くらい前ですかね, ちょっと古いんですが。この中で何を言っているか。ポイントは, 品質の向上にあるということです。生産性そのものは, CASE を導入した直後は下がると。そして, しばらくすると向上するんだけれども, それはけっして劇的に向上するわけではなくて, ある一定の規模でしか向上しないというわけです。向上し続けるのは (し続けるかどうかは判りませんが) 品質と保守ですね。これは, 高いレベルになると。

で, こういうのを見ますと, 皆さんのアンケート結果というのは, 非常に良い所を突いているわけですね。品質に関しては low-risk で, それなりの効果があるよ, と皆さんお答えになったわけですね。劇的な生産性というのは, high-risk で, もしあったら high-return かもしれないけれども, それは非常に risky な話だよね, ということで。やっぱり皆さんの CASE を見る目というのも, 生産性よりも品質にあるんだということで, さすが SEA の皆さんだ! と私は感動しました。

じゃあ, タオの話を熊谷さんがされたので, さっき書いた絵をちょっと見たいと思います。まあ, フィッシャーさんの見方とか, 熊谷さんのお話とかとも関係してくるんだろうと思うんですが, これに類似したことは SIGENV の長野のワークショップの報告書 [5] の中でも書いてますので, 細かな話はそちらを見て頂くともっとよく分かると思うのですけれども ••• 。西洋的な物の見方は, 何かと言うと「上からの見方」。例えば, ヘーゲルの弁証法的な物の見方というのは完全に自分は他人な訳ですね。どこか外から見ると。で, それは科学的な言葉, 技術者が使う言葉で言うと, “客観的” と言うわけですね。自分はそこに居ないわけです。どこか遠くに離れてウンウンとうなずきながら, 「俺は客観的に見ている」と言うわけです。

先ほど, ニュートンとアインシュタインの話がありましたが, 今までこういう物の見方をしてきたと。所が, こういう事例に関しては矛盾があると。で, こう考えたら。「こう考えたら」というのは一歩下がるわけですね。そこから, 物の見方としては, その矛盾がうまく説明出来るというのが いわゆる「弁証法」です。これは非常に西洋的なものの見方 (図.2) で, 当然この見方には限界があると。これを究極的に続けるとどこに行くかと言うと, それはヘーゲルが言ったように, 絶対他, 神の世界に近付くわけですね。で, そんなこと あると思う人は幸せですけれども, 多分無くて。

東洋的な物の見方というのは, どういうことかと言うと, 私は曹洞宗なんですが, 曹洞宗の人はここには非常に少なくてですね。SEA の主流はどこでしたっけ, 臨済ですね。で, 曹洞宗, 大本山永平寺の開祖 道元 という人がいますが, 彼の「正法眼蔵」という本の中に, ある言葉を引きましたが,

「万法はこびて修証をなす」

これはどういうことかと言うと「求めると得られない」ということです。例えば, あるものを求めるとですね, 逃げられるわけですね〜。ところが, ある状態に自分を達することによってそういうものは, 自然と自分に来る。それをちょっと絵に書くと, どういうことかと言うとこうなる。(図.3) つまり, 客観的, そこから一歩離れて何か物を見るのではなくて, その対象そのものになりきるということです。で, これの非常に良い話を。えっと, 私, 名古屋で ローカルな会をやっているのですが, 以前, 塩谷さんに来て頂いたときにですね, 以前 彼が仕事をしていた時にですね, コーディングリストが頭の中でスクロールしたとい

う訳ですね。これが多分1つの この絵のような状態になるのではないかと思います(笑)。もう, 凄いですよね。頭の中ですから, もうマルチウィンドウでもなんでもありですから。

で, そのものになってみるという見方が, 東洋的な物の見方, 多分 それは タオにも通じるんだろうと思うんですが, それは例えば, あの私はもともとハードウェア設計だったんですが, 例えば 3 次元というのは図面では書けないわけですね。三角法とか一角法とかを使って書くわけです。所が, 慣れてくるとですね, 図面を書くと 3 次元の図形が頭の中に出来るんですね。例えば, 核磁気共鳴装置というのがありますよね。あれに熟練した お医者さんというのは, 要は, ある断面図を見てゆくことによって, その患者の全体像が頭の中にイメージ出来ると。つまり何を言いたいかをいいますと, 「CASE は所詮ツールだ」ということです。それを使うのは人なわけです。で, 今回のタイトルを私のポジションも「道具としての CASE」というタイトルにしました。CASE は所詮ツールです。それを使うのは人です。で, 道具はあくまでテーラブルで使いやすくて, 我々が直接には見えないものを見せてくれればよいわけです。ただ, それを使いこなすのも我々です。で, その見えないものを見せるんだけれども, そのツールの在り方というのは, 実はそれとは全然逆だということです。で, これで最後にしますが, 私の翻訳した ACM の interactions[6] の中で, "道具の関わり" という話があって「見えないことが大事だ」というわけです。例えば, 近眼の人が使う眼鏡がありますね。眼鏡は非常に有用な道具だけれど, 我々は眼鏡を使って外を見る時に, 眼鏡を意識していないですよね。眼鏡を意識したら, 外は見えなくなりますよね。つまり道具の本当の在るべき姿というのは, 見えないことにある。そうすると, 今の, 多分ですね, 私, つっこみですので, 皆さんに嫌われるように言わないといけないんですが, 例えば「マルチメディア」とかですね, 先ほど フィッシャーさんのお話にあったような 「どこにでもあるコンピュータ」。それはどこかに埋め込まれて直接的には見えないかもわからないけれども, 例えばスタートレックにあるように, このへんを "ピッ" と叩くと, どこからか通信が出来て, どこからか音が聞こえるようなものですね。それから「知的エージェント」。で, これらは, CASE を道具だとすると "解" にはならない。 一言で言います。『まがいもの』です。良心的なレミングとならない, 真面目なソフトウェア技術者の扱うべきものでは無い。それは何故かと言うと, 見えない (ものであるべきだという) ことに反しているから。ということで, おわります。

司会 どうもありがとうございました。計画通りに進むということはなかなか珍しいんですが, 時間が押している以外は役割分担が巧くいったようです。では, 残り時間一杯, 会場の方からコメントとか御意見とか頂きたいと思います。

平野 (中央システム) 常日頃から感じていたんですが, 今回のレミング (lemming) という話を聞きながら最初に思ったのは, ソフトウェアはハードウェアよりも宗教に近いなということです。今回は lemming の後ろに engineering と付いていますから, 本来, 技術論に絞ったはずなんでしょうけれども, もっと広くソフトウェアを使う人まで考えてみると, 私のまわりには非常に Macintosh レミングが多いんです。ユーザがそれだけレミング化するということは, エンジニアリングがレミング化するのは当り前じゃなかろうかと思います。ある意味で, 先ほど フィッシャーさんのおっしゃられた宗教的な部分と似ています。ソフトウェアが, 目に見えない, 触れないものだから宗教と同じなんじゃないかと思うんですが, 特に 熊谷さん, どう思いますか?, タオの発想から見て。

熊谷 レミングとか流行というのは, 別にソフトウェアに限らず, 科学でも工学でもありますよね。例えばアインシュタインのようなすごい人が出てくると, 哲学や人間の生き方にまで影響を与えてしまった。そういう意味では, ソフトウェアだけに限らず, 常に不易と流行というのは, ずっと続いていると思いますね。ですから, "とりたててソフトウェアだから" っていうのはあんまり面白くない議論のように, 私には思いますけどね。

伊藤 私はもともとハード出身なんで, 半分御意見に同調して, 半分反対です。 ソフトウェアは見えないから, と言うんですが, プリントアウトすれば, "立派に" 見えます。一方, 私はもともとトラックのミッションの設計をやっていましたけれども, 例えば実際に, 新型ミッションを作って実験するとき, ミッションの中で自分の設計したギヤがどうなっているのかは, ハードであっても見えないんです。それを見ようとすると, ミッションを壊すしか手段は無い。だから, ソフトだから見えないという言い方は, それは正しくない。「ハードでも見ない部分は一杯ある」ということです。ただし, ハードと比べて, ソフトウェアが対象とするドメインは非常に広い。例えば, 私が自動車会社にいたときに, 対象としたのは自動車だけなんです。ところがソフトウェアに転職して私が経験した

ものは, 事務計算もありますし, 技術計算もありますし, 今やっているような環境の仕事もある。で, そういう意味で, ソフトは, ハードと違って非常に広い範囲を扱うメタなエンジニアリングであると考えていますから, その点では宗教に近い というご意見に同意しますということです。

平野 分かりました。1つだけフォローしますと, ちょうどアインシュタインが出てきましたが, "ソフトウェアというのは物理的な法則から自由" なんですね。それが人間を非常に不安感に導く。あるいは逆に, 喜悦に導くと。そんなことをちょっと考えたからでした。

長崎 今仰られたことと同じで, ソフトは自由度がハードに比べて非常に高くて, 作ったなりにそれなりに動いてしまうんで, どれがベストかというのがなかなか分かり難い 所があると思います。だから, 宗教と同じで, 誰かが示した道を皆が取り敢えずついていってしまうのだと思います。

司会 どうもありがとうございました。それでは 酒匂さん。

酒匂 (SRA) コメントなんですけれども。最近, 私が読んだレミングニアリングの話なんですけれども, 「相対性理論は間違っているんじゃないか」という話が広がりつつあるのだけれども, 何しろ 90 年間も信じられてきた理論だし, 大アインシュタイン博士の言ったことだから間違いなかろうというようなことで, この動きに関しては黙殺という雰囲気が流れているという話を聞きました。だからレミングニアリングがいけないと言いたいのではなくて, 相対性理論はちゃんと世の中の役に立っている。先ほど, 宗教と科学という話があって, 両者を二項対立で捉えてきたわけですね。私の考えでも, 宗教と科学というのは, 構造的にも似かよっているし, 少ない原理/原則から, それ以上の体系を演繹するという所も非常に似かよっていると思います。ただ, 一番最初の所の公理 (axiom) というところが, 科学は fact (事実), 宗教は fiction (空想) という両方とも "f" で始まるものです。そこで, 西洋の哲学者/科学者は "最初がつまずくと後が全部違ってしまう" ときっと言うでしょう。しかし, 例えば熊谷さんの考え方では, fact と fiction はあまり区別が付かないんですね。そうすると私達がレミングニアリングだと言って切捨てられないための方策は, この最初の "f" が fact もしくは fiction のうち, 少なくとも今の時点で, どちらに近いかを, 多分大学にいる人達とかが考える。そして, 技術者は, そこから演繹されている体系, もしくは帰納的に導かれる体系に関して少なくとも自分達が, 論理的態度をとっているかいないか, というようなことを, 日々, まあ何となく/ちょっと意識しながら進んで行く。といったようなことぐらいなんだろうか, ということを考えました。いかがでしょうか?

熊谷 酒匂さんに, 逆に質問しますが, 私はあれですよ, fiction と fact というのはね, ほんとはあまり差が無いんですよね。

酒匂 そう言いました。

熊谷 そう言いましたよね。そう言ったときに, 日々の我々の行動が "論理的" とか言わなかった?

酒匂 え〜, だから, せいぜい, 例えば西洋の人達からですね, レミングニアリングだ言って切捨てられないために, 私達が彼らに対しての出来ることで言えば, 自分達がロジカルな対応をしているかを意識しているフリをするというか, そのような態度を見せるといったようなことしかないのかと思った訳です。

熊谷 その時の "ロジカルな" という基準はどこにあるんですかね?。さきほどの 2 つの fiction と fact の間にある基準を設けて, それを認めているのでロジカルだ, というのは分かるのだけれども, それがほとんど同じような形だと言ってしまうと, そのロジカルというのは, どうなってしまいますかねえ?

酒匂 今ここでは, 単なる「三段論法」以上のことは言っていないんです。世の中には, 「A ならば B, だから D」なんて言い方がありますけれども, 例えばそういう問題には陥らないといった程度の話です。

司会 では, 次の方どうぞ。

宮田 (SRA) 現役プログラマのつもりの宮田です。話が飛ぶんですけれど, 今年の 1 月に「オラクル・オープン・ワールド」という大々的なショーが, 横浜パシフィコで行われました。この時のオラクルの宣伝は, 日本経済新聞を 2 面ぶち抜きでしたけれども, 書いてある日本語はたった一言「その日, あっと驚く」。実際に会場に行っても, イメージフィルムを流しているだけで, 商品がどうだとか, どんなものを売っているかという説明がまったく無いんです。私は主催者側にいて, 彼らが何をしているのか全く解らなかった。

もう 1 つ, マイクロソフトがテレビで, 「Windows, Windows, Windows, ... 」と喋りっぱなしのコマーシャルをさんざん流している。何の広告なのか素人にはまったく解らないんですけど, 延々流している。で, これも彼らが何をしているんだか私には全然解らなかった。けれども, 昨日, 予稿集をめくって レミングニアリングという言葉を見て, 彼らが何を狙っていたかやっと理解出来たんです。彼らは, なんとレミングを作り出しているんですね。そこで お願いが

あるんですけれども, レミングを作り出すことは, おそらく資本主義社会の中では簡単に成功するのだと思います。アメリカの企業は既に始めています。今, Intelligent Pad が出ていますし TRON とか色々なアーキテクチャが提案されていますが, 黙っていると消滅すると思うんですよ。で, Intelligent Pad レミングというのを作り出すような努力を是非やって頂きたいというお願いです。

長崎　Intelligent Pad レミングを作っているという話ではありませんが, 今, 既にコンソーシアムがあります。まず色々な所でデモをして, サンプルを使って色々とデモをしています。その他にも, 評価版というのを配ってまず使ってもらって, ということで色々と努力はしています。まあ, Intelligent Pad みたいなシステムの場合だと, 言っても なかなか判ってもらえない, とにかく見せて, 如何にも簡単だという所を見せなくてはいけません。そういった意味で, 今は一生懸命レミングをつくろうとしているところです。

伊藤　最初の話で, 「わからないコマーシャルが多い」という話をされたので, 「パネラの話は解らんからなんとかせい」というお叱りの言葉かと思った, という話は置いといて, 私は「それはやっぱりいけないよね」という立場です。

レミングがお金になるというのは, それで儲けようとされるのならよいんじゃないでしょうか。けれども, SEA としてソフトウェア技術者として儲けるために, 何かを求めるならよいんですが, そうでないとしたら, 追いかけること自体 意味が無いように思うんですが ••• 。

宮田　ここにおられる方には, レミングになって貰いたくないです。客観的に見ていて欲しいですね。

熊谷　レミングは, あれはみんな死ぬんだよ。だから, 行く先は, 断崖絶壁の海か何か知らないんだけれども, そこまでバーッと走って気付いても遅いんですよ。もう死ぬしかないから。唯, みんな人間は死ぬために生きていますから, それでもよいとは思いますけれどもね。

司会　じゃあ, 中野先生。

中野 (大阪大学)　熊谷さんから「課題を一晩寝かせておくと, 意識と無意識の間で協調してそれが解ける」という話があって, 熊谷さんは, その OHP を非常に気に入っておられるようなので, 誤りを訂正しておきたいと思います。え〜っと, 最近色々なことをやっているので, 私の専門がわかってもらえていないと思うのですが, 私は一応基本は「アルゴリズム屋さん」なので。 課題を一晩寝かせておいて, 意識と無意識の間で協調して出てきたことというのは, それは本当の答でなくて, トリガーだと思いますね。だから, それが, 色々うねうねと曲っていってようやく最後の答になる。だから, 一晩寝たぐらいでは, ちゃんとした答は出てこない。きっとそれは, 熊谷さんが無意識と意識と書いてあるけれども, "酔っぱらってふっと気が付いたら何かを思い出した, それは何かのトリガーになったんじゃない" という風に話をして頂きたい。結論は何かと言うと, その OHP はあまり使わないように! だって, そんな単純なもんじゃないって。

熊谷　さっき, 説明の時に言ったじゃぁないですか, "一晩とは限らないけれども" と。あれは 1 年から 10 年のこともあるんですよ。例えば, 私の経験でいくとですね, これはある 1 つの場合なんですけれども, 凄い難しいバグがあってですね, ××石油さんに納めたシステムが 1 年に 1 回ぐらい システムダウンするんです。それがどうしても解らなかった。でも, あるとき夢の中でバグを見つけて, 本当にやってみたらそのせいだったというのがあって •••。私が "寝かせる" といったのは, みんな寝かせずにシステム作りをやるから, みんなギスギスして 正解が見つからない。寝かせておくというのは, 今でも重要だと思っています。

中野　じゃあ, その「一晩」という所に, "1 年" と "10 年" というのを入れておいて下さい。(笑)

司会　次は高橋さん。

高橋 (電力中研)　レミングだったら, 何故悪いのかなぁ? みんな同じ方向に走るんだけれど, 実は, 川が来たら止まっているんじゃないかなと思っています。過去にも 例えば 構造化〇〇とか "FORTRAN はもう駄目だ" 神話に乗って走って行った人達で, 今こけている人がどれだけいるんですかね。自分が過ちを気がついたら, そおっとよそを向いて違う方向に走って行けばいいんだから, 結局誰も溺れ死なないじゃないですか。自信のない人間は, 人が走っている中を一緒に走って行くと, すごい安心感があるんですよね。とにかく, 我々はいつも真っ暗な中で歩いてるみたいなものです。で, 誰かがパッとランプを当てたら, 皆でそっちに走って行く。こっちにランプが点灯すれば, そっちに走って行く。

だから, レミングみたいに走ってゆくのは構わないと思うけれども, いざとなった時に, 方向転換出来るように次のことを考える人がいなきゃぁいけない。日本がアメリカや中国と戦争をやったときに, どんどん追い込まれていって, 引き返そうと思ったんだけれども, 引き際をしくじったんだと思うんですよ。だから, ソフトウェアの世界も, 引き際をしくじっちゃぁいけ

ないなぁっと思うんですよ。

司会 でもね〜, レミングになって, わっと走り出すような人達がいるとして, で, その中で冷静になってパッと「止まろうよ」って言う人がいると思います?

高橋 「止まろう」って言う必要は無い。「こっちの方が明るいよ」っと言えばよい。

熊谷 そうそう, 逆なんだよね。どんどん行って, もっと別の所に行けばよいんですよ。

高橋 だから「あれが失敗したから, これをやろうよ」と誰かが声をあげるでしょ, そうしたら, みんなワァーとそっちに走って行くじゃあないですか。エキスパートシステムだ, オープンだ, SI だ, とか。あっちのパラダイムがうまくゆかないから, こっちに走ろうよという流れをね, 同じ1つの会社がやっている訳ではないでしょうけれども, どっかがやったら, 「ワー」とみんなそっちに走って行く。その間に, こっちは貯め込んでおいて, 次のパラダイムを出せば, またこっちに人が来るという, そういうような順繰りが出来ているんじゃぁないかなぁと思うんだけれども。

司会 でも, それは崖から落ちなかったから出来た。そうなっただけであって, もし本当に崖まで行って落ちていたら ••• ?

高橋 いや, 落ちる時には, ユーザも含めて皆一連托生ですよ。今, ソフトウェアハウスが亡びたら, ユーザは皆 困るんですよ。生かさず殺さずしなけりゃぁいけないんだから, 落ちたりはしないんです。

熊谷 いや〜, 高橋さんのそばには良い人がいるじゃぁない。私の周りには, 断崖から落ちて死んだ人が一杯いますよ。だから, そういう人は, 人知れず死んでいるんですよ。だから生きている人達がね, 「死人は出てない」というのは間違いで, もう少し彼らの挽歌を聴かなければ駄目ですよ〜。

司会 じゃぁ, 二木先生。

二木 (北陸先端大学院) 「死ぬ」「死なない」という話ではないですが。ちょっと哀しいなぁと思うのは, みんな冒頭の表にあるのは皆外国で出てきたアイディアなんですよね。日本人が踊らされて死ぬの死なないのと言っているのがなんか淋しいですね。我々がプロジェクトを起てる時も, 日本独自のアイディアではなく, 向こうのアイディアでやっていることが多いですよね。で, 最近, "東洋思想" だとか, "全体主義 (トータリズム)" だとか出てきて。まぁ, ソフトもそういうことでね。日本の思想が, こう, タオかもしれないけれども, 本当言うと, 「出来るかなぁ」と思うわけですよね。で, 私は所詮プログラムだって, せいぜい Lisp を1万行くらい書いたとか, そんなことしか出来ないんですけれども, 比較的 西洋というか向こうの establishment アカデミズムの establishment に付き合う機会があって, 見てると, 彼らは自信を持っている。死ぬの死なないの言っても, 死に方にも自信がある。かなり覚悟を決めてやっているという面があります。

そこがちょっと, その迫力が違うというか, 淋しいなぁという感じがして •••。で, それはどうしてかなぁと言ったら, 先ほど言ったレミングで走る, 信ずる先がですね, 今日のは, 曹洞宗でしたっけ, 臨済宗, 曹洞宗の話が出てきたから心強いですけれども。今日 話題になったアイテムが, 全部向こうの人が言ったことなんで, 次回, こういう時にやるときにはですね, 伊藤さんの曹洞宗と熊さんの タオイズムをですね〜, タネに出来るかなぁと, そういうことをちょっと夢見てます。

藤野 先ほど, オブジェクト指向というのがレミングだと言いましたが, 確かにそうで, わざわざ言わせて頂ければ, レミンジニアリングという形の中の1つにしようとして, 色々と企んできていたわけですからね。で, それはそれでビジネスとして成立すればよいわけで。で, もう1つは, その, 1つはその pretend [5] してやっているんだ, という部分は, 前もってきちんとわきまえておいた方がよいと思いますね。

岸田 (SRA) 臨済宗の岸田です。そこにいる3人の人は, みんなオブジェクト指向レミングを作りたいように聴きましたが, 伊藤さんが, ほら, 客観的に物を見るのは間違いだ, と 西洋的な。でも, 客観的に物を見るのが, オブジェクト指向なわけでしょ?。熊谷さんも, タオイズムだとか言いながら, オブジェクト指向が好きとかいうのは, どうも自己矛盾しているのじゃないかと思いますが ••• (笑)。

伊藤 何時まで経っても, 臨済は曹洞に勝てないところでして ••• (笑)。私がお話したのはですね, 客観的な物の見方というのは, 近代科学を作ってきた1つの重要な見方です。で, 私が言ったのは, 単にですね, そういうヘーゲルの弁証法的な物の見方, いわゆる我々が客観的な物の見方の他に, もう1つの物の見方, それは「対象そのものに自分がなりきって, その対象そのものを把握する」という見方があるんじゃないでしょうか, ということをお話しただけです。

熊谷 その点というのは, 私のタオの考え方も全く同じなんですよね, 私も見方としては。ですから, さっき中野先生に誤解された "寝かせておく" というのは, 別に何もしないわけではない。ものごとの見方を客

[5] ... のふりをする

観的だとか, 客観的, 科学的, 分析的, とかじゃなくて, 主観的, なおかつ, その自然な形で見るということの方が重要だという意味ですから。

岸田 そうするとさぁ, オブジェクト指向の成果って再利用出来ないよね, 主観的だから ••• (笑)。

熊谷 なんですか, その "再利用" って

岸田 オブジェクト指向の "売り" は, "再利用" なんでしょ。

熊谷 "売り" はどうでもよいですけれども, 我々 人生というか 我々の社会全てが再利用で生きていますよね。その意味で, 再利用でない物は殆どありませんよね。人類の, ここら辺のフィジカルな物もそうだし, メンタルなものもそうだし, 文化とか文明とかいうものも, みんなそうですしね。そういう意味では, 再利用は, ただ単に私の理解によると「組み替え」に過ぎないですから。だから, 生産というのは, そもそも組み替えなんですよね。原料から組み立てているだけに過ぎないので, そういうのは全て再利用なので, "何を今更 再利用" という感じで, それは, 殆どもう議論の余地が無い所だと, 私は思っていますけれども •••。

伊藤 今の熊谷さんの発言は, 多分暴論だと思うのですけれども,「対象そのものに成り切る」というのは, どういうことかと言うと, ここに缶がありますね。これは缶として認識していますが, 私の方から見て (フロアーの方から見えているような) アルミの形状があるなぁと想像するわけです。また, あるいは「今, この中にジュースが入っているな」と想像するわけです。つまり, 客観的な物の見方というのは「客観」という言葉にだまされそうですが, 実はそこに色々な憶測というのを取り込んでいるわけです。で,「その対象に成る」という見方というのは, これは本当に東洋的な物の見方なんですが, ちょっと抽象的な物の言い方をしますが,「自分を殺す」(滅私) という言い方をします, え〜, 禅の世界では。それはどういう事かと言うと, 自分の憶測というのを全部除外して, 見えるありのままを自分が受け止め認識するということなんです。で, それは, あるがままに物を受け止め認識するということなんですね。

高橋 それはユーザの言いなりになるってことなの?

伊藤 いや。ユーザがこう望んでいるということであれば, それは,「ユーザがこういう事を言っている時は, このユーザは何時も嘘ばっかり喋っているから, きっとこんなこと望んでいないだろう」といった憶測を取り除いて, 自然な素直な気持ちでユーザの話を聞くという。だから, それが所謂「悟の境地」という訳で。

熊谷 まあ, そこらへんが, 私はさっき 言わなかった所なんですけれども。藤野さんがですね, 言葉というのは 1 つの共同幻想社会で, 言葉のコードがなければ, 例えば議論とかコミュニケーションが成り立たないと言っていました。 あれはあれでよいんですけれど, 本当は, そのコード系から抜けて, さっき言ったように, その物に成り切るというのはそれに殆ど近いんです。 そのコード系を抜けた形で 理解をシェア (共有) 出来るかどうか, じゃぁないかと思うんですよね。ですから, 制度とかコードを介してしか理解出来ない場合は, 本当は, それは理解し合っていないから, しょうがないんですよね, もう。

酒匂 伊藤さんと熊谷さんには, この道を object-oriental と呼んで, 西洋に向かって宣伝してゆきたいと思います。どちらかと言うと, inscrutable[6] object-oriented という感じですけれどもね。

司会 喋りたいだけ何時までもやってよいと言われているんですけれども, そろそろ一応締め括らなければいけません。当初予想したようにですね, 全然締め括れませんので, 締め括り方の 1 つとして, 一番最初にフィッシャーさんが上げて下さった「銀の弾丸はあるのか?」「銀の弾丸は何なのか?」という話を一言ずつして欲しいと思います。まず フィッシャーさんからお願い出来ますか?。つまり, 銀の弾丸というのは, 理想を追い求めた良い面で, Davis 流のレミングニアリングというのは,「ここまで行くと悪いよ」という例だと解釈したときに, じゃぁ, 銀の弾丸は何なのだろうという話にしてみたいと思うんですけれども。

フィッシャー 銀の弾丸がないというのはそのとおりです。もしも誰かが見つけてしまったら我々は皆, 職を失ってしまうし。しかし, ソフトウェアエンジニアリングは広範囲な問題を様々な観点から様々なアプローチで取り扱うことができるのがいいところです。アメリカには "千の華を咲かそう" という言葉がありますが, 私もそのとおりだと思います。

司会 ありがとうございました。じゃ, 今度は逆順にいきましょうか, 伊藤さんの方から。一言。

伊藤 単一の解は, 多分無いでしょう。で, あれば 皆さん, ここに集まっていないはずです。個々の努力の組み合せとして, 多分, 全ての工学がそうであるように •••。だから もしあるとすれば, SEA が銀の弾丸?

熊谷 格好良すぎる。格好良すぎて, 隣で私は気恥ずかしくて何も言葉を言えませんが •••。私の解釈では, 銀の弾丸というオールマイティがあるかという設問は, それ自体がナンセンスだと思います。私の感じ

[6] 探索出来ない, 不可解な

だと常に, “問題のある時にはソリューションがある” と思います。無ければ全然つまらないし, もう死んだ方がよいような気もあるから常に問題があるということを認識した時点で, 常にソリューションが有ると思っています。そういう意味では, そういうオールマイティの銀の弾丸じゃないんだけれども, 細分化された銀の弾丸はあると思いますね。それはまるでファーストの‘メンフィストフェレス’と “人間” の対応のような, 私は気がしています。

長崎 私は一応 オブジェクト指向というテーマを与えられていましたけれども, そのオブジェクト指向の分野での銀の弾丸, その色々な問題を一気に解決してくれるような方法というか, それが私は IP に関わっていますから, それが IP であればよいんですけれども。

司会 どうも, ありがとうございました。え〜, 長崎さんが, あまりハッキリ仰らないので「日本で生まれた技術である IP が銀の弾丸になるのかなぁ?」ということで, 締めたいと思います。どうも夜遅くまでありがとうございました。

参考文献

[1] Alan Davis, Software Lemmingineering *IEEE Software*, Sep. 1993 pp.79-84

[2] Fred Brooks, No Silver Bullet, *IEEE Computer*, April 1987 pp.10-19

[3] Gerhard Fisher, Cooperative Knowledge-Based Design Environments for the Design, Use and Maintenance of Software, ソフトウェアシンポジウム '90 論文集, June, 1990 pp.2-22, ソフトウェア技術者協会

[4] Kumiyo Nakakoji(中小路 久美代), Gerhard Fisher, Cooperative Knowledge-Based Design Environments for the Design, Use and Maintenance of Software, ソフトウェアシンポジウム '90 論文集, June, 1990 pp.264-270, ソフトウェア技術者協会

[5] 伊藤 昌夫, 青空の彼方から (2), *SEAMAIL*, Vol.9 No.2-3 (1994), pp.54–58

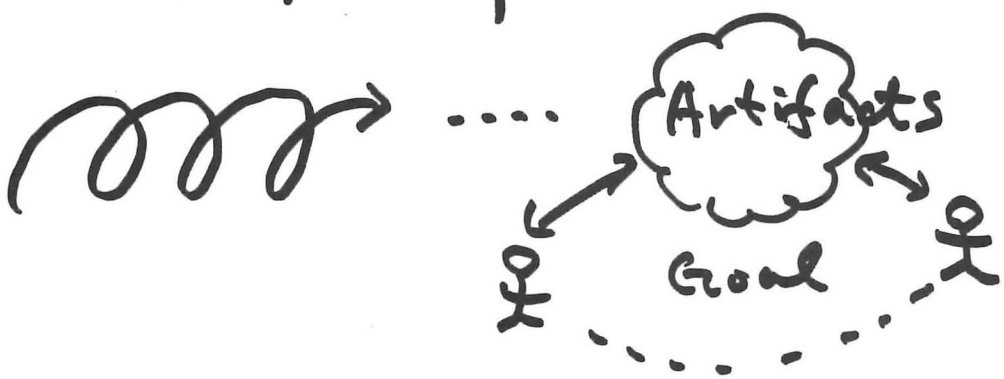
[6] Marc Weiser (伊藤 訳), 実世界は, 机上のようなちっぽけなものではない, *SEAMAIL*, Vol.9 No.2-3 (1994), pp.61–62

[5] 以上の文献は, 本文中に登場したものから, 編集部の判断で 関連すると思われるものを SEA の出版物を中心にポイントした。原典にあたられる方は, これらから孫引きして頂きたい。

Trail	Risk	Payoff	Lemmingineering	現時点における Trail の居場所
stuructured	0	medi	∞	assembler, C, fortran, BASIC, 仕様記述, etc...
OO-D,P,A	0	∞	∞	Smalltalk-80, C++, 設計方法, 物の考え方
process	0	0		!? (現場では未)
C	0	medi	0	ほとんど全員が使用
prototype	∞	∞	∞	PWB + ActiveBook (売れていない)
CASE	0	∞	∞	Hypermedia + methodology (未)
re-use	0	low	0	当然
comquats	0	low	0	–

表 1 . レミング度一覧 と 現時点における Trail の居場所 (熊谷)

Goal : Prototypeを作ることを
－分析，設計を通して－



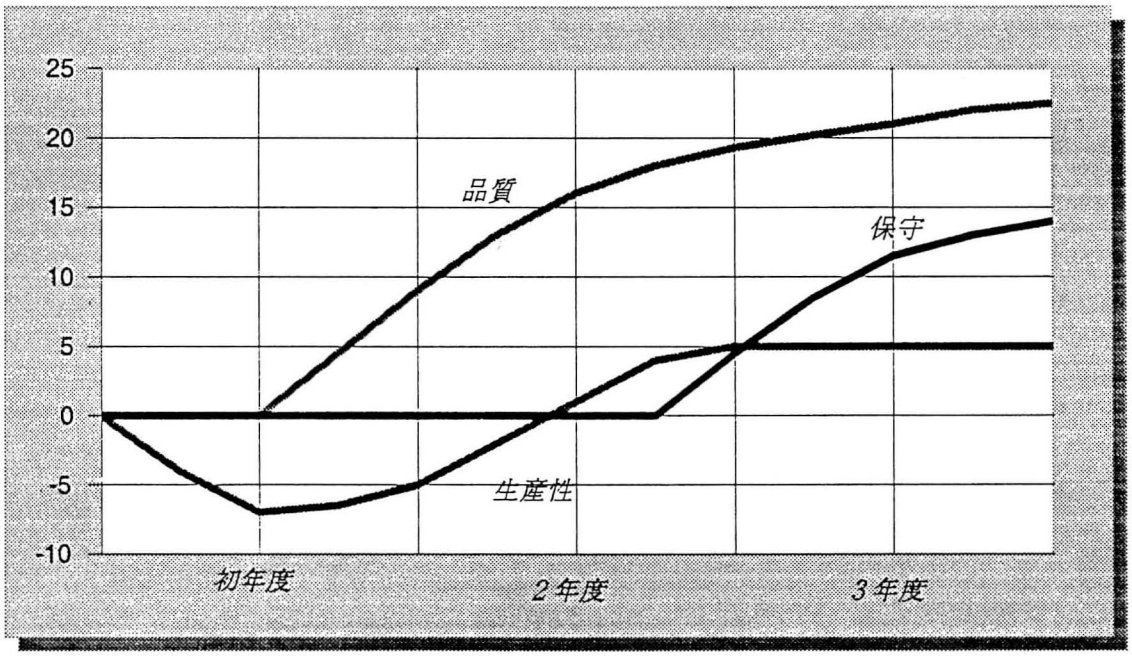
Artifacts : 表現形態は不問
－動的であれば尚よい－

Prototypeを介して様々な人々が communicationする．

Result : quick-and-dirtyでよい．

Prototypingは CASEの一種！

図 1．Prototyping に関して (熊谷)



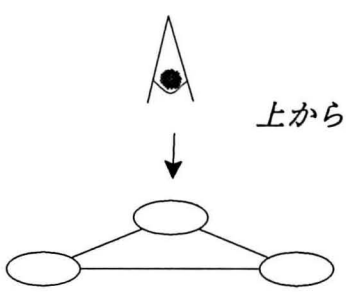
ＣＡＳＥ効果の波及

SS'94 @ Hakodate Jun / 14 / 94 (c) Masao Ito

表 2．CASE 効果の波及 (伊藤)

人間的活動の支援 －W型、E型

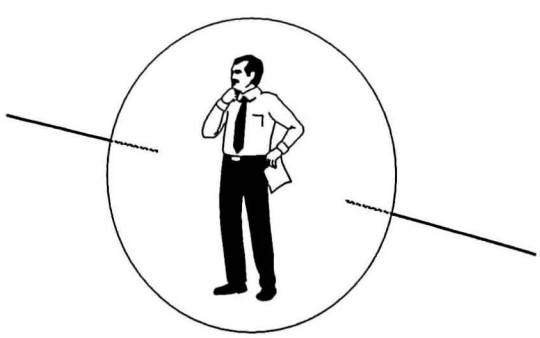
W型（西洋的）



上からの視点

・合成と分解
・推移的
・「説明が分る」

E型（東洋的）

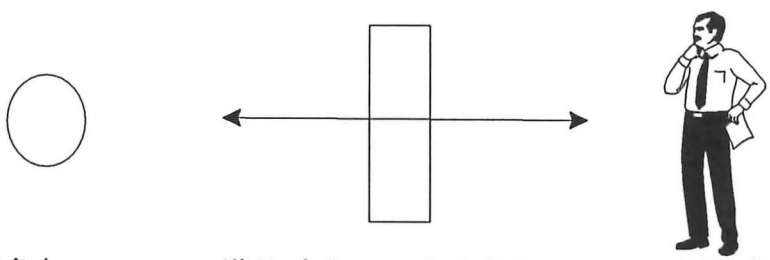


・経験、行為
・「あるがままに分る」

図 2．人間的活動の支援 － W 型, E 型 (伊藤)

「あるがままに分る」のために

- プログラマ中心主義
- 「見せ方」の確立－ソフトウェア工学の課題



・どう見せるか （何を隠蔽するか）－メトリックス、図式表示.....

➡見えないツールへ

図 3．「あるがままに分る」のために (伊藤)

結論 －ＣＡＳＥの目指すべき方向－

- 第一世代ＣＡＳＥ

特定方法論支援
特定環境（特定リポジトリ） ⇦ お絵描き
チェック

品質の<ある程度の>向上

- 第二世代ＣＡＳＥ

複数方法論支援
複数環境（標準リポジトリ）
メタＣＡＳＥからの導出 ⇦ プロセス中心
細粒度ツールによる構築

経験の<ある程度の>集積

- 第三世代ＣＡＳＥ (Nirvana)

"見えない" ツールと環境

図 4．結論 － CASE の目指すべき方向 －(伊藤)

Software Lemmingineering Summary

塩谷 和範 *

1. はじめに

そもそものはじまりは, 昨年 (93 年) 9 月号の IEEE Software 誌に "Software Lemmingineering" という面白そうなコラムを見つけた事だった。このコラムでは, Software 誌編集委員の Alan Davis と TRW 社の Winston Royce が, これまでにブームになった技術を取り上げて超辛口の批評[1]を行なっていた。

これに触発されて, 本年 (94 年) 6 月に函館で行われた第 14 回ソフトウェアシンポジウム (以下 SS94) のイブニングパネル [1] のテーマとして取り上げ, 活発な討論を行った[2]。この報告には, この時のミニアンケート分析結果の報告と, 来日した Alan Davis を囲んで行った "Software Lemmingineering Debate" の討論状況をまとめた。

2. Software Lemmingineering とは?

実は, アンケートそのものに十分な説明を付けなかったためと, SS94 論文集の "Software Lemmingineering" パネルの説明自体も Davis らがまとめた表と, 簡単な言葉の説明を載せただけであったので, 回答に際してはかなりの混乱があったようだ。そこで, いまさらながらの感もあるが, まず補足説明をする。

Lemmingineering "レミンジニアリング", Alan Davis の造語。
一般大衆 (The masses) が従っている技術に, それが適切であるかの考慮抜きで盲目的に従う工学システムのプロセス。

Lemming 短尾の柔毛足の齧歯 (げっし) 類の極地方の小動物。
ある欧州種のしばしば海に続いて飛び込み, 多数が溺死する周期的な集団移住で有名。 (Webster 第 9 版より)

2.1 Lemmingineering 要約

このコラムの副題を直訳すると,

「子供のように誰かが知らないことを知っていると羨む。
レミングの様に, 我々はリーダーに従う傾向にある。」

つまり, 自分が知らない新技術を誰かが知っていると我々は, それを羨むし, さらに, レミングの様に我々は, ともすれば (半ば) 盲目的にいまだ有効性が実証されていない新技術, あるいはそれを提唱するリーダに従う傾向にある。という危惧を, レミングという言葉とエンジニアリングという言葉を合成した造語によって提起したのが Davis のコラムの主旨だ。

Davis は, レミングの滅亡への軌跡 (trail) になぞらえ, 過去に我々がたどったブームとなった新技術の幾つかを, その効能 (達成目標) とそれに対する評価についてまとめている。なかなか辛辣な評価となっているが, 各項目 (軌跡) ごとに以下のような解説が付いていた。

2.2 構造化軌跡

70 年代, 構造化技術はソフトウェア産業の明らかに増大し続けるコストと, 増加を続ける顧客の不満足に対する一つの解決策として売られた。10 年間, 我々は隊列を組んで構造化ドラムのビートに向けて盲目的に行進

* (株)SRA 技術部システム技術グループ
[1] 正確に言えば, これら技術の受け入れ姿勢についての問題点の指摘
[2] SS'94 論文集 ならびに, 本誌 イブニングパネル報告参照

した。70 年代末には “構造化” は, “良い” ということと同義語になってしまっていた。我々は, “構造化” という形容詞の集中放火を浴びせられた。振り返ってみれば, 色々な時代で “構造化” と呼ばれてきた, 広範な異なるプログラミング慣行は, 我々の専門文化の中に溶け込んでしまった。今日, 我々は構造化プログラミングとは呼ばず, 単にプログラミングと言う様になった。

構造化設計/分析については, このようにはうまくいっていない。初期の構造化分析は, 過剰分析で我々を混乱させると共に, 複雑なリアルタイムシステム向きではなかった。後期の版では, Paul Ward, Derek Hatley, および Edward Yourdon の提案により, 過剰分析問題と, より簡単にリアルタイムシステムをモデル化するための制御機構を追加した。

しかし誰も, 我々はなぜ最初に構造化分析を行っているのかと言う問をしなかった。もしそうしていたら, 目的である要求仕様- システムをブラックボックスで記述する文書 - が階層的なデータフローあるいは制御フローでは書けないことに気づいたかもしれない。

構造化設計は, 構造化分析 (結果) を呼び出し階層に変換する単純なやり方の集合であり, ただ構造化分析と同程度に有効だった。

未だ, 構造化軌跡を集団暴走するレミングにはほとんど選択がない。もうどこにも行き先がない! 今や, 多数の生き残りたちは新たな群れを作り他の道を目指している。

2.3 オブジェクト軌跡

80 年代後半から 90 年代にかけて, オブジェクト指向技術は, コストと顧客満足度問題を解決する方法として売られた。今や “オブジェクト指向” は “良い” と同義語となり, 我々はこの形容詞の集中放火を浴びせられている。

オブジェクト指向プログラミングは, 20 から 30 年前に発見された, データ抽象化, 情報隠蔽, カプセル化, 継承などの品質を向上させるプログラミング技術の健全な原理に基づいている。これら実証された原理が広まるのは嬉しいことである。2000 年までには, オブジェクト指向プログラミング (OOP[3]) は, 単にプログラミングと呼ばれるようになるだろう。

オブジェクト指向設計 (とにかくその大多数の流派) は, 同様な確固たる基礎に基づいている。オブジェクト指向設計 (OOD[4]) は, 20 年前に Michael Jakson が説いた: “実世界の構造をまねることに失敗する設計は, 単に悪いばかりではなく間違っている!” の特徴を明示した。

それでは, OOP と OOD と記された軌跡の終端には何があるだろう? 保守性と信頼性の向上, および, おそらく開発コストの低減だろう。これらは確かにこの道にとどまるだけの十分な理由である。しかしながら, 劇的な生産性の向上と再利用における信じられないくらいの成功という標識は盲信するな! 本当かもしれないがそうでないかもしれない! 大多数のこの道を選んだ者は非現実的な期待を抱いている。

オブジェクト指向分析 OOA[5] については別の話だ。要求分析段階の目標 - 仕様 - を忘れずに, それが OOA の実施で可能かを自問してみよう。答は, “否” である。要求分析に適用できる OOA 技術はない。たとえ状態と振る舞いの記述でオブジェクトを改良しても, 構造化のときと同じ問題が残る: “全体システムのブラックボックスの振る舞いをオブジェクトの集合の個々の振る舞いによって調査できるか?”

OOA は, OOD への移行を容易にするという主張は馬鹿げている, いかなる工学原理も要求から設計への変換を容易にすることを主張していない。要求定義において, 保守性と信頼性の向上については, ほとんど意味が無い。(可能ではあるが) 証明されていないうたい文句の生産性の劇的向上, およびオブジェクト指向プログラミングと設計技術を使用することからの再利用は, 要求定義時においては正当化されることは少ない。

2.4 プロセス成熟化軌跡

今年 (93 年) の ICSE[6] では, 誰もが少なくとも片足をこの集団暴走に置いているようだった。目標は, 顧客とそのニーズを満足させる, 高品質 (信頼性, 保守性, 使用性...) ソフトウェアの, 予算とスケジュール内での一貫した製造である。このためには, 適切な背景 (知識), 訓練, 技術を持つ技術者と, 与えられたプロジェクトに合っ

[3] OOP: Object-Oriented Programming
[4] OOD: Object-Oriented Design
[5] OOA: Object-Oriented Analysis
[6] ICSE: International Conference on Software Engineering

た適切な開発プロセスを選択する能力と, その仕事を終わらせるための十分な資源 (期間, 予算, 道具) を持った管理者を必要とする。このようなモデルは, 繰り返し可能な, 測定可能な開発プロセスの重要性を強調する。これらは, 目標達成のための 2 つの重要な要素であるが, これ単独では十分ではないし, 必要でさえ無いかもしれない。適切な人々を得れば, 繰り返し可能な測定可能なプロセスなしに成功できる。適切な人々なしでは, 繰り返し可能な測定可能なプロセスであっても成功はおぼつかない。

現時点では, 適切なプロセスを各々のプロジェクト毎に選択することの方が, ある組織の全てのプロジェクトに有効な汎用プロセスを, (そのプロジェクトに) 合わせたものを期待することより重要である。

簡単に言えば, プロセス成熟化の道は, 長い高品質ソフトウェアに至る旅程の第一歩にしか過ぎないし, それ以上ではない。

2.5 C 言語軌跡

これは人気のある道のひとつである。そして, 我々を証明済の品質を浸透させるプログラミングから, ハッキングへと導く非常に危険な道である。C で高品質のプログラムを作成することが不可能とは言わないが, C を好むプログラマーは, ソフトウェアを早く作成することの方に興味がある。彼らは, (Ada プログラマと比較して) エラーを起こしがちな構造を避けるための時間を取ろうとしない傾向にあるし, ビットあるいはポインタレベルで遊ぶことを気にしない。C が人気があるのは, Unix の普及によるところが大きい。C によるコーディングは企業に, 発展するハードウェアに対しての最大の柔軟性を与えている。

C の道を離れる理由は今のところ無い。しかしながら, この道をとることについて自分を欺くな。(C が人気があるのは) 市場ニーズのためであり, 可搬性のためであり, 短期収入のためである (いずれも立派な理由だが)。それは, 高品質製品のためではないし, 長期の市場侵入 (確保) のためでも無いし, 長期の利益のためでも無い。

2.6 プロトタイピング軌跡

ソフトウェアプロトタイピングが人気が出始めたのは, 80 年代中期から後期にかけてであった。プロトタイプは, 必要なあるいは不必要な機能, あるいは, ユーザインタフェースの有効性のような事がらを発き出すのを助けるために, システムの初期バージョンをユーザに提供した。今日作成される多くのソフトウェアがユーザの問題の解決に失敗しているから, プロトタイピングは, 完全なシステム作成前にユーザ要求が明らかになっていることを保証する手助けをする。

プロトタイプは, 素早く作成されてこそ役立つ。それゆえ, 過度のテクニックとツールが現われ敏速なプロトタイプ作成を助けた。しかしながら, 90 年まで, "早いが汚い[7]" プロトタイプもソフトウェアコストとずれ込む納期の増大への対策として用いられるようになった。その論理は, "もしソフトウェアのプロトタイプを素早く作成できるなら, 単に, プロトタイプを製品と呼ぶことよって, 全てのソフトウェアを素早く出荷することができる。" だった。

考えてみてほしい。もし我々が高品質ソフトウェア製造の試みにも関わらず効率的でないとしたら, 試みもしない製品はいかに酷くなるだろうか? どのようにすれば適切なコストで高品質に製造するか知らなければ, 高品質かつ無視できるコストでいかにして製造できようか? 我々は, いかに品質をプロトタイプに付加するか明らかに分かっていない。

レミンジニアリング流儀の典型として, 重要なアイデアを取りだしながら, 応用を間違えてしまった。プロトタイピングは, ユーザ満足を保証するのを助ける重要な手法であり, それゆえコスト低減ができる。品質を保証する全ての技術を除去することによって開発コストを減らそうというのは, とてもまずいやり方である。

プロトタイピング軌跡上のレミングは, 突然崖に向かってそれて行ってしまった。

2.7 CASE 軌跡

元は, ほとんどの CASE ツールは, 初歩的な構文チェック能力を持つ図形エディタであった。CASE ツールは, ワードプロセッサが物書きを助けるのと同様に, ソフトウェア技術者を助ける。CASE ツールは, 劣っている技術者を良くはしないが, 各技術者を効率的にし, かれらの製品を見栄え良くする。

不幸なことに, その市場競争が激しいがために, CASE ツールは主たる価値ではなく, 幾多の他の機能: 自動

[7] quick-and-dirty

コード生成, 自動プロトタイプ生成, 自動テスト生成などで売られている。CASE ツールの評価時には, "只飯 (free lunch)" は無いことに気を付けよう。

この道のレミングは, 暴走している。しかし, 多くは売りすぎてしまった道の行方に何があるか気づいていない。

2.8 再利用軌跡

全ての工学分野では, 組み立てブロックの使用による製造を奨励している。これら分野では, このやり方は "利用 (use)" あるいは, "工作する (engineering)" といい, "再利用する (re-use)" とはけっして言わない。ただ, われわれの分野だけで "再利用" が, 流行り言葉 (buzzword) になっている。

明らかに, もし我々が新たなシステムを構築するときに, 組み立て式のソフトウェアコンポーネント (セグメントより大きな) が利用できるとしたら, 開発コストと工期は減少し, 製品品質は向上するだろう。我々は今や, 潜在的に再利用可能なコンポーネントのリポジトリの普及と, 最終的な再利用に向けての視点から新たなコンポーネントの構築に忙しい。そして, 驚くほど大量の努力がこれらの未熟な技術に費やされている。

我々は, 何が組み立て式か, 再利用可能なコンポーネントがどのようなものか本当に分かっているのだろうか? もちろんリポジトリ販売者は肯定するだろう。我々は, そのようなコンポーネントが何であるか, いかに保存し, 検索し, 組み立てるかを学ぶ前に, 明らかに沢山のコンポーネントとたくさんのリポジトリを経験しなければならないだろう。私 (Davis) は, 研究者及び実践者による広範囲な実験を支持する。しかしながら, 技術はいまだ非常に新しい。

他の多くの道と異なり, ここにおいての目標は極めて現実的であり, 最終的には達成されるであろう。しかし, 道はまだ舗装されていない。足下に注意, 短期的には自信を持ちすぎることなかれ。

2.9 COMQUATS[SIC] 軌跡

注文生産によるアプリケーションでのコスト削減と品質向上の努力は, ユーザ満足を得られない結果の繰り返しに終わっているため, もう一つの道として, COTS (Commercial off-the-shelf) つまり, 市販製品を使う道を提案[8]している。

既存の完全なシステムに, 市販のカスタマイズ可能なソフトウェアを追加することにより, 高品質かつ低コストのシステムを構築可能だとしている。この場合, 分野により, あるいは, どの程度カスタマイズが可能か? どの程度の品質か? にでき上がったシステムが依存することになる。むろんこれだけで完全なシステムができ上がるとは限らない。

したがって, COMQUATS とは, Commercial off-the-shelf quality software のことである。

(しかしながら, 後述する Davis との討論会で, 単なる冗談と言われてしまった。)

図 1 . 討論会時の Alan Davis 氏 (左から 田中, 青山, 新谷, Davis, 荒木, 鳥居, 塩谷, 渡邊 の各氏)

[8] ここで, "COTS は究極の再利用だ"(COTS is the ultimate in reuse.) と述べているので, これが有望な道と Davis が評価しているのかと筆者は考えていた。

3. ソフトウェアレミングの軌跡

Davis らの表にアンケート結果の一部を合わせたものを表.1に示す。

軌跡	活発な期間	達成目標	達成年	危険度	見返り度
構造化	1965-1986	開発コスト低減	1975	低 (低+)	低 (中)
		ドキュメント品質向上	1975	低 (低+)	低 (中−)
		顧客満足度向上	1975	低 (低+)	低 (中−)
オブジェクト	1980-2000	開発コスト低減	1990	低 (中−)	中 (中−)
(OOD & OOP)		信頼性向上	1990	低 (中−)	中 (中)
		保守性向上	1990	低 (中−)	中〜高 (中+)
(OOA)		開発コスト低減	1993	高 (中)	低 (中−)
		顧客満足度向上	1993	高 (中)	低 (中)
プロセス成熟度	1990-2005	プロセス向上	1993	低 (中−)	中 (中)
		顧客満足度向上	1993	高 (中)	低 (中−)
		品質向上	1993	高 (中)	高 (中+)
		開発コスト低減	1993	高 (中+)	中 (中−)
C	1980-2000+	開発コスト低減	1980	中 (中)	中 (低+)
		移植性向上	1985	低 (中−)	高 (中)
		品質向上	無理	高 (中+)	無し (低+)
		保守性向上	無理	高 (中)	無し (低+)
プロトタイピング	1985-2000+	顧客満足度向上	1993	低 (中−)	高 (中+)
		開発コスト低減[9]	無理	高 (中+)	無し (中)
CASE	1988-2000+	ほどほどの生産性向上	1993	低 (中−)	中 (中)
		劇的生産性向上	2013+	高 (中+)	高 (中−)
		ドキュメント品質向上	1993	低 (中−)	中 (中)
		品質向上	1993	低 (中−)	低 (中)
再利用	1988-∞	開発コスト低減	1996-2003	高 (中+)	応用依存 (中+)
		品質向上	1996-2003	高 (中)	応用依存 (中)
Comquats[10]	1988-∞	開発コスト低減	1996-2003	高 (中)	応用依存 (中+)
		品質向上	1996-2003	高 (中)	応用依存 (中)

表 1．ソフトウェアレミングの軌跡

ここで, Davis の評価と併記して, 後述するアンケート結果を, () 内に,

低 (0〜8%)
低+ (9〜24%)
中− (25〜41%)
中 (42〜58%)
中+ (59〜75%)
高− (76〜92%)
高 (93〜100%)

に分けて示している。

[9] 使い捨て版の配布によって
[10] Commercial off-the-shelf quality software - Davis の造語
COTS: Commercial off-the-shelf

4. ソフトウェアレミングアンケート

SS94 での“ソフトウェアレミンジニアリング・パネル”に先立ち, 事前に同一項目についてのシンポジウム参加者の評価を記入していただいた。この結果を表.2に示す。

軌跡	達成目標	危険度 (%)	見返り度 (%)	レミング度 (%)
構造化	開発コスト低減	6/31(10)	27/32(42)	28/29(48)
	ドキュメント品質向上	7/31(11)	25/32(39)	26/29(44)
	顧客満足度向上	9/32(14)	19/29(32)	20/26(38)
オブジェクト	開発コスト低減	25/31(40)	25/32(39)	26/29(44)
(OOD & OOP)	信頼性向上	19/31(30)	32/32(50)	26/28(46)
	保守性向上	24/31(38)	39/32(60)	37/29(63)
(OOA)	開発コスト低減	33/29(56)	21/30(35)	20/26(38)
	顧客満足度向上	30/29(51)	26/30(43)	22/26(42)
プロセス成熟度	プロセス向上	17/28(30)	28/28(50)	20/27(37)
	顧客満足度向上	25/27(46)	16/27(29)	14/26(26)
	品質向上	29/28(51)	36/28(64)	25/27(46)
	開発コスト低減	36/28(64)	21/28(37)	12/27(22)
C	開発コスト低減	30/29(51)	13/30(21)	13/27(24)
	移植性向上	23/30(38)	27/31(43)	20/27(37)
	品質向上	35/29(60)	8/29(13)	10/25(20)
	保守性向上	34/29(58)	6/29(10)	8/25(16)
プロトタイピング	顧客満足度向上	23/31(37)	44/32(68)	33/28(58)
	開発コスト低減[11]	38/31(61)	26/30(43)	25/27(46)
CASE	ほどほどの生産性向上	21/30(35)	28/30(46)	22/26(42)
	劇的生産性向上	36/28(64)	20/29(34)	13/25(26)
	ドキュメント品質向上	22/30(36)	30/30(50)	26/26(50)
	品質向上	23/30(38)	26/30(43)	23/26(44)
再利用	開発コスト低減	37/30(61)	43/31(69)	29/28(51)
	品質向上	34/30(56)	36/31(58)	28/28(50)
Comquats	開発コスト低減	29/27(53)	31/25(62)	15/21(35)
	品質向上	27/27(50)	29/26(55)	14/21(33)
Formal Approach		4/2(100)	1/2 (25)	1/2 (25)
Internet	情報入手容易化	0/1 (0)	2/1(100)	1/1 (50)

表 2 . ソフトウェアレミングアンケート集計結果

アンケート方法は, Davis の表の評価欄 (危険度, 見返り度) を空欄にし, さらに回答者自身のレミング度 (自分自身がどの程度その技術に入れ込んだか?) も併せて記入していただいた。加えて, 目標達成年度のうち“無理”あるいは, 未来日付になっているものについても空欄とした。最後に, 空欄の 1 項目を設け, つけ加えたい技術についても記入していただいた。

[10] 使い捨て版の配布によって

4.1 アンケート評価方法

評価方法は, Davis にならって, 高中低の 3 段階評価とし, それぞれ 2,1,0 を割り当てて集計し, 平均値 (%) を出した。表中の分数の分母が有効回答数, 分子が評価合計値, () 内が平均値 (%) を示している。

このデータを縦軸に軌跡 (技術項目) をとり, 横軸に危険度, 見返り度, レミング度をとり, 折れ線グラフ (図.2) で示した。軌跡毎にグラフは切れている。

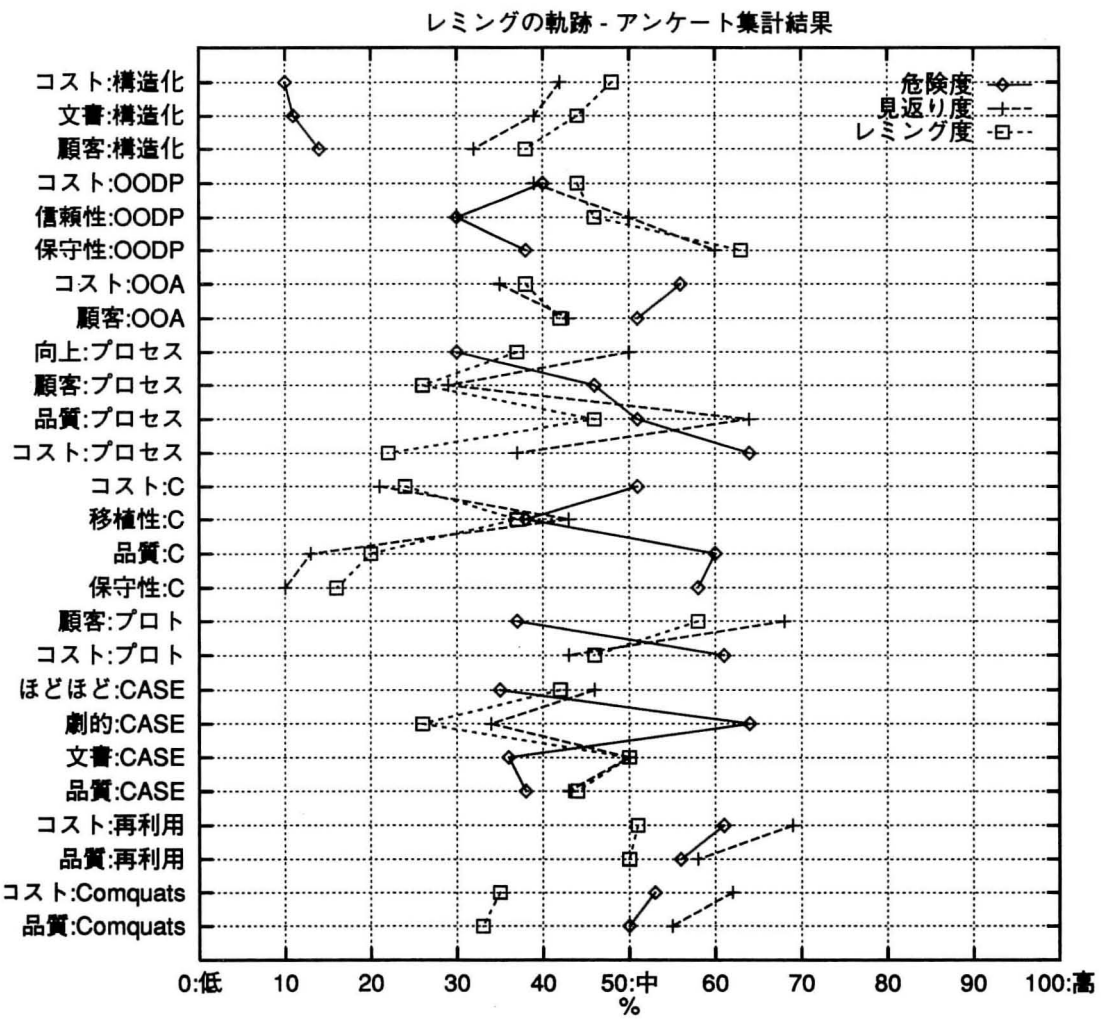


図 2 . レミングの軌跡 - アンケート集計結果

5. アンケート結果について

まず, 表.1で, Davis の評価とアンケート結果を比べると, およそ 30 人からの回答を集計したため平均化され, ほぼ "中" の評価となっている。しかしながら, おおむねその傾向は, 中に集中しているとはいえ, Davis の評価に近いようだ[11]。

構造化については, Davis の指摘どおり既に実践しているため, リスクは非常に低く見返りも "中" 程度と好意的に評価している。

OOD & OOP については, 危険度は "中" で高め, 見返り度については "中" で, 意外に冷めた目で見ている様だ。OOA については, むしろ好意的で, 危険度見返り度とも "中" と評価している。

プロセスについては, "中" に偏っているが, ほぼ Davis の評価と同様だ。

C については, Davis 同様に厳しい見方となっている。元の評価では, "見返りなし" となっていたところをあえて評価して頂いたが, "低+" という結果になっている。C による製品の品質向上と保守性向上に問題意識を抱いていることの現われだろうか? それにしても, コストと移植性についてはもう少し高い評価があっても良いのではないだろうか。

プロトタイピングは, 開発コスト低減については, "使い捨て版を配布することによって" という厳しい条件がついていて, さらに "見返りなし" となっているが, "中" 程度の見返りがあると回答している。これは, Davis よりも積極的にプロトタイピングの利用を考えての積極的な回答の様に思われる。

CASE については, SS94 のパネル時にもパネラーの伊藤氏からも指摘があったが, 劇的な生産性向上に対して, 危険度を高めに評価するのは理解できるが, 劇的な生産性向上つまり見返りが大きいことが保証されていると考えられるのに, 見返り度は, 危険度に比べて低めになっている。見返りを期待したいが, 現実はそううまくは行かないという, 過去の手痛い経験の反映なのか? あるいは, CASE ベンダーが主張する過剰な宣伝の裏を冷静に読んでいることの証明なのだろうか? この点以外は, "ほどほどの危険度でほどほどの見返りが得られる" と, 冷静に得失を考えている様に見える。

再利用と, Comauats については, 後述。

最後にアンケート結果のグラフ (図.2) で, 筆者が突然思い付いてつけ加えた "レミング度" に着目すると, 見返り度とほぼ比例する結果が出ている。見返りが多いと思うからこそ熱中するというのは, まさにそのとおりではないかと思う。ただ, ここに上げた全ての技術を実践できる人, あるいは実践した人は, 少ないと思われる。よって, レミング度 (見返り度も同様か?) には, かなりの期待値が入っており, 特に比較的新しい技術については期待度の方が大きいように思える。その例が, 再利用と Comquats というわけの分からない新技術に対しての, "リスクも見返り度も高いが, レミング度は低い" になって現われていると考える。

なお, 表.2の最後の 2 項目は, 回答者がつけ加えた項目である。Formal approach が 2 名, 話題のインターネットが 1 名だった。もう少し, 色々な追加があるかと期待していたが, 件数が少なかったためグラフには加えていない。

6. Software Lemmingineering Debate

SS94 が終わり, ほっとするのもつかの間, 6 月 23 日夕刻, 川崎の富士通クロスカルチャーセンターに, メールで希望者を募った結果 総勢 15 名が集まり, Alan Davis を含む IEEE Software 誌 編集委員会委員 5 名の参加を得てソフトウェアレミンジニアリング討論会を開催した。この時のもようを以下に簡単にまとめる。

6.1 Eagle-eye perspective

SS94 のパネルでも話題になっていたことだが, 何もレミングの様に皆が崖から落ちてしまわなくても, 気がついた人が方向転換して知らせれば良いではないか? という話が出ていたが, ここでの Davis の回答は, "ヘリコプターからの視点つまり鳥観 (俯瞰) による展望 (Eagle-eye perspective)" を用いることを勧めている [5]。この展望は, 実際には各軌跡を辿っている人々に聞いたり, 他の軌跡と比べたり, 実証された技術であるのかなどを調査したりすることによってのみ得られる。同様に, 軌跡を辿り始めたら先行者の動きをよく見たり, 斥候を出したり (プロトタイピングのことか?) することによって, 危険を察知できる。

[11] 注: 未記入項目, 0 あるいは無限大など, 指示したもの以外の回答は集計からはずしている。そのため項目毎に母数が違っている。表.2

6.2 ハメルーンの笛吹き

この討論会に参加された東大の玉井先生が, ハメルーンの笛吹き[12]の様に, 誰かが笛を吹いて新技術を知らせることも必要だ。との指摘をしたが, これについても「新技術の存在を知らせることは重要だ。しかし, ハメルーンの笛吹きに踊らされて, 水に飛び込まされたのは, レミングではなくて どぶねずみ (rat) だ。どぶねずみは理由を知らずについてくる。レミングは概念についてくるところが違う。」と言っていた。(このレミングは, 人を指すのか? あるいは, 死ななければならぬ宿命を悟っていることを指すのだろうか?)

6.3 5~10%の生産性増加で十分

CASE ベンダーなどが, 大きな生産性向上を宣伝しているが, 誇大宣伝だ。実際には, 5~10%の生産性増加で十分だと考えるべきだ。

6.4 AI レミングは未だ崖に達していない?

AI が表に載っていないことを指摘すると, 「少数の年寄り AI レミングはずっと以前からゆっくりゆっくり歩み続けているが, 未だ崖には達していない。まだまだ当分かかるだろう。」

6.5 よく見て飛べ (Look before YOU Leap!)

Davis の 7 つの助言 [3]

- **現実的な目標設定。** - 成功についても現実的であれ。
- **宣伝を盲信するな。**
- **納得できる技術を選べ。**
- **危険と確度, および効果を評価せよ。**
- **注意せよ。しかし (余裕がないからといって) 無視するな。**
- **当初の目的を忘れるな。** - 初心忘るるべからず。手段が目的になってしまってはまずい。
- **"皆で渡れば怖く無い" は止めよ!** - 数に頼るな。

この助言は, Bob Glass による反論記事 [4] に対しての反証として反論記事内のコラムとして掲載されている。

6.6 The Path Less Traveled

また, ソフトウェアレミンジニアリングコラムの反響が大きかった様で, 更に, 別の号でも低コストかつ低リスクの 6 手法 [5] も提案している。

- **顧客との対話**
- **プロトタイプによる事前検証**
  ここではけなすことなしに, 本来の意味でのプロトタイピングの活用を奨励している。
- **Trade-off analysis の実施**
- **Formal method の部分的使用**
  部分的にでも, 仕様の一部を形式記述することにより, 矛盾点・不整合などが発見できるので, 取り入れることを勧めている。
- **レビュー, ウォークスルー, コード検証の実施**
- **技術の掘り出し (聞いて回れ)**

Davis も, アンケートの回答にあった様に, 形式記述の利用を認めている。また, 辛辣な批評では悪く書いていたプロトタイピングについても, 正当に評価していることが分かる。

[12] "The Pied Piper of Hamelin" by Robert Browning

7. あとがき

もう大分たってしまい, テープ起こしをする気力も無いので, 当日のメモから拾ってみたが, 案の定 大したことは書けない。そこで, 記憶と Davis のコラムを参考にまとめてみた。Alan Davis 氏は快活な大男で, 多少 下品な言葉を使う傾向にあるが, 人の良い気さくな人柄で, 実務経験もある様で博識であり, ていねいに解説していただいた。IEEE Software 誌の次期編集長に決まったという話を後で知った。

先にも書いたが, Comquats (これも Davis の造語) については, コラム中で "COTS は究極の再利用だ。(COTS is the ultimate in reuse.) " と書いているが, この件について聞くと,

「そんなことを書いたはずは無い。」と否定するので,

「"ultimate in reuse" と書いているだろう?」と問うと,

「英語の "ultimate" には, 二つの意味があって, 否定的には, 大ぶろしき (too big/far) の意味だ。」

と笑われてしまい, 自分の理解の足りなさが露呈し少なからずショックだった。この項目については, 現編集長にも載せるべきではないと言われたそうで, 少し安心はしたが。。。。

7.1 アンケートについての反省

常々思っていることではあるが, アンケートというのは難しい。今回の様にできるだけ簡単に読んで回答できる様にと, 記入方法を簡略化しても, 項目自体についての説明が十分でないと, "定義が不明なので回答不能" となってしまう。今回に限っていえば, SS94 論文集のレミンジニアリングパネルの説明に各項目について十分な説明をしておくべきだった。

とはいえ, Davis のコラムにも説明が無いものについては, (自明な疑問については, あらかじめ解釈法を明記できるが,) それを参照する者が各自推定することによって補うしかない。今回も大多数の回答者が自主的に判断して回答していた。

もともと, 全員が回答したとして 100 名ほどであるので, 統計データとしての信頼度は低い。しかしながら, それなりにグラフを見ると興味深い結果が得られる様に思う。次回は, 経験の知識の有無, 現在の仕事分野などを加えると, また別の傾向が見えてくるのではないかと考えている。

7.2 謝辞

解説が不十分なアンケートに関わらず回答していただいた, 多数の回答者の皆さんのご協力に感謝する。またパネラーとして貴重なご経験とご意見を発表していただいた, Gerhard Fischer(コロラド大), 伊藤氏 (MASC), 熊谷氏 (PFU), 長崎氏 (ビジョン・コーポレーション), および Davis 氏との討論会参加者 (氏名は省略させていただいた) 諸氏のご協力に感謝する。

最後に, 快く翻訳引用することを許可していただいた Alan Davis 氏に感謝する。

1994.10.18 塩谷

参考文献

[1] 塩谷 和範, Software Lemmingineering !?, ソフトウェアシンポジウム *'94* 論文集 Jun 1994, pp.173-181.

[2] Alan Davis, Software Lemmingineering, *IEEE Software* Sep 1993, pp.79-84.

[3] Alan Davis, Counterpoint: There are no BAD guys, and No one has the ''Truth'', *IEEE Software* May 1994, pp.92.

[4] Bob Glass, From Wonderland to the Real Problem, *IEEE Software* May 1994, pp.90-92.

[5] Alan Davis, Rewards of Taking The Path Less Traveled, *IEEE Software* July 1994, pp.100-103.

これからのソフトウェア・プロセス・マネージメント*

パネリスト： 久保 宏志† 坂本 啓司‡ 桑名 栄二§ 兼子 毅¶ 司会： 大場 充‖

1. プロセスの位置づけ

司会 (大場) 最近, ISO9000 とか, CMM[1], SPICE[2], EXPERIENCE FACTORY ですとか, うんぬんかんぬんと沢山 色々なキーワードを耳にいたします。これは特に, 昨年とか一昨年ぐらいから, この傾向が強くなって来ている訳ですね。こういう物が話題になっている背景には, その後ろにある共通したコンセプト/キーワードとして「プロセス」が非常に重要なような感じがします。で, 今日は, その「プロセス」に焦点を当てて議論したいと思います。

で,「プロセス」は 今回の SS94 でも, 議論がありましたけれども, どのレベル (あるいは立場) で捉えるかによって非常に変ってくると思うんです。まあ, 極端な抽象化というか単純化をすれば, 要求が入ってきて, 最終的にプログラムとかドキュメントとかを含めてソフトウェアというものを出すわけです。で, それを生成する工程をプロセスと非常に大きく見ることも出来るし, それをさらに プロセスのプロセスとして細分化していったところを見ることも出来ると。で, これに影響を与えるものとして, 例えば "要求" も勿論ですけれども, "技術者", それから "利用者" というものもあります。あと勿論 "管理者" もありますね。で, "管理者" の側面から見ると, プロセスを改善したりするということは, 非常に興味がある。一方, ソフトウェア自身, ユーザから見る, その品質, コスト, デリバリなどということも問題になります。"技術者" から見れば, 当然, それを実行するのか? ということが問題になりますね。例えば, レビューをする。デザインをする。どうやってデザインをするか?, と。そういうことですね。で, ここでは, 一応「これからのソフトウェアマネージメント」ということで, ソフトウェアプロセスの管理[3]という側面。何の為にソフトウェアプロセスというのは必要で, 何の為に管理しなきゃいけないのか?, というようなことを, 色々な側面から考えてみたいと思っております。

で, まぁ 月並なところでは, ユーザの為, より良いソフトを作るというかですね, そういう感じもあると思います。それから, 技術者のため, より簡単に仕事をしたい, 経済ですけれども 楽に仕事をしたいとか, 管理者だったら, より高い効率というのは, 少いリソースでやりたいとかですね。まぁ「科学, 人類の為」というふうに大きく言いましたけれども, ソフトの開発工程って, いったい どうなっているんだろう, ということをもう少し深く知りたい。「何故, 失敗するのか?」だとか「何故, 失敗したか」ですとかですね, そういうことを説明したい。というような様々な観点や考え方があると思います。

そこで, こういうパネルを形成するにあたって, ここにおられる非常に御高名な方々にお願いしたところ, 御快諾を頂きましてパネルを形成することになりました。簡単にパネリストの面々を発表順に紹介したいと思います。

最初は, 富士通の久保さんです。え〜「仙人」と書きましたけれども, そういう超越した立場での発言を期待しております。今回の SS でも, 初日から色々と積極的に発言されておられますので, これは十分に期待に沿って頂けるだろうと考えております。

で, 2 番目。オムロンの坂本さんです。「メンター」[4]と書きましたが, 皆さんを導いてくれる, 非常にバランスのとれた方ですので, 理性的な発言があるのではないかと。特に管理面, プロセスと管理ということに関しましては, 現場で ずっと指導されておりますので, 色々な経験を通して得られた知見を披露して頂けると期待しております。

次に NTT ソフトウェア研究所の桑名さんです。桑名さんは, 現役のバリバリでやっておりますので,「親方的」と書きましたけれども "最先端で働いている人として, プロセスに対して どういう反感ですとか, どういう意識を持っているか?" ということをお聞きしたい, と。

最後に, 東大の兼子さんです。「リサーチャ」として

---

* 1994 年 6 月 17 日 9:30 〜 12:00

† 富士通 (株) システム本部
‡ オムロン (株) EFTS 開発センター 第二開発室
§ 日本電信電話 (株) ソフトウェア研究所
¶ 東京大学 工学部 化学システム工学科
‖ 広島市立大学 情報科学部

[1] Capability Maturity Model

[2] Software Process Improvement and Capability dEtermination

[3] ここで '管理' という言葉は, 非常にメタなものだと解釈して下さい。スケジュールを '管理する' とか, そういう意味ではありません。

[4] mentor : 良き指導者, 助言者

ですね,「書生的」と書きましたけれども, 良い意味ですから。そういう観点でですねお話をお願いしたいと思います。

で, 基本的な精神はですね, 50 代を代表して 久保さん。40 代の技術者を代表して 坂本さん。30 代の現役を代表して 桑名さん。で, ちょっと 'とう' が立っていますけれども, 20 代の意見を代表して 兼子さんというふうにアサインしました。それでは, 久保さんから, パネリストの面々に 最初にお話を頂いて, そのあとフロアの方に展開したいと思います。

2. SGI のパニックを例に

久保 50 代を代表する, というのはちょっと無茶で, かなり外れていると思うので, 典型的ではないと思いますけれども, よろしくお願いします。で, 私が貰った問題は「プロセスの標準化改善という課題をどう考えて, どういう態度で立ち向かうべきか?」です。そのときに "技術者として" というのがあるんですね。それを考えるために, 皆さんのお手元に資料[5]をお配りしました。これは シリコングラフィックス社 (Silicon Graphics Inc., 以下 SGI) で同社のワークステーション Indy 用の OS IRIX の開発で発生した問題点を記したものです。著者は Tom Davis という SGI の中の開発者で, 社内で偉くはないようですが, 会社を創立した Co-founder の 1 人のようです。

昨年 (1993 年)10 月の段階での SGI の中に場面を移して, その中で与えられた課題を考えてみようと思います。その時 IRIX 4.0.5 というのが市場にあるが, それだと競争力が無いので 5.1 というのを開発しているのですが, 結果的にこれは出荷できる品質に到達出来ず, 5 月に 出荷停止[6]という判断をしています。で, 12 月に, それのリカバリの為に 5.1.2 というのを出荷しようとしている。出荷するためには freez という段階が入るんですけれども, その間近になって このレポートは書かれている。で, 要するに「そのまま出荷するのはヤバイよ!」という話です。93 年の初めの段階で Tom Davis 達は 5.1 の出荷は殆ど不可能だと, そのまま行ったら不可能だと。だから, 機能の削減をやったらどうですか? という提案をしているんですけれども, それは会社の意志にはならなくて, 外注の導入ということが行なわれ, まぁ, 性能が悪いことは分っていたので, その性能改良の為に人をつぎ込むというディシジョンがなされた。で, お配りしたレポートの第 1 版が, その 5.1 の出荷直前になって書かれたわけです。その時は, 16MB で動くようなものにしないと競争力が無い ということでやってきたが 実際の製品ではとてもそんなメモリ容量では動作不可能。で, かつ memory leak[7] その他にも コマンドのレスポンスタイムが 4.0.5 から比べると 40%位増えている, というふうに色々な問題が見つかってきた。で, そういう状態のまま 出荷予定を迎えた。で 最終的にここで一般出荷は取り止めて, 限定出荷 だけ行なった。その段階でも, バグは沢山あって, それらにプライオリティを付けをみると, 修正しなければならない大事なバグがおよそ 500 個くらい残ったようです。けれども 「それは例外である。バグは 0(ゼロ) だ」と偽って出荷しちゃっているんですね。これが 5 月の出荷の話。

そして, このレポートが書かれた 10 月に至っても 殆ど事態は改善されていない。やっぱり, その年の初めに提案したと同じように混乱が続いている。で, Tom 達は 「5.1.2 の機能範囲を絞り込みましょう。そういうことでもやらないと残っているバグのフィックスは不可能だ。絞り込んで, あとはどこか的を絞ってコードウォークスルーとデザインレビューをやりましょう。そうやってなんとか 5.1.2 の出荷を実現しましょう」と提案している。また, それからその後, 長期的にどういうことをしましょうか, ということもこう書いてあります。が, それで 12 月にどうなったのか? というのは私はまだ知らない。現在, それから 8 ヶ月ないし 9 ヶ月経っているんですけれども, 今 そのプロジェクトがどうなり SGI がとうなったかは知らない。

以上のような場面の中で, 与えられた課題を考えてみたいと思います。まぁ, 色々にエクササイズを使うことは出来ると思うんですけれども, まずこういう状況の中で, そこに働いている人達の意識や行動がどうだったのか? というのを, ちょっと振り返ってみましょう。

全体としては, もう どう考えたってうまくいってなくてパニック状態なんですけれども, 何が悪いのか誰も解っていない。要するに, 事実が無しで, 色々な犯人捜しですね。「あいつが悪いんだ」とかいう憶測だとかが飛び交うわけですけれども, 事実が解らない。で, 多分, 相当に長時間労働も強いられただろうと思うんです。まぁ, 少々高い技術目標というなら 技術屋というのは頑張るでしょうけれども, あまりにも無

[5] comp.os.research からの記事を持参された。興味深い記事だが それをそのまま予稿集や SEAMAIL に転載するのは適当ではないのでしない。興味ある読者は, 久保さん (hkubo@fhl.fujitsu.co.jp) に直接 連絡をとられたい。

[6] 一般出荷は止めて, やむを得ない所にだけ出荷をする

[7] 動かしていると どんどん常駐が増えていって 1 週間もしていると 16MB を越えてしますといったカーネルのバグ。だから毎日 re-boot しないといけない!

茶苦茶やらされると やる気を無くすでしょう。そういう状態も発生したであろうと想像できます。その一方で, ハッピーになる人がいるわけです。要するに「俺に任せれば, ちゃんとやってやる」と言って偉くなっていって, だけど その通りにならなくたって「何故ならなかった」ということが錯綜して原因が解らないので, 偉くなったままという …。まあ, そういうことも起こる。

また, 不具合が多々あるのは解っていますから, 色んな人が 色んなことを言う。だけれど「こっちに行きましょう」という意志は固まらない。で, 例に挙げたのが性能ですが, 非常に悪い訳ですから, なんとかしないといかんわけです。それなのに, そんな時に「まず性能分析しないといけない。その為にはツールが要る」などという悠長な議論が始まってみたり, 「read-only の部分だけを どこかにまとめてキャッシングする」だとか, そういう暫定対処的な大技を主張するようなのが出てきたりして, なかなか意見が定まらない。それからもっと悲劇的なことは, 売る方と作る方とで 全然意見が一致しないまま, 時間だけが経過している。売る方は 16MB でないと売れないと言っている。作る方は 16MB でなんか収まらないと言い続けている。

このように, 色々な教材になりうると思いますが, ここではプロセスの'標準化' と '改善' に絞って考えてみます。私の持っている情報は皆さんの御手元にあるレポートだけで, 本当はもっともっと, 事実調査 (fact finding) をやる必要があって, それから色々な分析をしなければあまり核心的な確信を持てるような案は出ないのですけれども, 一応仮説を立てて見ることは出来ると思います。

2.1 MUST(必須) 項目

まず, 予想のプロセス が必須と言えるでしょう。要するに「今 こうだ。で, 且つ 予定していることをやっていくと, こうにしかならない」っと。そして, それをやる人は, プロジェクトメンバでもなく 営業でもなくて, 謂わば マネージメントのスタッフ的なところに持つということが MUST に近い。こういうパニック状態になりますと, 2 人の人だと, とにかく話がつかない。で勿論, もう 1 つの MUST は, そのマネージメントプロセスの中で, その予想のプロセスからの情報を尊重してプロセスの改善を計る。それは, 計画の見直し, 削減だとかという判断も助けてくれますし, あるいは予想の精度の為に正しい情報を提供するという という効果を持ちます。それからマーケティングの人達は, 戦略ですね。競合戦略, あるいは 戦術。まぁ そういった物を見直し, それから freezing が来たから freezing を始めるという事態になっているんですけれども, それの時期ですね。そういったものも合理的に行なえるでしょう。また, 先程の性能問題の時のように, 何かしらの対処をしなければいけないんだろうけれど, そのために色んな意見が出る。で, そういった色々な選択枝の中で, まあ一番良かろうと思うものを選択するというそういう合理性ですね。マネージメントの判断に合理性をもたらすというそういうことが効果として期待できる。そういうことが MUST であろうと思います。

2.2 SHOULD 項目

次に, SHOULD 項目ですね。プロジェクト管理ツールと見積りツール。これなんかも, 導入しているのかいないか分りませんけれども, もしも導入していないのならした方が良い。別にその, 導入したから仕事が早く出来るわけでもなんでも無いですけれども, 色々な精度が高まるということですね。それから, シミュレーションが出来るようになるという。「こういう前提だとこうなる」というシミュレーションが出来るようになる。それから, 見積りツール。これも, ツールの中に, それなりに経験値が入っているわけで, さらに足らないものは これからどんどんデータを蓄積してゆくというベースとしても入れた方が良いだろうと …。まぁ, こういう事が, 「長期的にはどうすれば良いのか?」への仮説ですね。

2.3 ソフトウェア技術者として

ソフトウェア技術者として どう考え, どう立ち向かうべきか? ということですが。1 つは, SPICE ですね。まあ, ISO があり, CMM があって それらをガチャっとくっつける感じで SPICE の活動が行なわれている。それに対して, 貢献をしてゆくというんですかね, ということが 1 つあるな, と。

それから「ちょっとくらいついてみたいなぁ」と思っているのは, 昨年の 10 月から今日まで, 一体 何が起こっているのか?という SGI の動きをフォローしてみたいなぁと。そのためには, 元のレポートが投稿された `comp.risks` のアーカイブを探せば, その周辺のスレッドが見つかる可能性がある。それを見れば, 10 月の段階で NetNews で, どういう議論が行なわれたかというのが分る。それから勿論, 現在の段階までを知れば, 現在どこにあって, どういう方向に向かっているのかが分るのではと思っています。勿論, 今の私の仮説と, 実際に彼等に起こっていることを対比してみるのもこれはまた面白いでしょう。で, レポートの

著者である Tom Davis と意見の交換をするという，そういうふうにしたいなぁと思っています。まだ残念ながら彼のメイルアドレスは判っていないんですが，comp.risks に記事を投稿した Jerry Leichter さんとはメイルを交換しています。彼からの情報では，幸にして Tom Davis さんは クビになっていないんですね。僕は，クビになっているのではなかろうかと非常に心配していたんですけれども，幸にしてクビにはならないで，まだしつこくこれを追究しているようです。まあ，ここまでが「どうあるべきか」。私が考えてみたことですね。

で，「どう立ち向かうべきか?」というのが，皆さんに対するメッセージになりますかね?。まず「私と一緒にやりましょう」というのが一番ですね。それと，その Tom Davis さんのレポートというのは，色々な点で学ぶべきことが沢山あるだろうと思うんです。で，彼が何故 ああいうものを流したか解りませんけれども，もし “Help Me !” あるいは “Call for Comment” として流したとすれば，ちょっとコンテキスト情報は足りないと思います。そういう「助けてくれ〜」といったときに，どういうコンテキストをくっつけるべきかというようなことをコメント/アドバイスしてあげることも，僕 多分出来ると思うんですよ。そういうことをちょっとやってみようと考えています。

それと，一番学ばなければいけないのは，こういう事例を，社会の共通の経験として蓄積していくっていうんですかね。まあ，そういうことは，なかなか決意しても出来ることじゃなくて，日本だと確実にクビになるだろうと思うんですけれども，やっぱり若干の固有名詞を伏せても 失敗体験とそのリカバリの体験ですね，これが大事なんじゃないかと。

それから，何の為のプロセスかという投げかけが 大場さんからありましたけれど，こういった経験知を科学にしてゆくということも大事だろうと。

それから，我々は一応 技術者ということで SEA に入っているんですけれど，プロセスは，エンジニアのプロセスとマネジャーのプロセスがある。で，どちらが悪いかというと，多分 マネジャーのプロセスの方が悪いと思うんですね。ですから，やはり 経営者とか管理者によるマネージメントプロセスの質を良くすることが，結果的に技術者がハッピーになるための道であると思います。こういう態度に，すぐ管理者というと嫌う人が大勢いらっしゃるんですけれど，嫌わないでですね〜，管理者を助けてゆくと。そうするのが，正しい態度である。ということで，私のポジションを終ります。

司会 ありがとうございました。それじゃ，坂本さん。

3. 現場のマネージャとして

坂本 まず，私に与えられた討論のスタンスですが

(1) 現場のマネージャとしての意見を述べて欲しい。
→ 40 代 すなわち “現場のマネージャ” だろうということ

(2) 企業の開発現場にいる者からみて，標準化の動きをどう見ているか?
→ それらの良い点と悪い点。

(3) プロセスの標準化や改善とか ISO-9000 の認証取得について，どういうふうな戦略を取れば良いのか?
→ 現実的な話を。

という 3 つです。

で，先に結論を言ってしまいます。私の本パネルでのポジションですけれども，まず 1 つ目は，標準化というのを ガイドラインとして使うと。‘ガイドライン’を別の言葉でいうと 「技術移転の道具」ですね。その為の標準化というのは大いに賛成である。また，ありがたいことである。

それから，そのガイドラインを間違って何かの規準にしようという動きがある。これに対しては反対である。反対の理由は，効果の割にはロスが多いと。要するに，“審査を受ける為のパワーというのは大変だが，それによって得られるものは，はっきり見えない” ということです。

で，3 つ目が，プロセスは 何のために議論しているのか? という，そのビジネスゴールをもう少し考えないと ダメなんじゃないかということです。

まず，今の 3 つについて 1 つずつお話します。

プロセス標準化というのは，技術移転の 1 つだと思っております。で，例えば ISO-9000 にしても CMM にしても，「プロセスを改善していくためには どんなことをしないとダメか?」という一般論としての項目は，ほとんど洩れなく出ていると思います。そういう意味では，非常に役に立つと思います。もちろん，それをそのまま開発現場に落せませんが，いずれの規格も 項目的には洩れが無く出ていますので，それを社内標準の作成のベースとして利用し，現場に適用することは十分に可能です。さらに，これは会社を跨っての標準ということになりますから，社外の人とコミュニケーションをする上で非常に有効なツールになります。それと，ちょっと泥臭い話なんですけれども，プロセスというのは，開発設計者，マネージメントとかそういったところの従来のスタイルを変えてゆくと

いうことが必要だと思います。しかし, それらを変えるには, すごいエネルギーが必要です。そのツールとして使えると。これは「葵の御紋」みたいなもので, 「世界標準ですよ」と言って, わけのわからないなんか抵抗する人をゴソッと抑えつけられるという, というメリットがあるということです。で, 具体的には, その関係者全員に「まぁ, みんな 世界標準に乗ろうよ」と言って 意識の方向付けが出来るということですね。改善意欲が非常に低い, いわば「ほっておいてくれ」という者に対しても「お前, それでは商売にならんよ」という話で圧力がかけられる。で, エネルギということでは, やはり改善をするためには, 経営的なリソースが必要なわけで, そのことを経営サイドに対して説得するのにも「世界的に通用するためにはこのぐらいは」というようなことで説得材料になるということです。で, それで, 今, 主に議論になるのは, ISO9000-3, CMM それから SPICE といったところになると思います。

3.1 ガイドラインとして標準をみる

私の基本的なポジションとして, ガイドラインとして標準を見た時にどうなのか? ということでお話をしたいと思います。

3.1.1 ISO9000-3

ISO9000-3 をガイドラインとして見た場合に, 項目的には殆ど網羅していますから, 非常に参考になると思います。但し, 「この項目をやりなさい」ということしか書いてないので, どのくらい深くやりなさいとかいうこと。それから, もう 1 つは, 殆どの未成熟な組織にとっては, ISO9000-3 の書いている内容というのは, ほんとに 宇宙に飛び出すほど大変なエネルギーが要る。それに対して, 一歩一歩着実に というところのガイドが無い。で, 優先順位付けをしていないというところが問題かと思います。というようなところから, 成熟度レベル 1 ですね。最近, SPICE では 成熟度レベル 0 というのが定義されているんですけれども, 1 というより むしろ 0 かもしれないです。0 というのは結構多いと思いますけれども 0 の未成熟な組織にとっては, かえって混乱する可能性があると。要するに, もう少し後でも良いやつを取り込もうとして, 余計にひどい目に会うということは考えられる, ということです。

3.1.2 CMM

次に CMM。プロセス成熟度ですね。この成熟度モデルというのは 5 段階になっているのですが, それぞれのレベルの感覚, 現場の感覚というのは非常に適合していると思います。また, CMM を推進している著名な方が色々な資料を作られていますので, そういった資料が経営層をはじめとして色々な立場の人に説得するための非常に良い材料になっています。例えば, ヨードンさんがプロセス成熟度モデルを紹介したものがあります [1]。また他にも, 成熟度レベルが 1 から 5 まで上がってゆくと, それぞれのプロジェクトの状態, 実績の分布が良く行く様子を非常に感覚的にも説得し易く説明したもの [2] など多数あります。それから, 現場レベルにあって, 無理の無い 意識付けが出来る。レベル 1 の人には, レベル 1 なりに 大変ですねぇと言って相談に乗ってあげられる。また, そういう現場レベルに合ったものだと。但し, これは 5 段階というのは, 私は粗いと思っているんですけれども, 非常に一所懸命 改善努力をして 1 ランク上がるのに まあ 2 年から 3 年, まあ非常にラッキーにいっても 1 年くらいはかかると思います。そういった, しつこく, ねばり強く改善して行くという, そういう取り組み, まぁ, それだけのエネルギーがもつかどうかというところですね。また, 各レベル, そこの組織はレベル 2 ですよ とかいうアセスメントする時に, 非常にディジタル的なアセスメント項目になっているます。で, そのアセスメント項目を裏返せば「レベル 2 になるためには, 何をしたら良いのか?」ということになるので, それを誤解してしまって, ディジタル的に改善項目を理解するなんていう人もいたりして, この辺りは ちょっと問題だな, と思います。

3.1.3 SPICE

それに対して SPICE というのは, 今までの色々な議論をここに集約しようという動きがあるということで非常に良いことですね。例えば, アセスメント内容が実態に適合しています。これは先ほど CMM の場合は 非常にディジタル的なアセスメントでしたが, SPICE の場合, 例えば「メトリックスのためのデータを収集していますか?」というアセスメント項目に対しても, レベル 1 はレベル 1 なりに, レベル 5 はレベル 5 なりに設定されています。これは非常に実態チェックに適合しているように思います。で, そういう実態に適合するために, アセスメント項目が 8601 項目という非常に多量になる。で「これは問題じゃぁないか」という意見もあります。ですが, 私は「問題かなぁ?。いや, むしろそのくらいあっても良いんじゃないの?」という気もします。また, SPICE は CMM と同様にアセスメントの結果によってレベル分けをしております (図.1) が, CMM は 1 から 5 まであり

ます。SPICE は, この 1 から 5 にほぼ対応しているんですけれども, さらに 0 を設定しています。実はこのあたりは, ワインバーグさんが同じように レベル 0 を設定していて [3] 面白いなぁと思ったんですが, その後 SPICE がこういうのをやってきたと。で, うちのメンバーの 田中 敏文さんが, 同じようにレベル 0 もあって, さらに レベル−1(マイナス 1) も定義しています。その レベル−1 というのは, 先ほどの久保さんのご紹介にあったような, ああいうパニックしたプロジェクトになると俄然と元気になる, 火事場が生きがいという, そういう人達がいる組織だいうことです。まぁ, そういうふうにレベル分けしているんですけれども, 私としては, まぁ, 先ほどの CMM も同様に 2〜3 年もかかって 1 つ点があがるというよりも, 8601 を満点としておいて 「今月は 10 点上がった」だとかいうほうが, 皆のインセンティブドとしては良いのかなぁ, などと勝手に思ったりしています。

3.2 審査基準としての標準をみる

まぁ, ガイドラインとして見る時には, そういうことで良いと思うんです。けれども, 審査基準として見るときにですね, 私は, 技術者としても, またマネージャとしても, こんなことやっても効果無いと思っていて反対です。理由は 色々あるんですけれども。 CMM は国防省の仕事をしない限り関係無い。我々が「審査してくれ」と言っても多分審査してくれないのではないかと思います。で, 色々あるんですけれども, 何故反対か?というのは, 要求仕様 (*Requirement*) があってプロセスを通してプロダクト/最終成果 (*Product*) を作り出す。それを式で書いてみました。

$$Product = Requirement * f(Pe, Pi)$$

f は関数 (function) です。Pe というのは明示知[8]。はっきりと, こうやれば, こうなるよ, と判っている部分。一方, Pi は, 暗黙知[9]です。誰かにノウハウはあるんだけれども, きちっと外に伝わっていないような領域です。まぁ, 明示知の中には, 規格とかも含まれると思います。例えば ISO9000 だとか SPICE なんかが, こんな領域だと思います。こういう領域に対して, 現場の工夫だとかノウハウという類の経験値とかそういうものがあると。そして, これは残念ながらプロセス全体の中では圧倒的に Pi(暗黙知) の方が多いし, その占める割合も大きい。

$$Pe \ll Pi$$

[8] Process with Explicit Knowledge : 規格, 標準, 手順, 論文, 書籍 etc.
[9] Process with Implicit Knowledge : ノウハウ, 現場の工夫・改善

ソフトウェアの開発の場合には。そう思います。で, その中で, Pe(明示知) の中で判ったことだけを規格にして MUST 項目にしたのでそれで良いのかという話があって, それはうまいこと Pe は Pi でくるんで現場で活用するということが無いと, うまいことゆかないと思っています。で, 我々がやるべきこととしては, こういう式があるんだけれども, Pe のあたりの定量化を一生懸命しているんだけれども, 結果のプロダクトをどう計測するんだという議論無しにされていて, ちょっと片手落ちだなぁと思っています。それから, 暗黙知を如何に明示知にしてゆくかという, f の内容の分析ですね。この関数の中身をもっとはっきりとさせてゆくということが必要だということです。

司会　はい, どうもありがとうございました。では次に, 桑名さん, お願いします。

桑名　私自身は, 個人的にはプロセスの研究には絡んでいないんですが, 大場先生から「プロセスの標準化や改善に対する考え方と ISO9000 の意義を述べて下さい」, 条件としては「開発現場の視点, また現場監督の立場で述べて下さい」とスタンスを与えられましたので プロジェクトの経験などを通じて感じていることを述べたいと思います。また, 私の最近の研究に関連して “これからの” というところも述べたいと思います。

4. 品質保証システムその解釈範囲の拡大

まず最初に, 私自身の経験の話を少しさせていただきます。私の基本的に考え方としては, 坂本さんもおっしゃられたように, こういったいろんな作業標準ですとか品質標準に関して, ガイドラインとして使うことは全く私は異論を持っていません。もう 10 年くらい前になるんですけれども, NTT 入社以来 ずっとガイドラインというものを使ってきました。数年前にプロジェクトは無くなったんですが, 入社してしばらく DIPS というメインフレームの部隊に属しておりました。そこでのガイドラインが「DIPS 作業標準」といわれるものです。私は, 大学でソフトウェアエンジニアリングを勉強してきたのですが, 最初にそれを見たときは, 「なんだ, これは?。なんでこういうのがなけゃいけない?。ただ良いプログラムを書けばそれでいいのでは?」というふうに思っていたわけなんです。ところが, いざプロジェクトで仕事を始めますと, このような作業標準が無いことにはどうしようもない, というのが段々わかってきました。そしてそれ以後, 私は 1 個の自分で核となる作業標準をやっぱり持っております。それは 今申し上げました DIPS 作業標準ですね (DIPS 作業標準というものは, 現在,

実在しませんが)。

で, 今まで様々な, 例えば Internet を使ったネットワークの設計ですとか, そういった作業を幾度も経験してきたわけですが, いずれもずっとこの中で使ってきた標準の核になるもの (考え方) は, この DIPS 作業標準に基づいた考え方を使ってきました。特に, その中でもドキュメントのフォーマットの考え方/位置付けへの影響というのは, 非常に大きかったかなぁと思っています。例えば, 設計文書の中には何を書かなねばならないとか, 詳細設計には何を書かなきゃいけないといったことですね。また, ドキュメントには, フォーマルなドキュメントとインフォーマルなドキュメントの 2 つがあると思いますが, この「DIPS 作業標準」では, 「検討項目表」「コメント表」「問題票」「議事録」といった種々のインフォーマルなドキュメントまで規定しています。例えば「コメント表」とは, ドキュメントに対して自分が思ったコメントを書いてドキュメント記述者に送ると, それからフィードバックがかかってくるというものです。また「問題票」。これは「PQR 票」と呼んでいますけれども, P は Problem, Q は Question, R は Request という意味です。これは, 何かあるルーチンを使おうとしたときに, 「これ, 何かおかしいなぁ」と思ったり, あるいは「こういった形で改造してくれないか」といった要望がある場合には「Request 票」を出すんです。で, そういった色々なインフォーマルなドキュメントの形態が, 実は作業標準の中に決められていて, それを個々のフェーズで使います。このような作業標準を, 例えば, Unix の GUI の設計/開発でも TSS の開発にしてもそうですけれども, 開発のフェーズの中でずっと使ってきました。

だけど, 使ってきたやり方が違っていまして, 個々の場合に応じて それを revise してきました。revise とはどういうことかと具体的に説明しましょう。例えば, 10 年位前は, 情報は みんな紙ベースでメンバに流していました。けれども今は, もうほとんど電子メイルであったり, Unix のブリテンボードの News システムを改造して使ったりと。このような工夫は, 現場の人間にとっては簡単なことが少くありません。ほとんど Emacs の mlisp をちょこちょこっと改善すれば, すぐ出来てしまう話なんですね。で, そういったものを活用して現実に開発をしています。もちろん, 開発をしているのですから, その基本となるものが必要でそれをきちんと定める必要はあります。しかし, その基本となるものを定めた上では, それを現場で revise していっても良いんじゃないかと。私の今日の発表のスタンドポイントの 1 点目は『開発標準というのはガイドラインとして使うべきであって, しかもフレキシブルでなければならない。そして, それは使う側にまかせて良い』ということです。

もう 1 点のポイントとしては, 「何をプロセスのベースにしてものを考えるのか?」ということです。例えば開発では, ドキュメント中心に考えるという考え方もありますし, ソースコードを中心にして考えるとか, 色々な考え方があると思うんですけれども, ともかく何か基本となるものを定めて, それで開発をしてゆけば良いんじゃないだろうか, またプロセスを見てゆけば良いんじゃないだろうかと思います。で, そこに, ぐちゃぐちゃと色々な物があるんですけれども, その大きな物を色々な物を適用して考えるから かえって複雑になるので, 何か 1 本, すじ道を定めて, それでやってゆけば, うまくゆくと。で, しかもやり方としては, 今までのプロセスに関しての経験から言うと, 大きなプロジェクトにいきなり提供しても これはぜんぜん駄目で, まずはスモールスタートで, サクセスストーリを作ることが一番大事じゃないかと。私自身, 入社以来, 今までずっと色々なプロジェクトでやってきましたが, その度に, 上司から「常にスモールスタートで行け。サクセスストーリを作れ」ということを, ず〜と言われてきたました。で, 最初のうちは「何かなぁ?」と, その言われる本当の意味が良く実感出来なかったんですけれども, それはサクセスストーリを作ることによって, 実はそのやり方というのが, 広まっちゃうんですね。1 つの部門の中でのやり方が, サクセスストーリであるという評判によってそのほかの部署も真似をし始める。それが段々と広まってゆく。このような進め方というのが, 組織の中で大事だと考えます。そして, これは組織構造の多様化だとか, 協調的な開発だとか, 技術者管理的側面, 利用者なりの統合とか使い分けも必要だと考えます。

それから, グレイゾーンが増大してきているという考え方があります。実はその 今までプロセスの中に, 考え方として, 先ほど坂本さんが

$$Product = Requirement * f(Pe, Pi)$$

という式を出してくださって, なるほどと思って感心したんです。だけど私は, *Requirement* をかけている部分に, そもそも間違いがあるんじゃないかと思っているんです。要求技術, 獲得技術なんてゆうことが, 昨今言われていますけれども, そもそも, 獲得するという考え方が間違いで, 実は *Requirement* を “書いてあげること” が大事であると思っています。で, ここの部分が, そもそも成り立たないのに, こんなこと

やってもしょうがないんじゃないかなぁっていう気も実はしてます。それはどういうことかと言うと，先ほどグレイゾーンの増大といったのは，ミシガン大学の Olson 教授らを含めた多くの研究者が主張したり [5], IFIP でも 13.2 というのが約 2 年前に出来まして，その中で User Centered Design というのを最近やり始めています。以前から言われていることですが ユーザの要求獲得というのは，本当に難しい仕事です。で，これを，もっとこうもう少し cyclic に進めよう。1 回作ってユーザに使ってもらって，もしくはユーザに実際にそのプロジェクトの中に入ってもらって，それを繰り返しながら，要求を明確にして行くだとか，システムを作ってゆくという考え方なんです。このような User Centered Design でやらなきゃならない，といったことが出てくると，もはやプロセス自身も明確に規定したり，要求定義からベースにして考えるということが，うまくゆかないんじゃないだろうかと考えています。
そのような観点からも，品質保証システムの解釈範囲がフレキシブルじゃないとうまくゆかない，と考えています。

5. ポスト資本主義とソフトウェアプロセス管理

最近読んだドラガーの「ポスト資本主義社会」という本が頭にずっと残ってまして。ソフトウェア開発というのは，工場の中で作るような考え方もあるんだろうけれども，もう 1 つの考えとしては，やっぱり知識的な労働者/専門家というのがどんどんどんどん増えてくるだろう。そうすると「それをどういうふうに支援してゆくのか?」ということをそろそろ考える必要があろうと思っています。

実は今まで知識というのは，道具とか工程とか製品に対して適用されてきたが，今後は，その，ダイナミックに増えたり変わったりする知識自身に，適用されるべきであると考えています。で，そうなると，プロセスというか，その問題同定だとかは，問題解決型タスクへ移行するだろう。これはどういうことかと言うと，問題自身が分かっていれば，それは時間をかけて，人をかけてやれば何とかすれば解けるんですね。ところが最近は問題自身が分からなくなってきている。先ほども久保さんがおっしゃっていたように「何が問題か?」というのが実は解らない状態です。皆，よって集まって色々と議論する。「お前が悪いんだ」とか「あいつが悪いんだ」と，皆 誰でも言いたいことを言うんだけれども，それは実は問題自身が同定出来ていないからじゃないか?。で，それを同定するような，何かツールだとか環境なんかを整備してかなければならないだろう，というふうに思っています。

で，今 私たちは 3 つの大きな目標があるんじゃないか考えています。それは

(1) 生産物のまず品質を上げる
(2) 組織効果を上げる
(3) モラル

という話におちつくんじゃないか?。この 3 つが，実はプロセスという物になるかもしれませんけれども。で，この中で当然モラルということに対しては，色々な従来からやられている研究があるし，これからも続けていかなきゃいけないテーマであることは自明だと考えています。例えば，杉村先生が面白い本を書かれていますけれども，この QWL(Quality of Working Life) というような形で言われている，色々な研究があります。また，デマルコさんのピープルウェアにしてもそうだろうと。で，こういう考え方のもとに，プロジェクトをその担当者に任せることによってモラルが向上も期待できますでしょうし，プロジェクトの志気の高揚も期待できると思います。

モラルに関しては，それはそれなりの人に任せておいて，私たち (ソフトウェア工学研究者) は，今，何をやらなければならないか?ということで，今向かっているのが，Work Group Computing for Intellectual Team Work です。先ほども申し上げましたけれども，ある何かを基本にして走ってかなきゃいけない。その走ってゆくというのは，データ中心でも良いし，スケジュール中心にしても良いんですけれども，その，何かを中心にして走ってかなきゃならない。
で，私は，Corporate Wide (全社大の) な Information Sharing, Documentation Sharing というのを中心にして Comunication and Collabolation Support to ... とか，マネージメントテクノロジを支える技術が必要だろうと考えています。そしてそれにトライしてゆく方向が，これからのソフトウェア開発者を救う道になるんじゃないだろうか，と思って研究しております。

司会 どうもありがとうございました。じゃあ，兼子さん。

6. QC 屋からみたソフトウェア屋のプロセス観

兼子 まず，このテーマの前に，私そのもののポジションについて，簡単に紹介させていただきますと，私はソフトウェア技術者ではありません。ソフトウェアでお金をとっていたのは，大学 4 年生の時に，ある雑誌の編集部で 2〜3 ケ月に 1 回記事を書いてプログラムを載せるというアルバイトを 1 年間だけやりました。ですから，バイトしていた時も，基本的に 1 人で

作っていたと。で, あるプログラムの時に, 友達2〜3人使ってプログラム作ったことがあるんですけれども, これは例えば「グラフィックのデータを作ってね」とか「音楽のデータ, 打ち込んでね」とか, そんなことでしたから, 基本的にチームで仕事をしたことも無いわけです, ソフトを作るのに。だから, そういう意味では, 皆さんと全然違う, というか肌では, 多分わかっていないんだろうなぁと思います。で, なんでそんな私が パネラとしてここにいるかというと, もともと私は東京大学で 品質管理や応用統計をやっている講座にいて, それが修士ぐらいからソフトウェアの品質管理をやろうということで研究をずっとやっています。それで, 私のポジションは, "20 代で" ということなんですが, 「顔は二十歳, 腹四十」というのが昔から続いていまして, 実は 33 歳ですので, お間違えのないように。で, 「書生論」というか, "リサーチャの立場から" ということになるわけです。

まず, どうしてこのソフトウェア屋さんというのは, 他人に評価してもらいたがるのだろうか? というのが非常に気になる。例えば, 例のハンフリーの成熟度の 5 段階のレベルにしても「何でレベルなんとかと他人に言われなくちゃいけないの?」。非常に気になります。物を作る時に, お客さんがハッピーであったらば, 要するに, 皆さんが作った物を, お客さんが受け取ってハッピーであったならば, なんのマネージメントなんか要らないわけですね。それは QC 屋から見ると, そう思うわけです。ともかく, 約束の納期までに約束のお金で, そこそこお客さんが納得する品質のものが手に渡っていたならば, お客さんはそれでハッピーなわけですね。で, 皆さんが 例えば何日残業しようが, 休日出勤しようが, それはお客さんに関係無いわけです。お客さんは, 成果物だけを見ているわけです。

で, こういうこと言うと, 色々言う方がいて「いや, NTT なんか作り方も言うよ」 とか言うわけです。「xx というドキュメントの標準を使って, yy とかやりなさい」などと, 作り方まで指示すると。でもそれは, プロがプロに対して仕事を出す時に「こうやってやれば多分うまくいくだろう」という蓋然性の高いことを知っているからそういう指示をするわけであって, それでやって欲しいというのは, 要するにメンテナンスのこと考えてるからだけです。だから, 要はどんなプロセスで物を作っても構わなくって, 成果物が, お客さんの要求をちゃんとあるいはそこそこ満たして, お客さんにハッピーなものが届いて。また後日「直してね」っと言ったらすぐ直っている/直すようになってると。そうなってれば, どんなプロセスでやっても良いわけですよ。だから, プロセスがこういうプロセスで動いてたらプロセスの成熟度がどうだとか, じゃぁプロセスレベルが 1 だからレベル 2 にしましょう, とかね。レベル 2 だから レベル 3 にしましょうっていう, その発想自身が, 大体僕は気に入らない。良いんですよ, レベル 0 でもレベル 1 でも。それでお客さん, 困ってなきゃ良い。

でも, お客さんが, 困っているんだったら, レベル 5 だって, レベル 6 だって 駄目なわけです。どうしてそういう発想の逆転が無いんだろうか?。何か自分の事ばっかり言っている。

また, ソフトウェア屋さんで思うのは, 誰か他の人が うまいこと考えてくれて, その成果をスポンと自分のとこに取り入れてやれば, 自分のところの問題は全部片付くという, 昨日あたり出た Silver Bullet[10] を求めているような人達が, どこに行っても多いような気がします。ある所にとっての Sillver Bullet は, 別の所の Silver Bullet であるわけがない。

大体, ソフトウェアと一言で言っていますけれども, 世の中には色々なソフトがあるわけですよね。で, 例えば, ここにいる皆さんが どんなソフトを作っているか知りませんけれども, A というソフトと B というソフトをうまく作る作り方というのが, どれもこれも同じわけがないじゃないですか!。だって例えば, 車, CD ラジカセ, テレビ それにロケット作るのが, 全部同じプロセスでいくと思います?。よく, ソフトウェア屋さんは, メタなプロセスの話をすぐしたがりますけれども, そんなこと考えるの止めた方が良い。そんなこと, 出来るわけが無い, と思った方が良い。で, 頭の良いやつは, すぐにメタなプロセスを持ち出して抽象化しようとしますけれど, そんなことしたってうまくいくわけがない。今, 自分のやっているプロセスを, 自分の所で作っているプロダクトが良い物出来るように改善してけば良いんですよ。そう思いませんか?。だって, 例えば, トヨタは, 非常にきっちりと精密なプロセスを作り上げています。ところが, あのトヨタの人達が, 「一般的な乗り物を作るため工業プロセスは如何にあるべきか?」というのを語ったことは 1 回も無いわけですよ。彼らが語っているのは, 例えば車だったら, こういう問題があると。それをちゃんと作り込むためには, どないしたら良いの?ということを, ず〜っとやってきてるわけですよね。で, それがきれいになってくると, もしかすると参考になるかもしれない?, 他の人達にとって。そのプロセス自身

[10] 銀の弾丸

が。だけども, 彼らは1回もメタの, 工業製品を作るとか, 乗り物を作るとか鉄を曲げて何か物を作るとかというメタなプロセスのインプルーブメントをやったっていうことは1回も無いはずです。だから, この中でメタの話をしたがる性向の人が多いですけれども, アカデミアの人もいるのかなぁ?, そういう人で。止めた方が良いと僕は思います。あまり望みがないですから。その, 極端な話を言うと, 何でも良いんですよね。

7. ISO9000-3 と第3者認証について

まず, ISO9000-3 について言うと, あれは物を作るプロセスについて一言も触れている積もりは無い。だから, あれでプロセスマネージメント云々の話題にのぼってくるのはおかしいんじゃないか。あれはあくまで, Quality System の element を書いたに過ぎない。だから, どういうふうに物を作れ, ということについては触れてない。だから, 作る時に, 品質を気にするなら こういうことやっといた方が良いよ, ということを言っている。それで果たしてプロセスって定義できるんでしょうか?。で, それとは違う所で, プロセスって定義出来るような気がします。だから, 単純に 物作りのプロセスとして ISO9000-3 を持ち出すのはちょっとおかしいと思っています。で, なんだかの物を作るプロセスの中で, 「品質を気にする」という状況が生じた時に, ISO9000-3 というのは役に立つかなぁ? という程度ですね。

それから, 第3者認証について言うと, 品質システムがどうなっているかによって 取引の条件になる/しうるということは, これは避けようが無いと思っています。それは, 何故 第3者認証みたいな話が出てきたかという背景には, やはりヨーロッパ経済統合の話があって, 他の新しい人達が商売に参入する時に, その Quality どうやって担保すれば良いのか?ということを考えた末に, あれを持ち出してきている訳ですよね。そうすると, 今までの経験ではなくて「今, こういう作り方しているから良い物を作れるに違いないから入札に入れてよ」ということを担保出来なくてはいけない。だから, 作った物ではなくて, 作るプロセスのようなもので取引に応じるか応じないかを決めるという, という仕組みは 多分無くせない。それが ISO9000-3 とか 9000 シリーズに基づくかどうかは別ですけれども。そういうシステムは, 人間が1回そういうシステムがあるということを覚えてしまいましたから, 多分消せませんし, これからも消えて無くならないと思います。

で, ソフトウェアの認証については, 今 ISO9000-3 を JIS 化するという動きはあって, 秋ぐらいには JIS になると思います。でも, ここにいらっしゃる皆さんは, はっきり言って認証なんか受けなくても, 商売殆ど困らないですよね?。一部のメカトロ屋さんとか別にすると, 所詮ドメスティックな中で商売しているわけですから。ISO9000-3 の認証をとるということで, それで例えば社内の品質システムを良くするという為に使うんなら, 別に構わないと思いますけれども, 認証をとることだけが目的ならば, 止めた方が良いと思います。

8. これからどういうアプローチをすれば …?

それじゃぁ, これからどうしたら良いのか?と。まず, 皆さんは, 今, ともかく何か作り方を知っていますよね, 物を作る時に。その作り方をまず, きちんと, 紙に書かなくとも良いけれど, 少なくともメンバ内では伝わるような形にはした方が良い。で, その通りやって, 仕事がうまく行かなかった時には, その仕事の仕方のどこに問題があるかフィードバックをかけなければいけない。で, それがドキュメント化されているというのは, 必ずしも 本質じゃぁないような気がするんですけれども, ともかく皆がどうなっているというのがわかってるシステムがあって, 仕組みがあって, そして それに基づいて作って, 問題があったらその作り方に改良のフィードバックをかける。これが, 多分 物を作る時に, 今良い物が出来ないけれど今後良い物を作るという時の常道ではないかと思います。昨日, 何かの時に, 要するに優秀なエンジニアを集めれば良い物が出来るという話がありましたけれども, おっしゃる通りです。優秀なエンジニアを集めれば, まあ多分良い物が出来るという慨然性は高くなる。けれども, 残念ながら優秀なエンジニアだけを集めて物作れるほど, 世の中悠長じゃないですよね。だから, 優秀なエンジニアだけを集めれば良い物が作れるというのは当り前だけれども, 優秀なエンジニアだけ集めて仕事出来ないという前提があるんだから, 要するに凡人使って良い物, あるいはあんまり大けがしない物を作るには, どないすればいいんだ? ってことを考えるのが当たり前であって, 今でも「やっぱり優秀なエンジニアを集めれば良いね」なんて夢を語っているんではとても駄目ですね。で, 凡人でも, ここにいらっしゃる方の何割かはすごく優秀な人がいて, で, やや優秀な人がいて, ちょっと普通の人かな位の人が何人かいるんだろうと思うんですけれどもね。こんなところへ, わざわざ来るくらいですから。で, そういう人達だけをメンバにして物作る時にどうやって作るんですか?。で, その為には, やっぱり身の振る舞

い方というのをきちんと決めて, その振る舞い方を直すという形でないと良い物は作れない。だから, ここにいらっしゃる方が, すごく優秀な人たちばっかりだと, 実は僕の言っていることは, 殆ど届いてくれないような気がしているんですけれども。だから「何を言っているんだ」と思うような人は 致し方無いと思いますが, 私はそう思う。

で, やはりプロセスというのは自分のところで, 自分の所で作っているプロダクトに合ったプロセスがあって, それをぐるぐる改良してゆく。それは自分達がどう失敗したかというのを分析して, きちんとフィードバックすることである。で, 実際に 失敗のフィードバックが全然されて無いように, 私は思います。どの会社に行っても, バグ情報を山のように取ってありますけれども, あれを 2 度と見ないですよね。以前, ある会社の開発部門に行って「1 人 1000 件持ってますよ」とか「見たきゃ, どうぞ」とか言われてびっくりしたことがありますが, まあ, この手の話は, 幾らでも聞くわけです。で, 大きい会社では, もう電子的な帳票にして貯めているし, 小さい会社でも, ともかく紙に書くなり電子的なメモ書き程度にはどもかく残っているわけですね。それは, やっぱりオープンなバグをきちんと修正したということをトレースしなくちゃいけないから, 普通は残っているわけです。でもオープンバグをフィックスした瞬間に, そんな情報は誰も見なくなるわけですよ。一体何が悪かったのか?。技術的な問題なのか, プロセスの問題なのか, それともたまたま前の日, 麻雀してて徹夜してても, 頭がクラクラしてただけなのか?。そういう分析を全くやらないままに。要するに過去の反省が一切されていないなまに, 漫然と仕事が続けられている。そんな状況なのに, 誰かが「こんなプロセスどぉ?」って言うと, バァーと行っちゃう。こんな愚かな, それこそレミングのようなやり方は止めて欲しいと思います。そろそろ時間だと思いますんで, 野次って終わりにしたいと思います。どうも失礼しました。

司会 どうもありがとうございました。以上で, パネリストからの提言を終わります。これからフロアの皆さんに, 議論なりコメントなり, 思うところを語って頂きたいと思います。積極的に議論に参加して下さい。それではどうぞ。

9. ソフトウェアの対象を考慮しなくて良いのか?

平野 (中央システム) 私は, 社員 200 名ほどのソフトハウスで, 十数年 仕事をしていまして, 昨今, 研究開発部門に移ったわけなんです。で, ドロドロの泥沼を這いずりまわるような開発をやっていた人間から見ますと, プロセスあるいは ISO9000 等に関して, 私が一番感じていることは, プロセスの標準化というのは非常に難しい。特に, 下手な適用の仕方をすると, 現場から反発を食らうだけ, という感じがすることがあります。それは, 今 議論されている規格の個々の点について何が悪いということではなくて, むしろ, ソフトウェアというものを ひとかたまりで扱おうとすることが無理で, もっと細分化しなければうまくゆかないのではなかろうかと思うわけです。例えば, 久保さんが例として示された OS を作るようなソフトウェアの作業と, それから比べてずっと小規模のアプリケーション・システムを作ることを, 無理に同じレベルで議論をしてしまう可能性・危険性があるのではなかろうかと考えているんですけれども。

司会 久保さん, どうでしょう。

久保 こういうふうに考えることは出来ないですかね?。色んなソフトの種類というか, パッケージか, カスタムかというのもありますし, それからドメインも違うし, さらにプラットフォームも違う。だけど ISO9000 は 1 つですね。それは, 色々と違っていても 人がチームを作って作業をするというのは同じ。仕事をするためには, 何かしら計画というのを作る。計画の中には, 達成したいゴールがある。ゴールは品質というので表現する。それで, 作ったらばトラッキングしていって, 達成を確かめて, 納めると。ここいらは, 同じに出来るんじゃないの?, という提案じゃぁなかろうかと思うんですけど。それではなかなか納得しないですか?

平野 要するに, そこで止めれば良いんじゃないかと思うんですけれど。気がつくと, もっと前へ前へと進まされるのが人間の常でありまして。

坂本 自部門に合わないと思えば, 止めればいいと思うんですよ。で, 自部門に役に立つと思うところを, 利用すれば良いんで。実は, 私の会社の中でも, ハンフリーさんの書かれたものは, 非常に利用させてもらいました。例えばこんなふうにです。「ソフトウェアは 何故いつも失敗ばっかりしてる?」と経営トップ層がいろいろと言う。それに対して ハンフリーさんの本を持って行って「この本に書いてあるようにプロセス成熟度の改善に取り組みましょう」と言っています。平野さんのご質問は, ソフトウェアのドメインによって標準化というのは役に立たないか? ということだったと思うんですが, 先ほどハンフリーさんの本を利用させて頂いた時に, 経営トップの方から「あの本に書いてあるのは, 要するにホスト系で, 我々のやっている組み込みソフトには関係無いんじゃない

か?」というのがまず最初にありました。で, それに対する回答として「いや, ここで言っているのは 経営者が決心して経営資源を投入し きちっとプロセス改善を実行することが唯一の道だと言っているわけで, それは組み込みソフトだろうが, ホスト系だろうが関係ないです」と言っています。要するに世間で議論されていることや, 標準化の動きについてその通りにする必要は全く無く, 自分で判断して“いいとこ取り”をすれば良いと思います。

桑名 私も, 別々であって良いんじゃないかと思っていますし, 過去に自分もそうしてきました。だけど, スタートポイントとして核になるものはあったんですが, それは どんどん年月が経つことによって, 時々に 他から色々なものを吸収して 変わっているんですよね。だから, 吸収したなりにカスタマイズして revision が出来ています。「これは使えるかなぁ」と思ってやってみて, うまくいかなければ また直せば良い, と思っています。

兼子 先程, 久保さんが言われた, うんと generic な意味で, 例えば, 設計してコード書いてテストして云々という generic な意味では, 多分有り得ると思うんですけれども。でも, ある特定の何かの物に, 良い物を作ろう!と作り込んだ時には, その物を実現するために必要な技術的な何かによって, プロセスが決まる部分はあるはずで, そこからさらに本当に日々の仕事に役に立つようにプロセスを書こうとすると, やっぱり, ターゲットですとかそういう物を分けざるを得ないと思います。また, そういうふうにすべきだと私も言ったつもりです。

司会 良いですかね? では, 次の方。

10. 標準は何のためにあるのか?

野口 (新日本製鐵) 私は, 実は初めは 鉄の製造の世界にいたんです。で, そこから ソフトに来ているもんですから, 何かこう, 今更プロセスだ, 標準化だと言われても, どうもなんか変な感じだなぁというふうに思うわけです。例えば, 兼子先生が言われた 「一般的なプロセスなんて言っても意味が無い。例えば, トヨタが自動車の作り方のプロセスとかを持っているけれど, あれを一般化するようなものを持っているわけでは無い」という話をされていましたけれど, 確かにハードウェアの世界ではその通りです。例えば我社の場合, 10 の製鉄所があって, その中で 同じようなプロダクトを作っている製鉄所というのは幾つもあるのですけれども, それぞれの製鉄所がそれぞれの製造の標準を持っています。そして, それぞれの現場の作業標準を見てみると, 同じようなところもあるけれども違うものもあります。で, それはやはり個々の場所で, そこの設備のそのやり方でないと, うまくゆかない。で, 例えば, 釜石というのは線材という針金の材料を作っているところなんですけれど, だんだん製造を縮小する方向にあって, で, スチールコードというタイヤのラジアルタイヤに使うコードを, 別の製鉄所の方が製造することにして, その移転をやっている時に, 私は丁度入社したんですね。で, そこの部門の話を聞くと, どうしてもうまく作れない, と。だから, そういうものであって, 標準書をいくら移転しても駄目な世界もあるし, で, そうは言っても, そこの場所での標準というのは必ず存在していて, これをやると逆に誰が来ても, つまり転勤者が来ても, その通りやれば何故かうまく仕事が進むというような世界があるわけですね。

で, 言いたいことは何かと言うと, 別に我社でも, 本社がこういうような作業標準あるいは製造標準で物を作れって言っている訳ではありません。それならば, それはソフトでも多分同じであって, 極端な話をすれば, プロジェクト毎に多分違うやり方があって, そのやり方でうまくゆく, というような場合も多いんじゃないかと思いますね。そうすると, じゃぁ, 標準というのは何のためにあるのかと言うと, 素人のプロジェクトマネージャーが, 取りあえず何とかプロジェクトをやるためにあるんじゃないかと, 私個人は思ってます。と言うのは, 私が入社して1年半位してから, 工場の自動化のあるプロジェクトに行って, いきなりプロジェクトマネージャーをやらされたんです。で, ソフトを作ったこともないし, そもそも何かスケジュールを引けと, ボスに言われても, そもそも一番初めに分析をやって設計をしてコーディングしてというプロセスを全く知らなかったんです。で, 取り敢えず, 昔のプロジェクトをやったものとかあるいは, 標準書らしきものを見ると, 何かそういうことが書いてあるんで, なんか見積りのやり方も書いてあるんで, 取り敢えずそれに沿って見積りをやって, そしてそうしたら, システムが1年後には出来上がっていたと…。まあ, こういうような経験がありまして, で, 何で出来たか良く判らないけど, とにかくその通りやったらうまく行ったと。で, それはまあ typical な例じゃないのかもしれませんけれど, しかしまぁ, そういうのが標準であって, そのあと何回か, まぁそういうプロジェクトを積み重ねてゆけば, そこから今度は工夫をして, 自分が段々そこから逸脱してゆくという世界もあるんじゃないかなぁ, というふうに思うわけです。だとすると, 「標準というのは, 一体 何の

ために作って, そして それを認証するというのは, 一体何のためにあるのか?」。どうも何か今ひとつよく分からないなぁ, というのがコメントです。

で, 兼子先生から「ソフトウェア屋さんはプロセスについて議論するのが異様に好きだと。で, 外の人が自分達のプロセスがどういうふうになっているかなんて, へんちくりんな話だ」とおっしゃいましたが, 実は, 鉄の世界でもあるんですよね。"パイプ" というのは, あれは, 外から検査官が来て, どういうふうに作っているかを検査するんです。だから, ああいうような鉄みたいな, 汎用品みたいな世界でもそういうのはあるんで, 別にソフトウェア屋さんだけが好きなんじゃないでしょうけれど, 多分ソフトウェアの場合は, どうも, なんか, 外から見るとブラックボックスで何だか良く分からないから, とにかく製造プロセスまで, 良く見ておかないと不安な感じにかられるんじゃないか, というふうに思います。

兼子 今のことに関しては, それはやっぱりプロセスとマテリアルというカテゴリというのは, ハードウェアにあって, 破壊検査をしない限り, 中の, 例えば, Quality が分からないというものに関しては, 当然, 作り方を見るわけですよね。今のパイプもそうですし, メッキなんというのも, 実は表面検査しても, 中のあれとか分からない。あれはきれいな所で, ちゃんとやっていますか? というのを見るという, 確かにプロセスとマテリアル。プロセスを見ることによって, というのはあるということはあります。で, ちょっと, もしかすると僕の言ったことが誤解されてたのかも知れないんですけれども, 僕はその, ソフトウェアで, プロセスを見るということは, やらざるを得ないというふうに思っているんですよ。というのは, 標準品を売るのではなくて新しく契約して作りましょう, と言ったときに, そこの実力を評価する, で, 商売の入札の条件に入れる, といった時に, 「今までこんなこと作ってきましたよ」ということは, 実は 機会平等の原則に反しているわけです。だから, 今, こういう物を作り得る, プロバビリティー, パシビリティーがあるよ, ということだけで, 商売に参入することを認めざるを得ない, というシチュエーションに今なっていて, で, そのシチュエーションで相手を評価する時に, 今, アイディアとして持っているのはプロセスを見るということだけだから, そのプロセスで評価せざるを得ないということは, 多分無くならないだろう。で, 僕が言っていたのは, 「ソフトウェア屋さんが, すぐ, メタの話をしたがっちゃうのが異様だなぁ」と。要するに, そんなメタな話をしなくていいじゃないのと。要するに, なんとかのなんとかプロジェクトチームで OS を作るのは, 通信管理作るのは, 例えば Windows のライブラリを作るのには, どういうプロセスでやったら良いんですかという議論が 100 個も 200 個も集まってくると, なんとなくソフトってこんなふうに作ってくのかなってゆうのが出てくりゃ良いんであって, いきなりメタの話から始めたってね, 抽象的なアジテーション以外の何物でもなくて, 実のあるようなプロセスがそこから出てくるようには思えないということを言っただけです。

坂本 先ほどの兼子先生の話, 私としては異論があります。それは, 先生がポジションペーパに書かれている内容についても, 現実には必ずしもそうなっていないんじゃないかということで, ちょっと気になっているところがあるんです。

例えば具体的に言うと, ISO を受診しようと思ったら, 最低 6 ケ月間 (多分 本当はもっと期間が要ると思うんですが) データを貯めないと駄目なんですね。で, 例えば, メッキのプロセスが妥当かどうかというのは, バッとやってみればそれで分かるわけですよ。ところが, ソフトウェアを評価するためにはそうはいかないわけです。だったら, プロダクトで検査したって たいして違わないんじゃないかと思うんです。

司会 司会が意見を言うのもなんですけれども, 今, 兼子さんが言われた, メタなプロセスの議論で, これはちょっとポイントをそろそろ変えた方が良いんじゃないかと思っているんです。それは, どういうことかと言いますと, その個々の物作りのプロセスが違うというのは当然で, それに対して最適化してゆくというのは当然の話で, それが取りも直さず 日本が今までビジネスで成功してきた理由だと思うんです。ところが, それで良いかと言うと僕は, もう一歩進まないといけないんじゃないかと。何故, メタな議論をするかと言うと, 例えば, 僕が日本の品質管理に対して disappointing なところがあると思っているのは, 世界に対して「こういうようなプロセスで, 品質管理をすれば良いですよ」というような, 非常にクリアーカットな議論がされていなかったと。そういう意味で, メタな議論をするということは, 知識を日本の中に閉じさせてないで, 外に持って行くということに関しては, 非常に重要な役割をする。そうしないと, 通じない話がある。そういう意味で, 僕は重要だと思います。で, これは, 知識を貯めるとか, 伝達するという意味で, そして科学という意味で重要だということです。

久保 じゃぁ, 僕にもちょっと喋らせて。

司会　どうぞ。

久保　やっぱり兼子さんにちょっと教えてあげなきゃいけないからなぁ〜。多分, 言葉使いの問題かなぁと思うんですけれども, やっぱりISOにしたってCMMにしたってメタだから利用価値があるんですよね。それから, 兼子さんたちがやっている品質管理だと 横展開, 水平展開という。あれもメタになるから伝わるのだろうと思います。それから, 僕らが知らぬうちに使っている道具がありますけれども, 品質保証体系図だとか。あれだって, ようは, どっかの工場で発明されたものが汎用品になっていったという。だから, まぁ, そういうふうにメタになっていってるというか。それとまぁ, ここまでは意見でしかないですけれどもね。ちょっと, 効き目あるのは, 彼は久米先生というところの助手ですよね。で, 久米先生から聞いたんですけれども, 海外から日本のQCを見学に来たいというんで, おみえになる。そこで, まあ企業は受け入れるんですけれども, そこで来た人は勉強して帰りたい。それから迎える方も, 何か勉強材料を用意するのですけれども, その時にやっぱり材料を提供出来ないんだそうですね。何があるかというと, QCサークルのことしか出ない, 極端なことを言うと。経営はどうやっているか? というと, 成功しているはずなのに, それを伝えられないということをもどかしがっていた, というの話があります。

司会　当然のことながら, 兼子さん, 反論があると思うので ••• 。

兼子　メタな議論をするなとか, 今おっしゃっていたみたいな意味で, メタな議論に意味が無いというふうに言ったつもりは無い。やれば良いんです。だけど, 今のソフトウェア屋さんのレベルというのはメタの話をする前にspecificな話をするのが先じゃぁないの?, と言っているだけですから, それは勘違いしないで頂きたいと思います。だから何人かの, そこら辺に座ってらっしゃるウンと頭の良い, どっちかと言うと浮世離れした人は, メタな話をすれば良いんですよ。で, そういうことをしなくちゃいけない, することが大事だというのはわかるけれど, 誰かが先, 三歩ぐらい, ふわふわ浮かびながら先を歩いている必要はあって, そういう人達がいなくちゃ困るんだけれども, 皆がずるずるとそれに引っ張られることは無い, と思います。

司会　兼子さんの名誉のために言っておきますけれども, 昨年, 私と飯塚先生とか何人かで世界一周したんですけれども, その時, 日本から是非 前から色々聞かれていますので, 日本の品質管理, ソフトウェアの品質管理って何か, ということを説明した方が良いんじゃないかということで, 私の方から提案させて頂いて, 飯塚先生にも快くお引き受け頂いて, いくつかのところで喋っていただいたんです。で, 非常に良いパッケージが出来ているんですけれども, それは実はですね〜, 彼が一生懸命, 土日も休まず作ったという ••• 。非常に良いパッケージがありますんで, 再利用して頂ければと思います。次は君島さんですね。

11. 工程の捉え方について

君島 (富士通)　まだ統計的品質管理やってるほうが, よっぽどメタだと思うんですけどね。僕が言っている, 個別品質管理 [6] さえやりゃあ良いんですから。もう, 品質管理の研究なんてやめちまえば良い。

それはともかく, ちょっとメタな話をいたします。で, 今, 標準工程と個別工程は相矛盾するような捉え方しているというのが, そもそもおかしくて, もう一回プロジェクト管理の勉強をした方が良いと思うんですけれども, 工程というのは固定の標準を守るというのが間違いなんですね。まずWork Break-down Structureという, あのSLCPみたいなものがあって, あれは順序はまったく規定していません。あの中から選びなさいと。で, 選ぶ時に, 白紙の状態からやると大変ですから, 雛型工程というのがあります。これはウォータフォールとかスパイラルとか, あるいは, そのアプリケーションドメイン毎の過去の事例でも良いですけれども, それはバラバラで良いわけですね。それの適当な近いのを見ながら, 納期とか新規性とかを考えて, 工程計画というのを立てるわけですよね。標準工程を守るんじゃないんですよ。プロジェクト毎に決めるんですよ, リーダが。最も納期が短くなるように。という, 「Work break-down structure」「雛型工程」そして「工程デザイン」という3つのファクタがあると。標準工程を守るなんていうのは, 全く初歩的な理解ですね。それから, 昨日言いましたけれど, プロセスというのは完全に決めるのじゃなくて, オペレーションしながら細かいところはスケジューリングしてゆくと。そんなの誰か休むかもしれないのに工程期間をビシッと決めるわけじゃぁない。そういうスケジューリングという概念もみんな知らないと。これは無理もないことで, 通産省のプロジェクトリーダーの標準カリキュラムを見て頂くと分かりますけれど, そんなこと一切書いてないですね。非常にレベルが低いです。まあ, あと, レベルが低い例で言うと, その通産省のカリキュラムは品質管理のところを, 兼子先生なんかはやってるかもしれないが, 統計的品質管理ばっかりやっている。個別品質管理を全

然載せていない。それから, テスト技術の所に, テスト項目設計基準書に従えということが一言も書いてありません。まあ, そういう意味で我々のこの学会のレベルがまだ, そこのレベルにしかいっていないと。成熟度が 1 くらいしかないと。だから, 僕は, 標準工程という概念と個別工程という概念は全く矛盾しないと思います。

司会 ありますか?。はい。

坂本 さすが君島さんで, まったく私も同意見です。社内に, 標準工程があっても, それがそのまま現場に活用出来ていない, ということなんですけれども, それは, 今, 君島さんが言われたように, 社内の標準工程と, 現場で使う個別の WBS を落とし込んでやるというのは違うんで, そこの間のギャップを埋めるという作業を, 我々としてしていなかったという反省があったということです。で, ちなみに, 我々の部門で使ってる標準は, バイブファイル 1 冊ほどの分量があるんですけれど, これは 6〜7 年前に私が中心になって作ったんですが, 私の部下には「あれを現場では無視しなさい」と言っています。勿論それには ちょっと前振りがあって, 良く勉強して無視しなさい, と こう言っています。その通りにはいかないでしょう, と。現場でうまく使うことが大事だということを言ってます。

司会 他にありますか?。良いですね。じゃあ, 次, 堀江さん, どおぞ。

12. サクセスストーリが広まらないが …

堀江 (日本電気ソフトウェア) 私も DIPS だったんで, 桑名さんに伺いたいのですが。サクセスストーリを作れという話についてなんですが, 先ほどのメタの話とも若干関係するかもしれませんけれども。私が DIPS 関連の仕事を終わったあとで, 小さな 7〜8 人で 1 年弱くらいのプロジェクトを頼まれました。で, これが凄いもので製品をリリースしたら 1 件もバグが出ない。リリースどころか, 8 ヶ月目ぐらいから全体を集めて各種のテストをしましたけれど, 全然バグが出ない。皆メンバが, 遊んでいるという状態で, 非常にラッキーなプロジェクトだったんですが, これも確かにいろんな標準を作ってやってたわけですけれども, これがどこにも広まらなかったという一例があります。

また, これはつい最近当社での十数億のプロジェクトの事例ですが, 指揮されたのは, 純粋のビジネス系のアプリケーションの方で, 初めて 50〜60 人を使ってやらなければいけない。「どうしたら良いだろうか?」と相談がきまして「メンバーは?」と聞いたら「主力になって動けるのは 2 年生しかいない」という返事。他は主任と課長がボロボロといるという。まあ, 昔の空洞化なんて言っていましたけれども, そのころの部隊で, あとは協力会社の人が一杯いるという。「まあ, こういうふうにしたら」という幾つかアドバイスをしてやりましたら, 途中色々と私もフォローしましたけれども, バッチリ成功して。で, 書け書けと一生懸命ものを書かせたんだけれども 全然社内へ広がらない。オムロンさんも, 今, 一生懸命頑張ってらっしゃいますけれども, これを全事業部にやったら大変なパワーがかかるだろう。例えば, こういうソフトウェアシンポジウムのような場で成功事例を発表することは, それはそれで意義があります。が, そうすることと社内から注目をされ, 広まって, 自分のところの全体レベルの向上に結びつくということはまず無いと思うわけです。そういう意味で, サクセスストーリは本当に企業内で, 一事業部を成功させて全部に広げるにはどうしたら良いだろうか? というとこあたりをお伺いしたいなぁと。

桑名 先ほどのサクセスストーリの話ですが, 詳細は NTT ソフトウェア研究所の長野が書いています [4]。それはどういうプロジェクトかと言いますと, 交換機ソフトの開発環境を構築したり, 交換機のソフトウェア自身を改良するというもので, もう 4〜5 年前からスタートしていました。サクセスストーリーが広まるか広まらないか? ということに関してですが。「何故 うまく広まっていったか?」という質問に対しては, トップダウン管理の問題に帰着すると考えています。各プロジェクトの上にいるマネージャがどういうアクションを起こすかだと思うんです。交換機ソフトの開発環境やソフトウェア開発自身のプロジェクトの例で言いますと, 数千人という開発者を持つ組織の長が, 組織の中で成功したこと (スモールスタートでやって成功したこと) に非常に理解を示して下さって「じゃあ, みんなでこれやろう」という大号令が上からかかっちゃう。意志の流通をするだとか, ドキュメントや情報の共有とかというのをそのサクセスストーリで用いた方式に習ってやっちゃうんですね。

それまで, 他の部署の人達は良いサクセスストーリがあることをみんな知らな。しかし一旦それを知ったら「じゃあこれ真似よう」というので。しかもマネージャクラスが, その上から「これをやって成功したから」という話を聞いていますので, それに刺激されて, みんなでやっちゃうわけです。素直ですから。そこがうまくいく点なんじゃないかと思います。要するに, 言いたかったのは『トップの意識の問題。理

解のあるマネージャが, 如何にそのサクセスストーリを取り入れて, 社内に広めるかっていう意思決定の問題』かと思います。

あと, これは最近のグループウェアなんかにしても, 実は, まさにそれがあてはまると思っています。一例を挙げますと“ロータスノーツ”という製品があります。会場の方で, お使いになられている会社の方, いらっしゃいますか?。え〜と, ちょっとおききしたいんですけれど, 全社でお使いになられていますか?

会場　はい

桑名　そうですね。

実は, 組織にいかに浸透させるかということでは, これと同じように, これも 4〜5 年前から グループウェアの世界で「何故グループウェアがうまくいかないのだろう?。今だに電子メイルばかりじゃないか」という議論があります。で, 色々分析していってわかってきたのは, その管理の問題であるということです。良いものは, とにかく, トップが判断して全社的にいれなきゃならないと。そうすることによって, それをみんなが使うことによって, ドキュメントシェアリングも進むし, さらに新しいツールが入ってきて, それによってプロセス/作業工程が改善されてゆくという話になると思います。そういう点からも, まさにこれからツールを導入することに関しては, トップの判断が非常に重要になってくると思います。

堀江　私も, 全く同意見です。私が反省してみると, 上にもう少しアピールする努力が足りなかったかなぁ, と思っています。トップから落としていかないと やっぱり広がっていかない。

ただ, ツール類は 比較的簡単に広まってゆくけれど, 本当に プロセスといったことが本当に広がってゆくか? っていうのは, ちょっと見えない所がありますね。

これは, 某社の OS なんか作っている事業部, そこはもう ISO を取りましたけれども, そこはバッチリとした管理標準を作って使っています。で, 他の事業部も「是非 欲しい」という。そこから, どちらかというとアプリケーションを作っている事業部が持って行きましたけれど, ここは全然死んじゃったという。これは, やはり OS を作っている部隊とアプリケーションを作っている部隊の仕事の特性の違いを考慮しなかったことによるのかなぁと思っています。アプリケーションの部隊は, 個々に仕様を区切って みんなバラバラでやっていますが, OS は 全員一団となって走って, ゴボっとくっつけますから。そういった特性を 全然やはりトップが理解しないで, ポンと持っていっちゃったという。プロセスをどう持ってゆくのか, 最終的には 桑名さんの言われた, 「トップがちゃんと意識して そこから引き上げてゆく」という改善をしてゆくようなことをしないと, プロセスを広めるというのはちょっときついのかなぁ, という感じがしています。

桑名　ツールに関してなんですけれども「ツールは広まるけれども」という話があったんですけれども, 僕は, ツールの中に実はアイディアが隠れていると考えています。

ロータスノーツを例に説明しますと, 実はツールが ただ出来たわけではなくて, あれだけのツールが出来たところには, もう十数年前からグループのデータシェアリングに関する研究成果が反映されているわけなんですね。

で, 今も メタな議論が出ましたけれど, “メタな議論”とは どういうことか, ということを言わさせて下さい。それは, もともと最初 色々なケースがあります。そのケースに対して次の段階としてオブザベーションという形がある。そのオブザベーションで, 何が同じなんだろうと考えて, その次に, 実はそれをさらに抽象化するアブストラクションという段階があって, さらにそれが上に上に進むにしたがって, 例えば理論が出来たり, 科学が出来たり, 法律が出来たりする。こういった一連の, 何か 世の中不変の考え方というのがあると思っています。で, そういう 抽象化されたレベルの上がった考え方というのが, 実はツールの中に埋め込まれているのではないか。また, そういうツールを使っていかなければいけない, と 私は思っています。

坂本　我社ではそんなに格好良く発表している以外の, 非常に泥臭い話があって, そんなにうまく行っているわけでは決してありません。で, 我々は, 特にホスト系の先進的な会社の方の後を一生懸命追いかけている, という認識をしているんです。で, その追いかける中でですね, 何をどういうことで心掛けているかと言うと, 要するに, 知識の移転, 技術の移転なんですね。技術の移転というのは, 何なのかということを考えて, それを如何に効率良くやるかっていうことだと思うんですよ。で, 技術の移転というのは放っておいても出来るわけじゃないんで, もし放っておいて出来るんなら, 学校なんて, 大学なんて成り立たないんで, それは, それなりのパワーが要るんだということを皆認めていると思うんですね。で, それは社内であっても同じであって, 社内でそういう技術の移転, 知識の移転をするのには, それはそれなりの経営資源

が必要だということを経営トップの方にわかってもらうことが必要だと思っています。で, 私が上の方を説得しているのは, ソフトウェアがおかしくなるのは色々あるんだけれども, やらんとあかんことは4つありますと言っています。それは,

(1) 標準プロセスモデルをきちんと教育すること
(2) 個別のプロジェクトの計画をきちっと立てられる, それを立てる技術も身につけること
(3) その状況を visual 化するためのメトリックスをきちっと現場に落とすこと
(4) 以上のようなことが, 現場の人達, 皆に分かるように技術移転をしないと駄目なんですが, そのためには経営資源が要ります

と。で, これは, 本当はお恥ずかしい話なんですが, この1〜2年, 初めて社内でそれだけを行う専属部隊が持てるようになりました。という状況で, 少しづつまわり始めたかなぁ〜というのが現実です。
司会 良いですか? じゃぁ, 酒匂さん。

13. 継続的な reward のシステムが必要では?

酒匂 (SRA) よその部署に, どう展開して行くかといった話を考えてプロセスの話を聞いた時に, 確かに私も 坂本さんがおっしゃったように, プロセスは標準的なガイドラインとして用いるのは構わないと思います。で, それを如何に浸透させるかなんですけれども, 私は 日本の状況が具体的にどうなっているのか分からないですけれども, systematic な reward[11]。個々人に対するですね reward のシステムを考えても, もうそろそろ良いのじゃないかなぁと思います。reward は, 現在は, 個々人のやる気の問題とかに とどまっているようです。ですから「自己研鑽の度が足りない」と言ってみたり。また, 管理者が「お前らたるんでるから」と判断すると 関係者を集めて「みんな頑張ろ〜!」とかやっているわけですけれども, 例えば良く出来る人ってなにかというと, 自分自身に如何に reward を出すか? ということを良く知っている人だと私は思っているんですね。要するに, 自分自身がこれをやることによって, どういう見返りを得るかということを良く知っていて, で, それに対して自分に対して, ある区切り区切りで「私はこれだけのことを達成した」とかですね, もしくは, それ自身を維持することに喜びを感じているという。でも, まあ, 人間だから, いつもそういうわけになかなかいかなくて, 日々会社の中で仕事をしていて, 例えばプロセスというものがやってきてですね, なんかやれという。で, 勿論, それ自身は良いことかもしれないけれど, それをやることによって, 自分自身が日々, なんかこう進んで行く実感が, あるいはあなたは進んでいますよというような, 要するに reward みたいなものがくると非常に良いんじゃないかと思います。で, 昨日のセッションでも会場からコメントがありましたが, 一度, 一生懸命 体系を整備しても, 何年かすると, 段々志気が下がっていって元の黙阿弥になって, もう1回全部再構成し直すといったことがあると思うんです。けれども, それは継続的な reward のシステムが無いから, 最初は要するに, 生産性が向上すること自体が, つまり, 何か, 要するに, プロセスを整備すれば, 生産性が, 一瞬でも向上したら品質は向上しますから, それ自身が reward になって, その瞬間はうまく行くんですけれども, それが当たり前になってしまうと, 他に reward がないですね。もし, product(汎用製品)を作っている会社ならば「product が良い」「社会的な評判が良い」ということで 社会的な認知度が上がる。そして, そのことで 個々の社員が reward を感じるということはあると思うんです。しかし, 受託が中心のソフトウェアハウスの場合なんかは, それを 維持するのは難しいんじゃないかと思っています。というわけで『精神論に陥らないシステマティックな rewarding システムはどうしたら良いでしょうか?』ということに関してご意見をお聞かせ頂けると。
司会 久保さん, どうでしょう。
久保 あまり参考にならないとは思いますけれど。今, 私は富士通のシステム本部というところにいますけれども, で, ISO の認証を取得する動きだとか, それから CMM のモデルをカスタマイズしているんですけれども。そういうものを出して, 今, 階段上がりなさいとかいう, 基本的に嫌がっているのをやらせているっていうね。で, その時に, やっぱり, なんちゅうんですかねぇ〜, reward ではなくて脅迫かもわかんないんだけれど, 要はね, 要するに, 認証って, もらわないと商売からシャットアウトされるぞ, という脅迫。ですから, そういう勲章, 認証を欲しがるっていうのを, なんとか作り出している。ですから, それなんかも reward になっているんじゃないんですかね?。
司会 それはネガティブな reward ですよ!
酒匂 例えば, ISO の認証を取るという目標がある間は, それに進むこと自体が1つのドライビングフォースになりますから良いんですけれども, もしそれだけだとすると, それをとっちゃったらどうなるんだろう? と疑問に思うわけです。最初は, 脅迫で始めても

[11] reward : 報酬, 見返り

良いと思うんですけれども，でも，企業は一過性に商売をやっているわけじゃありませんから，むしろ大きなドライビングフォースではなくて，日々のまあ，勿論，個々人によって 何を楽しみ，もしくは やる気と考えるかというのは，全然ちがう問題なんですけれども，それにしても，何かを導入しようと思った時には，そういった，如何にそれを目に見える形で評価してやるか? ということと表裏一対だと思うんです。

で，その表裏も，押しつけ的な評価じゃなくて「これをやることによって，確かに自分が何かをやり遂げていっている」といったものが必要になるんだと思います。同じ様な例は，多分，ツールだとか環境にも当てはまっていて，そもそも使う気にもならない環境だと，大体，進化のしようも無いし，使う気にもならないツールも進歩しない。同じように，使う気にもならないプロセスだと，そもそも標準化と決めておいても，それ自身は，標準化を決めた段階が一番良くて，で 段々と使えなくなって腐っていくということになりますね。ですから，最初の段階で，ある程度使う気になるようなシステムであって，なお且つ，人間に対してもそれなりの配慮があれば，放っておいてもうまく行くんじゃないかと。これは一種の理想論で，書生の書生論というような形になってきましたけれども(笑)。で，私が「システマティックな reward システム」と言ったのは，そういったような意味です。

久保 システマティックな reward system と言っちゃえば，1つ，やっぱり日本の TQC というのは，それがシステムになっているんじゃないですか?。好き嫌いはあるかもしれませんが。結構，それ，うまくシステムにしたという例じゃないかなぁ〜。だから事例を探すとすれば，僕は，まあ日本の中で，デミング賞なんかを受けていて，且つ，外に対してメッセージを発信しているようなところは，結構そういうような面で，うまく上から下までやっているってゆうか…。という例になると思いますけれども。システマティックと言われると，そんな感じかなぁ，僕は。

兼子 大学で学生を見てると，自分でどんどんやる奴と，この間 学校に来たの何時だっけ? みたいな奴に分れるんですね。そして「良くやっているね」という奴を良く観てみると，その日に やるべき事というのを自分で決めて学校に来ているんですよ。で，うちの研究室では，4年生が配属されると，初めの2週間ぐらいはインストラクションをしますけれども，それが済むと研究テーマを列べて「好きなの持ってけ」と言って，後は卒論まで何も指導しないんです。勿論，学生が何か勉強したいって言ったら 「この本読め」とか「一緒に勉強会しようか」とかなるし，月に1回はゼミがあって研究の進行状況をチェックしてますけれど，基本的に手取り足取りというのは一切無くて，テーマ丸投げして おしまい なんです。そうすると，受験勉強的なことをやってきた者は，いきなり途方に暮れちゃう。で，そうでない人っていうのは，毎日来て「今日は，自分はここだけやる」と言って。で，終わったら「よ〜し，酒呑みに行くぞ」とか言って帰って行く。だから，必ずしも システマティックではないんだけれど，自分自身に対して 達成可能な小さなターゲットというのをきちんと作れて，それを継続的に与え続けることが出来れば，実は，「半年かけないと」「1年かけないと」といった long lenge の前に，短いミッションをやり終えたという reward って言うのは与えられるんじゃないかな。で，それを，組織として与えるか，あるいはそういう仕事の仕方というのを身体で覚えてもらって，自分で self マネージメントしてもらうのは?。多分，僕は self マネージメントの方がはるかに効率良いし，そっちの方が麗しいと思っているんですけれども。

酒匂 私も 全くそう考えていて，多分 システマティックと言ったので，組織としてどうやって，人を盛り立てるかとか，さらに管理的な側面で誤解されたかもしれませんけれども。私も，今おっしゃられたように，個人が自分自身の細かい目標を日々設定してゆくのが良いと思っています。けれど，じゃあ それに対して，ただ漫然と期待しているのかということですよね。つまり，管理する側(というとまた誤解を招くかもしれませんが) がそれに対して，じゃあ 何が出来るか? もしくは，例えば自分の小さい目標設定をしたいと思ったり，色々な細かい試みをしたいと思った時に，それを受け入れられるだけの受け皿があって可能になるのではないかと思うわけです。「じゃあ私はちょっとこれでやってみよう」とやった時に「確かにあなたこうやりましたね」という形で，小さなものでも良いから 何か認められるようなものが，(今 具体的なやり方は思いつきませんけれども) あれば良いなぁと思いました。

で，TQC のことを久保さんがおっしゃって。私も TQC のイメージは持っていましたけれど，あの場合，あれは TQC の結果作られた製品が社会的な貢献をするというようなことが，で，しかも個々の職場の中でお互い高め合うような運動があったと思うんですね。と，すると，逆に私は問いたいのですけれども，今 ここで話題になっているソフトウェアプロセスを TQC でやれば良くなるんでしょうか?

司会 それはちょっとうまく行かないんじゃないんでしょうかね。時間も迫っていますので, 次に行きます。

14. 失敗をきちんとまとめることこそが重要では?

丸田 (東京電力) 実は, 私共の会社では, 今まで, 事務系や業務系のソフトウェアは何年かの歴史があるんですけれど, 制御系という 今までほとんど経験のなかったドメインに対して, 新しく自分達で開発してゆこうという取り組みを, 私をはじめとするグループで始めようとしているところです。で, これから新しく始めようということを逆手に取るという意味もあって, 本格的に着手する前に, 色々な標準を調べたり, 事例を学んだり, マニュアル作りとか環境の整備ということを始めているところですけれども, 兼子さんの話の中にもありましたように, 多少なりとも経験を持った人は, 例えばメタな議論をした結果であるとか標準みたいなものを見た時に「ああなるほど」というふうに, 行間を理解するような能力があるようです。ところが 比較的経験の浅い人とや, 今回 私達のプロジェクトに初めて入る人というのは, そういうことを一生懸命 教えようとしても, なかなか伝わらないことが多くて, 今, そういうことでも苦労しています。で, 私はかつて 別の部署でソフトウェアを作ったこともあって, 成功もあるのですが かなりの失敗もして来ました。で, そういうネガティブな経験というのが, きちんとまとめられたり, 外部に発表されるということが どうも少ないような気がするんです。例えば, プロセスだとかメタな議論を始めると, どうしてもサクセスストーリのことを広めようとか, うまくいったことを説明しようという方にどうしても向きがちで「こういうことを疎かにすると, こんなひどい惨めな結果に終わるんだ」というような, 例えば 標準書 + 惨めなサブ読本とか, そういったやり方で一種の脅しになるのかもしれませんけれども, うまく 標準ないしプロセスを理解させるとか, まわりに広めさせるとか, そういった, ちょっとえげつないやり方でやることは効果は無いのか?, あるいは, そういったことをやっている事例とかを御存知ではないでしょうか?

坂本 そんなに立派なことやってないんですけれども, 個別のプロジェクトというよりも, 先ほど私が言った 3 つ目のキーワードのメトリックスなんですけれども。何故 メトリックスをやるかと言うと, それは, 技術移転をし易くする為に, メタにしておけば技術移転は早いと思っているんです。その為に, メトリックスを計測して, こういうことをやればこういう結果が出ますよ, ということを言っているんですね。例えば, 各開発の工程毎の比率に対して, そのプロジェクトの成功率との相関をとるとかですね。で, それは結構, パッと, グラフにして見せれば。例えば, 横軸に review 比でも良いです。レビュー比とフィールドバグの数だとか, レビュー比と開発効率だとか。というふうに, まとめていっているんですね。で, それは結構, 説得力があります。何をすれば良いかというのは, 今までそういう概念の無かった人にも, わりと早く技術移転が出来る方向だと思っています。

久保 極端な言い方をしますけれども, 教えるというのは止めた方が良いと思いますね。それから, 勉強したくなる状態を作るというか。だから, やっぱり, 失敗させるのが一番良くて, その時, 失敗した時に, 如何に学習に結び付けるか?。そこじゃないだろうかと思うんですけれど。だから, 教育議論って, みなさん好きですけれども, 大抵, 何を教えるのか, ということばかり議論をしているようにですが, それは間違っているんじゃないだろうか, と思っています。むしろ, 学習することを, どうやってその気にさせるか? というふうに考えるのが正しいと, 僕は思っていますね。極端ですけれども, 教えるなんていうのは思わない方が良い。

兼子 物を作っている現場で QC サークルみたいなの, ありますよね。ああいう所で, 新しく取り込もうと思ったら, サクセスストーリを広めるっていうことをやるんですよね, 一般的には。だけども, もう少し上のレベルのエンジニアぐらいが, きちんとした技術標準を作ろうだとか, あるいは自分達の仕組みについて改善しようといった時だとかには, やっぱり失敗を書くんですよ。というのは, サクセスというのは再現性が無いですけれども, 失敗は再現性があるんですよ, 多分。だから, 何で失敗したのかを見ると, 自分達が何でどこが弱かったのかっていうのが判る。それは多分, 2 度も 3 度も同じ失敗をしてるんですよね。だから, プロセスをインプルーブメントをしようとか, どんなムーブメントでも良いんですけれども, 何かムーブメントを誰かがワッと起こして仲間作りの時には, サクセスストーリが要るんですけれど, それを本当に形のあるものにして強固なものに組み上げてゆく為には, 失敗を解析してゆかないと。サクセスストーリを解析したって, 多分, プロセスなんか良くならないんじゃないかなって思いますけれども。

司会 僕も 1 つコメントがあるんですけれども。一番良い, 一番 理想的な方法は, やっぱり "失敗を出来る環境を作ること" だと思います。なぜ大学教育が必要かというと, 大学教育というアイソレートされた環

境の中で失敗しても良いですよという環境の中で技術を学ぶことが出来るという所が，理論を学ぶことが重要だと思います。で，それが実社会で出来るか?というと，なかなかそうはならないと思います。良い会社は，そういうことをやってきたし，やってこれた。で，これからどうか判りませんけれども，出来なくなるでしょう。そうしたらやはり，その，他の失敗例から学ぶ。こういうSEAの会議で，お互い失敗例を あまり隠さずに交換を出来れば良いなぁと，僕も思います。賛成です。

で，次，時間もおしてますので，玉井先生。

玉井 (東京大学)　「標準化を推進する」議論で，例えば兼子さんの議論が典型的だと思うんですけれども，その根拠は「平均的な人間を」と。で，その“平均的な”と言ったときに“出来の良くない人も含む”というのが前提になっているのでしょうけれども，そういう人達を集めてなんとか生産をするときに，やっぱり 標準的なものは要る，という議論ですよね。

一方，「標準というのは役に立たない」という議論で，さきほど少し出てきていたのは，個々の現場のspecific な問題なり作業に対して，標準あるいは共通化したプロセスというのは やっぱり役に立たない，というネガティブな意見があったと思うんです。

けれども，どちらの考え方も非常に良く似ていると思うんですよ。どこが似ているかと言うと，ある作業にとっては，少し共通化された少し違う分野のプロセスというのは参考にならない。基本的には，他から学ぶことは出来ないと。あるいは極端に言えば，自分自身のインスタンスで過去でやっていることからも学ばない，という考え方ですね。これは『個々の人間が，学習してゆくとか，知識が増えてゆくとか，あるいは知識が移転してゆくとか，能力が向上するとかということがあまりあてにならない』ということが前提になっています。

で，昨日のAlan Davisのレミンジニアリングは「いろんな技術，ファッショナブルな技術というのは現場じゃ，役に立たないよ」と主張していて，非常に説得力があるんだけれども。しかし，ある意味では 非常にマッチョなというか，反技術的な，あるいは反知性的なものが明らかに根底にあるわけです。そこまでそういわれちゃうと，ちょっとインテリは もうどうしようもなくて，無言のまま引き下がらずを得ないところもあるんだけれども。でも，どうもその，標準に賛成の立場も反対の立場も，根はそこでつながっているようなところがあるように見えるんですけれど，それに対して何かコメントがあれば。

司会　皆さん考えていますんで，時間稼ぎをしますけれども。多分ですね，今，玉井先生が言われたのが，標準の良いところでもあり，功罪であるところだと思うんですよね。例えば，私の元居たIBMでも，非常に沢山の標準がありまして，会社に入った時に最初に読まされたのが，そういう，どうやってプロジェクトをやるかという標準なんですけれども，それは何も知らない人のためのものですが，それまで学生としてアルバイトで かなり長いプロジェクトを何回も経験してきた私でもすごく学ぶところが多かったです。で，そういう意味では非常に役に立つ。この会社でどうやったら仕事が出来るのかということを学ぶ。それは，一般的な generic なノウハウを持って入ってきた人間で，それをこのシチュエーションで何をしなきゃいけないか? というのを知るために役に立ったと。ところが，それが，例えば5年経って，同じ物が同じように役に立つかというと，まったく無いでしょうね。だから，そうなってくると，今度，逆にこうしたいのに，ここではこういう特殊な処理をしないと，マネージャあるいは社長さんの所に行って説明しなければいけない，というような足かせになってくるわけです。そういう面と2つあると思いますね。それは，もうどうしようもなくて，バランスの問題でどこに出来るかというのは，やはり企業としては，非常に抽象的なプロセスを取らざるを得ないし，個々の事業部とか，さらに個々の部門になったら，かなりのspecific なプロセスを定義せざるを得ないと思います。で，それをうまく活用してやっていく。で，それがですね〜，その，自分が日本人で悪いなぁと感じたのは，そういうものが，一度与えられると，与えられてもう変えられないものだというふうに，なんとなく思い込んでしまうわけですね。で，教えられたら，そのままやらざるを得ない，と決めつけてしまう。そのあたりが，アメリカ人と一緒に仕事をすると違っていて，「こうなっているけど，こういうふうに話を持っていて，誰かさんに話をして，で，承認してもらえば良いじゃないか」という話になって，誰かが出て行って話をして，とにかくそれで良いっていうふうにしてしまうという。そういった臨機応変さというのは，我々にはちょっと無いなぁという気がしました。そして，そのへんも非常に大事なもんだと思います。すごく時間がおしていますけれど，誰かありますか?

久保　なんだろう，こう，標準と言葉が 不適当なのかなぁ〜，という気がしないでも無いですけれども。やっぱり，経験を知識にして，で，どんどん知識が硬くなってゆくと，本になったりして，最後はツールに

なったりする。で, 標準なんて言われているものは, ツールにまでいかないんだけれど, やはり経験から学んだことを表現しているんじゃなかろうかと思うんですが。

司会 そうですね。

久保 だから, 標準というと, すごく …

司会 権威があるみたいにね …

久保 なんか, そうなっちゃって。

司会 そうじゃないですよね。

久保 だから, "知識の再利用の 1 つの形態" というふうに思っちゃったほうが, 気楽だと僕は思うんですけれども。

司会 僕もそう思いますね。最近, 非常にそう思っているんですけれども, 桑名さんがちょっと言っていた, 知識の話を最近好きでやっているんですけれど, そういう観点から見ると, 標準というのは, それは企業の標準とか組織の標準とかそういう意味ですけれども, そういうものを今言った知識という観点で見ると, 知識を持った色々な人が, その会社にはいるわけですけれど, 個々の人はエンジニアリングを学ぶわけですけれども, 企業とか組織というものが学ぶ為の手段として, そういう形に結晶/結実してゆくっていうことが非常に重要だ, と 僕は思います。

兼子 標準だとか, なんか, その類ですよね。まあ, ドキュメント化されていてもいなくてもなんでも良いんですけれども。それは, やはり組織が何か目的を達成しようとする集団として, 知識を蓄えるための枠組みというか入れ物。で, 少なくとも, こうやればうまくゆく, という類の積極的なものか, 少なくともこれでやれば大失敗しないよ, というネガティブなものかもしれないが, それを貯め込んでゆくための入れ物じゃないかなぁと。で, 標準というのは, 改訂されていることに意義がある。だから例えば ISO だって 5 年に 1 回改訂するというのが決められているわけだし, 企業の標準もやっぱり改訂されてなかったら意味が無い。だから, 改訂されていない標準を使っているっていうことは, 標準どおりに仕事をしていないか, つまんない標準で誰も見ていないかのどっちかなんだから。要するに, そんなものは止めちゃえば良い。で, やっぱり標準の良いところというのは, 「あいつが」とか「そいつが」という話を, 組織が持っている知識, 不十分な組織が持っている知識に還元することが出来る。で, 必ず, 個々人というのは標準を越えているわけで, やっぱりあいつがやるとうまくゆく, あいつがやるとどうもうまくゆかない, というのがあるけれど, それはやっぱり標準に反映させてゆくというプロセスの中で組織が知識を蓄えてゆくことになるんじゃないかと思うんですけど。

司会 時間になりました。最後は当たり前の結論になってしまいましたけれども, 長い間, 皆さん積極的に議論に参加して頂いてありがとうございました。最後に, パネリストの方々に感謝の意を表したいと思います。拍手をお願い致します。どうもありがとうございました。

参考文献

[1] E. ヨードン (松原 訳), ソフトウェア管理の落し穴, トッパン

[2] M.C.Paulk, B.Curtis, M.B.Chrissis, Capability Maturity Model,Version 1.1, *IEEE Software*, July. (1993), pp.18 – 27

[3] G. M. ワインバーグ (大野 訳), ワインバーグのシステム思考法, 共立出版

[4] NAGANO Hironobu (長野 宏宣), Creating An Exemplary Organization, *American Programmer*, Jan. (1994), pp.33 – 41

[5] G. Olson, J.Olson, "User-Centered Design of Collaborative Technology,", *Journal of Organizational Computing*, Vol.1, No.1, (1991)

[6] 君島 浩, 障害 1 件だけで根本原因を究明する方法, 第 12 回 ソフトウェア信頼性シンポジウム論文集, (1991), pp. 24 – 27

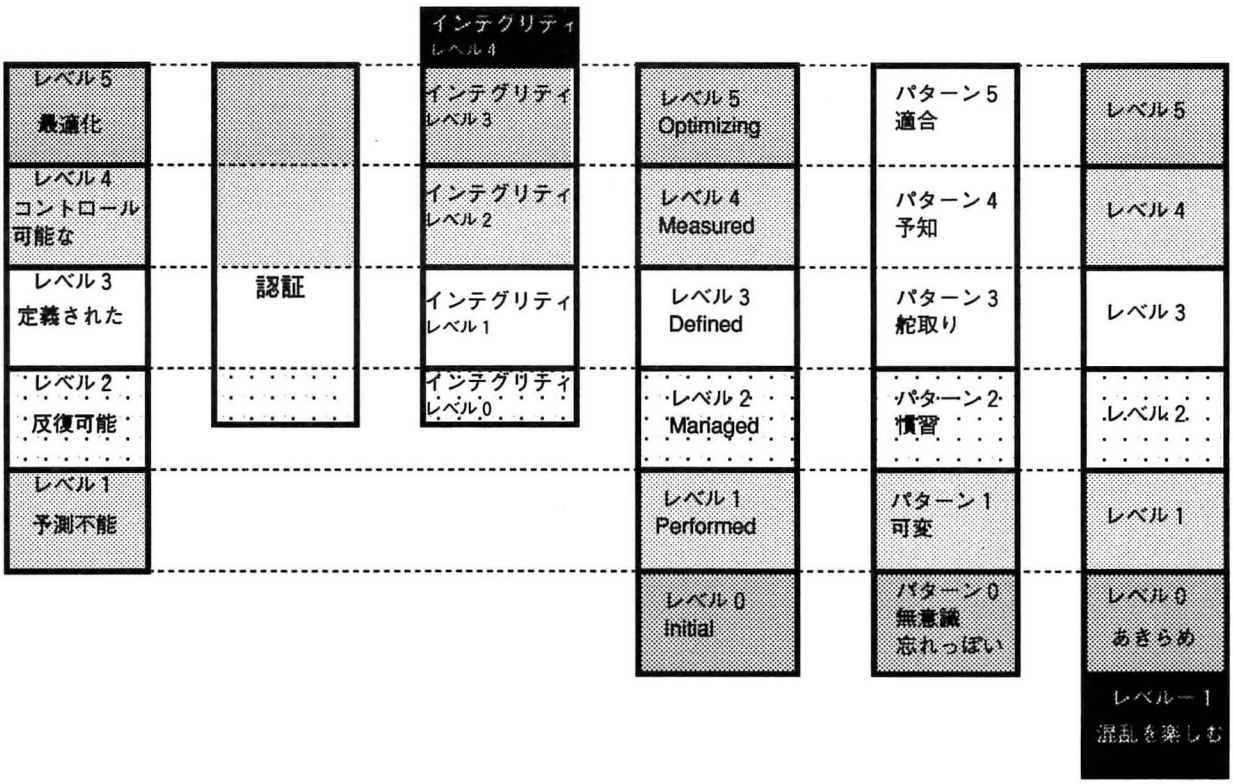
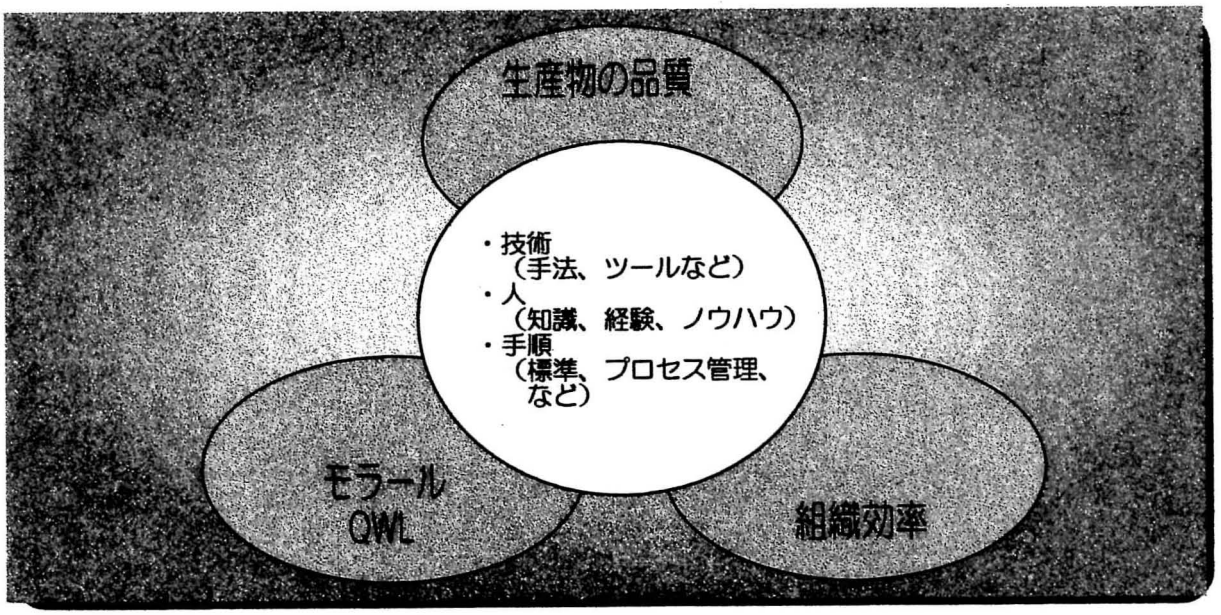
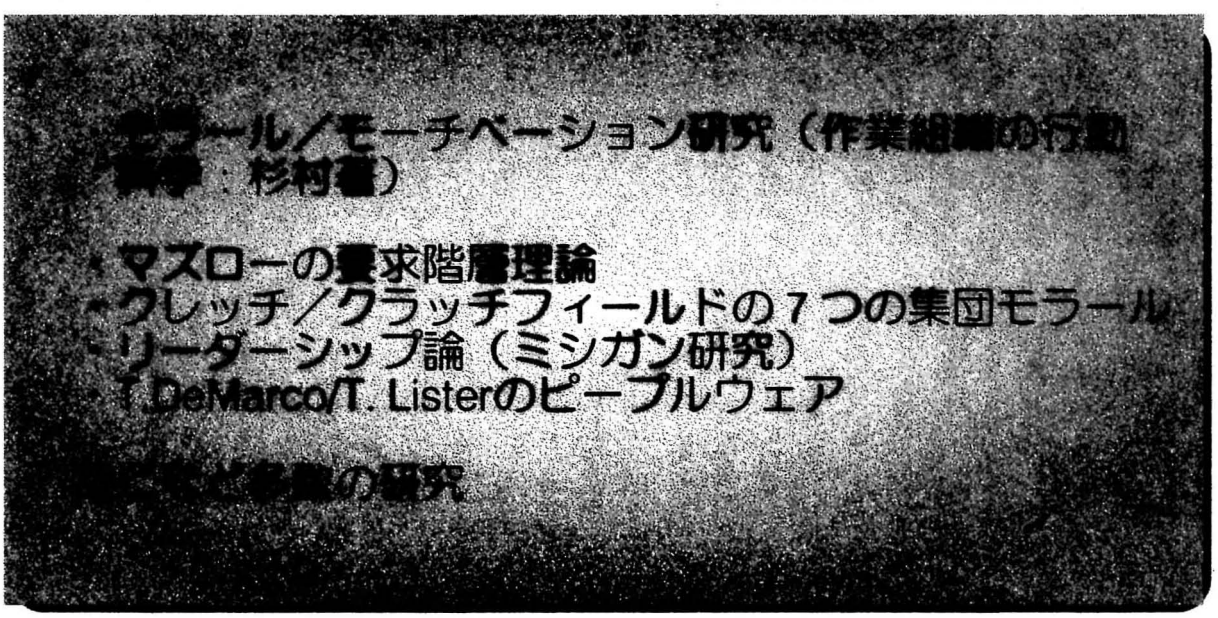


図 1．レベル分けの比較 (坂本)

プロセスの管理／改善：３つの重要な目標



モラール管理：QWL（Quality of Working Life）論など



SS'94

ポスト資本主義社会とソフトウェアプロセス管理

- 知識主義社会における専門家
- 組織における資産：専門家集団、知識、経験（シンフォニーオーケストラ型組織）
- 知識は「道具」「工程」「製品」に適用されたが今後は、ダイナミックな性格を持つ「知識」への適用
- 開発の現場（多種）：問題同定、問題解決型タスクへ移行

おわりに

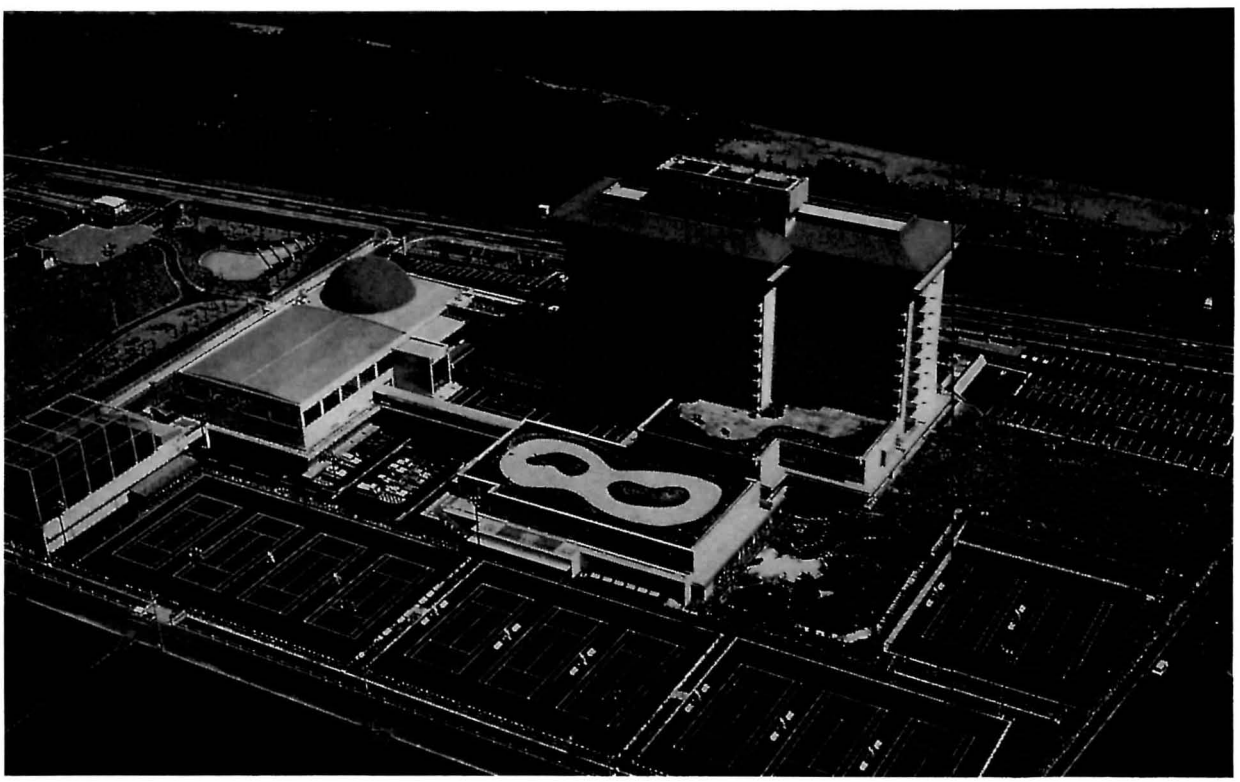
- ポスト資本主義を考慮すると、ダイナミックに変化する知識（労働）／組織構造に対して、現在までのソフトウェア品質保証の前提／考え方が適用できるのか
- 効率から効果へ
- ソフトウェア品質特性は現在のままでよいのか
- 新しいSE研究のアプローチ
- 考えるよりも適用、評価する

ソフトウェア・シンポジウム'95 にむけて

中野 秀男 *
(SS95 実行委員長)

今回, 実行委員長をやります大阪大学の中野です。SS95 は琵琶湖のほとりの眺めの良い所で行なうことになりました。ローカルな設定はオムロンの坂本さんの部隊を中心にしていますが, 全体的な話しは SEA 関西が全面的に支援するすることになりましたので, その統括責任者という事で私が実行委員長の肩書で動きます。参加者を集めることと会計的な事が大きな仕事ですが 今までにも SS では, ツール担当もチュートリアル担当もプログラム委員長もまがりなりにもやって来ましたのでその経験を生かして無事勤めたいとおもいます。わたし以外の両プログラム委員長や各担当の方からはりっぱな所信表明がでていますので, それらの事が実行出来るいろいろな環境作りをやらないといけないと考えています。

会場は琵琶湖大橋のほとりのリゾートホテルです。ホテルにあるプラネタリウムの部屋もセッション会場になりますので家族で来ても楽しいシンポジウムになりそうです。体育設備も揃っているようなので, 今から来年度の予定に入れて貰えればと思います。



* 大阪大学 工学部

ソフトウェアシンポジウム'95 論文の書き方

佐伯 元司 *
(SS95 プログラム委員長)

ソフトウェアシンポジウム'95 では, 技術論文および経験論文を募集しております。

技術論文は, ソフトウェア開発に関わる新しい技術について論じたもので, 提案している技術の本質が独創的で, 技術的にも矛盾がないものでなければいけません。

経験論文は, ソフトウェア開発における自分の経験をまとめて報告するわけですが, その経験がある程度の一般性を持ち, 他のソフトウェア技術者にとっても有効なものでなければなりません。

どちらのタイプの論文も, まず他人が読んで理解できるものになっていなければなりません。自分の持っている知識を他人も持っているだろう, 自分が抱いている疑問や問題をも他人も同様に抱いているだろう, などという他人が自分と同じ環境にあるという前提を置くことは, 論文をわかりにくくする大きな原因の一つです。技術論文, 経験論文, その性格は異なりますが, 論文を書くときの出発点は同じです。まず, 書き始めるにあたって, 自分が主張しようとしている問題は何なのかを明らかにしましょう。つまり, 現状がどうなっていて, その中のどこに問題点や疑問を感じているのかを, 自分自身でちゃんと把握する必要があります。これができていないと, 航路を見失った船のように, ふらふらとした, 何を言いたいのか読み手には理解不可能な論文になっています。技術論文では, 提起した問題に対して自分はどのようなアプローチや考え方をとったか, それらのどこが新しいのかを明確に述べ, 最後にその効果の評価, どこまで自分の考えていたとおりになったか, またそのとおりにならなかったとしたらその原因は何かを, 自分の主張として明記していきます。経験論文では, 論文で述べている自分の経験からどのようにして問題点や疑問点を発見したか, あるいはこれらを解決するのに自分の経験がどのように役立つのかを明確にしなければなりません。論文というからには, 技術論文/経験論文とも話が論理的に展開されなければなりません。話の展開が論理的に飛躍している論文をよくみかけますが, 事実とそれに関する自分の考え (そこから推論・推測されること) をきちんと区別し, 無理がないかどうかをよく吟味しましょう。また, 書いた論文を投稿する前に, 他人に読んでもらいましょう。他人からのコメントはわかりやすい論文を書く上で必須です。

論文はひとつの芸術作品です。自分がやったことをひとつの作品として, 他人にわかるように公開しない限り, 何もやっていないことと同じです。論文の書き方によって何をやったのかが評価されます。何をやったかのだけを列挙するのは, アサガオの観察日記をつけているのと同じです。十分に推敲して, 自分のやったことを芸術作品として仕上げましょう。

* 東京工業大学 工学部

ソフトウェアシンポジウム'95に期待すること

坂本 啓司 *
(SS95 プログラム委員長)

第15回ソフトウェアシンポジウム(SS95)のプログラム委員長を東工大の佐伯先生とともにさせていただくことになりましたオムロンの坂本です。「論文を通すためのハウツウを SEAMAIL に書いてはどうですか」という話しがあり,今年はまだ SEAMAIL に一度も投稿していないので,これを機会に Call for Paper の補足のような話しを書かせていただこうと思いました。「論文を通すためのハウツウ」などというのは私に原稿を書かすためにある人が冗談ぽく言ったのですが,ここでは真面目に,どんなことを考えていてどんなシンポジウムにしたいと思っているのか,またそのためにはどんな論文を期待しているかを書いてみたいと思います。

今年の第14回シンポジウムのプログラムに「Software Lemingineering!?」というパネルディスカッションがありました。詳しい内容はテープおこしされた SEAMAIL の記事を見ていただきたいのですが,プロシーディングの中に Lemingineering という造語の元になった Lemming について次のような説明が載っていました。

Lemming 一定数以上に固体が増え過ぎると,突然やみくもに見える集団暴走を起こし,水に飛び込むなどして個体数の調整を行う小動物。

ここからの話しはパネルの議論内容と余り関係無くなるのですが,実はこの Lemming の意味をごく最近まで全く誤解していました。どのように誤解をしていたかと言いますと,「小動物が多かれ少なかれ持っている一般的な属性」というように思い込んでいたのです。広辞苑でレミングを引くと「ネズミ科ハタネズミ亜科のうちレミング属,クビワレミング属などのネズミの総称」と説明があります。なぜそのように誤解したのかを考えてみたのですが,以前こんな話しを聞いたことがあったためだろうと思います。「アフリカの砂漠にすんでいるある種のアリはアリ塚を作って普段はその中で平和に暮らしているが,一定数以上に個体が増え過ぎると,突然みんながアリ塚を出て砂漠の中を走りまわり死んでしまう。この集団としての行動は特定のアリに限ったことではなく,広く一般に動物が持っている基本的な属性であると思われる。」というものです。そしてこれに続いて聞いた話も思い出しました。それは「動物が持っている基本的な属性なら人間も同じく持っているはずである。世界的な人口増加は指数関数的であり,地球環境の悪化と相まって人類は最悪の場合あと150年ほどでこのような状態になる可能性がある。」この話はかなりショッキングでした。ネズミやアリと人間を同じに扱うのはおかしいと思う人がいると思いますが,逆に否定する根拠もありません。有名な「パブロフの犬」のような動物実験が心理学の世界でよく行われすが,動物の例を人間に当てはめることの出来る場合とそうでない場合を判断する基準がほぼ出来ているそうです。さらに「宇宙の全ては同じ物質から出来ているので,宇宙全体も星雲も星も生物も同じ属性を持っていて同じような変化をする。」という人もいたりします。ここまで行くと,にわかにはうなづき難いのですが,ネズミやアリと同じかもしれないというのは少し考えてみる必要があるように思います。

ところで150年先と言うと今生きている人たちは確実に経験しない世界です。そこでこの話しを聞いたときの人々の反応はだいたい2つに分かれます。1つは「どうせ自分がいなくなってからのことだから知ったことではない。それよりも今日の稼ぎが大事。」というもので,もう1つは「そのような意見が当たっているか否かは分からないが,もし自分の努力不足でそのようになったとしたら子孫に対して申し訳ない。自分なりの努力をしなければ。」というものです。最悪の場合あと150年ほどと言った人(著名な科学者ですが)はこんなふうに言っていました。「人間も他の動物と同じである可能性は高い。しかし人間は他の動物と違って文明を持っている。文明の発達によって人類はこの危機をブレイクスルー出来るかもしれない。そのために,単に物質的なものだけでなく本当の意味で社会を豊かにすることと,科学技術の発展に貢献していかなければならない。」なんか,宗教の話のようになってしまいましたが,私は決して宗教かぶれではありません。とりあえず私は「自

* オムロン(株) EFTS 開発センター 第二開発室

分なりの努力をしなければ。」と思っています。そして具体的には，ごくごく平凡に「今の仕事を頑張らなくっちゃ。」と思っています。[1]

さて，ながながと前置きを書いてしまいましたが，ここからがプログラム委員長としての抱負です。プログラム委員長などという初めての大役を気軽に引受たのですが，その大変さに驚いています。これまで一見，淡々とこなされていた歴代のプログラム委員長には敬服するばかりです。その大変な役割をなぜ引受たかと言いますと「今の仕事の頑張り」に繋がると思っているからです。北海道がいいからとか琵琶湖に行ってみたかったとかといっても，それはあくまで冗談で，論文発表する人も一般参加者も，社外の人との情報交換で「自分の仕事の頑張り」になんらかの形で繋がるものを得られると思うから出てこられていると思います。そして私は，参加の度合いの 2 乗に比例して得られるものが多くなる，と信じています。それでプログラム委員長もやってみようと思った次第です。今まで一般参加しかされていない方も，一度 論文応募をしてみると，この 2 乗の法則が実感として感じられるはずです。ぜひとも，1 人でも多く，より深い関わりを持った参画をお願いしたいと思います。

ところでこのような情報交換の場であるシンポジウムについて少し考えてみたいと思います。百科事典で「シンポジウム」を調べますとこんなことが載っています。

シンポジウム 公開討論会の一つの形式。

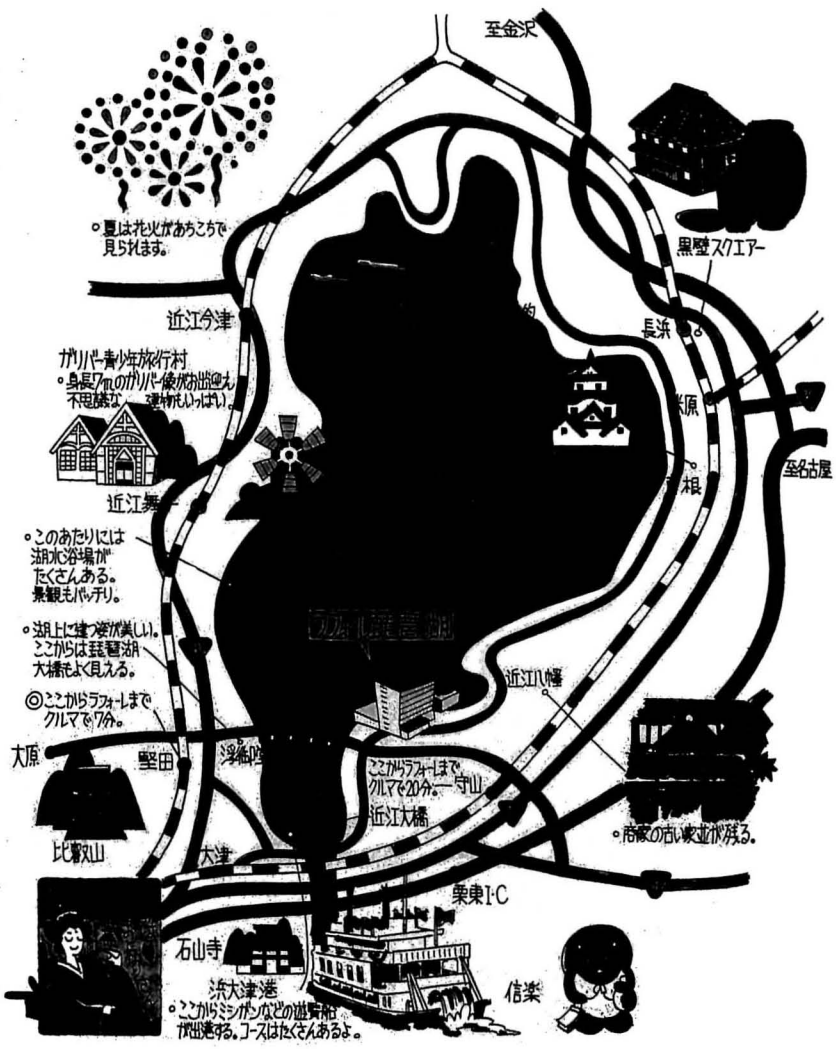
> 一緒に syn- 飲む posis の意のギリシャ語 symposion(饗宴) が語源。特定のテーマに関して討論を行う際，異なった立場や角度からの分析や論究が深まるよう，学識経験者などの専門家を数名選び，参加者に対してそれぞれが意見を述べ，それらの専門的な知見に基づき，質疑応答が主の全体討論会に移行する形式を取る。このような専門家集団による形式は，正しくはシンポジウム・フォーラムという。
>
> 参加者と同じ立場の人が壇上で討議する場合には，形式はほとんど変わらないがパネル・ディスカッションとかパネル・フォーラムと呼ばれる。講師の講演に続いて講師間の討論が行われ，次に一般公衆との間の討論に進み，最後に議長がまとめる。議長のまとめや公衆とのやりとりが省略されることが多いが，これだと効果が少ない。公衆に広い視野と多くの意見を能率的に与えるとともに，公衆の討論参加によって多くの人に，それぞれ独自の意見を形成させるうえで重要な役割をはたす。

こうやって改めて元の意味を調べてみると，「そうだったのか」というところと「これは少し違ってきているな」というところがあるようです。とくにお酒を飲みながらの討論などは SEA の集まりにぴったりで，SS95 でもイブニングパネル，懇親会などが楽しみです。「異なった立場や角度」とか「専門家」というところは必ずしもそのようには思いませんが，「議長のまとめや公衆とのやりとり」の大切さはやはりそうだなと実感します。いずれにしろ，意見交換をし「それぞれ独自の意見を形成させる」ことを目的に集まるのだということをまず認識していただきたいと思います。このために，一般参加者はただ話を聞くだけのために集まるのではなく，3 日間の間に最低 1 回は発言をするように心掛けて見ては如何でしょうか。また発表者は自分の「知見」を披露し，他の参加者を啓蒙すると同時に，討論のなかでさらに知見に磨きをかけていただきたいと思います。

このために論文応募者に 2 つのお願いがあります。1 つは発表内容が他の人にとってなんらかの形で役に立つ，または興味を抱く対象であること。発表者の個人的興味だけのテーマではシンポジウムの目的は達成できないと思います。2 つ目は発表内容が読む人なり，聞く人なりに理解できるものであること。いくら立派な内容でも受け手が理解できないものであれば，全く情報発信をしていないに等しいと思います。文学的文書と違って，科学技術文書にはサービス精神が必要であるということを書いていた人がいましたが全くその通りだと思います。発表者が，発表することのみを目的としたような発表に，決してならないようにお願いする次第です。要するに，一人よがりにならず，自分の発表内容は周りの人にとってどんな価値があるのかを理解し，きちっと自分の主張を伝え，意見交換によってお互いの知見をさらに向上させていただきたいと思います。私の聞きかじりの知識によると，このことを社会心理学の領域では社会的状況認知と社会的相互学習というのだと思います。SS95 をそのための場として，ぜひとも活用していただきたいと思います。

[1] この部分は論理の飛躍があると思われるかも知れませんが，ここではとても説明し切れませんので，ご容赦ください。

最後に, プログラム委員長をさせていただいている特権として, SS95 をどんな特色あるものにしたいかということを述べさせていただきます。論文募集は昨年のものをベースに一部変更して作っていますが, とくに意図して入れた言葉があります。それは「組み込み機器メーカー」という言葉です。私自身が 20 年以上も組み込みソフトに関わってきたというのが一番の理由ですが, それだけではなく世の中のソフトウェア全体の中で, 組み込みソフトの比重が今後益々増えていくだろうと思っているからです。従来, 組み込みソフトは小規模ソフトの代表のように言われてきましたが, 最近は開発量が急激に大きくなってきており, 1M step を越えるものも珍しくなくなってきました。このため少し前のビジネス系ソフトが 遭遇した「規模の拡大」という問題に遭遇し始めています。しかし「規模の拡大」を克服するための管理技術がなかなか組み込みソフト系には移転されておらず, いわゆる古典的な開発のトラブルが多々見受けられます。このうえに, 組み込みソフト特有のハードウェア性能やリアルタイム性といった制約による技術的課題や, ハード開発込みのプロジェクト管理が必要といった管理的課題をあわせて持っています。このように多くの課題を抱えているにも関わらず, 商品ごとの特質を強く意識するためか組み込みソフトについての技術交流があまりされていなかったのが実情です。そこで SS95 のプログラム委員には組み込みソフトを担当されている方に特に多くなっていただいており, 多くの論文集めと活発な議論を期待しています。皆さんも大いにご期待ください。

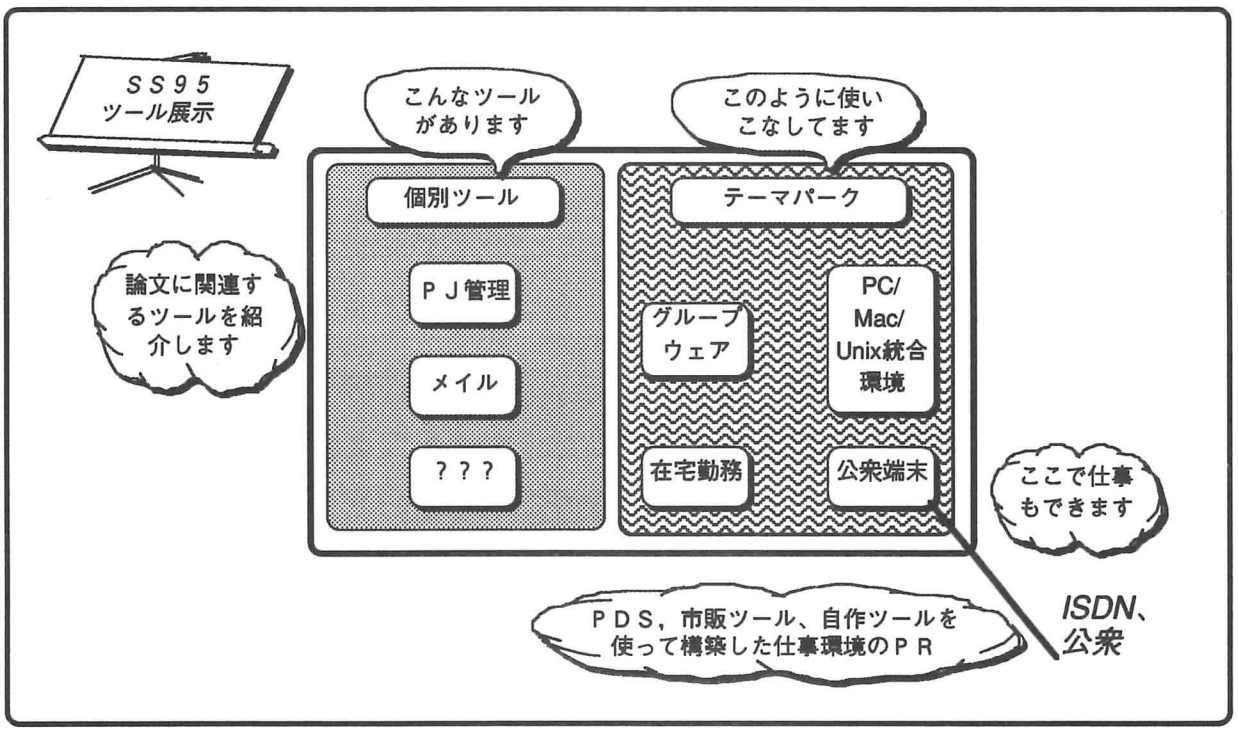


SS95 ツール展示

佐藤 正道 *
(SS95 ツール展示委員長)

はじめまして, オムロンの佐藤と申します。SEA とは, オムロンの坂本に紹介されて SS93 に論文を投稿させてもらって以来のおつきあいです。SEA を通して業界の大先輩から様々な考えを聞かせて頂く機会ができたことを, 大変感謝しています。

今回坂本がプログラム委員長に就任するにあたり, その兵隊の一人としてツール展示を担当させて頂くことになりました。『せっかくやるなら, 論文発表に負けない, 面白いコーナに!』を基本方針に取り組んでみたいと思っています。(どこまでやれるかは???ですが。。)



SS95 ツール展示コンセプト図

今考えているコンセプトは, , , 従来のツール展示に＋α して, 『ツールを使いこなして築き上げた仕事環境』の展示をやってみたいと思っています。見に来て下さった方と, 将来の環境の夢を語れれば最高ですね。

誰が出典してくれるの?, インフラ環境の準備はうまくいくか?, 予算は足りるの? など課題は山積みなのですが, きっと皆さんが協力して下さるだろう。。。と, 楽観的に考えています。

こんなツール展示を見てみたい! と思われた方, 是非協力をお願いします。実現のためにはあなたの協力が必要です。

sato@lsa.ncl.omron.co.jp

まで連絡を下さい。一緒にやりましょう。

ではでは。

* オムロン (株) 技術開発本部開発支援センタ SPI 推進課

BOFの守・破・離

江谷 典子 *
(SS95 チュートリアル実行委員長)

仏教の言葉に「守・破・離」というのがあります。「守」というのは, 物事はすべてが定石というものがあるから, 始めは, まずこの定石を徹底的にマスターするようにする。この定石が身についたら, こんどは定石を破り, 応用を試みるようにする。最後は, 定石や応用の意識からはなれ, その時その場に応じた調整をする「離」となる。この守・破・離の発展は単に経験の積み重ねだけでなく, 人との出会いや出会いによる知的触発が推進させるといわれます。また, 人が集まる場とは, 最初は互いの情報を交換し, その中から同じ志を持った集まり(BOF:birds of a feather) が生まれ, その集まりはやがて共通の目標に向けて行動を起こす集団へと成長していきます。15 年前, 通常のアカデミックなコンファレンスとはちがって, 開発現場で働くソフトウェア技術者たちが自らの経験を通じて獲得した実践的な技術や知識を交換する場が必要だという認識からシンポジウムが誕生しました[1]。そろそろ, 情報交換のための BOF から 事を起こせる BOF に向けて「破」の段階にさしかかっているのではないでしょうか?

ソフトウエア産業はまだまだ若い産業といわれつつ, この 15 年の間に「ソフトウェア技術者」や「開発現場」の意味が多様化してきました。

- コードを書いている人
- システム設計をしている人
- システム分析をしている人
- 要求分析やコンサルテーションをしている人
- ソフトウェア開発組織を管理している人
- 研究をしている人
- ソフトウエア開発環境の整備をしている人
- ソフトウエア保守をしている人
- ネットワーク管理をしている人
- ソフトウエアの品質管理をしている人
- 情報システムあるいはコンピュータシステムを利用している人

など様々な技術者がおり, 産業構造の変換とともに開発する対象も社会の要求, サービス, システム, プログラムなど一言で共有しきれないほどのソフトウエア技術者の開発現場があります。この多様化したソフトウエア技術者が一同に集まるシンポジウムは異文化コミュニケーションの場であり, 一つの組織の中ではありえない新しい発見や出会いが生まれる場でもあり, アカデミズムと産業界の本音の交流ができる場でもあるといえるでしょう。

まずは, 「発言をしましょう!」「心に思うことを言ってみましょう!」。論文として表現し発表するのもいいでしょう。同じ関心あるいは専門領域の方同志で問題を掘り下げる議論も大いにやりましょう。そして, 今は携わっていない, あるいはあまり関係がないように一見みえる領域にも, あなたの観点から意見を言いましょう。ソフトウエア・シポジウムの定石を大切にしつつ, 新しい一歩を皆様とともに築くことができれば幸いです。そこで, 今回のシンポジウムでの期待は.....

琵琶湖のほとりで多くの BOF が結成され, 多くの ML(mailing list) が作られること

シンポジウムは, 誰かが作ってくれるものではなくて参加者の方々がどれだけ意見を出し議論を行い盛り上げてくださるかが命であります。今回, 若輩者である私がチュートリアル実行委員長を務めさせて頂く上に, 企画および運営を SEA 関西支部で行うことになりました。どのようなチュートリアルにするかは SEA 関西のメンバだけでなく, SEA の先輩方々, シンポジウムに参加しようと考えておられる方々との "participatory design" によって成功するものだと思います。皆様の御力添えを頂きたく存じます。SEA 関西のメーリングリスト seakansai@comm.eng.osaka-u.ac.jp へ御一報頂ければ幸いです。宜しくお願い申し上げます。

* 富士ゼロックス情報システム (株) 技術部
[1] 1994 年ソフトウェア・シンポジウム論文集 岸田さん御執筆 : ソフトウエア・シンポジウムの歴史より

ローカルアレンジメントチェアからのメッセージ

田中 敏文 *
(SS95 ローカル・アレンジメント)

私がローカルアレンジメントチェアに任命された理由はいくつかありますが, その中で一番納得性の高い理由は, 私が SS94 の最優秀論文賞を戴いてしまったことのようです。「あいつに賞金をやっておけば後々いろいろ使えるだろう」という, 人なのか神なのかは知らないですが, 何らかの意志が働いたと見るべきでしょう。

「はめられた」という気持ちが全くないわけではないですが, その賞金を一晩でものの見事に使い切ってしまった私は, さらに「業務多忙やさかいそんなんやめとけ」という心ない上司に恵まれていない私は, 「やるっきゃない」状態に陥ってるわけです。(ちなみに, 実はツール展示チェアに任命された某S氏もその賞金を使い切ることに加担したのであります。)

まー, 慣れないことばかりですので “すったもんだ” もありましょうが, SS95 に参加される皆さんには, ローカルアレンジメントに関しては, 不自由を感じさせることなく, さらには快適さと楽しさを感じて頂けるような配慮と企画をしていきたいと思いますので, SEA 会員の皆様からの御協力をよろしくお願いします。

ローカルアレンジメントチェア就任に関する経緯と挨拶はこのくらいにして, 以下に「今こんなことを考えている」という内容を紹介して, 抱負に変えさせていただきます。

まず, 会場についてですが, ホテルの会議室以外にプラネタリウムが使用可能ですので, ひとつのセッション会場として使ってみようかなと思っています。宇宙空間に飛び出したような雰囲気の中での発表と議論が, 未来のソフトウェア技術に対する夢にまで広がっていくことでしょう。

次に, 恒例となっております皆さん御期待の懇親会についてですが, せっかく皆さん湖国にまでお来し下さるのですから, 湖上遊覧船でのパーティ(ディナーオンクルーズ)でも, と思ったのですが予算的にきついものがあります。プログラムとも相談しながらですが, 参加者負担によるオプション化とするか断念するかを余儀なくされそうです。代わりに, リーズナブルな予算の範囲内で懇親会の趣向を検討していきます。

最後に, プログラムを離れた自由時間の有効な過ごし方についてですが, 会場がリゾートホテルということで, 結構いろいろな施設が揃っています。当然シンポジウムで熱心に議論することが本分ではありますが, それ以外の時間で, テニスコート, 体育館, ゴルフ練習場, 屋内プール, クロッケーゴルフ, 浴場・サウナ (以上無料), レンタサイクル, ビューティサロン, ブライダル施設, レストラン (以上有料) などのホテル施設を有効に活用していただくために, ホテルやシンポジウムプログラムとの調整と参加者への情報提供に注力したいと思います。

以上, 簡単ではありますが, SS95 ローカルアレンジメントチェアとしての抱負を述べさせていただきました。

* オムロン(株) EFTS 開発センター 第2開発室 (toshi@eftses.krc.omron.co.jp)

論 文 募 集

ソフトウェア・シンポジウム'95

1995年6月14日(水)～16日(金)　於: ラフォーレ琵琶湖(滋賀県守山市)

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　情報処理学会　情報サービス産業協会

ソフトウェア・シンポジウムは、ソフトウェアハウス、コンピュータおよび組込み機器メーカ、エンドユーザ、大学、研究所などさまざまな場で活躍している技術者、管理者および研究者が一堂に会し、ソフトウェア技術に関する多面的な経験や知識を交流するユニークで貴重な場を提供してきました。一つの区切りとも言える第15回を迎える'95年のシンポジウムは、北上を続けてきた開催地を一転し日本の真ん中、琵琶湖のほとりで開催します。さらに今回はツール展示を併設し、技術交流をより充実したものとします。

毎年、現場に立脚した質の高い論文が集まるこのシンポジウムのレベルは、回を重ねる毎に着実に上がっております。今回も第14回と同様に単一トラックでセッションを構成しますが、そのため採録可能な論文は15編程度と少なくなる見通しで、高レベルの論文採用基準になると予想されます。皆様の積極的な御応募を期待いたします。

なお、従来通り最も優秀な論文に送られる最優秀論文賞は、シンポジウム開催時に選定します。論文は日本語の他、英語による投稿も可としますが、発表は日本語に限ります。応募論文テーマとしては、たとえば

- 分析／設計技法　・CASE　・開発環境　・ネットワーク　・CSCW/グループウェア
- プロジェクト管理　・品質管理　・プロセス改善　・メトリックス　・ソフトウェア産業の新パラダイム
- 標準化対応　・人間的要因　・技術者教育　・マルティメディア　・ドメインエンジニアリング

などさまざまなものが考えられますが、必ずしもこれらにとらわれる必要はなく、ソフトウェアの領域で他の人に参考となるテーマであれば何でも構いません。先進的な研究テーマの発表はもちろん歓迎しますが、開発現場での苦労・努力をまとめた経験報告も大歓迎です。奮って御応募下さい。

応募要領

応募論文は未発表のものに限ります。また、他への二重投稿はご遠慮下さい。今回もプログラム委員会での審査は本論文にて行ないますが、応募予定の方は論文概要を記入したカバーシートにより**1994年12月20日**までにエントリをしてください。カバーシートは必要事項を記入の上、極力電子メールで、電子メールが使えない場合はFAXまたは郵送で2人のプログラム委員長のいずれか宛にお送り下さい。本論文の様式は自由ですが、A4サイズで5～10ページ程度を目安とし、7部を郵送で**1995年1月31日**までにプログラム委員長宛にお送り下さい。プログラム委員会で内容の審査を行い、結果を3月下旬までに応募者全員に通知します。その他不明な点がありましたらプログラム委員長までお問い合わせ下さい。

主要スケジュール:

1994年12月20日(火)	論文概要によるエントリ締切り
1995年1月31日(火)	本論文、及びツール展示応募締切り
1995年3月下旬	採否通知送付、及びツール展示依頼送付
1995年4月下旬	カメラレディ原稿締切り

シンポジウムスタッフ

実行委員長	中野 秀男(大阪大学)		
プログラム委員長	佐伯 元司(東京工業大学)	坂本 啓司(オムロン)	
プログラム委員	青山 幹雄(富士通)	鯵坂 恒夫(京都大学)	荒木 啓二郎(奈良先端大)
伊藤 昌夫(MASC)	江谷 典子(FXIS)	大場 充(広島市立大学)	尾本 林貞(ダイキン)
岸田 孝一(SRA)	楠本 真二(大阪大学)	熊谷 章(PFU)	小泉 毅(キヤノン)
小林 允(日本ユニシス)	酒匂 寛(SRA)	砂塚 利彦(NEC)	高橋 光裕(電力中研)
田中 敏幸(シャープ)	谷津 行穂(日本IBM)	玉井 哲雄(東京大学)	中来田 秀樹(Next Foundation)
中谷 多哉子(FXIS)	布川 博士(宮城教育大学)	野中 哲(ATJ)	平山 伸一(さくらKCS)
二木 厚吉(北陸先端大)	古川 善吾(九州大学)	古山 恒夫(日本電信電話)	増子 泰弘(松下通信)
増井 和也(東芝AS)	松原 友夫(Peopleware)	松本 健一(奈良先端大)	
ツール展示委員	佐藤 正道(オムロン)		
ローカル・アレンジメント	田中 敏文(オムロン)		

ソフトウェア･シンポジウム '95

論文応募カバー･シート

氏名 (筆頭著者)：＿＿＿＿＿＿＿＿＿＿ (ふりがな) ＿＿＿＿＿＿＿＿＿＿

所属 (会社･大学･組織)：＿＿＿＿＿＿＿＿＿＿

部門 (学部)：＿＿＿＿＿＿＿＿＿＿ 役　職：＿＿＿＿＿＿＿＿＿＿

住　所：(〒　　)＿＿＿＿＿＿＿＿＿＿

TEL：　　　　(内線　　)　　FAX:

E-mail：＿＿＿＿＿＿＿＿＿＿

論文タイトル：＿＿＿＿＿＿＿＿＿＿

(ツール名)：＿＿＿＿＿＿＿＿＿＿

共同執筆者：(もしいれば。ツールの場合不要)

氏名:＿＿＿＿＿＿＿＿＿＿ 所属:＿＿＿＿＿＿＿＿＿＿

氏名:＿＿＿＿＿＿＿＿＿＿ 所属:＿＿＿＿＿＿＿＿＿＿

氏名:＿＿＿＿＿＿＿＿＿＿ 所属:＿＿＿＿＿＿＿＿＿＿

論文概要 (ツール概要):

＿＿＿＿＿＿＿＿＿＿

＿＿＿＿＿＿＿＿＿＿

＿＿＿＿＿＿＿＿＿＿

＿＿＿＿＿＿＿＿＿＿

＿＿＿＿＿＿＿＿＿＿

＿＿＿＿＿＿＿＿＿＿

＿＿＿＿＿＿＿＿＿＿

＿＿＿＿＿＿＿＿＿＿

＿＿＿＿＿＿＿＿＿＿

応募カテゴリ：(いずれかにチェック)　□：経験報告　□：技術論文　□：ツール展示

キーワード：(最大5つのキーワード挙げてください。)

1：＿＿＿＿＿＿＿＿＿＿　2：＿＿＿＿＿＿＿＿＿＿

3：＿＿＿＿＿＿＿＿＿＿　4：＿＿＿＿＿＿＿＿＿＿

5：＿＿＿＿＿＿＿＿＿＿

-------------------- 論文応募送付先　&　応募締切り：1994年12月20日 (火) --------------------

〒152　東京都目黒区大岡山2-12-1
東京工業大学工学部情報理工学科 計算工学専攻
佐伯 元司
E-mail: saeki@cs.titech.ac.jp
FAX: 03 - 3729 - 1399　TEL: 03 - 3726 - 1111 ext. 2192

または

〒525　滋賀県草津市西草津2丁目2-1
オムロン(株)EFTS 統轄事業部開発センタ第2開発室
坂本 啓司
E-mail: sakamoto@eftses.krc.omron.co.jp
FAX: 0775 - 65 - 5579　TEL: 0775 - 65 - 5048

-------------------- ツール展示応募送付先　&　応募締切り：1995年1月31日 (火) --------------------

〒617　京都府長岡京市下海印寺
オムロン(株) 技術本部開発支援センタ SPI 推進課
佐藤 正道　E-mail: sato@lsa.ncl.omron.co.jp　FAX: 075 - 953 - 7604　TEL: 075 - 951 - 5111 ext. 3407

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