SEAMAIL

Newsletter from Software Engineers Association

Volume 3, Number 9 1988

目　次

編集部から	1
第6回夏のプログラミング・ワークショップ特集	2
Call for Paper	2
プログラム	3
参加者名簿	4
グループ別討論	5
ポジション・ペーパ & 感想	24
5th International Software Proces Workshop	62
ソフトウェア・プロセス・ワークショップ参加者募集	63
会員状況	65

ソフトウェア技術者協会（ＳＥＡ）は，ソフトウェア・エンジニアの，ソフトウェア・エンジニアによる，ソフトウェア・エンジニアのための団体であり，これまでに日本になかった新しいタイプのプロフェッショナル・ソサイエティたることを目指して，１９８５年１２月２０日に設立されました．

現在のソフトウェア技術が抱える最大の課題は，ソフトウェア・エンジニアリング研究の最前線（ステイト・オブ・アート）と，その実践状況（ステイト・オブ・プラクティス）との間に横たわる大きなギャップを埋めることだといわれています．ソフトウェア技術の特徴は，他の工学諸分野の技術にくらべて属人性がきわめて強い点にあります．したがって，そうしたテクノロジー・トランスファの成否の鍵は，研究者や技術者が，既存の社会組織の壁を越えて，相互の交流を効果的に行うためのメカニズムが確立できるか否かにかかっています．ＳＥＡは，ソフトウェア・ハウス，計算センタ，システム・ハウス，コンピュータ・メーカ，一般ユーザ，大学，研究所など，さまざまな職場で働く人々が，技術的・人間的交流を行うための自由な＜場＞であることをを目指しています．

ＳＥＡの具体的な活動としては，特定のテーマに関する研究分科会（ＳＩＧ）や地方支部の運営，月刊機関誌（ＳＥＡＭＡＩＬ）の発行，各種のセミナー，ワークショップ，シンポジウムなどのイベントの開催，既存の学会や業界団体の活動への協力，また，さなざまな国際交流の促進等があげられます．

なおＳＥＡは，個人参加を原則とする専門家団体です．その運営は，つねに中立かつ技術オリエンテッドな視点に立って行われ，特定の企業や組織あるいは業界の利益を代表することはありません．

代表幹事： 岸田孝一

常任幹事： 臼井義美　久保宏志　熊谷章　佐藤千明　藤野晃延　松原友夫　吉村鉄太郎

幹事： 青島茂　稲田博　岡田正志　落水浩一郎　片山禎昭　川北秀夫　杉田義明　鈴木弘　武田知久　田中慎一郎　長井剛一郎　中園順三　中野秀男　西尾出　野村敏次　野村行憲　針谷明　深瀬弘恭　藤本司郎　村井進　盛田政敏

会計監事： 辻淳二　吉村成弘

常任委員長： 臼井義美（技術研究）　久保宏志（企画総務）　藤野晃延（会誌編集）　杉田義明（セミナー・ワークショップ）

分科会世話人
環境分科会(SIGENV)：田中慎一郎　渡邊雄一
管理分科会(SIGMAN)：相沢圭一　川北秀夫　芝原雄二　野々下幸治
教育分科会(SIGEDU)：大浦洋一　杉田義明　中園順三
再利用分科会(SIGREUSE)：青島茂　阿倍正平　村井進
ネットワーク分科会(SIGNET)：青島茂　野中哲
法的保護分科会(SIGSPL)：能登末之
ＣＡＩ分科会(SIGCAI)：大木幹雄　寺嶋裕一　中谷多哉子　中西昌武
ドキュメント分科会(SIGDOC)：田中慎一郎　野辺良一

支部世話人
関西支部：臼井義美　盛田政敏
横浜支部：熊谷章　林香　藤野晃延　松下和隆
長野支部：小林貞幸　佐藤千明　細野広水
名古屋支部：岩田康　鈴木智　西村亨
九州支部：植村正伸　小田七生　藤本良子　平尾一浩　松本初美　中島泰彦　能見巧　後藤芳美

ＳＥＡＭＡＩＬ編集グループ：岸田孝一　佐原伸　芝原雄二　関崎邦夫　田中慎一郎　中村昭雄　長井修治　成沢知子　野辺良一　藤野晃延　渡邊雄一

ＳＥＡＭＡＩＬ　Ｖｏｌ．３，Ｎｏ．９　昭和６３年１２月１日発行
編集人　岸田孝一
発行人　ソフトウェア技術者協会（ＳＥＡ）
〒１０２　東京都千代田区隼町２－１２　藤和半蔵門コーブビル５０５
印刷所　サンビルト印刷株式会社　〒１６２　東京都新宿区築地町８番地
特価　１，０００円　（禁転載）

編集部から

1. ふたたび，ワークショップ特集

編集長が変わっても，相変わらず船便シンドロームから抜け出せず，事務局の中島小姐のストレス性胃炎の原因となっている SEAMAIL です．なにせ，この夏から秋にかけて，イベントが立て続けにあり，編集ボランティアのほとんどがそれに巻き込まれているために，なかなか材料の整理がつきません．

何人かの在宅編集員のご協力のおかげで，以前に行われたセミナーやシンポジウムの記録が，少しずつ，音声磁気テープからディジタル磁気フロッピーに変換されてきていますが，最後の手直しとフォーマッティングがいま一歩です．

そこで，この号は，一応の記録と参加者のポジション・ペーパ（Rewrite 版）がまとまった「夏のプログラミング・ワークショップ」（第６回若手の会）の報告特集の形でまとめることにしました．これで２号つづけてワークショップ・レポートになってしまいましたが，どうか悪しからず．

2. 次号以降のとりあえずの予定

現在，この夏に行った環境実態調査アンケートの報告を，分析ＷＧのチーフ野村敏次さん（JIP）が取りまとめている最中です．多分年内には印刷にまで漕ぎ着けられるでしょう．また，９月末の教育ワークショップおよび十月初めの日中シンポジウム・ツアーの報告も，やはり年末を目標に原稿整理中です．これらの報告特集は，完成次第順次お手元にお届けします．

それ以外の原稿としては，６月のシンポジウムの招待講演者 W. Scacchi 教授の原論文の日本語訳が，ケーシーエスの平山さんの手で出来上がっています．また，前号にのせた信頼性シンポジウムのセッション・レポートの続きや，いくつかのセミナー／チュートリアル／フォーラムの記録も，あと一歩で完成の状態にあります．こうした原稿も，１冊分のボリュームにまとまり次第，迷わず印刷に回します．

といった具合いで，来春までには，船便症候郡から脱却し，中島さんの愁眉を開いてさしあげられるのではないか，と思うのですが，甘いかな？

3. とにかく，

野菜でも，お肉でも，魚でもいいから，お料理の材料を持って来て下さい，というのがキッチンからのシェフの悲痛な（？）叫びです．できれば，そのまま電子レンジに入れてテーブルに出せるような，レトルト原稿（つまりワープロでＡ４版にフォーマットしてあるもの）が，毎月何篇か欲しいな！

プログラマって，お酒を飲むとあんなにおしゃべりなんだから，月に１度くらい，ウィスキー片手にワープロの前に座って見るのもいいんじゃない？．．．．とこれは，２代目編集長の気取った嘆き節でした．

第 6 回 夏のプログラミング・ワークショップ

ソフトウェア・エンジニアリング技法のケース・スタディ

－ ソフトウェア開発の「銀の弾丸」を求めて －

参加者募集

主催：ソフトウェア技術者協会

後援：（社）情報サービス産業協会

これまで５年間「若手の会in盛岡」として開催されてきたこのワークショップも，いまではすっかり定着し，業界各社からも，次世代のエースを鍛える格好なトレーニングの「場」として高く評価されています．

第６回目を迎える今年は，会場を八ヶ岳山麓の別荘村に移し，運営スタイルも「若手の会」ＯＢを中心とするケース・スタディ方式を採用して，より具体的な課題を全員参加のグループ討論を通じて徹底追求することにしました．今回の討論テーマとしては，ややもすれば名前だけが先行し現場への浸透が遅れている開発技法のいくつかを取り上げ，それぞれの長所・短所を十分に比較検討しようと考えています．

具体的には：

・形式的仕様記述を用いたアプローチ
・オブジェクト指向のアプローチ
・エキスパート・システムによるアプローチ
・CASE ツールを用いたアプローチ
・プロトタイピングによるアプローチ

などを用いて現実的な例題にアタックし，F.Brooks の高名なエッセイに述べられた「銀の弾丸」（ソフトウェアという魔物を倒す最終兵器）の可能性を探ろうというのが，その狙いです．

多くの若手技術者方々の積極的な御参加をお待ちします．

開 催 要 領

１．期日： 昭和 63 年 8 月 31 日（水）午後から 9 月 3 日（土）正午まで（3 泊 4 日）

２．会場： 八ヶ岳高原・泉郷（山梨県北巨摩郡・JR 中央本線小淵沢駅からバス 10 分）

３．定員： 35 名

４．参加資格： 年齢 30 歳未満，経験 2 年以上のソフトウェア技術者

５．費用： 60,000円(SEA 会員，JISA 会員)， 80,000円(一般)

現地集合・現地解散とし，期間中の宿泊費（３食付き）および資料代を含みます．

６．スタッフ：

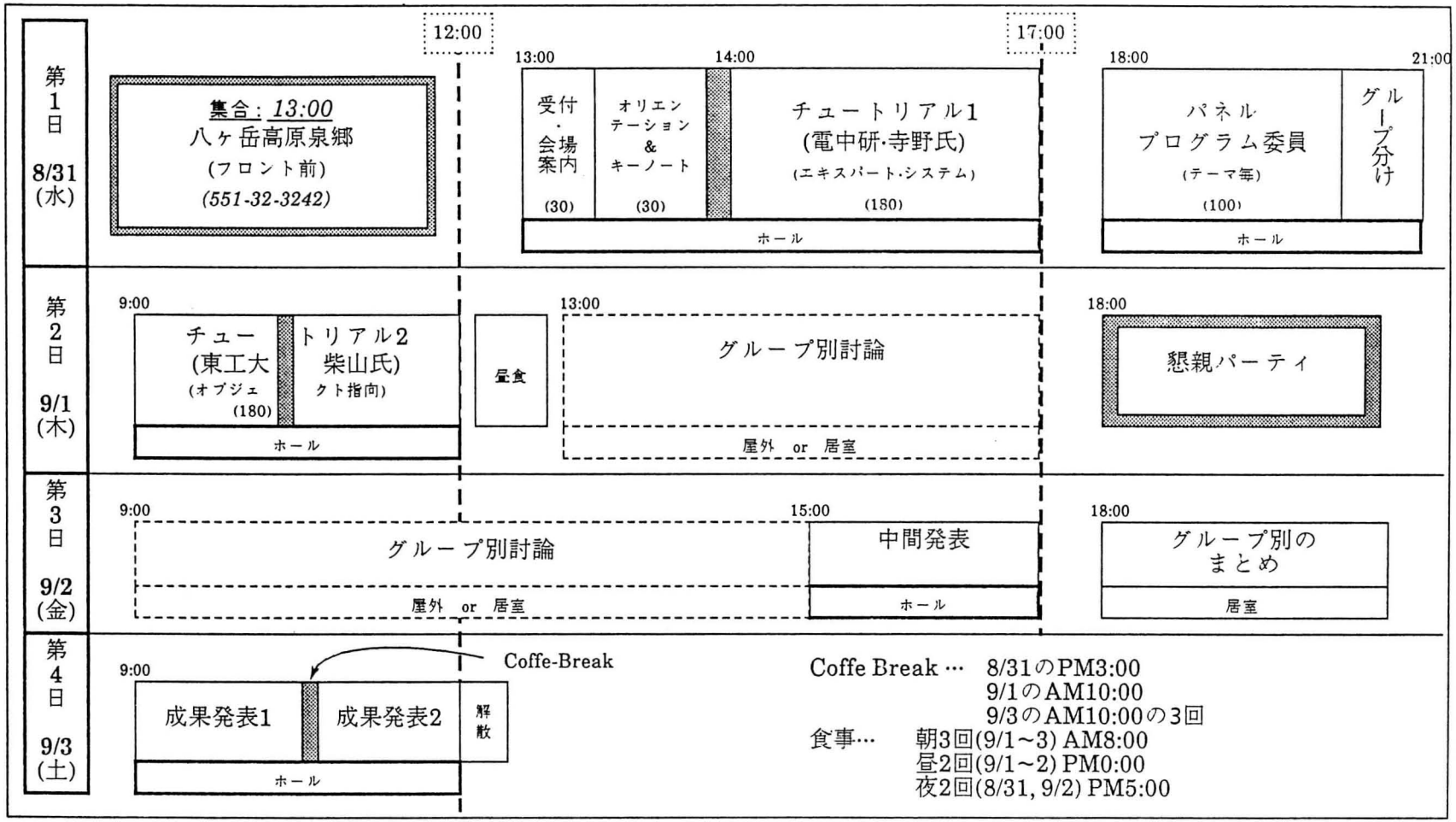
実行委員長： 藤野晃延（富士ゼロックス情報システム）

プログラム委員長： 坂下 秀（アステック）

プログラム委員： 岡芹晃（東芝エンジニアリング） 佐藤千明（長野県協同電算） 沢田寿美（ＳＲＡ）
中川中（電子技術総合研究所） 藤本良子（フリー） 松下和隆（三菱電気東部コンピュータ）
山本一成（ＫＣＳ） 渡辺雄一（電力計算センター） 他

招待講演者（予定）：柴山悦哉（東京工業大学） 寺野隆雄（電力中央研究所）

７．申込方法： 裏面の参加申込用紙に必要事項を記入の上，昭和 63 年 7 月末日までに，郵便または FAX で，実行委員長宛お送り下さい．申込者多数の場合には，プログラム委員会でポジション・ステートメントの内容を審査の上，参加者を選定させていただきますので，あらかじめ御了承ください


第1日 8/31(水)
12:00
集合：13:00
八ヶ岳高原泉郷
(フロント前)
(551-32-3242)
13:00
14:00
17:00
受付・会場案内
(30)
オリエンテーション&キーノート
(30)
チュートリアル1
(電中研・寺野氏)
(エキスパート・システム)
(180)
ホール
18:00
21:00
パネル
プログラム委員
(テーマ毎)
(100)
グループ分け
ホール
第2日 9/1(木)
9:00
チュー
(東工大
(オブジェ
(180)
トリアル2
柴山氏)
クト指向)
ホール
昼食
13:00
グループ別討論
屋外 or 居室
18:00
懇親パーティ
第3日 9/2(金)
9:00
グループ別討論
屋外 or 居室
15:00
中間発表
ホール
18:00
グループ別のまとめ
居室
第4日 9/3(土)
9:00
Coffe-Break
成果発表1
成果発表2
解散
ホール
Coffe Break … 8/31のPM3:00
9/1のAM10:00
9/3のAM10:00の3回
食事… 朝3回(9/1～3) AM8:00
昼2回(9/1～2) PM0:00
夜2回(8/31, 9/2) PM5:00

実行委員	藤野　晃延	富士ゼロックス情報システム
プログラム委員	坂下　秀	アステック
	佐藤　千明	長野協同電算
	岡芹　義範	東芝エンジニアリング
	澤田　寿美	SRA
	松下　和隆	三菱電気東部コンピュータ
	渡辺　雄一	電力計算センター
	佐原　伸	SRA
	藤本　良子	フリー・ランサー
テクニカル・スタッフ	中川　中	SRA
	山本　一成	KCS
	板倉　重利	呉電子計算センター
	中沢　賢光	日本電気ソフトウェア
	山田　良一	日本情報開発
	井上　文人	早稲田大学
	伊藤　達哉	早稲田大学
	前野　貴志	小松ソフトウェア開発
	王生　幸雄	小松ソフトウェア開発
	香西　哲也	ランド・コンピュータ
	平尾　一浩	ヒラタ・ソフトウェア・テクノロジ
	中山　聡	日本タイムシェア
	立木　勲	三井銀ソフトウェアサービス
	田中　慎一郎	SRA
	後藤　浩	シスプラン
	林　好一	SRA
	桜井　麻里	SRA
	西川　典子	SRA
	岡田　正之	SRA
	森　博幸	日立ビジネス機器
	ままだ　健	ディーエム情報システム
	蜷峨山　亨	三菱電機セミコンダクト・ソフトウェア
	望月　好光	ユニテック
	藤村　洋	岩手電子計算センター
	吉田　慎哉	岩手電子計算センター
	笠原　幸夫	システム・ブランニング
	末次　眞之	システム・ブランニング
	穂刈　顕	日本テクニカルシステム
	中村　伸博	三菱電機コントロール・ソフトウェア
	内海　保	三菱電機コントロール・ソフトウェア
	方　学分	静岡大学
	深野　寛子	日本情報開発
	太田　剛	静岡大学
招待講演	寺野　隆雄	電力中央研究所
	柴山　悦哉	東京工業大学

グループ別討論

第1グループ	太田　剛	----------	6
第2グループ	板倉　重利	----------	9
第3グループ	山田　良一	----------	13
第4グループ	内海　保	----------	16
第5グループ	岡田　正之	----------	20
第6グループ	嵯峨山　亨	----------	21

第１グループ 討論報告

㈱ソフトウェア・リサーチ・アソシエイツ　中川　中
㈱富士ゼロックス情報システム　藤野晃延
㈱日本情報開発　深野寛子
静岡大学大学院電子科学研究科　太田　剛

１．はじめに

第６回夏のプログラミング・ワークショップは、ソフトウェアというモンスターを倒すための銀の弾丸を探るというテーマのもと、８月３１日から４日間、八ケ岳高原泉郷に於いて開催された。

我々第１グループでは、銀の弾丸の候補として形式的仕様のアプローチをとりあげた。そして、代数的仕様記述法に基づいて設計開発された代数型言語ＯＢＪ２を用い、具体的な問題をＯＢＪ２で記述することにより、その有用性を確かめ実感し、銀の弾丸となり得るかどうかを検討した。

２．形式的仕様

形式的仕様とは、数学的理論を背景に持ち、シンタックスおよびセマンティクスが正確に定められた人工言語を用いて仕様を書くものである。そのために、次に示すような特長を持つ。

①シンタックスもセマンティクスも決まっているがゆえに、通常のプログラミング言語と同様に、この仕様はそのままの形で解釈実行することができる。そのため、仕様の段階でシステムの動作や効率の良さを確認することができる。

②数学的理論を背景に持つためにその正当性の証明ができる。現在のところ、仕様が正しいかどうかは、問題領域に関する経験や知識を多く持っている専門家の能力に依存しており、その正しさはシステムが出来上がるまで確認ができない。しかし、形式的仕様を用いれば、設計の段階でその正しさを確認できることになる。こうして正しいことを確認した仕様をもとに、自動生成するなり、人間が手でコーディングするなりして、既存のプログラミング言語に落とせばよい。

上に述べた特長はあるものの、現在のところ形式的仕様言語の中で時間の概念を扱うことは困難であり、時間の制御を主としているシステム、例えば銀行のオンラインシステムのようなリアルタイムシステムを記述することができない。この点は今後の研究課題であろう。

３．ＯＢＪ２[1]

我々第一グループでは、はじめにでも述べたように、形式的仕様言語としてＯＢＪ２を用いることになっていたので、まず最初にグループリーダーの中川氏より、ＯＢＪ２についての講義を受けた。

ＯＢＪ２は代数型の言語である。すなわち、ＯＢＪ２が持っている数学的背景理論は、代数である。

代数型の言語で問題世界を記述するには、まず問題世界を構成している対象をその種類（ソート）によっていくつかの集合に分割することから始める。次に、同じソートに属する複数の対象の間に同等関係を定義することによって、記述したい問題世界に意味を与える。各ソートに属するすべての対象は、定対象（ｃｏｎｓｔａｎｔ）に対して適当なオペレーターを適用して得ることができる。例えば、自然数というソートに属する任意の対象は、「１を加えた次の自然数を値とする」後者関数というオペレーターを定数０に対して何回か適用して得ることができる。

簡単な例として、自然数とその上のいくつかの関数をＯＢＪ２で記述したものを示す。自然数における定数（定数オペレーター）として０を定義し、「１を加えた次の自然数を値とする」後者関数と加算および乗算をオペレーターとして定義する。次に、これらのオペレーターを用いて、自然数間の同等関係を等式で与える。例えば、「ある自然数と定数０とを加算したものはもとの自然数に等しい」「ある自然数と定数０との積は０に等しい」等を等式で書き並べることになる。

```
obj INTEGER is
   sort Int.
   op   0  : -> Int.
   op suc  : Int -> Int.
   op _+_  : Int Int -> Int.
   op _*_  : Int Int -> Int.
   var N, M : Int.
   eq Int : N + 0 = N.
   eq Int : N + suc(M) = suc(N + M).
   eq Int : N * 0 = 0.
   eq Int : N * suc(M) = (N * M) + N.
jbo
```

以上、簡単にＯＢＪ２の紹介をしたが、詳細については参考文献その他にゆずることとする。

４．課題

さて、このような特徴を持つＯＢＪ２を用いて、実際に問題を記述してみて、その有用性を確かめることが我々第一グループの課題であった。問題としては、

①ビール製造過程のプロセス・コントロール

②簡単なプロダクション・システム

の２つが与えられ、①を深野が、②を太田がそれぞれ担当した。

ＯＢＪ２の記述法は、手続き的ではなく宣言的に性質を書き並べる方法をとるので、手続き的な記述に慣れている者としては、はじめ戸惑いを感じた。が、慣れるにしたがって、記述量の少なさおよび簡明さを感じるようになった。

課題を解決した結果としてのＯＢＪ２プログラムは、最終日の成果発表でも述べたし、スペースの関係上ここには示さないが、結局のところ記述量としては、Ｂ５ノートに換算して、①のプロセス・コントロールで２ページほど、②のプロダクション・システムで４ページほどであった。他のグループではマシンを持ち込んで体験学習（？）していたところもあったようだが、我々のところでは処理系を持ち込んだわけではないので、正しい記述がなされているとは必ずしも言えない。しかし、この程度の量の記述で一通りの動作を書き下すことができたという点で、形式的な仕様記述の威力を感じた。手続き的に記述した場合、この何倍ほどの量になるであろうか。おそらく１０倍程度では済まないに違いない。処理系があれば、この記述（ＯＢＪ２で記述された設計書）を解釈実行して動作を確かめることができたはずであり、さらにその威力を実感できたであろうが、残念ながら今回はできなかった。

５．まとめ

こうした課題を通して、ＯＢＪ２で仕様（設計書）を書くことに関して我々が感じたことを以下にまとめる。

・戸惑いの問題

手続き的な記述に慣れている者にとっての宣言的記述をすることへの戸惑いがあったが、これは純粋に慣れの問題であり、また背景理論の教育の問題である。よって、致命的な欠点ではない。

・記述量の少なさ

手続き的記述に比べて宣言的記述をしたときの記述量は少ない。これは、宣言的記述をサポートする言語処理系の中に、各宣言の実行順序の制御戦略が組み込まれているため、その分だけ記述が少なくてすむからである。この記述量の少なさは、プログラミング時間を減少させることにもなり、大きな利点であるといえる。

・正当性の問題

ＯＢＪ２は、数学的理論に基づいて設計・実現された仕様記述言語である。したがって、記述の正当性を証明することができる。これは、システムを開発する前に正しさを示すことができるということであり、設計者の知識や経験に依存している現在の設計方法と比較して大きな利点となる。

・仕様の解釈・実行

ＯＢＪ２で記述された仕様は、処理系によって解釈・実行することができる。これに

よって、仕様の段階でシステムの動作や効率等を確認することができる。この点は、ある意味で、プロトタイピングのアプローチをも含むことになるのではないか。

・プログラムの自動生成

正当性を確かめ、動作を確認した形式的仕様からは、（シンタックスもセマンティクスも定まっているために）機械的作業によって、プログラムを自動生成することができる（はずである）。また、人間が手でコーディングするにしても、正しいことがわかっている仕様から生成されるプログラムに虫が含まれることはないであろう。スペルミス等は別にして、設計上の誤りからくる虫は根絶できるのではないか。

・実時間に関する問題

ＯＢＪ２に限らず、形式的仕様記述言語は、現在のところ時間の概念を取り込むまでにはいたっていない。したがって、時間に対して大きな制約を受けているリアルタイムシステムをこれによって記述することは、今のところできない。この点は大きな欠点であり、今後の研究が待たれる点である。

以上の点を考慮して、我々は、形式的仕様のアプローチがソフトウェアというモンスターに対する銀の弾丸となりうると結論した。ただし、以下の点に注意する必要がある。

まず第一に、上でも述べたが、リアルタイムシステムを記述できるような形式的仕様記述言語を開発することである。リアルタイムシステムの設計は、どれだけうまくタイミングを調整できるかにかかっており、熟練設計者の豊富な知識と経験に頼っていることが多いため、これが開発できれば大きな意味がある。そのためには、時間の概念を組み込んだ理論を構築する必要があり、今後の研究課題である。

そして第二に、このアプローチが成功したとしても、プログラマーが必要なくなるのではなく、あくまでも主役は人間であるということである。非形式的（インフォーマル）な現実世界の問題を形式的（フォーマル）な形に書き下さなくては、計算機で実行することはできない。形式的仕様のアプローチは、現在の設計者が行なっている設計のうち、実現レベルに近い作業による負荷を軽減するものではあっても、決して非形式的な問題を形式的なものに変換してくれるものではない。この点は人間の創造的活動に任せるしかなく、これがソフトウェア開発の最も難しい点である。

６．おわりに

今回のプログラミング・ワークショップに参加するにあたって、実行委員のかたがたに種々の点でお世話になりました。紙上を借りて深謝いたします。

参考文献

[1] ”抽象データ型とＯＢＪ２”, bit Vol.20, No.9, Sep. 1988, pp.67-80.

第２グループ：討論レポート

進化型アプローチによるアプリケーション開発

――― ４ＧＬによるプロトタイピング ―――

（株）呉電子計算センター
板倉重利

メンバ（50音順敬称略）

板倉重利　：　（株）呉電子計算センター
桜井麻里　：　（株）ソフトウェア・リサーチ・アソシェイツ
佐藤千明　：　（株）長野協同電算 …… Ｐ．Ｃ．
高山　　：　日本ユニシス（株）
立木勲　　：　（株）三井銀ソフトウェア
穂苅顕　　：　日本テクニカルシステム（株）
篠田健　　：　（株）ディーエム情報システム
望月善光　：　（株）ユニテック

はじめに

会場を仙台から八ヶ岳に移して、初めて行われる夏のプログラムワークショップ、全国から参加者が集まってくる。初参加の人が多いせいか緊張した顔が目立つ。受付を済ませるとプログラム委員長の挨拶のあと、チュートリアル“エキスパートシステム入門”から予定通りのスケジュールでワークショップはスタートした。

パネルディスカッションにおける、ＰＣ佐藤さんの会場に実機まで持ち込んで来るという熱意と、これまで使ったことのない機械と言語を操作したいという欲望から、前述のメンバが集まり第２グループが誕生した。

グループ活動に入り、自己紹介するが、まだ、これから始まる未知への不安のためか表情は硬い。ＰＣ佐藤さん、日本ユニシス高山さんによるＬＩＮＣⅡの説明が進んでくると、ソフト屋としての習性からかＬＩＮＣⅡに対する興味も増し、質問も出始めグループ内の雰囲気もなごんできた。こうして、最終日の成果発表に向けてグループ討論はスタートした。

１．グループ活動の経緯

１日目：　パネルディスカッション後にグループ編成された第２グループは、ＬＩＮＣⅡという新しい言語と進化型プロトタイピングというアプリケーション開発に興味を持つ８名で構成された。

自己紹介で各人の仕事環境を話した後、ＬＩＮＣⅡの機能説明、今後の作業確認を行い、八ヶ岳の一日を終えた。

２日目：　作成システムは得意先マスタ、仕入先マスタ他からなる在庫管理システムである。実機を使って本格的にプログラミングを開始、初めて使う端末のため操作説明を受けながら悪戦苦闘である。慣れるに従って、キータッチも軽くなり、プログラムの入力を一部終え懇親会へと向かった。ほぼ予定通りの作業進捗である。懇親会後もマニュアルを見ながら、夜遅くまでキーボードをたたくメンバもいた。

３日目：　午前中にプログラミングを終えた。午後は在庫管理の実演である。ディスプレイにデータ入力用、在庫問い合わせ用画面が表示されると、メンバのほっとした表情が感じられた。

一通りの処理を終えＬＩＮＣⅡ（４ＧＬ）の実感を携えて中間報告へ臨んだ。

中間報告後、最終日の成果発表へ向けてのグループ討論は、潤滑酒も手伝い活発な論議が展開され、発表用の原稿が完成したとき、窓から小鳥のさえずりが聞こえた。

４日目：　いよいよグループ発表である。発表者二人は、幾分、緊張気味である。当初打合せとは多少異なった発表内容であったが、質疑応答も無事に終え、大役を果たした二人には、安堵感と満足感が伺えた。８月３１日に誕生した第２グループはグループ発表を終えて解散した。メンバはワークショップの成果を土産にそれぞれの職場に戻った。

2．グループ討論のまとめ

４ＧＬ（ＬＩＮＣⅡ）によるプロトタイピングを体感してのグループ討論は、まず４ＧＬ（ＬＩＮＣⅡ）の感想を述べ以下の内容でまとめた。

４ＧＬ（ＬＩＮＣⅡ）に対する感想

（１）画面上に項目を置いていくだけでできてしまうのはすごい。

（２）現状でやっていることはやっている。

（３）手軽にできてしまうことがデメリットになってしまう環境もありそうで疑問。

（４）システムが大規模になったとき耐えられるか疑問。

（５）要求定義のモデル化に対する解決が欲しかった。

（６）ユーザ、ＳＥ、プログラマの定義、役割分担が変わってくるのではないか、ＳＥがプログラマを兼ねなければならないかという不安を感じた。

２－１　４ＧＬの定義

一般論

・抽象度が高い言語である。

・非手続き型記述も可能である。

・データベースも扱っている。

・対話型である。

・端末の有効利用を図っている。

・エンド・ユーザが参加できる。

・プロトタイピングが容易である。

・レポートの出力が容易である。

⇩

グループ内結論

事務処理アプリケーションの高生産性ツール

２－２　プロトタイピングの意義

システム構築する上で、いかにユーザ・ニーズを取り込み、ユーザ・ライクに要求定義が容易にできたらと、どのシステム・エンジニアも考えるだろう。

品質向上を目指しても、品質不良の最大の要因は設計仕様の検討段階にある。

こうした中で、プロトタイピング手法に期待が持たれる！

⇩

なぜか

今までのシステム開発は、本格的に動いてみないと、その評価があいまいである。

⇩

そこで

実際のプロトタイピングにより、体感し、よりコミュニケーションを図り易くし、また、流動的なユーザ・ニーズを効率良くシュミレーションする。

⇩

プロトタイピングへの期待

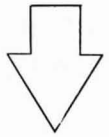
①使い捨てアプローチ：　システムの表面的な動きを検証

②進化型アプローチ　：　システムの検証・拡張そして実システムへの進化

2－3　プロトタイピング実現とその期待

やりたくてできなかったのは？

3GL（COBOL，FORTRAN，……）では、アプローチしていくには、工数，時間が膨大で、プロトタイプしていくリスクが大き過ぎた。



こんな夢みたいな仕掛けが今までにあったか。進化型プロトタイピングの実現

それが、4GLか

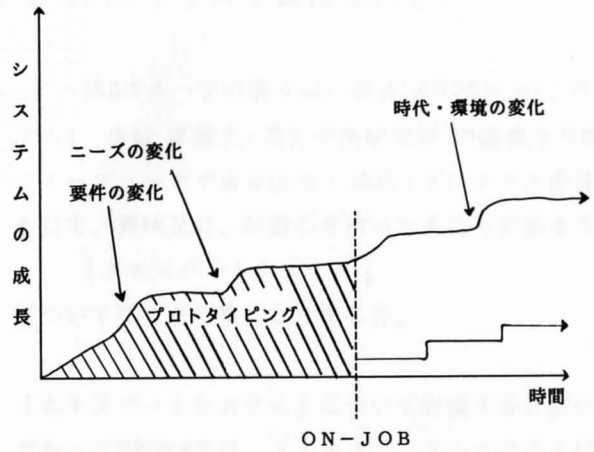
①早く、簡単に動いてしまうものを作ってしまうということでユーザとのコミュニケーションが容易にできる。

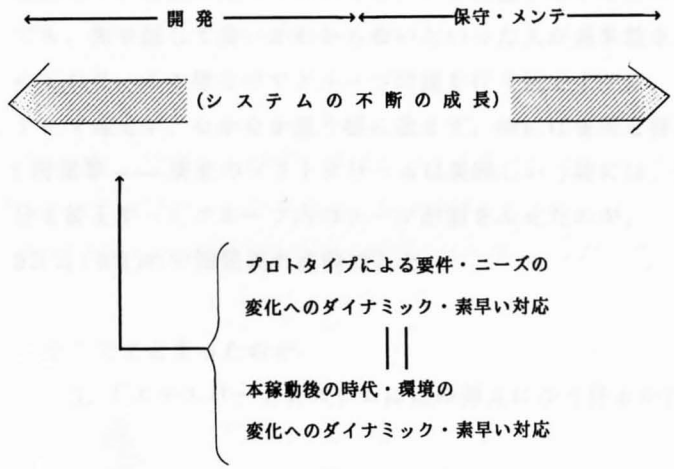
②ユーザニーズの変更を容易にでき、尚かつダイナミックにシステム全体に反映する。

③工数，時間の軽減

④刻々と変化する社会情勢を容易にシステムに反映できる。

2－4　成長型プロトタイピングによる成長型システムの構築





2－5　4GL（LINCⅡ）の課題

（1）パターン内蔵のため柔軟性にかける。

（2）オープンアーキテクチャでない。

（3）DB依存のため、利用方法が限定されている。

（4）既存システムとのインタフェースがない。（汎用でない）

おわりに

第2グループは実機を使用してプログラムを作成しなければならないこともあり、他グループに比べ実務的な作業が多かった。その分、目標も明確になっており討議も白熱したものとなった。

今回のテーマになった“4GLは銀の弾丸になるか”については、4GLと成長型プロトタイピングにより“銀の弾丸”に少し近付いたのではないかということで、グループ討議の結論とした。

3泊4日で行われた夏のプログラムワークショップ、ハードなスケジュールでしたが、最後まで協力と頑張りを忘れなかった第2グループのメンバの皆様お疲れさまでした。

－以上－

第6回
夏のプログラミング・ワークショップ
第3グループ 報告書

(株) 日本情報開発
山田 良一

テーマ

【 エキスパートシステムは "銀の弾丸"になり得るか 】

グループ構成 : 7名

1. 山本 一成 (ケーシーエス)
2. 藤本 良子
3. 前野 貴志 (小松ソフトウエア開発)
4. 笠原 幸夫 (システムプランニング)
5. 末次 真之 (システムプランニング)
6. 中沢 賢光 (日本電機ソフトウエア)
7. 山田 良一 (日本情報開発)

はじめに

今回、3日間にわたり行われた、プログラミング・ワークショップにおいて、我々第3グループの、『エキスパートシステムは"銀の弾丸"に成り得るか』をテーマとした、ディスカッションの様子を報告書としてまとめました。

エキスパートシステムについて!!

我々第3グループの面々は、初日(8月31日)の、チュートリアル1 寺野 孝雄氏(電力中央研究所)の講義を拝聴し更に、グループリーダである山本 一成氏(プログラム委員)の説明会にて、興味又は、好奇心を持った人たちが集まり

『エキスパートシステム』

について語り合う事になりました。

『エキスパートシステム』について討論すると言いましても、グループ7名中4名は、『エキスパートシステムとは何???』雑誌等で、言葉は知ってはいても、いざ討論するぞと言われても、何を話して良いかわからないといった人が過半数を占めており、この様な中でグループ討論を行う訳ですから、テーマ選定が、なかなか想う様に進まず、時には場所を替え(清里等 ------ 清里のソフトクリームは美味しい)時には、気分を替えやっとグループ内のテーマが形をみせたのが、3日目(9/2)の中間発表の直前でした。

そこでまとまったのが、

1. 『エキスパートシステムは銀の弾丸になり得るか???』

2. 『ソフトウエアの開発支援ツールとしての
エキスパートシステム構築』

でありました。

前述した通り我々のグループの過半数が、エキスパートシステム------???であり、それでは実際に、構築し実体験した方が、エキスパートシステムとは何であるか多少なりとも理解の度合いが違うのではと言う事で、本テーマに決定致しました。

さて、テーマが決定すると次は本題に対し討論を行う訳ですが、討論を行うに際し何に対し討論を行うかで、討論をおこない、そこで持ち上がったのが、皆が共通でかかえている、ソフトウエア開発上の共通の問題について語り合う事でした。

皆がかかえる共通の問題点とは大別して、

1. 見積り
2. 納期
3. 開発環境

が上げられました。

これらの問題を解決(又は、問題発生の軽減)して行く為に、まず、なぜこの様な問題がほぼ毎日発生するかについて語り合いました。

出てきた結果が簡単に申しますと、

1. 要求仕様が流動的である事
   (仕様変更等がランダムに発生する)
2. 物理(ハード的)環境等がシステム毎に変わる為、固定的な知識のみでは、要求をクリアする事が出来ない事
3. 組織及び人的要因においてプロジェクト毎(システム毎)に開発メンバーが、流動的であり、更にメンバーへの新人等の参加によりプロジェクトリーダーの心労に計り知れない物がある事
4. 工程においては、前記致しましたそれぞれの条件が、からみ合い当初予定していた工程がなかなかまもれない事が良くある事

等その他様々な要因は上がりましたが、総合すると以上の4項目に含まれる様な事であり、それらの要因に対し少しでも解決の糸口をみつけて上げ様とする目的で、目指したシステムが、

『指摘マン』

であった訳です。

本システム『指摘マン』は、問題の発生防止を主目的として、開発されるものであります。

そろそろまとめに入りますが、我々第3グループが構築しようとした『指摘マン』は、エキスパートシステムの推論機能を用いる事により過去に体験したあやまちを知識ベースとし、それを基に、これから開発しようとするプロジェクトの各条件をインプットし、推論し、結果としてソフトウエアの開発における問題となるであろう点を事前に指摘し対応をうながす為のものであります。

この様に我々第3グループの

『指摘マン』

は、これから世に羽ばたこうとした直前に時間切れとなり未完成のまま、ソフトウエア開発史にその名を残す事は、出来ませんでした。
でも、『八ヶ岳のワークショップ』に参加致しました、数十名の方々には、深く心に残っているものと信じて疑いません、

以上まとまりに欠ける点が、多々あると思いますが、これをもって第3グループのレポートと致します。

（第4グループ討論まとめ）

”オブジェクト指向”による銀の弾丸探し

三菱電機コトロールソフトウェア（株）
内海　保

１．現在抱える問題と解決策

現在、各グループ員がどのような問題を抱えているのかをS／W開発環境、対象分野と共にまとめたものが表１．１．である。

表１．１．開発環境・対象分野と抱える問題点

	機種	ＯＳ	言語	対象	問題点
井上	ACOS-6 ACOS-4	MVX/4.0	COBOL COBOL/S	事務	・プログラム：データ=多：１の矛盾の発生 ・データの変更に対する応答
内海	MX/3000 SUN MAC	UNIX	C	ツール	・ドキュメントの作成と管理
林	NEWS	UNIX	shell awk 簡易言語	エンジニアのＯＡ	・ドキュメント（書類）管理
平尾	NEWS SUN MAC	UNIX	C	ロボット制御 ライン制御	・ロボットの進化への対応
藤村	HITAC Mシリーズ Lシリーズ	VOS3 MIOS7	COBOL	市町村業務	・ドキュメンテーション ・先進技術からの遅れに対する危機感
森	HITAC Mシリーズ	VOS1 /ES2	COBOL 簡易言語	事務	・システム分析、設計の標準化

この表に示す通り各自の抱える問題が様々であったため、まずは、オブジェクト指向の概念が問題解決に役立つかどうかを考えながら、各問題の解決策の検討を行なった。以下、各問題毎にその結果をまとめておく。

（１）数種DBにアクセスする複数プログラムのメンテナス性の向上

問題点

数種のDBにアクセスするプログラムを複数製作したが、設計が個別に行なわれたために、DBへのアクセス方法がプログラムにより異なっている。DBの形態が変更されたときの既存プログラムのメンテナンスが大変である。

この問題に対しては、まずデータ抽象の概念に基づきプログラムを再構築すべきであるとの結論であった。具体的には、使用する言語がCOBOLであるとのことであるから、各DBを操作する手続きを明確に定義し、各プログラムは定義したDB操作手続きによってのみDBを操作するようにするということである。

（２）UNIX上のドキュメント管理が煩雑

問題点

現在UNIX上にて作成したドキュメントを管理しているが、この管理が煩雑である。
例えば、管理しているドキュメントを出力しようとした場合、出力コマンドがドキュメント毎に異なっており、素人では出力できない。

これについては、UNIXの既存コマンドにて問題解決できる。

（３）ドキュメント作成・管理の効率化

問題点

ドキュメントを如何に効率良く作成し、管理するにはどうすれば良いのか。

ドキュメントの問題は、レベルの違いはあれどもグループ員共通の問題であった。とにかく計算機にて作成・管理を行いたいのだがという声に対して、次の二つの環境が示された。

・roffの環境

S/W	troff, tbl, eqn, pic	RCS	make	その他UNIXコマンド
H/W	UNIXワークステーション			

・Macintoshの環境

S/W	EgBook, EgWord,MacDraw,... etc.
H/W	Macintosh

大量の、あるいは、同じ書式のドキュメントを何回も作成する場合には、S/Wエンジニアとしては、roffの環境のほうが便利である。

2．「鉄の弾丸」探し

各自の抱える問題の解決策を検討した結果、オブジェクト指向を用いた効果的な解決策の存在する問題はひとつもなかった。この原因としては、次の二つが挙げられる。

(1)問題領域が、もともとオブジェクト指向を適用しなくても解決できる領域であった。
例）ドキュメンテーションの問題。

(2)オブジェクト指向に対する参加者の理解不足で、解決策に適用しきれない。
システム分析・設計の標準化という問題に対しては、適用可能な気はするが、どのように適用していけばよいのかわからない。

ここで、我々は(1)のような問題領域に対しては、現時点での知識・技術レベルにて実施できる対策がいくつもあることを再確認した。それらの対策を実施することにより、S/Wの生産性を少しでも高める事ができるであろうということから、それらを「鉄の弾丸」と称することとした。以後、しばらくは「オブジェクト指向」からはなれ、「鉄の弾丸」探しに熱中する。（(2)の問題については、3．で触れる。）

2．1 目指す「鉄の弾丸」

グループ員を取り巻く環境（表1．1．参照）の違いから、あるところでは解決の糸口さえ見つからないないような問題であっても、別のところでは解決済みの問題であることがある。このような場合には、前者は後者よりアドバイスしてもらうことにより、たちどころに、深刻な問題を解決することができた。つまり、情報交換のためのネットワークが大いに役立つというわけである。

ここでいうネットワークは、次の二つのネットワーク両方を指す。

(1)コンピュータネットワーク
mail,BBSを使用して必要な情報を得ることができるし、他の人のアドバイスを受けることができる。

(2)人とのネットワーク
情報を交換し合ったり、アドバイスを受けたりして、普段からいろいろな人と話すことにより、視野を広くしておける。

次に情報だけを得られたとしても、物理的な問題（例えば、問題解決に必要なWSがない）により問題を解決することができない場合がある。このような場合には、当然物理的な問題を解決しなければならないが、この時に問題になるのは以下のことである。

(1)政策的な問題
提携関係がどうのこうのといった問題や、構築したい環境が決定された環境構築方針に合わないといった問題。

(2)金銭的な問題
この場合、本当に余裕がない場合に限る。単に予算が降りないのは、(1)に含める。

問題(2)は仕方ないにしても、問題(1)を解決するためのキーとなるのがマネージャ（権力者）である。マネージャーの同意がなければ、解決することはできない。

マネージャーは、この他にも重要なキーとなる。何か新しいことをやるには、時間がかかるし、必ずしも成功するとは限らない（最善はもちろん尽くすが。）。こんなときに、マネージャーの理解がなければ、思う存分に極めていくことはできない。

一エンジニアとして必要な「鉄の弾丸」をもう一度列挙すると、次の二つである。

(1)情報交換のためのネットワーク
(2)理解あるマネージャー
┣ 良い環境

3．オブジェクト指向

オブジェクト指向の話題に戻り、オブジェクト指向とはどのようなものであり、どのような分野に応用できるかについて話合った結果をまとめる。

3.1　オブジェクト指向適用の試み

グループ員からロボット制御プログラムの製作にオブジェクト指向を適用しようとしているとの提示があった。この話題は、第一日目に討論したものであり、2章の「鉄の弾丸」探しと前後するものである。

(1)適用対象のロボット制御プログラム

適用しようとしているのは、4軸のアームロボットの制御プログラムであり、このプログラムは、図3．1で示される構造をしている。

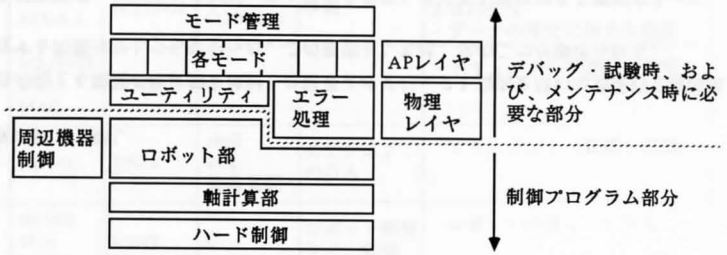


図3．1　ロボット制御プログラムの構造

ロボット制御プログラムは、大きく分けて、実際にロボットの動きを制御する制御プログラム部分と、その部分をデバッグ・試験したり、メンテナンスするときに必要な部分の二つに分けられる。

また、各部分は機能のかたまりにて分割されている。

(2)オブジェクト指向の適用の可能性

制御プログラム部分にオブジェクト指向の適用を考えると、この部分は次のようにモデル化できる。

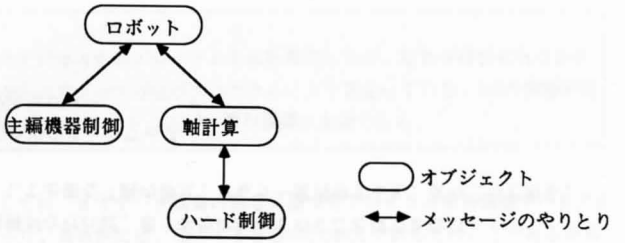


図3．2　制御プログラム部のモデル化

この図のオブジェクトに対するクラスを設定することにより、制御パラメータが少し違っているような場合に迅速に対応できる。例えば、軸計算のクラスでは、軸計算の手続きをメソッドとし、制御パラメータをそのクラスのインスタンスのデータとすれば良いであろう。

また、この部分にオブジェクト指向を適用すれば、ロボットの形態が異なっても（例えば、軸が6軸になっても）、クラス間の継承関係を利用することにより迅速に対応できると思われる。

3．2 オブジェクト指向に対する疑問

オブジェクト指向の適用を考えたが、オブジェクト指向そのものがよくわからないという意見があり、オブジェクト指向の体験学習と言うことで、HyperTalkによるプログラミング実習を行なった。Hyper Talkは、Macintoshで動くオブジェクト指向言語である。それと合わせて、オブジェクト指向に対する疑問点を洗いだし、その解答をPCの佐原氏に行なってもらった。

以下、オブジェクト指向に対する疑問点とその解答を簡単にまとめる。

(1)オブジェクト指向の適用が有効な分野は？

同じようなアプリケーションを何度も開発する分野。このような分野では、プログラムジェネレータの適用も有効であるが、メンテナンスを考えたとき、オブジェクト指向では、個々のプログラムでなく一つの枠（クラス）を手直しすればよいという点でオブジェクト指向の方が勝っている。

(2)オブジェクト指向型言語が利用できない環境のときにはどうするのか？

プログラミング言語によっては、オブジェクト指向を実現するようなプログラム作法が存在する。例えば、C++はCのプリプロセッサであるので、Cでオブジェクト指向を実現するプログラムを作成することができる。

4．今回のケーススタディで得たもの

最後に今回のケーススタディでの結論と成果をまとめる。

(1)人と話すだけで解決する問題もある。そのためには、情報交換のためのチャネルを持たなければならない。
（コンピュータネットワーク、友達の輪、SEA）

(2)新しい技術により、S／Wの生産性を向上させられる。そのためには、新技術の試使用ができる環境でなければならない。
（キーパーソンの抱き込み、マネージャやユーザの意識改革、密かな実績を積み重ねての説得）

(3)オブジェクト指向がどのようなものか、うっすらとではあるが理解できた。今回、受けた刺激を減衰させることなく、今後に生かしたい。

第４グループ

PC　佐原　伸（ＳＲＡ）

メンバー　井上　文人（早稲田大学事務システムセンター）
林　好一（ＳＲＡ）
平尾　一浩（ヒラタ・ソフトウェア・テクノロジー（株））
藤村　洋（（株）岩手電子計算センター）
森　博幸（（株）日立ビジネス機器）
内海　保（三菱電機コントロールソフトウェア（株））

****

グループ討論　　＜オブジェクト指向＋α＞

参加者　　坂下秀　　（プログラム委員）
沢田寿美（プログラム委員）
王生幸雄
田中慎一郎
岡田正之

作成者　　岡田正之

****

私達のグループは、今回のテーマである、ソフトウェア開発の「銀の弾丸」を求めて、オブジェクト指向＋αによるアプローチを行いグループ討論を行いました。

まずは、各自がどうしてこのグループを選んだのかと、今の自分の仕事で抱えている問題は何かということを発表し、その問題点を解決する為にはどうすればいいのかをみんなで考えてみました。

王生さんは、ソフトウェアの利用者が、まったくの素人の為に、間違ったオペレーションを行ってしまった時のリカバリー処理がとても多く、気を使っていると云っておられました。私は、いつも感じている、ユーザとの対応の問題を提議しました。

各自の問題点の討論が済み、私達のグループでは、ワープロの設計をしてみることになりました。ワープロ設計の目的としては、今まで自分たちが行ってきた方法をもって作業を割り振り、その割り振り方に問題はないのかなぜその様な設計方法をとっているのかをメンバー間で討論してみようというものでした。さらに、プログラム委員の方々が考えてみたオブジェクト指向的アプローチによる設計との比較を行い、メンバー間の討論で発生した問題点が解消されるかを考えてみようというものです。

まず各自が、よく使うワープロ等を頭に浮かべて、ワープロの設計を始めました。しかし、これが普段使っているワープロでも、いざ設計するとなると時間が短かったせいもありますが、なかなかみんな苦労していたようでした。各自の設計がとりあえず一段落したところで、３人の設計を１つにまとめることになり、各自の設計案を順番に説明し、わからない所とかを質問して３人により統一した設計案を作成してみました。

この段階に入り、各自の想い浮かべたワープロのイメージが既に違う事や設計による機能分け等も既に違うことがわかりました。

さて３人により統一された設計案ができた所で、プログラム委員の方々が考えたオブジェクト指向的アプローチによるワープロ設計との比較検討を行いました。私達が考えた設計の特色は、機能分けによりアプローチであったということで、まだまだ、細部まで考えていく段階で多くの矛盾を含んでいるということでした。

プログラム委員の方々の設計は、私達が考えたものとはまったく違い、ワープロを作成するという考え方からのアプローチではなく、文書作成をうまく行うにはどうしたらよいかというアプローチの仕方でした。

プログラム委員の方々は、まず、文書モデルを考えて、実際の文書作成にあてはめて必要な機能が実現できるか確かめるという手順で進められていました。

委員の方々が、設計をやってみて、感じたと言っておられたことは、文書の編集が自然にできる、自由度が高いのでいろいろな発想ができる、シュミレートがしやすい等の利点をあげられていました。また、欠点としては、実現が難しい、抽象度が高い等をあげられていました。

さて、最後に私達が討論と委員の方々のお話を伺って感じた事は、

田中さん、
・予想以上に時間がかかって細かい話まで整理できなかった。
・文章モデルを考えずに始めたのは良くなかった。

王生さん、
・ 発想の違いを感じた。
・ 自分の考えてものは、絶対動かないというよりも基本的に作れるのか否か疑問を感じた。
・ 「機能的に分割する」という事が、難しいということが判った。

私、
・ 型にはまった考え方ではなく、柔軟な考え方を持たなくてはいけない

事を実感しました。

ということでした。

第6グループ:討論レポート

作成　三菱電機セミコンダクタソフトウエア株式会社　蟇峨山　亨

渡邊　雄一　　コーディネーター
伊藤　達哉　　グループリーダー
香西　哲也
後藤　浩
中林　信博
中山　聡
西川　典子
方　学芬
吉田　慎哉
蟇峨山　亨

1　はじめに

当グループでは、"CASE ツール，オブジェクト指向について考える"というテーマで討論をおこなった。

2　討論の概要

近年のソフトウェア開発手法に関する議論は、従来のようにソフトウェアの評価やドキュメント整備といったソフトウェア開発の下流工程だけでなく、仕様設計および要求分析等ソフトウェア開発の上流工程に対しても行われるようになってきている。

特に、CASE(Computer Aided Software Engineering)ツールを利用した仕様設計およびオブジェクト指向の概念を取り入れた仕様設計は近年最も注目を集めている分野であり、今回のワークショップでも公開テーマとして取り上げられている。しかし、CASE ツール，オブジェクト指向とも開発現場ではまだまだ馴染みの薄いものであり、それぞれの有効性は未知数である。

そこで当グループでは、下記の 2 点について討論および実習を行い、CASE ツールおよびオブジェクト指向がソフトウェア開発の"銀の弾丸"となるかどうか検討した。

(1). CASE ツール StP を用いた図書館システムの仕様設計

(2). オブジェクト指向による図書館システムの仕様設計

以下に活動の詳細について報告する。

3　StP によるシステム仕様設計

CASE ツールの 1 つである StP(Software through Picture)を机上で使用し、CASE ツールの有効性を探ってみた。最初に StP を使用した経験のある方氏に StP について概説していただき、続いて方氏の指導のもとに（人間 StP）図書館システムの仕様設計を行ってみた。

3.2.2　データフロー図の作成

最初に、概略データフロー図（レベル 0:図 2）を作成した。ここでは、図書館システムを中心に窓口および図書管理者との間に発生するデータの流れを把握することにした。

次に詳細なデータフロー図（レベル 1:図 3)を作成した。すなわち、図 2 で library system とした部分の詳細なデータフロー図を作成した。データフロー図による仕様設計では、このように、まず概略のデータフロー図を作成することでシステム全体仕様の見通しを立て、順次各プロセスの詳細データフロー図を作成していく。

StP を用いると、レベル 1 のデータフロー図を作成する際、レベル 0 で作成したデータフローと矛盾が生じれば、StP からその旨の警告が発せられるので、修正をかけることになる。これにより、初めに捉えた図書館システムの概略イメージ（レベル 0 のデータフロー図）と矛盾のない詳細仕様設計を行うことができる。

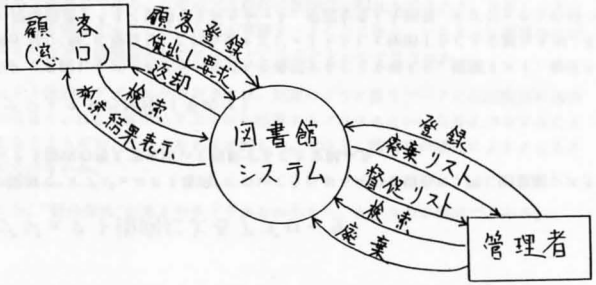


図2　概略データフロー図

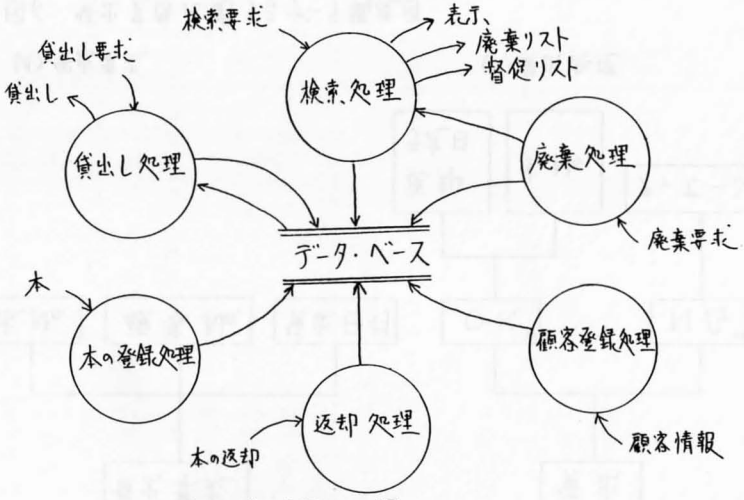


図3　詳細データフロー図

3.2.3 E-R 図の作成

データフロー図では、データ構造の詳細を捉えきれないところがある。そのため、StP では、データフロー図以外にデータ E-R 図またはデータ構造図を記述し、データ構造に矛盾が生じていないかどうかをチェックする機能を持っている。

そこで図書館システムに関する E-R 図を作成した。図 4 はその一例である。ところで、図 4 の例において、貸出年月日，返却予定日，返却日，借り手についてはその本が貸し出しされるたびに繰り返し記録が追加されるものである。しかし、このような繰り返し処理に対する表現方法を E-R 図は持っていないことに気がついた。

そこで、データ構造図(Data Structure)を作成することにした。作成例を図 5，図 6 に示す。 以上のような過程で StP による図書館システムの仕様設計を行った。レベル 0 のデータフロー図を記述する際に多少の戸惑いがあったものの、案外順調に設計を行えたものと思っている。

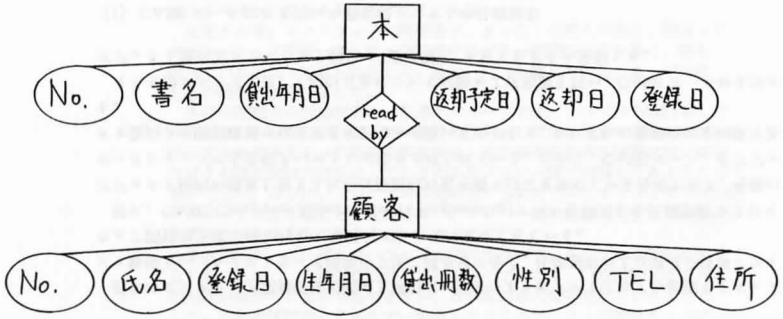


図4 E-R図

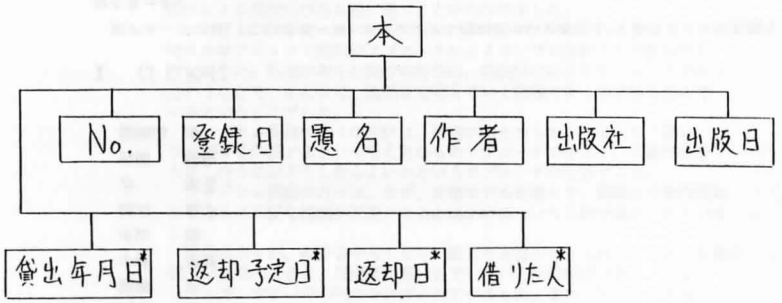
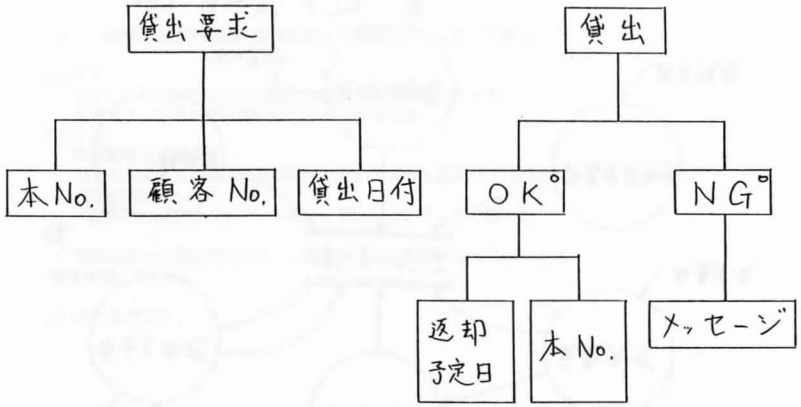


図5 本に関するデータ構造図



(A) 貸出要求　　(B) 貸出処理

図6 貸出業務に関するデータ構造図

4 オブジェクト指向によるアプローチ

2 日目の講演で"オブジェクト指向"についてうかがったのを機会に、同じ図書館システムをオブジェクト指向の考え方を用いて設計することを試みた。

4.1 オブジェクトの抽出（その 1）

まず、StP で考察したデータフローモデルを適用することを考えた。廃棄リスト，督促リスト，検索表示等の"物"に着目し、これらをオブジェクトとして抽出することを試みたが、オブジェクトの階層構造をうまく表現しきれなかった。原因を考えた結果、オブジェクトの持つアトリビュートについて全く考慮せず、データフローのみに着目していたため、仕様の表現ができなくなってしまっていたことに気がついた。つまり、オブジェクトの包含するデータセットについての吟味が不十分であったのである。

そこで、再度オブジェクトの抽出を行うことからやり直し始めた。

4.2 オブジェクトの抽出（その 2）

今度は、"本"に着目し、本を中心に顧客と管理者との相関関係を捉えようと考えた。図書館システムでは本に関する関するデータを中心にシステムの運用がなされる。従って、"本"オブジェクトをシステム仕様設計の中心に据えればオブジェクトの階層構造がきれいに表現できるのではないかと考えたわけである。

しかし、やはりオブジェクトの階層構造をうまく表現しきれていないようで、延々と議論が続けられたが、決定的な構造図は得られなかった。

ここで、第4グループのコーディネータ　佐原氏より"ライブラリアンをオブジェクトの中心にすえてみてはどうか"とのアドバイスを受け、ライブラリアンオブジェクトを導入し、図7のような階層構造を考えてみた。

しかし、結局すっきりした構造にまとめることができず時間切れとなってしまった。

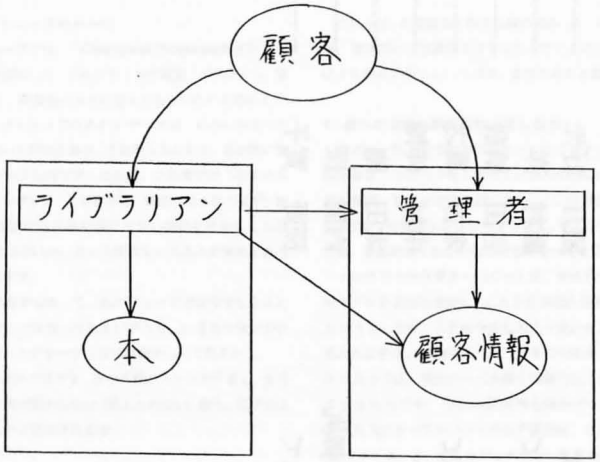


図7オブジェクト抽出例

4.3　討論内容の反省

今回のオブジェクト指向による設計がなぜうまくいかなっかたのだろうか。その原因の1つとして"オブジェクト"の捉え方が良くなかったという点があると思われる。

私たちは、オブジェクトを単に1つのデータセットと手続きとして捉えてしまっていたのではないだろうか。つまり、オブジェクトとは

(1). 自分の持つ内部情報が、読み出し，変更の対象になる

(2). 自分自身または他のオブジェクトにメッセージを送る（何らかの作用を行う）

(3). メッセージの送り主にあるオブジェクトを応答として返す

ものであるという概念をいま一つのみこめていなかったからではないだろうか。従って、たくさんの"物"を抽出したものの、個々のデータセットの取扱いにばかり注意が払われてしまっていたのである。例えば、本の名前をどのように処理すれば良いのか、とか貸出し記録をどのように記録すれば良いのか、といった具合いである。その結果、各オブジェクトの関係が混乱してしまったのではないだろうか。

また、データセットのクラス化をうまく行えていなかったことも原因と考えられる。確かに本やライブラリアンといったオブジェクトが持つデータセットを考えているが、1つのオブジェクトとしてまとまりのあるデータセットに絞り切れていなかったのでは、と思う。共通情報は何か、他のオブジェクトから完全に隠ぺいされる情報は何か、ということに対する吟味が不足していたことは否めない。

5　まとめ

今回のワークショップでは、CASEツールの使用を体験し、オブジェクト指向的アプローチを試みたという点で有意義であったと思う。ただし、程度の差こそあれ各メンバーは次のようなことを考えていたのではないかと思われる。

(1). CASEツールを利用すれば、データフロー図等の様々な図がCRT上に記述できるため、図の変更が大変容易である。この点では設計の効率化が図れるであろう。ただし、データフロー図等を構築することは、人間が考察すべきことであり、システムに最適な仕様設計を行うにはその開発者の能力によるところが大きいのではないか。

(2). オブジェクト指向的アプローチを行うには、従来のように扱うデータの相関関係に着目するのではなく、むしろ各データとそれを処理するプロセスといった捉え方をすることが必要となるようだが、これがなかなか難しい。やはり、開発者の能力によるところが大きいのではないか。

つまるところ、"銀の弾丸"があるのかどうかわからない、というのが現状であろう。

ポジション・ペーパ & 感想

八ケ岳夏のプログラミング・ワークショップ	澤田　寿美	----------	25
ワークショップの印象	渡辺　雄一	----------	26
ソフトウェア工学は役に立たないか？	佐原　伸	----------	28
ポジション・ペーパ	藤本　良子	----------	30
八ケ岳ワークショップにて	山本　一成	----------	30
夏のプログラムワークショップに参加して	板倉　重利	----------	32
分かりやすいソフトウェア	中沢　賢光	----------	33
夏のプログラムワークショップ	山田　良一	----------	34
人を中心に	井上　文人	----------	35
銀の弾丸とは	伊藤　達哉	----------	36
問題設定の視角	立木　勲	----------	37
ソフトウェア開発における生産性の向上及び品質の改善について	香西　哲也	----------	38
ソフトウェア・エンジニアリング技法のケース・スタディ	平尾　一浩	----------	40
つれずれなるままに	中山　聡	----------	42
若手最後の年に若手の会に参加して	田中　慎一郎	----------	44
ポジションペーパー	岡田　正之	----------	45
ワークショップ参加の後に思うこと	林　好一	----------	46
ワークショップに参加して	森　博幸	----------	47
異文化体験	桜井　麻里	----------	48
ソフトウェア開発プロセスの探求	嵯峨山　亨	----------	50
夏のプログラミング・ワークショップを終えて	望月　好光	----------	51
要求仕様定義からの自動コーディングを求めて	藤村　洋	----------	52
ワークショップに参加してみて	吉田　慎哉	----------	54
夏のプログラミングワークショップに参加して	末吉　真之	----------	55
ポジションペーパ	中村　伸博	----------	56
形式仕様の将来	太田　剛	----------	57
銀の弾丸探しから得た鉄の弾丸，銀メッキの弾丸	内海　保	----------	58
夏のプログラミングワークショップに参加して	深野　寛子	----------	60

八ヶ岳夏のプログラミング・ワークショップ

澤田 壽實
SRA

1. ワークショップのテーマ

私のグループでは，" Conceptual Programming"というネタを出発点にして，プログラミング言語，パラダイム，環境について，参加者の方々に考えてもらいたいと思います．今回のワークショップのメイン・テーマは，いろいろなパラダイムを用いて例題を解き，それをもとにして，具体的に検討しようというものです．これは，これまでの「若手の会ワークショップ」が，ともすると，焦点がはっきりせず，抽象的で，直接には仕事に役立たない議論に終わることがあったのではないか，という反省もいくぶんか含めてあるように思えます．

そうしたなかにあって，私のグループではややもすると抽象的なテーマになってしまいがちで，いささか気がひけるのですが，1グループくらいは勘弁してください．

さて，このネタ振りを，かなり雑にスケッチすると，当日まで本人の気が変わらない（変えられない）限り，以下のようになるものと想定されます．

2. 生産性は向上したか？

少しずつではありますが，ソフトウェアの生産性が向上してきているということについては，大筋では首肯していただけると思います．では，何がこれをもたらしでいるのか？

プログラミング・プロセスに重点をおいて，ソフトウェアの生産性を上げるという，旧来のソフトウエア工学で主張されていた方法は，思った程効果がないことが解っています（違っていたらご免なさい）．ソフトウェアを生産している現場に目を向けてみると，長年に渡り，似たパターンのシステムを開発してきたことによって，実践的な技術が，管理面も含めて蓄積されてきています．つまり，その世界での「常識」が形成されてきています．

一方，研究成果も，それらを具体的に反映した言語や環境として，それなりに開発現場に普及してきています．また，新しい考え方を提供する言語や環境という形で，あるいはいろいろなメディアを通した情報の流通という形で，開発現場に多くの刺激を与えてきています．

これらは，生産道具と利用技術の進歩，多くの新しい情報が，産学問わず流通するようになったことによる相乗効果による部分が多いというのが，真相であると思います．

3. 優れた言語と開発環境が最も重要

最近は，簡単な手持ちのパターンでこなすことのできる仕事は，かなり少なくなっているのではないでしょうか．超大規模，複雑な組込み形システムという最近の傾向もありますが，比較的小さなシステムでも，今までの経験や技法では，直接対処できないものが多くなってきています．

プログラマの仕事を一口でいえば，対象とする問題領域のモデルを正確に構築し，これを計算機の世界に実現する，ということです．これは今後もあまり変わることはないと思われますが，一般的にいって，今までの延長上にある枠組みによっては，満足のいく品質と効率ではこなせなくなっていきそうです．なにか革新的な物がでてくるまでは，我々凡人にとってのソフトウエア開発は，さらによい言語（パラダイム）と，優れたプログラミング環境を頼りとするものになります．

4. Conceptual Programming System

それでは，優れた言語，環境とはいったいどのようなものでしょうか？　タフな質問で，一意な解答はありません．

具体的（?）なアプローチのひとつとして有望だな，と考えているのが，Conceptual Programming というキーワードです．これは当然ながら，私のオリジナルな考え方ではありませんし，取り立てて新しい考え方でもありません．より問題領域に近いレベルでのプログラミングを目指すという点では，他の流行のパラダイムとなんらちがいはないと思います．

ただいえるのは，プログラマの持つ概念モデルを，特定の言語に翻訳することを強いず，それらを直接的に扱うことを可能にしようとしている，ということが特徴だと思います．従って，ある特定のモデル化手法や，方法論を提唱するものではありません．むしろ実践的なプログラミングに対するアプローチであると捕えています．

4. 1. プログラマの使う言葉

多くのプログラマは，システムやプログラムを設計するときに，表現形態として絵とテキトの両方を使って（他にもあるのかもしれませんが），いろいろなモデルをつくります．最終的には，これらの概念化の結果を，あるプログラム言語（複数の場合もあると思いますが）に変換しなければなりません．さらに，開発の途中では，プログラム・テキストとそれらに対応する概念的なモデルの表現との間を，行ったりきりする必要があります．多くの問題がここで発生することは，よく知られたことです．

4. 2. Beyond the Programming Language

このため，できるだけ問題領域を自然に表現できる抽象度の高いレベルでプログラミングできないものかと，努力されてきたわけです．より高級なプログラミング言語，新しいパラダイム，統合化されたプログラミング環境，自動プログラミング等，大枠では目指すところあまり変わらないのではないかと思います．しかし，それにしても最終的には，特定のテキストベースの言語で表現しなければなりません．これはいやだ，という発想です．

4. 3. 楽しいプログラミング

唐突ですが，プログラミングは，楽しいものでなければならない，という主張があり，これを相殺するような言語や環境は，優れているとはいえないという見解がありそうです．「楽しい」というのは，究めて主観的な感覚ですから，全ての人を満足させるような物はあり得ません．各自好みの言語や環境を使って，プログラミングすればよろしいのですが，諸々の事情によってこれは許されません．

これはひとつの例なのですが，単一の枠内で，ソフトウェアのような複雑なものを組み立てることは，不可能だということは明らかです．実際上困難をもたらすのは，管理やコストといった意味合いも含めて，ベースとしているものが全く異質であったり，他の物への変換が困難であると，対話や統合が不可能に近くなってくるといった，本質的な理由であると思います．

こうしたことによって，ひとつの基盤に立ったカスタマイズや拡張が可能な言語や環境が望ましい，という考え方がでてきます（短絡的ですが）．

4. 3. Conceptual Programming System

Conceptual Programming System の目的は，ベースとするモデル化手法や方法論がなんであるにせよ，プログラマが創った概念モデルを，直接に扱うことを可能とすることです．また新たなモデル化手法や方法論を実験するベースとなりうる，プログラマブルな環境を提供することです．Conceptual Programming System は，基本素養として自己拡張的であり，どんどん問題領域の「言語」を取り入れていくものとも考えられます．

5. ワークショップでは

それでは，このような考え方を実際にサポートするシステムは，どのようなものになるでしょうか．他のプログラミング・パラダイムや技法は，どのように位置付けられるのでしょうか．ひそかに隠れたベースとするパラダイムがあるのではないでしょうか．ワークショップで考えてもらいたいのはこれらのことです．

もちろん反対意見がだされたり，異なった考え方がでてくるとしたら，若手のワークショップとしてこれ以上の喜びはありません．

ワークショップの印象

渡邉　雄一
㈱電力計算センター

1．はじめに

のっけからずっこけて申し訳ないが，普段から何も考えずにただ漫然と仕事をしている「旧世代プログラマ」が，何故か昨年と同様にプログラム委員(PC:Program Commitee) に指名されてしまった。今年は辞退させて頂こうと思っていたのだが．．．（どうも反面教師の役割を期待されているようだ！）。私個人としては，昨年とは違った意味でいろいろと勉強させて頂けたと感謝している。しかし当然のことながら，他のＰＣのような「切り札」を持ち合わせているわけではなく困ってしまった。そこで以下では印象に残ったこと，及び参加要領にもあったブルックスの『銀の弾丸はない』［１］に関する私見を記すことで私のポジションペーパとさせていただく。

2．今年こそは小判ザメ！

ポジションペーパを書くべく『銀の弾丸はない』を読み始めたものの，いっこうに自分の意見がまとめられない。「なんと情け無い」と思いつつも，「今年こそは自分のために勉強に行くんだ！」と勝手に決め込んで（何の予習・準備もすることなく）ワークショップの当日となってしまった。

その初日の昼食は，ＰＣの顔合わせを兼ねたものであったが，予定されていたメンバが２人（岡芹，松下）も欠席するという幕開けとなった。今回のテーマに関しては特に準備らしき事をしてきていないだけに，内心ハタと困ってしまったが，昨年もかなり悲惨な状況を経験していたので「何とかなるさ」の気持ちでＰＣのグループ分けに臨んだ。私は，実行委員長の藤野さんの勧めもあって，方さん（静岡大）のＳｔＰ(Software through Pictures) を題材としてＣＡＳＥ(Computer Aided Software Enginnering) を考えるというグループの世話役として以後を過ごすこととなる。（何とか目論見は成功したのだが，果たして私にＣＡＳＥの何を喋れというのか？．いやいや私は勉強しにきたのだ．）

１日目の夕食（ＰＭ５：００～６：００でまさに“夕”食である）のあと，ＰＣのパネルディスカッションの開始である。私は当然のごとく飲んでいる。（実はこの時飲んだ「泉郷」というグリーンボトルの地酒が辛口の「純米酒」で大変おいしかった．だが，以後はグループ・メンバが気を利かして買ってきてくれた「醸造用アルコール入り」の酒を飲むこととなり，ついに２度とこの酒を口にすることなく八ヶ岳をあとにすることとなるとは，神のみぞ知ることであった）　さて一通りのことが済んで，グループ分けになる。なんとまた，私のグループに８人もメンバが集まってしまった。参加者全員の頭数から単純に計算しても５人だし，私のようなどうしようもないＰＣに集まるわけもないと高をくくっていたら．．．。正式なＰＣではないのでパネルスピーチをしていないのに，なぜみんな方さんの魅力を知っているのなぁ？。

3．今年のアプローチ　－オブジェクト指向だぜぃ！－

広い意味でのソフトウェア開発における問題点と解決策について「ＣＡＳＥ」というキーワードを通して考えることがねらいであったようだ。このように書くとプログラム委員として不真面目極まりないようであるが，そもそもＣＡＳＥが「あたりまえ」として利用されていない現在，いたしかたないのではなかろうか？。現実にＣＡＳＥツールとして商品化されているものの多くは，いままであったツールに対して最新のＭＭＩ（特にビットマップ対応のマルチウィンドウなど）へ対応するための皮を被せたにしかすぎないものが多い。だから今現在の国内のソフトウェア・ツールといわれているものの多くをみても，個々の作業工程の部分的改善には成りうるが，より大局的な見地からの評価はすべきではないような状況であろう。そのような中で，今までのツールとは異質な要求定義やシステム設計の工程を（ソフトウェア工学での成果を踏まえた上で）支援する汎用のツールとして，アメリカで開発されたＳｔＰの話は，ソフトウェアの開発をどうすべきかという問題に，現場で抹殺されているものに多くの示唆を与えてくれるものであった。しかしながら，いかんせん実態の無いＣＡＳＥに関する議論などというものは虚しいものである。特に，ＣＡＳＥなどというもの以前にソフトウェア・ツールによる開発すら十分に行われているとは言えない状況下で，いきなりＣＡＳＥのようなより進歩的なツールを議論することには，やはり無理があったことは否めない。しかしそこを「若手の会」の特権である若さ・バイタリティーでカバーし，そこに実際のツールがないことなどさしたる苦ともせず活発に討論・意見交換が出来たことは特筆に値すると言えよう。今回の討論では，ソフトウェア工学で言われているツールや技法の補足する意味からも，要求定義からシステム設計への橋渡しとなる過程での典型的手法による机上シミュレーションを行うことで，ＣＡＳＥツールの必要性・有用性に対する認識を深めるというアプローチをとることが出来たと言う点で，非常に有意義であったと私は勝手に解釈しているが，グループ・メンバ諸氏はいかがであったろうか？。

以下に私がＣＡＳＥを用いた要求定義支援において，個人的に興味のあることを列挙しておく。

・マルチウィンドウの上での「マルチシート」による仕様およびプログラムの記述
・フロー図やブロック図などの「トポロジカルなパターン情報」の検索
・仕様変更の正確な記述支援（特に要求変更とその影響の差分と合成の表現と表現情報の管理）

今年のワークショップの一番の特徴は，いままでになくプログラムが充実していたことだろう。特に２つのチュートリアル

寺野　隆雄（電力中研）：エキスパートシステム入門
柴山　悦哉（東工大）　：オブジェクト指向について

はいずれも興味深いものであり，期待通りであったと言えよう。寺野さんのチュートリアルは，世の中でＡＩブームが下火になりつつあるだけに，本当の問題はどこらへんに有りそうなのかを冷静に聴くことが出来たのが良かった。ただエキスパートシステムの多くは現在ではもう裸で作って・使ってどうこうという時代ではなく，如何にその中身を充実されるかという時代なので，内容の深さからも性質からも興味の持てない人が多くなりつつあるよう感じたのは私だけであろうか。

その一方で，柴山さんのオブジェクト指向のチュートリアルに刺激される形で私のグループもそれに染まっていったのは，私の影響の受けやすさ（＝バカさ加減！？）を見るようで，半ば恥ずかしくもあったが。システムの（要求定義を終え）設計の段階において，今まで幾つかの設計方法論があった。そのなかに新たに（まだ方法論とはいかないまでも）オブジェクト指向的な考え方で要求仕様を整理し設計に生かすというアプローチが良いという噂は，どこからともなく伝わってはいたものの，実際オブジェクト指向などというと「ＡＩだ！」「いやエキスパートシステムだ」などという世界の人の話のような気がして，ついつい敬遠していたのが私個人の実情である。勿論，要求定義の分野でもそのような考え方が，重要でかつ有用であることは幾つか指摘されていたが，具体例としては必ずしも私の理解を促進するようなものでは無かったために，食わず嫌いで済ませていた感が強い。しかし実際オブジェクト指向が何であるかを一通り聞いたあとで，さらにＥＲ(Entity-Relationship) モデルによる分析を考えたあとで，オブジェクト指向による分析を通して，物事の取られ方（視点の変化）によるシステムの表現の多様性を実感出来たことは大きな成果であったと言えよう。

4．なにが心易い者を突然狼男にしてしまうのか？

このことを考えなければ「銀の弾丸」は手に入らないのではなかろうか？。現在のソフトウェア開発の現場では，ある程度ソフトウェアが出来上がると（良くも悪くも）それを一人歩きさせて悪さをしてから戒める傾向があるように思う。実際の不都合を生んだ原因を（さほど）追求することなく，その結果に対する場当たり的な対応がソフトウェアその物の寿命を縮めている。「このようなソフトウェア」やその開発環境を改善しようというソフトウェア工学も「実務面での評価は欠かせない」などという理由から，時に悪いソフトウェアそのもののような研究活動をとるものも全くないわけではなかった。このような目立つことすれ得るところの少ない成果（＝弊害！）を棚に上げて「ソフトウェア工学などというものは単なる経験工学にしか過ぎない！」などという暴論を吐く輩（やから）まであらわれることとなってしまっている。

そのような反省から，多少堅苦しいかもしれないが，ぎちぎちに言語を規定するようなアプローチを取ってみたり，また一方で今まで言語から漏れていた機能を環境として取り込むことを規定したりする試みがなされている。そのどれもが決定打となり得ていないのはソフトウェアの奥の深さの間接的な証（あかし）であろう。現在の技術成果を応用して現実問題に近似解を出すことも工学という名を冠せられたものの責務と言える。それゆえ完璧を狙うための各種の試みは，その評価を早急に下すべきでは無かろうが，即効性を強く求められている分野への具体的成果功績が最近さほど聞かれないように思うのは，私の勉強不足だけなのであろうか？。

テストを十分に行うことの出来る環境を欲しいと思うのは誰しも同じだが，どの程度のテストを望んでいるのかは定かではない。結果として狼に変貌してくれなければ（自分の所以外ならどこででも）好きかってに飛び回っていてくれて構わないと思っているのなら余りにも安易である。エキスパートシステムを議論したグループから，ソフトウェア作成保守の知識を貯めていけば良いという安易な発表があったが，出来るくらいだったら誰でもやっていると思う。実際出来ないのは何故か，そしてそれを克服する手立てをもっと議

マシンベースで情報を適度に蓄えておけば，次に仕様記述言語が出てきた時に，役に立つ
ではないか。その程度の気持ちでメタな情報をきちんと整理しておく必要があるので，取
敢えずフリーフォーマットで書き貯めておこう！。
だからいろいろ考えても，将来必ずや努力が無駄にならないことは，将来のデータベースお
にに相応しいように何時でも変換可能なように，情報の意味も含めたデータの管理をしてお
くことである。但し，あまり目標も持たずにデータの整備だけをチマチマとしているのもな
なさせないので，データ整備の為の指針となるような，事務アプリケーション構築のため
のモデルを検討している。特にここで重要だと私が考えているのは，「時制」を巧みに表
現することが出来ないか？ということである。最善の答えは見出せてはいないが，暫定的
な対応は可能になりつつある。（詳細は来年のソフトウェア・シンポジウムで！）

7．おわりに

八ヶ岳の別荘というのは，たいそう涼しいところかと思っていったら意に反して暑いと
ころであった。天候不順で泣かされた今年にしては，まったく信じられない贅沢をさせて
もらえたわけだが。別荘村というのは（別荘の中は別として）討論の場所を気軽に選べな
いという点では車という移動手段を持たないグループにとっては，かなり苦しかったよう
だ。その点に関しては我々のグループは適度に場所を変えながら（清里も一応楽しみっ
つ）討論に集中することが出来た。また夜遅くまで議論し続けられるという点では，別荘
ライフというものも悪くはないとは思うが，ビジネスホテル程度の就寝設備も持たない場
所に面食らう者もないわけではないので，今後の検討課題と言えるだろう。

ワークショップが終了して，昼食を兼ねて主だったＰＣで蕎麦を食べに行った。その時
に色々話を聞いた限りでは，結構我々のグループは内容こそ地味であったものの，若手の
会らしく積極的に意見情報交換が出来て成功していたように感じたが，メンバ諸兄はいか
がであったろうか？。各人のレポートを楽しみに待ちながら筆を置くこととする。

〔謝辞〕夜遅く，私のためにわざわざ玉蜀黍（とうもろこし）を「取ってきて」くれた後
藤さん（シスプラン）伊藤さん（早稲田大），さらに調理に協力してくれた西川さん（Ｓ
ＲＡ）ほかメンバ各位に感謝します。

【参考文献】

［1］F.P.Brooks,Jr:No Silver Bullet:Essance and Accidents of Software Engineering,IEEE COMPUTER,Vol.20,No.4,pp.10-19(1987.4)
〔邦訳：ソフトウェア開発に王道なし，UNIX MAGAZINE,Vol.3,No.5,pp.87-99(1988.5)〕
［2］有澤誠：ソフトウェアプロトタイピング，近代科学社（1986.7）
［3］有澤誠：ソフトウェア工学，岩波書店（1988.5）
［1］の解説も含む．メンバの１人はこれをしっかり討論の場に持ち込んでいた．これを片手に議論を持ち掛けられて，私は冷汗ものであった．
［4］橋本正明：データ中心のプログラム仕様記述法，井上書院(1988.5)
［5］堀内一：データ中心のシステム設計，オーム社(1988.3)
データを中心としてシステム設計を取り上げた本で，内容的に共に補完関係にある．実務の上からは［5］が入りやすいが，全てを鵜呑みにするのは危険だし，そのことは，現場で叩き上げられた人には直ぐ分かるだろう！．
［6］渡邊，高手山：ソフトウェア部品の一考察，ソフトウェアシンポジウム’８８論文集

論してもらいたかった。私は，莫大な知識（情報）量の整理には，統計学を正しく応用しながら，計画的に知識の整理（知識表現の是正，不要知識の抹消など）をしていかなければ，１０の６乗を越えるオーダの情報には対処出来ないのではないか勝手に思い込んでいるのだが．．．。

5．何が銀の弾丸なのか？

時にはサイレンサも必要だし，弾丸は１発より数百発のほうが良いかもしれないし，散弾のような仕掛けが必要かもしれない。「狼を捕まえてみたら，以外と肉がおいしかった。弾だが弾をかなり使った倒したので食べられるところがほとんど無くなってしまった」ではつまらない。また，肉はとても食えたものではないかもしれないが，毛皮は結構上等かもしれないから安いミンクよりは，貴重な狼の毛皮のほうが（下手物好きの）彼女も喜んでくれるかもしれない。つまり，銀の弾丸を打ち込むべき対象である「狼男」の実態が良く分かっていないから，銀の弾丸が手に入らないのではないか？。また一方で，弾丸の製造およびその周辺技術（例えて言えば，弾丸があってもそれに耐えられる銃が無ければしょうがない！）が発展途上にあるのに，最終目標である銀の弾丸がすぐには得られるわけがない。

そのような状況で，銀の弾丸を議論すること自体が間違っているのである。例えばエキスパートシステムに関する研究成果が十分に出れば，自分で考える弾丸が生まれるかもしれない。言わば万能薬的な弾丸である。薬はいうものは，通常医師が患者を診察して幾つかの判断をして適当なものを処方して渡す。そしてその投薬効果を再び医師が評価してその薬の意味がある。（駄目ならほかの薬を処方する必要がある！）

しかしブルックスは，個々の技術を取り上げて云々している。しかし私が思うにまだソフトウェアに関する技術はそこまでとても到達していない。人類が過去に得た科学技術の知見を素直に応用出来た範疇の技術は，確かにコンピュータの世界でもすぐにその効果を表した。しかしそのような以外の要素を含む難問に計算機科学は立ち向かおうとしているる。そこで最終的に役に立つソフトウェア技術というのはもっと我々の考えを越えたもののような気がする。例えて言えば，古代エジプトで，測量の技術として幾何学が発展したことと，現在の幾何学の対象（例えば代数曲面論などというもの）を比較しようということが無理であるのと同じことである。そして現代の幾何学で使っている道具は，ホモロジー代数などの他の数学の分野の成果である。

私の言いたいことは『銀の弾丸は“今はまだ”ない』ということである。これはブルックスに言わせれば「だから『銀の弾丸はない』と言ったんだ」ということなのかもしれないが．

6．運用保守の問題を解決しないソフトウェアの改良は何も良い結果をもたらしはしないであろう

個人的には「運用・保守の諸問題を良い意味で開発時（もしくはそれ以前）にフィードバックしなければならない」と強く感じている。ただ少なくとも日本国内では，それらの問題が正しく整理されていないので，良くも悪くも開発以前（特に言語などの規格の制定段階）に反映されていかないのではと思っている。実際，ブルックスの話の中に運用でのプログラム仕様の問題のようなことには一切触れられていない。極端な言い方かもしれないが，現場の問題が忘れ去られて，しばしば言語やＤＢが理論的な背景を楯に使いやすくならないのは至って残念である。そこで愚痴ばっかり言っていてもしょうがないので，具体的な，私の『逃げの一手』を披露しよう。

私のしばらくの対応策は，然るべき言語処理系が（私の身近な）計算機にのってくれるまでじっと耐えることである。しかしただ漫然と待っているのも能がない。そのためには
①然るべき言語にふさわしいプログラムの構造（コーディングスタイル，モジュール分割，再利用部品（マクロ，パッケージ等々）の作成基準，オブジェクトの整理法等々）を出来るかぎり明らかにしておくこと
②然るべき言語でいつでも記述可能なように，現場での要求仕様を整理しておくこと
③然るべき言語にマッチしたデータベースに変換しうるようなデータを体系的に準備しておくこと
の３点が重要と考えている。これらのうち①は言語が固定してからも議論百出となろう。しかしそれではしょうがないから，今回体験したような擬似訓練をパソコンなりワークステーションなりを使って，講習会なり雑誌から情報を仕入れることによってトレーニングしていく以外にないであろう。実際どんな言語でも，わざわざ提案されてインプリメンテーションされるくらいだから，そうそう悪いものでもない〔実際使い心地という点では，良いのだが，実務で使うには信頼性や処理速度が問題だが〕。また②に関してもオペレーショナル・アプローチまで期待することとなれば，そのための記述言語を用意せねばならない。というと，再び言語に戻ってしまうのか？。そんなに肩肘張って身構えなくても，

ソフトウェア工学は役に立たないか？

佐原 伸
SRA

1. はじめに

「ソフトウェア工学は役に立たない」という意見をよく聞く．実際にはどうなのかを，個人的な経験をもとに整理してみた．結果は，「ソフトウェア工学は役に立つ」ということになった．

そこで八が岳のワークショップでは，「役に立つソフトウェア工学」という話題で話す．

2. ソフトウェア工学は役に立つ

「ソフトウェア工学が役に立つのか立たないのか」という命題は，個人的にはだいぶ前から結論が出ていた．すなわち，ソフトウェア工学のおかげで，筆者の個人的開発環境・生産性は飛躍的に向上した，というものである．

しかし，ソフトウェア工学は役に立たない，理論など役に立たない，という正反対の声を，メインフレーム上でCOBOLを使ってシステムを開発する人からよく聞くことが多かった．そこで，個人的に開発したプログラムの属性を時系列的に並べ，今と昔とで，何がどう変わっているかを調べてみた．

3. 個人的生産性

(1) 経歴

表－１をみていただきたい．これは筆者が作成したプログラムの主なものの履歴である．縦に作成年度，横には，OS・開発言語・開発時間・機能量・ステップ数・対象アプリケーション・開発方法論を列挙した．開発時間には，純粋にプログラムを作成した時間だけでなく，会議などの時間も含まれている．また，ステップ数には注釈文も含まれている．機能量は，対象アプリケーションがAddress，すなわち住所管理プログラムであるものについて，主要な機能の量を比率で表したものである．

これらのプログラムは，会社の仕事として作ったものも，個人的な趣味で作ったものもある．また，プログラムの走行環境となるコンピュータは，パソコンからミニコン，ワークステーション，メインフレームまで各種ある．古いデータの一部は，記憶に頼って記述したので，２割くらいの誤差が含まれている可能性がある．

(2)住所録プログラム

表－１の中で，開発アプリケーションの欄でAddressとあるのは，前述したように住所録管理プログラムであるが，４行目のものを除いて，筆者の個人的要求に基づいて作られており，毎年の年賀状の宛名書きに使われる予定だった(注１）ものである．その意味で，生産性を比較するのに都合がよい．

この住所録プログラムの生産性の比較したものが図－１である．領域グラフの部分は開発時間の比較を示し，線グラフの部分はそれを機能量で修正したものである．縦軸は開発時間，横軸は開発言語である．

この図をみると，生データの方で最大２００倍，修正データの方で最大１２００倍あまりの生産性の差がある．しかも，この中でもっとも開発時間の短いHyperTalkによるプログラムが，もっとも機能があり，使いやすく，速い．Fortranによる最初のプログラムが，カードを読み込んで一覧表を打つだけのバッチ処理プログラムだったのに対して，HyperTalkのそれは，すべて対話的に検索し，各種のキーでソートし，郵便番号から住所を自動付加をする．さらに，過去に作ったawkとinformixによるプログラムからデータを読み込み，選択した結果を一覧表と住所ラベルにプリントする．ついでに，電話の自動発信もする．

(3)COBOL パーザー

表－１でAppl欄にCoParseと書いてあるのは（１０行目と１２行目とでふたつ存在する），ほぼ同じ機能を持ったCOBOL解析プログラムである．このふたつを比較して注目して欲しいのは，３倍の生産性がある後者のプログラムの方が，OS (Unix) も開発言語 (Lex, Yacc, C) も，初めてのものであったことでる．また，前者のプログラムの方が，品質は圧倒的に良くなく，処理速度が遅い．

(4)電卓プログラム

表－１でAppl欄にCalcと書いてある電卓プログラムは，コンピュータの端末の前で電卓を使って，８進・１６進・１０進数字の変換を行なっていたプログラマへの贈りものである．変数を使用でき，桁数も普通の電卓より大きいものが扱える．このふたつのプログラムを比較してみると，電卓としての機能はほとんど同じなのに，３３倍の生産性の差がある．

注１：まだ一度も出したことがない．今年こそは…

3. 何が生産性向上をもたらしたのか

簡単にみてきたように，住所録，COBOLパーザー，電卓プログラムによる生産性の差が，数十倍から数百倍，場合によっては千倍を越えている．何故このような差が発生するのか，以下に見てみる．

(1)開発言語

最大の要因は，プログラム言語の進化だろう．生産性の悪いプログラム言語はFortran，Basic，COBOL等で，

- ・モジュール化
- ・情報隠蔽
- ・構造化プログラミング

が素直にできなかった．

その結果，既存のライブラリーがほとんどなく，自ら手作りしなければならなかった．例えば，COBOLパーザーのBasic版(注２）は，Unix上であれば標準関数として利用できる多数の関数を，相当な工数をかけて，自前で開発しなければならなかった．

一方で，生産性の高い言語は，モジュール化や情報隠蔽あるいは構造化プログラミング能力は当然備えているが，他にも住所管理やCOBOL解析や電卓に関する知識を，言語機能として持っていた．例えば，

- ・文字列の扱い
- ・文字列パターンの一致
- ・検索
- ・対話機能
- ・リレーショナル・モデル操作
- ・OSとのインタフェース

などに，生産性の悪い言語よりも遥かに優れた機能を持ち，しかもその機能の実行効率も優れていた．

(2)開発環境・ツール

次に重要だったのが開発環境とツールである．

優れた言語が動く開発環境・ツールの方が，よくない言語が動く開発環境・ツールよりも遥かに優れていた(注３）．例えば，

- ・エディター
- ・コマンド言語
- ・ツール群

などに優秀なものが多く，プログラムを段階的に少しずつ作り，それをすぐに確認することができた．これらのツールは，構造化設計，ツールキット方式によるツールの再利用，コンパイラー理論の使用などを支援してくれた．というよりは，これらのツールを通して優れた方法論を吸収し，生産性が飛躍的に向上したともいえる．

注２：それでもC上の普通のBasicよりはずいぶんといいBasicだった．

注３：実際はよくない言語に，開発環境・ツールはほとんどないといってよかった．

4. まとめにかえて

(1)ソフトウェア工学は役に立つ

上述したように「ソフトウェア工学の進歩」で生産性は向上しているのに，どうして「ソフトウェア工学は役に立たない」という声が出るのだろうか？　原因は簡単で，ソフトウェア工学の研究者は，ほとんどCOBOL，PL/IやBasic，そしてそれらが動く開発環境を研究していないからである．これらを研究しても生産性や信頼性を，２倍や３倍にあげることは何とかできるだろうが，１００倍，２００倍，あるいは１０００倍に引き上げることはとてもできない相談である．

一方でオブジェクト指向やハイパーテキスト，あるいはマルチメディアといった，劇的な改善を図ることができるかもしれない道が見えている時に，誰がCOBOLを研究するだろうか．つまり，「ソフトウェア工学は役に立たない」のではなく，「メインフレーム上でCOBOLで開発をしている限り，ソフトウェア工学は役に立たない」ということなのである．

(2)現実

しかし，現実にはメインフレーム上でCOBOLで開発しなければならない，という声が聞こえてきそうであるが，もう一度表－１を見て欲しい．この中のプログラムで，メインフレーム上でCOBOLで開発しなければ現実的ではないのは，７行目のプログラムだけである．約４０万件のデータを毎日処理する必要のあるこのプログラムでは，メインフレームでCOBOLでも仕方ないかもしれない．しかし，他のプログラムはそんな必要はない．これは特殊な例ではなく，今までに開発してきたプログラムないしシステムで，本当にメインフレームで処理することが必要だったものは１割もない．

(3)八が岳

八が岳のワークショップでは，ソフトウェア工学の役に

立ちそうな話題から，

- CASE ツールとプロトタイピング言語としての HyperTalk
- HyperTalk を中心としたオブジェクト指向プログラミング

を取り上げ話したい．

(4)蛇足

筆者の自宅と会社の机の上にあるワークステーションとパソコンの５台は，合計で 7M の CPU パワー，メモリー 16.5MB，ディスク 336MB を持っている．その上で動くソフトウェアは，

- つい２年前は伝説であった Smalltalk
- オブジェクト指向ライブラリー群を持つ Object Pascal
- ハイパーテキストとマルチメディアとオブジェクト指向をうまくまとめた HyperTalk
- 伝説のハッカー Richard Stallman の GNU Emacs・GNU G++・Bison

など，ひとつとしてメインフレーム上では手に入らないものばかりである．しかも以上３ソフトウェアの合計価格は，274 ドルで，買い取り一部ソース付き．GNU 関係はタダである．

夏のプログラミング・ワークショップ（若手の会in八ヶ岳）
ポジション・ペーパ

1988年11月9日(水)

藤本良子

1．はじめに.....

ワークショップには、以前参加したことはあったのですが、はじめてｐｃとして参加したその感想とワークショップを終えての感想をここに書いてみたいと思います。

2．それから.....

まず、反省すべきことは、今回参加した理由があいまいで、なにもプレゼンテーションするべきものの持たず、参加者と同じ感覚で、いえ、それ以下で、出席していたことです。
（みなさん、御迷惑をおかけしました。すみません）
まず、わたしは、エキスパートにおける技法を使ってというグループにはいったわけですが、わたし自身エキスパートの事は、全くわからず、最初から困難が待ち受けていたわけです。でも、ソフトウエア開発、それも設計におけるエキスパート・システムのアプローチということで、いままでの経験をはなすことで、少しは、役にたつことができるのではないかという甘い期待で、二日めにのぞみました。グループの中の人たちは、やっていることはいろいろ違ったことを行なっていましたがだいたいみんな年齢も同じくらいで、かかえている問題は、どこも同じで少し安心しました。その様々な問題をあまりよく理解していないエキスパートの手法で解くということは、大変、難しい事ではあったのですが、なかに、その手法を用いて仕事をしている人がいて、最終的に、少しでもエキスパートの技法に触れた事は、大変、勉強になったと思います。自分の勉強にはなったものの、参加者の人たちのお役にあまりたてなかったことをふかく反省しています。

3．ところで.....

今回、マシンを持ち込んでグループ討論を行なっているところがあったわけですが、二日めの夜、興味があったので、 4GL 、LINC のグループを覗いて見ました。
ちょうど、そのころアルバイトで、COBOL をやっていまして、どうにかして、あの同じようなプログラムを簡単に処理できないものかと思っていましたので、簡単にプロトタイプできてそれが商品化できるのには、大変興味を持ちました。
しかし、はたしてどれくらい現状のニーズに答えられるかは少々疑問が残っています。今後、どう発展していくか興味あるひとつになっています。

4．おわりに.....

各グループとも、違うアプローチで、［銀の弾丸］を求めたわけですが、この期間だけでは、最終結論まで導くのは、大変むずかしく、途中までのところが多かったと感じています。しかし、これを足掛りに、今後、自社にあったアプローチ法で、［銀の弾丸］を求めていくことは大切であると思います。
わたしも、今は、働いていませんので、役立てる事は難しいのですが、いろんな会社の人々と交流でき、ふだんの生活では得られないような、刺激を得られたことは、事実で、視野も広げる事ができたと思います。

八ケ岳ワークショップにて　　ＫＣＳ　山本　一成

『銀の弾丸』があるか？

ここで『無い』といってしまうことは、やはり淋しい。せめて『銀の玉の基』ぐらいは欲しいもんです。４日間いろいろな人の話を聞いていて、『銀の粉』と呼べるものは、あると実感しました。今回のグループ討議のテーマとなっているものは、全てそれに値するものと思います。

ソフトウェアは幾つもの工程を経て開発が行われています。そこで、私は、ソフトウェアの開発における問題には、

1. 各工程における問題
2. 工程間における問題

があるように思います。

現在ソウトウェアを開発するにあたっての様々な問題は、『要求分析をするときはこんな問題がある』、『設計書を作るときにはあんな問題がある』など、各工程ごとに捉えられていることが多いように思います。そして、そのような問題に対する解決策として幾つもの手法が考えられ、それらは、１００％の解決策とは言えないまでも、従来の方法に比べてとっても良いものです。おかげで、各工程における作業効率は随分良くなってきています。それゆえに、各工程における問題は解消されてきています。

次に、工程間における問題はどう解決すればよいのか。まず、浮かんでくることは工程の数をできるだけ減らすということです。工程の数が減ることで、当然問題の発生する数は減るからです。このことは、第一の問題と同様に現在では、自動プログラミング等に見られる用に、ドキュメント＝ソースということで、コーディング工程を無くすといった具合に、徐々に実現されてきています。しかし、幾ら工程を減らすことができるとしても、ソフトウェアの開発を一つの工程で実現できるとは思えません。一つの工程で開発できない限り、この工程間における問題は残ってしまいます。各工程ごとの『銀の粉』があっても、それらを『玉』にするなにか、『小麦粉』を『団子』にするには『水』が必要であるように、『銀の粉』を『銀の玉』にする何か『ソフトウェア開発における水』が必要だと思います。

では、どこに『ソフトウェア開発における水』があるのか？

実は、私にはさっぱり分かりません。現在すでに、このような良いものが世の中に存在しているのかどうか。コレゾというまでのものは無いにしろ、その基となるものくらいはあるのかどうか、ＵＮＩＸ上で実現しているネットワーク環境を利用することで『水』となりえるものができるのか、本当に分かりません。

私は、ここ３年あまりＡＩという分野であれこれと仕事をやってきましたが、そのひいき目で見て、このＡＩと呼ばれているもので使っている開発手法は、『ソフトウェア開発における水の基』ぐらいになるものを構築できるもののように思います。それは、主に次の理由からです。

① エキスパートシステムに代表するように、成長するシステムを構築できるということです。現在では、工程間における問題がどういうものであるのかが、明確に理解できていないこと、そして、その問題も常に変化しています。このようなことに対して適しているのではないかということです。現段階で明確な部分をシステム化することにより、基礎となる部分の構築を行ない、その後、発生した問題に対応していくというアプローチでシステム化が実現できる点です。

② システム構築の大部分が知識ベースを構築することであり、この知識ベースというものが、共有できるということです。一度構築した知識ベースは、たとえ利用形態が変化したとしても、集めた知識は、再利用できるからです。

③ これまでシステム化の難しかった分野に対して、ＡＩ的手法を用いることで、システム化を実現している例が幾つもあるということです。それは、過大評価して、『とてつもない凄いことができる』とかいうことではなく、これまでに実現できなかったことを、少しでも可能性のある手法を用いて試みるということです。ＡＩ的手法はその可能性のある手法の一つであると思うからです。

このような理由から、ＡＩ的手法が『ソフトウェア開発における水』に成りえるかも知れないと思ったわけです。『水』が見つかったならば、『粉』を『玉』にすることができそして、いずれ『銀の弾丸』することができるでだろうと思います。

これは、私の推測というよりは、ほとんど希望であり、できればこうなってくれたらなあ、というものです。

しかし、実際に今このことが実現できるとは思っていません。

① 如何にして知識を獲得するのか

② どのように知識を整理すれば良いのか

③ 構築するためにはどのような環境を用意すればよいのか

④ 既存システムとの融合

など、やはり多くの問題を抱えています。私にとっては、この問題の解決が先です。そして、それが『銀の弾丸』のための一歩でもあると思います。

第６回　夏のプログラムワークショップに参加して

（株）呉電子計算センター
板倉重利

今回のテーマは、ソフトウェア・エンジニアリング技法のケーススタディ（ソフトウェア開発の「銀の弾丸」を求めて）、具体的には、形式的仕様記述を用いたアプローチ、オブジェクト指向のアプローチなどとなっていた。

新しいシステムエンジニアリング技術の、実用度に対する期待と、ワークショップ初参加という不安と戸惑いが、交錯した中で過ごした八ヶ岳の４日間でした。

１．グループ活動について

これまで使用したことのないシステム（含ハード）と４ＧＬ（ＬＩＮＣⅡ）、および進化型プロトタイピングによるアプリケーション開発に対する興味から第２グループに参加した。

ＬＩＮＣⅡの機能説明から始まり、在庫管理モデルシステム実演を終えてのグループの総体的な反応は、これまでのソフトウェア開発手法に対する変革の喚起を予感したようだった。

ＤＢの自動生成機能、プロファイルの考えなど、これまでの言語仕様、データ管理と異なる部分を随所にみることができた。

グループ討論は、実機を使用してのモデルシステム作成であったこと、メンバがシステム開発に携わっており、それぞれの職場環境における現実問題を提示したことで、より具体的な結論を求めようとする姿勢から活発な討論が展開され、最終日の成果発表へとつながった。

前述のように明確なかたちで、成長型プロトタイピングを実感できたのは、ワークショップに実機を持ち込み、献身的な努力をしていただいた、ＰＣの佐藤さん、日本ユニシス（株）高山さんの協力、及びメンバ方の頑張りのおかげです。本当にお世話になりました。

２．ワークショップに参加しての感想

グループ発表後、プログラム委員長より次のような挨拶があった。“ワークショップで大事なことはネットワークを張ることです”。

ひとつは、人とのネットワーク。

ひとつは、知識、情報のネットワーク。

ワークショップは、人脈の拡大、知識、情報を吸収するためのアンテナを持つことで、自分自身の能力向上に役立てることです。

今回のワークショップ参加で、人脈拡大、知識、情報、情報吸収がどこまでできたか、効果の程はすぐに出るものではないが、多種多様な環境で働くＳＥを知ることができたこと、またソフト面では、新しい技術が近いうちに、我々の職場にも浸透してくるようなうねりを感じた。

多勢の中で、話題を持って会話を進めるには、コンピュータ技術に関するものだけでなく“ニーベルンクの指輪”と“ＯＢＪ２”との関係など、他の分野での知識、情報の必要性を改めて感じるなど、今回のワークショップ参加は有意義なものであった。

３．今後の課題

今回のテーマであるソフトウェア生産性向上のためのアプローチは、現実に直面している問題であり、今後のソフトウェア開発に対して積極的に活用を図りたい。

しかし、システム開発、ソフトウェア開発の生産性向上のみに目を奪われるのでなく、開発されたソフトウェアの運用面における生産性向上（処理時間、人件費など）をも考慮したものでなければ、真のシステムの生産性向上とはいえないだろう。

大規模オンラインの障害処理が、すべてホスト側に依存されているため、ハード、ソフト、人、設備負担の増大や大量のバッチデータを処理するための、処理時間の延長など、運用面における問題を多く抱えている。

これらを解決するには、従来の発想を変えた新技術（ハード、ソフト）に対してのアプローチにも傾注すべきだと考える。

－以上－

分かりやすいソフトウェア

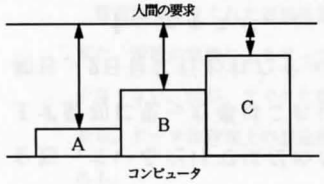
日本電気ソフトウェア株式会社
中沢　賢光

1　はじめに

ソフトウェアは、人間の要求をコンピュータに分かる形に表現したものである。人間の考え方とコンピュータに理解できる表現には大きな開きがある。この開きを縮めることがソフトウェア開発における生産性の向上に役立つ。人間の考えていることがそのままソフトウェアになれば、誰でもソフトウェアを作れるようになりＳＥは要らなくなる。

2　表現の変遷

下図は人間の要求とコンピュータに理解できる表現との開きを表わしている。初めはコンピュータの動かし方を直接書いていたのが、段々人間の表現に近づいてきて、今は人間の考え方を表現することが問題になってきているのが分かる。



A	マシン語 アセンブラ	コンピュータの動かし方を表わす
B	手続き型言語 関数型言語 オブジェクト指向言語 第四世代言語	人間の言語に近い表現でコンピュータの動かし方を表わす
C	エキスパート・システム 代数型言語	要求や知識を表わす

マシン語やアセンブラでは命令を一つずつ記述する。これらはコンピュータの動作を直接記述するから人間の考え方とは大きな開きがある。

コンピュータは一時に一つのことしかできないので、その命令形式は手続き的である。これを人間の言語に近い形にしたものが手続き型言語である。手続き型の表現形式は人間の考え方とは異なる。人間は手続き的には考えない。

関数型の言語は入力と出力を中心にコンピュータの動作を記述する。人間の考え方には入力と出力は出てこないから、関数型の言語も人間の考え方とは開きがある。また、関数型の言語の内部は手続き的なので、手続き型の言語の特徴を引きずることになる。

経験的にデータを中心に設計したほうがよいことが分かってきた。データに対し低レベルのアクセス関数を用意しこれを通してのみデータのアクセスをするというデータ抽象の考え方や、もっと進んでデータとそれに関する関数をまとめてオブジェクトとしオブジェクトによりプログラムを作るオブジェクト指向が現われた。同時に、知識工学の分野では人間の考え方のモデルとしてフレーム･モデルが提唱された。現在オブジェクト指向とフレーム･モデルはほとんど区別されていない。オブジェクト指向は人間の考え方に近いといえる。

第四世代言語はこれまでの一つの言語により記述するものと違い、データ･ベースの記述や画面の記述のために複数の表現法からなる。また、最近の開発支援ツールにはフローや絵を描けるものが多い。人間は言語だけで考えるのではなくイメージや音などいろいろな表現で考える。複数の表現でプログラムを記述できるのは、人間の考え方にあっている。

最近は要求記述を目的とした言語やツールがいろいろでてきたが、問題を持つものも多い。たとえば、要求を記述するための言語は自然言語が一番良いが、自然言語を処理するのは非常に難しいので形式的な言語で記述するものがある。しかし、あまりに論理的な言語であるため人間の考え方とかけ離れていて普通の人間には扱えない。また、プログラムを記述するものとは少し違うが、人間の知識を表わすことを目的としてエキスパート･システムがある。人間の知識を表わそうという試みは考慮すべきだが、適用できる範囲が狭いという問題がある。

以上の流れを見ると、表現法の性質がコンピュータをどのように動かすか(動作中心)から人間の要求をどのように表わすか(要求中心)に移ってきているのが分かる。

3　作業の変遷

動作中心と要求中心とでは開発形態も異なってくる。

現在の一般的な開発形態はウォータホール型である。ウォータホール型の作業はプログラムの作成を中心とし、人間の要求をコンピュータに分かる形に翻訳すること(設計、コーディング)が一番重要な仕事となる。この過程でドキュメントを大量に生産するが、このドキュメントはほとんど無駄になる。要求者は基本的な仕様が分かればよい、詳しいことは必要無い。保守者には役に立ちそうで役に立たない。実際のプログラムとこのドキュメントの間には食い違いが生じやすい、分かりにくい(ソフトウェア技術者はドキュメント記述が苦手である)、量が多すぎる(何千ページもあるドキュメントなんて見る気がしない)等の問題がある。ドキュメント作成は、この開発形態における大きな問題である。

要求中心の表現法に対する開発形態では、要求を記述することが中心になり、要求をコンピュータに分かる形に翻訳する作業(設計、コーディング)はほとんど自動化される。無駄になるドキュメントは少なくなりドキュメント作成の自動化も容易になる。ドキュメントが減り大幅に作業の効率化が望める。

プロトタイピングは昔から行われてきたが、今盛んになってきたのはなぜだろうか。昔はきちっと設計しないとプログラムは作れなかったが、今はプログラムの作成が容易になってきたのですぐ動くものが作れる。また、動作中心のときは動作が正確に行われることの確認すなわちテストが重きをなしていたが、要求中心になると要求が実現されることの確認を行わねばならなくなり、そのためにはプロトタイピングしてみるのが一番良い。要求中心になってきたので、プロトタイピングが盛んになってきたのではないだろうか。

4　人間の問題

新しい技術がいろいろと開発されてきたが、ソフトウェアの開発技術を向上させる上での最大の問題点は人間である。古い技術(考え方)にしがみついている人は、新しい技術(考え方)を取り入れることができない。意識の変革が必要である。動作中心の考え方から要求中心の考え方に変われば、自然に新しい開発技術を身に付けるようになる。逆に新しい技術を取り入れる人は要求中心の考え方が自然と身に付く。

第6回

夏のプログラミングワークショップ　in　八ヶ岳　'88

期　間　；　S63.8.31(水)　－　S63.9.3(土)

(株) 日本情報開発

山田　良一

今回のワークショップに参加して実感した事は、雑誌等で言葉を知っているだけではだめだと言う事でした。
まず最初に思った事はこれにつきました。
初日、2日目に行われたチュートリアルにおける

『　エキスパート　システム　』

『　オブジェクト　指向　』

について各々、寺野孝雄氏(電力中央研究所)、柴山悦哉氏(東京工業大学)両氏の講義を拝聴致しましたが、部分々々でとう理解、解釈して良いのか判断に苦しむ様な言葉が出ると表面上だけの知識だけではこれからのソフトウエア開発業務は成立しないのかなとふと考えさせられました。更にプログラム委員によるテーマ毎のパネルディスカッションにおいても追い打ちをかけられ、またまた考えさせられましたがしかし、現時点では自分なりの知識で十分に業務をこなしているという自負もあり複雑な気持ちでした。

但し、現時点での自身の業務にはあまり関係ないけれど新しい知識を吸収するんだという気持ちに切り替えるとわからない言葉に対しても反応が良くなった様な気がしました。話の内容(チュートリアルの内容)にしましても、今日においても部分々々ではありますが講義の内容が鮮明に思い出されます。それだけ、インパクトが強かったものと思われます。

今後、続けて参加しようとは正直思いませんが、でも若い人達が数多く参加する上では意義が有る事と思います。
今後の運営に期待致します。

最後に私事ではありますが私感を述べさせて頂きます。ワークショップ2日目，3日目のメインでありますグループ別討論において討論参加に対する強制に対し何を期待しているのか、個人的には自分でテーマを選んだ訳であり各人目的をもっての参加ではあると思いますが、グループ全体・プログラム委員が何を期待しているのか今回のワークショップの姿からはあまり感じられませんでした。

この様な感想を持ったのは参加者中私一人かもしれませんが、それでも一人いたという事実は認めて頂きたいと思います。

以上今後の運営に期待して終わりとさせて頂きます。

人を中心に

早稲田大学　事務システムセンター

井　上　文　人

一昔前のデータ処理の適用業務は、ほとんどが数字をザクザク踏み砕く、記録保管の適用業務だった。最近の適用業務はデータ処理というよりも、もっと情報処理を志向したものになってきた。そうした適用業務は依然として記録保管の適用業務であったが、かなりの説明が必要なデータではなく、管理用の情報を提供するものだった。

管理面でのすぐれた意思決定を支援するには、当初は利用できるデータが少な過ぎた。情報の急増にともなって、利用できるデータは豊富になり、多すぎることも少なくない。だが、すぐれた意思決定の支援に向いた形にはなっていないデータである。データは管理上の意思決定に役立つよう、圧縮して正確な情報にする必要がある。

データ処理に携わる人々は、データの収集・貯蔵・表示について比較的うまくこなしてきている。しかし、意思決定のためのデータを作り出す面では、それほどうまくいっていない。これは、必要性を上回るデータを管理者が得ていること、受け取ったデータについて彼らが理解していないこと、本当に必要なデータを得ていないことなどによる。

管理者にとっては、手元の意思決定にすぐ応用できるような、個人的な、言葉で表した形式ばらないデータがあると助かるというのが典型的なケースである。コンピータの作る報告書は、大型の、抽象的な内容の、表形式のものが普通で、これは問題の解決や保留しているものに役立ったとしても、ほんのわずかしか役立たないものである。こういう行き違いがあっては、コンピュータの報告書を使う気にはならなくなる。

管理に役立つ情報システムは、問題と直結した形で、意思決定支援を志向するものである。情報システムは利用者が理解でき、手元の意思決定に結びつく情報を作り出す必要がある。情報システムは、機械中心ではなく、人間中心に考える必要がある。システム設計者は、えてして機械とお友達になり、人間の我儘から逃げようとしがちである。

おわりに――――

今回のワークショップにおいては、専門的な言葉を理解するのがやっとで、オブジェクト指向や４ＧＬ等を使って、システム設計云々までは、到達できなかったがどのような方法にせよ機械の入口だけは、人間味溢れる人間にしてしまうまで、さまざまな問題が続くでしょう。

以　上

銀の弾丸とは

早稲田大学　事務システムセンター
伊藤達哉

1．銀の弾丸とは

今回のワークショップのテーマであった「銀の弾丸」とは、どのようなものであろうか。

現在のソフトウェアは、４ＧＬというような最新の言語が紹介されている反面、実際には、いぜんとしてＣＯＢＯＬ、ＦＯＲＴＲＡＮといった言語が作業現場で用いられている。いわば、新旧入り乱れての、混乱の様相を呈しているように思えるのである。

しかし、コンピュータの出現はそれ自体まぎれもない「銀の弾丸」であったはずである。こういった視点に立てば、ＣＯＢＯＬでさえ、出現当初は「銀の弾丸」であったに違いない。それならば、何故今「銀の弾丸」なのか。

コンピュータはこれまで、一部の専門家か、それ相当の知識をもった者にしか扱えなかった。その対象が、現在では徐々に広がり、あらゆる人々がコンピュータとのコミュニケーションを求めている。

したがって、今求められている「銀の弾丸」とは、こういったエンドユーザが扱えるソフトウェアのことをある一面では指していると思うのである。今の状況は、そこへ向かっての過渡期と考えられるのではないだろうか。

エキスパートシステムやオブジェクト指向といったものは、過渡期において淘汰されてしまうものかもしれないが、「エンドユーザが扱えるソフトウェア」への着実な１ステップであると思う。

2．ＣＡＳＥツール（ＳＴＰ）について

従来のシステム開発の問題点として、曖昧な要求をシステムエンジニアの経験や能力に頼ってシステム化しているという一面があった。しかも、構築しようとするシステムが、エンドユーザの眼にふれるまでに、かなりの期間を必要としている。この結果、出来上がったシステムはエンドユーザにとって使いにくく、したがって期待はずれのシステムとなる。この点をみても、エンドユーザの要求をシステムに反映させるのは、相当に困難な作業であることが理解できる。

今回のグループ討論では、上流工程において要求定義を作成するＳＴＰというＣＡＳＥツールを用いて、図書館システムの要求仕様を作成してみた。

従来の文字中心のドキュメントと異なり、視覚にうったえるような技法（データフローチャート等）を用い、しかも作成の際に編集やチェックなどがかかるツールなので、要求をまとめるまでの期間はそれ相応に短縮されると思う。また、修正等は比較的容易にできそうなので、この時点でプロトタイプを行い、下流工程に引き継ぐことにより、かなりの効果が得られるものと期待できる。

ＳＴＰでは、上流工程までしかサポートしていないが、今後、下流工程において４ＧＬ等と一体となった形のツールが提供されれば、全工程におけるプロトタイプが可能となり、「銀の弾丸」に一歩近づくように思える。

3．最後に

まるで異次元の世界にでもいたかのようなワークショップの４日間を終え、職場に戻ると、そこには厳然と、現実というものが存在し、ワークショップで聞いたり、討論したエキスパートシステムやオブジェクト指向などというものは、その影さえも見えないように思える。しかし、このような環境にいるからこそ「銀の弾丸」を求めることが大切なのであろう。そういった意味において、今回のワークショップは、自分にとって非常に意味深いものであり、また視野を広くもてるようになったと思う。

以上

88年10月13日

SEA88夏のワークショップAFTERPOSITION

問題設定の視角

三井銀ソフトウエアサービス㈱
立木　勲

このたびのワークショップに参加した結果を踏まえて

今回の参加を大まかに再整理しました。

1. 目的

ソフトウェア開発の先進的な技術研究の動向に接し
そのような技術研究を促している「背景＝時代」と
それにより解決をはかろうとしている「問題状況」をとらえる。

㋑「問題状況」の一般化を行い、日常業務の中にそれらを
見て取れるようにすることで日常業務の質的向上をはかる

㋺問題解決の方向に社会が進むものとし
10年20年先のシステム開発の姿がとらえられるようになり、
意識的な追求が出来るようになる

2. 今回の位置づけ

フレームワークができるようにする
自分の位置を測定する

3. 結論

未整理

4　技術.課題と問題状況.あるべき姿と現状

(テーマとしてとりあげられた技術を通じて)

(1)　エキスパートシステム

①フレーム
『知識』を個々人から抽出し、断片的、部分的なそれを整理し統合し
公共のものとし、誰もが再利用可能なかたちにする技術

②『エキスパートシステム』の現水準
省略

③ゆくさき
その思想と技術の向上と普及は、一人一人が自らのノウハウを普遍化し再利用
可能なように整理統合し公開することを可能とし、人々の問題解決能力が
飛躍的に向上している社会を想定することができる。

④あしもとから
問題解決におわれる日常業務では、解決の手段としての『知識』は問題が解決されるや
埋もれる傾向にある。一方で多額の資本を必要とする『エキスパートシステム』
の追求をみるならば『知識』の一般化は潜在的には時代のニーズである。
いや、まさに個々の職場、経営にとってこそ死活的ともいえる課題である。
であればそこにみをおく私は、個々の現実をどうとらえ対処していくのか、
これからの課題とする。

(2)　プロトタイピング

①感想
職場において今回のワークショップの自分なりの成果を最も感じているのが
4GLをもちいたこのパートである。
ここでふれたいくつかのものは、今未来にむけて業務を考えるときのポイントの
いくつかをかたちづくっている。これも以降の課題としておきたい。

以上

八ヶ岳Workshop'88

ソフトウェア開発における生産性の向上及び品質の改善について

㈱ランドコンピュータ

TETSUYA KANISHI

1988.09.30

Workshopに参加する前は，ソフトウェア開発工程の生産性の向上や，品質の改善については，現在の実務から考えて，ソフトウェアの欠陥を防ぐには要求分析からシステム設計における上流工程で幾度となくレビュー を行う事により可能となり，その注意点として 以下の事が考えられた．

1．レビュー 参加者は資料を事前に目を通して欠陥だと思うところを一覧表にしておく．

2．レビュー 中に全ての問題点を解決しようなどと思わないで，PENDING事項として次回までに 解決する．

3．チェックリストを用意しておく．

Workshopに参加した後は，ソフトウェア開発工程の生産性の向上や，品質の改善については，上記したものも大変重要なものであるが，あまりに人的負荷が大である．その人的負荷を減少させるためにCASE ツールを用いたり，4GL を 用いてソフトウェアプロトタイピングを 行ったり，エキスパートシステムを 使用したり，オブジェクト 指向により プログラミング を 行ったり 等の様々な技法を駆使すべきだと感じた次第である．

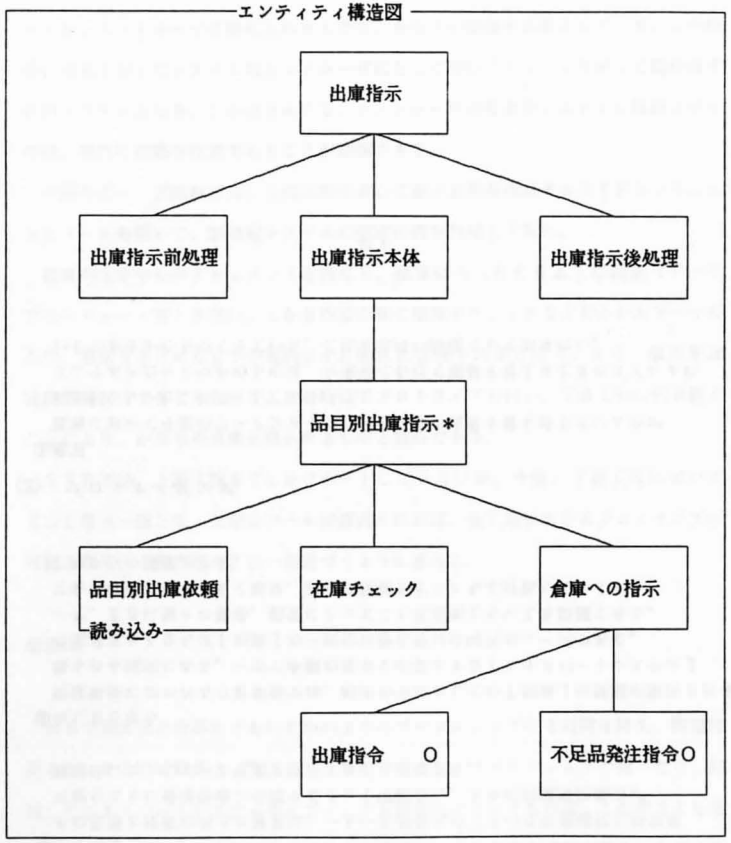
その中で特に今回のWorkshopに参加して学習したCASEツール を 用いた技法について述べてみたい（次頁に情報処理学会誌に『新しいプログラミング．パラダイムによる共通問題の設計』で取り上げられた酒倉庫問題を例にとり図形表現してみた）

今回のWorkshopではSTP（Software Through Picture）という ソフトウェア開発工程の上流工程において要求定義を作成するCASE ツールを学習したのが，このSTP とは今まで既に開発などで使用されている DFD（Data Flow Diagram），ER（Entity Rerationship），Structure Chart などを 使用して，Visual に表現して，要求仕様書を作成していくものである．

こういったCASE ツールを用いる事により エンドユーザが 簡単に要求仕様書を作成でき，図形表現しているため理解しやすく，修正も容易である．こうして作成された要求仕様書や 機能仕様書を元に，4GLなどを 用いて ソフトウェアプロトタイピング を 行うことにより，プログラム の バグは もちろん仕様による ミスがすぐに判明し，すぐに対処できると考えられる．

その他に考えれる事はCASEツール を 使用する際に DFD や ERなど知らなくてもエキスパートシステムや人工知能が自然に解釈して作成されるようになるべきである．

結論として ソフトウェア開発工程の生産性の向上や，品質の改善については，今まで述べてきた様々な技法を組み合わせることにより，早期実現が可能になると思われる．



以上のような図を，より細分化する事により，トップダウンのシステム開発が行われ，要求仕様が確立される．

酒倉庫問題

DFD とは4 種類の基本素子があり，それらの組み合わせて図面を構成する

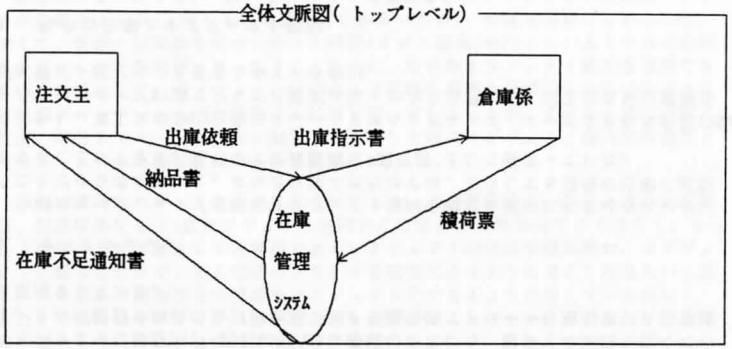
矢印 → は データ の流れを表す．データフロー と呼ぶ．

丸印○ は データ の 処理を表す．プロセスと呼ぶ．

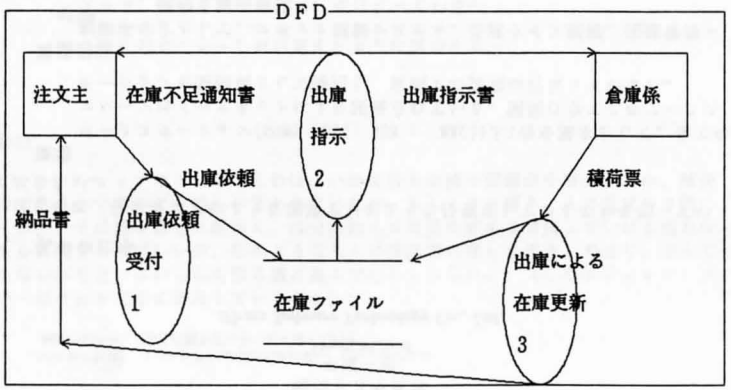
二重線═は 一時的にデータ を格納しておく場所．ファイルと呼ぶ

正方形□は システム外部の組織や人を表す．エンティティと呼ぶ

全体文脈図(トップレベル)



DFD



ソフトウェア・エンジニアリング技法のケース・スタディ

- ソフトウェア開発の「銀の弾丸」を求めて -

昭和６３年９月
平尾一浩
hirao@hst.junet
Hirata Software Technology Co., Ltd.

1. 私のお仕事

はじめに、私自身がどのような環境でどのような仕事をしているのかを知っていただきたい。

環境

ワークステーション(SONY NEWS; SUN ; MAC)1人1台を基本として、全てのマシーンはイーサネットにより接続されている。開発は全てこのワークステーションを開発機として使用し、実機での開発は行なっていない。

業務内容

制御系を主として、ロボット制御システム、生産ライン制御、生産管理システム、開発支援環境開発、を行なっている。

というような理想的(一般的に見て)な環境にいながら、数多くの問題を抱えている。その問題群の解決の糸口がオブジェクト指向的アプローチに潜んでいるのではと期待をしている。

2. ワークショップ

今回のワークショップで私はオブジェクト指向の概念を頭にたたき込みたいということだけを考えていた。本などを読んだだけでは、どうしても自分の仕事に直結させることができず、私の中では表面的な"流行語"として留まっていた。

しかし、東工大の柴山先生のチュートリアルとグループ・リーダをされた佐原(SRA)さんとのグループ討論で少しだけ鮮明になったように感じる。以下は自分の業務を例に挙げ、感じたことをまとめていきたい。

3. 私のお仕事とオブジェクト指向

ここで例として、生産管理システムを挙げる。*

私がこれまで経験した生産管理システムはそのオーダーによりさまざまであったが、基本的な機能として(共通機能として)以下のものを挙げることができる。

- □ 生産条件設定機能
- □ 生産実績管理機能
- □ ラインモニター機能
- □ エラー・リカバリー機能

加えて、これら4つの基本機能にも共通点がある。画面部分(グラフィック部分)と外部通信部分(対シーケンサ)である。

ここで通信を主に今の HST で構築している基本機能構成を挙げてみる。

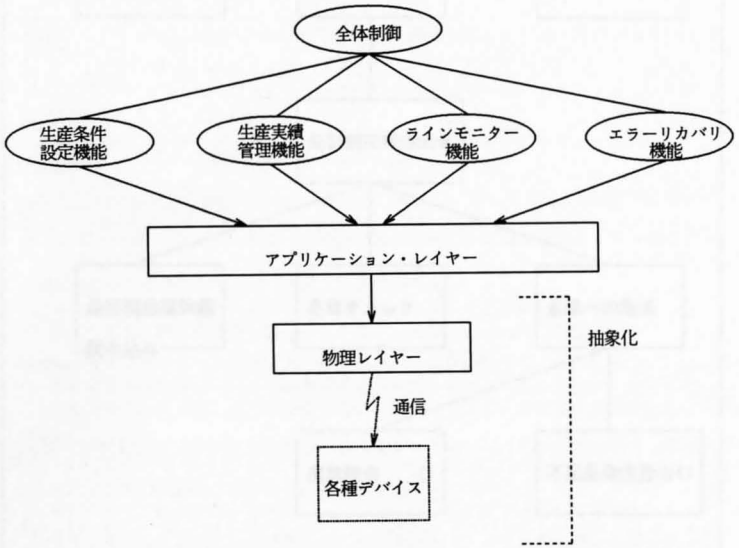


図1－ 生産管理基本構成

図１を見るとわかるように通信部分を2階層に分けることで、デバイスが変わった場合、例えば通信相手のシーケンサの機種や構成が変わった場合でも社内では通常使用している通信アプリケーション部分を使用することで、ハードを意識しない開発を行なうことができる。後はこれと並行して物理レイヤー部分を書き直すだけでよい。

この考えは通信だけでなく、画面を作成するグラフィック・ツール部分やプリンター出力部分などにも利用できる。(既に実現させている)

このようにソフト開発のなかでデバイスや特殊部分を全て抽象化することでソフト開発自体への影響を最小にすること(末端部分を抽象化する)で、その特殊部分のハード的変更や、その部分から独立した開発、テストが効率的に行なえるといった

＊ 実際のグループ討論ではロボット制御プログラム(システム)を例に挙げ、プレゼンテーションを行なったのだが、その話は私と同じグループの方は聞かれているのであえて違う例を挙げる。

ことが、結局はオブジェクト指向的なアプローチだとわかった。（一番の収穫）

グループ討論では、各メンバーのおかれている環境とそれぞれが抱える問題の差があったため、実際は踏み込んだオブジェクト指向の話しは出来なかった。しかし、私が行ったグループ内でのわずかなプレゼンテーションに対する佐原さんとメンバーの方々からのコメントで随分頭の中がクリアーになったように思う。

4. とにかく、まとめ

とにかくまとめると、 HST が行っている業務分野はオブジェクト指向的アプローチが非常にしやすく(同じものの繰り返しが多いため)、有効な分野だとわかった。加えて、実機と開発機を完全に分けた開発(クロス開発)を行っているとそれが自然とオブジェクト指向型になってくる。始めに、自分がオブジェクト指向を理解できないのを棚に上げ、オブジェクト指向なんて現場の泥臭さが感じられないなどと生意気をグループ・リーダの佐原さんに対し言ってしまったが、まったく逆で非常に泥臭い部分をスマートに、楽に開発しようとした時にオブジェクト指向型が自然と入ったシステムに構築されるものと実感した。

今後の課題はロボットの進化に伴うソフトウェアの拡張性と、同じような生産管理、生産制御をもう1度オブジェクト指向的に見直し、特殊部分だけではなく、もっと上層部までを対象として再利用できるオブジェクトの作成を積み重ね、オブジェクトの組み合わせで、ある程度のシステムを構築できるように考えて行きたいと思う。－共通基本機能部分の構成をオブジェクト化できるように考えていきたい－また、これにはオブジェクト指向型言語を使用することでデータの抽象化、情報隠蔽の強化も考えていかなければならないと考える。

結論

－銀の弾丸がすこぉ～しだけ見えたような気がする－
^^^^^^^^ ^^^^^^^^

独白

ワークショップなどというものは、いかに自分の持つ問題点を解決するか、解決の方向をさぐるかだと誰かが言われていたが、正しくそう思う。人の話ばかり聞いていて、その話を自分に置換え、自分の抱える問題の解決まで持っていける能力のある人間であればいいが、私のような凡人には非常に難しすぎる。やはり、とんでもない事を言いながら恥を掻き掻き進んで行くしかない。 （このポジション・ペーパーはそれを完全に実行しているなぁ）

今回もグループ討論では随分恥をかいたと喜んでいる。
ふっふっふっ、"ワークショップの恥は掻き捨て" というやつさ!?

つれづれなるままに

日本タイムシェア（株）　　中山　聡

①　「ソフトウエア開発のための技法」をどうする

英語と日本語の構文の違いとしてまず挙げられることは、英語が「どうする、何々を」と動作をまず始めにもってくるのに対して、日本語は「何々をどうする」と事物を先に持ってくることである。この言語の構文の違いを各々の民族の思考方法の違いに起因するものとしてとらえたとき、人間の思考によって作り出されるソフトウエアやその開発技法にも違いがあってもいいように思ってしまう。

ソフトウエア開発のための様々な考え方が米国から流れてくる。ここでもまた日本は米国の後塵を拝することになる。　オブジェクト指向なる考え方も米国から輸入されてきたわけだが、オブジェクト指向こそは日本から世界に向けて発せられるべき考え方ではなかったのか？　データの隠蔽、クラス、メッセージ、メソッドなどオブジェクト指向を語るときの単語はいろいろあるが、ここでは大まかに「ものを中心に置く考え方」と定義することにすると、「何々をどうする」と事物を始めに持ってくる日本語を日常使っている日本人がまず明文化してもよかったのではないだろうか。

また、XEROX社のSTARとかAPPLE社のMacintoshのマンマシン・インターフェースのひとつに「ディスプレイ上のオブジェクト（図形、テキストなど）をマウスで選択してからそれに対して動作を指示する」というのがあるが、これなども日本語的な（？）操作方法だ。

いろいろな開発技法が米国から流れ込んでくる。よくよく考えてみると、それらは考え方自体決して目新しいようには思われないものがほとんどである。現実の作業での人間の思考過程や開発方法を種々の技法に照らし合わせてみると類似点を見いだせる場合も少なくない。ただ、技法・手法という形で整理して論理的に明文化してあるかどうかの違いである。

このように考えてみると、ソフトウエア開発の青い鳥は案外身近にいるのかも知れない。

ソフトウエアの世界においては、OSを頭として舶来品で占められている（TRONがある？）。

ソフトウエア産業界の「カンサイ・ヤマモト」「ハナエ・モリ」出でよ！

②　CASEツールについて

CASEツールと呼ばれているものには、机上で人間がやっていること（技法・手法に基づいた作業）をコンピュータにのせただけというものが少なくない。

PAD，SPD，HCPなどの技法を支援するためのチャート・エディタなどはソース・プログラムとのトランスレータを用意して、ソース・コードの自動生成、またソースからチャートへの逆変換までやってくれる。これらのチャートを設計ドキュメントとして実際に手書きで作成している者にとっては、慣れれば有効なものとなる。チャート、ダイヤグラムなどと呼ばれているものの追加、移動、変更、削除などの編集作業は手書きではばかばかしいだけの苦痛で退屈な作業であるからだ。そのうえ、ソースにまで結び付けてくれるとなると便利この上ない。

ここ何年かDFDをサポートするCASEツールがたくさん出てきた。DFDは要求仕様から設計の上流工程までを記述するのに有効なダイヤグラム技法である。米国では結構生産性を上げているらしい。そこで日本でも・・・となるわけだが、果たしてソフトウエア・クライシスに有効な薬となりえるだろうか？

米国ではDFDをサポートするCASEツールが登場する前から、机上でDFDなどのダイヤグラム技法が使われていた。そして、その手書きの編集作業の煩わしさはソフトウエア技術者の思考の流れをしばしば中断させていた。だから、DFDなどをサポートするCASEツールの登場は待ち望まれていたわけで、出るべくしてぞろぞろ出てきたCASEツールであると言える。そのうえ、コンピュータを使うことでディスプレイ上でドキュメントを階層化詳細化して連関表現することができたり、ドキュメント上の、またはドキュメント間の矛盾不整合をチェックしてもくれる。また、エディタで作成された要求・設計工程段階の情報は、辞書とか百科事典とか呼ばれるデータ・ベースに蓄えられ、様々な観点から開発中、開発後のシステムを分析したり眺めるということも可能となる。

だが、日本ではDFDという技法自体が開発現場で広まる前に、DFDという技法がCASEツールに詰め込まれて輸入されてしまった感がある。この手のCASEツールを導入するためには、現状の開発環境、開発体制、開発方法、ドキュメント体系を改めなければならないところも多いと思われる。さあ明日からこのCASEツールを導入して使っていこうというわけにはいかない。何段階かのステップを踏む必要がありそうである。例えば、社内のある小規模のプロジェクトをテスト用プロジェクトにして試してみるとか、社内においてCASEツールでサポートされているダイヤグラム技法の教育を実施するとかの社内努力が必要である。

しかし、こうも予想できる。構造化手法がストラクチャード・プログラミング言語を使うことによって浸透し今となっては当り前になったように、諸々のダイヤグラム技法もそれらをインプリメントしたCASEツールを使うことで浸透し当り前となってしまうのかも知れない。

新入社員教育にDFDを取り入れている企業も出てきている昨今、輸入されてきたCASEツール群が今後どのように日本のソフトウエア事情を変えていくのか楽しみである。

CASEの次にくるもの

今は机上の技法をコンピュータで支援させるものばかりだが、CASEの次にくるものは、コンピュータを用いなければ成り立たないような技法を携えてソフトウエア技術者たちの眼前に現れてほしい。

③ 本当のソフトウエア・ツールは時代と困境を越える

本質的な意味でのソフトウエアの開発プロセスというものは小説を書き上げるプロセスに近いと考えている。また、私の理想とするソフトウエア技術者像は芸術家と呼ぶべきものでなければならない。

テーマ（ユーザからの要求、またはパッケージ・ソフトの開発要件）が出る。小説家（ソフトウエア技術者）は、それに基づいてひとつの物語（ソフト、システム）を書き上げるわけである。登場人物（エンティティ）の設定、登場人物間の関係（リレーションシップ）、登場人物の性格（アトリビュート）付けなどを考えたりしなくてはならない。イメージを固めるためには絵コンテ（ダイヤグラム）を描くことも必要である。大まかな話の流れを決め（概要設計）、話と話との間をうまくつなぎながら（モジュール間インターフェース）、徐々に話を具体化（詳細設計）していく。そして一応書き上がったら、推敲や校正をしたりストーリー（プログラム）に矛盾がないかチェックする（テスト、デバッグ）。

以上のようにソフトウエアの開発過程は小説家のそれと酷似している。このことがソフトウエア開発においては悩みの種になってくる。

原稿依頼（システム開発依頼）が出版社（ユーザ）から入る。締め切り（納期）は来月末で、しかも原稿量（ステップ数）が多い。これを書き上げるには、どうしても人手が必要だ。そこでアシスタントを雇って、彼らに指示を出しながら作業を進めることにしたのだが、アシスタントといってもズブの素人からある程度経験のある者までいろいろである。うまくいくだろうかと不安になる。しかし、また今後もこのような作業量の大きな仕事が入るかも知れないし、そのときにまた彼らに手伝ってもらわねばならない。今回は彼らの教育も兼ねてやってみるか、と作業に取り掛かるのだが、一人で書いていたときのようにはいかない。複数でやるのだから、まず作業分担をしなければならない。そのためには最初からストーリーを固定化する必要がある。こうしたほうが話がおもしろくなると思っても、途中で変更を加えることはメンバー全員の作業のやり直しとなる場合が多い。一人で書くときにはイメージの固まったところから書き始めればよかったが今回はそうはいかなくなった。抱いている登場人物のイメージがアシスタントたちにうまく伝わらない。それどころか誤解したまま書いている者、話を勝手に変えてしまう者もいる。締め切りだけが気になってひとりひとりの作業状況を把握する暇もない。

何とか締め切りに間に合ったものの、そこにはもはや芸術的な繊細さなどかけらもない。矛盾だらけの登場人物が支離滅裂のストーリーの中をただ無闇に動き回る。出版社は呆れ果て、読者は離れる。こんなことなら締め切りを延ばしてもらってでも始めから一人で書いていればよかったと思ったときには後の祭。

以上のようなことは、小説家の場合このような例はないが（二人で合作という例はあるが、その場合お互い気心も知れていて一心同体という感じなので一人と見なしてもよいと考える）、ソフトウエア開発はプロジェクト・チームでのグループ作業となるケースがほとんどなので、そのことから発生する様々な問題が開発の大きな壁となって立ちはだかる。

ソフトウエアを小説などと同じように人間の創作物と見なしたとき、プログラミング言語やコンパイラやコンバータなどのツールは重要ではなくただ無力な体をさらすだけである。そのとき必要となってくるツールは、人間の思考を助け考えを整理して固めることを支援するものでなければならない。アウトライン・プロセッサ、アイデア・プロセッサと呼ばれるものがそれに該当する。もちろん、このツールはソフトウエアの開発だけに使われるというものではなく、思考を作業とするあらゆる職業の人たちに使われるべきものである。

人間が思考活動を停止しない限り使われるべきツール。それは時代や困境を越えて存在する。

フローチャートから始まった図表による種々の技法は、よりよいアイデア・プロセッサを生み出すための様々な試みとしてとらえたい。

④ ニューソフトウエア？

ＶｉｓｉＣａｌｃ（スプレッド・シート・ソフト）がコンピュータ上でデビューを華々しく飾ったとき、人々はその使い易さに驚いた。しかし、それをどういうふうに利用すれば有効なのかという問題は残った。そこで、「いますぐ使えるビジネス・フォーム集」などというものが出されたりする。ソフトウエアがハードウエアの利用技術と定義されるなら、このようにソフトウエアを利用する技術を何と呼べばいいのだろうか？　ソフトウエアとして仕組みだけは用意されていてハードウエアを意識しなくてもいいが、そのままではすぐには使えない。そうしたソフトウエアを利用する技術、ソフトウエアのソフトウエアと言ったところか。

ＶｉｓｉＣａｌｃの次に登場したのがＨｙｐｅｒＣａｒｄであった。テキストや絵や音などで表現される分散した情報を、人間の好奇心に基づいて結び付けるのに適したシステム・ソフトである。ＨｙｐｅｒＣａｒｄを利用して作られたものはスタックウエアと呼ばれている。スプレッド・シートでのフォーム集に当たるものである。

ハードウエアは電子技術が、ソフトウエアはプログラミング技術が支えていた。スタックウエアではアイデアが支えとなっている。　つくづく使い易いアイデア・プロセッサが必要であると考えてしまう。

若手最後の年に若手の会に参加して

(株) SRA
田中　慎一郎

ふと気がつくと３０目前．『若手の会にも参加してみたかったなぁ』と「チラシ」をみると，今年はいつにも増して面白そう．オブジェクト指向だのエキスパート・システムだのの話を積極的にすることはできないけど，『これは何とか参加しなければ』とばかりに書いたポジション・ステートメントがこれ．私や今回の PC の一人の渡邊さん達でやっている「 SEA 環境分科会(SIGENV)」の現在のテーマ「プロトタイピング」から，ちょっと今考えていることをならべたててごまかそうと思ったわけね．

プロトタイピングについて

軸になることは以下の３点であると考える．

- 分野知識等の収集およびその体系化
- プロトタイピング手法およびその方法（論）の整備
- これらに基づく支援環境の整備

まず，ソフトウェア開発は基本的にプロトタイピング・アプローチをとるべきであると考える．しかし実際にはまだきちんとした方法（論）の裏付けがあるわけではない．現状では「適当にやるのがプロトタイピング」と言うのもあながち的外れではなく，特にうまくできなかったシステムは全て「プロトタイプ」であると言ってすませる傾向が顕著である．このため，目的あるいは特性から見たプロトタイピングの分類／整理を行ない，かつそれに呼応した方法（論）を整備する必要がある．

また現実的な場面を考えると，不明なところ全てに関してプロトタイプを作成することが，工期／工数上許されないことが多い．そのため自然と特に問題になりそうなごくわずかの部分を対象にすることになる．この場合，プロトタイプした部分とそれができなかった部分との関係，例えばインタフェイス部分の整合性といったものなどは非常に難しい問題となる．これに対処するための支援環境といったものが是非必要となる．

もちろん一般的な開発支援環境／プロトタイピング支援環境の整備も当然必要であり，この際もっとも有効とされるものは現在非常に属人性の高い各種知識の活用であろう．しかしこれらの情報をどう収集し，どう体系化するかということを一朝一夕に決めることは不可能である．現在できることは，実際のソフトウェア開発がどのように行なわれているのかを客観的に見つめ，ごく低レベルな試験環境を作成し，試用を繰り返すことのみと思われる．

プロトタイピング環境の作成自体が，プロトタイピング手法の確立に貢献することになろう．

このあたりに関して多くの方の考えを聞いてみたい．

... で，実際に私が参加したのはというと，「オブジェクト指向＋α」というタイトルが誰かによって付けられていたグループ．ま，大体そんなもんでしょう．

ただなんとなく面白そうだったんで，ここに決めてしまったんだけど，普段あまり考えてないことについて討論（になるかどうかは別にして，考えてみることが）できるのではと思ったわけ．

実際にどんな話が出たかということは，グループ討論・レポートに譲っちゃうとして，問題は，

『これじゃポジション・ペーパの書き直しなんかできないじゃないかぁ』

ということ．

それで，ポジションの方は別にどうも変化しなかったということに（勝手に）して，ワークショップそのものについて感じたことをまとめてみたりする．

１．あまりたくさんの人と話せなかった．

ワークショップにはかなりたくさんの人間がいたはずなんだけど，あんまりたくさんの人と話した印象がないんだよね．雑談はそれなりにしたんだけど，ワークショップらしく突っ込んだ話を満足いくまでできたという感じはしてないな．

一つにはうちのグループが小さかったためなんだろうけど...　もう少しうまく配分できる方法はないのかなぁ．でも，強制しては全く意味がないしねぇ...

もう一つは，たぶん参加者が妙に大人しかったためかな．全体セッションの時でも何かいうのは大体決まってたし（それもほとんどPCだったりする），夜の呑み会にいたっては，（え〜，途中で仮眠をとったものがいうのも何ですが）『一体なんだってみんな揃って，あんな早くから寝てしまうんだ』と思うほど人がいなかった．

これって，ジェネレーション・ギャップなのかしら???

２．討論ポイントが漠然としすぎていた．

グループによって多少違ったんだろうけど，うちのグループでいえば，討論に持ってくきっかけがいつまでも絞れなかったという感じ．限られた時間しかないんだから，グループ討論にはいった瞬間から，すぐに討論が立ち上がるように，PC がもっと方向付けをしてしまってもよかったんじゃあないのかな．（それをひっくり返すやつがいるんなら，それはその方がいいわけだし...）

ただこれはやってみたからわかったことで，別に PC に責任はないんだと思う．むしろみんなが大人し過ぎる方が問題なんだな．案外 PC が引っ張ると，今度は結論まで PC が用意した通りになってしまったりするかもしれない．（確かに下手に何かいうと，何でもみんな納得されそうな恐さはあった．）

あと，全体テーマが何かあって，それをそれぞれ違う方向からまとめていって，最後に全体で討論するみたいなやり方の方が，他のグループの話を聞く価値も高いんじゃないかな．今回のまとめはどこも中途半端だったみたいなんだけど...　OBJ2 も LINC も HyperCard も StP もやっぱり自分でやってみなくちゃわからないから，やった話だけ聞いても，自分のとことどう結び付ければいいかわかんないもんなぁ．ほとんどのグループでは「銀の弾丸」なんて，言葉すら出てこなかったし...　やっぱり統一テーマが欲しいな．

３．場所がよくない

最大の問題はこれ．いくらなんでも，もうちょっと夜中に遊びにでれるとこの方がいい．部屋ん中で呑んでるだけだから安上がりだったんだけど，やっぱり何か間違ってる．新鮮な環境での呑みニケーションは，議論をずっと進めると思う．来年はもう私は関係なくなっちゃうけど，どこかもっといいとこ見つけて欲しいな（第２の盛岡を見つけるんだ!!　またとられたりして:-)).

****

ポジションペーパー　　（株）ＳＲＡ
岡田 正之

****

ソフトウェア開発工程の生産性や品質改善について、まず第一に上げられるのは、システム分析、設計技術者等の人材の育成にあると思います。現段階では、システム分析等をはじめるにあったて、該当分野の専門的システム技術者を置かないため（置けないというところかもしれませんが）、たまたま仕事のあいた人などを当てる場合が多いようです。この様な現状では開発工程の生産性は望めないのではないでしょうか。

そこで、各専門分野のシステム分析、設計者を長期に育成することが必要になり、各システム技術者が問題解決（今回の議題として取り上げられてた多くの開発技法なども含めて）の方法をより多く身につける事が要求されると思います。そのためにも、仕事に追われる事なく充分な知識習得の時間や機会を設けることや、このような開発技法をソフトウェア開発に積極的に取り入れていく姿勢が大事になると思われます。さらに、品質改善を望むのなら、問題解決の技術を身につけた専門のシステム分析技術者による、ディスカッションを行えばよりよいものが得られるのではないでしょうか。

また一方では、ユーザとのミーティングを重視することにより、ユーザがソフトウェアに対して何を望んでいるのかをはっきりさせる事が必要だと思います。例えば、ユーザが、強いて必要とは思われない機能を増してくれと言うことがありますが、その機能を増やすことによって、処理速度が遅くなったり、ディスクの使用量が増えてしまい、システムがユーザの考えていたものと大きくかけはなれてしまうことが、ままあります。その様な事がないようにシステムの分析、設計時にユーザとのミーティングを充分に重ねて、機能を増やす事によって発生するだろう問題点をユーザに指摘すれば、ユーザの考えているものにより近いものを作れるのはないでしょうか。

このように、これからの生産性や品質の改善には、ユーザの考えているものをうまく引き出す環境や技術が大切になり、また、その要求を実現する為の多くの開発技法を身につけて技術者の養成が必要になることでしょう。

さて，規定の２ページをクリアしたところで，さすがにこれじゃああんまりなんで，討論内容に関するコメントを最後に少し．

★　開発すべきシステム内容を「機能的に分割する」ということと，「オブジェクト指向」ということは相容れないのか．

我グループでは「普段自分は，一体どのような観点から設計を行ない，複数の人間に作業を割り振ろうとしているのか」ということを見つめ直すことを討論の出発点にした．一つの結論として，「機能的の分割することはあまりよくない」というようなものも出た．それに付け加えられる形で，「やはりオブジェクト指向的な方がいい」というものもあったと思う．

ここの疑問が解消されないのだ．一体「機能的に分割する」ということは，「オブジェクトを考える」ということとそれほどかけ離れてしまうことなのだろうか．

機能 -- function -- 関数....

要するに，いつまでたっても「オブジェクト」だとか「オブジェクト指向」だとかいったもののイメージが，自分の中にできなくて困っているだけの話なのだが，関数だとか function だとかいう場合には，「オブジェクト指向」とは相容れないものだとはいわれないような気がする．なぜ「機能」はダメなのだろうか．

作表する機能，作図する機能，編集する機能，表示する機能

こういったものはどうして「オブジェクト」としてみなせないのだろう．これが私にはわからない．これは「もの」ではないからだろうか．

すると，

作表する機能を持つもの，作図する機能を持つもの...

のように言い替えればそれは「オブジェクト」になるのだろうか．そうだとすれば，単なる言葉の遊びになってしまう．

確かに下手に機能分割してもいいシステムはできないとは思うが，それは単なる実現方法の問題なのではないのだろうか．このあたりをすっきりさせることが，当面の課題といえる．

とかいう感じで，ちょっと真面目ぶってみたけど，これって要するに自分の無知をさらけだしただけだったりするわけだな．

まあそれはそれでいいんだけど，何かどうもあんまりまとまった感じのしないワークショップというのが私の本当の感想．（チュートリアルはとってもよかったけど，）何かすごく楽しかったという印象もないし，すごく勉強になったという感じでもないし... 出張帰りでつかれてたからかなぁ... とりあえずは他の人がどんな rewrite してくるのかとても楽しみだったりする（と言うか，最初のやつみてないから何ともいえないんだけど...）．

まあとにかく，実行委員長／プログラム委員長をはじめPCの方々，本当にご苦労さまでした．参加者の方々，いずれまたどこかでお会いしましょう．

.... これってどこがポジション・ペーパなんだろう．まあいいや．

ワークショップ参加の後に思うこと

林　好一, (株) SRA
yosikazu@sra.junet

1. はじめに

このワークショップに参加申し込みをする際, 生産性向上のためにどんな技術が必要かを述べるポジション・ステートメントを求められた. そこで私は, プロトタイピング, ひいてはものごとを試すという段階が当り前になることが, 鍵であると書いた. ワークショップを終えた今, この考えは少しも揺るがないどころか, かえって強くそう信じるようになった.

2. 巷の新技術

ワークショップ期間中, 寺野孝雄氏による「Expert System 入門」, 柴山悦哉氏による「オブジェクト指向入門」と二つのテュートリアルを受けた. どちらの話題も近年興味を持ちながら, 勉強を怠っていたものだったので, たいへん興味深く拝聴した.

しかし, である. こういった最新技術をどうやって現場に活かせば良いのだろう.

3. プロトタイピング

プロトタイピングと呼ばれる手法がある. ユーザ要求を見極めるためにユーザ・インタフェイスの部分だけを作ってみるとか, リスクが大きいシステムの死活的な部分だけを試作してその実現可能性を見るなど, その内容は多岐にわたるが, 共通しているのは,「部分だけとりあえず作って動かし, 実際のシステム構築に必要な情報を得る」ということである. つまりタメシである. ここに, 新技術を, 研究開発でない現場の開発に持ち込める可能性がある.

4. タメシの必然性

近頃はやりのボージョレー・ヌーボー. これは, 通常収穫の翌年にならないとできないワインを, 特殊な方法で年内に作り上げてしまったものである. 作り方はわかっている. だが日本のメーカが同様なものを作るとき, いきなり大量生産をはじめただろうか. 否. まず少しだけタメしてみなければならなかったはずだ. 大量生産ではなく, その度に違うものを作らなければならない受託ソフトウェア開発では, なおさらである.

そこでプロトタイピングの登場である. ユーザやマネジャの目をかいくぐり, タメシをやるのには絶好の口実となる.

ところで, そのプロトタイピングを当然とする風潮を作ったり, そもそもそこでタメすべき新技術を知ったりするのには, どうしたら良いのだろう.

5. 情報ネットワークは人間ネットワーク

人に聞くだけで問題が解決する場合, 解決までいかなくても, 同様な経験を聞いて, 参考にできる場合, それを傍らで聞いていて, もしかすると自分の問題にも適用できるかも知れないと気づく場合がある. こういったことが起きるためには, 社内外での普段からの無駄話, SEA のセミナーやフォーラムのような場, 組織の枠を越えたネットワーク上での掲示板システム(BBS)などが, 重要な役割を果たす. 実際, ワークショップで上記のような例をまのあたりにしたものである. (詳しくは, 第4グループの討論報告を参照されたい)

現在, 開発とは直接関係ない仕事をしているが, 一時期開発部隊の人たちと交流がとだえてしまったとき, 自社のシステムのアップデートの内容さえ理解できなくなり, 自分の持っている技術情報の少なさに狼狽したことがある. 最近は社内の BBS システムを通じて, GNU の何なにのベータ・バージョンがネットワークのどこどこにあるとか, あるプロジェクトのためにこんなツールを作ったがほしければあげるとかの情報が, 座して手に入るようになった.

だが依然, 細かい情報はメイル, 電話, 直接会うなどして確かめなければならない. その時, 人間としてのつながり／ネットワークが必要になる. これがなければ, メイルも電話も無視される可能性がある. 結局のところ, 自分一人でできることばかりではないのだから. この人間ネットワークの実践は, オブジェクト指向の考え方や, 分散環境の構築法とも通じるところがあり,「モチはモチ屋」, あるいは「適材適所」という, 個人の良さを最も有効に引き出す組織の一形態に当てはまる.

種々の「モチ屋」の知合いを抱える, これは暮らしやすい環境を確保するための重要事項である.「モチ屋」の中にはマネジャ(意志決定者)も含まれる. 理解と先見の明のあるマネジャと話せるようになっておくことは, エンジニアにとって(そして組織にとって)良い環境を得るために必須と言って良い. つまるところ, マネジャが首を縦に振らなければ, 基本的に金も手間暇もかけられないのだ.

6. おわりに

プロトタイピングをはやらせよう. だがそれは, 単なる隠れみのである. 本当に大事なのは, プロトタイピングを許す環境なのだ. それがエンジニアにとっての「暮らしやすさ」であり, これがなくては, 個人も環境も恒常的な進歩は望めない.

ワークショップに参加して

(株)日立ビジネス機器
第2開発部第1グループ
森　博幸

1．はじめに

近年、高度情報社会の発達により、ソフトウエアのバックログが増大しているといわれている。そのため、ソフトウエア生産性向上は我々ソフトウエア開発者の急務な課題であるといえる。私は、ソフトウエア開発に携わる１人として各種の新しい開発技法に興味を持ち、本ワークショップに参加した。ここでは、ワークショップでの討論内容および参加後の視点でのソフトウエアの生産性向上について述べる。

2．グループ討論を通して

本ワークショップはテーマが「銀の弾丸」ということで、各種の具体的な新技法についてグループ討論およびディスカッションを行ってきた。その中で私のグループはリーダ佐原伸さんもとでオブジェクト指向、プロトタイピング指向について討論した。

私は入社以来、メインフレームのユーザプログラムの開発、保守に従事してきた為、ソフトウエアの生産性向上については興味があり、参加したわけであるがオブジェクト指向、プロトタイピング指向とは何かということを十分に理解せずに参加したため討論についていけるか内心不安であった。しかし、グループ討論は私の不安に反して、交流が主な目的の様な気軽なものであった。

第１日目は、各人の現状の仕事から問題点を提起しその中から新技術が使えるものを探したが、全員とも新技術を使う以前の問題に直面していることがわかりテーマとは少し離れた方に討論は展開した。

第２日目は、気分を変えてレンタカーを使い清里方面へ出かけ討論した。討論内容は前日のつづきで問題点への対応についてで、テーマとはあまり関係ないため、明日の発表について少し不安になった。その夜、テーマに関係のあるＨｉｐｅｒ　Ｃａｒｄの講習を佐原さんから受けた。感想としては「メインフレームではこう簡単にプログラムは出来ないのに、ワークステーションは進んでいるなあ」が実感であった。

最終日は各グループの発表であり、当グループはテーマを若干無視して討論過程と結論について発表し、最後にＨｉｐｅｒ　Ｃａｒｄの感想を私が発表した。

私は全体の討論を通じ、また各人の問題点からソフトウエア生産方法はまだまだ発展途上であるなと感じた。本討論では「銀の弾丸」は見つからなかったが、今後の仕事に参考になるものであった。

3．ソフトウエアの生産性向上について

討論ではオブジェクト指向、プロトタイピング指向といった新技術について具体的に論ぜられなかったが、今後のソフトウエア生産はこうした新技術を取り入れたものになっていくと考える。私の会社では数年前よりシステム設計からプログラム開発、システム保守間でを一括管理するシステム開発支援ツールを使い開発を行い、成果を上げている。しかし、これもＣＯＢＯＬ、ＰＬ／１といった手続型言語を対象としているため、生産性向上にも限界あるのではないか思う。そこで、今回のワークショップの中でも話題に出てきた第４世代言語が必要になると考える。私は、近い将来(５年後位)には新規システム開発において、第４世代言語とプロトタイピング指向を組み合わせた開発方法が主流になり、ソソトウエア開発の生産性、品質は確実に向上するものと思う。

しかし、現実的な問題としてこれらの新技法を社内及びユーザーにいかにして定着させるかがある。ベテランＳＥ、プログラマーには職人的気質が見受けられ、新技法に消極的なところがあると思う。従って、我々若手が率先して新技法を取り入れていかなければならいと思う。

4．おわりに

残念ながら本ワークショップでは、私の仕事にすぐに役立つものはなかった。しかし、現在考えられているソフトウエアエンジニアリングの最先端の技法について学びまた、他社の方々と交流出来たことは大きな成果であった。

また、私を含めメーカー系の参加者はごく僅かである反面、ソフトハウス系の方々の参加が多かったと思う。今後は、メーカーの方々も積極的に参加してメーカーの立場から、新技術について意見を交換してほしいと思う。

最後に、ソフトウエアエンジニアリングはこれからますます進んでいくと思われる。私は既存の技術にとらわれることなく前向きに新しい技術を取り入れていきたい。

－以上－

異文化体験

― SEA 夏のプログラミングワークショップ '88 に参加して ―

株式会社 SRA 環境開発部
桜井麻里
mari@sra.junet

1. 概要

今回のワークショップは、テーマがあってないようなものだった。私は、そういうのも好きである。

参加前には、ソフトウェア開発の方法論と、CASE ツールについて見知らぬ人と語り合おうと思っていたのだが、結局、『LINC の実習を通して4GL を使ったプロトタイピングについて考える』というグループに参加して、LINC を使った。informix-4GL を使ったプロトタイピングなら、仕事で体験して、有用性も認めていたので、4GL を使ったプロトタイピングについては、いまさら議論することはなかったが、LINC には興味があったし、一度ぐらい汎用機を使ってみるのもいいだろうと思って、このグループを選んだ。

結果的には、私は、参加したグループの中で異端児となってしまったし、グループ討論をなかなかかみ合わないものにさせることになってしまった。グループの皆さんには大変申し訳ないことをしたが、日頃、自分が味わうことのできない異文化を体験したことは、私にとって有意義であった。

このペーパーでは、その異文化体験の一端と感想を思いつくままに述べ、ワークショップの参加報告とする。

2. 会社における私の作業環境と業務内容

異文化について語るためには、まず、私の属している文化を説明しなければならない。それをいちいち説明することはここでは無理なので、私の作業環境と業務内容について、簡単に紹介する。

2.1. 作業環境

私がいるフロアには、1988年9月20日現在、机72個、SONY-NEWS37台、SUN3台、その他の UNIX ワークステーション5台、Macintosh シリーズ10台、SmallTalk 用マシン4台、LISP マシン1台、PC9800シリーズ28台、レーザプリンタ3台、レーザライタ1台があり、これらの多くはイーサネットで結ばれている。PC98は VAX や NEWS など UNIX マシンの端末として使われている。社内の他部門、社外ともつながっていて、JUNET を通じてメイルやニュース[1]で情報交換ができる。コピー機一台、ミニコピー機一台もあって、いつでも使える。このフロアでは、常時50人ぐらいが仕事をしている。

私は、その中の一台の NEWS のビットマップディスプレイを占有している。主なコマンドはこの NEWS にあるが、プロジェクトのリソースは別な NEWS にあるし、かな漢字変換のサーバーはまた別の NEWS だし、バックアップをとるのに必要なストリーマも別の NEWS についているし、プリンタも別のマシンについている。情報交換に欠かせないニュースも別のマシンにおいてある。だから、ネットワークが落ちたり、自分に関係のあるマシンが落ちたりするととても困る。でも、普段はそんなことちっとも気にしていない。

各種文書作成、プログラムの作成、デバッグ、テスト、各種情報交換、スケジュール管理、勤務報告書の管理など日頃の作業はほとんどワークステーション上で行なっている。

[1] いうまでもなく、電子メイル、電子ニュースのことである。ミーティングのお知らせや報告、各種質問事項のやりとり、意見交換など、さまざまな情報交換は、たいていメイルやニュースで行なわれる。今やこれをなくして、仕事はできない。

2.2. 業務内容

私は、入社以来4年半、ずっと UNIX 環境下で、事務アプリケーションの開発支援ツールの作成／保守／デモンストレーションをしてきた。現在は、米国の企業で作られたリアルタイム構造化分析手法を支援した CASE ツールの日本語化／デモンストレーション／サポートを担当している。

要するに、私は UNIX と MS-DOS しかさわったことがなく、UNIX 文化にドップリつかっていたわけである。ただし、HyperCard や SmallTalk など、最近話題になっているものは、身近に使っている人がいて、実物もよく見ていたので文化としては知っていた。

3. 異文化体験

上記のようなわけで、私は、汎用機も半二重端末も使ったことがなかった。今回、汎用機で LINC を使うというわけで、LINC 以前にいろんなことに感動してしまった。思いつくままにあげてみると以下のようになる。

3.1. 半二重端末と汎用機のこと

- キーボードにキーがたくさんついている!
  処理操作のための重要なキーが、たいてい、ホームポジションから遠くはなれたところにある。あれでは、疲れてしょうがない。
- 頼みもしないのに行番号がついてくるテキスト・エディタ、ディスプレイの右下端に必ず表示されるカーソルの位置は、"行番号や桁位置に重要な意味のある言語を対象としているんだゾ" という自己主張をしていた。C, Pascal, Lisp, Smalltalk, Prolog, 日本語, 英語などを編集するのには、行番号や桁位置は不要だからだ。やっぱり、汎用機というのは、事務処理アプリケーション実行専用機だ。
- 半二重端末であるということが、アプリケーションのユーザインターフェースの構築に多大な影響を及ぼしているということがわかった。あれで、快適なユーザインターフェースを持った開発環境を構築するのには、かなり無理があると感じた。

3.2. LINC のこと

- よく考えられて作られている。あの前提に乗っ取った作業をする限り、使い易い。LINC の考え方は、どことなく UNIX に通じるものがあるように感じて、私には、なじみ易かった。きっと、LINC は、使いたい人が自分たちのために、作ったツールなのではないかと思った。そう思って資料をよく読んだらやっぱりそうだった。10年以上前から、その核はあったそうだ。
- 少なくとも、旧バロースのマシン同士の環境の中で使われる分には、とてもよくできている。あれで、端末非依存、メーカ非依存であれば、すごいのに。でも、そうなったら、あそこまで一貫したサポートはできないでしょうね。
- はじめて使った汎用機のソフトが LINC で幸せだねとグループリーダの佐藤氏にいわれた。私も、そんな気がする。

3.3. グループ討論のこと

グループ討論のまとめは、他でやっているはずなので、ここでは省略。

私以外の全メンバは、4GL 初体験だった。プロトタイピングの効用も知らなかった。みんな、LINC のよさは認めたものの、自分が関わっているような大規模システムの開発には使えそうもないと感じていた。そこで、私は、大規模システムだと何が大変なのか、どうして LINC のようなものでは使えないと感じるのか、ということに関して質問して、問題点を引き出す役をすることになってしまった。かくして、論点は、だんだんずれていき、一時アブナイ雰囲気になってしまった。どうなることかと心配したものの、みんなの努力により、やっと合意が見られ、発表用のＯＨＰが完成した。朝の5時だった。

4. おわりに

グループ討論には、LINC の実習のためについてきて下さった、日本ユニシスの営業の片山氏も、ずっとつきあって下さった。議論の方向修正や、ＯＨＰ作成についての適切なアドバイスをいただき、本当に助かりました。どうもありがとうございました。

無料で LINC を使わせて下さった日本ユニシスの方々、LINC を使えるように交渉して下さった長野協同電算の佐藤さん、ワークショップを運営して下さったプログラム委員の方々、どうもありがとうございました。そして、一緒に討論につきあって下さったグループの皆様、おつかれさまでした。また、何かの機会にお会いすることがございましたら、気軽に声をかけて下さい。

それから、私にとって一番面白かったのは、実は、はじめのチュートリアルでした。面白いお話を聞かせて下さった電力中研の寺野氏、東工大の柴山氏、ありがとうございました。

5. おまけ

- ワークショップは、やっぱり、おいしいご飯が食べられるところがいい!! 全員一緒の食事も悪くはないけど、メニューが決められているのは、やっぱりつかれます。日に一食は、好きな食事をしたかった。来年は、やっぱり、盛岡にしましょうよ。
- ワークショップの一ヶ月後に行なわれたデータショウの会場で、偶然、あの片山氏にお会いしました。グループ討論の感想をお聞きしたら、「いやあ、だんだん自分の立場がわからなくなっていきましてねぇ、疲れましたよ。」とおっしゃっていました。片山さん、本当にご苦労さまでした。

ソフトウェア開発プロセスの探求

三菱電機セミコンダクタソフトウエア株式会社
第一技術部ツール課　槌峨山　亨

1　はじめに

私の所属するグループの業務は、マイクロコンピュータ用アプリケーションプログラム開発支援ツールの開発および技術サポートを行うことである。本レポートでは、開発業務の現状および開発効率向上への取り組み方について述べる。

2　開発の現状

マイクロコンピュータ用開発支援ツール開発は、次の点で汎用マイクロコンピュータ用のそれと大きく異なっている。

(1). マイクロコンピュータのリリースと同時にツールもリリースしなければならない

(2). 廉価かつ一定レベルの機能を有すこと

従って、ソフトウェア開発を非常に短期間で完了しなければならない。マイクロコンピュータ用のツール開発では、実際にマイクロコンピュータを動作させながらツールの評価を行わねばならない場合もあるので、量産前のマイクロコンピュータを使用して半ば試行錯誤的に開発を行わねばならないこともある。また、マイクロコンピュータ量産前からアプリケーションプログラム開発に入る顧客のため、ツールの方も間に合わせの未完成品を出荷する場合もでてくる。

このように間に合わせ的にツールを開発していくと、開発行程に乱れが生じ結果的に生産性が低下することになり、製品の品質低下にもつながることになる。そこで、私達は、ソフトウェア開発のプロセスを的確に把握し、ソフトウェアの生産性，信頼性を高めることを考えた。

3　開発プロセスの把握

実際に私達が試みようとしているのは、図１に示すようなプロセスモデルをつくりましょう、そして各開発行程のなかでなすべきことを明確にしましょう、というものである。従来は、各開発工程が明確でなかったため、ソフトウェア仕様の検討や評価についても充分に時間をかけていなかった。また、各工程で何をなすべきかということすら各人が認識していなかった。これではソフトウェアの品質は、向上しない。まず、開発工程をビジュアルにすることが大切であると考えたわけである。

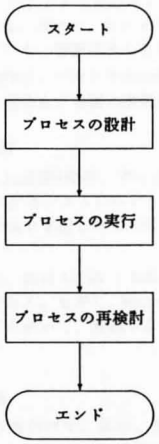


図１開発プロセスモデルの構造図

4　効率化の手段

従来、私達がソフトウェア開発手法を議論する場合、その主題はデバッグ方法であった。つまり、コーディング完了後いかにしてバグを収束させるかということが重要なテーマであった。しかし、ソフトウェアの品質を向上させるためには仕様設計から入念に行わねばならない。このことに気づいたことが、プロセスモデルの導入につながったといえる。

仕様設計の段階で開発物の品質を決める要因は何であろうか。１つにはプロトタイピングがあげられると思う。新規開発の多い私達のグループでは、プロトタイピングが開発プロセスのなかで特に重要であると考えている。そのためにはどのようなツールを用いることができるのだろうか。

この解はまだつかめていない。現在模索中であるというのが、正直なところである。

夏のプログラミング・ワークショップを終えて

㈱ユニテック　東京営業所
第２システム課　望月好光

1.　ワークショップに参加するにあたって

ワークショップに参加するにあたって、ポジション・ステートメントを書く為に参加募集内容に５つのアプローチが記述されていました。しかしこの５つのアプローチの意味が理解できませんでした。開発技法という事で記述されていたので、現在行っている金融・損保などの、汎用機にアプリケーションの開発技法というより、工程の中の１つであるテスト仕様書についていかに効率化していくかについて書きました。

しかし、後で送られてきたテキストの中のエキスパート・システム及びオブジェクト指向を読んでいる内に、５つのアプローチがどのレベルのものであるかが理解でき、自分が書いたポジション・ステートメントとは全く違うレベルのものであると気がつきました。

COBOLのアプリケーションの世界しか知らない自分にとって技術の先端を行く５つのアプローチについて、全く未知の世界であるが近い将来必要になってくるものであるので、このワークショップでそれに携わっている人達の話を聞き、どれか１つでも理解し視野を少しでも広げていく事を目標に参加しました。

2.　デスカッションにおいて

グループ討論を行うにあたり、プログラム委員による６つのテーマについて説明がありました。各テーマともにプログラム委員の方の確立した考え方があり、どれも面白そうだったのですが、その反面テーマが大きすぎてつかみどころがないようにも感じられました。

また質疑応答など白熱してきて各アプローチで専門用語が飛び交い、私自身理解に苦しむ内容が多々ありました。

討論後の内容を聞いてみて思った事は、まずオブジェクト指向＋αのテーマの中で＋αが、何であったのか、オブジェⅡの現状及び今後の課題はどうなのかが明確でなかったなど、全体的に話がまとまりきれてなかったように思いました。グループ討論する時間が足りなすぎたような気がします。

私はユニシスのマシンでＬＩＮＣを体験したいと思い「４ＧＬによるプロトタイピング」に参加させてもらいました。ＬＩＮＣを体験し４ＧＬ及びプロトタイピング（進化型）について話合う場に参加できた事など、有意義なグループ討論でした。

3.　ＬＩＮＣを体験しての感想

４ＧＬを認識する為にＬＩＮＣという開発環境を体験しました。ＤＩＳＰＬＡＹに画面を、作成していく、それによってＤＢの定義体及びトランザクション処理用プログラム、トランザクション管理ソフトウエアを生成し、オブジェクトまで作ってしまう。まずは’凄い’と思いました。　これならばプログラマーは必要なく、コンピュータを知らないエンドユーザーが、システム開発してしまう事がＬＩＮＣでは可能ではないか、私自身プログラム開発のマネジメント作業を行っているものとして、こういった作業は必要なくなり、かなりの工数削減になるだろう、と感じました。

4.　進化型プロトタイピングの課題

ＬＩＮＣに象徴されるように進化型プロトタイピングの必要性というものを、プログラム委員でありリーダーである佐藤さんが強調されていましたが、私自身プロトタイピングというより雛形というイメージのものしか使った事がないので、理解するのに時間が掛かりました。

自分なりに考えてみて、最終的に進化型プロトタイピング＝実システムであるならば、プロトタイピングにかなりの柔軟性が必要になってくるだろうし、４ＧＬがどこまで対応し、３ＧＬより簡素化していけるか、という事が今後の課題になっていくような気がします。

5.　４ＧＬの未来

近い将来必ず４ＧＬの時代は来ると思います。しかし、今は一部のマシンでしか開発できない、ＤＢの利用方法が限られている、既存とのインターフェースの問題、柔軟性に欠ける点、など課題は多々有ります。これから改良されて全ての課題がクリアされた時、４ＧＬが「銀の弾丸」の如くこの世界に殴り込みを賭けて来る事は間違いないでしょう。

私は今汎用機による３ＧＬでのシステム開発を行っていますが、すぐ後ろには４ＧＬが迫っているといっても過言ではないような気がします。

6.　ワークショップを終えて

今回ワークショップを終えて一番印象に残った事は、同じコンピュータの世界で仕事をしても自分の知らない分野で仕事している人達の話が身近で聞けた事です。しかし、聞くばかりでその人達の話の中に入り込む事ができず、自分だけが取り残されているような気がしました。

これからは、もっと先を見ていかなければ４ＧＬなど次の世代に対応して行けなくなるのではないか、と自分に対して危機感を感じました。

以　上

ポジションステートメント

要求仕様定義からの自動コーディングを求めて

（株）岩手電子計算センター
藤村　洋

はじめに

私は過去に日立製作所のSPDSを利用してあるシステムを開発したことがある。定められたフォーマットのドキュメントをまとめていくだけで設計書ができ上がることは大変効率的ではあったが、手書きをベースに作業をするため、信頼性とメンテナンス性に問題があるように思う。また、設計内容を顧客に確認してもらうためには、プログラムを作成して、テストフェーズを経てからでないと出来ないため、かなり時間がかかり問題である。ここでは、いくつかの問題をピックアップして、期待する開発環境を考えて見る。

1　要求仕様定義書からプログラム仕様書作成時の問題

(1)要求仕様定義書で記述した内容とほとんど同じ内容をプログラム仕様書にも転記している。そのため記述量が多いし、記述漏れが発生する可能性がある。

(2)機能単位に記述されている要求仕様定義書の内容をプログラム単位に分解する作業が繁雑であり分類漏れが発生する可能性がある。

(3)プログラム仕様書に追加記述した内容を要求仕様定義書にフィードバックするさいに意識的に人間がメンテナンスをしなければならないため記述漏れが発生する可能性がある。

2　プログラム仕様書からプログラム作成時の問題

(1)プログラム仕様書で記述した内容とほとんど同じ内容をプログラムにコーディングするようなものであり、漏れなくコード化しているか不安であるし、記述漏れが発生する可能性がある。

(2)プログラム作成時に追加記述した内容をプログラム仕様書にフィードバックするさいに意識的に人間がメンテナンスをしなければならないため記述漏れが発生する可能性がある。

3　システム開発中の顧客対応時の問題

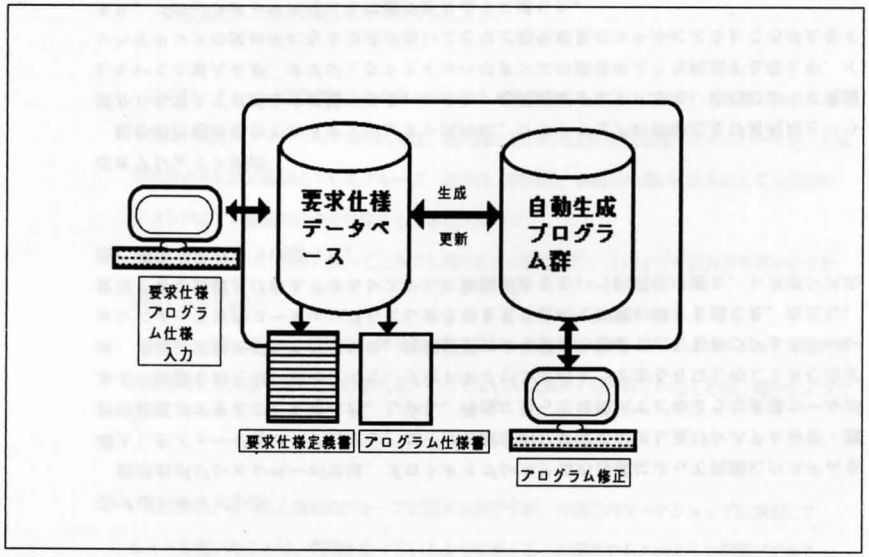
(1)開発の途中経過を仕様書として顧客に提示しにくいし、その資料を作成しても開発は進行中なので、すぐに陳腐化してしまう。

(2)設計内容を顧客に確認してもらうためには、テストフェーズを経てからでないと出来ないため、開発期間待ってもらう必要がある。

4　要求仕様定義からの自動コーディングシステム

すぐに、実現できるかどうかはともかくとして、望まれる開発環境システムを次ページの図をもとにを考えてみる。

（1）基本とする考え方
（a）情報（システム仕様）は、データベース化する。
（b）情報（システム仕様）は、それぞれのドキュメントに変換される。
（c）情報（システム仕様）は、プログラムとして生成される。



（２）入力について

(a)要求仕様を入力する。

(b)プログラム仕様書を入力する。

(c)コーディング内容を　入力する。

（３）処理について

(a)入力された要求仕様定義書のそれぞれの内容をプログラム単位に分類し、仕様書を作成する。

(b)仕様書からプログラムを作成する。

(c)要求仕様定義書,プログラム仕様書、プログラムへの修正はお互いに反映される．（データベース化）

（４）出力について

(a)要求仕様定義書を編集して打ち出す。

(b)プログラム仕様書を編集して出力する。

(c)プログラムを出力する。（自動コーディング）

最後に

要求仕様定義をもとに、プログラム仕様書、システム仕様書を作成するためにはエキスパートシステム的な考えを必要とするであろうし、自動コーディングのためには４ＧＬの発展が必要であろう。

現在の、鉛筆による力仕事のＳＥという環境から、仕掛けをスマートに創造する環境へと、１日も早く脱皮したいものである。

－　ワークショップに参加してみて　－

㈱　岩手電子計算センター
吉田　慎哉

はじめに

ワークショップ参加前に提出したポジションペーパでは、『ソフトウェア開発工程の生産性や品質の改善について、今後どんな方策を講じるべきか』というテーマで、

・システム分析におけるユーザを含む組織化したプロジェクトの編成と、メンバーの役割の明確化。
・プロトタイプシステムの作成。
・システム開発方法論、システム開発支援ツールの検討プロジェクトの発足。

という３つの方策を上げ、中でも、検討プロジェクトの発足が重要と考えていることを述べた。今回のワークショップは、いくつかの開発技法を取り上げそれぞれの長所・短所についてグループ討論によって比較検討を行うといういわばこの検討プロジェクト的な内容で行われ、その中で実際に開発技法を体験することによって改めて重要性を再認識することとなった。

①プロトタイピング

前回のポジションペーパでは、プロトタイプシステムの作成によって実際にシステムを導入したイメージをユーザに示し、ユーザの潜在的な要求の引出し及びシステム分析・設計の確認ができることを述べた。しかし、作成に当っては現状でどのような支援ツールがあるか知識を持たない私にとって、プロトタイピングの１つである４ＧＬのＬＩＮＣなどは、容易に実働モデルを作成でき、仕様変更による修正も容易で、なおかつジェネレーションによって自動コーディングしてしまうのを見て改めて知識の薄さを感じた。ただし、使用できる装置及び導入できるシステムに制限があるという汎用性の面と、レスポンスの面に問題があるように思った。

②オブジェクト指向

既存の技術の集合であるオブジェクト指向は、ソフトウェアの標準化及び再利用という面からも考えて有効な方法論である。しかし、応用範囲が広く、また、応用によって発展していくと言えるが、オブジェクトとインヘリタンスの関係のような相反する考えや、インヘリタンスの決め手になる方式が無いことなど開発要員のスキルによるところが大きく、また、ブラックボックス化にも問題があるように感じた。

③グループ討論（ＣＡＳＥツール）

私達のグループでは、ＣＡＳＥツールの１つである上流工程開発支援のＳｔＰを使用し、図書館システム開発を行った。しかし、ＳｔＰは、実際にはＮＥＷＳのビットマップディスプレイ上で作動するということだが、今回は装置が無いことから、机上でＳｔＰの手法で行った。そのために、ＳｔＰのメリットの１つである操作性や自動生成の面を見ることはできなかったが、要求定義決定における上流工程での多方面からの妥当性のチェック等いろいろな機能があり、それらによって上流工程における生産性及び品質の向上に効果が期待されることがわかった。続いてＳｔＰとオブジェクト指向の比較まで手を伸ばしたが、白熱した討論により（酒のせいという話もあるが．．．）、流れが発散方向に向い、ついに結果を見るには至らなかった。しかし、ワークショップの趣旨である『結果ではなくそのプロセスが重要である』ということからすれば、良かった．．．ことにしよう。

最後に

今回のワークショップは、『銀の弾丸』の可能性を探ろうと、いくつかの開発技法が取り上げられた。しかし、私自身『銀の弾丸』の可能性を探る前の段階で、開発技法の一部を体験したに過ぎないが、現状の開発環境の把握に役立ち、啓発されたことに意義があった。また、これからは、いかにして開発環境を使いこなしていくかにあると感じづにはいられなかった。

夏のプログラミングワークショップに参加して

末次真之 〔株式会社システムプランニング〕

このワークショップに参加された方々は，大きく色分けすると二つに分かれると思われます．一方は，現在のソフトウエア業界内では，最先端と思われる第四世代言語やエキスパートシステム等の考え方に既に馴染んでいるグループ，他方は，制御系，事務系の違いはあるにしても従来のソフトウエアの範疇のなかで仕事をしてきたグループ．

但し，私自身この業界に入って三年にも満たない人間なので，上のような区分が妥当かどうか判断できませんが．

しかし，ワークショップのテーマとなった４ＧＬ，エキスパートシステム，オブジェクト指向等の考え方にすんなり入っていける人と，そうでない人に大きく分かれていたことは，確かなようです．

私自身は，もちろん後者のグループに属する訳ですが，今回このワークショップに参加してもっとも驚いたことは，先端を走っているように見えるこの業界でもこのような階層ができてしまうということ自体です．

今後もこの業界に当分いるつもりでまたコンピューター般の幅広い分野で仕事をするつもりである私にとっては，大きな刺激となる一方で，このままではとり残されるのではという焦りも感じさせられました．

ところで，従来のソフトウエアの考え方で仕事をしてきた人々は，今回そうであったようになぜ，詳しい（あるいはかみくだいた）説明を聞いても考え方のとっかかりすらなかなか掴めなかったのでしょうか．（私のことですが）

おそらく，ソフトウエアに対する考え方の基本的な枠組み自体が，異なる（古い）のだと思います．

しかも，当事者自身（どちらの側ともに）は，意識していないことが多いとおもわれるので，その人が，使っているソフトウエアの方法そのものによって二つのグループが，自然とできてまうことは，私（たち）にとってかなり重要なことだと思います．極端なことを考えると，その人の仕事の性質によっては，いつの間にその人の技術が，ほかではまったく通用しなくなっているという事態もありえます．

もっとも，これは一般的な話で４ＧＬが先にそうなる可能性もありますが．

技術の変化（進歩？）が直線的なものでなくランダムなものだとすれば，結局は何をやってても危ない訳で，つまみぐい程度でもいろんな方面に首を突っ込んだ方が，私（たち）の延命の手立てになるのではないかと今回痛感しました．

いずれにしても，自分の知識，技術，アイデアを過信（あるいは盲信）すると，その瞬間からその人の技術的変化（進歩）や知識の根本的な変更が望めなくなるのは，事実のようです．

ところで，そのようなつまみぐいをするには，今回のワークショップはなかなか良い機会だったと思いますが，初日にいきなりハイブローな話をプログラム委員の方にされると，私と同様に，頭が混乱してしまう方々がいるようなので，意気込みはわかるのですが，もうすこし従来の人間にとっかかりやすいように，共通の土俵から話をはじめてもらえればよっかたとおもいます．

なお，今回のワークショップの内容そのものについては，私の頭の枠をはずれたことで，ついていくのが精一杯でしたので特に感想といったものは，ありません．

ただ，エキスパートシステムもすでにＡＩ研究の副産物となるつつあるという話から，では一体ＡＩとは，具体的には何をやっているのか？という疑問がわいた程度です．

最近，日々の業務およびそれにまつわる勉強に埋もれつつあった私にとっては，今回のワークショップへの参加は，タイムリーな刺激となったと思います．

後輩や同僚にもぜひ参加を勧めようと思いますので，ワークショップがますます刺激的なものとなるようプログラム委員の方々の御活躍を願っております．

ポジションペーパ　三菱電機コントロールソフトウェア（株）　中林伸博

私は今回初めてこのワークショップに参加させて頂いたわけだが、ワークショップというものを体験したのはこれが初めてである。講演会やシンポジウム、セミナーとは異なり自由な雰囲気で参加メンバーと一緒に討論するというワークショップのあり方は素晴らしいものだと思う。夜遅くまで討論したことがとても印象的なものとして心に残っている。今後も機会があればどんどん参加してみたいと思っている。

今回のグループ討論では、私は第６グループに参加し”ＣＡＳＥツール、オブジェクト指向について考える”というテーマで討論を行った。私がこのテーマを選択した理由は、テーマにもあるようにＣＡＳＥツールについてとても興味をもっていたからである。ＣＡＳＥツールはソフトウェアのライフサイクルにおいて、要求分析、要求定義、仕様設計といったソフトウェア開発の上流工程をサポートするツールである。

私の仕事の内容も会社に入った頃から比べると、上流工程の定めた仕様に基づいてプログラムを設計、製作していく業務から、上流工程の仕様を決定する業務へと変わってきている。このような上流工程の仕事は始めたばかりなので、なかなかうまく行かない事が多く苦労している。その理由としては、私自身、客先の要求をうまくまとめる能力がまだまだ足りないためであると考えている。 また、そのような能力をつけるような機会を得ようとしなかったためである。このような時に、ＣＡＳＥツールの存在を知り、その内容について詳しい話を聞く機会を探していた。そして今回偶然にもこのワークショップでＣＡＳＥツールについて知る機会があったので、迷わずこのグループに参加した。

グループ討論の中で簡単にＣＡＳＥツールについてのレクチャを受けたのが、ＣＡＳＥツールについてもう少し詳しく説明すると、今まで考案されてきた数々のソフトウェア開発の方法論（技法）の有機的な集まりであるといえる。従来のソフトウェア開発の方法論には、データフロー図（Ｄａｔａｆｌｏｗ　Ｃｈａｒｔ),データＥ－Ｒ図（Ｅｎｔｒｙ　Ｒｅｌａｔｉｏｎｓｈｉｐ）、データ構造図（Ｄａｔａ　Ｓｔｒｕｃｔｕｒｅ）などがある。従来は、これらの方法論を用いて人が机上で紙に絵を書きながらソフトウェアの設計を行ってきたわけである。ＣＡＳＥツールは絵を書くという作業を、紙の上からワークステーション上のビットマップＣＲＴに移したものである。また人が設計したその内容に対して、ある程度のチェックを行ってくれるものである。これにより人は設計作業だけに、没頭できるので、能率的であると考えられる。

このＣＡＳＥツールは現在Ｓｕｎワークステーション上にて使用することが出来るそうであるが、非常に高価なものである。（バイナリ　３００万円）　値段のことはさておき、このＣＡＳＥツールが日本で広まるかという事について、討論の場で話題となったのだがまだまだＣＡＳＥツールは日本では普及しないだろうという結論になった。というのは先程述べたようなソフトウェア開発の方法論自身が日本ではあまり普及しておらず、現場の実作業に使われていないため、ＣＡＳＥツールを導入しても、それを使いこなす土壌がないのではないかというのがその理由である。私自身の場合を考えても、当然勉強不足の点も大きいのだが、ソフトウェア開発のためのこれらの技法に関して殆ど知識がなかったのが実状である。このような状況ではＣＡＳＥツールを導入しても、成果を上げるまでには多くの金と時間が必要となる。そこで、グループの活動方針としては、ただ話を聞いただけではなんの成果もないだろうということで、ＣＡＳＥツールを体験してみようということになり、ＣＡＳＥツールの一つであるＳｔＰ（Ｓｏｆｔｗａｒｅ　ｔｈｒｏｕｇｈ　Ｐｉｃｔｕｒｅ）にメンバ自身がツールになりかわり、何かある例題を設計をしてみようということになった。これによりＣＡＳＥツールの有用性を探ろうとした訳である。このことは私にとっては願ってもないことであり、この作業を通して、このようなソフトウェアの手法があるのだということを実体験できたことがよい経験になった。今後の私の業務に関して、このようなソフトウェア開発の手法を取り入れて、設計フェーズに対して実践して行きたいと思っている。

さて、もう一つのグループのテーマであるオブジェクト指向だが、会社のなかでも試しに二、三取り組んだ事例があるだけで、私自身の業務の中にはこのような考え方はなかった。だたなんとなく興味があったので、どのようなグループ活動ができるのかとてもたのしみであった。結果としては、メンバの大半がオブジェクト指向というのの考え方をきちんとつかんでいなかったため、活動も中途半端なところで終わってしまったが。しかし、グループ活動の中で繰り返された試行錯誤というものは、私にとって、今後のためになる体験であったと思っている。このグループ活動で得られた体験を基にして、オブジェクト指向による設計をやってみようと考えているからである。

チュートリアルでも話があったように、オブジェクト指向は、いままでの既存の技術のよせ集めであり、その定義は流動的なものである。そして理論もほとんどないに等しいので、私にオブジェクト指向による設計作業が出来るのかは不明だが、とにかく何かを始めようという気持ちで取り組んで行こうと考えている。

ＣＡＳＥツールやオブジェクト指向が、私の業務に対して果して『銀の弾丸』になりえるかどうかは、私のこれからの仕事の進め方次第なのだが、願わくば、『銀の弾丸』になって欲しいものだと期待している。

以上

形式的仕様の将来

静岡大学大学院電子科学研究科　太田　剛

'８８夏のプログラミングワークショップは、ソフトウェアというモンスターを倒すための銀の弾丸を考えるというテーマのもとで行なわれた。銀の弾丸になりそうないくつかのテーマごとのグループに分かれて討論を行なったが、私は形式的仕様のアプローチをとるグループに参加した。

ソフトウェアの開発の上流工程の中で、作成された設計が正しいかどうか、あるいは効率が良いかどうかといったことは、設計対象領域のことをよく知り、理解している専門の設計者の能力に依存しており、それを評価することは実際に稼働するシステムができあがるまではわからないのが現状である。この問題点を解決すべく提案されたのが形式的仕様のアプローチであると考えている。形式的である、すなわち、シンタックスとセマンティクスが正確に決められているために、形式的仕様はそのまま実行（インタープリット）することができ、正しさあるいは効率の良さ等が、設計の段階で評価できる。また、形式的であるがゆえに、設計書からプログラムを自動生成することも可能である。この場合、コーディング時に侵入するバグを防ぐことにもなる。このような点が、形式的仕様の特長であり、十分に銀の弾丸となりえるものと思う。

しかしながら、現在のところ、形式的仕様の中で時間の概念を取り扱うことが困難であり、銀行のオンラインシステム等のリアルタイムシステムを記述できないという問題点がある。これは、今後の研究課題であろう。また、形式的であるがために、書きにくいとか読みにくいとかいう点もあるだろうが、これは慣れの問題であろう。それよりは、形式的仕様言語の背景にある理論の教育のほうが大切だと思う。

いずれにせよ、銀の弾丸ができたからといって、ソフトウェアというモンスターがそれだけで完全に倒れることはないだろうと思う。銀の弾丸を使うのが人間である以上、それを間違えて使うことも起こるだろう。また、それ以上に重要なことは、現実世界にある非形式的（インフォーマル）な問題を計算機を使って解決するために、形式的（フォーマル）な形に書き下す作業が不可欠であり、これは計算機にはできない人間固有の活動であるということである。そして、これができてはじめて形式的仕様のアプローチ（あるいはその他のアプローチ）が生きてくるのである。形式的仕様のアプローチが成功して、世の中すべてがこの方法を使ってソフトウェアを作成することになったと仮定したとき、現在の意味でのプログラマーは存在しなくなってもなお、ソフトウェア技術者（現在の意味での設計者、ＳＥ）が足りないと叫ばれることになるのではないかと考えている。そういった意味で、本当の銀の弾丸は、能力のある人材の教育のし方なのかもしれない　:-)

「銀の弾丸」探しから得た「鉄の弾丸」，「銀メッキの弾丸」

三菱電機コントロールソフトウェア（株）
技術統括部　開発課

内海　保

1．「銀の弾丸」探しから「鉄の弾丸」探しへ

今回のワークショップでは、「銀の弾丸」候補として”形式的仕様記述”,”AI”,”4GL”,”オブジェクト指向”を挙げ、各候補が本当に「銀の弾丸」と成りうるかの討論が行なわれた。ここで自分を取り巻くS／W生産環境を考えてみると、たとえ「銀の弾丸」が見つかったとしても、それを発射できるピストルもなくガンマンもおらず、多分発射することができないのではないだろうかと考えさせられた。それなら、現状で誰にでも発射できる「鉄の弾丸」（＝現在の技術レベルでも有効なS/W開発支援手段）でまずは武装したい。

2．私の「鉄の弾丸」

「鉄の弾丸」による武装対象として、まずはドキュメンテーションを考えたい。このドキュメント作成には結構な時間を割いているので、その１割でも時間短縮ができればしめたものである。

ドキュメンテーションに対する「鉄の弾丸」は、UNIXワークステーションを用いたtroff環境である。現在のところ、S／W開発に伴うドキュメント作成は、手書き、あるいは、ワープロにて行なっている。修正の容易さ、流用ということだけを考えればワープロでも用は足りるが、大量のドキュメントの管理・検索という点でワープロは役不足である。この点に関して、troff環境ではUNIXの検索コマンド、バージョン管理ツールと言ったものが使用でき、ワープロより数段優れている。それに加え、troff環境の何よりもの魅力は、S／W製作環境とドキュメント作成環境が統合化されることである。

次なる「鉄の弾丸」は、情報交換のためのネットワークである。今回のワークショップでは、情報の僻地にいる私にとって魅力的な情報をいろいろと得た。この中には、直面している問題に対する解答のヒントや、今後の開発環境を考える上で大いに役立つ情報が、多く含まれていた。グループ討論での結論を引用するなら、〝人に聞くだけで解決する問題もある〟というところである。

二番目の「鉄の弾丸」を手に入れるためには、情報の入手経路が絶対に必要であるのである。そういう意味で、今回のワークショップで他の技術者の人との交流をもてたことは、口径3mmぐらいの「鉄の弾丸」を手に入れられたも同然である。

3．「銀メッキの弾丸」

ここで、「銀の弾丸」探しについて少し触れておきたい。

「銀の弾丸」候補の中で、私が一番興味があったのが”オブジェクト指向”である。この”オブジェクト指向”については、今回チュートリアルとグループ討論にてその概念を自分なりに整理することができた。それに加えて”オブジェクト指向”に対して、次の点で認識を新たにした。

> ・オブジェクト指向の概念は、C言語のような言語でのS／W開発にも注入することができる。

ここで、”オブジェクト指向”の”S／Wの部品化・再利用”への適用について少し触れたい。

”S／Wの部品化・再利用”にて重要と思われるのは、次の二点だと考える。

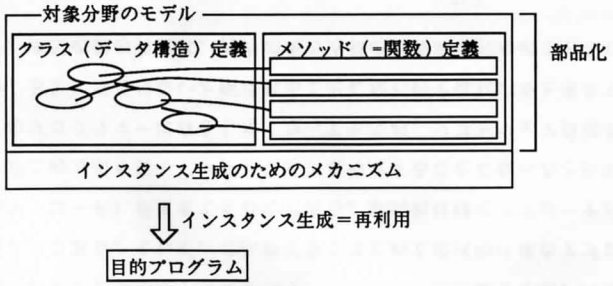
1)部品化において、いかに個々のプログラムを一つの塊とするか。
2)再利用において、必要な部品をいかに同定するか。

この問題をうまく解決するのが”オブジェクト指向”の概念だと思う。

そこで、S／Wの部品化・再利用への”オブジェクト指向”の応用を「銀メッキの弾丸」（なぜ銀メッキかというと、「鉄の弾丸」より「銀の弾丸」に少しは近いのではないかと、自分自身で決め込んでいるため）として位置づけたい。

4．「銀メッキの弾丸」を用いたS／W開発

「銀メッキ」の弾丸の材質は、C言語とする。この時のS／W開発のイメージ図を次に示す。



ここでインスタンス生成のためのメカニズムをどうすればよいかについては、具体的なイメージが浮かばないので今後の課題としたい。

5．最後に

今回のワークショップでは、「銀の弾丸」を見いだすことは非常に困難であるがゆえに、見いだすためには地道な努力が必要であることを痛感した。今後は、今回のワークショップで受けた刺激を忘れず、微力ながら「銀の弾丸」探しを行なってみたい。

SEA　夏のプログラミングワークショップ in八ヶ岳　'88に参加して

㈱日本情報開発
制御システム部ソフト第2課
深野　寛子

中央線に乗り込み、3泊4日の合宿生活に参加した感想を書き留めてみました。

今回のワークショップが終了して社会生活に復帰した今、糧になったものは何だろうと思い返しています。

一日目、二日目の、チュートリアル「エキスパートシステム」、「オブジェクト指向」を拝聴しながら、(様々な単語の意味不明な点も有りましたが、自己流の解釈をさせていただきました)　業務上にどのようなに、関わるのかと考えました。

確かに、読み物、聞き物としては、興味も覚えましたし、雑誌等を、読み返すこともしましたが。。。。

業務上、すぐに役立つ内容ではありませんでしたし、取りあえずは、業務上必要に駆られて得た知識があれば業務には支障をきたすことはありません。

この為言葉として、知ることは出来ましたが、知識としての蓄積にはまだ成り得ません。

また、色々な方々と話をして分野が同じでも、立場の違いを感じました。

今回のテーマが、前回、前々回に比べ、プログラム委員会の要求するものが明確にしにくいように感じました。

グループの成果発表の際、感じたことは、グループリーダーと着眼点が違うというものがあったようでしたが、何故グループリーダーは、軌道修正をしないかったのでしょうか。

グループリーダーの、位置付けが、テーマの提供者だけなのか理解できませんでした。

取り合えず、色々な事を、考える機会にはなりました。

最後に、ポジション･ペーパの、観点からは変更が有りませんので。規定のマージンで、作成し直したものを送ります。

生産性と品質

㈱日本情報開発
制御システム部ソフト第2課
深野　寛子

生産性…①原材料の量や労働者の数に対して、製品の生産される割合。
　　②どれだけ多く生産するかという割合。
品質　…[良･不良が問題になる]　品物の性質。

－ 三省堂　新明解国語辞典　より－

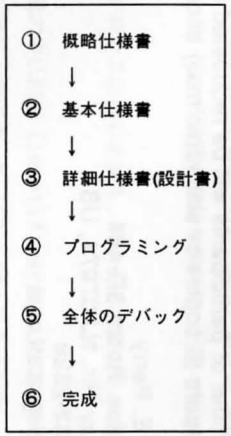
問)　ソフトウェアの品質とは?　生産性とは?
答)　無駄が無く、早く、バグが無い。
　　外部環境に左右され無い。
　　短期決戦。　ノウハウの蓄積。
　　壊れない。　消えない。。。。

＜　提　起　＞

プラントの開発において、短期間に一定の水準のシステムを作成するという目標が掲げられる。
目標を実現する際に要求されるのが、効率よく作業を進めることです。
いかに、手際よく作業を進めていくかにより品質等にも影響がでてきます。
効率を上げる為に、過去のシステムで実績があるパッケージやツールを使用するのも有効なことです。
しかし、むやみやたらにパッケージやツールの継ぎはぎでは、良品と呼ばれるものはできない。
　与えられた内容に対して、どこに着眼しなければいけないか。
何に注意しなければいけないのか、何を必要としているのか、といった点もきをつけなければならない。
そうしないと、せっかく苦労して生み出したものでも、不良品のレッテルがはられる可能性もあるのです。
　例えば、生産性やメンテナンス性等から、過去のシステムで実績があるパッケージやツールを組込んだところ、新機種のために有効な支援には、ならない状態となり、かえってデバック等に時間を浪費してしまったケースもありました。
　このように、工数に大きな問題を与える場合もありますが、どんなに、短いプログラムにおいても様々な、問題を含んでいます。
その、問題をできるだけ的確に掴むことが要求されてきます。

＜　問題解決とは　＞

プログラムの開発手順を左に示す。一般的な問題解決について考察します。



プログラム完成までの手順

③から⑥までの、工程はここでは、考えから外すことにします。

①と②で大切なのは、要旨と目的の明瞭化、共有語の定義、問題解決の方法ではないかと考えます。

①の段階では、要旨と目的の明瞭化が要求される。

（　プログラムを組む人でなくても読めるもの　）

②の段階では、実際のプログラムに則した問題解決の方法が必要となってくる。

この作業は、山道を歩いていて麓にある村へ行きたいのだけれども、分かれ道に道しるべがなくてどちらへいったらよいのかわからないような気がする。

目標が明確であるにもかかわらず、そのための手段や方法が不明である。

この時、再生的に作業するのではなく、創造的に作業していくことが大切だと思います。

何故ならば、再生的作業は、過去の経験における方法や知識のなかから当面の問題にあてはまるものを選び出すやり方なので、ノウハウだけが要求されることになる。　また、他人のやり方を真似するだけに成りかねない。

創造的作業とは、過去の経験に頼ることはもちろんであるが、それをもとにしてまったく新しい方法を作り出すというやり方です。

問題場面に出会ったとき

① 適度に動機づけること　（　モチベーション　）
② 場の位置を変えること　（　違った角度から問題をみること　）
③ 積極的に操作すること
④ 対象の機能的性質を吟味すること
⑤ 固着をさけること

＜　ソフトウェアの設計段階での考察　＞

基本設計

(1) 何を実現しようとしているか、機能的にとらえること。
(2) どの様に機能分割されているかわかること。
(3) タスク間のデータの流れが明確であること。
(4) タスク間の制御の流れが明確であること。
(5) タスクの機動要因(　外部　)が明確であること。
(6) タスクのシステムから見た位置づけが明確であること。

詳細設計

(1) 全体の中のどの部分であるか明確であること。　（　基本設計書と共通　）
(2) どの様なモジュールにより構成されているか明確であること。
(3) データの入出、流れがアイテムレベルで明確であること。
(4) Programの流れが明確であること。
(5) Coding方法がみえること。
(6) Programレベルでの特異な点が明記されていること。

＜　結論にかえて　＞

仕様作成の段階で、ムリのない仕様で、ムダのない設計を心がけることが大切です。
（　ここで言うムリとは、時間的･技術的の双方を含む　）
ムリとムダのある物を強引に進めていくと、その結果、ムラのある製品ができてしまう。
具体的な方法は提示できませんが、常に問題点を意識して、慣例に縛られないことです。
無意識におこなうのではなく、自分なりの考え方を、意識して作業すること、又、自己満足で終わらせずに、改善しつずけていくことだと思います。
ソフトウェアを作り続けていく上で、生産性と品質の向上は、追求されつづけるテーマだと考えます。
言い換えれば、明確な答えのないテーマだと考えます。

5th INTERNATIONAL SOFTWARE PROCESS WORKSHOP

Experience with Software Process Models

Kennebunkport, Maine, USA, 10-13 October 1989

(Sponsored by the Rocky Mountain Institute of Software Engineering in cooperation with ACM SigSoft and IEEE-TCSE)

ORGANIZING COMMITTEE

Mark Dowson	Takuya Katayama	Masatoshi Matsuo
David Notkin	Dewayne Perry	Colin Tully

The 5th International Software Process Workshop will focus on experience using (and evaluations of) software process models, particularly enactable (that is, executable or interpretable) models. This experience will reveal a number of crucial issues in the prescriptive and descriptive use of these process models and in their underlying support. Among the more important issues are the following:

Suitability of Existing Models

A model should be effective in describing and prescribing a wide range of actual software processes; it must also address a wide range of activities within each process including, in particular, individual and group interactions. Experience using the model should reveal:

- how *usable* are the models for guiding and controlling software development and maintenance?
- how well do the models *scale* from processes-in-the-small to processes-in-the-large?
- what activities are *covered* in the development and maintenance life-cycle?
- what policies of individual and group *coordination and cooperation* are prescribed by the models and what mechanisms (tools) and structures are needed for *supporting and/or enforcing* these policies?

Instantiation of Process Models

A software process model must be fully or partially *instantiated* to obtain a specific process definition for all or part of a project. Generic process models, which can be *customized* for particular projects by being instantiated in various ways, enhance the potential for process model reuse.

- what are the *characteristics* of widely applicable process models?
- how is an appropriate model (or appropriate customization of a generic model) *selected* to meet the specific needs of a particular organization or project?
- what are the mechanisms needed for *customizing and creating* enactable model instances?

Dynamics of Process Models and Model Instances

Process models will evolve over time as experience accumulates, the requirements change, and the applicability of a given model is extended; process model instances may change during their enactment, either adding detail not prescribed by the model (refinement) or modifying (and possibly creating) detail to accommodate unpredictable contingencies.

- to what extent can software process models be *evolved or adapted*?
- how are the models *verified and validated* and what are the mechanisms for gaining and using *feedback* on model usage?
- how are instances *refined and modified* dynamically?

The three day workshop, which will be held at The Shawmut Inn, Kennebunkport, Maine, will consist of intensive discussion of these issues by, at most, 35 participants. Prospective participants should submit a maximum 3 page position paper by 1 March 1989, explicitly addressing one of the workshop issues and suitable for publication in the proceedings. A small number of participants will be requested to prepare short keynote presentations to initiate discussion. Position papers (6 copies or electronic mail) should be sent to:

Dewayne E. Perry
AT&T Bell Laboratories, Room 3D-454
600 Mountain Ave, Murray Hill, NJ 07974, USA
tel: 201 582-2529
email: dep@allegra.att.com

Electronic mail submissions should include author's full postal address and telephone number.

第1回

ソフトウェア・プロセス・ワークショップ

参 加 者 募 集

主　催：日本ソフトウェア科学会 ソフトウェア・プロセス研究会
情報処理学会　ソフトウェア工学研究会（予定）
ソフトウェア技術者協会

期　日： １９８９年2月2日（木）～4日（土）　　**会　場：** 機械振興協会伊豆研修所（静岡県伊東市）
定　員： ３０名　　**参加費**（予定）：　3万円（一般）　8千円（学生）
実行委員長： 片山卓也（東京工業大学）
実行委員　： 阿草清滋（京都大学）　落水浩一郎（静岡大学）　岸田孝一（ＳＲＡ）
熊谷　章（ＰＦＵ）　久保宏志（富士通）　紫合　治（日本電気）
野村敏次（日本電子計算）　二木厚吉（電総研）　松本吉弘（東芝）

ソフトウェア開発の生産性や品質を考えるさいには，まず，いったいわれわれがどんな工程（プロセス）でどんな製品（プロダクト）を作っているのかを，はっきり認識することが必要です．

国際的なソフトウェア・エンジニアリング・コミュニティでは，ここ数年，とくにソフトウェア・プロセスを形式的にモデル化し，その特性を解明することに多くの関心が寄せられています．第８～９回の ICSE では，連続して「プロセス」が会議の基調テーマとして取り上げられ，それと関連して，ソフトウェア・プロセスに関する国際ワークショップ（ISPW）が 1984年以降すでに４回も開催されてきました．

このワークショップから生まれた主な成果としては，たとえば，開発にともなうさまざまなリスクの管理を含んだ渦巻型のプロセス・モデル（B. W. Boehm）や，木構造の契約型モデルにもとづくプロジェクト管理支援環境 ISTAR（M. M. Lehman），あるいは，厳密なアルゴリズムによるプロセス記述を目指したプロセス・プログラミングのアイデア（L. Osterweil）などがあげられます．

次回の ISPW は来年秋に米国 Maine州 Kennebunkport で開かれます．また，その次の第６回は 1990年のやはり秋に，今度は**日本での開催**が予定されています．討論テーマの候補としては，既存の各種モデルと現実のプロセスとの対比，特定の問題やプロジェクトに対するモデルの適応のメカニズム，モデルの進化とその支援メカニズム，モデルの自動的な解釈／実行およびそれルをベースとするプロジェクト管理，プロジェクト運営の途中での動的なモデルの変更，などがあげられています．いずれにせよ，モデルの現実への適用に関する実際の経験を参加者各自が持ち寄って，地に足のついた討論を展開しようというのが，当面の ISPW の基本的運営方針です．

そういう意味では，さまざまな工業分野において，すでに世界に冠たる製造技術・品質管理技術を持ち，そのソフトウェア領域への応用を推進しつつある（？）日本からの技術的な貢献が，かなり期待されているといえましょう．そこで，今回，第６回 ISPW の日本開催のための準備活動の出発点として，また，第５回 ISPW への参加申込みを Encourage する（Position Paper の締切りは来年３月１日）という２重の目的を持って，日本で初めてのプロセス・ワークショップを開催したいと思います．

ソフトウェア・プロセスに関心を持ち，何らかの実践あるいは研究を心掛けておられる方々の積極的な参加をお待ちします．

参加希望者は裏面の申込用紙（Identification + 簡単なアンケート形式の Position statement）に御記入の上，FAX または郵便で，実行委員の岸田孝一氏（SRA）宛お送りください．**締切りは１２月１０日（必着）**です．ただし，申込み多数の場合には，実行委員会でPosition Statement の内容を審査の上，参加者を選考させていただきますので，あらかじめ御了承ください（選考結果は，１２月２５日までに全申込者にお知らせします）．

********** **裏面に申込用紙があります** **********

送付先：　　**締切：12月 10日**

〒102　千代田区平河町 1 - 1 - 1 （株）SRA　　専務取締役　岸田孝一
TEL：03 - 234 - 2610　　Fax：03 - 237 - 7006

第1回　ソフトウェア・プロセス・ワークショップ
参加申込用紙

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

年　齢：＿＿＿＿＿＿（歳）　　性　別：□ 男　□ 女

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

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

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

TEL：[　　－　　－　　内線　　]　FAX：[　　－　　－　　]

Position Statement:

次の３つの問いのいずれかに，簡単にお答えください．紙面が足りない場合は，別紙を追加していただいてかまいません．

Q1. あなたは，ソフトウェア・プロセスの形式的モデル化や，そのモデルの現実への適用にどのような関心をお持ちですか？

Q2. あなたは，既存のいろいろなプロセス・モデルのうち，どれが最も現実的だ（あるいは興味深い）と思われますか？　それはなぜですか？

Q3. あなた自身の周囲にある現実のプロジェクトにおける開発プロセスは，どんな風にモデル化できそうですか？　また，そのプロセスにおいて，改善を要する主な問題点は何ですか？

ＳＥＡ会員状況（昭和63年11月19日現在）

正会員	1530名
賛助会員	56社

正会員の勤務地および居住地域分布

	勤務地域	居住地域
北海道＝	8	8
山形　＝	1	1
宮城　＝	4	3
岩手　＝	7	7
福島　＝	1	0
秋田　＝	1	1
新潟　＝	7	7
栃木　＝	7	7
群馬　＝	4	3
茨城　＝	10	11
埼玉　＝	20	107
千葉　＝	18	99
東京　＝	823	498
神奈川＝	120	269
山梨　＝	1	1
長野　＝	37	37
富山　＝	3	3
福井　＝	4	3
石川　＝	3	3
静岡　＝	18	18
岐阜　＝	2	8
愛知　＝	50	40
和歌山＝	2	3
三重　＝	1	3
滋賀　＝	9	10
京都　＝	18	28
大阪　＝	200	151
奈良　＝	4	11
兵庫　＝	34	61
愛媛　＝	2	2
徳島　＝	5	5
鳥取　＝	0	1
島根　＝	1	1
岡山　＝	2	2
広島　＝	4	4
福岡　＝	16	17
大分　＝	4	4
熊本　＝	32	30
宮崎　＝	1	1
佐賀　＝	1	1
鹿児島＝	6	6
沖縄　＝	1	1
韓国　＝	1	1
米国　＝	4	4
豪州　＝	1	1

＜男女分布＞

男　＝　1451
女　＝　78

＜年齢分布＞

20以下　＝　0
20_24　＝　57
25_29　＝　331
30_34　＝　408
35_39　＝　358
40_44　＝　230
45_49　＝　87
50_54　＝　24
55_59　＝　17
60以上　＝　15

＜血液型分布＞

A型　＝　591
O型　＝　447
B型　＝　331
ＡＢ型＝　161

賛助会員会社名

ジェーエムエーシステムズ，日本システム，ＳＲＡ，リコーシステム開発，辻システム計画事務所，ＰＦＵ，近畿日本ツーリスト，東電ソフトウェア，セントラル・コンピュータ・サービス，構造計画研究所，ケーシーエス，サンビルト印刷，日本システムサイエンス，富士ゼロックス情報システム，ＩＮ情報センター，富士通，富士通ビジネスシステム，エムテイシー，マイクロ・キャビン，東海クリエイト，情報システムサービス，インターナショナル・データ・リサーチ，日立ビジネス機器，アトラス情報サービス，三菱電機セミコンダクトソフトウェア，伊藤忠エレクトロニクス，三菱電機，富士通ＢＳＣ，千代田製作所，村田技研，ニコンシステム，日本ビジネスデータプロセシングセンター，阪神計算センター，昭和電工コンピュータサービス，エヌ・ティ・ティ・システム，日本情報システムサービス，ヒラタ・ソフトウェア・テクノロジー，日本エム・アイ・シー，日本電気ソフトウェア，ソニー，コマス，三菱電機コントロールソフトウェア，インターフィールド・システムズ，ジャストシステム，サンマイクロリサーチアソシエイツ，松下ソフトリサーチ

ソフトウェア技術者協会
〒102 東京都千代田区隼町2-12 藤和半蔵門コーフビル505
TEL.03-234-9455 FAX.03-234-9454