SEAMAIL

Newsletter from Software Engineers Association

Volume 4, Number 2-3 1989

目　次

「プロトタイピング特集号（その１）」の発刊にあたって		1
環境分科会例会から		
プロトタイピングの場合わけに関して（P01.1-01）	中村　昭雄	2
プロトタイピング雑感（P01.1-02）	渡邉　雄一	5
プロトタイピングの役割（P01.1-03）	塩谷　和範	7
プロトタイピングの場合わけに関して（P01.1-04）	田中　慎一郎	8
ポジション・ペーパに対する感想（P01.2-01）	田中　慎一郎	11
ポジション・ペーパに対する感想（P01.2-02）	塩谷　和範	12
第１回プロトタイピング討論 議事録（P01.*-01）	田中　慎一郎	13
プロトタイピングまとめ（P02.1-01）	中村　昭雄	14
「骨格型」の解釈に関連しての私見（P02.1-02）	渡邉　雄一	15
プロトタイピングの分類（P02.1-03）	田中　慎一郎	16
プロトタイピングへの期待（P02.1-05）	近藤　直文	18
プロトタイピング私見（P03.1-01）	近藤　直文	19
プロトタイプ／プロトタイピングに関する個人的な解釈／言葉の定義（P03.1-02）	渡邉　雄一	26
有澤氏によるプロトタイピングの分類について（P03.1-03）	塩谷　和範	28
プロトタイピングに関する言葉の定義について（P03.1-04）	田中　慎一郎	30
第３回プロトタイピング討論 議事録（P03.*-01）	田中　慎一郎	32
第３回プロトタイピング討論（９月７日）の覚書（P03.*-02）	塩谷　和範	35
プロトタイプについて（P05.1-01）	海尻　賢二	37
プロトタイピング討論会への論点の整理（P05.1-02）	渡邉　雄一	38
第５回プロトタイピング討論 議事録（P05.*-01）	渡邉　雄一	40
ＳＩＧＥＮＶ－Ｆｏｒｕｍ@ＭＯＲＩＯＫＡ顛末記	田中　慎一郎	47
九州支部設立一周年報告	平尾　一浩	52

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．４，　Ｎｏ．１　　平成元年５月１５日発行
編集人　岸田孝一
発行人　ソフトウェア技術者協会（ＳＥＡ）
〒１０２　東京都千代田区隼町２－１２　藤和半蔵門コープビル５０５
印刷所　サンビルト印刷株式会社　　〒１６２　東京都新宿区築地町８番地
特価　１，０００円　（禁転載）

「プロトタイピング特集号（その１）」の発刊にあたって

すでに予告のとおり，本号はここ１年ほど環境分科会（ＳＩＧＥＮＶ：Special Interest Group on ENVironment ）で進められてきた「プロトタイピング」に関する資料／討論レポートのうち整理のついたものをお届けします．

２００余名でキックオフしたＳＥＡの活動も，今年でついに４年に入るわけですが，このような形での特集号が発刊出来るまでになったことは，ボランティアベースの活動の１つの夢が実現したと思えてなりません．設立当初からの会員の皆さんは御存知でしょうが，ＳＩＧＥＮＶはＳＥＡ設立以来の伝統ある分科会の１つです．とくに最初の１年目は，初代世話人の方の御努力で大変魅力的な話題が毎回議論されていて，そのいくつかは（あの伝説の黄色い表紙の）ＳＥＡＭＡＩＬにも掲載されました．

しかし２年目は，初代の世話人の方が多忙のため退かれたため以前のような華々しいテーマが設定出来ず，活動が停滞しかかったこともありました．そこで３代目の世話人にスイッチしたこの１年ほどは，ともかく地道に息の長い活動することを目標とし，１つのテーマに関して（時間的な間隔があいても良いから）継続的に議論／検討をするという形で活動を続けてきました．

具体的には，それまでのように毎月，日を決めて集まるというのではなく，まず文書で各自の意見をまとめ，それを交換しあった段階で次の開催日を決めるという形式をとることにしました．また，こうした方針から「在宅会員制度」も採用し，従来のように会合に参加できなければ一切活動ができないということがないようにと考えました．こうした方針の通り，きちんと運営ができているとは言い難いところもありますが，現在，かなりの量のポジション・ペーパと毎回の議事録が手元に残っております．これらの資料は，我々がそれなりに時間と労力を割いてまとめてきたものでありますから，このまま捨て去ってしまうのはいかにも残念です．そこで，本特集号の編集方針として，そのような資料を原則として無修正で載せることとしました．それは，議論の過程で意見／立場が変わっている人も少なからずおられるのですが，折角の貴重な意見ですし，またもし意見が変わったのであっても，その過程を記録しておくことはあながち無駄ではなかろう，という判断からです．

さて，最初に問題になったのは，どのように討論を開始するかということです．現在「プロトタイピング」という言葉はあまりにも多義的に使われています．どのようなことでも同じだと思われますが，同一の用語に対して異なる意味／ニュアンスを各人が持っていては意見交換の妨げになると考えられます．「個人個人が異なる意見を持つ」ということとは別に，共通の理解を得るという意味では，何か１つ軸になるものが必要です．そこで我々は，当面，山梨大学の有澤誠先生の書かれた「ソフトウェアプロトタイピング（ソフトウェア工学ライブラリ１６，近代科学社，１９８６年７月）」という本を出発点にすることにいたしました．我々の内部で共通の基盤ができるまでは，なるべくこの本にかかれている内容を通して討論することにしたわけです．具体的な出発点としては，「プロトタイピング」のイメージ合わせをするという意味で，「プロトタイピングの場合わけ」をとり上げました．本でいうと，

2.1 試作品としてのプロトタイプ，

2.2 試用品としてのプロトタイプ，

2.3使い捨てプロトタイプと骨格プロトタイプ，

が中心となりました．

プロトタイピングは，言葉としては恰好良いのですが，議論してみると現場でのどろどろとした問題の葛藤から出てきた考え方のような気もしてなりません．そのような意味では，議論すべき点は数多く残されていますし，併せて実践の余地も多分にありますので，幅広い分野の技術者に共通のテーマと言えるのでないでしょうか？

おわび

今回の SEAMAIL は，環境分科会で乗っ取ろうとしていたので，構成が少し特殊です．そこに九州支部から原稿が届いたので，ホットなうちにと思い，この号に入れさせていただきましたが，構成の関係上，付け足しのようになってしまって申し訳ありません．関係者の皆様にお詫びいたします．

それでは手作りのＳＥＡＭＡＩＬを，ごゆっくりとお楽しみ下さい．

プロトタイピングの場合わけに関して

中村　昭雄

プロトタイピングはどんなものか，言葉は知っていても分類についてとか，場合分けについてとか，理解はしていませんでしたので，有澤先生の本を読んでまとめてみました．［別紙１，別紙２を参照］特に別紙１は完全に出来ませんでした．（切り方が悪いのか？）

現実に企業の情報処理部門がプロトタイピングを考えて実行するにはどうか考えてみました．

１）　システム開発要求仕様書の詰めの甘さが結果としてプロトタイプになる．
プロトタイピングを意識して作ってはいません，最終製品を作っているつもりです．
ところが出来上がったソフトウェアはカットオーバーの前に大幅な変更を受ける，結果として試用品のプロトタイプとなるし，骨格型であるといえます．

２）　このプロトタイプというか製品というか，ソフトウェアは使われれば何もありませんが，使えなければ結局，闇に葬られ，使い捨てのプロトタイプとなるしかないのです．
（日経コンピュータの動かないコンピュータのモデルにされないうちに素早く処分する）

別にこの様なことは，意識してプロトタイプを作っているわけではなく，最終製品を作っているつもりですが，ウォータフォール・モデルでは宿命とも言える上流工程のまずさが，そのまま下流工程に垂れ流されて来た結果なのです．

実際にプロトタイピングを意識して作るときが全然無いわけではありません．それを場合分けしてみると，

・試作型プロトタイプとして

目的は　新しい処理方法の確認
ＤＢＭＳの新しいＤＭＬのテスト
４ＧＬ導入の評価

今までに使った事の無い命令を組合わせてＤＢを更新する．メーカーを信用していないわけではないのですが，ＯＳのバージョンアップにより新しい命令が使えます，と言われて使ったらシステムクラッシュした，と言う例は多々有ります．これでは安心して全国オンライン等は出来ません．無駄な様でも確認する必要が有ります．この様な時に簡単なプログラムを作成します．

・試用型プロトタイプ

上記の様な技術的な意味でのプロトタイプを除けは，例えばオンラインの入力をチェックしながら操作性の確認をユーザと行い，画面のレイアウトを修正したり，他のオンラインプログラムとの接続をチェックしたりして作成する時は試用品であり，骨格型であると言えます．

以上の様に場合わけをしてみましたが，試用品は何等かのプラスをして実際のシステムに生かしていく（＝骨格型である）と思うし，前述のオンラインの例の様に修正したのだけれど結局修正出来なくて最後は捨ててしまう事もあると思います．

試作品でも，スケルトンは別のプログラムに使用していく（企業の場合試作でも設計時に転用を意識する事が多い）こういう事を考えると，試作，試用，使い捨て，骨格とかに分けるのは良いけれど，分けてどうするのか疑問です．そのような意味では学者の世界かなという気がします．

分ける目的，メリットは何か，私には不明確です．

現在のシステム開発を見ると使い分けは本当に存在し，又，使い分けられるかチョット疑問を感じます．

分けることを考えるよりも，この本で言われている様なシステム開発をすると，自然にプロトタイピングが行われる様な仕組みを考えたほうが良いと思います．

（なかむら　あきお：山武ハネウエル）
８８－６－２０

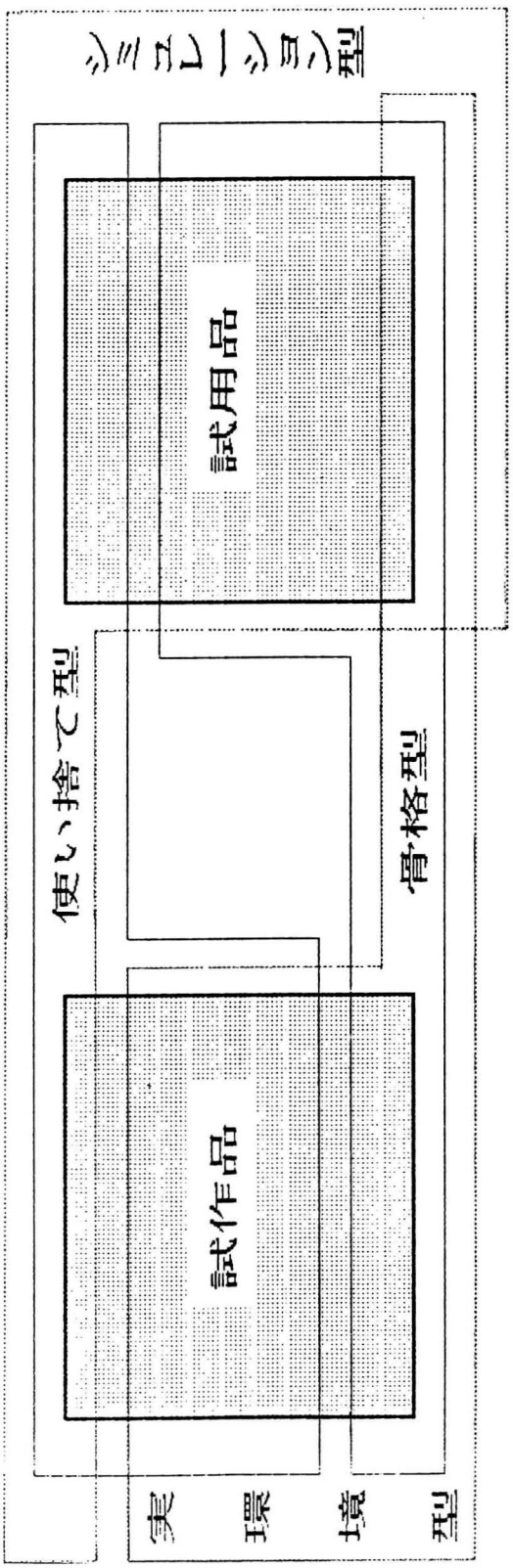
［別紙１］

プロトタイピングまとめ

種　類	使用目的（使用場面）	関連する種類	狙いは			使用するのはいつか	言語、開発手法	開発環境
			実現の可能性	利用者要求の確認	思考錯誤か			
試作型	・未経験の要素を含むとき ・複数の可能性のうちからひとつを選択する ・作り方の摸索	使い捨て型 実環境型	◎	○	◎		早く用意に作れるならどんな言語でも可能 BASIC,APL,AWK	最終システム本体の開発環境で行う プロトタイプを試作しながら最終システムの環境を整備していく
試用型	・要求の妥当性の検証 ・問題点の発掘と解決策を得るために共通の認識を物場の設定 ・機能及びマンマシンインターフェースのデモンストレーション	骨格型	○	◎	○	開発過程のかなり早い段階（設計段階）	最終システムを記述する言語と同一の言語(OS,ハードウエアも) ISD 法、決定表	マンマシンインターフェース や通信OS等 ユーザー要求を入出力機器で制約しない様なハード やOS
使い捨て型	・早く安くプロトタイプを作る ・早い時期に安いコスト で作成しユーザーの反応を見る	試作型 試用型	◎	○	◎	設計の骨格自体に自由度が残されている状態	早く用意に作れるならどんな言語でも可能 BASIC,APL,AWK	
骨格型	・設計の段階でのユーザーの参加を可能とし詳細をつめる	試作型 試用型	○	◎	○	システムの骨格が固まった状態	最終システムを記述する言語と同一の言語(OS,ハードウエアも) ISD 法	形式的な仕様から実行可能なコード に変換し最終製品を作りだせる様な物 プログラム 自動作成、再利用等
シミュレーション型	・ユーザーの試用の際に統計データ をとる事が可能 ・性能面のチェックが可能 ・独立なプロセスが相互に干渉し協調してデータ や情報とやりとりするモデル	試用型	◎	◎	○		Smalltalk-80 ESDL GPSS	
実環境型		試作型	◎	○	○			最終システム本体の開発環境で行う

［別紙２］

プロトタイピングの関係



プロトタイピング雑感

渡邉　雄一

プロトタイピングについて，過去に１年ばかり富士通のユーザ会の研究会［２］に参加しながら検討を加えていく過程で，個人的には以下のようなたわいもないことを考えるようになった．『プロトタイピングを支援する環境は何故SEを対象しているのであろうか？．食堂でメニューを見て，食べたい物を注文するように，我々が実現出来るソフトウェアをあらかじめ用意しておき，利用者がそれらを選んで「プロトタイピング・ツール」に依頼すると，なかば自動的にソフトウェアが生成され，それにユーザ自身が「プロトタイピング・ツール」を通して改良を加えてゆくことで最終的なシステムが完成されないものかと．．．』

プロトタイプとは言葉通り解釈すれば「雛形」である．その雛形を用いて利用者との意識のづれ（例えば要求仕様の設定記述の間違い，要求仕様内の矛盾の摘出，要求仕様には記述されていなかった事項など）などを発見し，取り敢えずまたそのプロトタイプを修正することにより最終的な対象ソフトウェアを得ようとするアプローチである．特にここで重要と考えることは，

１）プロトタイプの変更・変形は，「意識的に意図的にそしてある種の計画性をもって」行われなければならない．

２）変更を加えることを通じて（時には元の要求仕様に修正を加えることをも恐れずに）要求仕様と実現される対象ソフトウェアとのズレを最小限にするべくという作業過程を体系的に整理することにより，手法としての確固たる地位を得ようとしている．

ということである．個人的な興味は，さらにこれらを支援するソフトウェア環境にあることは言うまでもない．特に要求仕様に対する変更を如何に効率的に正確に記述するのかという「仕様の差分記述」（こんなことほんとに出来るのだろうか？）と，「概念モデル」等の新しいMMIを組み合わて始めて実現されるアプローチに関心がある．

さて一種野蛮なその場しのぎなやり方を，実際の対象システムの「雛形」として正統化し用いることで，一応のステイタスを確立しつつあるプロトタイピングだが，冷静になって考えてみると，幾つもの問題があることが分かる．まずプロトタイプなるものは，如何なる物なのか？，そして如何に作られるものなのか？．そしてどのような形で修正・変更を加えれば良いのか？（その背景に思想や方法論は存在するのだろうか？）．そして変更されてきたものは，いつプロトタイピングの過程を抜け出したと言えば良いのだろうか？，また過程での費用等の見積もりはどうすべきか？

個人的には，プロトタイプは対象ソフトウェアの範疇（個々の業務）に依存した内容を含まねばならないため，機能としてプロトタイプの持つ特性を論ずることは出来ても，その実現方法を特定することは極めて困難であると考える．そのようなことからも，対象分野に依存しないプロトタイプの共通的な特性を洗い出すとともに，その為のソフトウェア環境（プロトタイプの開発と本システムへの移行環境）の実現への用件を，具体的な開発事例を調査したり紹介しあうことでまとめられればと考えている．その為には，［３］のような先進的な研究成果・論文の詳細な解説を勉強することも重要だが，実践的にもう少し皆で共通に認識出来る事例を中心に，地道に議論検討を加えていくことを提案したい．特にプロトタイピング手法を用いて“失敗した”事例の発表がないことは，方法論・手法として体系化していない現在，かなりの属人性をもっていることの裏返しである．ならば今後プロトタイピング手法がむやみに使われだしてからでは遅すぎやしないか？．それまでに心ある人々の参加を求めて，少なくとも言葉のニュアンス位は統一しておかなければ危険極まりない．実際，今回の議論の中心となる「使い捨て」「生き残り」とかについては通常生活のなかで用いられているニュアンスでも通るが，「試作品」「試用品」「骨格」といった用語は一部の人には通用するが誤用誤解のもとになり，ひいてはプロトタイピングそのものの誤解にもつながりかねない．

またプロトタイピングを行う際のポイントが幾つか挙げられようが，その中で一番私が気になることは「はじめからどのようなプロトタイピングを進める積もりだったのか？」ということがプロトタイピングの事例に何も述べられていないからである．例えば「或る目的の為にプロトタイプを作りまして，何々の問題があったため修正を行いまして．．．」といった資料を見ても，結果としてそのなったかもしれないけれど，何故そのように進めようとしたのかいった決断（意志決定）したことが明確にされていない．このような重要な作業プロセスの分岐点で，さほど理由もなく，ただ経験者の個人的な判断によるのならば，それはプロトタイピング

の成功として認めるわけにはいかない．プロトタイピングが「手法」として定着する（させる）ためには何らかのソフトウェア工学で検討された「仕掛け」「裏付け」が欲しいと思う．

その為には，現在挙げられているプロトタイピングの「分類」についてディスカッションして言葉の意味を我々なりにハッキリ（出来れば定義）させたい．私はいろいろなパターンのプロトタイピングの進め方があると思う．但し，業種なりソフトウェア作成工程の部分でなり，それぞれ向き不向きはあると思う．それを最低限でも整理してみたい．

気になる話を１つ．去年の夏のワークショップ（若手の会）に参加していろいろな話をしていたおり，佐原伸氏（SRA）が「証券などの大規模システムはみんな最初から大掛かりなプロトタイピングをやっている．あんな巨大なものを，最初からちゃんと設計出来る人などいない．」という趣旨のことを言われた．そして先日のソフトウェアシンポジウムのパネルディスカッションで「このあいだの兜町の証券取引所のシステムのトラブルはアルゴリズムそのものに問題があるのではないか？」という（具体的な予想）指摘をされていた．もしそうだとしたら，プロトタイピングの進め方に問題があるのではないかと思う．プロトタイピングは概ね，対象システムの幹になる部分や，主要なサブセットを取り出してきてプロトタイプを作り評価するので，データ量は少なく，例外処理などもことごとくハショルのが相場である．つまり私が言いたいのは，プロトタイピングをすることで対象システムの仕様が明確になるという利点がある反面，下手に骨格型のプロトタイピングを採用して，その時はなんとか耐えられるシステムを作っても，その問題点（レスポンス，大容量への配慮）が設計の段階でおろそかにされたり検証段階で見逃されたりするのは，かえってマイナスであろうと思う．

また良く言われることだが，プロトタイピングを進めていくとドキュメントをおろそかにしがちであるということがある．たしかに不要なドキュメントはいらないが，上にも言ったように「なぜ不要と判断したのか」が記録として残っていないと困る場合も今後あろうかと思う．但しそれらが全てプロトタイピング手法の責任というものではないが．．．．

言い訳ですみませんが，以上思いつくままに書きました．あとの議論はSIGENVの集まりの席で楽しくワイワイとやりましょう．

【参考文献】

［１］有澤誠：ソフトウェアプロトタイピング，近代科学社，１９８６
［２］「プロトタイピングによるシステム開発」，富士通LS研６１年度報告書，１９８７．６
［３］伊藤ほか：プロトタイピングツール，啓学出版，１９８７．９
［４］花田収悦：プロトタイピング技法を実践的に解説した書，日経エレクトロニクス，１９８８．５．３０（No．４４８），pp．２３３～２３４

（わたなべ　ゆういち：電力計算センター）
８８－６－２０

プロトタイピングの役割

塩谷　和範

1）基本的には“総てのソフト開発にプロトタイピングは有効である”
プロトタイピングは全体に適用するのが一番効果的である．が，部分的なプロトタイピングもその他の部分との関係を充分考慮すれば，有効である．しかし，これを総合的に支援する環境においてもっとも効果を発揮する．

2）構造的手法による問題点
従来の構造的手法による開発において次のような問題点が発生していた．

- 最初から総ての要求を定義するのは不可能であり，
- 要求定義が完全でなければ作成するソフトは不完全とならざるを得ず，
- 従って仕様変更・変更要求が発生する．

これを解決するには，

1. 明確な要求から定義し適宜作成していく
2. 不明な点に着目してこれを先に解決していく
3. 不明な点を含めて動くプロトタイプを作る

などの方法が考えられる．
ここで 1.は後になってから問題が発生し手戻りが生じる可能性が高く， 2.は総ての不明な点を洗いだすのが難しく，しかも総てを明らかにするのは難しい．

3）プロトタイピング
その点 3.はユーザ要求を要求定義者の解釈でプロトタイプを作成し，これを動かして見せる事によってユーザとの間の解釈の違い・要求の漏れ・矛盾などが明確になる．かつ少なくともユーザインタフェース部分についてはユーザの承認を得ることができる．
しかし次のような問題点がある．

1. 従来の要求定義より時間と工数がかかる．
2. プロトタイプがうまくできても目的システムが成功する保証はない．
3. プロトタイピングのための支援ツール・環境が存在しないか使えない．
4. 既存システムの修正・改造などには適用しにくい．

1.については，プロトタイピングに工数がかかっても手戻りがなければ総合的には効果がある．
2.は確かにそうだが同じことが従来の手法にも言え，従来の手法より要求が明確でありかつユーザの承認を得ている分だけ失敗の可能性が少ない．
3.確かに現在までのところ総ての場合に有効なツールは存在しないが，部分的にプロトタイピングすることは可能であり効果を挙げることができる．
4.新規システム作成時には総てをプロトタイプできるため有効だが，既存システムの改造・サブシステムの追加などの場合，既存部と新規部をうまく分離できるケースは少ないため既存部分の扱いが問題となる．
3.の部分的プロトタイピングの場合も含めて，対象システム全体を考慮する視点（方法論）が必要となる．

4）検討課題
従ってこれらの問題点を踏まえて，

(1) 対象システム
（制御系/アプリケーション・データベース/シミュレーション・図形/文書処理・会話処理/バッチ処理など）
(2) 適用条件
（全体/部分・新規/改造）

を検討し各々の場合に有効な方法論とこれを支援するプロトタイピング環境を構築する必要がある．

（しおや　かずのり：ＳＲＡ）
８８－６－２０

プロトタイピングの場合わけに関して

田中　慎一郎

１．　はじめに

有澤氏はプロトタイプの場合わけを次の視点から行なっている．

- 試作品／試用品
- 使い捨て型／骨格型
- シミュレーション型／実環境型

これらの分類は，作成されたプロトタイプ，あるいはプロトタイピングのために環境を考える場合にはある程度の意味をもつと思われるが，プロトタイピングそのものを考える場合にはさほど正しい視点とは思われないふしがある．これはプロトタイピングを考える際に最も重要なことは，その目的を考えることであり，手法，あるいはその後の利用云々はあくまでも二次的なものと思われるためである．

以下ではまずこの分類に対して感じる不自然さを提示した上で，J.R. Johnson 氏のレベルわけを参考にした，「目的から考えたレベルわけ」に関する私見を述べる．

２．　有澤氏の分類に関する言い掛かり

有澤氏の分類は，「プロトタイピングとは何か」を理解するための分類であり，プロトタイピングそのものを分類したものではないこと，および上記の目的としては非常に適切であると思われると言うことを先に述べた上で，半ば言い掛かり的にこれを否定してみたいと思う．

２．１　試作品／試用品の分類

まずこれはプロトタイプの「使用目的」から考えると，開発者向けプロトタイプ／利用者向けプロトタイプのようになると思われる．つまり，氏の言われることも良く理解できるのではあるが，「試用品」であってもそれがプロトタイプである限り「試作品」であることには間違いがなく，「試作品」であってもそれが「使われない」ことはあり得ないためである．

例えば，単に「試用」と言う意味であれば，完成品を宣伝やデモのために（すなわち「利用者」のために）使う／使わせることはいくらでもある．また逆に「試作」されたと言うことは，何らかのデータをとるなど，「開発者」が利用するために作られたと言うことを意味しているはずである．仮に「一度書いてみただけ」で，捨てられてしまうものだとしても，氏も言われているように，「それを書こうとして考えたことが新しいプログラムを書くときに生かされるから」，「開発者」にとっては立派に役に立っている（使われている）ことになる．

つまりこれは，

- 開発者要求を満たすために試作する．
- 利用者要求を満たすために試作する．

と言う分類の方が妥当であるように感じられるのである．

２．２　使い捨て型／骨格型の分類

ついでこれはプロトタイプの「作成時期」から考えると，骨格ができている段階でのプロトタイプ／まだできていない段階でのプロトタイプのようになると思われる．つまり，「プロトタイプ」という言葉の指し示す範囲に関して，広義と狭義との間で混同が起きているように思われるのである．

まず「骨格型」のプロトタイプの場合，それが完成品でない以上，それが全体として「広義」でのプロトタイプであることには間違いがない．しかし「プロトタイプ」は開発システムのごく一部を対象とするものであっても構わないわけで，この考え方で言うと，捨てられることのない「骨格」はすでに「確定された部分」であり，「捨てられる可能性のある部分」が「狭義」でのプロトタイプということになる．

もし「骨格」と「肉」とは不可分であり，「肉」だけ削ぎ落すことができないとするならば，それは単に設計が良くないだけであり，プロトタイピングの価値がない以前に，プロトタイピングですらないと思う．従って，「捨てられない」のは「骨格」の方であり，プロトタイピング対象部分はいずれにせよ「捨てられる」のである．

反対に「使い捨て型」の方に関していえば，「骨格」がまだないので最低限必要な部分を「狭義」での「プ

ロトタイプ」と合わせて作っているだけで，もしこれが「プロトタイピング支援環境」のようなものでサポートされるとしたら，「不要」な部分である．

つまりこれも，

- まだ骨格ができていない段階での試作．
- 骨格ができている段階での試作．

という分類の方が妥当であるように感じられる．

ところでこれは当然のように時期や手順の問題になり，開発のごく初期の段階から骨格ができていることはほとんど無いであろうし，内部処理の検討と並行して帳票や画面などのユーザ・インタフェイスを確認しようとする場合なども当然のように骨格はできていない．逆に内部処理の細部を試作する場合などは，すでに骨格は決定していることが多い．このあたりの混同はおきていないのであろうか．

２．３　シミュレーション／実環境の分類

さらに，記述言語ないしはその開発環境に関してであるが，これ自体はプロトタイピングそのものとは本質的には関係が無い問題である．どちらが好ましいかというようなことであれば，それなりに議論の対象にもなるかもしれないが，分類としてはどうかと思う．これは例えば何らかのシステムを作るときに，ＤＢを使うか使わないかで一般的にそれを分類しようとしているようなものである．

３．　レベルによる分類

以上のような点から，氏の分類は過去に作成されたプロトタイプ，あるいは行なわれたプロトタイピングの事例を分類し，プロトタイピングをまず理解するためにはかなり有効であるが，一般的にプロトタイピングを論じるときの視点としてはやや好ましくないとして，次のような J.R. Johnson 氏［２］の分類をアレンジしたものを提案したい．

レベル１：　実物大模型
レベル２：　模擬実行
レベル３：　動く模型
レベル４：　研究開発型

３．１　実物大模型

「スクリーン・レポート・フォーム等をサンプル・ファイル上に作成し，画面，プリンタ等に出力することにより，実物と同じイメージでユーザに評価させる．」

手書きサンプルまで含めれば，プロトタイピングなどという言葉が登場する以前から行なわれていたことの延長ではある．但し当時は「早い段階から」などといった方法論が欠如していたと予想される．その意味であまり「プロトタイピング」らしくない．全体的な裏付けとなる方法（論）が必要となる．

これは時期的にも，目的的にも，明らかに「使い捨て型」と言える．早期に，他の設計／開発等と並行して行なわれる．

３．２　模擬実行

「主にスクリーン・サンプルに対し，処理過程に沿ったスクリーンの流れをシミュレートする．個々のスクリーンだけでなく，業務の流れに応じた評価が可能となる．ファイルやデータベースのアクセスは実現しない．」

一般に持たれているプロトタイピングの一つのイメージであろう．これも明らかに早い段階から，ユーザ要求を確認するために行なわれるもので，従来のウォータフォール的な開発プロセスに取り込む場合もあるが，内部処理の設計／開発と並行して行なうような新しいプロセスとして適用する可能性もある．

基本的に，「試用品」，「使い捨て型」，「シミュレーション型」と考えられる．

３．３　動く模型

「開発するシステムの一部または全部の範囲に対し，ファイルやデータベースへのアクセスおよび処理もある程度含めて作成する．内部処理に近いレベルの評価が可能となる．」

一般に持たれているもう一つのイメージであろう．これは実際に近いかたちで作成／試用を行なうもので，実際に近い分だけその用途は広い．

基本的には「試用／骨格型」，「試作／骨格型」，場合によっては「試作／使い捨て型」にも使用される．実環境における何らかの状況が問題になる場合に行なわれることが多く，作成段階は，内部処理がある程度確

定した段階となる．最初から「全体を捨てる」ことを予想しているケースは少ない．但し，「試作／使い捨て型」の場合は，早期に行なわれることも予想されるが，実際にはこれはレベル４に入れるべきであると思われる．

３．４　研究開発型

「全く新しい種類のシステムの開発といった場合や，ユーザ要求の変更が多く予想される場合などに，研究開発のような試行錯誤的プロセスを経ながら，徐々に評価・改良をすすめる．これはむしろ，逐次改良型開発というべきかもしれない．」

一番広い意味でのプロトタイピングであろう．１～３のすべてのレベルのプロトタイプが作られる可能性がある．逆にいえば，全く新規なものの開発においては，従来からもこれに近いことは行なわれていたはずである．

早い段階において，「試作／使い捨て型」，「試用／使い捨て型」が多用されることが予想される．また場合によっては，開発システムとは直接の関係がないが，設計方針などを確定する際の基礎データをとるための「実験装置」を開発することもあり得る．

「実験装置」（あるいは観測装置）を作成することは，広い意味での「プロトタイピング」だとは考えられるが，これ自体は「プロトタイプ」（原型）とは言い難い．何らかの名前付けを行ないたいものである．

４．　まとめ

かなり大まかなものではあるが，プロトタイプの作成目的に応じて，上のようなレベルわけをすることは可能であろう．これはまだまだ借り物でよく考えられていない上，考えが及んでいないところも多々あるかと思われる．しかしながら用語や概念の統一を行なうことは非常に重要なことであるから，こうしたものに対して，私たちなりの分類と定義と名前付けをまず第一に行なっていきたいと思う．

参考文献

[１]有澤誠：ソフトウェアプロトタイピング，ソフトウェア工学ライブラリ１６，近代科学社，１９８６年７月

[２]ジェームス・Ｒ・ジョンソン：ソフト開発期間を大幅に短縮するプロトタイプ手法，日経コンピュータ，１９８４年２月６日

（たなか　しんいちろう：ＳＲＡ）
８８－６－２０

ポジション・ペーパに対する感想

田中　慎一郎

(1) 渡邉さんのペーパに関して

「ユーザ・プログラミング」のための環境が必要かどうかはよくわからない．「一般ユーザにも使えるような」環境という意味では大賛成ではあるが，現状では難しいと思うし，いかに環境として優れたものができようとも，それなりの技術なり考え方なりを身につけていないものが使用するのであれば，ますます「粗悪プログラム」の大量生産に拍車を書けることにもなりかねない．したがって「最終システムまで作れる環境」ではなく，「ユーザの求めるユーザ・インタフェイス部分を容易にシミュレートできる環境」というか，「ユーザ・デザイン」のための（が可能な）環境でよいのではないか．

これは「プロトタイプ作成＝機能（外部仕様）設計」という考え方になるかと思うが，このあと専門家が実現上の問題点等を整理／分析し，プログラム構成を決定するなどの「内部仕様設計」を行ない，実際に作成していくといいのではないか．

ただし，この時作成する資料を現在のようにそのまま残すためのものとして書くのでは意味がない．システム完成後（確定後か？）に機能仕様書（あるいはユーザーズ・マニュアル）と内部仕様書をメインテナンス用に作成すべきであると思う．現在ドキュメント書きはプログラマ（ＳＥ）がやっていることがほとんどだが，それでは必要なことは大体記述されない．ある意味ではこの部分は切り離して別作業として，専門家を養成する方向に向かうべきなのかも知れない（別な意味でのアナリストになるか？？）．

ほとんど思いつきでいってしまっているが，いずれにせよきちんと方法（論）を作り上げなければ，プロトタイピングは「単なる失敗プロジェクトのごまかし方」に過ぎないものとなってしまう．

同時に「見積りの方法」ということも重要になる．しかしこれも計画性があれば，プロトタイピング固有の問題ではなくなると思われる．

(2) 中村さんのペーパに関して

確かに分類に関していえば，有澤先生の分類は一つの視点で行なわれたものではないようなので，きれいに切り分けるのは無理なようである．もう少し重複部分の少ない形で分類したい．

現状でのプロトタイピングに関していえば，プロトタイプを作るだけではプロトタイピングではないことを明確にしなければいけないように思う．プロトタイピングはプロトタイプを作成し，それをもとにして行なわれる開発方法だとは考えられるが，プロトタイプが作られれば全てプロトタイピングになるかといわれると，決してそうではないといいたい．

とにかく作られたプロトタイプを分類しても意味がないように思われるので，意味のある分類／定義をしていきたい．

(3) 塩谷さんのペーパに関して

確かにプロトタイピングはその裏付けとなる有効な方法（論）を伴ってはじめて，意味のある手法になると思う．全体で議論していきたい．

(4) まとめとして

要するに，用語の統一／定義，特性から見た分類，それに対応する方法論（方針，計画），などについてやっていきたいと思うが，ご意見をいただきたい．

（たなか　しんいちろう：ＳＲＡ）
８８－６－２９

ポジション・ペーパに対する感想

塩谷　和範

私見ですが，プロトタイピングというのは机上の議論ではダメで実験・経験が必要だと思います．ですから，私としては具体的に何かをしてみることが重要だと思います．

しかしメンバだけの力ではプロトタイピングを実践することは難しいでしょうから（これが一番重要なのですが）全員で一つの方法を選んでせめて一つ一つ具体的に検証・思考実験していくことが必要だと考えています．

また手法とツール（特に現状の）は分けて考えなければツールの機能に制約されて本来の手法の意味を見失ってしまいます．逆にある手法を考えたときにはどんな支援ツール・環境があればこの手法を活かせるかを考える必要が有ります．

私のセミナーの時にも触れましたがプロトタイピングを成功させるためには既存部分に対する考慮が必須です．これは大きな問題でまだどのシステム（ツール）も取り組んでいません．

現実の問題の複雑さ・汚さを考えると元々この部分はシステム側(プログラム)では少なくとも当分の間は扱えない部分だと思います．そうは言ってもほっておけない問題ですのでシステム的な対処が必要です．例えばプロトタイプ出来ない部分の特性の記述をプロトタイプに書けるようにするとか，その取り扱いについて規定しておくことなど考えられます．

この問題はみんなで討議・検討したいところです．

（しおや　かずのり：ＳＲＡ）
８８－６－２９

第１回プロトタイピング討論 議事録

日　時：　昭和６３年６月３０日　１８：３０～
場　所：　ＳＲＡ　ミーティング・ルーム
出席者：　青島　茂（ＦＸＩＳ）　塩谷　和範（ＳＲＡ）　田中　慎一郎（ＳＲＡ）

人数も少なかったので，フリー・ディスカッションの形式で行なった．

１．　有澤先生の分類に関して

「試作品型／試用品型，実環境型／シミュレーション型の２つの分類に関しては，一応独立の視点を持ち，その内部を２つに分けていると感じられるが，使い捨て型と骨格型という分類が同じ視点からのものとは思われない」という疑問を田中が提示した（その理由は P01.1-04 参照）．

これに対し，「骨格」というのはスケルトンのようなものの意味ではなく，nucleus （核とか心の意）の訳であるから，もとになる具体的な何らかのもののこと．したがってこれは「プロトタイピングの具体的成果物（この場合はソース・コード）を一切使わないか，それとも流用していくか」という視点であるという解釈を塩谷が説明した．

田中は「骨格」というのはスケルトンのようなものとして解釈していたので，この説明で一応納得した．しかし今後の方針を議論している際，有澤先生の本では（ p44 ），形式的な仕様記述から実行可能なコードを生成するようなことも含んでいることが指摘され，全員に新たな疑問が発生した．→次回に見送り．

２．　今後の方針に関して

「有澤先生の分類を出発点にすることに間違いはないが，この分類は３視点×２という大雑把なものであり，お互いの分類の間で重複部分が大き過ぎて議論に乗りにくいので，何らかの違う分類をしたい」ということを田中が提案した．

目的から入っていくべきか，方法からはいっていくべきかに関して種々の意見が出たが，うまくまとまらない．

そこで，「目的と方法を考えなければいけないのだから，これでマトリックスをつくって考えていくのもいいかもしてない」という提案が青島からなされた．

しかし，「プロトタイプの目的は１つではないだろうし，対応する方法も１つではないだろうから，実質上無理ではないか」という疑問が塩谷から提示される．

在宅会員の中村さんが作成した図を見ながらこれが利用できないかを考える．

・一度２×２×２＝８通りのパターンで検討してみてもいいのではないか．
　現実にあり得るか，狙い，時期，言語等を考えてみる．→次回ペーパ？

３．　まとめ

“適当にやるのがプロトタイピングでは悲しい．”

今後は，「プロトタイピングの方法論を何とかする」ということを目標とする．

この際特にポイントとしたいのは，「現実的なプロトタイピング・アプローチを考える」ということである．

例えば，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．整合性の検証はどのようにやるべきなのか，またプロトタイプ側でやるべきか，それとも外側でやるべきか等々，いろいろ考えられる．これらのことを片目で睨みながら討論を進めていきたい．

一方当面の目標としては，有澤先生の分類から単純に８パターンを生成し，これをたたき台としてプロトタイピングについて見識を深めていくことにする．これについては参照用の表を別途用意するので，これに基づいて次回打ち合わせ前にまた文書により意見を交換したい．

４．　その他

現在存在するプロトタイピング支援環境等に関して，可能なものについてはデモを見せていただけるように設定したい．手始めとして，PWB (Prototyper's Workbench) をお願いしたい．

（記録：田中　慎一郎）

プロトタイピングまとめ　　中村　昭雄

分類			狙い			具体的場面等	適用時期	言語／手法／環境等
			実現の可能性	利用者要求の確認	思考錯誤			
試作型	使い捨て型	シミュレーション型	○	○	◎	新しいＤＢＭＳの確認	システム開発 要求定義 ＋基本設計	４ＧＬ　ＰＬ／１
		実環境型	◎	○	○	モジュールの動作確認 ４ＧＬの導入評価	システム開発 要求定義 ＋基本設計	ＣＯＢＯＬ
	骨格型	シミュレーション型	○	○	◎	新しいサポートツールの確認	基本設計 ＋詳細設計	ＣＯＢＯＬ
		実環境型	◎	○	○	オンライン処理の処理スピードの確認	基本設計 ＋詳細設計	ＣＯＢＯＬ
試用型	使い捨て型	シミュレーション型	○	◎	○	あまりない？	システム開発 詳細設計 （プログラム設計）	４ＧＬ ＰＬ／１
		実環境型	○	◎		あまりない？	システム開発 詳細設計 （プログラム設計）	ＣＯＢＯＬ
	骨格型	シミュレーション型	○	◎	○	オンライン操作性の確認	詳細設計 ＋プログラム作成	ＣＯＢＯＬ
		実環境型	○	◎		画面，帳票のレイアウトおよび出力項目の確認	詳細設計 ＋プログラム設計	ＣＯＢＯＬ

手法，環境は特にプロトタイプだから別とはしていません

「骨格型」の解釈に関連しての私見

渡邉　雄一

前回の会合に出席できず申し訳ありませでした．小人数ながらかなり盛り上がったようですねェ．さてまた独断と偏見で雑文を書きます．

1）前回のポジションペーパに関連して

－ 1）中村さんのポジションペーパ（P01.1-01）に関して

特に後半で出されていた疑問点「分ける目的，メリットは何か，私には不明確です」ということに関して，私は効果の違いのように思います．例えば中村さんがその前で言われているように「試用品は何等かのプラスをして実際のシステムに生かしていく＝骨格型であると思うし・・・」というのが出来れば理想だと思うのです．しかしそれは開発規模にも左右されるでしょうし，予算などとの兼ね合いからも必ずしも理想のプロトタイピング環境で物事を進めえるとは限らないでしょう．そのような場合，例えば「ラピッド」プロトタイピングを狙うのか，「スケルトン」プロトタイピングを狙うのかは重要な選択と言えると思います．そして（個人的には田中さんなどとも話していることですが）同じ用語を使っていながら，この『プロトタイピング』という言葉ほど，そのニュアンスが幅広く，悪く言えば好き勝手に使われている（実際各種の論文・報告書ではそうです！）言葉もないような気がします．だから敢えて不適切な分類を踏襲していく必要はさらさらないと思いますが，用語の解釈は統一したいと思うのです．そのような意味で「理解するための分類」という田中さんの意見（P01.1-04）が妥当と思います．

－ 2）私のポジションペーパ（P01.1-02）に関して

田中さんから指摘のあったように（P01.2-01）当面は「ユーザインタフェース部分をシミュレートできる環境」である意味で十分だと思います．ただここで意外と難しいのは，通常（特に汎用機やオフコンの世界では）ハードウェアの制限が厳しいことから，例えばパソコンのようなプロトタイプ作成支援環境を作りやすい物であっても，そこでの自由度（色や罫線など）をターゲットにあわせて制限する必要が生じてくるということです．以前に大型汎用機のユーザ会で意見交換をしたときも，「生き残り型のプロトタイピングを検討するなら最終段階は，実稼働機と相当のものでないと仕様の再変更が必ずあるだろう」という意見が多く出されました．

蛇足ですが，先日SRAのセミナーで拝聴させていただきましたPWB（Prototyper's Work Bench）をもし大型汎用機での事務アプリケーション開発に適用するのなら，ここらへんの人間の感覚の誤差が問題になるのかな？という感じをいだきました．ただそれは逆に言えば極めて容易な解決方法をもっているともいえます．なぜなら，ジョンソン［１］の主張する場合分けのペインティングレベルで十分と考えられるので，PWBの世界のターゲットシステムのエミュレーション機能を取り込むことさえ出来ればかなりの効果があがるように思うからです．

2）前回（第１回プロトタイピング討論会）への私見　－「骨格型」の解釈－

たしかに塩谷さんの言われる解釈がこの場ではあたっているように思う．しかし田中さんの言われる指摘も良くわかる．骨格という言葉に期待するものがイメージを広げ発想を豊かにしてくれることと，現実からの逃避とは別にしなければならない．

【参考文献】

［１］ジェームス．R．ジョンソン：ソフト開発期間を大幅に短縮するプロトタイプ手法，日経コンピュータ，（１９８４．２．６）

（わたなべ　ゆういち：電力計算センター）

プロトタイピングの分類（その１）　　田中　慎一郎

分類			狙い			具体的場面等	適用時期	言語／手法／環境等
			実現の可能性	利用者要求の確認	思考錯誤			
試作型	使い捨て型	シミュレーション型			○	ユーザと開発者が同一か，もしくはユーザ要求を開発者が非常によく理解している場合.	初期. 要求の細部が確定していない段階.	扱い易いものであれば何でも可.
		実環境型	○		△	同上.	初期. 具体的実現方法が確定していない段階.	本開発と同様に，なるべくきちんと作ることが大切．使い捨てだからといって，あまり適当に作るのは好ましくない.
	骨格型	シミュレーション型		○		全体の制御部分を簡易言語等で作成しておき，注目部分から順次作成／確認をしていきたいような場合.	全般.	逐次開発. ボトム・アップ（？？）.
		実環境型	○		○	一部に具体的な実現方法の確定していないところがある場合.	全般.	特になし. 世の中のでたらめプロトタイピングの全てがこれに入るのでは？？

プロトタイピングの分類（その２）

分類			狙い			具体的場面等	適用時期	言語／手法／環境等
			実現の可能性	利用者要求の確認	思考錯誤			
試用型	使い捨て型	シミュレーション型		○	△	画面の動き，処理の流れ等のユーザ・インタフェイス部分を確認するような場合.	早期.	簡単に早くできるものが好ましい.
		実環境型		○	○	処理速度等，マシン性能に関わる部分にユーザの要求が密接に関わる場合. 利用者要求＝実現の可能性といった感じなので，ほとんど試作型といえるような気がする.	早期.	しかしそもそも実環境型の使い捨て型というのは存在し得るのだろうか. . .
	骨格型	シミュレーション型		○		想定できない. なぜシミュレーション型でやらねばならないのか. どんなメリットが考えられるのか.		
		実環境型		○		試作か試用かの観点だけで無理矢理分けることしかできないのではないか. 現実的には実環境骨格型では，試用／試作は常に両方起こるのではないか.		

プロトタイピングへの期待

近藤直文

先ず最初に私の立場を説明します．私は，主に中小企業や大企業の一部門といった所謂オフコンクラスの事務処理のシステム設計からプログラム作成，運用コンサルタントといった仕事をしています．端末数で数台から十数台，プログラム行数だと数万行とか，ソフトだけの金額だと数M¥から10M¥未満といった小規模な仕事です．当然，ユーザもプログラムはもとよりシステム設計が何かわからないといったレベルの人が殆どです．

この種のシステム設計でＳＥに要求されるのは，ユーザの話を旨く引き出すテクニックです．現状分析，理想システムから実現可能システムへのブレークダウン等々以前の問題として，経営者は現場の実務を知らず，現場は改善の方向を知らず，又隣の現場の実務を知らないといった具合です．プログラム完成後のテスト段階で，社長や実務担当者の間でＳＥそっちのけで現状の実務フローについての議論になってしまう事もめずらしくありません．つまり，ユーザの誰も自社の実務について現状さえも把握していない場合が多いのです．かといって中小企業のオフコンクラスの小規模事務処理で専従者を置いてプロジェクトチームを作ったりしたのではコストパーフォーマンスが合わなくなります．

そこで考えられるありふれた一つの方法に，ユーザに参加者意識を持たせシステム設計に参加させる事です．ユーザが一方的にＳＥに要求を述べ，ＳＥがシステム設計書を作ってくるといったスタイルではなく，ユーザ自ら設計書なりプログラムの一部を自分の手で作る事によりシステム設計への参加意識を向上させる事です．

この為には，専門用語の羅列や動かないハードコピーではなく，実際に動作するモデルが必要であり，この点においてプロトタイピングに期待しているわけです．

現在行なっている例をあげてみます．オフコンは対話型処理であり，通常ユーザと数回打ち合わせをすれば，ポイントとなるデータ入力は検討がつきます．そこで，手持ちのプログラムで比較的類似のものを若干手直ししたり，ＤＢアクセスのない画面だけを画面定義ツールで作ったりして，ともかく動く画面を作ってしまいます．以後，打ち合わせはこの画面を動作させながら適宜ハードコピーをとったり簡単な変更はその場で行なったりしながら進めます．この方法のメリットは，文章よりわかりやすいというだけではなく，ユーザの要求で画面がどんどん変わっていきますので自然にシステムの検討に熱が入り，データ入力という詳細設計にとどまらずシステム全体についてより深く真剣に検討を加えられるという事にあります．又，コンピュータ嫌いの人にもとっつきやすいデータ入力画面のモデルですので，社内の各部署の人の意見も集めやすく，通常見逃しがちな例外処理等も把握しやすくなります．

帳票出力も手書き仕様書ではなくワープロでもよいからプリント見本を作り，ユーザの修正をその場で書き込んでいくようにします．最近はＳＱＬやレポータが使い易くなってきているので，可能な場合はユーザにも勉強してもらい，簡単なプログラム変更や検索位はユーザで出来るようにしていきます．又，事務フローや運用管理マニュアル等もプロトタイプをＳＥが作成し，後は出来るだけユーザに書きこんでもらうようにしていきます．中小企業の事務処理クラスであれば，入力と出力を押え，運用ルールが決まれば，システム設計はほぼ出来たようなものです．

さて，以上のような要求や仕様をプロトタイプ分類から見直すとどういう事になるでしょうか．

まず，設計の最初の段階から見本となるモデルを作っていき主目的も要求仕様の確認とユーザの参加者意識の向上にあるという点で試用型といえるでしょうか．この場合のプロトタイプ自身は，次の３つのパターンになります．

（１）他のシステムのプログラムを流用した単なる見本の場合．プロトタイプの分類外か？
（２）画面生成ツールを使用し，ファイルアクセスのない画面だけを作成するデモ的なもの．使い捨て型．
（３）手直しを加えロジックを追加しながら最終製品にしていく場合．骨格型，試用型．
（４）帳票出力で，レポータを使用しユーザの要求をその場で，可能ならユーザ自身で修正しながら作成していく．レポータの機能確認というユーザにとっての試作目的の面と，単なるユーザへのプログラム教育や，システム設計上の注意点の説明という教育的な面，データベース設計の検証という意味での試用的な面，参加者意識の向上といった面等を合わせもつ．

項目３以外は，プロトタイプの範囲ではないかも知れませんが目的は同じです．又，１と２の折衷で，既存のプログラムの画面まわりだけをツール等で変更して使用し，使い捨てとする場合もあります．

プロトタイプには，工数をかけずに模型を作るという事も必要ですので，ツールのみでなく既存のプログラムを流用するプロトタイプ手法というのもあっていいのではないでしょうか．又，プロトタイピングの目的がそのプログラム仕様の確認という事よりも，ユーザの参加意識の向上や問題発掘のきっかけ，コンピュータアレルギーの解消といった所に重点が置かれている所も通常のプロトタイピングと異なるかもしれません．

以上思いつくままに，書きちらかしてみました．こういう小規模なシステムだと証券取引所システムといった大規模なシステムを対象にした話とは噛み合わない部分が多いと思いますが，コンピュータの需要の急激な増加とＳＥ不足，ＳＥの質の低下が問題のおり，小規模システムのシステム設計にも目を向ける必要があるかと思いとりあげてみました．

（こんどう　なおふみ：ソフト工房）

プロトタイピング私見

近藤直文

プロトタイピングについて私なりの意見を纏めようと思って書き始めたのだが，例によって時間切れになってしまい思いつくままに書き止めたメモになってしまった．

【１】有沢誠：ソフトウェアプロトタイピング

有沢氏の本は，各種のプロトタイピングに対する考え方や用語，手法について手際よく紹介した本になっている．しかし，第２章の分類については分類の定義や目的がやや不明確に思えるところがあるのでこの章に従って思いつくままに述べてみる．

１．試作型／試用型プロトタイピング

氏は，プロトタイピングを，工業製品の試作品作成と同じアプローチとしての『試作品としてのプロトタイプ』とユーザの要求を確認する為の『試用品としてのプロトタイプ』に分類している．前者の試作品としてのプロトタイプは，従来から行なわれていた事であり，ソフトウェア開発の第３世代としてわざわざ取り上げる必要のないものであり，試作／試用と分ける事によりソフトウェア工学上の新しい思想としてのプロトタイピングの概念が明確でなくなり，従来の試作・研究用プログラムと混同されるおそれがあるのではないだろうか．むしろ，従来の試作とは異なる新しい概念としてのプロトタイピングを強調した方がわかりやすいのではないだろうか．つまり，試作は，ｍｏｃｋ－ｕｐ又はｂｒｅａｄｂｏａｒｄ（ｐ．３０参照）であり，ｐｒｏｔｏｔｙｐｅではないと言い切ってしまった方が後の議論がすっきりすると思うのである．くどくなるが，試作は，ユーザ＝試作作成者であり，エンドユーザの要求を確認するために行なうのは試作ではないと明確にすべきである．

２．使い捨て型／骨格（ｎｕｃｌｅｕｓ）型プロトタイプ

『骨格型』という名称が疑問だが，とりあえず『非使い捨て型』＝『骨格型』という理解としておく．

先ず，『使い捨てられるもの』が何なのか明確でない．使い捨てられるものとしては，機械実行形式，ソースコード，プログラム仕様書，要求仕様書，要求モデルや，要求の元となる背景といったものが考えられるが，氏の記述からみて，ソースコードのように思われる．しかしこういった分類では，同じレベルのプロトタイピングでもＣＯＢＯＬで記述すれば使い捨てで，仕様書言語で書けば使い捨てでない即ち骨格型であるという事も生じるわけである．又，要求モデルの全てのサブシステム／モジュールがどちらかの分類に当てはまると言うことは考え難く，実際は両者のまじったものになるはずである．したがって，要求モデル全体のプロトタイプについてこのどちらかに分類しようとすること自体に無理がある．

次に，結果として不採用になったものも使い捨て型に含めるのかが不明確である．氏は『使い捨てか骨格かの区別は，プロトタイプの試用結果が設計に及ぼす影響範囲によって決まることになる』と述べているが，もし氏がプロトタイピング結果によっての分類も考えているとすればあまり意味のない分類のように思う．あくまでも，どういう目的で，どういう背景の上に行なうかという分類に終始すべきである．

骨格型についても『検査より早い段階で，設計に選択枝が残されている段階での参加』とだけ定義してしまうと，後述の『パラダイム』のようにユーザの要求仕様とシステム作成者の設計仕様を突き合わせる為の共通の土俵としてのプロトタイピングといった，プロトタイピングの本来の定義と思われるものが抜けてしまうのではないだろうか．

結論として，使い捨てか否かの単なる分類はあまり意味のあることではなく，むしろ，見本型か核型（ｎｕｃｌｅｕｓ）とか，プロトタイピング言語の手動翻訳を要するか，自動生成で機械実行形式に変換出来るかといった分類の方が意味があるように考える．

３．シミュレーション型（／実環境型）

この区分は，正直にいってよくわからなかった．ターゲットマシーン上で動作させるか否かとか，ＤＢの構成，データ量，負荷等の実環境にどれだけ近付けられるか否かといった議論としてとらえてよいのであろうか．

その場合，何でインタプリタが中間にあたるのか不明である．ＢＡＳＩＣインタプリタで最終プログラムを作る人はあまりいないだろうが，レポータ等でインタプリタ型で実働プログラムとして十分使えるものはあるし，ＬＩＳＰ等インタプリタのプログラムも増えていくのではないだろうか．

４．ラピッドプロトタイピング

早く安くという意味でだけとらえれば，氏のいう『使い捨て』と同じ分類になってしまうのではないだろうか．又，２～３週間もかかるのがはたしてラピッドなのであろうか．

むしろ，ユーザとの打ち合わせの時に目の前で仕様変更した結果のプロトタイプを提示するような事をラピッドプロトタイピングと言うべきではないだろうか．

５．プロトタイピング支援環境（大規模開発環境型／道具箱型）

氏も述べておられるように，両者の中間的なものもある．あまり意味のある分類には思えない．むしろ，具体的な手法で分類すべきであろう．

６．構造的手法の補助手段型／オペレーショナル手法型

第２章の分類にはないが，３章以降の従来型手法との対比にむしろ本質的なプロトタイピングの分類があるように思う．

【２】大野豊：ソフトウェア工学の背景と展望

首記文献の概要次の通りです．

人間の概念の表現であるソフトウェアと物質的な製品の生産技術には基本的に相違があり，従来のものの工学とは異なった工学体系が展開されることが望まれる．ソフトウェア工学には，従来の工学で基礎とする自然界の科学的法則，絶対に守るべき物理・科学的法則が欠如している．人間の知的働きの法則を基礎にした新たな原理を求める必要がある．

ライフサイクルモデル（ソフトウェアの動的発展特性も考慮したものもある）として次のものがある．

ウォータフォールモデル　理想であって現実は実行不可能に近い．

コストモデル

プロトタイプモデル

逐次改良版モデル又はスパイラルモデル

現実のハードウェアの制約にとらわれることなく，人間の特性をより深く反映した新たなモデルに基づいたソフトウェア表現法と構成法を『ソフトウェアパラダイム』と称し，従来の手続き型に対して『新形態プログラミング』という．

『パラダイムという言葉は，もともとモデル，パターン，例などを意味するが，ここでは問題を解くときに用いるモデル／アプローチをいう．ソフトウェアパラダイムというときには，用いる言語クラス（関数型，論理型，対象指向型など），その言語をサポートするソフトウェア開発環境，その環境でシステムを作るのに用いるソフトウェア工学上の規律などの意味が含まれている．

ソフトウェアの定義：プトグラムだけでなく，プログラムを導入し，使用し，また開発・保守を行なうために必要なすべての文書を含んでいる．

【３】松本吉弘：ソフトウェアに対する要求の形成

氏の意見の概要は次の通りである．

従来のライフサイクル論では，要求者の解釈（要求モデル）と，製作者の解釈（実現モデル）とが独立に作られるべきとした為，前者から後者に移行するときに問題を生じてきたとし，この解決の為に，ライフサイクルを通して要求者，製作者に共通なただひとつの解釈（モデル）を持つべきだとしている．

ここで重要なのは，要求者と製作者が交信し，改善するのはこのモデル只１つであり，モデルから実行可能プログラムへの変換は自動化しこの過程に新しい要求や修正をもちこませないようにする仕組みにするとしている点である．

氏は，このような考えに基づいて設定される解釈（モデル）及びこれに基づいたプログラム生成過程／生成方法をパラダイムと呼び，次の三つのフェーズに分類している．

１．プロトタイピングパラダイム

プロトタイピングとは，プロトタイプを作り実行することによりモデルの正当性を高めていく行為の事．で次の三つに分類される．

（A）シナリオ型（紙芝居型）プロトタイプ
対話部分についてあたかも目的プログラムと同じような効果を使用者に与える為だけに作成するもの．使い捨てになる場合が多い．強いて当てはめれば，有沢氏のいう『試用型＋使い捨て型』であろうか．
（B）原版型プロトタイプ
有沢氏のいう『骨格型』に当たる．
（C）機能検証型
有沢氏のいう『試作型』に近いのだろうか．

２．操作的(Operational)仕様パラダイム，
ＪＳＤ法，ＰＡＩＳＬｅｙ，Ｇｉｓｔ，Ａｒｇｕｓ，オブジェクト指向型操作的仕様，他．

３．形式的仕様パラダイム
所謂歩行可能な仕様ではなく，文書・図表を読んだり見るものである．半自動的に実行可能仕様に変換される．ＳＡＦＥ／ＴＩシステム，ＰＳＩシステム，ＨａｖａｒｄＰＤＳ，Ｄａｒｃｏ，等のシステムが研究されている．

【４】プロトタイピング私見

思いつくままに勝手な意見を述べてみます．とんちんかんな事や，矛盾した事も言っていると思いますが思いつきを書きなぐっただけですので読み流して下さい．

１．新しい思想としてのプロトタイピングの必要性
コンピュータのハードウェア技術の急速な進歩によりその性能対価格比は桁違いに向上し，従来採算のあわなかったような分野も含め，あらゆる分野でコンピュータが使われる事となった．即ち，従来コンピュータとは無縁であった人達迄コンピュータを使う又は使わざるおえなくなった．この為，ＭＭＩの高度化が求められるばかりでなくコンピュータに関する知識の全くない人や誤解をもっている人を含め経験や知識の大きく違う人がコンピュータシステムにかかわるようになってきた．
しかるに，要求仕様書作成からプログラム作成及び保守に至るソフトウェアの生産技術にはハードウェアに見れれるような画期的な向上はない．１９５５年から１９８５年の３０年間にハードの生産性は１０の６乗向上したのに対し，プログラマの生産性は僅かに１３倍になったにすぎないという報告がある（注１）．この大きなギャップが今日のソフトウェア危機を招いている．
つまり，単に需要が増えたので生産が追い付かないという次元の問題ではない．ハードに比べれば，ソフトウェアの生産性や生産技術はたいしてかわっていないといっても言い過ぎではないのに，世の中では『ハードの急激な進歩＝コンピュータシステムの進歩（含むソフト）』という誤解が一人歩きし，これが今日のソフトウェア危機の真の原因となっているのではないだろうか．
この問題を解決するには２つの道しかない．１つはハードウェア技術の進歩に匹敵するようなソフトウェア技術の進歩を得る事である．もう１つは，『コンピュータシステムの片方の車輪であるソフトウェア技術は，ハードのようには進歩していない』という共通認識のもとに，コンピュータシステムに於けるソフトウェアの位置付けを変えてしまう事である．
後者は，ハード（工業製品）の製造技術とソフトのそれとは本質的に違うものであるとういう考えである．ハードウェアのように完成品としのソフトウェアがユーザに納入されるという幻想を捨ててしまい，ユーザとソフトウェア技術者が逐次改良・変更を続けながらソフトウェアを育てていくという考えに変える事である．これは同時にウォーターフォールモデルを捨てる事である．
まず，完全な要求仕様が先に存在するという考えを捨てる．
汎用機が今日のパソコンよりも小さかった頃なら，コンピュータ化の対象も現実モデルの極く一部だけであり，仕様書化しやすく，又ユーザサイドも業務に精通したベテランを中心に十分に要求仕様検討をする余裕があったので，実用上十分といえる要求仕様書を先に作ることも可能であった．然し，今日では，コンピュータ化の対象はあまりにも広範囲であり，アナリストは元より肝心のユーザ自身もコンピュータシステムに対する要求仕様そのものをつかみきれないの状況にある．又，コンピュータ化の為の事前準備にユーザ業務に精通しているベテランを割り当て多大の費用をかけられないようなケースが増加してきているのも，要求仕様の把握を困難にしているともいえる．さらに，本質的な問題として，外部事情の変化や，時間経過とともに，或いはユーザの部分的要求がコンピュータシステムとして実現する事によりユーザの要求自身が変化する（潜在要求

の顕在化でなく文字通り変容する）という事がある．

次に，保守サイクルという考え方も捨てる．

ユーザの要求の変化やソフトの不具合は保守サイクルで吸収し，保守のコストが新規システムの作成コストを上まわるようになったら次期システムの作成にかかり新たなライフサイクルが始まるといった考えも捨てる．コンピュータシステムは，最初から完成してものがあり，それを保守するのではない．ユーザの業務内容と同様，常に内部に問題点や矛盾を抱えており，それをたえず改善し，経営トップの要求に応じて変更していけるものでなければならない．

新しい手法は，コンピュータ化対象業務だけではなく，ユーザの業務内容そのものをモデル化するものである．

（１）現実モデルを作成する．業務マニュアルとか事務フローを集大成したものである．

（２）同モデルをコンピュータ化も一つの手段として，経営要求に合わせて改善する．ワークデザイン手法等が使用できる．人，物，情報等の動きをシミュレーション出来るツールやワープロよりましな記述ツールがあると望ましい．又，改善案の内，コンピュータに関するもので問題になりそうな部分は，実行可能なモデルを作成し，コンピュータ化すべきか否か，ハードを含めてどのような方式がよいかを評価し，又，発想の転換の刺激材や仕様の漏れがないか等の確認にも使用する．既存のプログラムを流用したりプロトタイピングツールを使用し，出来るだけ動作可能な部分モデル何通りか付け評価するる．ここで作成したプロトタイプを，そのまま又は改良する事によりボトムアップ型のアプローチを部分的にとる事も可能である．

（３）コンピュータ化の対象とその周辺システムのモデルを作成する．データ構造や業務フローの大枠を決定する．又，エンドユーザが追加変更してよい部分と，データベースマネージャ等のシステム責任者の判断を要する部分を分ける．

（４）ラピッドプロトタイピングを併用して細部仕様の決定とプログラム作成を同時並行的に行なう．

プロトタイピングは，（２）の要求仕様作成及びその前作業といえる部分で主として使用され，後は，細部仕様の決定に補助的に使われる．理想は，実行可能な仕様書言語によるモデル作成の実現であろう．又，エンドユーザのプログラム作成や非定形業務の実行は，それ自体プロトタイピングの一種といってもよいのではなかろうか．

このモデルには，ウォーターフォールモデルでいう保守サイクルはない．つまり，経営要求等トップレベルの要求が変わった時は（２）のモデル改善を実行し，その結果（３）のコンピュータモデルの変更になる場合と単なるプログラム追加になる場合と基幹システムには影響なくコンピュータ以外の部分の改善で終わる場合とに分かれる．

肝心なポイントは，経営方針といったトップレベルの要請から業務マニュアルといった細部の規定迄をもモデル化してしまい，現実のシステムとモデルを一致させていく事である．このモデルを知識ベースやデータディクショナリ等コンピュータ処理可能なモデルにする事ができれば，システム変更時の補助ツールとして大きな効果をもたらすだけでなく，将来は改善案のもたらす効果や問題点をシミュレーションできる可能性も開けるのではないだろうか．

以上かってな事を述べてみたが，問題点は（３）のコンピュータシステムのモデルの変更にどの位の手間がかかるかという事であり，画期的なツールが出来なければ，システムの硬直化は避けられず，結局は現状のウォータフォールモデルと同じことになってしまう．

しりきれとんぼのような話になってしまったが，ソフトウェア危機の本質は，ソフトウェア自身の生産性向上だけで解決する問題ではなく，ユーザ自身が要求や問題点を的確につかみきれないでいるところにあり，今後はこの部分を解決することにも目を向けていく必要があると考える．

２．プロトタイピングの問題点

有沢氏も『ユーザの潜在的要求をしることが大切だといわれる』と述べ，『客の描いた間取り図通りの家を建てても客から喜ばれることは少ない．と建築のたとえがよくひきあいにだされる』と述べているように，単にユーザの要求通りのシステムを作っただけではコンピュータ化の目的を達した事にはならない．むしろユーザの要求の先をどこまで読めるかがアナリストの腕のみせどころである．

つまり『経営者のトップレベルの方針を如何に具現化するか』といった視点からの検討が必要である．従来の手法では先ずトップレベルのシステム化目標があってそれをモデル化する過程でエンドユーザの要求と現状分析を参考にしていったので，結果としてはエンドユーザにとって使いにくい失敗したシステムになったとしてもトップの方針，システム化の基本方針自身を忘れた訳ではなかった．

しかし，エンドユーザによるプロトタイピングツール等が実用化したばあいともすれば，経営トップのシス

テム化の方針を忘れさった現状拘泥型のシステムや過剰仕様の情報公害型のシステムになったり，又，セクショナリズムを助長し，従来手法より悪い結果に陥ることはないだろうか．

この心配は，プロトタイピングが所詮はマンマシーンインターフェースのプロトタイピングにすぎず，コンピュータシステムを含む事務システム全体をシミュレートするものではないからである．プロトタイピングのもつ強烈な印象の前にユーザもアナリストも本来の大きな目的を見失い小手先のＭＭＩにとらわれてしまわないかという事である．

プロトタイピングは飽く迄も要求仕様作成のツールにすぎず，システムそののものの動機付けを行ない，発想を転換し，トップの要求に対する対処法を見つけるのは，プロトタイピングとは無縁のそれ以前のフェーズである事を忘れてはならない．

（注１）

孫引きですが，大野に引用されていた表の抜粋です．

	1955	1965	1975	1985
機械（ハードウェア）性能 1	10**2	10**4	10**6	
プロフラマの生産性	1	2.4	5.6	13.3

出所：Art Benjamin Associates Ltd.cited in Electronics, May 8,1980, p.143. "Computer Technology Shifts Emphasis to Software:A Apecial Report".

（参考資料）

（１）　玉井哲雄『ソフトウェア開発におけるプロトタイピング』　ｂｉｔ１９８３年１２月号　Ｐ．１４４９

（２）　牛島和夫他『日本語文章推こう支援ツールのプロトタイピング』第４０回ソフトウェア工学研究会Ｐ．４３

（３）　ジェームス・Ｒ・ジョンソン『ソフト開発基幹を大幅に短縮するプロトタイプ手法』日経コンピュータ１９８４－２．６　Ｐ．８１

（４）　大野　豊『ソフトウェア工学の背景と展望』情報処理　１９８７　ｖｏｌ．２８　Ｎｏ．７　Ｐ．８４５

（５）　松本吉弘『ソフトウェアに対する要求の形成』情報処理　１９８７　ｖｏｌ．２８　Ｎｏ．７　Ｐ．８５３

@@@@@@@@@@@@@@　有沢氏の本　@@@@@@@@@@@@@@

＊使い捨て＝『早く安く』．コストをかけないで早い時期に実現するために“使い捨て”にする．

→@使い捨てられるものはプロトタイプのどの部分か？

機械実行形式
ソースコード
仕様書
設計方針
要求仕様
文書化される前の原要求

＊骨格（反使い捨て）＝『設計段階でユーザに参加してもらおう』

ＳＥがわの選択肢が複数あるときユーザの反応によって決める．

変更はしない．『設計方針は基本的には固まっており』

歩行可能な仕様書．標準部品化．

＊変更：選択

＊試用結果が設計に及ぼす範囲により使い捨てか骨格か決まる．

→@単なる結果論では？たまたまなった事ではなくどういう方法を選択し，それが成功したか失敗したかという話にすべきである．単なる結果のプロトタイプなら従来と同じ．

＊シミュレーション型：実環境型
シミュレーション：性能面も可．
試作：実環境のみ．
試用：何方でも．

＊ラピッドプロトタイピング

（第３章：Ｐ．４７から

＊構造的アプローチ　　使い捨て
要求＝問題の理解と実現をはっきり分けて，作る側にひきずられないようにする．
→@ここが問題．作る事を理解せずに要求が纏められるのか．そもそも要求仕様は存在しうるのか．

＊オペレーショナル　　骨格型

＊ユーザの追加要求は事前に推定出来ない．潜在要求も同じ．

（第４章　Ｐ．６７～
＊構造的な手法とは独立
＊構造的手法にとりいれ欠陥を補う

＊失敗から多くを学び取るのであって無駄にはならない．
＊試作品は殆どの場合使い捨て．
未経験の要素を含む時．
→@使い捨てられるもの：プログラムコードそのもの
＊ことばの問題　＃[Gregory84] mock-up,　breadboard

＊試用品
＊要求試用書作成後システム完成迄ユーザが関与しないのが問題．関与出来るようにユーザにわかるものとして試用品を作りフィードバックさせる．
→@ユーザの要求自身の変化と要求仕様実現の失敗の区分．

＊試用品としてのプロトタイプ：『ユーザの要求を設計製造の担当者たちが正しく理解しているかどうかを調べ，されにユーザの潜在要求を早い時点で察知して設計に取り込むことを目的とする．』．早期に実施要．
機能面，ＭＭＩ．性能面，文書，エラー対策は含まない．
定常状態でのデモンストレーション．
使い捨てとそうでない場合（＝骨格 nucleus）の両方あり．
→@骨格は frame を想起させ誤訳．核型のほうがよい．

@@@

第１世代　コンピュータに出来る事だけを
第２　ユーザ要求を参考にコンピュータの専門家が実現可能な事だけを．最初に，コンピュータの出力／入力があり，ユーザはそれに合わせるしかない．それしかハード／ソフト（ＯＳ／言語）の能力なし．ユーザは，コンピュータ自身に触れる事はない．別室に隔離されている．
第３　コンピュータがユーザの居場所（部屋，机）に近づき能力の向上により多様な要求に対応可能．だが，要求が何かわからなくなってきた．実世界のモデル化しやすい部分以外もコンピュータ化の対象になってきた．

工業製品の試作品からのプロトタイプ類推．

工業製品のプロトタイプには２つの種類がある．１つは技術的問題点を調査する為の試作品であり，研究的性格のものである．外観は二の次で，機能中心の動くモデルである．もう１つは，完成品からユーザが受けるイメージを類推する為に，設計図と実物のギャップを埋める為に作る模型である．建築物や土木構築物の模型，自動車のクレーモデル等芸術的あるいは心理的反応を見る為のもので，外観や手触り等の人間の５感にかんする事項を重要視する．動かない静的なモデルが多い．

これに対しソフトウェアのプロトタイプでは主に
★ユーザには対立する要求がある．セクショナリズム．誰が纏めるのか．

モデルが大事か，動機付けが大事か．

色見本型　模型でなく実物の（素材）の小片
詳細な検討可．
全体との調和や，本来重要でない箇所に力がさかれる
郵便番号辞書は大量入力には無用

住宅建築の例
基礎知識習得
モデルハウス
類似の家を見る
務店の評判を聞く
間取り図
概観図
図面
基礎工事立ち会い
色合わせ
上棟式立ち会い
完成検査引き渡し
３か月点検・細部手直し

ライフサイクルルは本当にあるのか
単にソフト開発技法／ツールに限界があるためそうなっただけ．
現実モデル自体はライフはない．
仕様書＝実行可能になれば，保守と新規作成の差はあいまい
ハード上の限界等による単純なリプレースは

例えば，『バグのないソフトウェアはない』という認識のもとに，虫とコンピュータシステムとの共存をはかろうという考えもその一つと

（こんどう　なおふみ：ソフト工房）
１９８８－０９－０６
訂１ 1989-01-26

プロトタイプ／プロトタイピングに関する個人的な解釈／言葉の定義

渡邉　雄一

はじめに

前回の討論会での宿題であった表記の件について，取り敢えずはまず，有澤先生の本をもう１度注意深く読んでみた．

〔記号の説明〕

『』：有澤［１］からの引用を示す
→：上記の引用に対する私の解釈・意見
☆：私の主張

【試作品】

『何らかの意味でこれまでにない新しい面をもつ製品を作るときに』，『何かこれまでに経験していない要素を含むとき』
→試作の意図は概ね「新機能」の実現可能性の判断！？

☆試作目標・目的が設定された計画にそって，その意図を満たすための生産物．その生産物は最終製品ではない．

『試作品としてのプロトタイプは，ほとんどの場合，目的をはたしたあとは捨ててしまう．すなわち，使い捨て品である．』
→目的は事前に設定されていなければならない．

『試作品は，作りかたを模索するもの』
→開発者による，開発者のための「プロトタイプ」

私の考えた幾つかの切り口

〔適用時期・場面〕システム設計開始直後～詳細システム設計（プログラミング仕様）まで

〔中間生成物〕なし（期待出来ない）

〔最終成果物〕機能実現のための手法．それ相当のソースコード．

〔期待される効果〕
・具体的コードとして機能が示されているので，
1 システム作成時の誤解は原則として在りえない
2 検証が用意なため，システム設計担当者の責任が明らかになる
3 当然，開発での生産性が従来手法に比べて上がる

〔危惧される効果〕
・機能実現の過程が評価し辛い

〔その他〕
・機能の実現可能性の評価は，プロトタイプ作成者の技量に依存するので，ある意味でシステム設計と同等である
・プロトタイピングの過程はさほど重要でない

【試用品】

『試用品としてのプロトタイプは，ユーザの要求を設計製造の担当者たちが正しく理解しているかどうかを調べ，さらにユーザの潜在要求を早い時点で察知して設計にとり込むことを目的とする．』
→設計の補助・支援を意図する．プログラムやシステムとしての実現可能性の検討ではない．使い心地などのＭＭＩ（マン・マシン・インタフェース）の向上なども意図している．

目的は期待こそされ，結果として明確になる場合が多く，とかく曖昧（あいまい）になりがち？で設定しにくい．

『試用品のほうは作り方自体は確立されている』
→システムの内部構造よりも，最も基本的な要求仕様に対する（操作性含む）機能検証のために用いる．
→開発者による，開発者と利用者との接点のための「プロトタイプ」．

私の考えた幾つかの切り口

〔適用時期・場面〕システム設計終了段階～詳細システム設計（プログラミング仕様）まで

〔中間生成物〕部分的な仕様

〔最終成果物〕機能検証されたシステム設計のための要求仕様

〔期待される効果〕

・最終成果物（ターゲットシステム）を期待したものを想定しているので，それの規模を計る目安が得られる．
・プロトタイプからの出力は，最終仕様に極めて近いものであるので，
　1 最終仕様書，利用者向けドキュメントなどの事前作成が可能
・部分的な中間成果物からでも，システム開発に部分的に着手可能な場合がある

〔危惧される効果〕

・機能評価が中心となるので，
　1 被験者が複数の場合にはその意見の取捨選択は微妙である
　2 時に同道巡りをすることにもなる
　3 「要求の定義ミス」と「要求の表現不足」が，元要求仕様の変更を促すため，要求仕様の目標・欲求がなかなか確定しない．
・プロトタイプとターゲットとの実行性能の差が，利用者に過大な期待を与えてしまうことがある
・被験者へのプレゼンテーションは，頻繁に行うと返って効果が薄れることもある

〔その他〕

・機能の実現可能性の評価は，プロトタイプとは別にされるべきである．
　特に実際にインプリメントする際のアルゴリズムや例外処理など利用者・被験者とは直接関係ない（システム屋の仕事の範疇の）部分は慎重に検討すべき
・プロトタイピングの過程は重要

（わたなべ　ゆういち：電力計算センター）

有澤氏によるプロトタイピングの分類について

塩谷　和範

分類	目的	適用場面 / 注釈
試作型	実現可能性の検討 最適仕様の選択 性能評価	試作品を作る必要があるのは何かこれまでに経験していない要素を含むとき。 ＋ 要求定義段階で作る。 ＊多くの場合目的を果たした後は捨てる。 ＊実環境で作る事に意義がある。
試用型	ユーザによる機能確認(吟味) (ユーザ要求の解釈の確認) ユーザの潜在要求の発掘	骨格が整った時点で主として機能面及びＭＭＩについてユーザに確認を求める。 ＋ 設計段階で作る。 ユーザ要求に幅があるとき。 ユーザによる複数の実現案からの選択。 ＊実環境で作る必要はない(両方可)。 ＊骨格とすることは可能。 ＊試作／試用兼用型は使い捨てとなる。
使い捨て型	プロトタイプを早く安く作る。 複数の実現案の作成。	ユーザによる試用から設計情報を得る。 ＊言語は何でもよい。(簡便な方が良い) 短期間に実現でき・手直しが出来る。 例：BASIC,APL,AWK, Smalltalk-80,Simula-67 Prolog (論理式に近い仕様の時)
骨格型	目的システムの骨格となる。	基本的な設計方針は固まっていて、ユーザの反応による根本的な設計変更がない事。 ＊システム骨格上に仮に実現した諸機能とインタフェースをユーザに提供し、要求の満足度を調べる。 ＊目的システム記述言語と同一言語を使用 ＊仕様を作るのが目的では現場になじまない。仕様からの自動変換が出来ることが必要。 ＋ 設計段階でユーザに参加してもらう。
シミュレーション型	条件を変えて実験する。 性能面をシミュレーションする。 独立なプロセスの協調としてモデル化する。	様々な実験が出来る。 ユーザ試用の際の統計データ収集可。 要求仕様の自然なモデル化が出来る。
(インタプリタ型)	仮想機械をシミュレートする。	シミュレーション型と同様な利点
実環境型	目的システムの環境で作る。	

出典：「ソフトウェアプロトタイピング」/近代科学社

以上は有澤による分類だが,曖昧な点があるので次のように解釈・定義した.

分類の分類

視点による分類: 試作型/試用型
手段による分類: 使い捨て型/骨格型
環境による分類: 実環境型/シミュレーション型

当然プロトタイピングを行なおうとする時にはプロトタイピング計画(P02.*-01)を最初に作り，その中で目的・手段・環境を決定し，適切なプロトタイピングが行なわれなければならない．さもなければ"適当にやるのがプロトタイピング"(P01.*-01) と言うことになってしまう．これでは悲しい．

まず一番問題となっている"骨格型"プロトタイピングだが，骨格の意味を最終システムあるいは目的システムの一部となるものと解釈した．

そこで"骨格型"プロトタイピングとは，
「成果物が最終システムの一部となることを目的に行なうプロトタイピング」
と定義した．従って，試用・骨格(・実環境)型で目的プログラム（最終システム記述用言語による成果物で最終システムの一部となる）を出力するケースに加え，何らかの変換機能により目的プログラムを作成するケースも骨格型プロトタイピングであると解釈する．
例としては，前回ミーティングで発表のあった山本さんのケース(P02.1-04)も最終システム記述用言語(FORTAN)を使ってプロトタイプしユーザによる試用後それを最終システムに発展されているので試用・骨格・実環境型のプロトタイピングと言える．

"使い捨て型"は，手段としての使い捨てを意味する．したがって
「プロトタイプを早く速くかつ安くたくさん作るために再利用を前提とせずに作ること」
と定義する．当然"もったいないから後で使う"ことも起こり得るが，これはあくまでも結果であるので骨格型に分類し直すことはしない．

次に"実環境型"は，
「最終システムを実現する環境と同一環境・同一言語を使用するプロトタイピング」
と定義する．
広義に解釈すれば，最終システム記述言語を使用しなくても実行環境さえ同じであればよいとも考えられるが，こういうケースは"シミュレーション型"に分類した．

"シミュレーション型"は，「"実環境型"でない場合」とした．

この理由は，狭義のシミュレーション型は実動作環境を疑似(Simulate)するものだが，実動作環境で4GL・高級言語など最終システム記述言語と違う言語を使ってプロトタイピングをすることは最終システムの動作を疑似することに他ならないからである．

試作型/試用型については，特に曖昧とは考えないので再定義しない．

（しおや　かずおのり：ＳＲＡ）

プロトタイピングに関する言葉の定義について

田中　慎一郎

１．　はじめに

挙げ足取りのようになってしまって嫌でもあるのだが，少し気になることの一つに，「プロトタイプ」と「プロトタイピング」という二つの言葉の間での意味あるいは使用法の混同がある．

私の中の漠然としたイメージの中では，「プロトタイプ」とは何らかの目的を持って試験的に作られた「もの」を指し，「プロトタイピング」とは，通常「プロトタイプ」を中心に，使用者／作成者が確認を繰り返しながら，より良いものを作り上げていくある種の手法を指している．

主題から少しはずれるが，ここでもう少し説明しておくと，プロトタイプを作れば（結果として存在すれば）必ずしも全てプロトタイピングになるわけではないし，プロトタイピング手法を採ったからといって，必ずしも明確なプロトタイプが存在するわけではないと思う．言い替えれば，「プロトタイピング」と言うのは，開発手法として「明確な思想」を持っていて然るべきものであり，逆にその思想に従っている限り，必ずしも「プロトタイプ」を必要としないだろう．

このように少なくとも私にはこの二つの言葉は，「もの」と「手法」とを表す明確に異なるものなのだが，それが意識されていない使われ方をかなり頻繁に耳にするのである．例えば，「プロトタイピングを作る」，「プロトタイプでやる」等々．後者はまだそれほどおかしくは感じられないのだが，それでもこれらの用法を採る人の意見を聞く時には，焦点を絞るのが難しいように思われる．つまり彼らの言う「プロトタイピング（プロトタイプ）の問題点」とは，手法と試作物のどちらを指しているのかが全くわからなくなるためだ．

プロトタイプあるいはプロトタイピングという言葉が非常に一般的になってきた以上，まずこのあたりから押えていかなければならないと思う．

２．　有澤氏の分類に関して

２．１　試作品／試用品

まず，P01.1-04 で述べたように，分類の用語としては用いたくない．もし使う場合は，「開発者が使用して，開発者が想定あるいは期待した性能を満たすことができるかを確認するためのプロトタイプ」／「利用者に使わせて，利用者の要求が正しく反映されているかを確認するためのプロトタイプ」という分類ではなく，これら二つは「試作品」とし，「（利用者に）使わせてみることによって，（既出の要求の確認ではなく）これまで顕在化していなかった新たな要求を顕在化させるためのプロトタイプ」のみを「試用品」と呼びたい．しかしこれも適切だとは思えない．

つまり，もとの分類は，「利用者から見て」，「単に作られただけか」／「使うことができたか」というだけの意味しか持たないように感じられ，分類としての意味があまりないため，上のような定義を持ち出したわけだ．しかし，もとの意味での「試用」の中からも新たな要求が顕在化することもあるし，言葉を流用することは避けた方が好ましい．従ってこのような意味で使うのであれば，むしろ主目的によって明確に，「確認型」／「模索型」のような言葉を用いたい．

より適切な言葉はないだろうか．

２．２　使い捨て型／骨格型

さて，P01.1-04 で私は大きな誤解をしていたようだが，これに関しては「骨格型」という言葉の選び方にも問題があるように思われる．ここは「拡張型」のようなものにしていただきたかった．

もう一つ．何を捨てる／捨てないといっているのかが良くわからないところも問題である．プロトタイプのソース・コードが少しでも残っていれば「使い捨て型」ではないように読みとれるが，一方，仕様からプログラムが自動生成できるようなシステムは「骨格型」を支援するものであるようにも書かれている．この場合プロトタイプのソース・コード相当のものが，最終成果物に残っていることは少ない．これが「骨格型」であること自体にあまり違和感はないのだが，分類用語としての視点（基準）が良くわからない．

ここには二つの視点が入り組んでいるようだ．つまり，捨てる／捨てないという視点と，本開発に入る前に行なう／本開発自体を逐次開発的に行なうというような視点とが感じられる．必要があれば，別の言葉で分離させるか，より本質を明確にするような言葉で表したい．

２．３　シミュレーション型／実環境型

この言葉にも若干の疑問がある．これは「プロトタイピング用の簡易言語のようなものを用いるか」／「最終成果物と同じ言語／環境を用いるか」という様な意味合いだと考えられるが，実環境でもシミュレーションは行なえるようにも思われるので，どうも落ち着かない．

ここでいうシミュレーションとは何だろうか．

２．４　その他の分類

この他にもいくつかの視点を導入しなくてはならないのではなかろうか．例えば，同一マシン上で４ＧＬから３ＧＬに落して動かすような形態の時は，一体どのように分類すれば良いのだろうか．

この場合，シミュレーションも行なうような気がするし，実環境上で作成されると言えなくもないし，「使い捨て型」／「骨格型」の分類上も問題がありそうだし，利用者が作成者だったりすることもある．それでもこれはプロトタイピングの一つの典型のようにも思われる．なかなか難しいものを感じる．

３．　方法論に関して

プロトタイピングを考える上で，見過ごしてはならないのは方法論だと思う．ここで言ってるのは，教科書に載るようなきちんとしたものという意味ではなく，ある程度の方針／計画性がなくてはならないという意味である．計画性のないプロトタイピングがあまりにも多いような気がする．せめて，「何のためにプロトタイピングをするのか」，「あるプロトタイプは具体的にはどういう目的のために作られたか」，「それがどのように評価され，どのように反映されたか」程度はまとめられるものであって欲しい．

もう一つ．プロトタイプを「使い捨てにするつもりか」／「核にするつもりか」というのは，かなり重要な方針だと思われる．

使い捨てるつもりで，なおかつ本当に使い捨てるのであれば，それは単なる仕様の確定であり，ある意味ではどのような作り方をしてもかまわないことになる．一方，核にするつもりであれば，これは開発が始まっていることになるため，安易な「試み」はなるべく控えなくてはならない．適当でないものがそのまま最終成果物に残されてしまうことがあり得るからだ．

用語の定義とともに，いろいろな状況下における注意点／示唆なども徐々にためていくと面白いように感じる．

（たなか　しんいちろう：ＳＲＡ）

第３回プロトタイピング討論 議事録

日　時：昭和６３年９月７日　１８：３０～２３：００
場　所：ＳＲＡ　３Ｆ
出席者：渡邊　雄一（電力計算センター）　青島　茂（富士ゼロックス情報システム）
　　近藤　直文（ソフト工房）　塩谷　和範（ＳＲＡ）
　　田中　慎一郎（ＳＲＡ）

前回の集まりは，事例報告に終始し，本来の討論ができなかった．このため実質的には今回が討論第２回目である．

さて今回は，「プロトタイピングに関する言葉の定義をする」ということになっていたが，そのあたりと関係して，一度有澤先生をおよびしてミニ・フォーラムでも開きたいという話もでた．また，プロトタイピング自体に関する捉え方などについても話が及んだが，時間の関係で充分には討論できなかった．次回に継続する．

以下ではこれら３点に関して，簡単にまとめてみる．

１．　ミニ・フォーラムに関して

１．１　プロトタイピング・フォーラム

早急に我々なりの言葉の定義／解釈を行ない，１１月のはじめ頃に有澤先生を招いてミニ・フォーラムを開催したい．長野支部などとも相談し，可能であれば合同で行ないたい．次回幹事会で佐藤千秋さんと相談し，受け入れ体制が固められれば，有澤先生に連絡をとってみる．

このフォーラムの結果を踏まえて，プロトタイピングに関するアンケート案を作成し，１１月の環境ワークショップで予備調査をする．

１２月中に本調査を行ない，２月中くらいにまとめ，その結果等を利用し，３月頃に東京でフォーラムを開催したい．

（現状では十分な準備が出来そうもないので，フォーラムに関しては一旦保留したい；９／９，田中，渡邊）

１．２　ＯＳ２＆ＬＡＮ（もしくはパソコンとＬＡＮ）フォーラム

９月のデータ・ショウで，製品が出るようであれば，メーカの会員の方に相談し，１２月あたりにフォーラムを開きたい．

２．　有澤先生の用語に関する我々の定義／解釈

なるべくその言葉が示す本質的なところを取り出すこととし，それぞれの関連はまた別に考えることとした．つまり，「試作型は使い捨て型であることが多い」と言った時点でその意味するところがぼやけてしまうので，これをできるだけ避けることとした．

２．１　試作型プロトタイア

（合意されたと思われるもの）

- 試作型と試用型は，対立する概念ではない．
- 試作／試用の分類は，アプローチの決定にはあまり関係がない．
- 開発者による，開発者のためのプロトタイプ．
- エンドユーザの要求を確認するために行なうのは試作ではない．
- 実現可能性の検討／最適仕様の選択／性能評価などに適用する．

（その他の関連意見）

- モック・アップあるいはブレッドボードと呼び，プロトタイプと明確に区別すべきでないか．（近藤：プロトタイピングに対する考え方と関連する．３参照）
- 主目的によって，さらに確認型と模索（創造あるいは発見）型とにわかれるのではないか．（つまり，分かっている（つもりの）要求を確認することが主目的か，何も分かっていないので（試行錯誤的に）要

求を生み出すことが主目的か．）むしろ試作／試用ではなく，その視点で分類した方がいいのではないか．（田中）

２．２　試用型プロトタイプ

（合意されたと思われるもの）

- 開発者による，開発者と利用者との接点のためのプロトタイプ．
- ユーザ要求の解釈の確認／ユーザの潜在要求の発掘などに適用する．

（その他の関連意見）

- このタイプがプロトタイピングの本質だと思う．（近藤）
- やはり主目的によって，確認型と模索（創造あるいは発見）型とにわかれるのではないか．（田中）

２．３　使い捨て型プロトタイプ／骨格型プロトタイプ

（合意されたと思われるもの）

- 使い捨て型と骨格型は対立する概念であるから，どこで切り分けるかを明確にすればそれで良い．
- 「結果として」使われたか／使われなかったか，あるいは設計に影響を及ぼしたか／及ぼさなかったかについては，この分類基礎とすべきでない．「はじめの目的がどうであったか」を軸にすべきである．
- 使い捨て型は，再利用を「前提」としないプロトタイプ．
- 骨格型は，最終システムの一部となることを「前提」とするプロトタイプ．何らかの変換機能により目的プログラムを作成するケースも含める．
- 使い捨て型は，複数の実現案を早く速く安く作って，設計方針を固める必要がある時などに適用する．
- 骨格型は，基本的な設計方針が決まっていて，細部の調整等をする必要がある時などに適用する．

（その他の関連意見）

- 「骨格型」という表現はあまり適当ではないように思われる．しかし新しい言葉を無闇に取り上げるべきでもないような気もする．（核型の方が適当か？？）
- この分類よりむしろ，見本型か／核型か，手動翻訳が必要か／自動生成できるか，というような分類の方が意味があるのではないか．（近藤）
- 本開発前に行なうか／本開発内で行なうかの分類ではどうか．（田中）
- 使い捨て＝本開発前／骨格＝本開発内というようには分けられない．（塩谷）

２．４　シミュレーション型プロトタイプ／実環境型プロトタイプ

（合意したと思われるもの）

- あいまいになるので，「実環境」というものをなるべく狭義に定義した方がいい．
- 実環境型は，最終システムを実現する環境と同一環境・同一言語を使用するプロトタイプ．
- シミュレーション型は，実環境型でない全てのプロトタイプ．
- 実環境型は，プロトタイピングのための環境が何もないような時，あるいはどうしても実環境でなければ目的が達成されない時などに適用する．
- シミュレーション型は，条件を変えて実験をする／性能面をシミュレートする／独立なプロセスの協調としてモデル化する際などに適用する（この際，統計データの取得も可能）．

３．　プロトタイピングとは何か．

これは今回のもともとのテーマではなかったが，一部言及されたので，簡単に記憶の範囲でまとめておく．誤りがあれば全て田中の責任である．次回以降の討論で訂正願いたい．

（近藤）

ユーザの要求仕様とシステム作成者の設計仕様をつき合わせるための共通の土壌としてのプロトタイピングが本来の定義．

つまり，ユーザの真の要求と作成者の理解との距離が大きいので，これをつめようとするのがプロトタイピングであり，その意味で，開発者が自分のためにやるのはプロトタイピングではなく，ブレッド・ボードのよ

うな違う名前で呼ぶ方が好ましいと考える．

（渡邊）
ユーザの目の前で動かして見せられないようなケースでも，プロトタイピングと呼んでいいのか疑問．特にすぐ代替案が作れるようでないといけないのではないか．
（塩谷）
汎用機の世界でのプロトタイプはなかなかそういう風にはいかない．それでもパイロットシステムなどの作成がプロトタイピングでないとは思えない．ただし，代替案がいくつか用意できないようでは，適当ではないと考える．
（青島）
プロトタイピングというものは，what に注目するもので，代替案云々はむしろ how であるから，それほど拘る必要はないのではないか．ただ，how のなしで what を語るのは自然ではないというだけではないか．
「物」がなければプロトタイピングではないのかというのは面白い問題ではある．

（田中）
今回のメモ（ P03.1-04 ）にあるが，プロトタイピングは手法として「明確な思想」を持っていなくてはならないと思う．その「思想」がプロトタイピングであり，それにかなっていれば「物」がなくてもプロトタイピングだと思う．
その「思想」が何かというのはまだ良くわからないが，その根本は，「ユーザとは打ち合せ／確認を頻繁にしながら物作りをしなければいけない」というごくごく当り前のことではないだろうかと思う．今まではこれが難しかったのであまりやられていなかったが，ハードの進歩などによって次第にできるようになってきたので，「よりうまくやるためには？」というような感じで「技法」として取り上げられるようになってきただけではないか．プロトタイプと同じで，物が出てきたので要求（理論？？）も顕在化し始めたのではないかと思う．もちろんこれから成熟していくだろうが，この意味から手書きの画面イメージをユーザに見せながら打ち合せをするのもプロトタイピングだと思う．

（青島）
プロトタイピングによって扱われる物には，要求と作り方があって，その扱い方にはそれぞれ，

要求	確認する 明確にする 創造する
作り方	確認する 明確にする 創造する

ようなレベルがあるのではないか．このような分け方でもう少し考えてみたい．
（ここの表現に関しては，P03.*-02（塩谷氏の覚書）も参照のこと）

４．　今後の予定，その他
次回は，10 / 6 （木） 18:30 からＳＲＡ 3F で開催予定．内容としては，「プロトタイピングの指針／目的について」，「アプローチについて」，「プロトタイプの評価基準／評価法」などについて検討するものとし，この際，特に３であげた青島氏の提案を軸に考えてみることとする．
また当議事録とは別に，塩谷氏が覚書（P03.*-02）をまとめてくれたので，そちらも参照のこと．
なお，９／３０までに田中まで届いたポジション・ペーパに関しては，事前に全員に再配布する．在宅会員の方を含め，よろしくお願いする．

（記録：田中　慎一郎）

第3回プロトタイピング討論 (9月7日)の覚書

塩谷　和範

言葉の解釈/定義

プロトタイプ

"目的システムの一部または全部を表す模型または原形"

プロトタイピング

"プロトタイプにより要求の確認/引き出しを繰り返し行なうことによって要求者の満足が得られるシステムを作ろうとする手法"

試作(型)プロトタイピング

"実現可能性の検討・最適仕様の選択・性能評価などのために開発者側で行なうプロトタイピング"

試用(型)プロトタイピング

"ユーザ(要求者/使用者)側と開発者側の解釈の違いを埋めるため・ユーザの潜在要求を引き出すためにプロトタイプをユーザに使ってもらうプロトタイピング手法"

使い捨て(型)プロトタイピング

"再利用を前提とせず早く速く安く作るプロトタイピング手法"

骨格(型)プロトタイピング

拡張型(田中氏)/非使い捨て型(近藤氏)プロトタイピングの意味.

"成果物が目的システムの一部となることを前提にして行なうプロトタイピング"

これにはジェネレータなど何らかの変換機能を介して目的プログラムを成果物から生成するケースも含まれる.

実環境(型)プロトタイピング

"目的システムと同一実行環境を使用して行なうプロトタイピング"

シミュレーション(型)プロトタイピング

"統計データの取得・性能要求の検証・さまざまな実験などを行なうためにシミュレータなどを使って実環境を疑似(Simulate)して行なうプロトタイピング"

歩行可能要求仕様をプロトタイプとして実行させることも含む.

インタプリタ(型)プロトタイピング

"インタプリタを使用して統計データの取得などを行なうより柔軟なプロトタイピング"

シミュレーション型に含む.

目的システム

"プロトタイピングを通じて最終的に作成するシステム"

目的プログラム

"目的システムを構成するプログラム"

プロトタイピングアプローチへのアプローチ

塩谷はプロトタイピングアプローチを検討するための切口として次のようなプロトタイピング手法の分類を行なってみた.

視点による分類: 試作型/試用型
手段による分類: 使い捨て型/骨格型(=非使い捨て/拡張型)
環境による分類: 実環境型/シミュレーション型

また,田中氏は試作型/試用型に替えて"確認型/模索型のような言葉を用いたい"と提案し,近藤氏もプロトタイピング支援環境を分類し"大規模開発環境型/道具箱型" 及び従来手法との関連から構造的手法の補助手段型などの分類があることを指摘した.

またすでに田中・渡邉・近藤各氏は,各々ジョンソン・松本の次のようなプロトタイプの分類を紹介していた.

レベルによる分類(田中によるアレンジ): 実物大模型/模擬実行/動く模型/研究開発型
-- J.R.Johnson (日経コンピュータ 1984/2/6)

プロトタイピングパラダイム: シナリオ(紙芝居)型/原版型/機能検証型プロトタイプ
-- 松本 (P02.1-51)

これらの分類及び前記の用語の定義はプロトタイピング手法を分類し,その意味を明らかにしようとする試みであり,これによってプロトタイピング計画を立てる時の指針を得ることを目指した.

しかし全員で討論を重ねていくうちに,試作型/試用型プロトタイピングは対立概念ではないことが近藤氏により指摘され,有澤氏による8種の分類はプロトタイピング手法を選択する際の指針としては有効ではないことがわかった.

そこで,青島氏により"要求の確定法/実現法の確定法"の組合せによる以下のような分類が提案されこれを検討することになった.

要求の確定法: 発掘/明確/確認
実現法の確定法: 創造/明確/確認

なお討論を通じてプロトタイピングアプローチ(計画)・プロトタイプの評価基準/評価法なども重要であることが認識され,上記に加えて検討していくことになった.

また最終的には現実的なプロトタイピング手法・計画・評価基準などを確立していくことが必要であることが認識された.

(しおや　かずのり)

プロトタイプについて

海尻　賢二

まずは言い訳から：参加を表明しておきながらposition paperは全く書かず、おくられてくる資料も忙しさにかまけてほおっておきました。１１月になってようやく読んでみました。

まずプロトタイプには関係ありませんが、環境という言葉について一言述べたいと思います。この１０年程環境をテーマに研究を進めてきましたが、（他の分野もそうかもしれませんが）環境というイメージも大幅に変わって来たようにおもいます。特に現実の大学教育の場にいますと、そのプログラミング環境は大きく変わってきています。

環境についても大学における理想的な環境と、企業に於ける環境とは違うでしょう。(programming-in-the-smallとprogramming-in-the-largeか)当然目的とするところも違います。私の感覚はあくまで大学人としてのものであり、プロトタイプに関するいままでの議論も理解でき、そのような観点から考えていくことも可能ですが、いま一つ実感を持てません。

プロトタイプにつきましてもp03.1-01で近藤さんが否定されているような意味でのものしか行っておりません。我々（私だけかも）が行なっているプロトタイピングとは、「アルゴリスムの現実問題への適用可能性をさぐるもの」とでも言ったらいいのでしょうか。我々はどこからかの要求があってソフトウェアを作るものではありません。ソフトウェア工学上の各種問題点を解決、または改良するためのモデルを考え、その現実問題への適用可能性を（デモとして示せるように）システムを試作（プロトタイプではない）します。

最近は少し（ソフトウェア工学を対象とした）エキスパートシステムにも頭をつっこんでいますが、これなども日本では典型的は試作のみで終っている例ではないでしょうか。ある人はこれを次の様に例えていました。

エキスパートシステムは鳥を作ろうとしている。アメリカは力まかせでも鳥にみえるものを作る。しかし日本では Man Power がないため最初から飛ばないことが分かっている鳥を作って満足していると。

けっして飛ばない鳥で満足しているわけではないのですが、色々な要素から飛ぶ鳥に発展させる余裕が無いわけです。

いままで行なわれている議論とほとんど関係のないはなし申し訳ありません。１１月の神戸でのワークショップには参加の予定ですので、全くなにも書かないと無視されるかとおもい、感想を書きました。

今後はできるだけコメント／意見等を出していきたいと思いますので、宜しくお願い致します。

（かいじり　けんじ：信州大学）

プロトタイピング討論会への論点の整理

渡邉　雄一

０．おことわり

この資料を早急に作ると宣言したにもかかわらず，たいへん遅れて申し訳ありません。一応，ＳＩＧＥＮＶ＠神戸のプロトタイピング討論会に向けて，論点をもう１度整理するべく今までの資料をひっくり返してまとめてようと試みてみましたが，あえなく失敗しました。その罪滅ぼしというわけで，この間紹介した有澤［１］から関連しそうなことを拾い集めて私見を述べることとします。

１．有澤の主張のその後

前回の資料にも述べたが，有澤［１］が出版されていた。それから我々の検討内容に密接に関係している部分を抜き書きする形で５．１のみを（私見も含めて）整理してみた。（〔〕は渡邉が挿入した）

§５．１　ソフトウェアプロトタイピングの背景

1) 構造的な手法の欠点

1 ライフサイクルの発送はソフトウェア開発管理に都合の良いものであり，開発自体〔特に生産性の向上〕に寄与していない

2 ライフサイクルの考え方では，ユーザと開発者たちのコミュニケーションの機会が要求分析の段階の初期から，受け入れテストの段階までの間，皆無に近い。（中略）その結果，ユーザの真の要求を把握しないままでソフトウェア開発が進んでしまう恐れがある。

3 ライフライクル方式は純粋にトップダウンであるため，重要な判断を早い時点で下す必要があり，いったん高い水準で下した判断は，もし低い水準で不都合が生じても，修正することができないか，あるいは後戻りして修正するため時間と労力がかかる。

4 （補足）ユーザの要求は要求書に書いてある通りとは限らない。またユーザの要求は時間と共に変化していく。ライフライクルの考え方では，こうしたユーザの要求変化に追従することは極めて困難である。これは２3の両方に関係しており，構造的なソフトウェア開発の最大の欠点である。

2) プロトタイプに関して

プロトタイプはシステムのもつ主要な機能だけを果たせばよく，データのエラーチェックや例外処理などは省いてもよい。また実行速度については問題にせず，システム容量も実際より小さくてよい。いわば目標とするシステムのミニチュア版がプロトタイプである。

〔既存の工業製品での〕試作品は開発者の経験不足から生じる困難を解消することが目的である。（後略）

（前略）ソフトウェアでも試作品作成の段階を明確に含めることにしようと考えたわけである。ここで作成した試作品がプロトタイプ(prototype) であり，これは最終製品を作る際には捨ててしまう．いわば使い捨て，あるいは作り捨て(throw away)のプロトタイプである。〔試作品＝使い捨て〕

☆試作品プロトタイプ：

- あまり時間や労力をかけず，短期間に速く作成すべき．長時間を費やすことは得策ではない
- rapid prototyping
- quick prototyping
- しかし〔時間を気にしていて〕あまりいいかげんな試作品では，それを作ったことの意義が薄れてしまうため，適当なバランスが必要である。〔このための判断は逆に経験や勘によってしまう？！〕
- いかにしてソフトウェアを作ればよいかを調べるためのもの

☆試用品プロトタイプ：

試作品プロトタイプを作ったとき，どのみち部分的な設計変更を行うのであるから，そのプロトタイプをユーザに試用してもらって，ユーザからのフィードバックも含めた設計変更にすることを意図した

- ソフトウェアの開発の比較的早期に行う必要がある。ユーザの潜在要求を設計に反映することがねらいである。
- early prototyping

・はやさの目安は，時期として２カ月以内，期間としては３週間いないといったところ 〔ここら当たりは現場の状況と比較検討したい！〕
・現実のプロトタイプによっては，試作品でもあり，また試用品としても役立てるという場合もある。〔試作品と試用品は対立する概念ではない〕
・試用品としてのプロトタイプは，必ずしも使い捨てとは限らない。→骨格型

☆骨格プロトタイプ：
プロトタイプを核としてそれに肉付けして，最終製品を仕上げることも可能である。このようなプロトタイプを骨格プロトタイプあるいは単に骨格型とよぶ。
試作品プロトタイプが骨格型であることはほとんど考えられないが，試用品プロトタイプでは使い捨て型と骨格型のどちらもありうる。

☆その他のプロトタイプ
・ユーザとのインタフェースを適切に設計すの手段に限定したもの
・「言い訳プロトタイプ」：逃げ口上にプロトタイプという言葉をつかっている場合．いいかげんなソフトウェア

3）プロトタイプの実現方法について
試作型 →実環境かそれに近い状態でプロトタイプを作成
試用型 →まったく異なった環境に構築してもかまわない
作り捨て→最終製品と独立な環境でも作成可能
骨格型 →最終製品と同じ環境か，それに近い環境で作成する必要あり

補足）
ソフトウェア開発のひとつの段階としてのプロトタイピングは，試作品としてのプロトタイプよりも，試用品としてのプロトタイプのほうがずっと意義が大きい。構造的なソフトウェア開発，あるいはライフライクルモデルでの欠陥である，ユーザの要求を的確に把握することのために，試用品プロトタイプは有力な方法である。これによってユーザからのフィードバックをソフトウェア開発段階に吸収できる。

私の感想：
前々回に議論し確認されたことが，簡潔な文章でより整理されていると思う。一部，頁数の関係のからか，シミュレーション型や実環境型についてはプロトタイピングの実現という観点から数行述べてある程度で，（有澤自身も）特に重要な分類といて固執してはいないと思う。
我々の討論で指摘されていたことは，いずれもこれら主張を補足する上での，重要な意味付けを与えるものであろう。だが，分類ということでは，理解するためのものであり，「プロトタイピングの指針／目的」「アプローチ」「プロトタイプの評価基準／評価法」といった前々回の議事録の検討課題はそのまま放置されている。

そこで：
今回は特にＰＷＢがtool展示であるので、塩谷さんにまずＰＷＢの狙いを簡単に紹介していただいたあとで、ＰＷＢなどのプロトタイプツールとしての印象を調べることを意図してアンケート用紙を作成することを主たる話題にしたいと思います。特に、ＰＷＢを作成している立場からの「狙い」「期待する効果」が、初めてそのようなツールに接した人にも正しく受け入れられるであろうか？、また良い意味で正しい評価を受けてもらうために工夫するようなことは何かあるのか？、さらに他の様々な環境で仕事をしている人の考えているプロトタイピングやツールとこのＰＷＢは方向性・考え方に大きなズレなどはないであろうか？といったことが得られるような問いかけができれば最高なのですが、、、:-)

【参考文献】
［１］有澤誠：ソフトウェア工学，岩波コンピュータサイエンス，岩波書店，１９８８
［２］田中：環境分科会議事録（第３回プロトタイピング討論） Ｐ０３．＊－０１
［３］塩谷：第３回プロトタイピング討論（９月７日）の覚書 Ｐ０３．＊－０２

（わたなべ ゆういち：電力計算センター）

第５回プロトタイピング討論 議事録

－ＳＩＧＥＮＶ@神戸－

日　時：１９８８年１１月２３日（水）　１８：３０～２０：００
テーマ：プロトタイプとは？
場　所：元町のうどんスキの中浜で！
出席者（敬称略）：塩谷，田中（慎）〔ＳＲＡ〕，熊谷〔ＰＦＵ〕，藤野〔ＦＸＩＳ〕，
井川〔ＪＩＰ〕，渡邉〔ＤＣＣ〕ほか多数

１．はじめに

今回は，予定通りＳＩＧＥＮＶ@神戸ということで，いろいろな方と意見交換することが出来た。また環境ワークショップではツール展示もあり，別の意味で刺激的であった。結論から言えば「プロトタイピングは一筋縄にはいかないということが，色々な人の意見から分かった」ということになろうかと思う。以下では，記録係を名乗り出た渡邉の可能な範囲で印象的なことを書き綴る。意見・解釈の食い違いなどもあろうが，取り敢えずここにまとめたので今後の意見交換のネタにでもなれば幸いである。なお会合はインフォーマルな形式で行われたこと，話題の傾向がそれぞれにおいて微妙に異なっていることなどから主な発言者の名前を勝手に掲げさせて頂いたことをお断りしておく。

［第 5 回討論となっておりますが，やはり間に一回プロトタイピング事例報告が入りましたので，実質討論としては 3 回目になります．これら 2 回の事例報告は，次回「プロトタイピング特集号（その２）」で紹介させていただきます．（田中）］

２．プロトタイピングとは？

何故か鍋の具もほぼ片づいてきて，最後のうどんを入れようかという頃になってきて，突然（いや必然的に？）『プロトタイピングとは何か？』という議論が始まってしまった。只テープを回すだけで（酔った勢いで）議論に参加してしまった記録係としては，お姐さんの声や鍋をつつく人を声の彼方から辛ろうじて聞こえてくる主役〔その実態は酒の席に似つかわぬ脇役？〕の発言を苦労しておこすこととなった。結局大人数の（しかも酒が十分にはいった！）討論会の記録を取ろうなどということは無謀なことであり，それにアプローチすることは悲惨であることがつくづく身に滲みた．それにしても神戸訛りのお姐さん声が中心的に録音されていたので，本当に悲惨だった．

まずＳＩＧＥＮＶの今までの活動経緯〔特に有澤先生の本のことや言葉の定義を中心にプロトタイプについて考え直してきたことなど〕を田中さんが説明した．その後，参加者の１人の藤野さんから「プロトタイピングは，まずその判断のための基準(criteria)をはっきり決めるべきである」という発言があり，以下ではそれについて議論している．みんなかなりお酒が入っているので，辛ろうじて討論内容が聞き取れる所から・・・

田中：（意見の違いを明らかにしてから議論をするために）なるべく共通な言葉で話さないと，同じつもりで話していることが（実際には）違うことすら気付かないで話していることになるから，その“共通語”の１つとして有澤先生の本を読むことで，議論を詰めてきた
藤野：ベースがもともと違うのに（議論しても噛み合わないのでは？）
田中：だからそれを可能にするために，個々の言葉について議論してみましょうという・・・
渡邉：だからその言葉について，現場の人間が普段仕事をしながら考えているというか感じている気持ち（解釈）と，大学の人間である有澤先生などがイメージしていることとにズレがあるわけだから・・・
田中：だから（そのような）つまらないような所から，やっていくしかないじゃないですか？
渡邉：例えばプロトタイピングということについても色々な解釈があって，それをモックアップと呼べば良いという人もいるし・・・
藤野：モックアップと言う場合は，オペレーショナルでないんじゃない
田中：だとすると，今，藤野さんはプロトタイプをオペレーショナルなものだと思っている訳でしょ．僕は必ずしもそういう場合だけとは思わないもの
藤野：エェ～！．オペレーショナルでないプロトタイピングなんて・・・
田中：僕の主張は，プロトタイピングという言葉は，現象はどうであれ，その背後にプロトタイピングとしての何らかの思想があるか否かではないかということ．だからその思想がどんなものかを考えてみたいと思っている．でも，実際に行なわれる際の表現形はそれぞれに違うものだと思う

渡邉：オペレーショナルな情報をもっていなければいけない場合もあるかもしれないが，単にＭＭＩの情報を確認出来れば良いと思っている場合も少なくない
藤野：でもそれはオペレーショナルだよ！．だって少なくとも，ある（ユーザからの入力といった）反応に対して・・・
渡邉：それをオペレーショナルなアプローチとして，そのプロトタイプをインプリメントするかというのは，二の次の話でしょ！
藤野：えっ！　違うじゃない
田中：そうすると，今度は“オペレーショナル”という言葉の定義付けが違っているんですよ
藤野：オペレーショナルということは，操作出来るということで・・・
渡邉：だからそこまで進歩的な環境でなくとも，ある程度生産性に寄与するような（いわゆるモックアップ的な）アプローチは可能で・・・
藤野：それは（オペレーショナルでないからプロトタイプとして扱うのは）おかしいョ～！
田中：じゃあ，例えば事務系のアプリケーションで，帳票なり画面なりに対して「これで良いですか？」といって，無意味なことかと言えば，そうじゃないでしょ？．そしてそれをプロトタイプと考えちゃいけないか？といったら，非常に微妙なところでしょ？
藤野：だってそれは「貴方はこういう操作をしたら，これこれになりますが良いですか？」と言葉を使って（操作して）いる訳でしょ？
田中：だってそこまで考えて，オペレーショナルという言葉を使っていれば話は別ですけれども
井川：それは（２人の間で考えている）オペレーショナルと言っている意味が違うよ！
田中：そこまで言ってしまったら，オペレーショナルという言葉を使う必要性が全然ないんじゃない？
藤野：エッ，どうして？
渡邉：通常オペレーショナルと言った場合は，ＪＳＤのような記述言語があって，それによって操作されているというような状況を考えたくならない？
田中：どっちかというと，フォーマルスペック的な方向に行っちゃうでしょ，オペレーショナルという言葉を使うと
藤野：違うよ．オペレーショナルというのはマニュピュレータと同じで，貴方はこうやったから（システムの動きは）こうなるのヨと教えてあげる・・・
田中：だから，その言葉はどういう意味なのかなぁということを考えてみないと，話にならないでしょ，といっている訳なのですよ
渡邉：例えば画面のＩ／Ｏの確認の為に，ダミーのデータで良いからインプットを与えて，そのリアクションが必ずしもあるとは限らないわけですよ
藤野：でも，どういう形ですらアウトプットがあるわけでしょ，リアクションという言葉を使うからには．だって，入力があって何も状態が変化しないというのは，異常なわけでしょ
渡邉：だから，ただ単に画面のデザインだけして，項目の位置がどうのこうのという場合もあるでしょ
藤野：いや～，それはモックアップで（オペレーショナルなものでは無いから，それについては）プロトタイプと呼ぶべきじゃないよ
渡邉：いや，だからモックアップも何も一緒にして「プロトタイプ」という言葉が走っちゃっているんだ
田中：僕が思うに，プロトタイプとプロトタイピングは違うわけだし
藤野：いや，少なくともソフトウェアの場合は，プロトタイプとプロトタイピングは殆ど一致しているでしょ．ユーザから見ればプロトタイプがあって，それを作ることがプロトタイピングでしょ
田中：いや，作ることがプロトタイピングではないと思うんですよ
藤野：僕に言わせると，プロトタイピングは，（何からの明確な）criteriaがあって，その実現方法が分からなかったり，あるいはそれをやった時の効果が分からない，という場合に，見えない部分があるから，この道はどう行ったら良いのかなぁ，本当に（その先へ）行けるのかどうか分からないから，懐中電灯を照らしながら行って，駄目だったら戻りましょう，良かったらその道を行きましょうというのがプロトタイピングだ．その道が抜けられるか抜けられないかの判断は，criteriaがはっきり定まっているから出来る話なのです
渡邉：いや，そういう場合もあるが，（一方で）ユーザの仕様が不明確でそれらを確定させる為にやるプロトタイプもあるでしょ
藤野：それも同じで，ユーザの仕様を確認するcriteriaがはっきり定まっているか否かの違いでしかない
田中：それは良く分かります．でも一緒くたにしてまずい事って，いっぱいあるでしょ
渡邉：実現方法は既に分かっているが，ＭＭＩや，ちょっとした画面の推移が確定出来ないから，そのような

ものを確定させるためには・・・
藤野：ユーザとか自分たちとか分けるからおかしくなるんだよ
田中：え～，分けなければいけない所って多分にあるでしょ
藤野：違うよ～．だってプロトタイピングというのはcriteriaを設定して（プロトタイプ作成の）作業を進めるだけでしょ．それに対して評価をしながらチェックするということがプロトタイピングでしょ
田中：それはあまりにも自分中心的な考え方ではないですか？．自分が分かっていることが，全て正しい情報として，分析作業を進めることは凄く危険ではないです？
藤野：自分の道が分からないときに，西も東も分からない時に，ただそこに黙って立っていても一行に問題の解決にはならない．だから場当たり的かもしれないが，しょうがないから行ってみて「あ～ダメだった．じゃあ戻って別の道を進もう」と進むべき道を探すしかない．だから最初からcriteriaすら分からないときにも，１つの仮想的criteria を設けてそっちにぶつかってみようとする
田中：それがないときには，プロトタイピングと言うのは多分あまりにも間違っていると思う
藤野：僕に言わせれば，プロトタイピングというのはたった１つで，criteriaの設定をきちんとしているか？，それに対する評価をきちんとフィードバックしているかということだけが問題なのだ
田中：その際にそれをどう設定するか？というのが出てこないとまずい．方法論なり何なりが出てこないとエンジニアリングという話に乗っからない
藤野：そう，それは大切よ！．criteriaの設定が大切で，僕が常々不満に思っているのは，プロトタイピングについて，criteriaの設定について日本の場合にはほとんど皆な考えていないから嫌いなのだ
田中：そこは（藤野さんの意見に）全く問題がないのだ
藤野：そのフィードバックのメカニズムが重要で，どの程度チェックしてどうのこうのということをしないで，また最初からcriteriaの設定すらしないで「これはプロトタイピングだ」などと言うから嫌いなのだ
田中：（有澤先生の言うところの）「言い訳プロトタイピング」ね（笑）
藤野：criteriaは変化しても構わないわけネ．一番初めの段階では自分のcriteriaはここでした．でも次の段階ではこう移行しましたとはっきりさせれば問題ない
田中：そこは全く反論がないし，僕もそう思う
藤野：僕に言わせれば，ユーザだとかユーザの仕様の確認するためだとか，自分がアルゴリズムが分からなくて良いアルゴリズムを試すためだとか・・・
田中：でもこういう場合があるでしょ．ユーザ要求ははっきりしていて，ユーザ要求というのは早い話，外部仕様だから，こうなれば良いというのが書いてあったとして，それをどう実現するか？というのが分からなくてやるプロトタイピングというのがあるでしょ
藤野：それは，どう実現したら良いか分からないという，やり方を試してみたいということで，それはcriteriaの質が違うだけで・・・
田中：だからその質が違うから，その質の違いに着目しなければいけないのでは，という議論をしている
藤野：僕は，前もってそのようなcriteriaをきちんと設定しないこと自体が問題と主張している
田中：それは問題ですヨ．それは当然のようにしてです
藤野：でも，今まで（世間一般では）きちんと議論されていないでしょ
田中：それは（ＳＩＧＥＮＶの議論では）早い段階から批判として指摘されている．そんなものをプロトタイピングと呼んで，学問的な扱いをするのは間違っていると
藤野：ユーザ要求が明確でないとか，自分のインプリメンテーションのためにどうのこうのということは問題ではなくて・・・．例えばデータタイプが違うとかタイプの問題を議論しているようなレベルであって，それよりもっとデータタイプがあるとか必要だとかいう議論が重要なのと同じ・・・
田中：いや，そこまで認めたとして「あー，そうですね，良かったですね」で済ませる訳にいかなでしょ，という話をしている
藤野：だから僕は「そのcriteriaをどうやって設定するか？」といったオペレーションするためのとのためのメソドロジー（方法論）というかフレームワーク（枠組み）みたいなものを考える方が先じゃないかと思う
田中：だからそのことを議論しようとしているんですヨ
藤野：な～んだ．だったら同じじゃない！（笑）
田中：すごく感覚的に直観的にそういう風にやった方が良いにきまっている訳で，ただだからそれをあまりにもいい加減にやりすぎたり，言い訳的にこれはあまり考えていないからプロトタイプなんだとかいう行き方をしているのが現状だから，それは間違っていると思う
藤野：だからそこを「あんまり考えていなから」といってゴチャゴチャにしちゃうからイケナイ．自分の考え

がまとまらないことを，キチンとまとめるためにその方向づけをするために，１回は作ってみないと何が良くと何が悪いのか確かに分からないというのは（現実問題として）確かにあるんだもの

田中：それがはっきりしていれば良いんですヨ．分かっていないからやってみて・・・

渡邉：研究開発的な現場では，そういうことは許されるだろうけれども，例えば実務的にあるものをインプリメントしなければいけないと言う場合には，既に予算だとか納期だとかが決まっていてある部分で妥協しなければいけないから，ユーザから全部ニーズを言われても，ある部分では技術的に実現不可能であるから削らなければいけない．そういうような制限を，もし理想のプロトタイプツールがあったなら，インプリメンテーションする時の実現性の条件として埋め込んでおいてやれば・・・

藤野：でも，それはプロトタイピングとは関係無くて，ただ設計とかrequirement をどれだけ実システムとどれだけ近いところでやるのか？という，トレードオフをサポートするツールの話じゃないの？

田中：本当はそうなりますねェ．だけど・・・

藤野：それじゃプロトタイピングというのとは，やっぱり違って・・・

田中：でも，両方とも考えなきゃいけないことじゃないですか？，現場に導入しようとしたら

藤野：あくまでもトレードオフのディシジョン（判断）というのは，設計者のrequiremen tation（要求分析）の（作り出した）仕様として設計者が責任をとるべきであって，その為に（システム設計者が）自分がシミュレーションが必要だと思う人は，そのプロトタイプ使うか否かを判断すれば良い訳であって，それ自身をプロトタイプの重要な要素と考えるのはおかしいのでは・・・

田中：勿論（そうだが），考えなきゃいけないことが一方向から考えていて宙に浮いてしまうような話ばかりしていてもしょうがない，ということを言っている訳ですよ

井川：それはでも「誰かのrequirement はいつも明確なものである」という前提に立っていません？．現場から言えば「ユーザからのrequirement というものは，ほとんどいつも不明確なもの」なんですよねェ．だからインプリメンテーションをユーザのＳＥがするようなものはプロトタイピングでないという意見だと思いますが，それだったらばユーザのrequirement は明確であれば確かにそうですけれど・・・

藤野：僕の言っているのはそうじゃなくて，トレードオフの責任というのは設計者にあるのであって・・・

田中：（藤野さんの言わんとしていることは）わかりますヨ．でも，それ（真の意味のプロトタイプツール）を本当に使えるものとして，プロトタイピングの概念や方法論としてしっかりしたものを実際の開発の現場に持ち込もうとした時には，多分問題になるのは一番はそこ（トレードオフの責任を如何に軽減することを併せて支援するということ）でしょ．そこをサポートする機能なりなんなりを（真の意味のプロトタイピング支援機能と）いっしょにサポートしなければ導入しきれないでしょ！

藤野：そうするとさぁ，あなた方の言っているのは，requirementを作るときだけの，requirement-spec を作るdeveloper のトレードオフをサポートする為だけのツールとしてしか位置付けていないじゃない？

田中：要求仕様を作るために，プロトタイピングをすることは意味が無いと思いますよ．『そこだけ』を考えるべきだなんていっているわけではないですよ．

藤野：そうじゃなくて，requirement-spec を作る時に developer がトレードオフをディシジョン（判断）しなくちゃいけない訳でしょ，そのサポートツールとしてとか捕らえていないんじゃないかと・・・

田中：「そういう場合もありますよ」「そういうことも考えた方が良いですよ」といっているんですよ．「そこ（トレードオフの判断材料）は関係ない話ですよ」と完全に切離してしまったら，（プロトタイピングツールとしては）いくら良いものでも結局使えないものなのではないだろうか？

藤野：だったら，そういうふうにライフサイクルで言うところの，どのフェーズにいるのかによってプロトタイピングのタイプが違うということならば，そのライフサイクルモデルと，対象であるのはユーザなのか自分たちであるのか，即ちインターナル（な利用）を考えているのか，より広範囲なものなのかを明確にしなければならないのでは？

田中：それはそうだといっているんですよ？！

藤野：でも，そんな明確には言っていなかったじゃない．だったら，ライフサイクルと各プロトタイプのタイプの間にどんな相関があるのか明確になっていないじゃない（だからするべきじゃない？）．少なくともrequirementのレベルで一番多いのは，トレードオフよりはシステムの分析してシステムを明確にすることのほうでしょ

田中：そこはまたチョット違いますよ

藤野：エェ～，どうして？

渡邉：だってオンラインのシステムなどでは，実現方法はほとんど確定しているけれども，そこでのＭＭＩなり画面の推移なり使い勝手を試そうと思ったらば，あくまで画面推移だけのプロトタイプの簡単なツールがあって，それで仕様が決まったならばゴリゴリ作りだせば良いじゃないという話があるじゃない．

そこでの実現性のための環境はＩ社のｘｘｘとかＦ社のｙｙｙとか
藤野：えっ，そんな．それは仕様が明確になっているんじゃない．Ｆ社のｙｙｙを使うとか，ＤＢ／ＤＣを使うとか・・・
渡邉：でも現場では・・・
藤野：でもそれは，そこまでｓｐｅｃが明確になっているんだから
渡邉：でも，そういう所でもＭＭＩの誤差は十分解消できる訳で・・・
藤野：それはおかしいヨ！
渡邉：おかしいといったら（議論は）始まらないんで，そういう物をプロトタイプと言う人もいるわけだよ
藤野：それは構わない．それをそう呼ぶことに関しては．それには問題ないわけで，何故そのフェーズに限ってしまうの？
田中：限ってないですよ．「それも考えなきゃいけない」「それも排除すべきではない」と言っているんです
藤野：さっきから言っているようにタイプ分けをちゃんとはっきりすべきじゃないの？
田中：（我々も）「はっきりすべきだ」という主張をしているだけですよ・・・
藤野：だっておかしいじゃない．さっき言っていた話では「ユーザの要求が明確でないためにrequirement-specを定義する場合で，そのトレードオフを決定する為に必要だ」と言っている訳でしょ？
田中：（その場合だけに固定している訳ではないという意味で）そこは違うじゃない
熊谷：そういう場合“も”あるんだ
田中：ユーザの要求をいっくら明確にしても，たとえユーザが本当にやりたいことはこうですよと明確になったとしても「これを実現するのにいくら機械がかかってどれくらい金がかかりますよ」といったら，ユーザはそんなものを作ってもらっても困るという場合もあるでしょう
藤野：そういうタイプのものがあって，もう一つはインターナルにデザイナー（システム設計者）が（システムの）デザイン・デシジョンする時に，トレードオフをするために自分自身のデシジョン決定の必要なもの（プロトタイプツール）があるでしょ！
田中：そういうものも“ある”という話をしているんです．“ある”はずなのに，そこを明確にしないで，プロトタイピングなりプロトタイプなりみたいな話で言ってしまっているからいけない．プロトタイプと言われているものの中にも（様々あって），プロトタイピングの検討にしっかり沿ったものと，単にそれ以前に「データ収集をしたいからその（目的に合うような）テストツールみたいなものを作ってそれを使ってデータを取りたいよ」というような物も，今ではプロトタイピングと呼んじゃっている訳ですヨ（だから問題だと思っている）
藤野：少なくとも，さっき言ったように形式（タイプ）を明確に分けるつもりであれば，そのフェーズみたいな形で考えていくか，あと１つはそのプロトタイプを使って，その結果(output)を必要とする人は誰なのか？ということをはっきりさせるべきではないか？
田中：現状を言うと，それ（藤野さんの指摘した場合分け）を考えるときに「自分の言葉だけで話をしていても，話が通じなくなるから１つあの有澤先生の（本で述べられている）言葉から入ってみよう」と進めてきた．それに対して一応，その言葉使いなりおかしな（矛盾）点が整理出来たところで，それと今の（藤野さんの指摘してくれたような本質的な問題）ところまで辿り着いていないのですよ．だから（その方向で）やらなきゃいけないという話は出ていて，一歩一歩それに向かって進んでいる段階だと思って頂ければ良い
渡邉：だから今，藤野さんの話してくれたような（本質的な）プロトタイプの話をもっと聞きたい訳なのですよ
田中：ただそこを一気に解決出来ないし・・・．また（有澤先生の本は）そういう視点でものを見ていないんですよネ！
渡邉：あれは研究者の立場で，「これは試作型」だとか「試用型」だとか言っているにしか過ぎなくて，（目的と対するプロトタイプを）作る前の段階でそれが分かったら，それは理想な訳だけれど，それを分かるためにはどうしたら良いのかといった「工学的」な裏付けが（まだ現状では）無いわけなのです
田中：だから今出ている結論の１つとしては，（有澤先生の本のプロトタイプに関する）分類の仕方というのは，「プロトタイプとは何なのか？，どういう物なのか？」ということをイメージを持つ為にはとっても良いものではあるけれども・・・
渡邉：だからまったくプロトタイピングを理解していない人に「ああゆう考え方もあります」「こうゆう考え方もあります」と説明する場合は良いのだけれども，それらを予備知識として持った人か，これからプロトタイピングをどう進めていったら良いのかと考える際に，また実践的にプロトタイピングを進めて行くうえで「使い分けて」いくための場合分けでは無いことがどうもはっきりした訳

田中：だから色々な（場合分けに関する）言葉があるのだけれども，それぞれがきっちりと（明確に）切れているわけでは無くて，微妙に重複している・・・

藤野：僕にはどうも，みんなが話していることが，どうもプロトタイプを本末転倒をしているように思える．何故なら“プロトタイピング”というのはあくまでも『手段』でしょ

渡邉：プロトタイピングは手段ですね

藤野：プロトタイプというのは，プロトタイピングするためのアウトプットでしょ？

渡邉：アウトプットというか，必ずしもアウトプットだけではなくて・・・

藤野：（それは）嘘だよ〜．（何故なら）「プロトタイピングする」ということは，さっきも言ったように，criteriaと比較評価して，そのフィードバックで新しい情報を二次的に得なくちゃならないんだ．それを確定するための行為で・・・

渡邉：そういう意味でアウトプットだけれども，そのアウトプットが，常にその実際の本番システムと作成作業と・・・

〔注：ここで“アウトプット”という言葉に関して誤解が発生していることは，賢明な読者諸兄はもうお気付きであろう．藤野さんは，プロトタイピングという１つのプロセスで得られる情報を総称して言っているのに対し，渡邉の方は１つの帳票のような物のイメージを持って話をしている〕

藤野：だからそれがおかしいんだって．プロトタイプとツールと一緒に考えるからおかしいんだって．もっと大切なのはデザイナというシステムディベロップメントを進めていく人が，どこでどういうトレードオフの判断（ディシジョン）をすべきか？という話なんだよねぇ．一番大切なのは，デザイナがシステム開発の際に「どこでどういう判断をしたか？」ということであって，その判断をサポートするためのツールとしてプロトタイピングとかプロトタイプを使ってフィードバックの情報を得る，という，つまり判断をするためのより良い確実な情報が欲しくて，そのプロトタイピングを使うか使わないかも，判断としてはまず開発者（ディベロッパ）の，まず最初のメタな判断な訳なんだよねぇ．（システムを設定していく上での）情報が足りないから，この情報はプロトタイピングをすれば得られるのじゃあないか？と思って働き掛けるのであって・・・．そのよう動機付け（モチベイション）が先にあって，その動機付けがあるということは，さきにこうしたいという評価基準が先にあって（プロトタイピングということを）やっているはずなんだよネ

田中：仮のものでも（あらかじめ評価の対象として）置かなければ意味が無いと思いますよ．それがなければいけないというのは私はそう思うし，皆もそれで問題ないんじゃないですか？

藤野：そしたなら，その方向でずっと考えていけば，プロトタイピングが「〜型」というのは殆ど関係ないんじゃない？

田中：それは，そう納得してしまったら，何にもならないんじゃない．つまり藤野さんの言ったことを，何からの形で（より具体的な）モデル化をしなければ何にもならないでしょ

藤野：僕がもしモデル化するのであれば，今言ったようにそういう形で・・・

田中：「そういう形」というのは，それは抽象的過ぎるじゃない

藤野：そうでもないじゃない．さっき言ったようにフェーズで（プロトタイプの）タイプ分けをすれば良いじゃない

田中：だからそういうことを言っているんです．だからどういう風に分けたら良いのか？ということを（議論して）やっている段階なのですよ

藤野：少なくともウォータフォール・モデルが良いにしろ悪いにしろ，ソフトウェアのライフサイクルが繰り返されるように各工程が繰り返されることを前提としたとしても，少なくとも，『要求（requirement）の確認をしたい』のか，それとも『設計段階での実現の為に，どっちの方向に行ったら良いのか分からなくて（プロトタイピングを）やる』というのと２通りがあるよねェ．それをウォータフォール・モデルのように，最初から固い殻のような枠組みで考えるのではなくて，繰り返しをすることによって・・・

田中：だからそういうことを言っているつもりなんですが，どこが違うのですか？．だから（議論が）そこまで至っていないだけで，本当にやりたいのはそこ（藤野さんがいったようなことをきちんと整理すること）なんですよネ

藤野：だからやれば（笑い）

田中：そう言っても，いきなり「やろう」といっても出来ないでしょ

藤野：それだったら，プロトタイピングの本なんか読まないで，もっと自分たちでフェーズとはどういう物なのか，どこでどういう（プロトタイピングに関する）一般的な捕らえ方をしているのか，どういうシステム設計の判断が行われているのかを・・・

井川：それはやっぱり（議論として）偏りすぎますよ
藤野：えぇ，そう？
井川：やっぱり自分たちとは距離をおいたものをサンプルに取り上げるなら，それは正攻法だと思いますが，「自分たちが論じているものが何か？」ということを客観的に見るということは，自分達と距離のあるものを評価をして，そしてそれに対して自分の立場（スタンス）を明確にすることが出来るかということだと思いますが
藤野：でも（さきほどまで自分の主張していたことが）大切だということは，はっきりさせるべきだと思いますよ
田中：だから一つには，先ほどいったアンケートを配ってみようというのは，もろ現場の方から（プロトタイピングについて）攻めてみたいということなのです．またそれとは別に（ソフトウェア工学などの多少理論的な）お話のほうから攻めたいネという状況なのです．だからやれば良いという（藤野さんの主張や）分からないわけではないが，そこを考えていくと，いきなり沢山のことを考えなければならなくなって，収拾がつかないでしょ．やっぱり１つづつ，１つづつ，積み重ねていかなければならない訳だから，どの辺から当たりをつけていくか？という話なわけです
藤野：だから，さっき言ったようにフェーズ分けをして，トレードオフとか，そのディシジョンされている所に焦点をあてて
田中：そうすると，その（そのような議論のための）叩き台が簡単に作れるか？というと凄く難しいでしょ？「誰かチョットやってみてヨ！」などと気軽に言えないでしょ．そうするとそこからはやれないことになってしまうのですよ．そうすると，いままで研究者が既にやっていた所から入っていくのは，取り合えずのとっかかりとして仕方のないことだと思うんです．やっぱり基礎の無い所に，いきなり物を建てようとしてもどうしようも無いでしょ？．で，感覚的に「そうだなぁ」と思っても，じゃぁ何処からとっかかりを付けようとしても，どうしようも無いのですよ
藤野：ソフトウェア・エンジニアリングの無い所で，プロトタイピングを論じても何も意味が無いと思うけど・・・
田中：でも今の日本の現状って，まさしくその通りでしょ
藤野：やっぱりそうか！（全員笑い）
田中：だって，「プロトタイピング」とか「プロトタイプ」とか言う場合，殆どの場合は適当に作ってしまったから「これはプロトタイプだ」と言うのがほとんどでしょ
渡邉：プログラムを作ったことの無い人が管理職をやっている（現場の）現状からするとそれは当たり前でしょ（笑い）

［かくして議論（？？）は続き，このあと近くのピザ屋，田中の部屋，と渡りあるいたのですが，話は一向に収束しません．翌日藤野さんは風邪をひいてしまうし．．　いやあ，どうも済みませんでした．でも藤野さん，この話どこかでまとめて話してくれません？？（田中）］
［因みにここの店の勘定はワリカンで４１００円也でした（渡邉）］

３．おわりにかえて

この記録がどの程度の出来であるかについては，それぞれのミーティングに参加していた人の評価を得なければならないが，決して十分な記録が出来ているわけではないと深く反省している．どうも酒もはいる所での討論会には，絶対にテープを持ち込まなければダメそうである．（酒を飲まずに記録などしているのが馬鹿らしくなるし，またそのような場のほうが参加者の本音の意見が早口でバシバシと出てきて面白いからつい自分も参加してしまうので・・・）

（記録：渡邉　雄一）

ＳＩＧＥＮＶ－Ｆｏｒｕｍ@ＭＯＲＩＯＫＡ顛末記

世話人代表：　田中　慎一郎

１．　はじめに

去る３月４日（土），分科会レベルではおそらく初めてと思われる『ミニ・フォーラムの地方開催』という無謀な試みを行なってしまいました．どのくらい無謀であったかというと，関係者みんなが『実にいい経験をした』と思えるほどの無謀さでありました．この『いい経験』を記録に残しておかないと次回（あるいは他の分科会の）ためにならないということで，フォーラムの本質とは関係のないところを中心に，ことの顛末を書き記してみます．

２．　企画について

そもそもこの企画は，『スキーにいきたい』という非常に『純粋な』動機に端を発するものでありました．つまり，「どうせこのメンバでスキーにいくんなら，ついでになんかやってみてもいいんじゃない」という感じで始まってしまったわけです．この発言自体は，はるか昭和の昔から出ていた話ではあったのですが，具体化に向けてことを起こしたのはなんと平成元年も２月にはいってからでありました．

企画が遅かったのですから，開催日程も遅くすれば良さそうなものなのですが，そこはそれ，『今年は暖冬』でありますから，『スキーに行く』という第一の行動原理に矛盾をきたすわけです．さて，そんなこんなで『各人の都合がつく範囲でもっとも早い金曜日か土曜日』という条件だけを考えて，３月４日（土）という日程が決定したのでありました．

（賢明な読者はこの日程を聞いて，何が起こったかすでにおわかりでしょう．．．）

３．　宿がとれない．．．．

さてさて，無事チラシも作って会員に配り，さらには非会員のＤＭリストから東北地方の人を選び出してこれにも配り，『会場もスピーカもＯＫ，宿は参加者が勝手にとる事にしたからこれで万全．４日の夜は市内に泊まって，朝スキー場に移動だ．さあて一番繁華街に近いホテルはどこかな』っと，盛岡の人に紹介してもらったリストを見ながらもう一人の世話人が受話器をとる．一軒目，二軒目，．．　ちょっと不安がよぎる．十軒目，十一軒目，．．．　『宿がとれない！！』．そう，何と３月５日というのは岩手大学の入試の日なのでありました．

さあ大変．時刻表の後ろを見たりして何とか捜し当てたのが，『バス／トイレ別，ツイン２つのシングル１つ』という発表者人数丁度の部屋数．他が全て満室なのに空いているという事実と合わせてなんとなく不気味さが漂っている．それでもないよりはまだましと，とりあえずその数だけ押えて，他をあたったのであります．何しろ空いているホテルを見つけなければ，宿泊を必要とする人に紹介もできない始末．しかもこの時点ですでに，結果としては東京からの唯一の一般参加者シスプランの後藤さんの申込があったのです．

さてさて，この電話問い合わせは延々と続きまして，ついにはホテルガイドまで買いに行って３０軒近くのホテルに電話をする事になるのでしたが，全てダメという悲惨な結果が待っているだけなのでした．まあ，最終的には先にビジネスホテルにもう少し空きがあるということが判明し，ことなきを得ましたが，冷汗ものです．

（後で聞いたところによると，そのころ先の後藤さんは５０軒目の宿に断られていたということでした．）

教訓：　楽をしようとすると，かえって大変になる．

４．　こ～ゆ～ことも．．．

時間は少しさかのぼるのですが，他の連絡のついでに他のメンバにホテルの状況を連絡すると，『高めのホテルならとれるんじゃない？』という声．『ちょっとやってみるから』との意見に待つこと５分．『６人分とれたよ～．場所がよくわかんないから，ＦＡＸ送ってもらうことにした．また連絡するから．』

１０分ほどして連絡があり，そこで場所を確認した私は，その近辺の宿は全て塞がっているのに，なぜそこだけ．．．とちょっと疑問に思いました．

『３月４日の土曜日とってくれたんですよねぇ．』

『えっ，３月３日の土曜日でしょ？』

『．．．土曜日は４日ですけど．．．』

もしこの会話がなかったら，盛岡の夜を楽しもうと考えていた男６人が，盛岡の夜を楽しく走り回っていたことだけは間違いありません．暖冬とはいえ，北の国の夜はやっぱり寒かった．．．　危ない，危ない．

５．　スキーは？？

さてさて，この手の話はこれで終るわけではありません．『スキーをやるんだ』と盛り上がっていた割には，スキー場に詳しいやつがいなかったりするわけです．

『安比でいいんじゃあない？』

『ホテルはいっぱいだった．東八幡平だったらあるけどどうなのかな？』
などと暢気なことをいっていたのが２月の２３日の木曜日．その週末には結構雪が降ったようで，『市内でも積雪２６センチ』という情報を聞いて大喜びで，月曜日を休んでまでスキーをする意思のあるものを確定して，やっと予約の電話なぞするわけです．

え〜，ところが雪が降ったのを聞いてスキーをする気になるのは何も私らに限るわけではないようで，くだんのホテルはすでにいっぱい．幸い比較的近くの『プータロー村』というところがとれまして，しかもこれがこの展開からは考えられないくらいの『あたり』だったのと，盛岡の方々が車を出してくれて５日には安比で存分に滑べれたのが今回の企画の数少ない救いでした．

何しろ『２６センチのドカ雪』はそのすぐ後の『ドカ雨』で全て消えてしまったそうですし，さらに許し難いことには，４日夜にはスキー場まで全面雨が降ってしまって，私らの滑べったのはすでに本来の『岩手の雪』ではなかったのだそうなのです．

６．　新幹線にのって

さて，フォーラム当日．上野朝９時発という『早朝』にも関わらず，乗る予定の人間はみんな揃って一安心．今日使うＯＨＰを作らなくてはとビールを飲みながら考えていると，車内検札がやってきました．丁度真中近くの車両だった為に，車両の後ろから来た車掌と，前から来た車掌が出会うところのようでした．

前から来た車掌にドアの付近でさっきからウロチョロしていた若いのが話かけました．

『済みません．指定席空いてませんか？』

『座りたいの？　じゃあこの空いているところに座って．券持っている人が来るまでね．はい切符見せて．』

『はい．』

『．．．．あれ？？　あんた指定とってるじゃない．．．．』

そのころ後ろから来た車掌が中ほどの席のオバタリアンと話をしていました．

『さっき，自分の席がここだという人が来てね．．．』

『はは．車両の間違いですか？』

『それが車両も席もあってね．ちゃんと９時発のやまびこ３７なんだけど，そ〜ゆ〜ことってあるのね〜．』

『．．そうですか，．．　普通そういうことはないんですけどねぇ．．　変だなぁ．．．　まあ切符見せて下さい．．．　あれ〜，これ，明日の指定券ですよ．．．』

車両の間違い，列車の間違い，色々と実際にでくわしもしましたが，日にち違いというのはさすがに初めてでした．しかも間違っている方が正しい方を追い返しちゃうなんてのは．．．　いやいやいい経験をさせていただいた．

まあこちらのホテル騒動も人のことをいえた義理ではありませんが．．．

７．　生うに丼

さてさて，盛岡に着いてしまった以上，多少時期はずれだろうと何であろうと，『生うに丼』を食べなければいけません．ちょうど昼過ぎに盛岡駅におりたった我々は，前日の夜行で発ったメンバと合流し，早速駅ビルの地下にある店に直行したのであります．ところがところが，喜び勇んで出かけていった我々が見たものは，うまそうな『生うに丼』のサンプルと，裏がえった料金表だったのであります．夜行メンバが言うには，つい先ほど我々を向かえに来る前までは表だったそうなのだが．．．　ことほどさようについていないのでありました．

聞くところによると，この時期はもう盛岡ではうには取れず，北海道でしか取っていないとか．それも全体の一割くらいしか身がないということです．駅で食おうとかいう方はご注意のほどを．．

８．　さてフォーラム

うに丼を食うことが出来なかった我々は，カキフライ定食（生ガキ，椀付き９８０円，安い．）なぞを食い，あわててフォーラム会場に向かったのであります．１時の約束だったのに１時半にならんとしていたので，ほとんどの準備は岩手の方が済ませてくださっていました．どうも申し訳ありませんでした．

フォーラムの内容は別途報告書が出る予定ですので省略いたしますが，話題提供者を含めて総勢２１名，ＳＥＡ入会者も２名出るなど，なかなかの成功と言えるものとなりました．しかし，皆が皆『話すことがあまりないなぁ．時間が余ったらどうしよう．』などと言いながら予定をオーバーしまくりましたので，何と討論の時間がほとんど取れないという不満もちょっと残ったりしたことも事実です．『来年』はこのあたりも気を付けなければ．．．

９．　南部百姓屋

さて，２時から５時半までの予定ではじめたフォーラムも，何だかんだで７時過ぎまでかかってしまいました．その後も，一部のメンバが代表でチェック・インにいっている間も非公式で続けられ，９時前にしてようやく夜の町へ出かけることにあいなりました．

『いかの一本づけが食いたい』という主張が通り，南部百姓屋という店につれていっていただきました．席に付くやいなや，ビール好きのメンバが『とりあえずビール１０本』．１３人の席にビール１０本は喉をうるおす量としてはちょっと多かったので，ここで酒を飲むために必死にビールを片付けるという作業が始まるのでした．しかし，いかの一本づけにボラとアイナメの刺身，．．．　というつまみにではどうしても酒が飲みたい私は，まずビールを片付け，吟醸古酒を一杯たのみ，生酒に移るころには，いつものように睡魔に襲われていたのでありました．．．
（次の日にずいぶんと私の食えなかったものの名前を聞いたような気がする．．．）

１０．　食堂園

一度寝てしまったからといってそこで終りにならない私は，南部百姓屋をあがる時に起こされると，すかさず『冷麺が食いたい』と主張したのであります．もうろうとした意識の中でも，しっかりと『特辛』を注文した私は，冷麺が来るまで引き続いて一眠りし，．．たのでありますが，覚えているのは，最初に一口食って『うまい』と思ったのと，最後に麺を探っていたのと，金を払ったところだけという，見事にポイントを押えているというか，良くわけのわからないというか，まあいつものことだったのでありました．
（一言弁明しておくならば，これは酔っ払って覚えていないのではなく，眠ってしまって覚えていないのであります．通常眠る直前の記憶はかなり鮮明な方ですので，どうやら今回はほとんど眠りながら冷麺を食ったようでありました．）

１１．　『さあみんなで滑べろう．．』

さて，いよいよ噂の『バス／トイレ別』のホテルへとたどり着きました．既に１１時をちょっとまわっていたので，風呂には入ることができませんでしたが，今思えば，どんな風呂であったか見ておくべきでした．しかし思っていたよりもこぎれいなところで，ちょっとした研修宿泊施設といった雰囲気のところでありました．ただ本来はシングルで４０００円たらずとガイドブックに出ていたのですが，ツインで１部屋９０００円と，受験日値段になっていたようでした．おそらくは部屋も，入っていた机をどけて，無理矢理ツインにしたのではないかと思います．

他の人は知りませんが，既に酔っていた（眠っていた？？）私はすぐに眠りにつきました．同室の人間が７時に部屋の目覚しをセットしたのですが，デジタルであったため，何と夜７時にセットされており，まったく役にたちませんでした．ただ私はとってもとっても良く眠ったので，別に何の問題もありませんでした．はい．

朝，水分を求めて部屋の外へ出ると，ホテルの入口脇の食堂は受験生らしき若者で溢れておりました．決して彼らが悪いわけではありませんが，ホテル騒動で大変だった私は，スキーの身支度を整えて再びその前に立つと，上のセリフを大声で叫びたい衝動にかられたのでありました．『さあみんなで．．．』．

１２．　安比高原へ

日曜日は，岩手の方々の車に分乗させていただいて安比高原へと向かうことになっておりました．車のトランクに荷物をつめていると，一人が帽子と手袋を手に持って立っていました．ちょっと勘違いした私はあわてて帽子を取り出そうとしました．

『あっ，いけね，帽子出さなきゃ．』
『大丈夫だよ．おれはあとでスキー靴出さなきゃいけないんだから．』
『ああそうか．』

するとドライバーの岩手の方が言いました．

『あっ，いけない．靴忘れちゃった．』

ということで，我々は彼の家経由で，一路（？？）安比高原へと向かったのであります．

１３．　貸スキー

やってきました，天下の安比高原．はっきりいって，ばかでっかいとしか言いようのない駐車場（７０００台収容）とスキーセンター（貸スキー１０２５台保有他）が出迎えてくれました．しかし今年の雪不足は相当なもので，何とスキー場の駐車場にも雪が全くないという惨状．このドロドロにぬかった駐車場に止められた車は，とてもスキー帰りとは信じてもらえないことでしょう．

さて私他２名は，今回は貸スキーですませてしまおうと考えていたもので，早速スキーを借りに出かけました．と，そこには田舎ものにはとても新鮮な風景が展開されておりました．オープン・マーケット方式とかいうようですが，なまじっかなスキー売場には負けていない量の，スキー，靴，ストックがズラッと並んでいたのです．

さて，まず一体どうしたらスキーが借りられるのか．．．　ここから始まってしまいます．人を待たせていることも忘れて，感心しながらボーッと見ていると，どうやら入口に近いところの人だかりが申し込みのものらしいということがわかりました．そこでは，名前，住所，身長，体重，スキーレベル，板の長さ，靴の大きさ，ストックの長さ，．．　という用なものを記入させられたあと，身分証明になるものを取り上げられました．次に，その用紙を持ってずずっと奥へ行き，自分で靴を履いてみて，その辺にあるいている兄ちゃんに良さそうなものを示すと，そこで靴にふってあるナンバーを記入され

ます．そのあとスキーのあるところにいって同じように選び，それを持ってドライバーを持っているおっちゃんのところへと進んでいくのでした．

いまではこういうところが多いのかもしれませんが，私は初めてだったのでなかなか混乱してしまいました．良く考えれば，せっかく選べるのですから，もっと真剣にスキーを選べば良かったのですが，そのときは全く気がつかず，一番近くにあったものを選んでいました．ううむ．ちっとも仕組みが生かされていない．．．

１４．　リフト券

ここはリフト券も『新方式』だそうで，１日券は裏がシールとなっており，これを二つ折にして，ファスナーの穴に通した針金を挟み込むようになっておりました．ホルダーがいらないので，便利といえば便利なのですが，どうにもこれが値札に見えて仕方がありません．３５００円で売りに出されているような気がしてならないので，しばし考えた末，手持ちのホルダーに入れてしまうことにしました．ゲレンデに出てみると，多くの人はやはり手持ちのホルダーに入れて使っておりました．まっ，新方式なんてものはどこでもそんなものです．

１５．　ゲレンデ

ゲレンデは広い，というよりも１本が非常に長いという印象を受けました．また，今回は雪質が悪かったせいか，はたまた東京の田舎もんが大挙して押し寄せるにはちょっと遠いせいか，我々にとってはかなりすいているという感じでした．何というか，１本のリフトに柱が２５～６本あるという感じで，実際かなり長い．ペアリフトがほとんどなので，女の子とでも乗り合わせて話でもしていればすぐですが，結構すいているので慣れていないと一人になります．またやっぱり誰でもいいというわけではないですし．．　途中リフトから落ちた女の子がいたそうですが，目撃者によると『きっと隣のおっさんが手を．．．』ということでしたので，きっと向こうもそう考えていることでしょう．しかしリフト待ちがほとんどないので，結構ご機嫌に（気分だけは．．）滑べることができました．ただ，午前中はまだしも，午後も３時頃になると早くもガチガチの斜面になってしまい，地元の人もせっかく来たのに残念ですねといっていました．『来年』はきっともっといい雪で滑べることができるでしょう．．

１６．　プータロー村

色々あって，安比で滑べって東八幡平の近くに泊まるという，何というか岩手の方のご協力がなければ大変うっとうしいことをしたわけですが，先ほどもいったようにここは『あたり』でした．

まあ大騒ぎするほどのところでもないかもしれませんが，ちょっとしたスペースにコテージが点在しているような作りの場所なのです．何がよかったかといいますと，まずおねえさんが可愛くて親切．（これはもう一人の世話人がいいました．わたしも賛成します．）また，コテージに入るともう既に暖房が入れてあるという細かい気の使い方なども気に入りました．夕食は，自炊でも，レストランでも，はたまたそれをコテージに持って来てもらってもＯＫです．我々は，持ってきてもらうと酒の追加注文ができないと聞いてレストランにしたのですが，一升瓶の一本も用意しておけばそんな心配いらないんですよね．実際レストランで食事をしたのは我々だけだったということです．

用意してもらう場合，食事は予約制で，２０００円，３０００円，４０００円というコースがありました．３０００円のやつにしてみたのですが，はっきりいって食べきれないほど出てきました．

（ちなみに私は全部食べましたので，『田中が食べきれないほど．．．』と誤解して驚かないようにしてください．）

ものはすきやきを中心に，刺身，焼き魚，味噌汁，ちゃわん蒸し，山菜，ええとその他に四品くらいはあったと思います．朝食は１０００円で，これもまあまあちょっとしたものでした．

また温泉があり，もちろんコテージでも入れます（ただしこれはかなり狭い）．その他に受付のある建物に一つと，敷地内にもう一つ大きいのがあります．単純泉のようですが，お湯は熱めで，なかなか気持ちのいいものでした．夏であればテニスコート，プール，アーチェリー，フィールド・アスレチックの公認コース，野球場なども使えるようですし，マージャン道具も借りられます．ちゃんと研修所もあるようですから，ＳＥＡの若手の会などここでやってもいいのではないかなど考えました．が，盛岡駅からバスで７０分ですから，さすがにちょっと遠いかもしれません．でも八ヶ岳よりはずいぶんきれいな作りだし，温泉もあるし，食いものもうまいとは思うんだけど．．．　ちなみに料金は，４人用のコテージ，１５０００円を３人で借りたのでちょっと割高になりましたが，１泊２食税・サ込みで１００００円チョイでした．

１７．　東八幡平のゲレンデへ

さて明けて６日，泊まった３人のうち１人は滑べらずに帰ることになっていました．時刻表でプータロー村に寄ってくれるバスは日に４本くらいしかないということを発見した我々は，近くのバスセンターまで送ってもらえるかどうかを飯の前におねえさんに聞きに行きました．前日，ゲレンデまではいつでも送ってもらえるとは聞いていたのですが，バスセンターまではちょっとありそうなので時間指定だろうと思っていたのです．でも２人がチェックアウトしてしまうとバスが来るまで２時間もあって困ってしまう．いっそのこと一緒にゲレンデまでいって，そこからバスに乗ろうかなどといいながらご相談に行くと，いとも簡単にＯＫの返事．良かったね，などといいながら飯を食って戻ってくると，『車の用意ができてます

のでどうぞ』とのこと．見ると車の脇に別のおねえさんがしっかりとお迎えのかっこで立っている．こちらはというと一人がトイレにいっていてまだ戻っていなかったので，向こうを待たせる格好になってしまい，かえってビビッてしまいました．

じきに３人揃ったので，ほっとしながら車のところに行くと，なんと運転手がいない．．．　何だ焦ることなはかったのかなと思いながらも，もしや．．　いや有り得るなと考えていると，案の定，我々をのせてドアを閉めるとおねえさんは運転席の方にまわったのでありました．トイレにいっていた男が，ご機嫌でおねえさんに話かけたのはいうまでもありません．

東八幡平のスキー場は，リフトを２～３本登ると一番上という比較的小さなところでしたが，やはりリフト１本はかなり長いものでした．月曜日ということもあって，リフトは乗り放題でしたが，２時間で１０本滑べるのがやっとでした．もちろんこちらが下手なせいもありますが，こちらは安比に較べて標高が低いせいか，雪質はさらに悪く，前日に滑べったあとがそのままガチガチに固まっていて滑べりにくいということもありました．

滑べりにくいといえば，最近のスキー靴の性能は大したものだということを再認識しました．安比と違ってこちらは昔ながらの貸しスキーだったのですが，スキー靴もかなり昔のというか，安いタイプのものだったのです．今の靴はすねの中ほどくらいまで高さがあり，前傾姿勢が保ち易くなっているのですが，ここで貸しているのはそうではなかったので，自分でかなりふんばって前傾を保たなければならないのです．斜面が悪いこともあって，かなりシンドイ思いをしました．

しかし１泊くらいで自分のぼろスキーをかついで行く元気もないしなあ．．

１８．　うに丼の夢消えて

東京に５時頃帰りたかったので，残念ながらきっちり２時間滑べって，路線バスで盛岡駅に向かいました．１時半頃盛岡駅に着く予定だったのですが，さすがに路線バスだけあって，１時間を過ぎるころから，乗ってくる方がものすごく増え出しました．とてもスキーの荷物を置いておくような場所はありません．仕方なく無理矢理ひざに抱えておりました．しかし途中で高校生が大挙して乗ってきたのには驚きました．確かに高校の前か何かのようではありましたが，何であんな時間にあんなに帰る高校生がいたんだろうか．．

盛岡駅に着いて，来た時に果たせなかったうに丼を食うという作業に取り掛かろうとしたのですが，何と連れがまだ冷麺を食っていないというので，泣く泣く諦めました．というのは冗談で，おそらくまたうに丼はなくなってしまっているだろうと思ったので，時間もなかったし，さっさと冷麺を食いにいったのでした．駅の向かいにあるビルの２階にある店でしたが，ここのやつもなかなかおいしいものでした．

しかしうに丼が食えなかったのはやっぱり悲しい．『来年』はきっと何とかするぞ．．

１９．　終りにかえて

見事なまでに本題を外して書いてきましたが，今回の最大の教訓は『とにかく準備期間をたっぷりとる』ということでした．『来年』もそうですが，とりあえずは今秋にやるはずの『ＳＩＧＥＮＶ＠ＫＯＢＥ＆ＪａｚｚＳｔｒｅｅｔ』の時には注意をしなければ，と考えています．秋のものはフォーラムにするのかどうかまだ（当然のように）決まっていませんが，Software Documentation を予定しています．今度はなるべくはやめに案内を出しますので，是非ご参加ください．

いやいや，それでもまあ何とか無事に終って，良かった良かった．めでたし，めでたし．

２０．　おまけ

というような感じでこの文は終るはずだったのですが，『あわてるな，世間はそんなに甘くない』とばかりにまだ続きがあったりするわけです．

翌７日，私がオフィスに顔を出すと，机の上に何かチラシのようなものが置いてあるのです．

『ふむふむ，３月のＳＥＡＦｏｒｕｍの原案か．．．

『おお，ソフトウェア・プロトタイピングの効用ね．．

『あれっ，コーディネータ，田中？？

『ん～～，盛岡ミーティングでの討論報告．．．』

というようなわけで，ほとんど討論ができなかった討論会の討論報告をどうしたらいいのだろうとＥＮＶのメンバは現在バタバタしているのでした．

お～い，どうしよう．．．

［３月のフォーラムは我々にとってはかなり悲惨なものでしたが，他のスピーカの方々に助けられて，そこそこのものにはなったと思っています．次回「プロトタイピング特集号（その２）」では，盛岡ミニ・フォーラム，３月月例フォーラムの報告を中心にお届けします．お楽しみに．］

九州支部設立一周年報告

平尾 一浩
Hirata Software Technorogy Co.,Ltd.

はじめに

関西、横浜、長野、名古屋に続く5番目の支部として九州支部が設立されたのは、昨年(昭和63年)の3月12日でした。

この一年間は試行錯誤の繰り返しでまとまったテーマで何かをするということができませんでした。これは、世話人である私の計画性のなさと実力のなさの現れでもあると反省しています。(本当はB型という素晴らしい血液型のせいだと思って、あまり反省していない。ねっ!)

しかし、SEA幹事の方々をはじめ、事務局、各支部、分科会の方々に支えて頂き、なんとか一周年を迎えることができました。この一年で行なったいくつかのイベントも多くの方々の協力と理解があって出来たことで、皆様にはほんとうに感謝の気持ちでいっぱいです。

今回はこの一年を振り返り、来年へ向けてのスタートを切るということで、先日行ないました SEA 九州支部設立一周年記念講演会の報告も合わせて、設立一周年報告としたいと思います。

1. 活動状況

今のところ、熊本を中心に行なっています。しかし、沖縄や鹿児島、宮崎、大分の方々には集まり難い思いをしていただいています。発足時は各県で持ち回り制にしようという意見がありましたが、会員の少ない県では、その受け入れ体制が作れず、実現できませんでした。

会員数はこの一年で、35名から67名、賛助会員 3社にまで増えました。しかし、この一年間に行なったイベントに参加できた方は一握りの方だけです。各地区の活動を主にしてはいますが、会員数の少ないところは活動自体が困難な状態です。

1.1. 支部活動

支部の活動、会合としては、5月にICSE報告として代表幹事の岸田氏においでいただき、講演をお願いしたのですが、内容は急きょ変更となり、『これからのソフトウェア開発環境』というテーマで、講演が行われました。これは、2種類のビデオを基に、最新の開発環境の流れをわかりやすく説明していただきました。事務局から中島さんにも参加して頂き、なかなかの盛り上がりでした。そのまま夜の部に突入し、歓迎の踊りなども披露され、九州支部らしい異常な世界が醸しだされました。(参加者数:50人)

第2回として11月には、熊本県、熊本情産協、熊本テクノポリス財団の後援で、九州ソフトウェア・シンポジウムを開催しました。参加者160名という大きなもので、今までにない(熊本ではなかった)技術者レベルの会合となりました。このシンポジウムはテーマを【情報産業のインフラを考える】として開催し、特別講演、集中討論、フリーディスカッションの3つを行ないました。特別講演としては熊本大学の山川先生(現在は九州工業大学)に『ファジィ情報処理のおもしろさ』を話していただきました。

集中討論としては、そのころは名前がまだ存在していたΣサブセンターを題材に、その期待ということで、チェアマンに林氏(SRA)、パネリストとして岸田氏(SRA)、中野先生(阪大)、藤野氏(FXIS)、そして地元熊本から飯冨氏(システムイイトミ)、筏井氏(HST)にそれぞれが考えるΣサブセンターへの期待を話題提供として語って頂き、討論に入りました。 (発表、及び討論内容は誌面都合により割愛させていただきます。 :-) どうしても、ど〜しても詳しい内容を聞きたいと言われる方は平尾まで御連絡下さい)

フリーディスカッションでは、日中ソフトウェア事情というテーマでチェアマンに深瀬氏(ASCII)をむかえ、日中ソフトウェアシンポジウムに参加された日本側団長の熊谷氏(富士通)をはじめとする、中心メンバー(岸田氏(SRA)、林氏(SRA)、中野先生(阪大)、藤野氏(FXIS)の5人の方々)にスライドを使用した話題提供を行って頂きました。参加者の中からは、JUS(日本UNIXユーザ会)はアメリカばかりを見て、SEAは中国ばかりを見ているような感じを受けるが、もっと足元を見た方が良いのではないか、などという意見も出され、なかなかの盛り上がりでした。さらには熊谷氏が言われた『中国はとにかくスケールエラー』という言葉から、その後スケールエラーが流行語になるくらいでした。

話題提供では中国のソフトウェア開発の実情や、開発環境の紹介までをして頂き、今までほとんど無関心だった方々にも中国のソフトウェア開発、及び開発環境に興味を持って頂けたようです。

1.2. 地区活動

各地区の活動としては、福岡、佐賀、長崎、沖縄地区がネットワークを中心に事例を基にした発表形式で例会を開いてきました。熊本、大分地区は全く統一的なテーマはなく、"設計"、、"フリーソフトウェア"、"汎用機による開発環境"などまったくバラバラなテーマで例会を開いてきました。宮崎、鹿児島は、会員数が少なく、活動自体が困難な状況で、例会実績はありません。福岡での

例会はいつも10人位で行われ、熊本での例会は大体20〜30人で行っています。

2. 平成元年

今年の計画としては、以下のものを予定しています。

1: 九州支部全体の会合
年2回(4月、9月)
4月分は、九州支部設立一周年記念講演会を開催。

2: 各地区活動
毎月一回
熊本、大分地区は第2土曜、福岡、佐賀、長崎、沖縄地区は第3土曜で、宮崎、鹿児島地区は立ち上げを目標に動く。

3: その他イベント
第2回九州ソフトウェア・シンポジウム(10月)

4: SEA本体のイベント九州誘致
教育ワークショップ(9月)

以上の計画を行っていますので、九州へお越しの際には、是非九州支部へお立ち寄り下さい。

昨年の熊本、大分地区活動は、まだ全員が不慣れということもあり、受け取るだけで出すのもがなかなか無いという状況でした。ですから、世話人中心の偏った内容になり、広がりというものが感じられませんでした。技術移転などというもののまだまだ前の段階で、人的交流が何とか出来たという具合です。しかし、一年が過ぎようとしていた位に、多くの会員の方から、是非、例会の担当をしたいという話が出てきました。皆さん、それぞれにいろんな経験や、知識、技術をお持ちなのですが、今まで SEA の会合に慣れずにおられたようです。そこで、今年は2ヶ月単位で担当制にし、担当月は統一テーマでそれぞれの話題を提供しあうということになりました。ここえ来てやっと踏み込んだ議論や技術移転というものができるようになるという期待をしています。

3. SEA九州支部設立一周年記念講演会

4月22日、支部設立一周年を記念して、講演会を開催しました。講師として、松原氏(日立ソフトウェアエンジニアリング)、加藤氏(東京工業大学)の両氏においで頂きました。その講演内容を簡単に紹介します。

松原友夫氏

【ソフトウェア開発の人間的側面】
- Peopleware -

プロジェクト管理の諸問題を論ずる場合、特にその人間的な側面を考慮することが重要です。松原氏は最近邦訳された T.DeMarco と T.Listrt の名著「ピープルウェア」の翻訳、監修を手掛けられました。今回、その「ピープルウェア」をもとに講演して頂きました。

内容に関して、残念ながら講演の録音が失敗していまして詳細にお伝えできないのが残念です。そこで、当日使用されたOHP の内容を御紹介することにします。

<人的資源の管理>

☆今日もどこかでトラブっている

・ありふれたアプリケーションでも多い失敗ー技術問題でない証拠。
・ 500のプロジェクトを調べたら、
15%が動いていない。
25人年以上のプロジェクトでは25%が未完
・その原因は技術に非ず。

・問題の本質
・政治的要因と言うが、実は社会学的要因
・難しいのは技術よりも人
・管理者は、人の管理について教えられていない

・ハイテクの幻想
・ソフトウェア開発はハイテクとは程遠い
・「先端技術」ではなく「人間関係技術」

☆チーズバーガー生産・販売基準書

・ファーストフードのやり方は開発作業では使えない

・誤りの割当
・品質のよくない部分は潔く捨てろ
・間違いは許されない雰囲気が間違いのもと
・少々の間違いを大目に見る
・「特に問題なし」は最悪の答

・管理とは尻を叩くこと?
・尻叩きは頭を使う仕事には使えない
・仕事への愛着を信じよ

・手配師
・多くの管理者は人を取り替え可能な部品として考えている
・回り道して交換がきかないことを悟る
・キーマンがやめるのが怖いのに、キーマンはいないと思い込む
・「ジョージ・ガーデンハイヤがくたばったので、新しいのを頼む。今度はあまり生意気でない方がいいな」
・「金よりも欲しいものは自宅でのよい端末」

・プロジェクトと死後硬直
・プロジェクトの目的は製品を出して完了させること
・安定するのは終ったとき
・プロジェクトの動的な面にこそ神経を集中させる
・彼女が入ったプロジェクトは必ず成功 : その秘

密は??
・触媒の重要性
・考える暇があったら仕事をしろ!
・実質的に仕事をする時間として、どれ位を割当るべきか?
・「その他」の重要性
・「考えている暇があったら手を動かせ」
・考える時間を確保するのが大変
・ソフトウェア開発者は本を読まない

☆ウィーンがあなたを待っている。
・チーム員の離婚や子供の不良化を自慢する管理者の話
・スペイン流管理
・スペイン流---富の量は一定------搾取
・イギリス流---富は作り出すもの--産業革命
・今でも生きているスペイン流の価値観
・タダ働きで生産性を上げるのはサギ
・家庭からの嘆き
・家庭ではまったく違うことを言われる
・大切なものは一人一人の人生
・残業なんかくそくらえ!
・オーバータイムとアンダータイム(水増し時間)
・見えない時間、サービス業、アンダータイム
・残業は全力疾走
・仕事中毒
・一種の一時的な病
・もっと大切なものを犠牲にしていると気づいた時
・生産性:試合に勝って勝負に負ける
・生産性を改善すると退職を増やす危険
・生産性を高水準にしたが全員やめた話
・スペイン流管理のつけ
・せかせても良いものはできない。早くなるだけだ

☆品質第一・・時間さえ許せば
・感情を入れずに仕事はできない
・感情は自尊心が傷つけられることによって高ぶる
・自尊心と仕事の質
・品質第一主義からの逃避
・時間を節約すると品質を犠牲にする
・品質と生産性は両立する。日本にはこの二つのトレードオフいう考えはない
・品質はタダ、しかし・・・
・タダなら絞り取れ、という誤解
・品質はタダ、しかし喜んで金を払う人に対して
・品質第一は文化
・出荷拒否権
・日本の例

☆パーキンソンの法則の復活
・与えられた仕事をするのに時間はいくらあっても余らない
・パーキンソンの法則とニュートンの法則
・人生はだらだらと仕事をやるにはあまりにも短い
・仲間にこんなことは言えるだろうか?
・自然の欲求を持つ
・パーキンソン流に扱うな
・ニューサウスウェールズ大学のデータから
・プログラマが目標を立てると生産性が高い
・システムアナリストが立てるともっと高い
・目標設定をしない場合はさらに高い!
・パーキンソンの法則をちょっと変えたら
・不要、不急業務の増大

☆癌によく効く ? 「ラエトラル」
・杏の種のエキス、ラエトラルは癌に効く?
・末期患者は藁をつかむ気持ちでこれを飲む
・生産性向上に効く技術的ラエトラル
・眠っているうちにやせられる
・山程ある生産性倍増商品の広告
・決定版は「眠っているうちにやせられる」
・技術面でおくれていても落胆する必要はない
・管理者が陥る7つの錯覚
・生産性を上げる方法がまだある
・他のところでは2倍も3倍も上げている
・技術は進む、おいてきぼりにならにか
・方法を変えれば上がる
・積み残しが多いから生産性を上げよ
・何もかも自動化
・プレッシャーをかければよく働く
・これこそ管理だ
・ふらふらで仕事をしているトムにシャロンが出してくれた温かいスープ
・「人を働かせる」ではなく「人を働く気にさせる」

<オフィス環境>
・遅れの原因限りなし、つける薬は何もなし

☆施設監査役
・「本当のオフィス環境にはお構いなし」
・施設監査役の精神構造
・見かけの素晴らしさ
・いずこも同じプラスチックのオフィス
・誰が窓際のいい席をとるか

・気持ちが良さそうなところは廊下
・地下室も理想的
・ガリガリどなる社内放送

☆プログラムは夜開く

・残業の真の目的は質の向上
・仕事がはかどるのは夜か早朝

・会社の言い訳
・一番多い苦情にほほかむり

・プログラミングゲーム:生産性の要因を探る
・同一企業内の2人で一組
・競技は別の場所、チームであり個人の競技
・仕様書からテストまで
・延べ600人が参加

・プログラマの個人差
・トップとどん尻は10倍の差
・トップは平均の2.5倍
・上半分は下半分の2倍以上

・生産性と無縁な要因
・言語、経験、残存不良、年収

・誰も書かなかった生産性要因
・誰とチームを組んでいるか
・同一企業内の二人にバラツキはない
・企業におけるプログラマの能力さも
・企業自体の生産性も10倍の開き
・企業文化が優秀なプログラマをひき寄せる

・オフィス効果
・作業環境と生産性の相関関係は顕著

・で、何がわかったか?
・オフィス環境の改善は管理者の責任

☆スペースに金をケチると

・ケチった金より大きな生産性低調による損失

・環境の破壊
・「解放型オフィス」という災難
・考慮されないプログラマの生産性

・悪口は中断して数値を示そう
・IBMのサンタテレサ研究所

・オフィスの良否とプログラムの品質
・不良 0 : 騒音レベルは我慢できる 66%
不良1以上 : 騒音レベルは我慢できる 8%

・ノーベル賞級の大発見
・人工密度と比例する騒音

・オフィスから雲隠れ

☆頭脳タイムと肉体タイム

・1人による作業 30%
2人による作業 50%
3人以上による作業 20%

神経集中
・フロー状態の幸福感
・割り込みで少なくとも15分はムダになる

・いつまでも続く騒音、割り込み

・勤務時間と精神集中

・E計数=割り込み無しの時間/机の前に座っていた時間

・オフィスは赤いバンダナの園
・割り込みお断りのサイン、バンダナ

・考えごとは家で

☆電話

・別世界にて

・電話してくる客の方が大事??
・ベルが3回鳴るまでに受話器をとれ!

・電話倫理の改訂
・電子メイルシステム
・割り込みお断りのボタン
・ベルに紙をつめる

☆ドアの奪回

・あきらめるのはまだ早い

・ケバケバしさの問題
・機能性より概観が気になる

・創造的空間
・対症療法と根治療法
・左脳と右脳

・活力あるオフィス

会社のしきたりを破る
・自由なレイアウト

☆傘が開くように

・永久不滅の建築法

・アレキサンダーの組織均衡論
・会社組織も進化論的に変遷する

・組織的均衡のパターン
・よいオフィスにはよいバランスが必要

・第1パターン: キットによる組立

・第2パターン: 窓

・第3パターン: 屋内と屋外

・第4パターン: 共通の場所

・パターンのパターン

・現実に戻ってみると

〈適材〉

☆ホーンブロワー因子

・フォレスターの海洋冒険小説の主人公
ホンブロワーは究極の管理者

・先天性と後天性
・最初に適材を獲得するべし

・型にはまった人間
・失敗の多くは外見にとらわれた採用
・画一性の要求は不吉な兆候

・標準的な服装の悪影響

・合言葉:それらしさ
・「らしくない」ことはやっては困る

・企業のエントロピーも増え続ける

☆奇術師を雇う

・奇術を見ないで雇おうとするのはばかげている

・成果一覧
・面接指向の採用
・求められない候補者の成果物

・能力テスト
・役に立つのは最初の仕事
・左脳の能力のみ
・役立つが採用に使うのは良くない

・オーディション開催のすすめ

☆ここにいるのが楽しい

・退職率、退職コストに答えられるのか

・退職というムダなコスト
・アメリカの平均在転期間は約2年
・採用コストは1.5〜2ヶ月分の給料
立ち上がりのムダ、交替のロスを加える
とザッと5ヶ月分

・退職の隠された害
・短期的視野のツケ
・早過ぎる昇進
・頭でっかちの組織
ものを作っているのは入って1〜2年の人

・なぜ会社をやめるのか
・腰掛けメンタリティ
・使い捨てにされる予感
・忠誠心強要への反抗
・退職は伝染する

・会社の移転に伴う病理
・一人の仕事を移すだけではすまない
・昔はできたが今では愚行
・ATT移転の悲劇

・永続性のメンタリティ

・低退職率の会社の共通した特徴:
ザ・ベストの追求
コニュニティ感覚
再教育プログラムの充実

☆事故修復システム

・決定論的システムと非決定論的システム
・決定論的システムは自己修復機能を失う

・作業規模の弊害
・意味がなくても書いてある通りやる
・考えるな、その通りやれ

・狂気の作業規定
・書類の山は災いのもと
・問題を解決しなくなる
・責任感が薄くなる
・違反すれば無能力者→意欲喪失

・悪意の追従
・わざとその通りやる

・赤ん坊の湯あみのように
・強制は反発をもたらす
・穏やかに誘導する
・現場で検証されたものだけが規定となる(デュポン)

・ホーソン効果
・ソフトウェアの生産性向上成果もホーソン効果

〈生産性の高いチームを育てる〉

☆チームは個人の和よりも強し

・結束チームとは
・個人の集合より大きな成果を生むチーム
・結束しはじめたら成功間違いなし
・共通の目標が重要

・感情的楽観主義
・放っておいても組織の目標を受け入れると信じる
楽観主義

・ナバロンの要塞
・目標は勝手なもの、スポーツを見よ
・結束チームは要塞さえ襲撃する
・目標は達成でなく一体になれること

・結束チームの特徴
・人が入れ替わらない
・不注意な配置転換は命取り
・生産物共有意識

・チームと派閥
・そよ風とすきま風

☆黒装束チームの伝説

- **伝説のいわれ**
  - ・もとは普通の出荷テストチーム
  - ・チームに起こった大きな変化
  - ・プログラムに試練を与える喜び
- ・哀れな地球人よ!
  - ・プログラムが止まるとケタケタ笑う
- ・後日談
  - ・人が入れ替わってもチームの個性は残った

☆チーム殺しの7つの技法

- ・自己防衛的管理
  - ・部下を信頼しない管理者
  - ・管理者による技術的干渉
  - ・信頼されていないと感じれば結束しない
- ・官僚主義
  - ・ドキュメントの洪水は目標意識を育てない
- ・作業場所の分散
  - ・チームの結束より周りの人と仲良しに
  - ・チームの文化は生まれない
- ・時間の分断
  - ・1人が2つ以上のプロジェクトを分担するとチームは結束しない
- ・品質低減製品
  - ・早かろう悪かろう、は自尊心を傷つける
- ・さばを読んだ期限
  - ・部下への信頼も上司への尊敬も生じない
- ・派閥統制
  - ・チームは終ったら解体し、一緒には動かさない傾向
- ・意識的にではないが結果的にチームを殺している

☆スパゲッティディナーの効果

- ・チームリーダになる人から自宅での夕食の招待を受けた
- ・チーム効果が現れる
  - ・みんなで買物や料理や片付けを分担
- ・続いて起こったこと
  - ・プロジェクトより共に働ける機会
  - ・唇を持つ鶏がわかるか
- ・誰が本当の責任者?
  - ・自然の権威

☆オープンキモノ:問題をさらけだす

- ・信頼できる人に全てを委せる
  - ・全てをさらけ出すオープンキモノ経営
- ・合宿作業
  - ・オフィスから逃れることの効果
- ・規則は破るためにある
  - ・スカンクワークプロジェクト(闇のプロジェクト)
  - ・否認されても生き残ったプロジェクト
  - ・反抗賞のメダル

☆チーム形成の不思議な作用

- ・品質至上主義
  - ・「ほぼいいだろう」は弔いの鐘
  - ・品質至上主義はチーム結束の触媒
    - -カキの中の砂
- ・結婚したから愛している
  - ・終結による満足感
- ・エリートチーム
  - ・「我社はエコノミークラスの会社ではない」
  - ・チームのアイデンティティー
- ・ヤンキースを解散させないで
- ・チームはネットワークモデル、階層モデルではない
- ・中華料理メニューからの選択
  - ・異分子の作用、女性のいるチーム
- ・全てを一つに収める
  - ・小さな町の伝統的なコミュニティーのように

<きっとそこは楽しいところ>

☆混乱と秩序

- ・混乱状態をキチンとする快感
- ・進歩こそ大問題
  - ・秩序が欠けていたから新鮮
  - ・小さな混乱の建設的導入
- ・新手法試行プロジェクト
  - ・期待されるホーソン効果
  - ・全てを試行プロジェクトとする会社
  - ・一度に一つの手法を
- ・プログラミングゲーム
  - ・小さなプログラムを
  - ・全く普通の仕様書で
  - ・週末を利用する
  - ・ルールや目的を前もって知らせる
  - ・居るのは参加者だけ
  - ・誰にも何等かの賞を
  - ・興味あるデータを記録する
- ・ブレーンストーミング
- ・教育、旅行、学会、お祭り、そして保養

☆自由電子ー会社の枠組みから飛び出す人達

- ・つまみ食い症候群
- ・フェロー、グルー、インタープルナー
  - ・新しい雇用形態

・おせっかいはやめて

☆眠れる巨人、ホルガーダンクス－まとめ

・だが、なぜ私が?
　・たった一つの変化でも一人ではできない
　・闘牛士　エル　コルドベス
　・それはみんなの問題
・眠れる巨人
　・危機の際だけ目を覚ます
　・企業の中のホルガーダンスク
・ホルガーよ目を覚ませ!!

上に紹介しましたOHPの内容を見てもわかるように、講演全体がピープルウェアに沿って進められ、理解しやすいものでした。

会員の中からは、「この本は僕のために書かれたような気がする。読む度に涙が出そうになる」などというコメントが出され、笑いに包まれるという一幕もありました。その他に、チーム立ち上げ時に関して、チームの概念やベクトル合わせの必要性についての意見や、経験の積分値の利用法などが話されました。その他にも、【管理とは人を働く気にさせること】ということに関して、実際の管理をされている方から「それは働く気のある人がいることが前提」といった意見も出され講師の松原さんをはじめ、多くの方が同意されていました。結局は管理する側もされる側も、お互いに対して多くの期待はするなということだったようです。

(ちなみに、HSTでは、話の中で出た赤いバンダナを即実行した人間がいましたが、バンダナの意味に対する質問責めにあい、割り込みがさらにひどくなったと、止めてしまいました)

加藤朗氏

【The JUNET】
- その歴史、現状、将来 -

JUNET　は、計算機研究者の情報交換の環境を確立するため、また、ネットワークの研究基盤として利用されることを目的とした実験ネットワークです。同ネットワークの運用は1984年10月から3つのローカルエリアネットワークのゲートウェイである東京工業大学、慶応義塾大学、東京大学を結合して開始されました。現在はその範囲も日本全国に広がっています。このJUNETの構築に初期の段階から携わってこられた加藤氏に、その歴史と現状、将来について講演して頂きました。

以下に加藤さんの講演内容を簡単に御紹介します。

はじめに

☆ N1-NET .....JUNET 以前のアカデミックネットワーク。

全国の大学の、 7 つの大型計算機センターにあるメインフレームを接続したネットワーク。

□ サービスとしてはリモートログオン、ファイル転送の 2 つ。
□ 高い課金制度であったが、当時としては画期的なネットワークであった。
□ 電子メール、電子ニュースのサービスはなし。
（最近電子メールのサービスがはじまった）
□ 閉じたネットワーク他のネットワークとのインタラクション

はなかった。

一方アメリカには APANET の構築がなされ、主な大学の研究者達は電子メールで情報交換を行なっていた。

□ 電子メール、電子ニュースの必要性。
□ 海外の研究者との情報交換。
□ ネットワークの研究、開発の促進。

↓

JUNET の誕生。

JUNET の動機

☆ JUNET 発足の三大動機

□ 電子メール、電子ニュースによる情報交換。
□ 海外のアカデミックネットワークとの接続。
□ ネットワーク自身の研究活動をプロモート。

☆ JUNET の制約 .....VAN 業者との競合を避ける。

□ 営利目的に使用しない。
□ 不特定多数の使用に供しない。
□ 専らネットワーク運営を行う組織ではない。
あくまで片手間で行う。

☆ ボランティアベースで行なう実験ネットワークとして運用開始。

□ 計算機科学、社会科学、心理科学などのいろいろな実験を行なえなければ
いけない。
□ メッセージが fail しても責任を持てない。

JUNET の歴史

1984/Oct.....東工大、慶応、東大でスタート。

□ VAX, 4.2BSD, 300bps (V.21) モデム。
□ 通信プロトコルは UUCP を使った電子メールの実験開始。

1985/Jan.....KDD 研究所が USENET に接続。
□ 電子ニュースを配送するネットワーク。
□ 国際接続の開始。

1985/May.....ドメイン形式アドレスの採用。
□ 電子メールの fail する数が減った。

1985/Automn.....電子ニュース、電子メールの日本語化。
□ 当時は漢字 UNIX がなかったため、7 bit 表現。
・JIS C6220
・JIS C6226
□ 漢字端末が少なかったため PC 上にターミナル・エミュレータを
走らせて使っていた。
□ ESC を通すように news ソフトを改良。

1986/Mar.....CSNET とのリンク。
□ やや高信頼なリンク。
□ ゲートウェイは東大。
□ 課金制度。
□ 利用は大学に限定。

1987/May.....INETCLUB 発足。
□ 民間企業も利用できる国際リンク。
□ アメリカとヨーロッパとの接続が確保された。

1988/Oct.....東工大、東大が Internet Protocol で接続。
□ イーサ・ネットと同じプロトコルなので、イーサ同様のサービスが
受けられる。
・リモート・ログイン
・ファイル転送
□ 学術情報網の整備によるリンク。
・東大 - NSF (Washington D.C.) (専用線)
・東大 - 阪大 (専用線)
・京大、東北大、九大
□ 公式 IP アドレスの取得促進。
・バルク・アサインメント
□ 電子メール、電子ニュース以外のサービスを準備中。

1989/Apr.....アドレス表記の改訂。
□ ISO 3166 国別 2 文字コード
・日本 .....JP
・イギリス .....UK
・韓国 .....KR
など。
□ 属性を表す第 2 レベルドメイン名
・大学等 .....ac.jp
・政府関連機関 .....go.jp
・企業 .....co.jp
・非営利団体 .....or.jp
・ネットワーク管理母体 .....ad.jp

☆ ボランティアベースの実験ネットワークという正確は保持。
☆ トラフィックが増えて管理が大変。

JUNET の現状

☆ 参加組織.....200
□ ほとんどが東京と京阪神地区に集中している。
□ 九州は 13 + 1 組織。
☆ 接続計算機
□ 不明
□ 推定では 2500 - 3000
☆ 平均的なリンク
□ 9600 bps、ダイアルアップ・コネクション、UUCP。
□ 2 時間に 1 回程度のコネクション。
☆ 4.[23]BSD 系の UNIX
☆ 約 12 のバックボーンノード
□ 九州では九大が、東京では東大や東工大などが行なっている。

電子メール

JUNET の代表的なサービスである。
☆ 正確な数は測定不可。
☆ 着信メールの約 1 割は海外から。
☆ 国内メールの平均到達時間は 3.2 時間。
☆ 多くのメーリングリスト。
☆ アドレス表記

kato@cs.titech.ac.jp

kato .....ローカルパート
ユーザ名、メーリングリスト
cs .....第 4 ドメイン名
組織内サブドメイン
titech .....第 3 ドメイン名
組織名
ac .....第 2 ドメイン名
組織の種類
jp .....第 1 ドメイン名
日本

□ JUNET で管理しているのは第 3 レベル (組織名

）まで。

海外とのメール交換

☆ 現在 2 つのゲートウェイが稼働中

□ 東京大学 .....大学関係、政府機関関係。

relay.cs.net

□ kddlab...KDD ではなく INET が運営。

uunet.uu.net, mcvax.cwi.nl (nl は NetherLands オランダのこと)

現在はユーザが陽に指定

☆ UUCP、PMDF によるバッチ式リンク。

ある一定時間毎に電話で呼び出す。

☆ 1989/Apr より NSF との専用線上の X.25 IP 接続

□ Internet から 10 分程度で mail が到着。

□ 低廉価な課金額。

□ ac.jp（大学）, go.jp（政府関連機関）に使用が限定。

□ 9.6kbps のため mail で手一杯である。ftp、telnet には解放不可。

電子ニュース

パソコン間で行われている「電子掲示板」と基本は同じものである。パソコンの場合は中央のコンピュータに login して、そこで news の読み書きを行なう。つまりメッセージのコピーは中央のコンピュータにのみ蓄積される。

JUNET の場合はメッセージのコピーが動的に配布される。

☆ 国内分で 1 日に約 100 通が投稿されている。

☆ 海外分も合わせると 1 日に約 580 通が投稿されている。

☆ 国内分のほとんどは日本語で表現されている。

☆ メーリングリストの再投稿が約 3 割。

□ news に自動投稿することにより多くの人に読んでもらえる。

例）X-Window 関係のメーリングリストがあるが、news で配布するようにした。

☆ 投稿量が徐々に増加。

現在では 3000 を越える量が投稿されている。

☆ 電子メール、電子ニュース共、ユーザ・インターフェースは多数揃っている。

JUNET の貢献

☆ 日本にアカデミックネットワークを張った。

☆ 大学だけではなく、民間企業とも接続。

☆ 電子メール、電子ニュースによる交流。

☆ 学会に比較して、ホットでタイムリーな議論の場。

☆ 作業環境の整備。

□ news 用のソフトの日本語化。

□ 上に派生して諸々のソフトの日本語化。

□ PDS、Free Software の品質の向上。

☆ UNIX コミュニティの広がり。

☆ ネットワーク管理や設計のノウハウ交換。

☆ 電子ニュースによる Q & A。

☆ 名刺、論文に電子メールアドレスが入るようになった。

☆ 研究会の開催案内。

☆ LAN を通じて非常に多くの人に利用。

現在の JUNET の問題点

☆ 低信頼性、高遅延なメール配送。

□ 実験ネットワークだから許される？

□ 専用線でつないでみる。

☆ 電子メール、電子ニュースのみのサービス。

□ニックネーム・サーバ（メールアドレス情報）。

☆ 管理作業の増大。

☆ 海外リンクの帯域、費用。

□ 9.6kbps というスピード。

□ 企業が使えない。

□ 料金が高い。

など。

☆ ユーザ、管理者の教育、育成。

☆ 潜在的ゲートウェイの不足。

□ 大企業からの協力がない。大学や小さな会社ばかりである。

☆ 日本人風コミュニケーションサポート。

□ 表現の問題。（けんかになったり、意志が伝わらなかったり）

□ news の保存。

☆ 東京、京阪神への集中。

□ 東京、京阪神以外からの参加が少ない。

☆ 大学の参加が少ない。

□ 全体の 1/3 程度。

新しいサービス

☆ InfoServer など、メールによるアーカイブ

□ あるドキュメントが欲しい場合、メールすると自動で送られてくる．．．など。

☆ ニックネーム・サーバ

☆ リモート・ログイン、ファイル転送、Phone。

☆ 計算機を扱っていない非業界人でも使えるようなインターフェース。

□ UNIX は誰でも使いやすいというものではなく、使いやすいと思っている人
が使っている。

□ UUCP のようなバッチ式リンクでは困難。

† ボランティア・ベースのネットワークであるから、作ってくれる人がいるので
はないかと期待。

JUNET/Wide

☆ 広域ネットワーク環境の実験、研究プロジェクト。

☆ 組織ネットワーク間を接続するインターネットワークの構築。

☆ 分散環境のサポート

□ ネームサーバ
□ 経路制御
□ スクリーニング
□ セキュリティ
□ アーカイブサーバ

などの問題がある。

☆ 種々のリンクのサポート。

□ 9.6kbps 専用線
□ 3.4kHz 専用線 + エラー訂正モデム X.25
□ 64kbps 専用線
□ ダイアルアップコネクション
□ 64kbps ISDN

など。

組織間リンク

☆ 政治的問題が表面化

□ 組織外部の資源へのアクセスは特に問題がないとしても、組織外部からの
アクセスに対して政治的、生理的に嫌悪する人が多い。

・セキュリティ
・イントルーダ（ハッカー）
悪意を持った人をどのようにして食い止めるか。

分散環境へのアプローチ

☆ アイツはどこの計算機で仕事をしているんだ？

□ネームサーバに問い合わせるとつないでくれる。
□ アクセスコントロールも必要になってくる。

JUNET の今後

☆ JUNET である限りは、ボランティアベースの実験ネットワークの正確は保持。

☆ 多くの人の協力が必要。

□ ネットワークの運営
□ 研究開発
□ ユーザ教育

Q & A

Q. ROS(Read Only Site) の問題について加藤さんはどのように考えられますか？

A. これは電子ニュースにおいての話でして、会社や大学などで、電子ニュースの投稿を禁止しているところがあるんですね。そのようなところを ROS と呼んでいます。

で、どうしてこれが問題になってきたかといいますと、みんなが協力してニュースを投稿したり、ソフトを作ったりしているのに、自分のところばかり情報を収集しておいて、一切外に情報を出さないというのはけしからん！！ ということなんです。

これが、今ニュース上で議論されていまして、ROS 問題と呼ばれています。

それで、私の考えなんですが、ROS というのがあること自体よくないとは思いますが、ROS の人達も、ニュースを投稿できるように、いろいろと努力しているので、そんなに責めることはないと思います。

それから、JUNET っていうのにいろいろ誤解があるんですが、例えば、「うちの計算機は JUNET に入っています。」という言い方はあんまり正しくないんですよね。

JUNET を通じてメールを出せる計算機を JUNET の計算機かと言うと、そうではないんですね。それはあくまでその組織の計算機であって、決して JUNET の計算機ではありません。

JUNET を使っていく上では、各参加者の組織のポリシーと、全体のポリシーと受け取る側のポリシーとの共通点で運営していかなければいけないわけです。

それで、私達のやるべきことはその共通点をなるべく広くするという努力をするということだと思います。

ROS の人達もニュースを投稿できるようにいろいろ努力しているわけですから、今、ニュースを簡単に投稿で

きないからといって、いじめてしまうのはよくないですね。

まあ、そういう問題があるっていうことはわかっていますが、あんまり悲観的に考えることはよくないと思いますし、 ROS と呼ばれている側の人達ががんばっているのに水をささないように、なるべく応援してあげるようにしてあげるのが大切ではないかと思います。

Q. INETCLUB へのアクセスは、どのようにすればいいのですか？

A INETCLUB にもいくつか会員種別としてランクがあるわけですが、普通の会員は、ゲートウェイに電子メールに登録しまして、このアドレスは、お金を払いますから通してください、と申請するわけです。

それから、もう 1 つ上のクラスに特別会員というのがありまして、これは、会社のすべての人のメールを通してください、と申請します。

あと 2 年くらいはこういう方向で運営されていくと思います。

Q. 今後の活動として、専用線を引く、というのがあるんですが、なにか具体的な計画はないのですか？

A. それはむずかしいですね。

といいますのは、それにはお金がからんでくるんですよね。 JUNET というのは、基本的には隣の組織までの電話代を払うという以外は、特に課金というのをしなかったんですよね。

それは、メールがデストネーションまで行くのにいろんな経路を通っていったり、場合によっては、経路上のマシンが落ちているので迂回路を通っていったり、いろんな場合がありますので、それら 1 つ 1 つをトレースして、課金データを作成してお金をもらうとかやっていたら、きりがないわけです。

ですから、国内に関してはドンブリ勘定で、隣の組織との間の電話代を出す、ということだけでやってきました。

専用線を使うとうことになりますと、近いところは安くおさまるんですが、ちょっと離れると高くなってしまいます。月に 10 万とか 20 万円くらいかるくかかってしまいますから、非常にむずかしいですね。

私たちができることは、「専用線を使うとこんないいことがありますよ、ですからお金を出してください」、ということくらいなんですよね。

ということで、バラ色のモデルをつくるというのが、私たちに課せられた使命ですね。それが、ほんとうにバラ色であれば、上司も説得しやすいですからね。

ですから、今のところ具体的な計画、例えば 3 年後に全部 IP 化してしまおうとかいうのはないですね。

Q. 加藤さんは学校に行かれて、帰られるまでの間、何をやっていらっしゃるのですか？

A. この質問の意味を「私が JUNET 関係でどれくらいの時間をとられているか」と解釈しますと．．．．

ゲートウェイの管理と、東工大を含む、あるいは自分自身のローカルエリアネットワークも含めたネットワークの管理というので、1 日に 30 分とか 1 時間くらいです。

手間をかけずに高い管理レベルを維持できるという技術を開発することが今、私に課せられた問題であると思います。

Q. JUNET が実験ネットワークから抜け出すということは考えられるでしょうか？

A. それはありません。

JUNET は運用ネットワークの性格もだいぶ持ってきてはいますけど、基本的には実験ネットワークです。

多分 JUNET という名前はずっと実験ネットワークであって、それを他のネットワークが、今 JUNET が置かれている立場をテイク・オーバーするというのはあり得ますけど、私たちは、あまりそんなことは考えていません。

Q. JUNET はボランティアベースのネットワークということなんですが、技術面でサポートできないようなところでも、できるボランティアというのはないでしょうか？

A. 「ボランティア」というのをあまり堅く考えてほしくなかったんですけど．．．。

例えば、JUNET に加入するさいに、強力なボランティアを強制するようでは、誰も加入しませんよね。ですからそれは、自分のできる範囲で構わないと思います。

もちろん、積極的にソフトを出すとか管理をやるとかいった直接のボランティアもありますけど、ニュースを投稿するというだけでも充分ボランティアとして貢献すると思います。

私はあまり狭い意味でボランティアというのをとらえていなくて、もっと広い意味でボランティアというをとらえています。

さっきの ROS の議論なんかも、あれもニュースの中だけのことなんですよね。もっと広いところから見れば、その人たちも何らかの形で貢献してくれているわけですしね。

もし、ニュースを投稿しないことが罪悪だというのであれば、それくらいの小さいところはもっと他にいっぱいあるわけですよね。

というわけで、そういう狭い意味でのボランティアに貢献しなくちゃいけない、というふうには私たちはとらえていません。もっと広い意味で何らかの貢献をしてく

れればいいな、と、また貢献したいという気持ちを持っていてくれたらいいなと、それでいいと思っています。

Q. 「JUNET の文化とはこういうものだよ」というのがあれば紹介していただきたいのですが．．．．

A. JUNET の文化の 1 つは、みんな平等であるということではないでしょうか。

お互い年格好がわかりませんから、誰か権威のある人がなんか言ったからといって、そこで議論が終ってしまうわけではありませんし、漢字コードにしても、○○大学の××先生がこう言ったから、じゃそれにしようというふうに決まったわけではありませんし。

それから、Free Software や PDS を流しているのも、ソフトウェアでゴハンを食べている人には変かもしれませんけど、「ソフトウェアはタダであるべきだ」という主張も少し入っているからなんですよね。

そういったところの融合が JUNET の 1 つの文化を創ってきたのではないかと思います。

割と自由な雰囲気というを保っていけたらな、と私は思っています。

Q. 9 割の電子メールが 6 時間以内に着くということなんですが、サイトの 9 割が東京などの中央に集中していますよね。

ということは、残り 1 割のサイトが地方ということになるんですが、地方に出すメールは、どれくれいの時間で到着するのでしょうか？

A. この場合、距離によって時間がどうだ、というのはありません。

距離よりも、メールが到着するまでに、一体何ホップするかというのが問題になります。

あと HST の場合ですと、SRA との間は 1 ホップなんですが、やはり、遠いからしょっちゅう電話をかけるということもできなくて、 3 時間か 4 時間に 1 回ポールすることになっています。

で、間が悪かったりすると、メールが届くのに、その 3 時間なり 4 時間なり待たなければいけなくなってしまう、というのはありますね。

ワーストケースでは中継するマシンのクオリティというのが効いてきますし、平均ではポールするタイミングというのが効いてきますね。そして、一番早いときは、その人に運があったということですね。

Q. JUNET をつないでいる回線の使用料などが極端に安くなったり、スピードが極端に上がったということを想定したとき、 JUNET にどういう展開が予想されますか？

A. これはものすごいインパクトがあると思います。

例えば熊本から東京まで 64kbps クラスの専用回線でつないだとして、それが、1 万円くらいですむとしますよね。そうなると、熊本の人は東京の人と 1 フロア隣にいるような感じで仕事ができるようになりますね。

それで、今度は遠隔地との共同プロジェクトや共同開発なんかのしくみもできてきたりして、JUNET があたかも LAN の一部になったかのごとく扱えるようになりますね。

そうすると、企業間を超えた分散環境システムという話がでてきたり、それが急速に普及したりして、今までの電子メールや電子ニュースでは、考えられなかったコミュニケーションの手段が出てきたりしますよね。

私たちも、もしみんなに専用線が引けるようになったときにどのようなことをやっていこうか、ということを考えていきたいと思います。

Q. 大学に JUNET を広げるために、大学に対して何か積極的なアプローチはなさっていますか？

A. 積極的なアプローチというか、勧誘は行なっていません。

仮に勧誘したとしたら、私たちに教育する義務が発生することになるんですが、今そこまでカブレませんね。

向こうから「つなぎたいんだけど」と言ってくるのは、向こうにそれなりのモティベーションがあるわけですよね。すると、こちらから「○○を読んで、××をインストールしてくださいね。」と言えば、向こうでやってもらえるわけですよね。

それをこちらから勧誘したりすると、人を派遣してインストールしなければいけないかもしれません。

それは私たちが行なう仕事ではないと思っています。

ほんとうはいっぱいつながっている方がうれしいんですけど、そこまでして勧誘するほどのことではないな、と思っています。

Q. 加藤さんが電子メールを管理する上で、こういうことをやられたら困る、とか、こういう失敗談がある、というのをありましたら教えてください。

A. 困るメールは何かと言いますと、「長いメール」ですね。

たとえば、よくネットワークを知っている人が、自分のマシンから相手先のマシンまでの経路なんかも知っていて、それで判断して送る分にはまだいいんですけど、中継に 2400bps のモデムなんかを使っているのに 200kbyte のメールを送られると 30 分くらいそれでふさがれてしまうんですね。

東工大は、専用のマシンはあるんですけど、電話回線は 2 回線しかないので、1 個ふさがれると他に迷惑がかかってしまいます。

それに、200kbyte のメールが 199kbyte 送った後、残りの 1 kbyte のところで fail したら、もう 1 回はじめから 200kbyte を送り直さなければいけません。

そういうことで長いメールは、いくつかに分けて、1日に1個ずつとかにして送っていただけるとうれしいな、と思います。

Q. 将来 JUNET を IP 化したいというお話があったのですが、 IP のアドレスがパンクするのはもう目に見えています。アドレスの確保は、どのように考えていらっしゃいますか？

A. 今、中クラス以上の大学は、クラス B のアドレスをサブネットで分割して使っていますが、クラス B のアドレスは世界でも約 60000 程度しかとれなくて、近いうちパンクするのは目に見えています。

それで、クラス E というリザーブされている領域を使って、可変長のアドレスをサポートしたらいいのではないかとかいう話もあります。あと、楽観的な見方で、じきに OSI になるのだから、あれは最長 23 億テートまでサポートしていますから、無限にあると思っていいですよね。

ただ、今のところは日本でバルクでガメてきたといってもそんなにたくさんガメてきたわけではないので、ほんとうに今整合しておかないと困る、すぐに IP でつなぐ可能性がある、というところを中心にやっています。

別に審査はやっていません。例えば、○○マガジンや××広報センターに IP アドレスを割り振りますから、メールください、という広報はやっていません。

Q. IP の上位層（トランスポート・レイヤ）というのは決まっているのですか？

A. 私たちは新たなトランスポート・プロトコルを開発する力に欠けているんですよね。

私たちがとるアプローチとしては、多分 TCP をそのまま使うか、あるいは VMTT みたいなものを使うか、というところで、いずれにしても、既存のトランスポート・レイヤの上に、広域分散環境で使いやすいようなアプリケーションを作るというゴールがあります。

あとは、その間にどういう rpc が必要で、トランスポート・レイヤでは足りなかった部分をこうやって誤魔化そうとか、 TCP の上にもう一つトランポート・レイヤをフィードバックしようとか、ありますね。

能率という観点からはまだまだなんですが、一応動くという環境を提示しておいて、徐々に良いものにしていきたいと思います。

Q. JUNET にインターネット・ウォームというものはなかったのでしょうか？

A. 幸いなことに現地時間の 12/2 現在でアメリカの APANET と IP をやっていたところは 1 つしかなかったんですね。そこは、たまたま CSNET がわりと早い時期にマシンを落してしまったので、日本には上陸しなかったんですね。まあ、対策のための伝搬も遅れてしまいましたが．．．．． 一応日本では被害はなかったと思います。

ただ、 JUNET も IP 化すれば、いつでも起こり得ますね。事実 sendmail にちゃんとパッチをあてている人というは少ないしですね。それから、メーカーの意識もイマイチです。というのは、東京で 12 月に UNIX Fair というのをやったのですが、そこで、Exhibition の一環として、ネットワークを張っていました。その説明会のときに「必ず対策を行なってください」と私が言って「OK」と言ってくれたところは少なかったんですね。

具体的な対策を持っていなかった、というのが実情だったんですね。

最後に

SEA九州2年目はすでに始まっています。早く他の支部に負けない内容の会合を持ちたいと思っています。そのためにまた、皆様のお力をお貸しいただけるようお願いすることもあると思います。その時には御協力をよろしくお願いいたします。

SEA九州の会員、67名の方々へ.

SEA九州支部は、より多くの会員の方々の参加をお待ちしています。皆様が抱えておられる様々な問題を出していただき、様々な分野の方達と話を交えながら、その解決に向けての議論なども行ないたいと考えています。時には東京などから講師をお招きし、最新の技術情報なども提供して頂いています。せっかく SEAに入っておられるのですから、SEAをもっと利用して頂きたいと願っています。皆様の積極的な御参加をお願いします。

SEA九州に関する要望や意見などございましたら、SEA九州支部連絡先の平尾(TEL 096-344-2611:HST内)まで御連絡下さい。

ＳＥＡ会員状況（平成元年6月10日現在）

正会員	1644名
賛助会員	58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＝	9	8
青森　＝	0	0
山形　＝	1	1
宮城　＝	6	5
岩手　＝	10	10
福島　＝	1	1
秋田　＝	1	1
新潟　＝	7	7
栃木　＝	6	6
群馬　＝	5	4
茨城　＝	12	13
埼玉　＝	21	119
千葉　＝	18	100
東京　＝	873	522
神奈川＝	134	296
山梨　＝	1	1
長野　＝	36	37
富山　＝	3	3
福井　＝	4	3
石川　＝	3	3
静岡　＝	19	19
岐阜　＝	2	8
愛知　＝	56	44
和歌山＝	2	3
三重　＝	1	4
滋賀　＝	9	10
京都　＝	21	31
大阪　＝	220	164
奈良　＝	4	14
兵庫　＝	35	65
愛媛　＝	2	2
徳島　＝	6	6
高知　＝	0	0
香川　＝	0	0
鳥取　＝	0	1
島根　＝	1	1
岡山　＝	2	2
広島　＝	5	5
山口　＝	0	0
福岡　＝	18	20
大分　＝	6	6
熊本　＝	34	33
長崎　＝	0	0
宮崎　＝	1	1
佐賀　＝	1	1
鹿児島＝	5	5
沖縄　＝	1	1
韓国　＝	1	1
米国　＝	4	4
豪州　＝	1	1

＜男女分布＞

男	＝	1560
女	＝	84

＜年齢分布＞

20以下	＝	0
20_24	＝	37
25_29	＝	315
30_34	＝	444
35_39	＝	380
40_44	＝	284
45_49	＝	115
50_54	＝	32
55_59	＝	19
60以上	＝	16

＜血液型分布＞

A型	＝	632
O型	＝	479
B型	＝	360
AB型	＝	173

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，リコーシステム開発，ＳＲＡ，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，日本能率コンサルタント，構造計画研究所，ケーシーエス，サンビルド印刷，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，エムテイシー，クレオ，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，マイクロリサーチアソシエイツ，松下ソフトリサーチ，シャープ，東洋エンジニアリング，日本ユニシス・ソフトウェア，清鳳オフィスメーション，新日鉄情報通信システム，テックシステムズ，熊本電子計装工業，ＮＴＴ九州技術開発センタ，立石ソフトウェア，三菱電機関西コンピュータシステム，日立エンジニアリング，スインク，テスク

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