SEAMAIL

Newsletter from Software Engineers Association

Volume 11, Number 2 September, 1997

目　次

編集部から		1
第 13 回 若手の会 報告	中谷 多哉子	2
SS-97 基調講演 および SEA Forum June	Alex L. Wolf	28
Does Better Process Lead to Better Product?		29
Software Architecture		48

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

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

その主な活動は，機関誌SEAMAILの発行，支部および研究分科会の運営，セミナー／ワークショップ／シンポジウムなどのイベントの開催，および内外の関係諸団体との交流です．発足当初約200人にすぎなかった会員数もその後飛躍的に増加し，現在，北は北海道から南は沖縄まで，700余名を越えるメンバーを擁するにいたりました．法人賛助会員も30社を数えます．支部は，東京以外に，関西，横浜，長野，名古屋，九州，広島，東北の各地区で設立されており，その他の地域でも設立準備をしています．分科会は，東京，関西，名古屋で，それぞれいくつかが活動しており，その他の支部でも，月例会やフォーラムが定期的に開催されています．

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

代表幹事： 坂本啓司

常任幹事： 荒木啓二郎　高橋光裕　田中一夫　玉井哲雄　中野秀男　深瀬弘恭

幹事： 市川寛　伊藤昌夫　大場充　河村一樹　窪田芳夫　熊谷章　小林修　桜井麻里
酒匂寛　塩谷和範　篠崎直二郎　新谷勝利　杉田義明　武田淳男　中来田秀樹
布川博士　野中哲　野村行憲　野呂昌満　端山毅　平尾一浩　藤野誠治　二木厚吉
堀江進　松原友夫　山崎利治　和田喜久男

事務局長： 岸田孝一

会計監事： 辻淳二　吉村成弘

分科会世話人 環境分科会(SIGENV)：塩谷和範　田中慎一郎　渡邊雄一
教育分科会(SIGEDU)：君島浩　篠崎直二郎　杉田義明　中園順三
ネットワーク分科会(SIGNET)：小林俊明　人見庸　松本理恵
プロセス分科会(SEA-SPIN))：青山幹雄　伊藤昌夫　坂本啓司　高橋光裕　田中一夫　増井和也

支部世話人 関西支部：臼井義美　中野秀男　横山博司
横浜支部：野中哲　藤野晃延　北條正顕
長野支部：市川寛　小林俊明　佐藤千明
名古屋支部：筏井美枝子　石川雅彦　角谷裕司　野呂昌満
九州支部：武田淳男　平尾一浩
広島支部：大場充　佐藤康臣　谷純一郎
東北支部：河村一樹　布川博士　野村行憲　和田勇

賛助会員会社： ジェーエムエーシステムズ　東芝アドバンストシステム　SRA　PFU
東電ソフトウェア　構造計画研究所　さくらケーシーエス　サン・ビルド印刷
富士通　ジャストシステム　新日鉄情報通信システム　オムロンソフトウェア
カシオ計算機　キヤノン　中央システム　安川電機　富士通エフ・アイ・ピー
SRA東北　アスキー　新日本製鉄エレクトロニクス研究所　ダイキン工業
東北コンピュータ・サービス　オムロン　アイシーエス　SRA中国　日本電気ソフトウェア
富士電機　ブラザー工業　プロダクト・ソリューション　(以上29社)

SEAMAIL Vol. 11, No. 2　　1997年9月3日発行
編集人　岸田孝一
発行人　ソフトウェア技術者協会（SEA）
〒160　東京都新宿区四谷3－12　丸正ビル5F
T: 03-3356-1077　F: 03-3356-1072　sea@@sea.or.jp
印刷所　サンビルド印刷株式会社　〒162　東京都新宿区築地町8番地
定価　500円（禁無断転載）

編集部から

☆

お約束通り，船便シンドローム脱出を目指してのSEAMAIL Vol.11，第2号です．

☆☆

企画運営をどうするかで議論が長引き，しばらく間隔のあいた第13回SEA若手の会ワークショップは，今年の1月末に，ヤングタウン静岡で開催されました．プログラム委員の1人: 中谷多哉子さんにその報告をまとめていただきました．中谷さん，どうもありがとうございました．

☆☆☆

なお，次の若手の会については，盛岡の野村行憲さん(ICS)を中心に，幹事会メンバおよび過去の若手の会関係者有志でMailing Listを作り，これから企画の議論が始まるところです．興味のある方は，waka-sc@sea.or.jpに連絡してみてください．

☆☆☆☆

6月のソフトウェア･シンポジウムにおけるAkex Wolf先生の基調講演は，先日お配りしたプロシーディングスには入っていません．そこで，当日使われたOHP資料のコピーを，SEA Forum Juneの講演資料と会わせて収録しました．

☆☆☆☆☆

第13回　若手の会報告
–ソフトウェア技術者のサバイバル戦略–
1997.1.22-1997.1.25
静岡

中谷 多哉子* 酒匂 寛† 谷口 悟‡ 中来田 秀樹§ 保坂 利夫¶

1　はじめに

静岡で1997年1月22日-25日の4日間に渡って第13回SEA若手の会ワークショップが開催された．今回はソフトウェア技術者として，21世紀をどのように生き延びるかを大テーマに掲げ，グループ毎に討論を進めた．この報告書では，会が開催される前に提出されたポジションペーパをもとに，技術者が「若手の会」で何を考え，何を感じたかを参加者の生の声をもとに編集したものである．全体の構成は次のようになっている．

1. はじめに
2. 若手の会概要
3. 準備（ポジションペーパ）
4. 山崎利治氏 招待講演概要
5. Aグループ成果報告
6. Bグループ成果報告
7. Cグループ成果報告
8. プログラム委員報告
9. おわりに

2　若手の会概要

- 期間：1997年1月22日～25日
- 開催場所：静岡市商工会議所の会議室
  静岡市黒金町20番地8

*Sラグーン
†Designers' Den
‡日本電気(株) C&C システム SI技術本部 販売推進部
§(有) ネクストファウンデーション
¶足利システムサービス(株)

時間に制約がなく使用できたのはよかった．買い出しの便もよかった．ビルを出れば「にんにく料理」や「桜海老」,「とろろ」の店などもあり，環境はまずまず．

- 宿泊場所：ホテルシティオ静岡
  静岡市伝馬町1-2

  静岡という土地柄，夜は早めにお店が閉まるので，夜、ホテルを出て，ちょっと軽く何かを食べようとすると，同じ店内に他のグループがいたりした．

- 参加人数:「若手」15名，プログラム委員5名，招待講演者1名(山崎 利治氏)
- 招待講演：山崎利治氏の講演
- スケジュール

  – 準備
  参加者のポジションペーパ提出

  1/22 13:30- 集合，オリエンテーション
  14:30- グループ討論
  簡単なオリエンテーションの後，各グループに別れて討論を行い，ワークショップで討論する議題を決定した．各グループとも，同世代の技術者との交流と各自のポジションペーパの意見交換や「サバイバル戦略」という課題の捉え方から討論が始まったようである．

  1/23 9:00-11:30 グループ討論
  13:00-14:00 各グループ中間報告
  14:00-15:00 招待講演：山崎利治氏
  15:00- グループ討論

17:00-19:00 懇親会

中間報告では，活発な質疑応答が行なわれ，招待講演も好評であった．

1/24 終日グループ討論

1/25 9:00-12:00 各グループ最終発表

3 準備

3.1 質問票

ワークショップの参加者は，次の質問票に対する回答をポジションペーパとして提出した．

【ポジション・ステートメント】: 以下の 3 項目の質問にお答え下さい

Q1. あなたが今持っている「ソフトウェア技術者」のイメージは?

Q2. 急激な技術進歩の渦中 に置かれている「ソフトウェア技術者」として，日常どのようなこ とを準備・実践して いますか? または準備・実践したいと思っていますか?

Q3. 上記の実践において障害となっていること，なるであろうことは何ですか？

3.2 ポジションペーパ (五十音順)

3.2.1

(株)SRA 中部支社
石川 雅彦

A1 技術者のイメージ

- ユーザの要求仕様をまとめることができる
- 陳腐化していない技術を取り入れて設計ができる
- 必要であれば，実装の細部 (プログラミングレベルなど) を理解し，必要な支援を実装者に対して行うことができる

A2 日常の準備と実践

- 情報の収集—本，メーリングリスト，ネットニュース，WWW を通じて社外組織へ参加し，刺激をうける
- 他人のプロジェクトで何をしているか知ること
- 実作業を行なう—OS やアプリケーションのインストール，小規模なプログラミングを行う，新しい機器はとりあえず触ってみる

A3 実践上の障害

時間，能力，経済力．しかし，今持っているそれらの資源 (時間，能力，経済力) を自分に与えられた条件と考えれば，それらの資源を使って得た知識を交換しあえる仲間が不足しているのではないか．

3.2.2

オムロン (株) EFTS 統轄事業部開発センタ
第 2 開発部第 2 開発課
上田 秀樹

A1 技術者のイメージ

解決すべき問題領域から，適当なモデルを作成し，それを実装する．この一連の作業をうまく行える人．

A2 日常の準備と実践

ソフトウェアを利用するにあたって，いきなり新技術を使うというのはかなりリスクの高いことと思います．もしもこれらの技術を使ってうまくいかなくなったとき，情報量が少ないために容易に解決できないという可能性もあります．急激な技術進歩をやみくもに追いかけるのではなく，すこし身を引いて，とりあえず使ってみて，その状況をレポートしノウハウを蓄積することが必要と思われます．

A3 実践上の障害

ノウハウの蓄積にどのくらい時間をかけることができるかという，時間の制約でしょうか．あまりかけすぎると，新しい技術に追いつけなくなるかもしれませんし．

3.2.3

山一情報システム (株)
オープンシステム第 1 部
神山 純子

A1 技術者のイメージ

ハードもソフトも進化が激しく，それに追いついていくことが難しくなっています．今のそして，これからのソフトウェア技術者は，これから発展していくであろう技術をいち早くキャッチして、それを身につけ実際の業務に役立てていくべきであると思います．その先見の明を持った人は，年齢などは関係なく仕事をしていく上での実力者になると思います．私は，そのような技術者になりたいです．今のソフトウェア技術者は新しい多量の情報と昔からのシステム設計技法の間に入って模索しているような気がします．

A2 日常の準備と実践

雑誌，インターネットなどの多量の情報の中から本当に自分に必要で，これから役に立つであろう情報を抽出できるようになりたいです．そして，それを見極めてその技術を習得したいです．そのために，情報媒体に積極的に目を通すようにして情報を受信できるようなアンテナ (その技術を理解できるような技術を身につける) をはるようにしています．

A3 実践上の障害

日本よりもアメリカのほうが，ソフトウェア技術は進んでいると思います．英語を自然に聞き，話し，読めるようになりたいです．そのために，英語力を身につけなくては ….

3.2.4

(株)SRA SI2 部 2000 年対応グループ
許 林興

A1 技術者のイメージ

「ソフトウェア技術者」って名前の通り，ソフトウェアの技術をたくさん持っている人たちを指します．プロジェクトの開発を担当できる人たちも当然「ソフトウェア技術者」の範囲に属しています．しかし，ソフトウェアに対して，ハードウェアの知識がちょっと不足かなあというイメージがあります．そして，ソフトウェア技術者は市場に対する経験と敏感性に関しても欠けているところがたくさんあると思われます．

……

これは私が今，もっている「ソフトウェア技術者」のイメージです．

A2 日常の準備と実践

1. 「ソフトウェア技術者」として，もちろん新しいソフトウェア技術をいち早く身につけるということ．
2. 激しいコンピュータ市場意識の養成：市場はどんなソフトがほしいか．また，どんなソフト製品が売られるか．「ソフトウェア技術者」としての私たちも考えてみたい．
3. ソフト製品は単純な製品 (普通の電器製品) ではなくて，生命製品と言えるかもしれませんが，私たちに開発されたソフト製品の生命周期はどのようにすればできるだけ長く伸ばせるのかについて，日常の仕事で考えてみてほしい．例えば，2000 年問題などの古いマシン (汎用機) の生命はほぼ終ると言えるでしょうか．対応，修正しても，たくさんの人力，物力，財力がかかるでしょう．ですから，ソフト製品の開発の最初からできるだけこのような潜在的な問題にも気を付けなければなりません．

A3 実践上の障害

私は外国人として日本のコンピュータ産業を考えてみましたが，驚いたことに気づきました．電器製品 (テレビ，ビデオ，冷蔵庫...) のほとんどが “Made in Japan” で，世界でも大きなシェアを持っている日本の電器産業に対して，ハードも基本ソフトも外国製，応用ソフトも日本語ワープロ以外の主要なパッケージはほとんど輸入品という現状に陥っている情報産業との間の差はとても大きい．この原因は何か？Windows の世界でまだ別の Windows(あるいは Doors) が必要でしょうか．インターネットの風に乗ってなにか新しいチャンスを生じるか．私は時々これらのことに考えさせられます．そして,「ソフトウェア技術者」として，一番やるべきことは何か.「ソフトウェア技術者」自身の生命周期はいつまでか．是非，今回のワークショップを通じて，皆さんのお話しを伺いたいと思います．

中国の情報産業 (特にソフト産業) はまだ未熟ですが，これからだとも言われています．これらの問題を考えながら,「ソフトウェア技術者」の一員として，是非，若いころ，多くの新しい技術を学んで，いろいろな面で実践してみて，将来の道をちゃんと見つけられるために今から準備しなければなりません．

3.2.5

(株)SRA オブジェクト指向グループ
阪井 誠

A1 技術者のイメージ

自ら進んで新しい技術を着実に身につけ続ける．プロジェクト全体を見渡して物事を考えられる．お客様に提案ができる．

A2 日常の準備と実践

現在抱えているプロジェクトだけに閉じこもらないように新しい技術の動向は常に押さえるように心がけている．

さらに1歩踏み込んで仕事と関係ないプロジェクトを仮想的に実施して新しい技術を身につけたいと思っているが仕事の関係でなかなかできない．

A3 実践上の障害

情報提供媒体が余りにも増えすぎたため，ノイズの少ない情報源を得ることが難しく，年齢的に体力の限界が見えつつある．

立場的に自分の時間が持ちにくいので，手を動かして技術を習得することが難しい．

3.2.6

オムロンソフトウェア (株)
企画室 経営計画グループ
阪長 俊之

A1 技術者のイメージ

顧客の要求を実現するためにソフトウェア関連のあらゆる技術を動員してプログラムを作る人

A2 日常の準備と実践

- 最新の技術動向/情報の収集
- 組織的ソフトウェア作成の仕方を学ぶ

A3 実践上の障害

- 技術動向の収集には限界がある
- ソフトウェア工学や人間工学の組織的研修が行われていないし，自己流の習得は難しい
- 知らない→外注ばかりになる→空洞化する→身につかないの悪循環になるおそれがある

3.2.7

Designer's Den
酒匂 寛

A1 技術者のイメージ

「設計」をする人「アーキテクチャ」を決める人

A2 日常の準備と実践

流行に左右されない基礎技術からの観点を重視する

A3 実践上の障害

Tools for Thought の欠如

うつろいやすく脆弱なプラットフォーム

3.2.8

(株) 日本 NCD 技術部
鈴木 勉

A1 技術者のイメージ

暗い，風変わりな人，残業が多い

A2 日常の準備と実践

多くの情報を得ること，そしてなるべく自分で試してみること

A3 実践上の障害

時間の制約と自分のやる気

3.2.9

オムロン (株) ESB
園田 修司

A1 技術者のイメージ

- 楽をすることに努力する
- 本質を見極める
- 自由な発想とこだわり

A2 日常の準備と実践

- よく考え，はっきりと意見をいう
- 会社に依存しない
- 手に職をつける

A3 実践上の障害

- 会社や職場の風土
- 時間の確保
- 身近に教えてくれる人が不在 (自分の能力 :-)

3.2.10

(株)NEC C&C システム
SI 技術本部 販売推進部
谷口 悟

A1 技術者のイメージ

以下の 3 つの能力を保有している人に接すると「この人はすぐれた技術者だ」という印象を受ける.

- 情報収集力〜現有の知識・ノウハウを常に持っていなくてもいいが，必要な情報・ノウハウをその場その場で学んでいける人が結局「良い仕事」ができているのではないか.
- 思考能力〜優れた技術者は提案書，仕様書一つにしてもあらゆる方面まで考え抜いて作成しているように思う.
- 基礎知識〜最新の技術動向はよく知っているが，ソフトウェア等に関する基礎知識がない人が私の部署には若干存在する (特に若手). 彼らは技術的につっこんだことが記してある文書に接するとある程度以上は理解できなくて苦労している.

A2 日常の準備と実践

- 基礎知識のリフレッシュ〜私がソフトウェア技術を学び始めたころにはなかった技術・ノウハウが最近登場している (特にネットワーク回り). 仕事の合間を見て専門書・雑誌等を読み返してこれらについて再確認している. 本当は，自分でもこれらの技術を研究してみるべきだが，業務内容とかけ離れてしまうため，現在のところできていない.
- 教育情報工学のノウハウの活用〜以前学んでいた教育情報工学の各種ノウハウ (現在の仕事の前は教育関係の仕事だった) を活用して情報整理に活用している. ノウハウは文書化することにより，早く習得できるように思う.

A3 実践上の障害

- 「販売推進部」に在籍しているため，技術情報はほとんどドキュメントベースで，「実物」に接する機会はほとんどない. 建前のみ頭に入れて現場でのソフトウェア開発作業に今後活用できるのかどうかが疑問.
- 社内・社外ともに，一番接するのが「広報資料」となっている. これらの資料には，実際にはまだ実現できていないことが記してある等誇張が多く，事実が把握しにくい (特にシステム構築事例等).
- 現場で実際に開発作業を行っている第一線の方から技術情報を収集しようとしている.「販売推進部」でありながら，営業活動，システム提案のノウハウを保有していないため，社内 (場合によっては社外) の優れた事例を収集する (来年から本格的に活動開始).

3.2.11

日本電子計算 (株)
マルチメディア事業プロジェクト
永岡 栄吾

A1 技術者のイメージ

コンピュータについての充分な知識をもった技術者ともいえるが，最近ではいつまでも少ない技術で稼いでる集団とのイメージもある．

A2 日常の準備と実践

技術雑誌等で取り上げられている技術については，必ず目を通したり，Web 上に置かれてある情報をチェックしておき必要な時に参照できるようにしている．

A3 実践上の障害

日常の仕事や雑用が一番の障害となっており，時間が取れないことが多い．

3.2.12

(有) ネクストファウンデーション
中来田 秀樹

A1 技術者のイメージ

日々切磋琢磨している人物

A2 日常の準備と実践

言っていることとやっていることが多少の異なるのは仕方ないとして，できるだけ情けないことをしないで生きていければと考えます．

A3 実践上の障害

やりたいこととやらなければならないことの差が障害だと考えます．ただ，それは殆どが自分自身に起因していることですので，やはり日々の精進 (修行) が足りないことになるのだと認識しています．

3.2.13

S ラグーン
中谷 多哉子

A1 技術者のイメージ

ユーザの要求を理解することよりも，なぜそのような要求を出しているのかを理解し，要求を創造してあげられる人．(注：ねつ造ではない)

A2 日常の準備と実践

日経 XX は読まない：読むとそれが実際に今起きているかのように見えるが，そう見えてはもう技術者として遅れていると思う．またニュースは単なる大騒ぎである可能性もある．そういうことに惑わされてはいけない．

WWW は見ない：メールと同じでとてつもなく時間が消費される．その時間があったら他に創造的なことをやれ... といつも自分に言い聞かせている．

... とは言え，

新鮮な情報があることは否めませんが... 悩みですね，ここが．絞り込めばよいのでしょうが，絞り込むと，やっぱりすぐに情報源が陳腐化するし... WWW をいつも見ている友人と仲良くして，その友人から情報を貰って必要なら情報源を確認するという方法が妥協点かもしれません．

ただ，技術進歩が早いといっても，同じスピードで進歩しているわけではなく，S 字の曲線を描いていますよね．すべての技術に精通する必要はない，と私は思っているのですが，新しい技術や古い (?) 技術が，今，S 字のどの段階にあるか位は知っている必要があると思います．そういう情報を入手するには，いろいろな分野の専門分野の方々と話す機会を積極的に作るのが良いのではないでしょうか．それらの方々からは，今後の重要な技術になるとか，あるいは世間はレミングしているだけだ，とか，本当の情報を入手できることがあります．つまり，これが S 字曲線上の位置づけを捉えるための情報入手です．もっとも，こういう情報を入手するためには自分も専門分野についてだけは情報発信できるような技術者になる必要があると思うのですが，どうでしょう．つまり，WWW を見たり，日経 XX を読んだりする時間があるのであれば，自分の専門分野の勉強をした方がよいのではないかと... こういうわけです．

「専門は?」と聞かれたとき，さあて，何と答えましょうか．これをいつも自問自答することが，質問の「実践」の内容でしょうか．あまり答えになっていませんね．ワークショップで練りたいと思います．

A3 実践上の障害

「遊びたい」「本を読みたい」「映画を観たい」「おしゃべりをしたい.」でも,「仕事をやらなきゃ」といった時間消費型業務の競合.

とにかく時間が足りません. 深く掘ったつもりの専門分野が1年たってみたら上っ面になっていた, ということに気が付いて愕然としたりして. こういう脅迫観念も, 実は,「私の専門は...」と胸を張って言うための障害になっているかもしれません.

3.2.14

富士通 (株)
第3システム事業部品質システム部
藤坂 政志

A1 技術者のイメージ

苦労が多い

A2 日常の準備と実践

社内教育受講が中心

A3 実践上の障害

たくさんの技術情報があり, ほんとうに必要な情報をさがすのが大変

3.2.15

足利システムサービス (株) 開発一部業務課
保坂 利夫

A1 技術者のイメージ

「理想は」

私自身が「こうありたい」と願う「ソフトウェア技術者」像は,

1. プログラムを「書く」ことについては, いつまでも現役であり続けている人.

   当然, 現在トレンドとなっている言語については, 取りあえず「書ける」し, 言語ごとのポイント (押さえるべき点) は, 把握している. 仕事と自分の能力開発の方向についても, 適度なバランス感覚で調整できる. (やな仕事ばっかりとか, 自分の時間が作れない等の「たわごと」はいわない.) 当然作成するプログラムも「一流」である. 私の頭の中での「一流プログラマ」とは, 明示された品質を達成するのはもちろん, 明示されていないものや, 潜在的なものまでも具体化できるひと. ちなみに「二流」は, 明示された (または明示させた) 品質を達成するのみ.「三流」は, とにかく動きます. という程度. (私のプログラムは, きっとこのレベルだったと, 今は思っています.)
2. 言葉での表現は難しいが, アプリケーションソフト関係の表現で言うと,「業種を越えた, ソフトウェア作成の知識」がある人.

   たとえば, お菓子屋さんの社長がコンピュータメーカーの社長になれるように, 分野を越えた才能 (英語では「capability(手腕)」と言うそうですが) を持っている人. (すいません. 自分でもよく分からないけど)
3. とにかく「凄い人」.

   言葉通りです.

「現実は」

そうは言っても現実の「ソフトウェア技術者」は, どう見えるかというと,

1. 残業もビシバシして, 納期の無いプロジェクトをどんどんこなすが, 新しい技術や, 知識を付ける暇も, また, その気も起きない人. (でも, 新技術が必要な仕事でも, 臆せずやる.) ほとんど「体力」勝負を勝ち残る人. 当然自負心も強く,「仕様漏れ」はすべて顧客や依頼者の責任となる.「言った, 言わない」の口論を, 当然のごとく勝ち残る才能がある. 当然残業時間の長さは問題ではない. 一見顧客の要望を満たすために顧客本位で活動しているように見えるが, 仕事が延々と途切れない. 言葉を変えると,「彼のタッチするプロジェクトは, 社内的には大赤字」と言うことが間々ある. (私も一時こういうことがあったが, 今考えると, 単なる見積り違いだったのかなあ, と反省している今日このごろです.)
2. 「コスト」意識が高く, 慎重なタイプ. 自分で「見きる」ことができないと, プロ

ジェクトに参加したがらない．自分の技術基盤や知識の範囲で考えるので当然，「新技術」については消極的．(ただし，いったんタッチすると，品質は高い．個人的には，こういうタイプの人は好きですが，企業体の中では，目立たない人になっていたりします．)

ちょっと分類が「甘い」と自分でも思いますが，ワークショップの中で考えていきたい．

A2 日常の準備と実践

「実践していること」

1. SEA に入会したこと
   まだ，効果の程はわかりませんが.....
2. プライベートで「技術習得」の時間やコストをかけている．
   (実際「捨て金」になることのほうが多いけれど..)
3. 「何とかしなくちゃ！」と考えている．
   (あせるばかりで，何も進まないのが問題．)

「実践したいと思っていること」

1. 「転職」
   自分を追い込むにはいい方法だとは思うが，なかなか決断は難しい．
2. SEA の活動に積極的に参加して，視野を広げる．
   まだ，団体の性格が見えないので，消極的だが．

A3 実践上の障害

1. 所属企業体以外での活動を促進する土壌がない．たとえば，技術者の交流の場としてSEA を考えた場合，非常にいいと思うが，現在の所属企業には，それを積極的に活用する土壌と人がいない．私自身は比較的自由に行動が起こせる立場ではあるが，この環境が 20 代から提供できるようになれば，現在の社員のスキルももっと違ってくると思う．そのためには所属企業体を離れた研鑽の場が必要だが，難しい．
2. 画一的な教育になれてきた企業では，個人ごとのスキル管理や，教育ができない．(まだまだ，なじまない)
   企業体としても，技術者の個人スキル管理がまだできないし，ましてや，個人別のカリキュラムなどは，不可能．本来は，各人が自分の目標を定め，それをサポートできればいいのだが，社員もなれていないし (サラリーマン化してるのも事実だが,「目を覚ます指導」ができていないのも事実．私自身も見えない訳だから，当然かも．)，企業もなかなか小回りがきかない．
3. 中小零細企業では，日々変化する技術を追いかけて，投資する資金的余裕も時間もない．ましてや，社員にたいして個別対応が取れない．
   まったく視点が違うが，複数の企業体で技術者を相互活用する形態ができれば，もっと技術者育成や企業ニーズが一致するかなと考えるときもある．

以上，ちょっと「支離滅裂」なところもありますが，自分の「うやむや」していることを書きました．

よろしくお願いします．

3.2.16

(株)SRA 先端技術応用グループ
堀口 利子

A1 技術者のイメージ

日進月歩するコンピュータ (ハード及びソフトウェア) 技術に乗り遅れないよう必死で対応している？最新技術を開発している技術者は一部で，多くの技術者は急激に進歩していく技術を身に付けるのに苦労している．

A2 日常の準備と実践

- 視野を広く持ち，様々な観点からソフトウェア技術は勿論のこと，その他の技術や情報も見る．
- 最新情報に敏感でいて，その情報収集を怠らないようにする．

- 自分が得意とする分野について専門知識を深める.

A3 実践上の障害

- あまりに情報が多く自分自身が本当に必要とする情報を得ることが大変になっている. そして, あれも, これもと欲張ってしまい, 結局中途半端な知識になってしまう.
- 自分自身の興味, 目標と技量のギャップ.
- 自分自身の興味, 目標と実作業のギャップによる時間的制約.

3.2.17

(株)SRA 中部支社
諸岡 隆司

A1 技術者のイメージ

ソフトウェアの作成, 運用に携わる技術者-毎日, とても忙しい技術者

A2 日常の準備と実践

- 新しい (足らない) 技術の習得
- 新しい情報の収集
- いろんな角度から物事を考えられるようになること

A3 実践上の障害

- 仕事
- 収入
- 環境

3.2.18

山一情報システム (株)
オープンシステム第一部
山崎 望未

A1 技術者のイメージ

設計屋さん. 多くの材料 (アプリケーション) を組み合わせて, ひとつの商品 (システム) を作りあげるというイメージがあります.

A2 日常の準備と実践

常に最先端の情報を得るために, 多くの雑誌・書籍を読む必要があると思っています. また, その中で必要となる技術を選び出し, 実際に使いこなせるなりたいと思ってます.

A3 実践上の障害

まず, 日々の業務に追われて勉強の時間がなかなかとれないことです. また勉強しなければならない技術, 身につけたい知識等の数が多すぎて, 取捨選択が非常に困難です.

3.2.19

東京電力 (株)
システム企画部システム開発センター
山家 昇

A1 技術者のイメージ

システムアナリスト (SA), システムエンジニア (SE), プログラマなどを超えたコンピュータシステム, 情報処理のエキスパート.

進歩の早い技術分野なので, 情報収集のために常に多忙のイメージがある.

A2 日常の準備と実践

- 新技術, 新製品の情報収集のため, 雑誌, 新聞にひととおり目を通す.
- 他社事例が参考となる場合があるので, ユーザ会などで知り合った人脈を大切にする.
- ソフトウェア技術だけによる解決でなく, 経営の視点から業務の効率化を目指した仕事の簡素化・割り切り, 業務革新を提案していきたい.

A3 実践上の障害

- 上記の実践において障害となっていること.
- 私個人の問題として, 英語の読解力がないこと.
- (業務システムを開発する際に障害となっていることですが) 新製品のリリースが米国に比べて遅くなるケースがあること.

(日米同時リリースくらいのサービスをして欲しい)

- ＊自己紹介

  情報システム部門の経験は 7 年 3ヶ月で，この内訳は以下のとおりです．

  - コンピュータ設備計画：約 4 年
  - 情報システム企画/計画：約 3 年
  - システム開発管理：3ヶ月

  従って，システム設計/開発に関わるソフトウェア技術の経験は，あまりありません．

3.2.20

東電ソフトウェア (株)
システム技術部開発技術グループ
吉澤 賢一

A1 技術者のイメージ

これといってなし

A2 日常の準備と実践

情報収集と試行．具体的には Web 検索，雑誌類の購読．セミナー参加．ただし，どれもうのみにしないこと．目の前にある興味対象は必ず自分で試す．

A3 実践上の障害

受託業務でしょうか．やはり時間的な制約でしょう．

4 山崎 利治氏 招待講演概要

報告：谷口 悟

1. はじめに：略歴等

   コンピュータの創世記にユニバック (現：ユニシス) にてプログラミング作業を担当した．当時はプログラミングは配線作業であった．その後 OS が整備され，コンピュータはますます扱いやすくなった．

   とは言っても，プログラムはそれを書くことを専門にしないと，なかなかできないものである．800 行程度のプログラムですら，素人ではプログラムが拡散してしてまとまらなくなるに違いない．玄人には仕様が必要であり，構造化やアーキテクチャーも必要である．また，プログラムの反復使用を考えるのであれば，そのための考慮も必要となる．

   素人でも，ある程度作成可能という点では，プログラミングは料理に似ている．

2. ソフトウェア技術者として

   - 基礎を学ぼう

     今回の参加者の中には，実戦技術不足を感じている人も多いと思う．だが，技術は基礎からの積み上げが必要である．基礎なしで技術を学んでも応用・類推が通用しなくなり行き詰まる可能性が大きい．しかし，根本原理を見直すことができれば，うわべの技術の流行に惑わされなくてすむようになるだろう．

   - 例

     仕様の記述には文章・図が多いが，仕様を記述するために Z や BNF をいずれ知る必要が出てくる．コンパイラをなぜ学ぶ必要があるのかと思うかも知れないが，一度コンパイラの基礎が分かるとプログラミング言語も学びやすくなるものである．

   - 参考文献

     岩波講座 ソフトウェア科学シリーズ 全巻, 岩波書店

     基礎技術習得のために，全部は無理としても，一部だけでも一読をお薦めする．

5 A グループ成果報告

報告：許 林興

参加者：石川 雅彦，阪長 俊之，山崎 望未，山家 昇，許 林興
プログラム委員：酒匂 寛

5.1 討論の軌跡

5.1.1 テーマ

「ソフトウェア技術者のサバイバル戦略」というテーマで，討論すべきことは次に挙げるようにたくさんあると思います．

- ソフトウェア技術者と良いソフトウェア技術者
- ソフトウェア技術者の道
- ソフトウェア技術者と組織 (環境)
- ソフトウェア技術者のやりたいこととやらせること
- .....

A グループは，最初に「技術者」という立場から現在の日進月歩の技術に対して，自己の位置付けと取り組み方法について議論し，問題を洗い出して，最後に我々の考えと答えを出そうという議論の流れに合意しました．

ソフトウェア技術者のサバイバル戦略を考える上でまず，自分自身の位置付け–つまり今，自分がどこにいるのかを知らなければなりません．そして，現実のソフトウェア世界に直面しなければなりません．我々は日進月歩のソフトウェア世界を感じています．同時に時々，この世界に戸惑っています．今，一番の問題，あるいは戸惑っているところをまとめると，以下のようになります．

- 技術進歩が速いことに戸惑う
- ソフトウェア技術者としてどうすればいいのかに迷う
- 情報量が多過ぎることに戸惑う

しかし，進歩が速いと言われている技術の実体は何でしょうか？本当に私たちに必要な技術の全てが速い進歩を遂げているのでしょうか．もしかしたら，私たちは，正体をつかむ前に漠然とした危機感に襲われているだけかもしれません．対処を考える前にまず，それらの正体をつかむことが先決だと思います．そのために私たちは以下のような戦略を考えました．

5.1.2 討論

- 戦略その 1 : 普遍的な知識を身につけ，技術者としての基礎を固める

  確かに日進月歩のソフトウェア世界に対して，どうすれば厳しい生存競争に打ち勝つのかを迷っている技術者が少なくありません．しかし，よく考えてみれば，良いソフトウェア技術者になるために基本的な知識と専門的な知識

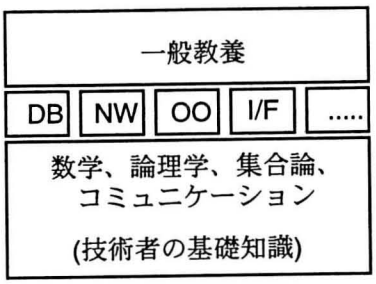


図 1: ソフトウェア技術者に求められる能力と知識

をうまく身につけ，うまく自分の仕事に実践できれば，大丈夫ではないかというのが我々の考えです．

図 1に示すようにソフトウェア技術者に求められる能力，知識には基礎知識と専門知識があります．その中で，技術進歩が速いのはほとんど表層的な知識ではないかと思われます．例えば，新製品情報，URL，新しい言語，インターネットなど．一方その深層にある知識=原理，原則 (基礎的な知識) はあまり変わらないと思われます．

例えば，ソフトウェア工学に関する以下の本は 20 年以上も前に刊行されたものですが，現在でも読まれ続けています．

- F. P. Brooks, Jr. : The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley, 1975 (山内正彌 訳: ソフトウェア開発の神話, 企画センター, 1977)
- J. M. Weinberg : The Psychology of Computer Programming, Van Nostrand Reinhold A Division of Wadsworth, 1988 (木村泉他訳: プログラミングの心理学, 技術評論社, 1994. 原著は 1971 年に発刊された「ソフトウェア工学」の講義テキスト)

また，情報量が多すぎることについては今に始まったことではなく，以下の本が指摘しています．この本は 30 年以上も前に出版されたものです．

- 加藤秀俊: “整理学”, 中公新書, 1963.

ですから，どんな技術を狭くかつ深く，どんな技術を広くかつ浅く，身につけるかが大事なことです．もちろん，技術者自分自身の状況 (能力) によって，どの範囲で深く掘り下げるのか広くカバーするのかに唯一の答えはありません．しかし，技術進歩に直面して，このように考えれば，もう迷う必要がないというのが我々の結論です．

- 戦略その 2 : ソフトウェア技術者の技術マップを作る

  先ほど述べたように，現在の技術発展は本当に速いのですが，技術者にとって厳しい世の中で生きるため，一番にやるべきことは何なのか，また，どのような知識を身につければよいのかに迷っている技術者はたくさんいると思います．

  しかし，我々の考えは，まず技術者として現在どの位置にいるのか自分で確認しなければなりません．そして，これから，どの位置を目指して努力していくのかが大事なことです．これらを分かるようになるため，技術マップのようなものが必要だと考えました．例えば，私たちの仕事の流れと，その中で技術者が果たす役割について整理し，見直した方がよいと主張します (図 2，3参照)．

  マップを作るときには，次のような情報処理技術者試験の種類が参考になります．

  - プロジェクトマネージャ
  - プロダクションエンジニア
  - マイコン応用システムエンジニア
  - システム運用管理エンジニア
  - データベーススペシャリスト
  - 第一種情報処理技術者
  - 第二種情報処理技術者
  - .....

  具体的なマップを示すのは難しいのですが，我々の結論は次のとおりです．

  - マップを自分で作る
  - 絶えず変更，改良，訂正していく
  - 唯一のマップは存在しない

デザイナ（GUIなど）	DB スペシャリスト	NWスペシャリスト
プロジェクトマネージャ	セールスエンジニア	OOスペシャリスト
プログラミング言語スペシャリスト	ハードウェアスペシャリスト	環境アドミニストレータ
.....	.....	.....

図 2: ソフトウェア技術者の技術 (役割)

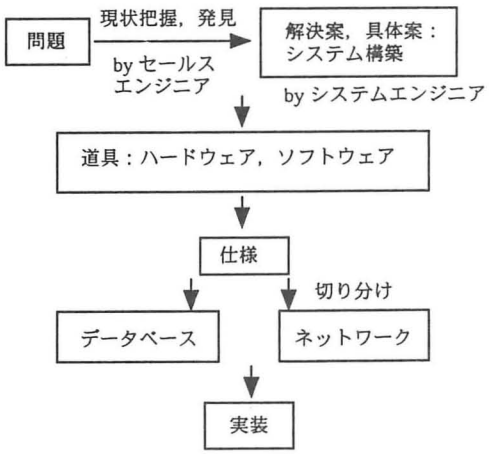


図 3: システムと役割

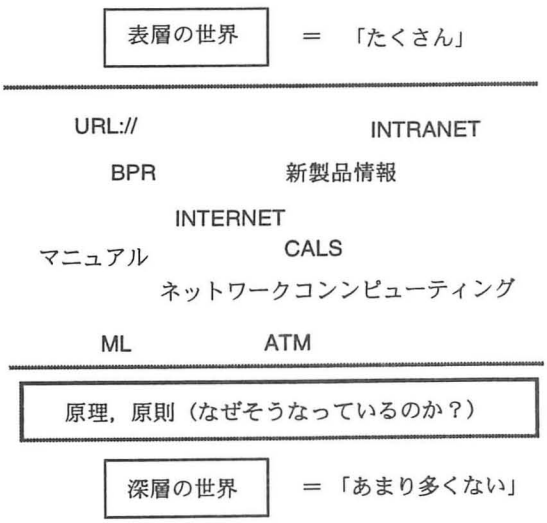


図 4: 自分の MAP と自己の位置づけ

- 戦略その 3 : ソフトウェア技術者の本棚を持つ

  たとえ，自分のマップをつくって自己の位置づけができても，速い技術進歩と多過ぎる情報という現実にどう対処すべきか，どう準備すべきかなどの問題に迷っているソフトウェア技術者がいると思われます (図 4).

  我々はそれらの問題を解決するために自分に合うデスクトップ (本やデータなど，本質の理解に役に立つものがすぐ取り出せるようなもののリスト) を作ることを提案します．本棚は技術マップ作成の情報源，それ自体，本質の理解などに役立つものだと考えています．もちろん，現在の情報収集に関してはいろいろな方法 (人に聞く，URL，本，...) がありますが，それぞれの利点と欠点もあると思われます．自己の位置を決めて，さらに自己が目指す位置に向かって，自分の本棚を持つのは大事ではないかというのが我々の結論です．

5.1.3 まとめ

今回の若手のワークショップでは，上述のような流れでいろいろな討論を通じ，そして，他の二つのグループからの発表を聞いた上で，最初に示された質問票にも一部，答えられるようになった気がしています．

大変勉強になりました．

A グループでは，今回のワークショップ「ソフトウェア技術者のサバイバル戦略」について以下のようにまとめました．

- 速い技術進歩に迷う
  - まずどんな知識 (深く，狭く，広く，浅く) が自分にとって一番必要なのかを決めること
  - 自分の現状 (能力) に準じて，うまく (深く，狭く，広く，浅く) 新しい知識を身につけること

  ====⇒ もう迷わない

- ソフトウェア技術者としてどうすればいいのかに迷う
  - まず技術マップを作り，自分の位置を正しく見つけること

  ====⇒ もう迷わない

  - これから，技術マップで目指していく目標を定めること

- 情報量が多過ぎることに迷う
  - まず，自分に合う本棚を作ること
  - 本棚を作りながら，自分の情報収集能力を上げること

  ====⇒ もう迷わない

最後に，上記のものはあくまでも一般的なものであり，ソフトウェア技術者は，自分の現状をベースにして，上記の方法を日常の仕事の中で実践できれば，と考えています．もう一つは上記の方法 (マップ，本棚など) は唯一決まるものではなく，これからどんどん変わっていくということをいつまでも忘れないようにしていきたいと思います．でも，第一歩はやはり実践でしょう．

謝辞

今回のワークショップに参加する機会をくださった SEA のスタッフ全員と関係者に深くお礼を申し上げたいと思います．

5.2 参加者の感想

5.2.1 有意義なワークショップ

石川 雅彦

有意義なワークショップでした．4 日間という期間は “ソフトウェアサバイバル戦略” という大きなテーマの結論を出すにはあまりに短いものですが，最終日には満足のいく形にまとまったと思います．また，場所をいろいろと変えて，朝から晩まで (晩はお酒が入りつつ) テーマに直接関係あること，ないこと，いろいろなことを話し合う過程も有意義で楽しいものでした．その後，メーリングリストができて，引き続き話し合いが続いていることも特筆すべきことです．今後も引き続き，みんなと交流していきたいと思います．

5.2.2 今回のセミナーは大変勉強

阪長 俊之

いつもネットワーク上で対話をしていても，なかなか同じ技術者として対話することが少なかっ

た私にとっては，今回のセミナーは大変勉強になりました．表層を常に追っかけていたところに基礎の大切さを思い知らされました．これからもここで結んだ仲間たちとのつながりを大切にしてすごしたいです．

5.2.3 共通の迷いを持っていることに驚き

山崎 望未

今回のワークショップでは，ソフト技術者の皆さんが，技術の進歩と情報の氾濫にどう対応していくべきかという共通の迷いを持っていることに驚きました．これまで会社以外の人と話をする機会がなかったので，非常に勉強になりました．なぜ迷うのか，どうすれば迷わずにすむのか，明確な回答はでませんでしたが，今回のワークショップでひとつの道標を得ることができました．これからは，その道標を手がかりにどう進んでいくかが大切だと思っています．今後は通信ネットワークを通じて，人と人とのネットワークをより深く，より広くしていけたらと思っています．今後ともよろしくお願いします．

5.2.4 同世代の技術者と考えや悩みを共有

山家 昇

SEA 若手の会ワークショップに参加して，同世代のソフトウェア技術者が考えていること (特に悩み) を共有し，理解することができました．会社内の同世代も同じことを考えていると想像できますが，こういったことは，今回のように，なかなかじっくり話し合う機会がありません．意外とみんな悩んでいることは似ていると感じ，安心感や連帯感が生まれたような気がします．また，山崎先生をはじめ，業界の先輩方々のご意見は，大変，刺激的かつ貴重で，今後，ソフトウェア技術者として「サバイバル」していくためにも大変参考になりました．今回知り合った仲間との絆を大切にして，引き続きコミュニケーションしていこうと思います．

5.2.5 今回のセミナーで考えるきっかけ

許 林興

今回のセミナーに参加する前は，ソフトウェア技術者のサバイバル戦略についてあまり考えていませんでしたが，今回のセミナーのお蔭で，いろいろなことを考えさせられました．また，お酒を飲みながら，同世代の日本の若者の本音 (望んでいること，悩んでいることなど) を聞くこと (普段仕事場ではなかなかできません) もできました．この点でいうと，これから，お酒にも強くならないと.....それに他の方々に比べまして，自分の不足とこれからやるべきことも分かるようになりました．いい経験ができました．これから，ここで知り合った友達と長くつき合っていきたいと思っています．

6 (ちょっとまった!) のBグループ成果報告

報告：永岡 栄吾

参加者：藤坂 政志，諸岡 隆司，園田 修司，神山純子，永岡 栄吾
プログラム委員：谷口 悟，保坂 利夫

6.1 討論の軌跡

6.1.1 討論のテーマ

1. 定義

   技術者＝設計者，プログラミング，マネージメント

2. 環境

   個人 ⇔ 組織

   やりたいこと，好きなこと (ギャップ) やらせること

3. 技術者への道

   技術修得＝時間，お金，興味 (やる気)，流行 (本質)

4. 何をするべきか

   所属組織＝上司へのアプローチ (継続必要)，異動

独立・転職 ⇒ 成功するか？

6.1.2 討論

始めの時点では，技術者の定義も話題となりましたが，あまり真剣には議論をしなかったと記憶しています．何をもって技術者とするか，ということより，自分のしたいことをどのように実施するかが重要であると考えました．

我々のアプローチは，まず,「やりたいことができない環境をどうするか」という観点での話しがベースとなりました．その中で，中間発表では，自分のスキルをあげるためには何をするべきかを，現在自分の所属している組織との関係から検討しました．現在の仕事を,「自分がやりたいこと」と「組織がやらせたいこと」の関係で見た場合，ギャップがなければいいのですが，ギャップがあった場合は，そこに「なやみ」が生じます．その関係を，

「個人」$vs.$「組織」

という普遍的な事項に定義することもできますが，反対に，如何に協調関係を築くかという関係からも見ることができます．

また，自分がやりたいこと (身につけたいスキル？) を習得するためには,「技術者への道」にあるように，物理的にも，精神的にもリソースが必要となります．そのリソースを獲得するためには，自分で行動を起こす必要があり，また，所属組織に対しても行動を起こす必要があります．

以上までが中間発表の内容ですが，私たちのグループは，この中間発表時点では，あまり話しが煮詰まっていなかった，というのが正直なところです．またこの時点では，したいこと，習得すべきスキル，スキルの内容などなどについては，まだ議論がされていなかったので，まだ曖昧なところが多くありました．

まず，自分が何を目指すのかが明確でないといけないという点がポイントになると考えました．そこで，習得したいことや必要と思うことなどを，山崎さんの「ソフトウェア工学が大切」というお話なども参考にしながら，図 5の OHP を作成しました．この図は,「変わることがない，または変化が少ない技術領域」として「基礎技術」という項目があったと認識しています．その上に，時流に乗って変化していく「利用技術」があるという前提で作成しました．

自分自身が現役の技術者として生きるためには,「基礎技術」も重要ですが，即戦力となる「利用技

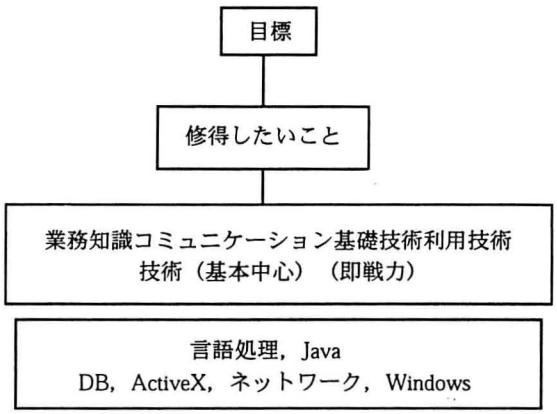


図 5: 最終発表 OHP(1)

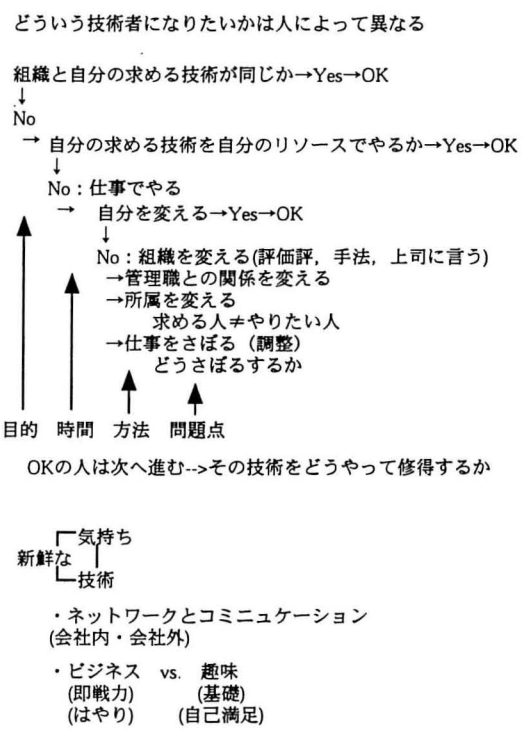


図 6: 最終発表 OHP(2)

術」も必要との認識が我々メンバー間では共有されたと思います．この時点で出たかははっきりしていませんが，まだ入社年数が少ない技術者には，「目標」や「やりたいこと」が見えていません．しかし，その場合は，所属組織が用意してくれた「仕事」をきちんとすることが重要であり，それが「目標」となるという議論もされました．つまり，その人は「NoProblem」ということです．

我々は，現在の自分と組織の関係をはっきりさせて，その中で，スキル習得やスキルアップの道を探るために，図6に示したYes/No図を作成しました．この中で，まず,「仕事をさぼる」ということについて，組織の中での「さぼりかた」については，かなり発表のときに意見交換がありました．その中には,「技術者はうまくさぼるもの」という話しには，良い意味ならば賛同するという意見もありました．つまり，現在の与えられた仕事を工夫し，効率化しつつ，自分の自由になる時間を作り，その時間で，技術習得やより高度な技術利用を考えるという意味では，うなずける部分が他の参加者にもあったのだろうと思います．

また，組織との対立関係で自分を中心に考えるのではなく，もう少し協調も考えては，との意見も出ました．例えば，現在の環境に不満があった場合,「根気強く上司にアピールをしていけば道が開ける場合もあるよ．いつかは」という意見も出ました．確かに,「この人ではだめだが，こっちの人は大丈夫」などということは，結構ありますから．

我々の発表が，あまりにも「自分が，自分が」と聞こえてしまい，自己中心的な意見と思わせる部分もあったので，酒匂さんからは「技術とは誰の，または，何のためのものなの？」という質問も出ました．ただし，図6のYes/No図は対立関係を見るためのものではなく，自分の現在の位置を客観的に見るために有効ではないかと思っています．

また，仕事や状況は常に動的なものなので，この図は，その一瞬を見ているに過ぎないという趣旨の話しも出ました．それはその通りで，自分の位置も常に見直す必要もあると感じています．

最後に，参加された皆さんが仕事に対しそれぞれの見地で，とても真剣に考えられていることを知り，世の中の全ての人がこんなふうに考えているのかな…?，と正直なところ自分の考えの甘さを自覚し，もっと考えて生きていかなければと反省した次第です．

6.1.3 まとめ

非常に進歩の早い世界にいると思われているのに，個人的な環境や状況によって若干異なるものの，その中にいる技術者の悩みは，長年，ほとんど変わっていないという状況を見せられて，やや気落ちしました．しかし，そんな状況にありながらも，問題を再認識して解決方法 (実現可能かどうかはやはり個人差が大きいと思います) を模索できたことは，大変意味のあることであったと感じています．

とはいえ，そう急には変わらないとは思いますが….

ワークショップの期間中，こんな私の意見にも真剣に耳を傾けていただいた皆さん，ありがとうございました．

6.2 参加者の感想

6.2.1 へ？感想ですか？こまった．

保坂 利夫

へ？感想ですか？こまった．実は，私自身は初日から風邪のため体調 (特に内臓) が最悪で，中来田さんのように「パワー全開 !! 」とはいかず，振り返るとかなり不本意な状態でした．しかし，準備や段取りが不慣れではありましたが，議論や出会いについては「ほぼ満足」です．各人のおかれている状況はいろいろで，それを共有するというのは無理でしたが，共通の「思い」は感じました．

たとえば，

- 移ろいやすい技術情報や製品に日々振り回されている状況．
- 何か「核」となる確固とした技術を模索する状況．
- そうはいっても目の前の仕事は「やらないと」という焦り．
- その中で「指針」もしくは「信頼できる先輩」を探す無意識の姿．

などなど．ちょっと，カッコ良すぎか．

私個人としては，体調が良ければ，もっとプログラム委員の方々や，山崎さんとお話をしたかったのですが，まあ，また機会はあるでしょうしね．

6.2.2 ソフトウェア技術者の悩みは変わっていませんね

谷口 悟

終了直後の所感：ソフトウェア技術者の悩みは変わっていませんね.

- 新技術に対する焦燥感,「乗り遅れ」への強迫観念
- 情報洪水，真実がわかる情報の不足，基礎知識の重要性
- 長い目で見たスキルアップの重要性
- 組織を動かすことの難しさ
- ビジネスとスキルアップの両立 (またはトレードオフ) →どちらが優先か？

個人的には，今はソフトウェア技術者ではなく，(極端に言えば) 手配師としての活動しかしていませんでしたので，今回のワークショップは刺激になりました．参加してよかったです.

6.2.3 入社 1 年目の方から 10 数年の方までが参画し，有意義

藤坂 政志

入社 1 年目の方から 10 数年の方までが参画し，有意義な時間をすごせたと思います．会社は違っても，共通の悩みや問題意識があるのだなと改めて感じました．私はおじさんの部類 (気持ちは若手のつもり) でしたが，昔話 (私がソフトウエア技術者として育ってきた環境等) が多くなってたような気がしてます．おじさんと話をする機会も会社内で結構あるかもしれませんが，聞き流すことも大事だと思います．たまに為になることを言うので，大事だなと思ったことは聞き逃さないで下さい (これは，一般論です).

1. 討論内容について

   ソフトウエア技術者の定義については，なかなか難しいと感じました．最終発表の OHP で記述してましたが，業務知識，コミュニケーション技術，基礎技術，利用技術の切り口があり，要は自分がどの分野でエキスパートになるか，目標 (長期，短期) を設定し (無理をせず) 地道に活動することに尽きるかなと思います.

2. 地道な活動についてやはりスキルを身につけるためには，勉強しないといけないですね.

   - 仕事をこなしながら習得する.
   - 社内教育，研修をきっかけに習得する (社外教育，研修含む).
   - 興味のあることについて，自己学習する.

3. その他

   市内観光は，とても楽しかった．駿府公園，登呂遺跡，芹沢美術館，いちご狩等．またの機会にお会いできることを楽しみにしています.

6.2.4 「参加したら？」の言葉に，気軽に

諸岡 隆司

私が今回の若手の会に参加したきっかけは，社内の方から,「参加したら？」の言葉に，気軽に「はい」と答えてからでした．SEA のワークショップに参加するのは初めてで,「まわりの人の話についていけるのだろうか？」という不安と，－技術者はいま何をなすべきか？－という題材に「本当の答えがあるのだろうか？」「道徳的，精神的な話で終わってしまうのか？」という思いで参加することになりました.

ワークショップ参加当日は大雪になり外に出るなり転んでしまったのが不幸の始まりで，恐る恐る歩いて駅にたどり着くと，電車のダイヤは大幅に乱れて全然来ないし，新幹線のホームも大混乱でした．結局，セミナールームにたどり着いたときにはすっかり始まっていました.

私たちのグループは PC の方を含めて 7 人で，年齢的に 23～37 歳と幅広かったため，新入社員の方から管理職の方までいろいろな立場の意見や，環境の違う職場でのいろいろな話を聞くことができました．グループ討論の題材は最終的に，－技術者はいま何をなすべきか？－を話し合うのでは無く－私たちは，いまやるべきこと (やりたいこと) をどう実現するか？－というテーマについて話し合い，過激な意見もでましたが，全員で真剣 (？) に話し合いました．野外活動では，比較的天候に恵まれ，－技術者は健康のためにビタミンの補給を－ということで (私が勝手に言っているのです

が)，場所が静岡で時期的にもシーズンのため，苺刈りに出かけました．これがとてもおいしく，皆さんにお土産として持ち帰るのを忘れるほどでした (すいません)．これで今年は風邪をひくことなく過ごせると思います．他にも登呂遺跡を見学したりして，今，仕事に追われる毎日を考えれば夢のような日々でした．夜の部 (毎日，当然のように飲みにいってましたが....) でも，仕事の話や趣味の話などいろいろなお話ができ，楽しく過ごせました．(ここで再確認できたのですが，SEA の皆さんは大酒のみが多いです)

今回のワークショップに参加して，このようなスタイルのセミナーを体験でき大変おもしろかったです．実質 3 日間だったのですが，あっと言う間に過ぎてしまいました．残念だったことは，前回のセミナーでは “わんこそば大会” なるものが催されたみたいですが，今回は無かったということです．(あれば私の活躍の場であったのかも....) 是非，次回のセミナーでは○○大会を催して頂きたいです．

－技術者はいま何をなすべきか－という題材の発表内容で「目先の最新技術に追われることなく基本技術を習得する」という話があり，影響されやすい私は「う〜ん，やっぱり基本は大事だ」とワークショップ後に参考文献として紹介された本を読んでみたのですが，これがとても難しくてすぐ読むのを辞めてしまいました．(これだからダメなのですが....) 確かに基礎技術の習得が将来的に技術者として生き残る道なのかもしれませんが，なかなかできないのが現状です．(最新技術に触れているうちに自然と基礎技術がつくようであればうれしいのですが....)

今回参加された皆さん，いろいろとお世話になりありがとうございました．皆さんと話をして，とてもソフトウェアが好きなんだなぁと感じ，刺激を受けました．私も皆さんに負けないぐらいソフトウェアに興味をもち，ソフトウェアに接していきたいと思います．また，このようなワークショップに参加させてもらったことに大変感謝しています．ありがとうございました．

6.2.5 「若手の会は楽しめるところ」との前評判通り

園田 修司

「若手の会は楽しめるところ」との前評判通り，とても楽しいワークショップでした．初日関西は大雪で交通機関が麻痺していて，静岡まで行くのはとても大変だったのですが，がんばって参加した甲斐がありました．(静岡は晴天でなんじゃこりゃと思いました)

SEAMAIL で会社に詳細が伝わることを考えると，もっと消極的に (!?) いかないといけなかったのかもしれませんが:-p，今回のテーマ設定は極めて私向き:-) だったと思えるので，いろいろと意見交換させていただいて大変よかったと思います．

発表内容は 3 グループとも切口が違い，いまから考えるとそれぞれが補完しあってるようにも思えますので，SEAMAIL で並べてみるのが楽しみです．ところで，私は最後のプレゼンテーション担当だったのですが，プレゼンテーション (＋質疑応答) のあとにすぐ解散だったのがちょっと残念でした．ネタ的には皆さんに楽しんでもらえたようなので，個人的な初期の目的は達成されて:-) よかったのですが，プレゼンテーションの後，質疑応答とは違った形の意見交換をする時間があれば，もっとよかった気がします．

最後に，参加者の皆さん，PC の皆さんどうもお世話になりました．また違う形で会うこともあると思いますが，そのときにはよろしくお願いします．

6.2.6 これからどう生きていくべきかを考える機会

神山純子

これから，私がこれからの技術者としてどう生きていくべきか (ちょっと大げさかも !?) を考える良い機会だった．今まで (といっても 10ヶ月間ですが) は与えられたことをこなしていくということに精いっぱいだったが，これからは，自分の将来を念頭において何をした方が良いかを考えて行動するようにしたいと思う．

技術者は，

1. 自分がどういう人生を歩いていくのか目標を決める
2. 目標を達成するために努力する

をまず行わなければいけない．

情報は，世の中に氾濫している．正しい物も間違っているものもある．取ってきた情報が良い物か

悪い物かを判断する，また，今後その技術が発展するのか没落するのかを判断するのは難しい．まず，その見極めに役立つと考えられるのは，先人が歩いてきた基礎的な技術である．基礎は普遍である．その上にある応用は基礎の上に成り立っている物なので，応用技術を本当に学ぶためには，基盤となる物がわかっていないと表面的な理解になるのではないかと考えられる．流行に流されない本当のソフトウエア技術者になるためには，ソフトウェア工学を学ぶ必要があるのではないか．でも，この方法を身につけるにはかなり長く険しい道が待っている．もっと，要領よく判断できる知恵がないのかとも思う．

日本全国からソフトウエア技術者が集まっていた．大企業から中小企業，新人 (私) から管理職一歩手前という人まで多種多様であった．どんな所でも，ソフトウエア業界は同じような悩みを持っているのだなと思った．いろいろな話しを聞けて刺激になった．みんな実際の業務をしながらこれから自分はどうすればいいのか悩んでいるんだなと思った．

～静岡の感想～

静岡は，思ったよりもおいしい物が多かった．まず，石垣いちご．静岡市の海岸沿いにいちご農家があって，いちご摘みをしてきた．大きくて甘くてスーパーなどに並んでいるものの 7 倍くらいおいしかった．食べ放題だったのでついつい食べ過ぎてしまった．次はうなぎ．浜名湖産 (静岡市ではないが) のものだった．身が柔らかくて，たれもジューシーだった．そして，さくらえび．えびはゆでたら赤くなって，生きている時は黒い物だと思っていた．でも，桜えびは違った．お刺し身や，から揚げなどを食べたが身がぷりぷりしていて日本酒とよく合った．お好み焼きに入っている桜えびは，実はプランクトンが多いということも初めて知った．今まで食べていた桜えびって一体…と思った．

またソフトウエア業界は狭く，技術者は酒飲みが多いということもわかった．思わぬところで人がつながっていた．楽しい充実した 4 日間だった．また，SEA の会があったらぜひ行きたいと思った．

6.2.7 たまには外の空気を吸って

永岡 栄吾

SEA がどのような団体なのかという予備知識も与えられず，「たまには外の空気を吸ってくれば」という一言で，若手の会に参加しました．行くまでは，どうのような話をするかが不明だったので，正直気持ちが重かったです．しかし，実際に若手の会で話し合いをしてみると，普段自分が考えていた会社と自分の関係や一人前の技術者とは？などの問題に対して色々と意見交換ができたので，(こう言っては失礼ですが) 思った以上に収穫があったというのが，素直な感想です．

細かい話し合いの内容は，自分が参加したグループしか分かりませんでしたが，発表内容を聞く限りでは，内容が自分のあまり想像してない方面に広がっており，その意味でも勉強にはなりました．しかし，ソフトウェア技術者の抱えている悩みや問題はあまり変わらないものなのですね．

今回のワークショップで大きな不満というのは無いのですが，同じグループの人とはかなり話し合えたと思うのですが，他のグループの人とはメインとなる話題でほとんど話すことができなかったというのが，心残りといえば言えなくもないです．まあ，時間的な制約もありますし，あまり大人数で話してもまとまりがなくなりそうなので，結果的にこのような方法で良かったと思います．

しかし，最大の収穫というのは技術者として問題点や悩みを再認識できたことと，仕事と直接関係ない人との繋がりができたことだと思います．今後も継続して話し合いなどが持てれば最高だと思っています．もちろん，今回のような内容以外の話しでも得られることが多かったというのは，当然のこととして，私の感想を締めくくりたいと思います．

7 Cグループ成果報告

報告：鈴木 勉

参加者：吉澤 賢一，上田 秀樹，阪井 誠，鈴木 勉，堀口 利子

プログラム委員：中谷 多哉子

7.1 討論の軌跡

グループ発表の電子化したレポートということで，OHP 中心にしてみたのですが，討論の流れといっしょになって見にくくなってしまったかもしれません．

7.1.1 テーマ

1. ポジション・ステートメントについて

   Q1 あなたが今持っている「ソフトウェア技術者」のイメージは?

   Q2 急激な技術進歩の渦中に置かれている「ソフトウェア技術者」として，日常どのようなことを準備・実践していますか?または準備・実践したいと思っていますか?

   Q1 では「ソフトウェア技術者の定義」を，Q2 では「情報収集」について意見を交換しました.

2. ソフトウェア技術者の定義

   広義：ソフトウェアに携わる人 (SE，プログラマ…)

   狭義：

   - 何か動くものを作る時，全体を設計できる人
   - 作る時に最適策を取れる人 ⇒ ソフトウェア技術を使いこなせる人
     他!は作業者？
   - 会社組織という枠の中で，新たな枠を作れる人.
   - 先見の明がある人.

3. 情報収集

   旬のものを見続けるために情報収集をしているが.

   - 情報を集めることはできる (パンフレット etc) が，それを活かせない.
   - 結果，広く浅く情報を集めるだけになってしまう.
   - 集めた情報が必要なときに引き出せるインデックスが欲しい.
   - 自分のしたいことと会社の要求にギャップがある.
     - 広く浅く学びたい→ギャップあり
       あまり仕事の役には立たない傾向がある.
     - 狭く深く学びたい→ギャップなし
       自分のやりたいことであれば探求できるが，会社の要求とのギャップでなかなか十分に行えない.
     - 広く浅く/狭く深くのバランス
       現実は狭く浅くなってしまう.
   - 自分の将来像がわかれば自分に必要な情報がわかるのではないか.
   - メンバそれぞれの将来像には違いがある.
     これは年齢の差か ??

   情報収集について討議しているうちに，経験年数によって仕事に対する認識が違うのではないかということになり，経験年数による仕事に対する認識の遷移図を作成することになった.

表 1: 経験年数と仕事に対する認識

経験年数	認識	該当者
入社		
–3 年目	言われたことを完成させることに夢中	上田 秀樹
–5 年目	今までの仕事はほとんどわかる	
–8 年目	自分にとっての仕事のやり方がわかってくる	吉澤 賢一
–10 年目	....	堀口 利子
:	?	阪井 誠
:	?	鈴木 勉

7.1.2 討議

ソフトウェア技術者として生き残るためには？

- 個々のモノ作りの技術は当然必要である.
- より抽象的な物が重要である.
- 抽象化したものと，抽象化していくプロセスが重要である.
  個別の言語等ではない.
- もちろん基本理論は大切である.
- 実践も伴う必要がある.
- 理論を知らないと誤った理解をしてしまうことがある
- 本質を知ることが重要である.

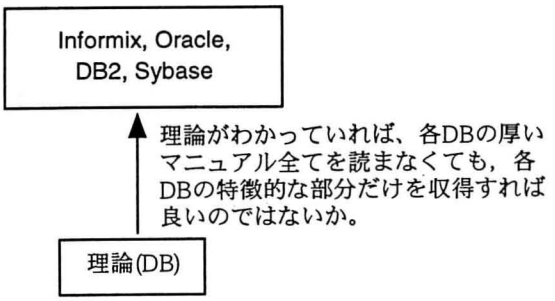


図 7: データベースの例

- モデルができても実際にモノ作りをするには，実装する具体的な技術が必要では？

  本質を知っていれば，実現する手段はその時々の技術を組み合わせて可能なはずである．

図7に示したように，データベースや ATM システム開発の討論を経て，**抽象化したもの，抽象化することが重要である**．という結論に至り，グループ発表ではボトムアップに議論した内容をトップダウンで示すことになった．

抽象化 (したもの/すること) が重要

↓

＜モノの作り方の作り方の作り方＞

↓

＜モノの作り方の作り方＞

↓

＜モノの作り方＞…言語，アルゴリズム

↓時代により変わる

モノ　　…プログラム

グループでまとめたソフトウエア技術者の定義と，ソフトウェア開発と情報収集のキーワードを掲げる．

- 技術者とは

  物事をモデル化して「○○とは何か」を考えられる人

  → 上田秀樹さんのポジションペーパ…
  (現場の経験は役に立たない?!)

- 技術とは
  - ソフトウェア開発 (理論を含む) とは何かを知っている
  - システムにはどのようなものがあるかを知っている
    * ATM システムとは何かを知っている
    * 会計システムとは何かを知っている
    * 在庫管理システムとは何かを知っている
    * DTP システムとは何かを知っている
    * 電話設備管理システムとは何かを知っている
    * ..... とは何かを知っている

ATM システムについて考えてみる．A 銀行の ATM があり B 銀行の ATM がある．A 銀行の ATM システムにはいくつかのバージョンがあるかもしれない．ATM システムを構築するには，ATM の業務知識が必要であるし，A 銀行のシステムを構築するには，A 銀行の業務知識が必要であるに違いない．

ATM(ATM の業務知識) とは，A 銀行，B 銀行，標準的な銀行の ATM システムに使える知識である．現在の ATM がなんらかの新技術によって新しい ATM に変化したときに，この新技術を習得するための一つの手立てとして我々は情報収集をするのではないか？

- 情報収集とは

  改めて今回のワークショップのテーマを確認．

  > 『ソフトウェア技術者のサバイバル戦略』
  > ソフトウェア技術者のキャリアパスが設定されていましたがもはや存在していません．その上，技術はまさに日進月歩で，しかしこのような状況で今すぐやっておくべきことは何か．

キャリアパスが存在しないのなら，自分でキャリアパス (ライフスタイル) を設定していけばよいのではないか．どうして技術は日進月歩なのだろうか．それは抽象化されていないからではないか．

7.1.3 まとめ

日進月歩の技術とは抽象化されていない技術である．今すぐやっておくべきことは，重要な技術を抽象化して，身に付け，「自分で自分のキャリアパス」を持ち，モデル化ができる技術者になること．

7.2 参加者の感想

7.2.1 他の参加者の思いのたけを聞けた

吉澤 賢一

先日は皆さん御疲れ様でした．私の参加理由ですが，上司からの命令 (に限りなく近いもの) だったので，当初はこのワークショップに期待する気なぞ，正直言ってまったくなかったわけです．しかし，実際に参加してみて議論の結論そのものもさることながら，他の参加者 (社) の方の思いのたけ (笑) を聞けたことは大変有意義でありました．

今後ともよろしくお願いいたします．

7.2.2 自分と同じように悩みや不安に驚きと安心

堀口 利子

上司から薦められてこのワークショップに参加したのですが，まず始めに，ワークショップに参加されたほぼ同年代の方々が，

「氾濫する情報の中，自分に必要な物が何なのかわからない.」

「自分の専門とする技術を身に付けたいが，自分が興味を持っていることと仕事で必要な技術が必ずしも一致しないため，自分の進む方向がはっきりしない.」

「ソフトウェア技術者として，今のままの自分で生き残っていかれるのか？」

等，レベルの違いは在るにせよ，自分と同じように悩みや不安を抱えていると知り，驚きと，またある種の安心を感じました．

約3日間のワークショップを終えて，学んだことが十分に自分の中に活かせられるかはわかりませんが,『こういうふうに考えていけば良いのかな…』というものがぼんやりと見えてきたような気がします．

・業務内容，職場，立場等は違ったとしても，自分自身の捉え方でそれぞれの状況の中でも学ぶべきことはある．

・ソフトウェア技術者としてだけでなく，社会人として生き残っていくにも自分のキャリアパスを自分で作っていかなければならない．

7.2.3 何をするべきかを考える機会

上田秀樹

若手の会に初めて参加させて頂きました．入社して2年が過ぎて，思うことはいろいろありますが，ソフトウェア技術者として，何をするべきかを考える機会になって，よかったと思います．今回の我々のグループの結論である，物事を抽象化して考えるということは，ソフトウェア技術者に限らず，どんな仕事をする上でも適用できることだと思います．当たり前の結論といえばそれまでですが，実際に仕事をしていると，目先のことにとらわれて，考えることがおろそかになってしまいます．立場的にいろいろ教えてもらうばかりになって，私のほうから何もできませんでしたが，自分で何ができるかを考えることができるようにはなりたいと思います．また，機会がありましたらよろしくお願いいたします．

7.2.4 2つのことをあらためて確認

阪井 誠

今回のワークショップに参加して，私は2つのことをあらためて確認することができた．

1. 体験した業務に関係なく，入社後の年数により同じような経験をしていること
2. 大学で教えられている「論理的な思考」がサバイバルに役立ちそうなこと

まず，1番は若手 (といっても少し老けているが) の会ならではの経験である．色々な会社から同世代の人間が集まったからこそ，確かに「社会経験はどこでどのような仕事をしていようと同じように蓄積される」という実感を持てたと思う．(最近はSEA関西でも多忙なためか同世代の参加が少ないのは残念である...)

次に2番は私の大学での経験から，Cグループの結論である「モデル化こそサバイバル技術である」という結論を見直せたものである．大学は兎角現場と離れがちと批判されているが，優秀な研

究者でもある私の恩師が「(研究者でなく，論理的な思考のできる) 高度な技術者を育てている」といわれていたのを思い出した．大学をもう一度教育機関として (直接参加するのか，そのエッセンスを吸収するかは別として) 見直すべきだと思った．昔の基準では若手にならない私は，社会人を長く経験し，何事も力仕事で片づけようとしがちであるが，原点に立ち返って論理的な思考や，山崎さんが述べられていた基礎的な知識を得る必要を感じた．

最後に，スケジュールに追われていたため参加を諦めていたが，無理矢理参加させていただけてよかったと思う．

7.2.5 自分は疲れてしまっているようです

鈴木 勉

ワークショップに参加して，自分を見つめるいい機会になりました．自分は疲れてしまっているようです．皆さんと比べてパワーがないんですね．ワークショップから自分の仕事に戻ってみると，時間の制限というか，仕事に追われたりしてほかにできることが限られているなとも感じています．時間は作るものだという考えもありますが... まずは，リフレッシュして，ゆっくりと地道に歩いていこうと思います．

8 プログラム委員から

8.1 時代に取り残されるという「強迫観念」を解消しよう

谷口 悟

8.1.1 感想

1. 技術者の悩みは共通

   今回異なる業種・担当分野の技術者にて意見交換を行ったが，これほど悩みが共通しているとは思わなかった．

   現在は最新技術?が次々に登場しているが，追いつけている実感のある人は皆無で，時代に取り残されるという「強迫観念」に悩まされている人が多いのであろう．これは今回のワークショップ参加者に限らず，世間一般のソフトウェア技術者にいえることかも知れない．

2. 「流行」より「基礎」

   上記の「強迫観念」への対抗策として，今回のワークショップで強調されていたのが「基礎知識」の重要性だったと思う．この基礎知識，習得するには時間がかかる上に，実務上「即戦力」を期待されがちな現状では，最新技術に強い人間が重宝されるため，どうしても勉強を適当に済ませてしまいがち．

   でも，私の周囲でも，最新技術の情報には強いが，何か技術資料をまとめるとなると，やたら時間がかかっている人というのが多い．周囲の流れに目を奪われず，時々は基礎知識の習得に時間を取りたい．ただし，最近は「パラダイムシフト」を要求される時代でもあり，必要な「基礎知識」も変化しているように思う．

3. 個人的に

   現在は，社内の販促活動がメインで，ソフトウェア技術からは遠ざかっている．この状況を逆利用して，基礎をじっくり習得する機会を多く取っていきたい．

8.1.2 今後の展望

いくら「基礎」が大事といっても，即戦力を期待される身では,「最新技術」に通じている必要もあろう．特に最近はデータウェアハウスのような凡人には構築が困難なシステムの構築を強要されて難儀している人も多いはず．

時々は,「時代に追いつく」機会がこの「若手の会」で設定できると良いのでは？

8.1.3 最後に

プログラム委員にも関わらず，運営面では何一つお手伝いできず，失礼しました．連日の「飲み会」はあまり酒になじみのない私にはきつかった面もありましたが，その分貴重な情報交換の機会となり，参加できてよかったと思っています．

8.2 できるところからやらなきゃ！

保坂 利夫

取り急ぎ，感想を送ります．花粉症＋風邪＋日曜日午前1時までの勤務のため，ちょっと支離滅裂かも知れませんが，ご容赦を．

8.2.1 感想

メーリングリストに誘いがあり，なんとなく参加したのですが，まさかプログラム委員として参加することになろうとは，夢にも思っていなかったというのが，正直な気持ちです．現在の社内の立場が管理職であり，個人的には現場を離れてしまった悲しさ (?) が強かった時期なので，ついついメーリングリストのわなにはまってしまったというのが，本当のところでしょうか．

しかし，参加してみて非常に刺激になったのも事実でした．とにかく，できるところからやらなきゃ!というのが，私として感じた最大のことだったように思います．

8.2.2 今後の展望

私には (かろうじて 30 代ですが..) よかったのですが，本当の「若手」の技術者にとってどうだったのでしょうか?本当に「若手」の人たちが企画して，手配して，参加して，というのは現実にはかなり所属組織のバックアップがないと難しいと感じました．私の場合も，社内では結構「勝手に」動き回れる立場ではありましたが，参加するためには，かなり神経を使いました．(気が弱いだけだったりして)

そんなことから考えると，30 歳未満の技術者が参加するのは，結構いろいろな壁があり，大変だろうと感じます．今回は SEA 幹事の方々の尽力で若い方々も参加しましたが,「出ろといわれて..」状態の人がいたのも確かです．(参加した後は，かなり感想は違って，よかったと感じてくれたと信じたいですが....)

真の「若手の会」を考えた場合，今後はもっと工夫が必要なんだろうなと感じました．いっそのこと，誰かがメーリングリストで言っていたように「中年の会」として開催し,「若手」を「ご招待する」なんていうのもいいかも．

8.2.3 最後に

取り留めなく書きましたが,「プログラム大量生産時代」の発想からなかなか脱却できていない会社がまだまだ多いこの業界のなかで，技術をまじめに議論している数少ない団体としての SEA の意義は大きいと思っているので，その予備軍を育成する上からも，今後も「若手の会」は続けてほしいと思っています．今回の参加者を核として，次回開催までもっていければ何とかなるでしょうかね?

8.3 1 年半遅れの若手の会

中来田 秀樹

1 年半遅れで若手の会を開催することができました．周りでは若手のパワーが無くなり，意見が聞こえなくなって来ていると言う話しが出てきており，実際に開催するまでもなかなか積極的に手伝って頂けそうな方を探すまでに時間がかかってしまいました．色々と始める前までは考えていたことも，実際に開催してみるとあっさりと大きな問題もなく実行でき，開催中はパワー溢れる (特に夕方より未明までは) 若手の方々と論議でき，色々な意味で有効だったのではないかと信じています．内容も大切なのですが，各社の若手の方々が一つに集まり，各自の問題を話し合う．各自の考え方を話し合う場を設けることは自己啓発と言う意味でもとっても大切なことだと考えています．

できれば，跡絶えること無く続けて行けたらと考え，若手の方々にも積極的に手伝って頂ければと希望いたします．

8.4 今回は 40 歳以下と大幅に規制緩和

酒匂 寛

8.4.1 感想

若手の会に前回参加したのは，もうずいぶん昔のことである．そのころは参加資格が満 30 歳以下であることが条件であったが，日本の高齢化社会を反映して，今回は目安として 40 歳以下と大幅に規制緩和が行われた．

今回のテーマは,「ソフトウェア技術者のサバイバル戦略」ということで，ソフトウェア技術者のキャリアパスをこれからどう考えて，将来に備えていけばいいかを議論できればいいとと考えていた．実際に，活発な議論が行われた．内容に関する報告は他の人に譲るとして，以下，ワークショップを通して，心をよぎった考えを述べたい．

8.4.2 溢れる技術情報

現在世間には膨大な技術情報が溢れている．とはいえその技術情報は多くの場合，特定のサービ

ス (API) の利用法であり，ちょっとしたバージョンの変化によって利用法が変化するような不安定な技術である．それでもこうした必要な情報と技術を一つ一つじっくりと勉強して，実際の開発に応用するという道が確保されていればまだ良いが，実際には開発に直面した段階で，必要な技術が判明し，それを勉強する時間はとれないというのが実際だと思う．

理想的な進め方としては，要求定義を終わらせた段階で，システム仕様化の段階に移行し，その上でアーキテクチャや利用すべき技術，ハードウェアなどが順次決定されて行く筈であるが，特に日本ではアーキテクチャや利用ハードウェア等が要求と仕様が明らかにされる前に決まってしまっていることも多い．

ともあれ細かい技術情報が溢れかえっているわりには，肝心な基礎体力を与えてくれる情報が少ないのではないか，そのようなことに注意を払う機会が少ないのではないか，というのが今回のワークショップの基本的なトーンであったように思う．

山崎代表幹事のお話も，移ろいやすい技術に手を出す前に，非常に基本的で厳密な技術・理論を身につけておけば，一見複雑に見える技術も，単純な応用に過ぎないことに気が付くはずある，だから基礎を大切にせよ．という内容であった．

8.4.3 役割と技術者

ソフトウェア開発は複雑なプロセスであり，それが大規模になればなるほど参加する人間の役割も細分化されていく．人間は役割を与えられると，意識的・無意識的にその殻を守ろうとする傾向があるようである．これが意味することは，いかに良くできたプロセスと，その中のアクティビティを定義したとしても，それを役割として人間に与えると，本来アクティビティの意味を考えながら間をつないでいくという意識が欠落し勝ちになるということである．

昔アーサーケストラーの「ホロン革命」という本が話題になったことがあった．乱暴にまとめれば，全体は部分の総和ではないというのが，著者の主張である．分割された役割に徹する技術者は，安定状態では非常に楽であるが，一度変化が始まると，全体を統合するための，基本的なエンジニアリングセンスが身に付いていないため，その変化に対応することが難しい．

実際にはこうした役割の再定義を，役割実行者としての開発者に期待することが，そもそもの間違いなのであろう．ここで悩ましい点は，では翻ってたんにアカデミックな議論に与すれば良いのかといえば，実践の視点を欠いた役割とプロセスの議論では，現場で受け入れられないような代物にしかならないことは周知のとおりである．

なおここでもより上のレベルの「役割」が出てきたことにお気づきであろうか，ここでは「役割実行者」と「役割を定義するもの」という二つの”役割”を述べている．先に述べたように，こうして役割を割り振ってしまうと，人間はとかくそれだけに専念しようとする傾向にある．

あなたはあなた，わたしはわたしというわけで，それが全体としてうまく働いているのかどうかという議論がなおざりにされがちである．(全体の結果に対する善し悪しは評価されるにせよ，犯人探しはそれぞれの「役割」の人々が，きちんとその「役割」を果たしているか，といったところから始まりがちなので，全体的な本質論になりにくい)．

8.4.4 今後の展望

ソフトウェア技術者が，何かを作り出すために知らなければならない情報は，今や非常に膨大である．一人で学ぶことは難しいし，自分が扱いきれなくなったときに，どこに手がかりをもとめて検討を行えばいいかも非常に不明解である．

こうした際に常に力を持つのはいつでも「抽象化」である．最終的には開発に携わる人々一人一人の頭の中に，それぞれ関心がある視点から，全体像が与えられるような抽象化の手法が必要であろう．

そうした抽象化の世界では，可能な限りの「形式化」を行うべきである．構造，論理，集合といった基本的な道具をうまく使いこなせることによって，この抽象化の作業は非常に円滑にすすめることができるようになるはずである．

こうした抽象化の世界で必要なツール自身も，年々数が増えてきてはいるが，街にあふれる「How to ...」の類に比べれば，数もずっと少ないし，数年も勉強すれば一通りのことは行えるようになるだろう．残念ながらこのレベルに使いやすい支援ツールがないことが残念である．

最後に．最近特に思うことは，技術者は自分自身の情報リポジトリを持つべきであるということ

である．自分自身の技術情報，作り出した文書，書き下ろしたソースコード，理論などだけでなく，自分自身の開発プロセスで利用すべき様々なフォームの雛形までも含んだようなものを，個々人が自分の使いやすいように考えながら構築するべきであろう．何も本格的なDBMSを利用する必要はない(なんらかの強力なリポジトリが利用できることに越したことはないが)．でき得れば，そうした情報を技術者同士が交換できるようになればいいと思う．既に私たちに与えられ始めている強力なネットワークパブリッシングの機能を使えばこれも決して夢ではないであろう．

8.5 結論は「やらなきゃ」「かわらなきゃ」

中谷 多哉子

8.5.1 感想

「若手の会」のプログラム委員は今回で2度目である．前回は金沢で開催された会に協力した．実は，そのときも最終日に台風が近づくという天候のために，1日早く切り上げなければならなかった．だから，初日の大雪情報に少し不安があったが，事故がなくてなによりだ.「朝，家の前で転んだ」という参加者もいたようだが.

とかく「サバイバル戦略」などと題名をつけてしまうと，大愚痴大会になって収集がつかなくなるのではないか，という危惧もあったのだが，参加者から寄せられた上々の感想に，ほっとしている．各グループとも，結論は「やらなきゃ」「かわらなきゃ」ということだったように思う．静岡で若手の会が開催されてから随分たってしまったが，参加者の人生改造計画は順調に進んでいるのだろうか．がんばってもらいたい．

話は変わるが，今回の参加者から「若手の会は参加するまでは飲み会だと思っていた」という意見を聞いた．これから参加を考えていらっしゃる方や部下の参加を検討されている管理者の誤解を解くために，会の方針をお知らせする必要があると思う．従来，若手の会はグループ討論を中心にスケジュールが組まれてきた．今回も，2日目の午後と最終日の午前中の発表時以外は，すべてグループ単位の自由行動になっていた．自由行動中は，どこで，何を話し，何を飲み食いし，どのような観光をしようと主催者側は一切関知しない．自社以外の同世代の技術者と交流をすることと，ひとつのテーマについてとことん議論し，発表時に成果を出すことが「若手の会」の目的である．だから，発表の準備は(飲んでばかりいられるほど)楽ではなく，私が担当したグループでも，討論には，延べにして24時間以上の時間が費やされていた．その討論の成果が，参加者の新たな認識の獲得や従来の認識の再確認につながっているのではないだろうか．社内で仕事をしていると，業界全体における自己の位置付けを見失いがちであるが，この会に参加することによって，参加者が何かに開眼できればとプログラム委員一同は願っている．

8.5.2 今後の展望

さて，来年はどのような会になるのだろうか．是非，今回の参加者が中心になって，来年の企画をたててもらいたい．

8.5.3 最後に

今回はこの特集の編集も担当した．ただの報告書にしてもつまらないかなと思い，多少脚色させていただいた．参加者の感想の項につけたタイトルは中谷の責任である．あまり「よかった，ためになった」という意見だけでは，怪しげな宗教宣伝のようになってしまうとので，少し工夫をしたつもりだが，効果のほどはいかがだっただろうか．もし，参加者ご本人の意図に反したタイトルがついていたとしたら，この場を借りてお詫びする．

最後にお願いがある．今後の「若手の会」も，21世紀の若手を育てるためには時流に沿って進化しなければならないはずである．そこで，この記事をご覧になった会員の皆様には，ご意見をSEAまでお知らせいただきたい．そこから，次回の有意義な会を企画できればと思う．

最後になってしまったが，多忙のなか，参加者に4日間の出張を許可された関係者に感謝する．なお，参加者の所属部署は参加当時のものである．

Does Better Process Lead to Better Product?

— Software Symposium 97 基調講演 —

および

Software Architecture

— SEA Forum June 1997 —

Alexander L. Wolf
(University of Colorado, Boulder)

6月のソフトウェア·シンポジウム in 福岡では，コロラド大学 (ボールダー校) の Alexander Wolf 教授を基調講演者にお迎えした．

Wolf 先生は，マサチュセッツ州立大学で博士号を取得された後，AT&T ベル研究所で，ソフトウェア工学 (特に大規模ソフトウェア·システムの開発·管理技術) の実践的研究に従事され，1992 年からコロラド大学に移り，コンピュータ·サイエンス学部で学生の教育にあたれるかたわら，同大学のソフトウェア工学研究実験室 (Software Engineering Research Lab) で，企業とのさまざまな共同研究に従事されている．

1. SS-97 での基調講演

ソフトウェア·シンポジウム 97 での基調講演は，「よりよいプロセスはよりよいプロダクトをもたらすか？」というタイトルであった．その内容は，コロラド大学ソフトウェア工学研究所が AT&T ベル研究所と共同で行なったソフトウェア·プロセスのモデル化に関する実践的研究の成果にもとづくお話であった．

ソフトウェア·プロセス研究の目標は，現実のプロジェクトにおけるプロセス改善を行なうための技法やツールを開発することである．そうした改善の効果は，対象とするプロセスのモデルおよびそのモデルが実際の開発プロセスをどの程度まで反映しているかについてのわれわれの知識に依存する．

Wolf 先生のグループは，AT&T との共同研究によって，現実のプロジェクトで収集された管理データを分析して，あらかじめ与えられたプロセス·モデルと現実のプロジェクトの挙動との違いを明らかにし，より実際に近いモデルを発見するための Balboa と呼ばれるツール·セットを開発した．

以下 29 ページから 47 ページまでに掲載したのは，福岡での基調講演で使われた OHP のコピーである．なお，Balboa ツール·セットに関心を持たれた方は，Web Page:

http://www.cs.colorado.edu/users/serl

を見られるとよい．そこには Balboa プロジェクトについての詳細な説明 (テクニカル·レポートや学会発表論文) がある．また，ツール自体も，フリー·ソフトウェアとして公開されている．

2. SEA Forum June

6月の SEA Forum は，シンポジウムの後，東京に立ち寄られた Wolf 先生をゲスト·スピーカにお迎えして，ここ数年アメリカのソフトウェア·エンジニアリング·コミュニティでホットな話題となっている「ソフトウェア·アーキテクチャ」をテーマに，講演と討論を行なった．

講演要旨は次の通り:

Software architecture is the principled study of large-grained software components, including their properties, relationships, and patterns of combination. This is an emerging area of study in software engineering that brings together several threads of research conducted over the past 25 years, including design, analysis, reuse, and process. We survey the current state of software architecture practice and research. We describe the origins of software architecture and its relationship to other parts of software engineering, discuss its current use in industry, and provide a look at several promising research directions.

48 ページから 68 ページには，その講演で使われた OHP のコピーを収録してある．

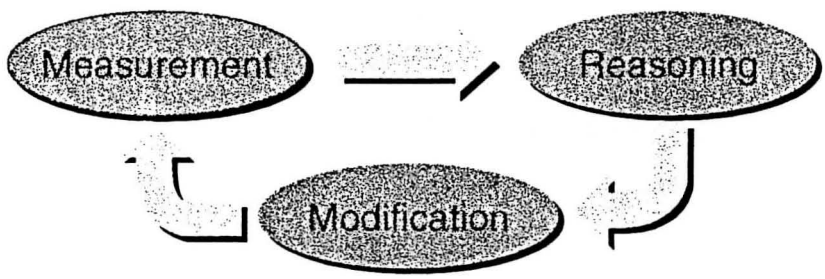
Does Better Process Lead to Better Product? A Case Study in Process Data Analysis

Alexander L. Wolf

Software Engineering Research Laboratory
University of Colorado at Boulder

http://www.cs.colorado.edu/users/alw

Dynamic Process Improvement Cycle



A map won't help you plan where to go unless you already know where you are

Definitions

- Process Model
  - abstraction of a process captured in a notation
  - often represents *desired* process performance
- Process Execution
  - *actual* performance of a process by actors
- Process Event
  - identifiable, instantaneous *action* used to characterize the *behavior* of a process

An Event Taxonomy

EVENT CLASS	EXAMPLE
Automation	invoke tool
Communication	call manager
Analysis	review document
Work	modify document
Decision	accept document
Workday	attend training

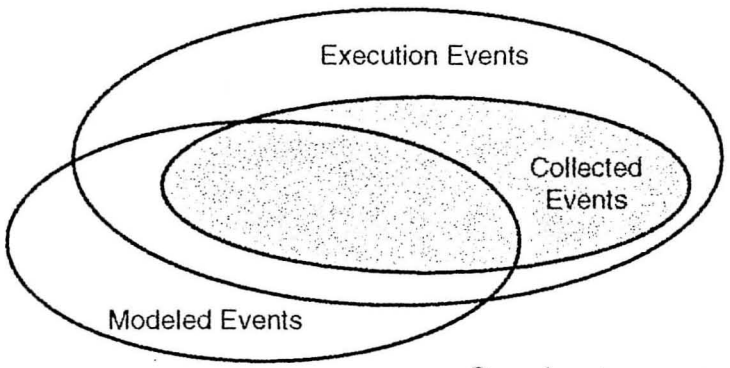
Process Event Data Analysis

- Create Visualizations
- Infer Behavioral Properties
- Discover Process Models
- Validate Process Models and Executions
- . . .

Dilemma: Collecting Process Events

- On-Computer
  - tool invocations, document modifications, ...
- Off-computer
  - reviews, inspections, decisions, ...
- Communication: On or Off Computer?
  - person-to-person, telephone, email

Which Events are Visible?



Can be too *expensive* to collect *all* events

Opportunistic Data Collection

- Direct Observation Logs
- Participant Logs
- Tool Logs

- Result: an interwoven stream of events

Many organizations are already collecting process event data --- but they don't know it!

Sample Event Stream

95/01/19 00:00:00 **carod-abstract** IC34811 mtso support - software load problems
95/01/19 00:00:00 **carod-create** IC34811 jcook clos CTSO-IC 52334 2 support1
95/01/19 00:00:00 **carod-dcreated** IC34811 mach!reserve
95/01/23 00:00:00 **carod-dupdated** IC34811 mach!support2
95/02/16 09:39:44 **mr-createmymr** developer1 command=createmymr abst=AI: CAROD IC34811 add
95/02/16 09:47:46 **mr-acceptmr** developer1 g=1000L6.1f mr=MR52334 tiu=nontiumr
95/02/16 09:47:48 **mr-assign** developer1 g=1000L6.1f mr=MR52334 pgmr=developer1
prio=medium tiu=nontiumr
95/02/16 09:48:02 **mr-asnmymr** developer1 g=1000L6.1f mr=MR52334
95/02/16 09:51:09 **code-checkin** l6sccs/db/forms/com/s.ecp.fs change developer1 MR52334 1
95/02/16 09:52:49 **code-checkin** l6sccs/db/ftamrt/com/s.ecpinup.c change developer1 MR52334 1
95/02/16 00:00:00 **code-inspect** 52334 ECP_FORM 1 0 6 0 0 0.0 0.0 1 0 0 1
95/02/16 10:37:31 **mr-test-plan** (developer1)
95/02/20 12:55:01 **mr-smit** developer1 g=1000L6.1f mr=MR52334
95/02/20 12:56:26 **mr-smitsys** developer1 g=1000L6.1f mr=MR52334
95/03/22 07:26:26 **mr-bwmbuilt** bwmadm g=1000L6.1f mr=MR52496,MR5
95/03/28 17:50:31 **mr-acptsys** bwmadm g=1000L6.1f mr=MR52496,MR5
95/04/19 00:00:00 **carod-solved** IC34811
95/06/20 00:00:00 **carod-update** IC34811 clos 06-20 closed

Industrial Case Study

- Conducted at AT&T (Lucent Technologies)
- Collaborators
  - David Christensen (Bell Labs Wireless)
  - Jonathan E. Cook (University of Colorado)
  - Lawrence G. Votta (Bell Labs Research)
  - Alexander L. Wolf (University of Colorado)

Definitions for Case Study

- Good *Process*
  - a process that follows the prescribed model
- Good *Product*
  - a product that is accepted by the customer

Motivating Question

Much effort and money are spent in defining a "good" process model

Does following this process result in a "good" product?

A Further Motivating Question

Projects routinely collect process data, but either don't know they exist or don't know what to do with them

Can useful information about the process be gleaned from these readily available data?

Case Study

- Goal: to correlate process behavior with customer acceptance of product
- Design: a naturally occurring process
  - process execution is the independent variable and is not manipulated
    - » exploit readily available data
  - decision (accept or reject)n by the customer is the dependent variable
  - 141 accepts vs. 18 rejects

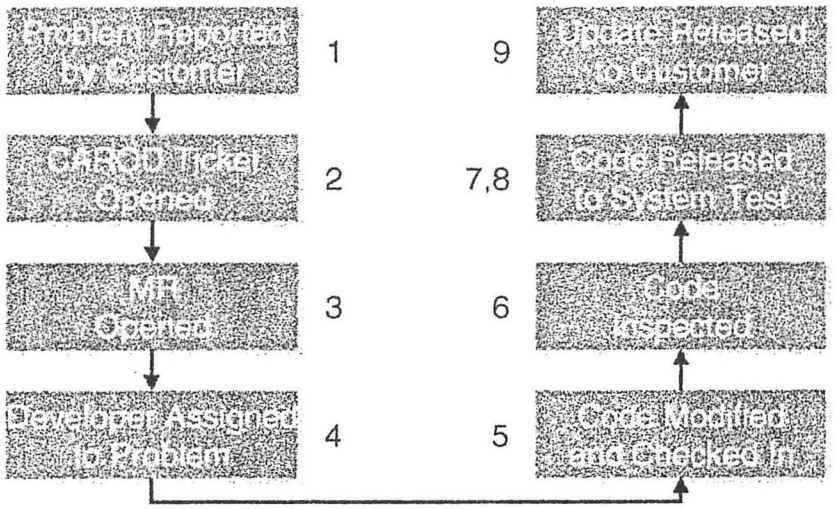
Threats to Validity

- Construct Validity
  - customer rejection as the defect metric
  - most other data automatically collected
- Internal Validity
  - no conclusions of causality
- External Validity
  - small, repetitive maintenance process

Process Under Study

- Customer-problem fix process
- One developer per fix
- Software update released to customer
- Customer either accepts or rejects the update

Process Under Study (Simplistic View)



Sources of Data

- CAROD database
  - provides customer interaction events
- MR logs
  - provide change tracking events
- SCCS logs
  - provide source code change events
- Code inspection database
  - provides inspection events

Aggregate Metrics

Metric	P-Value	Accept	Reject
NCSL	0.23	217.22	166.22
NFiles	0.32	2.75	2.94
NEvents	0.25	30.34	35.22
CTime	0.19	165.94	180.83
DTime	0.00	18.20	68.17
MTime	0.72	96.89	87.94
Developer	Significant		

Results: Aggregate Metrics

- Developer performing the fix correlated with customer acceptance of fix
  - developers have areas of expertise
  - some subsystems harder to change
- Delay in starting the fix correlated with customer rejection of fix
  - problem may be harder to understand
  - developer's understanding may degrade

Are Readily Available Data Useful?

- Answer: YES
- Uncovers well-known correlation of developer to quality
- Uncovers interesting relationship of delay time to success/failure of process
  - organization now tracks delay time

Model/Execution Correspondence

- How well does model predict execution?
- How well does execution follow model?
- Process Validation Methods
  - measure the correspondence of a process execution to a process model
  - identify areas in the model where failed processes differ

Process Validation

- View event streams as sample *strings*
  - a language over the alphabet of the process
- Execution event streams *collected* as data
  - represents single execution of the process
- Model event streams *derived* by analysis
  - many possible streams
- Use *string distance* as conformance metric
  - number of insertion and deletion operations

A Simple Example

model: BCDBCDEFGHJKMN

execution: BCDCDEEFGLKMN

transform: BCDBCDE FG HJKMN

Insertion and deletion operations transform the execution stream into the model stream

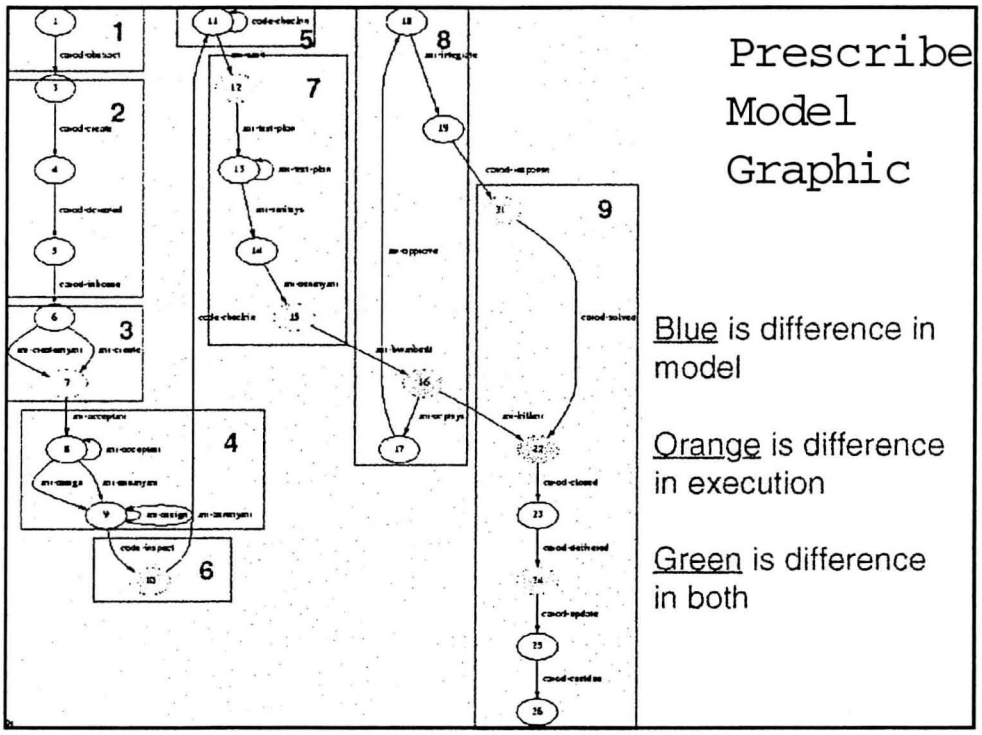
Insertion interpreted as missed event, deletion interpreted as extra event

Two Process Validation Methods

- Simple Stream Distance (SSD)
  - simple weighted sum of operations normalized against maximum weight and length of stream
- Non-linear Stream Distance (NSD)
  - exponential weighted sum of operation blocks normalized against maximum weight and length of stream

Prescribed Model Metrics

Metric	P-Value	Accept	Reject
SSD	0.19	0.65	0.72
NSD	0.01	29.75	88.58
Matches	0.62	17.93	16.11
Missed	0.80	6.96	7.33
Extra	0.09	10.77	14.56
M-Blocks	0.22	3.30	3.17
E-Blocks	0.45	4.88	5.22



Results: Prescribed Model

- Model an ideal sequence of process steps
- Only about half of the behavior matched the model
- Area around code check-ins measured significantly less correspondence for rejected fixes
- Area around mr-test-plan also measured significantly less correspondence

Discovering Model of "Normal" Process

Only about half the process behavior is accounted for by the prescribed model

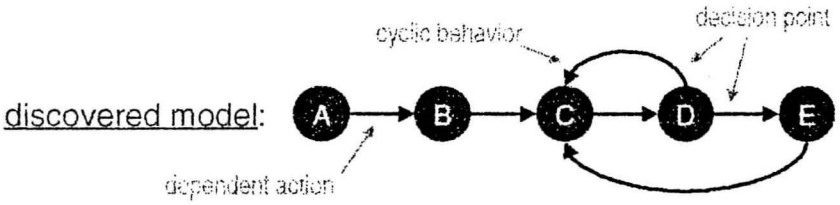
Is there a model of the process that more accurately captures the process as executed, as opposed to prescribed?

Process Discovery

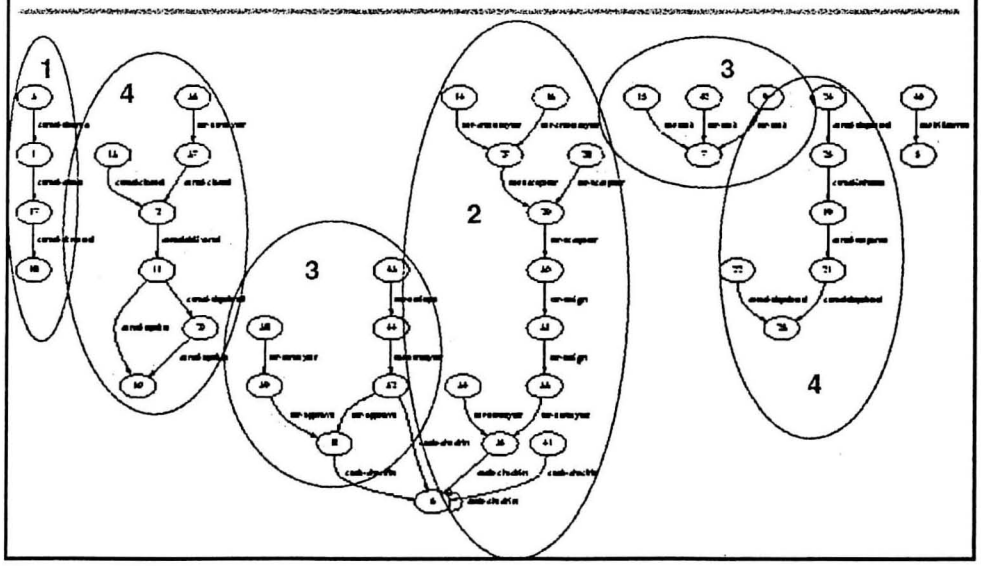
- View event streams as sample *strings*
  - a language over some alphabet
- Use *grammar inference* to discover the language
  - finite state machines
- Infer the *behavioral structure* of the process from the finite state machine
  - sequences, selections, and iterations

A Simple Example

sample event stream: A B C D C D E C

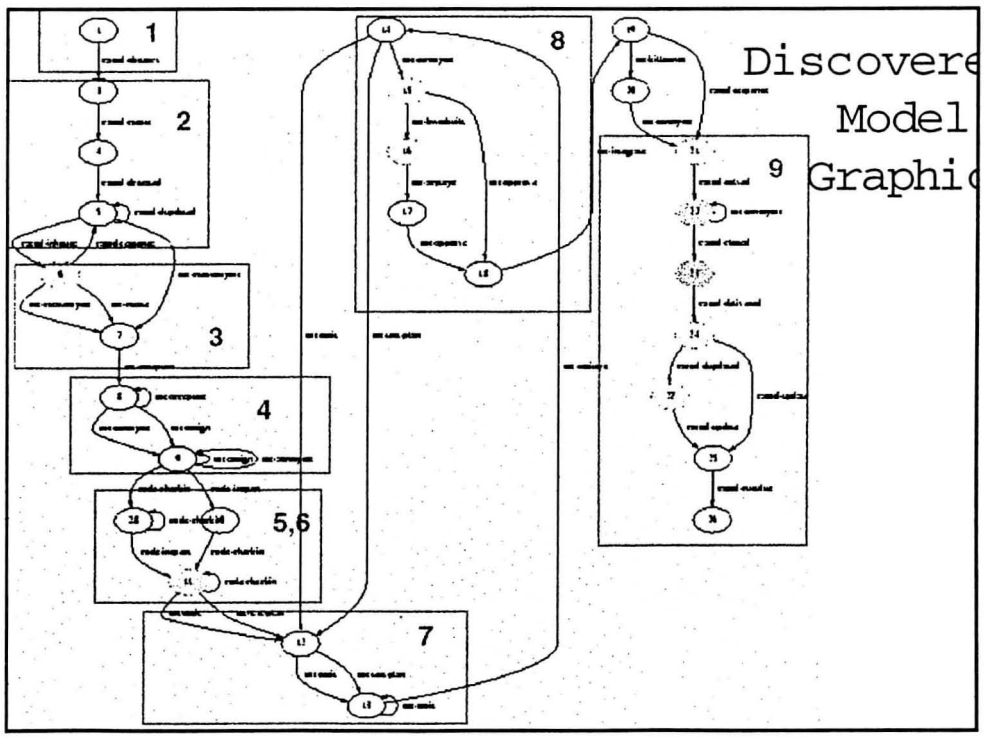


Discovered Model



Discovered Model Metrics

Metric	P-Value	Accept	Reject
SSD	0.41	0.56	0.61
NSD	0.00	24.49	59.96
Matches	0.79	21.15	21.39
Missed	0.10	7.18	8.28
Extra	0.27	7.71	10.17
M-Blocks	0.49	3.48	3.22
E-Blocks	0.74	4.20	4.22



Results: Discovered Model

- Model was generated from the data itself using process discovery methods
- About two-thirds of the events matched the model
  - indicates much variability in the process
- Code check-ins again significant
- Customer delivery significantly different

Can Conformance Lead to Quality?

- Answer: MAYBE
- Converse true
  - Process deviations correlate with product problems
- Questions raised about quality of model
  - organization now reformulating the model upon which decisions are being made

Case Study Conclusions

- Readily available data can be useful
  - avoids time/cost of undirected instrumentation
  - points to areas for in-depth study
- Statistical results show that deviating from the "normal" process can be dangerous
  - but, prescribed process may not be normal
  - first documented evidence of correlation

Important Lessons Learned

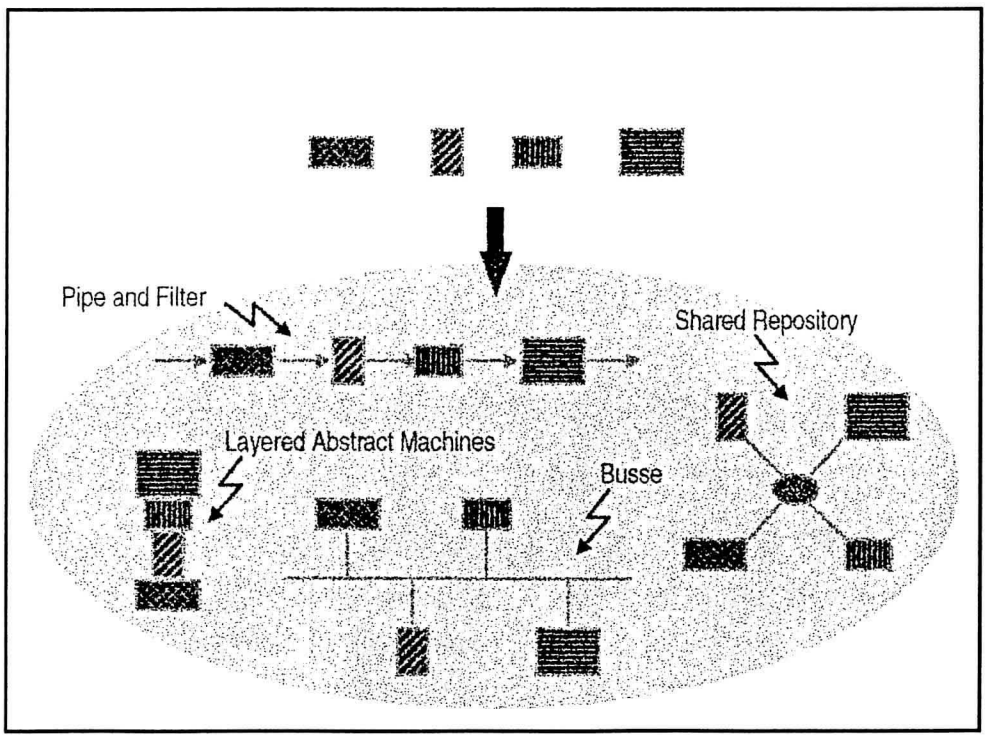
- Process data may be hiding in your organization: Put them to use!
- Management models of processes may not be accurate: Don't trust them!
- Process data analysis is a powerful tool that does not have to be expensive: Try it!

Software Architecture

Alexander L. Wolf

Software Engineering Research Laboratory
University of Colorado at Boulder

http://www.cs.colorado.edu/users/alw

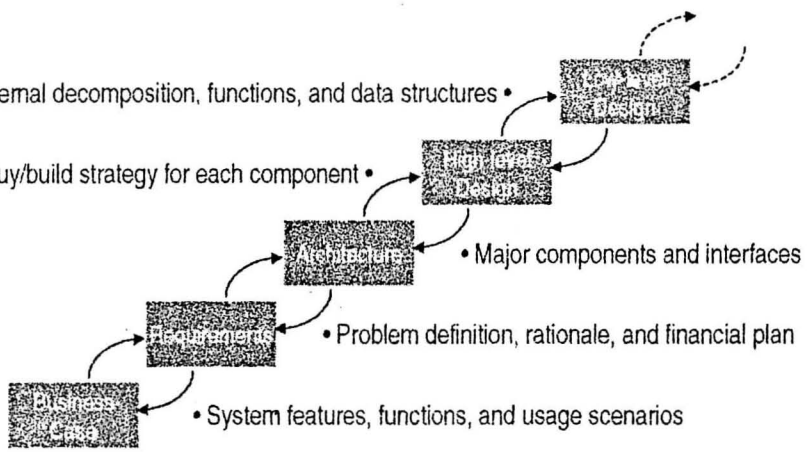


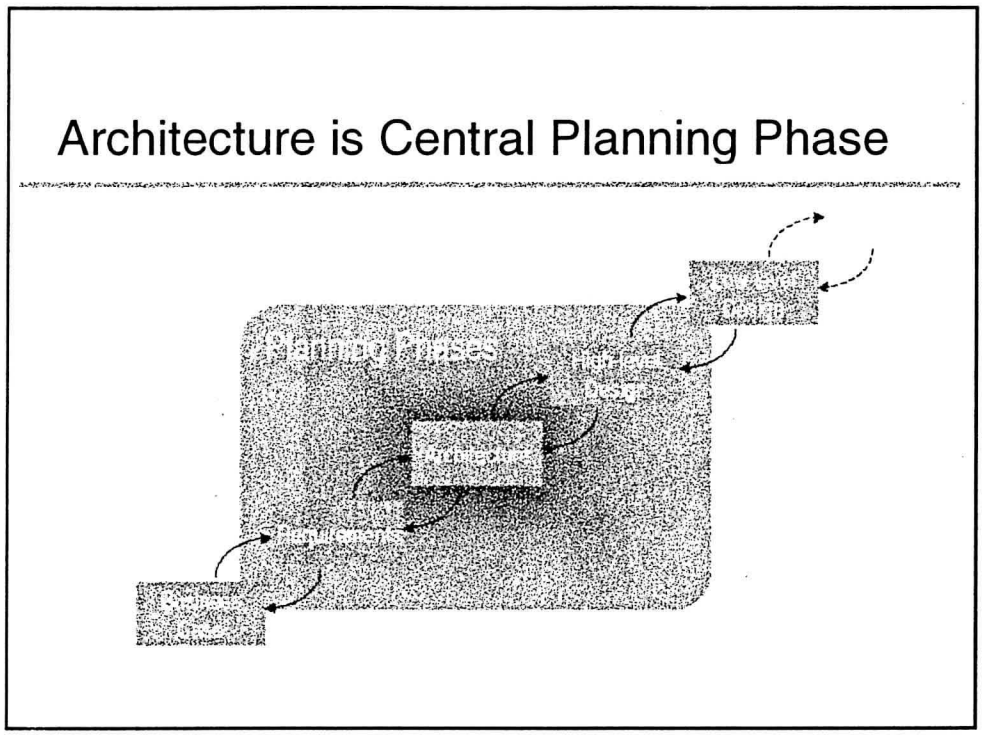
Software Architecture

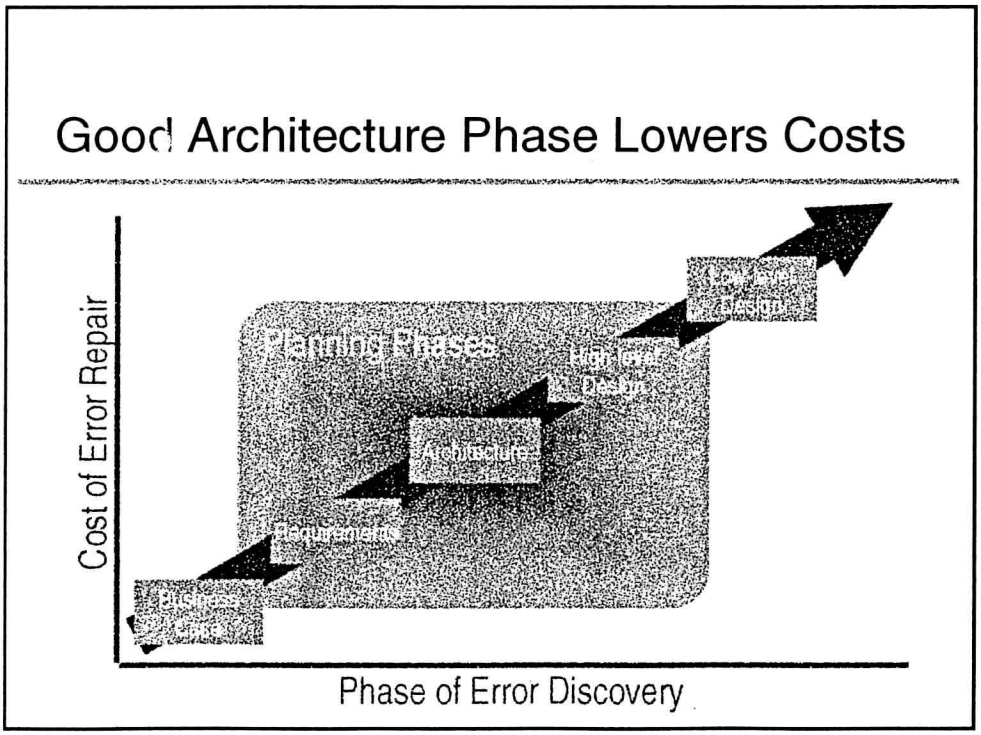
The principled study of software components, including their properties, relationships, and patterns of combination

Also, a particular set of software components as combined in a particular software system

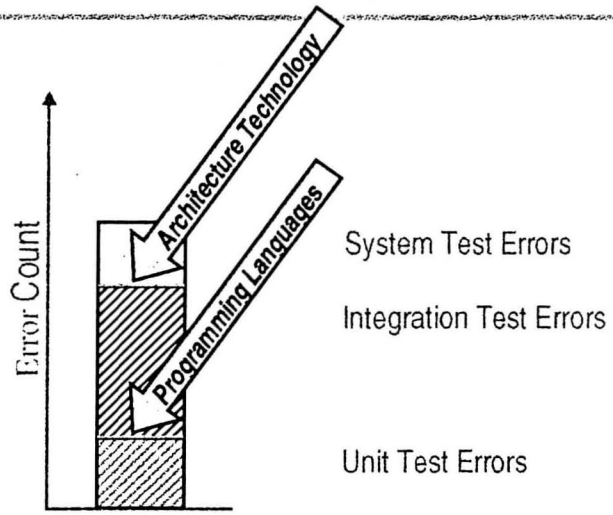
Role of Architecture in the Process




Architecture is Central Planning Phase
Planning Phases


Good Architecture Phase Lowers Costs
Cost of Error Repair
Planning Phases
Phase of Error Discovery

Architecture Affects Integration Issues



Architectural Domains

Application Domain

- Translation
- Data Processing
- Transaction Processing
- Telephone Call Processing
- Flight Dynamics Control

Computer Science Domain

- Busse
- Pipes and Filters
- Layered Abstract Machines
- Shared Repository
- Clients and Servers

Problems ↔ Solutions

Application Domain

A program is compiled by successive application of lexical analysis, syntactic analysis, semantic analysis, and code generation.

Computer Science Domain

The output of one component is the input to another component.

Problems ↔ Solutions

Application Domain

Each agent in the telephone network is a separate computational entity.

Computer Science Domain

Each operating system process is a separate computational entity.

Problems ↔ Solutions

Application Domain

Network services are organized as layers, where each layer adds value to services of lower layers. Services at a layer are defined independently of how they are performed.

Computer Science Domain

Components are organized as layers, such that each layer provides services only to next higher layer and uses services only of next lower layer.

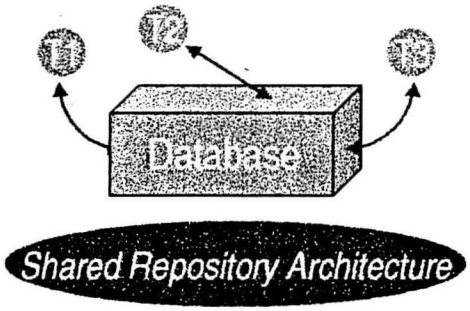
Example: Database Architectures

- Traditional Business Data Processing
  - processing steps are independent programs
  - each step completes before next step starts
  - data stored and passed through tapes

Batch Sequential Architecture

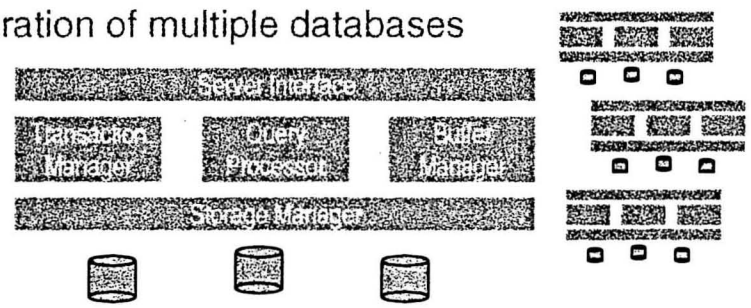
Example: Database Architectures

- Centralized Database
  - monolithic query/update/store engine
  - applications as serializable transactions



Example: Database Architectures

- Modern Database Toolkits
  - discrete, interchangeable components
  - applications as distributed clients
  - integration of multiple databases



Software Architecture Specification

- Design Elements
- Form
  relationships among elements
- Rationale
  justification or arguments for choices of elements and form
- Constraints
  properties and weights

Design Elements

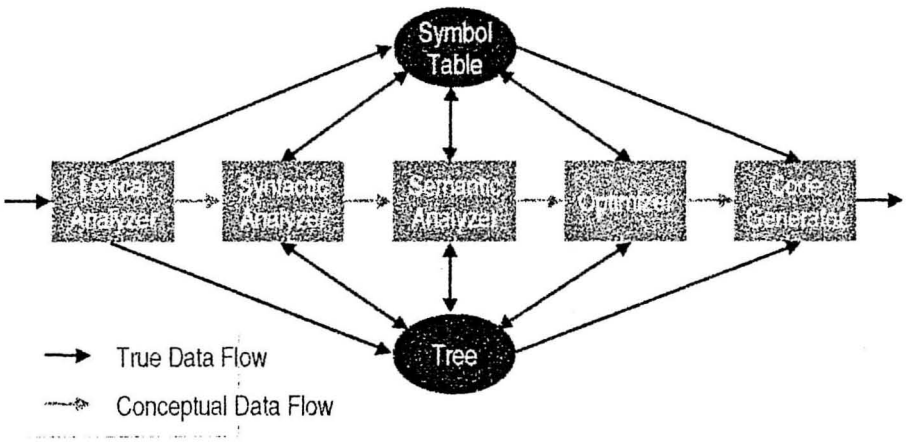
- Processors/Functions/Transformers/Actors
- Data/Information
- Connectors/Glue

- A Useful Metaphor
  Consider Polo, Water Polo, and Soccer: Similar in processors and data, but differ in connectors

Form

- Approaches
  - Formal
  - Picture
  - Analogy
  - Prose
  - Prototype
  - Contrast
- Abstraction
  - Complete
  - Modular
  - "White lie"
- Audience/Purpose
  - Adoption
  - Building
  - Education
  - Comparison
  - Analysis

Example: Language Compiler



Example: Language Compiler

processing element *Lexer*
imports data element *Characters*
exports data element *Tokens*

processing element *Parser*
imports data element *Tokens*
exports data element *Phrases*

connecting element *Pipe*
connects *Lexer* to *Parser*

Major New Thrust: Design Patterns

- Design Method for Object-oriented Systems
- Generic Solutions to Generic Problems
- Design by *Pattern Selection*
- Distilled from Practical Experience
- Implementations Readily Available

Some Design Patterns

- Creational Patterns
  - Factory
  - Builder
  - Singleton
- Structural Patterns
  - Adaptor
  - Bridge
  - Facade
  - Composite
  - Decorator
  - Proxy
- Behavioral Patterns
  - Command
  - Iterator
  - Observer
  - Interpreter
  - Mediator
  - Strategy

Elements of a Pattern

- Name
  How to refer to the pattern
- Problem
  Where to apply the pattern
- Solution
  What abstractly constitutes the pattern
- Consequences
  When to use or not use the pattern

Are Patterns Architecture Technology?

- Encourage Reuse
- Rigorously Defined
- Higher Level of Abstraction

But...

really a programming aid

very small grain, at level of individual

Formalizing Architectural Specifications

- Structure (Form)

  How is the system organized?
- Function

  What does the system compute?
- Compatibility

  When is a system properly composed?
- Specializations

  How are generic systems constrained?

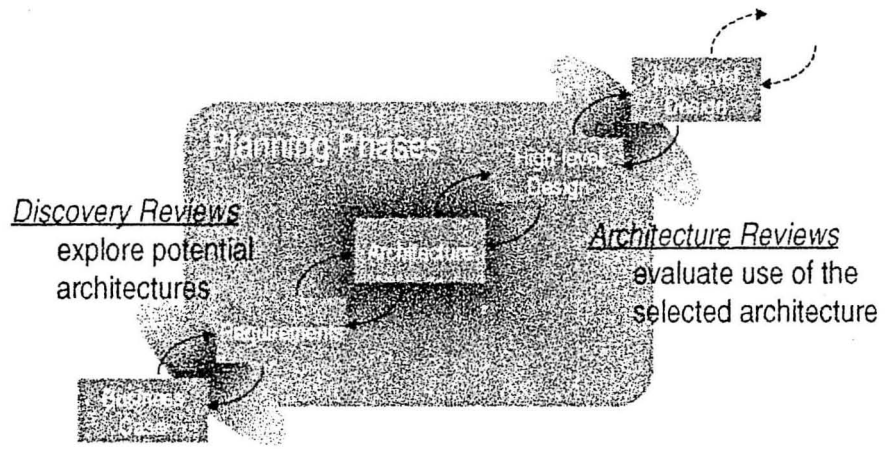
Benefit of Formalization: Analysis

- Consistency of Style Constraints
- Satisfaction of Style by Architecture
- Satisfaction of Requirements by Architecture and of Architecture by Implementation
- Consistency of Structure and of Behavior
- Effects of Changes

Software Architecture Validation

- Quality Improvement Technique
  - developed at AT&T Bell Laboratories
  - part of *Best Current Practices* program
- Goals...
  - assure reasonable and cost-effective designs
  - detect fundamental design errors early
  - reuse corporate asset base
- Uses Formal Reviews and Checklists

Timing of Reviews



Formal Reviews

- Team Effort Involving Whole Project
  - raises sensitivity of engineers to architecture
- Participants
  - *Organization Manager*
  - *Project Manager*
  - *Project Team*
  - *Planning and Architecture Team*
  - *Architecture Review Board*

Organization Manager

- Sets Tone for All Projects in Organization
  - highest-level buy-in is critical
- Provides Leadership and Resources
  - importance of architecture and reuse
  - costs of creating documentation
  - costs of reviewing documents

Project Manager

- Acts as Advocate for Customer Needs
- Seeks *Simplest* Design Possible
- Identifies High-valued Customer Features
  - uses scenarios to walk through implementation

Project Team (PT)

- Creates Design Documents
- Prepares for Presentations at Reviews
- Refers to Checklist
- Maintains On-line Activity Diary
  - open to all team members to monitor status

Planning and Architecture Team (PAT)

- Responsible for Developing Architecture
- Membership: Small-team Subset of PT
  - Product Managers (marketing)
  - System Analysts (requirements)
  - Engineers (design and development)
  - Testers (quality assurance)
  - Users (customer)
  - Chairperson (team facilitator)

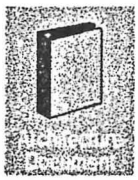
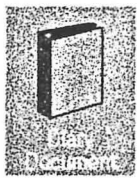
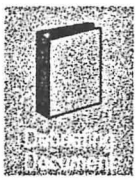
Planning and Architecture Team (PAT)

- Approach
  - iterative refinement of architecture, with members as catalysts for information flow
  - architecture represents consensus view
- Life span
  - created after business case
  - first event: discovery review
  - second event: architecture review
  - monitoring role during implementation phase

Architecture Review Board (ARB)

- Develops a Project Assessment
  - based on requirements and design documents
  - generates list of questions
  - identifies strengths and weaknesses of design
- Membership
  - chairperson
  - 3-5 outside experts

Project Design Documents

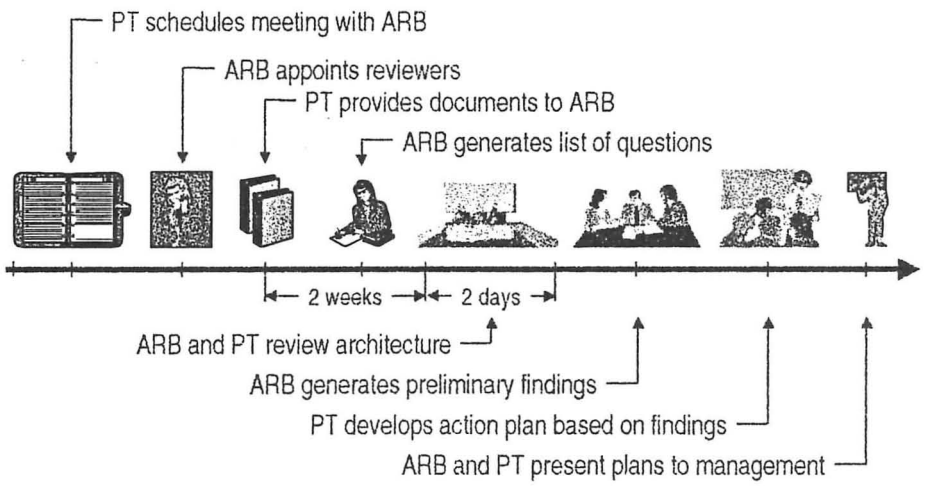




- project scope
- assumptions
- constraints
- data/process flow
- performance goal
- risks

- decisions
- rejected options
- rationale
- audit trail

- postmortem
- good experiences
- bad experiences
- recommendations

Formal Review Process



Experience

- Over 200 Projects Reviewed Since 1988
- Benefits
  - preparation for review led to better understanding of product by entire project team
  - checklist standardizes the design process
  - geater attention paid to OA&M, performance, and error recovery
  - information exchange led to greater reuse

Checklist

- Stimulates the Thinking Process
  - used during design and review preparation
- Based on Experience
  - cover areas of major concern found by ARBs
- Areas Covered by Checklists
  - error recovery (36)
  - operations, administration, & maintenance (53)
  - performance (74)

Checklist Examples

Can the system operate without the database?

What is the error rate expected in parsing data input?

What are the failure modes?

Is there anything that prevents running on a multiprocessor?

How many environments are there for maintaining the system?

Can disk space be recovered while the system is active?

How much headroom have you built into the architecture?

Are there processes through which the system is single threaded?

How many bytes per second of message data must be processed?

Software Architecture Today

- A Leading Research Topic
  - formal approaches
  - context for reuse
  - context for reengineering ("archeology")
  - domain-specific architectures
- Gaining Acceptance in Industry
  - formation of architecture teams
  - architecture-driven product strategies
  - view of architectures as corporate assets

Some Significant Open Problems

- Extra-functional Properties
  - performance, cost, experience, reliability, ...
  - need specification and reasoning techniques
- Education and Training
  - need "architects", not "carpenters"
  - need rewards for architects, not hackers

ソフトウェア技術者協会
〒160 東京都新宿区四谷3-12 丸正ビル5F
TEL.03-3356-1077　FAX.03-3356-1072