SEAMAIL

Newsletter from Software Engineers Association

Vol. 14, Number 6 May, 2005

目　次

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

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

その主な活動は，機関誌 SEAMAIL の発行，支部および研究分科会の運営，セミナー／ワークショップ／シンポジウムなどのイベントの開催，および内外の関係諸団体との交流です．発足当初約 200人にすぎなかった会員数もその後増加し，現在，北は北海道から南は沖縄まで，400 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 19社を数えます．支部は，東京以外に，関西，横浜，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．

ＳＥＡＭＡＩＬ Ｖｏｌ．１４，Ｎｏ．６ ２００５年５月１０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）
〒１６０-０００４ 東京都新宿区四谷３－１２ 丸正ビル５Ｆ
T: 03-3356-1077 F: 03-3356-1072 E-mail: sea@sea.or.jp URL: http://www.sea.jp
印刷所 市田印刷株式会社 〒１１４-００１４ 東京都北区田端２－３－２５
定価 ５００円 （禁無断転載）

編集部から

☆

あいかわらず原稿集めに苦労しています．

☆☆

巻頭の蔵川先生のエッセイは，６月に富山で開催されるSS2005でのパネル討論の企画案です．興味深い討論が展開されるものと予想されます．みなさんもぜひ富山へお出かけください．

☆☆☆

玉井先生からは，ものを「書く」職業としてのプログラミングについて文章技法の観点からの分析を試みたエッセイをお寄せいただきました．

☆☆☆☆

鈴木裕信さん＆新谷勝利さんからの投稿は，中国およびアメリカで開かれた国際会議の参加報告です．SEA 会員のみなさんもいろいろな会議に参加される機会が多いと思います．簡単なメモ程度で結構ですから，その報告をお寄せいただけると，編集部はたいへん幸せです．

☆☆☆☆☆

次号は２月末に行われたデザインワークショップの報告を特集する予定です．

☆☆☆☆☆☆

これから夏にかけては，さまざまなイベントが国の内外で予定されています．それらの報告も次々号には載せたいと考えています．

☆☆☆☆☆☆☆

メタ設計論による設計行為の理解

－ SS2005 in 富山におけるパネル討論の企画 －

蔵川 圭

(NAIST)

来る６月８～１０日に富山国際会議場で開催される予定のソフトウェア・シンポジウム 2005 では，一昨年の SS2003 クロージングおよび昨年の SS2004 におけるパネルの後を受けて，表題のようなパネル討論が計画されています．コーディネータは NAIST の蔵川圭先生，パネル・メンバーは，伊藤昌夫(ニルソフトウェア)，落水浩一郎(JAIST)，酒匂寛(Designers' Den)，佐原伸(JFITS)の４人です．

以下に載せるのは，蔵川先生がまとめられたパネル企画の趣意書です．きわめて内容の濃い充実した討論が期待されます．SS2005 への参加をお考えの方々はぜひこれをお読みになって，フロアからの活発な発言を期待します．　　(編集部)

１．はじめに

日本のソフトウェア産業の技術にかかる国際競争力の一面を捉えることのできるデータがある．総務省統計局科学技術研究調査統計表を見ると，2003 年度産業別技術輸出対価受取額および輸入対価支払額(企業等)について，製造業が 1,469,012 億円および 1,317,629 億円であるのに対し，ソフトウェア・情報処理業では 5,801 億円および 8,744 億円である．技術輸出入対価とは，パテントや技術指導などの対価であるという．ソフトウェア技術は輸入超過であり，わが国独自技術の向上がいっそう必要であることが見て取れる．

木村英紀はこれまでのソフトウェア振興策が，わが国のソフトウェアの弱さを表面的に捉え，企業の目先のニーズのみに注目して場当たり的に対処してきたことに根本的な問題があると指摘する[1]．また，藤本隆宏は，モジュラー設計思想のデジタル財で米国経済が躍進し，この分野で日本企業が不振だったことから，逆に，一部日本企業の得意技が「統合型もの造り能力」と「擦り合わせ製品」であるという歴史的構図が浮き彫りになってきたと指摘する[2]．ソフトウェア技術を向上させるような本質的なわが国における展開が必要であるということであろう．

さて，ソフトウェア技術を向上させ，ひいてはソフトウェア産業の活性化を図るためには，ソフトウェア独自の知識体系を理解しなければならない．計算論から始まり，計算機アーキテクチャ，OS，プログラミング言語，言語処理系，データ構造とアルゴリズム，ネットワーク，ソフトウェア構成論などであろう．我々はこれらのことがわかることと同時に，ソフトウェアを製品としてQCDを意識して市場に投入しなければならない．ソフトウェアを対象としたエンジニアリングを指向しなければならないのである．

ソフトウェアエンジニアリング技術向上のためには，哲学的議論から具体論まで様々なレベルで，要求定義，設計，コーディング，テストなどの技術論から，品質保証，組織論まで産業構造のすべてを網羅して攻略する必要がある．特にそのエンジニアリングの中心にある設計を対象として，ソフトウェアの設計とは何かを問うことは独自技術を獲得するには必要不可欠な理解である．そこで，本パネルでは，ソフトウェア設計の本質を議論によって明らかにすることを目指す．

なお，本パネルのテーマは一昨年(SS2003)の熊谷による基調講演「実践的なソフトウェア開発方法論」の問いかけと昨年(SS2004)の「メタ設計論による設計行為の改善」と題したパネルの続編である．

2．議論のアプローチ

ある概念を理解するためには，ある概念と同時に対としてある概念でないものを想起する必要がある．たとえば，「いす」と「いすでないもの」，「自動車」と「自動車でないもの」という具合であろう．すると，ここでは「ソフトウェア」と「ソフトウェアでないもの」の区別をつける必要があろう．さらに設計というコンテキストを枠組みとして規定して，「ソフトウェアの設計」と「ソフトウェアではない設計」の双方を理解することによって，よりソフトウェアの設計とは何かという本質に迫ることが期待できる．ソフトウェアの設計とは何かという問いに対し，ソフトウェアの構成要素を基礎として既存の設計方法を比較して議論することを外延的理解，ソフトウェアの設計と他の設計を比較して議論することを内包的理解と呼ぶことができよう．本質に迫るためには，この両側面からの理解が必要である．

パネル討論のために，まず，蔵川より，ものづくり・人工物設計という視点から俯瞰してソフトウェア設計について内包的理解を提供する．そして，伊藤，落水，酒匂，佐原，各氏より各自のソフトウェア設計についての外延的理解を提供する．討論では，蔵川の問いかけに応じて，伊藤，落水，酒匂，佐原の各氏が答えつつ，全体として議論を深める．

3．ソフトウェア設計の内包的理解から外延的理解へ

人工物一般を対象とした設計についての議論は，1968 年に経済学者である H.A.Simon が The Sciences of the Artificial[3]に著わしたのがはじめであろう．Simon は，広範囲にわたる膨大な文献を参考としながら，設計とは意思決定過程であることを基礎として，設計対象を設計対象とその環境とに分割して捉え，設計者の認識について議論している．設計対象そのものは，設計者の認識の中で複雑性を帯びたものであり，複雑性に対処する方法として階層的構造をとる場合が多いことを述べている．

また，工学者である吉川弘之は 1979 年に，設計を機能集合から属性集合への写像として位相空間論を用いて数学的に定式化した[4]．数学者の角田譲は一般設計学を数学的に基礎付け，Barwise & Seligman の情報の流れの理論であるチャンネル理論を用いて，設計を観念の世界と現実世界を結ぶ情報の流れとして定式化した[5]．

ソフトウェアの設計を他の工学分野における設計と比較するときに，設計対象の性質に注目することもできる．玉井哲雄によれば[6]，ソフトウェアという対象は機械や電気のような物理性をもたず，抽象的なものである点が大きな特徴であり，ソフトウェア工学は理学としての計算機科学や数学との距離が近いという．また，プログラミング言語による記述や要求仕様・設計の記述という点で言語表現行為に近いという特性もあり，人文科学とも親近性を持つという．

ソフトウェア実体の本質を IBM OS/360 を主とする実務体験から F.P.Brooks は言い当てている．Brooks が The Mythical Man Month[7]の著作で根源的に言いたかったことは，ソフトウェア実体の本質であり，それによってソフトウェアにかかわるエンジニアが他の工学設計に比べて対処すべき特徴点についてであろうと思われる．ソフトウェア実体の本質とは，データセットやデータ項目間の関係，アルゴリズムや機能呼び出しなどが組み合わさったコンセプトで構成されたもので，同じ概念構造体が多くの異なる表現で表されるという点で抽象的であるが，それにもかかわらず，非常に正確で十分に詳細なものであるという．これによって，ソフトウェアの本質的性質は，複雑性，同調性，可変性，不可視性を持つと結論付けられる．

現代のより大規模・複雑化したソフトウェアシステムにおいても，Brooks のいうソフトウェア実体の本質的性質は変わらない．ソフトウェア設計方法論の変遷は，このようなソフトウェア実体の特徴的性質に対面して，QCD にかかる市場の要求の変化やソフトウェアの利用方法の変化というエンジニアの置かれた状況に呼応している．歴史的に見れば，1968 年に NATO 会議においてソフトウェア危機が叫ばれ，ソフトウェア工学の必要性が認識さ

れるのを期に，構造化分析・設計・プログラミング，オブジェクト指向分析・設計・プログラミング，アスペクト指向プログラミングと徐々に以前の技術を基盤として発展してきている．

たとえば，構造化手法に関していえば，Dijkstra による構造化プログラミング，Jackson による JSD がある．より複雑性や再利用性に対処するためのオブジェクト指向法には，Shlaer と Mellor の Object-oriented system analysis，Wasserman の Object-oriented structured design，Coad と Yourdon の OOA/OOD，Rumbaugh の OMT，Booch の OOD，Jacobson の OOSE/Objectory，それらの統合した表記法としての UML，および開発プロセスとしての UP をあげることができる．

再利用性を考慮して様々なモデリングの粒度を捉えれば，フレームワークやアーキテクチャ，コンポーネントという概念が生まれ，それらにおける典型例としてソフトウエアパターンがあるのであろう．より上位レベルのモデリングからソフトウェア自動生成を考えれば，Model-driven architecture(MDA)となり，MDA を指向した再利用性の向上はソフトウェア適用範囲を限定する一方で効率性を追及する Domain engineering や Software product line につながる．

また，開発プロセスという視点からすれば，一つの典型として UP がとりあげられ，Use case driven object modeling ということに成るのだろうが，Agile method/XP や Test-driven development という方向性もある．クリティカルソフトウェアのようなより設計の完全性を追及する必要のあるシステムのためには，要求仕様の完全性や平行プロセスを含むような複雑な状態遷移を対象とした安全性を検査する Formal methods や Model checking の技術は必要となる．

与えられた要求仕様をどのように作るかだけでなく，顧客の必要とする要求を正確に捉えるための方法も必要であり，要求工学においては Goal-oriented requirements analysis や Scenario-based design をあげることができる．さらに，顧客満足度を向上させるためにはソフトウェア使い勝手を向上させる Usability engineering へと対象は拡大されるのである．

4．論点

SS2003 における熊谷の問いかけは次の 5 つであった．「ソフトウェアの作り方は何から影響を受け変化するか？」，「現在多く使用されているソフトウェア開発方法にはどんなものがあるか？」，「コンピュータによる開発方法の支援は本質的か？」，「他の分野とソフト

ウェア開発方法との関係は何か？」，「近未来のソフトウェア開発方法に求められるものは何か？」

SS2004 におけるパネルにおいて設定した質問は次の 5 つであった．「『設計方法論』の最低構成要素は何か」，「設計方法論の構成要素の適切なバランスとは何か」，「ツールは設計行為の何を効率化できるのか」，「概念モデリングにも『アーキテクチャ』は有効か」，「ソフトウェア開発における適切な抽象化の段階はどれほどか」，「ソフトウェア開発に必要な抽象化能力とは何か、またその能力はどのように訓練されるのか」

本パネルではこれらの質問を念頭に置き，自らの考えを述べ，ソフトウェア設計とは何かという本質的な問いに対し何らかの知見を得たい．

1. 木村英紀，モノつくりからコトつくりへ—横断型基幹科学技術がめざすもの，設計工学シンポジウム講演論文集，pp.23-30，(2004)
2. 藤本隆宏，「設計」概念と産業競争力，設計工学シンポジウム講演論文集，pp.79-84，(2004)
3. H.A. Simon, The Sciences of the Artificial (3rd ed.), MIT Press, (1996) （邦訳：サイモン「システムの科学（第3版）」パーソナルメディア，1999.）
4. 吉川弘之「一般設計学序説」精密機械 Vol.45 No.8, pp20-26 (1979).
5. Kakuda, Y., A mathematical definition of synthetic emergence, Proc. of IWES'99, pp13-20 (1999).
6. 玉井哲雄，ソフトウェア工学の基礎，岩波書店，(2004)
7. F.P. Brooks Jr., The Mythical Mon-Month (20th anniversary edition), Addison-Wesley, (1995) (邦訳：ブルックス「人月の神話[新装版]-狼人間を撃つ銀の弾は無い-」ピアソン・エデュケーション，2002)

プログラミングと文章技法

玉井 哲雄
(東京大学)

以下は(株)日立製作所が実業出版株式会社と共同で発行している「スクール COBOL Report」という雑誌から依頼を受けて書いた原稿を，編集者の許可を得てほぼそのまま SEAMail にも投稿するものである．「スクール COBOL」とは，日立が主に高校などの学校の実習で使われることを意識して製品化している COBOL のようである．その広報用にきれいな冊子を出していて，そこにこれまでも湯淺太一(京大)，大岩元(慶大)，黒川利明(CSK)，西村恕彦(元農工大)，中田育男(法大)，植村俊亮(奈良先端大)，筧捷彦(早大)といった人達が，大体プログラミングを話題にした文章を寄せている．依頼元は私の場合もそうだったが，いずれも元日立，現東京国際大の今城哲二氏であるらしい．

以下の文章がのった雑誌は，2005 年 3 月に発行済みである．

1．3 つのメタファー

ソフトウェア開発のメタファーとして，「書く」「作る」「育てる」のいずれが適切か，という議論がある．あるいは「記述」「工学」「育成」のいずれか，と言いかえてもよい．

歴史的にはまず「書く」があった．確かにプログラミングはプログラミング言語という人工言語による記述作業であり，一種の言語表現行為であるといえよう．プログラムをキーボードで入力し，あるいは視覚的プログラミング言語を用いてマウスによる図形操作でプログラムを作成する現在でも，「プログラムを書く」という表現は生き残っている．

バベッジの解析エンジンのためのプログラムを作り，世界最初のプログラマといわれるエイダ・バイロン(ラブリス夫人)は，その作業を編み物を「編む」ようだと言ったという．また，1960 年代から 70 年代のプログラマは「プログラムを組む」という表現をよく使った．これはもしかすると「活字を組む」作業の連想ではないだろうか．とにかく，これら「編む」や「組む」というメタファーも，「書く」に近い発想である．

これに対して，「工学」として製品を「作る」のがソフトウェア開発だとする見方は，「書く」メタファーへのアンチテーゼとして出てきたものと言えるだろう．言語による記述と

はあくまでも個人的な営みである．しかし工業製品としてのソフトウェアは，確立された方法論に基づき，組織によって系統的に開発されなければならない．すなわち従来の機械工学や電気工学と同じようなエンジニアリングであるべきだ，というわけである．こうして 1960 年代末に「ソフトウェア工学」と命名された分野が誕生した．

しかし，ソフトウェア「工学」が伝統的な工学に追いつこうと努力を重ねるうちに，やはりソフトウェアというしろものは，機械やトランジスタとはだいぶ違うのではないかという疑念も生じてきた．ソフトウェアは抽象的であり，物理的な実体がないという特徴を持つ点で機械やトランジスタと違うということは，初めから分っていた．その上で，ソフトウェアには手を加えやすい，後から機能を追加することが日常茶飯事である，という性質のあることが強く意識されるようになる．機能追加ならよいが，もともと潜んでいる欠陥(バグ)を，運用に入ってから気づいて取り除くということもこれまた普通のことで，他の工業製品では考えられないことだが，このようなバグの修正も「保守」と呼ぶ．しかしこれも，ソフトウェアに手を加えやすいという性質の一面を示すものともいえる．

そこでソフトウェアは，全体の設計を一度にすませて丸ごと開発し提供するというよりは，少しずつ手を加え，機能を高め，使いやすくなるように「育て」ていくというメタファーが有効ではないか，という議論が生れてくる． そうして成長した大きなシステムの内部は，入り組んだ構造となる．それにさらに手を加えていくには，中がどう仕切られその間がどうつながっているか，どこに何が置いてあるか，というような生活的な知識が必要となる．そこでソフトウェアを家や巣に見立て，そこに「住む」あるいは「棲む」というメタファーを考えることもできる．これも家の修理，改築，増築を念頭においているという意味で，「育てる」に近い発想である．

筆者は昨年「ソフトウェア工学の基礎」という本を出し，日頃ソフトウェア工学を研究し教育している立場ゆえ，「工学」メタファーがもっとも適切といいそうなものであるが，実はこの 3 つのメタファーはいずれもある面の真実を表していると思う．中でも一番古い「書く」というメタファーは，いまだにその意義を失っていないと考えるので，ここからは話をそこに絞ってみたい．

2．文章読本

プログラミングが文章を書くという行為に似ているとすれば，よいプログラムを書くにはよい文章を書く技術が参考になるに違いない．こういう考えは昔からあり，有名な文章作成法の本，ストランク&ホワイト著"The Elements of Style"(初版は 1959 年．現在第 4 版が出版されている)を下敷きとして，カーニハンとプローガの"The Elements of

Programming Style"が書かれたことは，よく知られている．

しかし，これは英語の文章法の話をもとにした話である．日本人であるわれわれは，日本語の文章法を参考にしたほうがよいのではないか．筆者は，「文章読本」の類を昔からかなり読んできた．同じ文章読本という表題を持つ書物を，多くの人が書いている．たとえば，谷崎潤一郎，川端康成，三島由紀夫，中村真一郎，丸谷才一，井上ひさしといった作家達が出していて，これらをすべて愛読してきた．そこへ斎藤美奈子「文章読本さん江」(筑摩書房，2002 年）という本が登場した．これは世の文章読本を並べてまとめて料理してみせようという，人が考えなかった企みで作られた書である．そこで取り上げられるのは作家達の文章読本だけでなく，清水幾太郎「論文の書き方」，木下是雄「理科系の作文技術」，本多勝一「日本語の作文技術」のような，文学的な作品より論文や実用文をどう書くか，というものもかなり含まれる．これらもまた，筆者はかなり読んできた．さらに知人が書いたもの，したがってプログラミング畑に近い人によるこの手の本を挙げておくと，木村泉「ワープロ作文技術」，杉原厚吉「どう書くか一理科系のための論文作法」がある．

しかし，斎藤はさらにおびただしい文章読本・作文技術本を捜してきて，その徹底振りは驚嘆に値する．このように汗牛充棟の文章読本ものが共通に勧めるのは，

・簡明に書け
・起承転結を大事にせよ
・ 文のリズムを大事にせよ

といったあたりである．プログラミングにとっても簡明に書くこと，全体をきちんと構成することは，もちろん重要である．文のリズムに直接相当するものはあるいはプログラミングにないかもしれないが，モジュール分割の歯切れよさなどが思い浮かぶ．

また，文章の修練方法でだれもが挙げるのが

・ 名文を読め

である．よいプログラムを読むことがプログラミングの学習にとってきわめて有効であることは，多くの人が主張することであるが，それほど広く実践されてはいないようだ．すぐれたプログラムを鑑賞するには，それなりの読解力と審美眼を必要とする．しかし，プログラマが読まなければいけないのは，残念ながら名プログラムばかりではない．プログラムの保守のためには，およそ駄作・愚作と思われるものも，動いて役に立っている以上，読んで理解する必要が生じうる．その場合でも，読む力が高ければ作業の効率は増す．さ

らに審美眼があれば，そのプログラムをどう書き直すべきかという指針も自ずから生れてこよう．

3．文芸とプログラミング

ちょっとこじつけ臭いと思う読者もおられよう．しかし，プログラムは計算機が読むものでコンパイラに通りさえすればよい，というものではない．やはり自分自身も含めた人間が読むものでもある．それを強調したのが，ドナルド・クヌースの Literate Programming である．これは文芸的プログラミングと訳されたことがあったが，ここでいう literate は「人が読める」というほどの意味ではなかろうか．もちろんそれには，人が読んで楽しめる，鑑賞できる，というニュアンスが含まれているから，「文芸的」というのもあながちおかしくはないかもしれない．で，「文芸」であればまさに「文章読本」の世界ということになる．

クヌースには聖書についての著作もあるし，小説も書いている．しかし文学に造詣の深い計算機科学者はクヌースばかりではない．チューリング賞受賞者のトニー・ホアも，ジャクソン法で著名なマイケル·ジャクソンも，ともにオクスフォード大学の文学部の出身である．かつて筆者はマイケル・ジャクソンの著書を翻訳する機会があったが，イギリス文学が縦横に引用されるので苦労した．とくにすでに古典となっている文学作品の場合，書名や登場人物の表記には定訳があるだろうから，勝手なものをひねり出す訳にいかない．幸い，筆者の勤務先の同僚にディケンズ研究者がいて教えを乞うことができた．

別にプログラマに文学の素養が必要だと言いたいわけではない．しかし，プログラマが書かなければならないのはプログラムだけではない．仕様書，マニュアル，解説書，報告書など，書くことを要求される文書は山ほどある．それを書くのに，簡にして要をえた文章をものす力は，やはり必須のものだろう．

Debian Mini-Conf 2005 / CodeFest 2005 報告

～ちょっとに北京に行って来ました～

鈴木 裕信
(フリーソフトウェアイニシアティブ)

1．はじめに

2005年2月28から3月3日まで北京で開催されたDebian Mini-Conf 2005(2/28-3/1) とCodeFest 2005(3/1-3/2)へ参加してきました。備忘録がわりにランダムに書いていきます。

この旅は、「花粉だらけの日本を抜け出す」ということではなく、ちょっと真面目に次の2点に関してチェックを入れてこようかと思ったからです。

(1) 運営状況のチェック
(2) CJK圏での草の根的活動の動向チェック

第一点目は今後 FSIJ がアジア圏内で協力をどうすべきなのかの考慮材料とするためのものです。北京で開催される草の根ボランティアの運営状況を観察することで、同様な趣旨の会合をわれわれが開催する時に参考となるでしょう。そもそも、アジア圏内でこのようなボランティアベースのカンファレンスが開かれることは、滅多にありません。日本以外の国ではどう対応しているのか観察するのには、よい機会だと考えました。

第二点目は、アジアのCJK圏内の草の根レベルの活動の動向を調べることです。国ではなく意図的に言語圏で区別しています。中国語圏（中国／台湾／香港）の経済規模は、既にかなり大きいことを認識しなければなりません。2003年の GDP 比較で比較すると、中国語圏の経済規模は日本の約4割強になります。経済規模的にはかなり大きなマーケットだといえます。しかし、これまで CJK 圏での Linux の状況は，政府や企業の観点からだけ語られてきており、草の根的な情報は，きわめて断片的な形でしか上がって来ていませんでした。今回は、ボランティアベースの内容を知るには絶好のチャンスでした。

2．概要

2月27日から3月4日までの期間，中国政府は，中国 Open Source Week 2005 ということで、

Asia Debian Mini-Conf 2005(2/28-3/1)
Asia CodeFest 2005(3/1-3/2)
The 5th Asia Open Source Software Symposium 2005 (3/2-3/4)

という3つの会合を主宰しました。実行委員長はRoger Soという Debian 方面ではよく知られている人でしたが、私は Debian な人ではないので、その辺はよくわかりません．オーストラリア出身で現在は香港にいるとのこと。

会場は ADMC2005 と AOSSS2005 が、北京市内にある北京新世紀日航飯店でした。CodeFest 2005 は、北京郊外にある中国のシリコンバレーこと中関村ソフトウェアパークの国家応用軟件産品質量監督検査中心でした。そこにあるビジネスコンプレックスのビルの中に入っています。このビルにはベンチャー企業のオフィスやインテルなど有名企業のブランチも入っていました。

なお、3月2〜4日に開催された The 5th Asia Open Source Software Symposium には、筆者は参加しませんでした。理由は、この会は各国から政府と業界が集まる Political Issue な内容の会議なので、参加するだけ時間の無駄だと思ったからです。

以下は Asia Debian Mini-Conf 2005 のスケジュールです。

Day 1: (Monday, 28 February 2005)

09:00-09:30	Registration
09:30-09:45	Welcome
09:45-10:45	Towards a Systematic Quality Assurance Approach in Debian - Martin Michlmayr
10:45-11:45	How To Help Debian Without Deeper Knowledge - Alexander Schmehl
11:45-13:15	Lunch
13:15-14:15	Status Report of Debian in Taiwan - Andrew Lee
14:15-15:15	RAYS installer -- a customized debian-installer - Stanley Peng
15:15-15:30	Break
15:30-17:00	Key-signing Session

Day 2: (Tuesday, 1 March 2005)

09:00-10:00	Custom Debian Distributions - Andreas Tille
10:00-11:00	Mass Management of Debian Desktops - Scott Dier
11:00-12:00	Searching and CJK - Takatsugu Nokubi
12:00-13:30	Lunch
13:30-14:30	The Debian Linux Kernel - Simon Horman
14:30-15:00	The Linux Public Platform - Wei Chen
15:00-15:15	Break
15:15-16:15	Bootstraping the M32R Architecture - Yukata Niibe
16:15-17:15	The Many Uses of Apt-Listbug - Masato Taruishi
17:15-17:30	Closing

日本からの発表者は、産総研・FSIJ 理事長の g 新部裕氏、凸版印刷・FSIJ の野首貴嗣氏、VA Linux の樽石将人氏の 3 人でした。ADMC と CodeFest の日本人参加者は全部で 10 名でした。Debian の中心人物である Martin Michlmayr らがドイツから招かれていました。

メインスポンサーは Debian ベースの商用 Distribution を開発している新華科技（南京）系統軟件有限公司です。Roger So はここの会社の香港ブランチに勤めているとのこと。日本企業である NTT Data もスポンサーとなっていました。他に日本の組織が 1 つ。これはスポンサー名に名前が出ていませんでしたが、産総研です。会議の実態はといえば、中国 Open Source Week の一環として開催されたという運営方法ではなく、いつもの Debian Conf の規模の小さいものをこのタイミングで行ったという感じでした。このタイミングに会わせたのは新華科技が中国政府へのアピールするためではなかったかと考えられます。

http://www.swhss.com.cn/jp/

* ADMC 2005

http://debian.org.cn/en/events/admc2005

スクール形式で 40 名程度の部屋を使っていましたが、後の方に椅子席をずいぶん追加していたので実際には 60 名以上の参加者がいたと思います。登録は 80 名ぐらいだったということです。

Debian メンバーのプレゼンテーションは、主に Debian とは何かという初歩的な内容でした。どの内容も基本的なというか、導入的な話題と紹介を中心にしていたので、内容的に特にこれはといったものはありませんでした。

私が一番面白かったのは The Linux Public Platform の時に出た質問です。プレゼンの内容は、中国政府が御墨付で開発している Linux 関連の状況紹介でした。Dr. Wei Chen が、いやにテンションの高い演説口調でプレゼンを行いました。超早口の中国語、間違わない、よどまない。「あなた 100 回はそのネタ話していませんか？」という感じで、ほぼ 30 分間スピーチが続きました。立て板に水というのはこのことでしょう。内容的には、中国政府もいろいろ開発支援に力を入れているという事実をアピールしたものでした。質問の時間がきてすぐに、広東省の公営 Linux センターから来たという高級工程師（シニアエンジニア）の人がぱっと手をあげました。

「Linux の開発はコミュニティとの関わりが大切である。そうやって国の支援で開発したコードはどのようにコミュニティに還元するのか？」

これには思わず、g 新部氏と筆者は拍手し、親指をたてて "Good Job!" と声援を送りました。質問を受けた側は「100 回このネタを話したが、今まで、こんな質問は一度として受けたことはない！」みたいな顔をしています。ようやく返ってきた回答は役人的なものでしかありませんでしたが、まあ、それは仕方がない。しかし「コミュニティへの還元」と真面目に突込みを入れる姿を見て、中国も上意下達でやっているのではなく、本当の意味での草の根的な広がりが出てきているのだな、と感心しました。

3．セッションの合間に

今回の ADMC2005 は、所詮は中国における Debian の顔見世興業でした。プレゼンの内容はあまり重要ではありません（プレゼンした人ごめんなさい）。重要なのは、アジア CJK 圏の有力な Debian デベロッパが Face to Face でコミュニケーションできる機会を持った

ということです。ですから、セッションの合間や、セッション後が最も重要な時間です。

今回私は特定非営利活動法人フリーソフトウェアイニシアティブの最高執行責任者という肩書で参加していたので、台湾と香港からきたコミュニティ組織運営に携わる人たちといろいろと話しました。

香港は、せっかく立ち上げようとしてもスポンサーがみつからないという問題があるようです。何かイベントを企画しようにも、先立つ資金が集まらない。香港という土地柄は流通ばかりで生産がないため、開発への投資がない。そうした状況でスポンサーを集めようと思っても、どうにも集まらないという話です。中国にしても、台湾にしても、開発を行っている企業があるので、その流れでスポンサーを集めることはできる、しかし香港はそれがない。開発のために畑を耕す必要も種を撒く必要もないというのは、いかにも香港らしい悩みでした。

台湾は、愛好者的ユーザ会が多く、個々の活動が小さく閉鎖的な傾向にあるというのが問題なようです。そこそこに資源や資金を投入（といっても個人負担できる程度ですが）できるので、小さなコミュニティ活動には支障がない。一方、個人の趣味をちょっと大きくした程度のコミュニティでは、活動自体を大きく発展させていくことができない。こじんまりとやっていれば、そこそこにユーザ会は回り、それなりのおもしろさ（個人的興味の充足）が得られる。しかし、さらに規模を大きくしようとすると、組織化のための労力が必要で、趣味の範囲を越えてしまう（つまらないことが多くなる）ので、やっぱり小さなままでいようとする力が働く、といった状況のようです。そこそこに満足できているという贅沢な悩みが、この問題の底流にはあり、それは日本の状況とよく似ています。

中国本土の人の問題は「英語をまともに話せる人間が見当たらない」ことでしょう。台湾人，香港人は，ちゃんと欧米の人たちとコミュニケーションができるのに、あの名門！清華大学の学生でも、その英語たるや，何をいっているか全然わからない。ちなみに去年アメリカで開催された CRYPTO2004 でセンセーショナルな発表をした山東大学の助教授も同じだったことを思い出します。私の回りにいる中国本土出身者はみんな英語がうまいのにどうしてでしょうか？

また、積極的にコミュニケーションしようという姿勢がないという姿勢も問題だと思われます。自国の中で閉じていても独自でやっていけるから、苦労して国際的なコミュニケーションをする必要がないということなのでしょうか。よくわかりませんが、このコミュニケーションの問題は、日本とよく似ているように感じました。

韓国については、少なくとも今回私は、韓国からの参加者と話をする機会がなかったので、よくわかりません。

ADMC2005 だけではなく、日本での会議もそうですが、アジア圏のカンファレンスに参加して思うことは、ソーシャルプログラムが弱いということです。お互いが集まって情報交換することがこの種の集まりの主目的であるという意識は、まだまだ薄いように感じられます。つまりリストアップすれば次の通り：

- ソーシャルプログラムがきわめて重要なのに貧弱である
- 前日に簡単なウェルカムパーティがあればよかったな
- レセプションも前後の時間に余裕が欲しい
- 昼食も前後に時間的な余裕が欲しい

4．大陸中国的時間尺度

勝手に私は CTSP (Chinese Time Scale Problem)と呼んでいました。ADMC2005 初日夜のレセプションではペキンダックのレストラン店に行きました。レストラン自体はとてもよかったのですが、そこまでの移動が大変。ホテルからバスで移動したのですが、まず、バスが来ない。結局 1 時間ぐらい遅れてやっと到着しました。CodeFest2005 の会場に向かうバスもそうでした。集合時間はあってなきがごとし。噂に聞く中国時間です。集合時間ぴったりに集まるのは日本人とドイツ人のグループだけでした。さらに、観光スケジュールとかも、行きあたりばったり。レセプションの後で、夜の天安門まで観光に出たはいいのですが、そもそも、天安門は夜間は閉鎖されています(そりゃそうだよな)。北京のくそ寒い夜気の中、天安門の周辺を流浪するしかない。翌日も同じようにトライして、やっぱりダメでした。

他の日本人が不満をぐちる中、私自身はこのいい加減さ、時間のルーズさ加減はぜんぜん苦になりませんでした、むしろ馴染んでいるくらい。私の中の大陸的な北海道人の血がそうさせているのでしょうか。ただし、北海道人なので寒さには弱い。正確にいえば、寒くてもいいけれど、ちゃんと室内で身体が温まる所がないと辛い。実は、北京ではこれが一番辛かったです。

5．CodeFest2005

ハッカーが集まってみんなでプログラムをしよう！Face to Face で相談しながらやればあっというまに問題解決さ！という主旨の集まりです。こちらの方は、私自身は、敵状視

察が主目的だったので，自分の手でコードを書く予定はありませんでした。

朝 8 時 30 分が集合時間。でも出発したのは 9 時 30 分近くになってから。バスに 30 分ちょい揺られます。幹線道路の広告パネルには AMD や VIA の広告が延々と続きます。中国では「この通り全部」とかバルクで広告を売っているんだろうか?

さて、清華大学の前を通って、着いたのは中関村軟件園孵化器という建物。周囲はつくば学園都市のような感じで、のっぺらぼーな敷地に、モダンな建物がぽつぽつあるという風景。まわりには立派なレノボのキャンパスがあったり、中関村軟件園孵化器の別の棟には甲骨文(Oracle)が入っていたりしています。同じ建物の中には、IBM Rational Software 軟件工程技術実験室という看板をかかげている部屋などがありました。

コンピュータ教室を CodeFest の会場に使いました。席数（PC 数）は 35。といっても、参加者のほとんどは自分のノート PC を持参し、それで開発しているので、あまり関係ありません。一応、入口付近には立派な CodeFest2005 のスケジュールが掲げられていたのですが、あまり関係なく、一度部屋に入るとあとは自分のペースで作業を進めていきます。

PC には新華科技の RAY LE というディスクトップ用 Linux が入っています。参加者は root のパスワードをもらって使えます。私は開発をする気がなかったので、自前のハードウェアを持参せず、ここの Linux 環境を使わせてもらいました。

g 新部氏が M32R のハードウェアを持ち込んで組み込みをハック。半田さん(m17n.org)と守岡さん(CHISE)は多言語ライブラリをハック。ark-田中さん(ruby-dev) は Ruby 本体をハックと、それぞれ思い思いにハッキングしています。

まあ暇潰しにと、自分の作品であるサーバプログラムを OpenPKSD.ORG からダウンロードしてインストールしてみました。ところが、思いのほか簡単にインストールできずに悪戦苦闘。自分のインストールマニュアルに追加事項やトラブルシューティングを書き始めてしまいました。こんなはずじゃなかったのだけど...

小一時間もすると 11 時 30 分になり、混まないうちに社員食堂にいって早めの昼食。ビュッフェスタイルで、もちろん、いうまでもなく中華です。麺類のコーナーでは、刀削麺を目の前で作っています。私は、いろいろプレートに盛り付けて、ガツガツ食べましたが、社員食堂だと考えれば、けっこう美味しかったと思います。毎日食べても OK です。

さて、部屋に戻り、今度は、サーバが実際に動くかどうかのチェック。うまく動いてー

安心。でも、それだけではつまらないので、鍵の登録を若干早くするためのハックを開始。「ほんとうは視察だけで、午後からは一人で市内観光する予定だったのになぁ」とか思いつつ、プログラムのパフォーマンスのチューンをしている自分がいました。

ここで日本人約 1 名リタイヤ。来た時から体調が悪く休憩室で寝ていたのですが、体調は回復せず、残念ながらタクシを呼んでもらってホテルに帰りました。後で病院へ行って対処してもらったようです。ホテルに帰ってから電話をした所、急性胃腸炎だという診断だったそうです。病院で点滴を打ってもらい回復し翌日無事日本に戻られました。

気が着くと、すでに夕食の時間。残業食を出しているので、そこでまた食事。帰ってくると「夜食のために」と大袋リッツクラッカー、カップ麺、そして 6 本入ソーセージが主催側から配布されました。このソーセージは、日本の魚肉ソーセージではなく、鳥・豚・牛の肉を使っているので、全然味が違います。これまで中国圏で口にしたもので食べられないものはありませんでしたが、唯一これは例外でした。

また、ふと気が着くと、既に 8 時を回っているのではないですか。ちょっとやるだけだったのに。ちなみに、他の参加者は会場で徹夜です。寝袋と休憩室が用意されているので、寝ることもできます。「みんながんばれ！私は帰るけど」。残業帰りの客待ちしているタクシーが何台もメインゲート前に止まっているので、それに乗ってホテルへ戻りました。29 元。

気がついた点を箇条書きにすると次の通り：

- やっぱり自分の開発環境を持って行くべきだった
- 作業場所は、もっと自由に使えるタイプの方がいい
- 海外への高速ネットワークは MUST
- 24 時間以上ぶっ続けなので深夜の飲食が大切
- ちゃんとした休憩場所の確保
- シャワーがあるとハッピー
- 病気の人が出た時のためなどホスピタリティスタッフの確保
- 最初の 1-2 時間は、方向性やグループを作るためのディスカッションをすべき
- 来る前に Wiki で企画をたてておこう
- 来てからも Wiki で情報交換しよう

6．雑感・まとめ

今回は、中国で始めてのユーザ側から立ち上げたオープンソース関連のカンファレンスだったので、様子見の雰囲気もなきにしもあらずだったのですが、小規模な会議とはいえ色々な人が集まったので、なかなか面白かったです。ただADMC2005に韓国からの積極的な参加がなかったのは残念。

香港、台湾にはアクティブなデベロッパがいることをあまり知らなかったので、今回、それが大きな収穫でした。中国でも、われわれの感覚と同じようなセンスを持った人たちが草の根的にはいるんだなぁと、少しうれしく思いました。

Asia Open Source の方はアジア圏内の各国から政府からの御声がかかりの参加者が会議をしていたみたいなのですが、まあ、かんばってくださいということで。

ADMC2005によってアジア圏内の草の根レベルの交流は、確実に第一歩を踏み出したのではないでしょうか。VISA 問題があるので，中国人に参加してもらうということを考えると、中国国内で開催するのがベターというのはわかるのですが、しかし、3月の北京はあまりにも寒いので、もう勘弁してほしい。次回はぜひとも暖かい広州あたりで.!.

追記

北京の首都国際空港の正規のタクシー乗り場で、ちゃんとディスパッチャーに従って乗り込んだタクシーが白タクでした。国の表玄関で規則正しく乗り込んでいるのに「それはいくらなんでも反則だろう」です。案の定、請求金額は予想の倍ぐらいふっかけて来ました。国の代表的な空港にもかかわらず、ライセンスなしのタクシーが待つことができるタクシー乗り場があるというのは、世界中でここだけじゃないでしょうか。うーむ、市場の自由化だ（ちがうって！）。

17th SEPG Conference 2005 参加報告

新谷 勝利
(IPA-SEC)

1．コンファレンス概要

1）会期・場所： 2005年3月7～11日
@ Washington State Convention Center, Seattle

2）参加者数（全体合計：1800名）
－日本からは30名弱
－韓国からは30名強
－インドおよび中国からも多数

3）セッション
－以下のトラックにて10時半から17時まで並列発表（全体合計：106）
－プロセス改善：３２
－特別セッション：８
－CMMI：３１
－TSP/PSP：１９
－測定と分析：１６
－基調講演が2日目、3日目の朝8時半から10時までそれぞれ2セッション、計4
－全体的な印象
SEPG はプロセス改善を目的とする専門家集団あるいは組織を意味する．しがたってれぞれの体験・経験・知見を共有するセッションが一番多く、具体的なツールである CMMI そのものに関するものがほぼ同数。プロセス改善への「人の動機付け」の観点から TSP/PSP、「データによる裏付け」の観点から測定と分析のセッションが準備され、セッション選択に当たっても全体的に統一された観点からできるように配慮されていた。

2．参加したセッションからの報告

2.1. Relection on Process Improvement (John Vu, Boeing)

Software Industry Benchmarking Study 2001 を引用しながら現状を説明するとともに提案

－なぜプロセス改善活動は失敗するか？
　－「評価－＞改善」の順になされるが、「評価」そのものを強調し過ぎ
　－明確な方向性および測定目標無しに、「成熟度」のみに焦点を当てる
　－改善活動を推進する知見ある体制の欠如
　－次から次へと世間で語られる「用語」と自分の現場との差に関する混乱
　－改善活動が適切に管理されていない

　＊ 失敗していることを示すベンチマーク
　　－評価した後に、72％が改善の成果に結びついていないと報告
　　－改善活動にとりかかった組織の内、83％が3年以内に改善活動破棄
　　－上記活動破棄組織の内、57％がその後に再開
　　－改善はうまくいっていると称する組織でさえも、データと共に説明できるのは1％にも満たない

－多くの組織において、「ある成熟度を達成」することのみを目標とし、それを達成すれば「問題が全て解決する」と誤解。

　＊ 仮に「成熟度」達成を外的刺激として改善活動をするとしても、バランススコアカードで定義しているようなビジネス目標を達成することにどのように寄与しているかが説明される必要あり。

－ソフトウェア開発プロセスにおけるワークロード配分は、要求分析に5％、設計に10％、実装に40％、等となっているが、欠陥の発生源をしらべると、要求分析から45％、設計からは25％、実装からは15％、等となっている。長い間よく知られ、しかし、有効な対応策の適用が困難。

　＊ 個別のプロセスの改善というより、プロジェクト・組織全体にPSP/TSPを導入することにより、当該チームの次のリリース全体でプログラムの大きさが2.36倍になったにも関わらず、欠陥数は75％も削減
　＊ プロセスそのもの改善というより、「人」を如何に動機付けさせるか

－IEEE Software Industry Benchmarking Study 2001からの結論：
　－プロセス改善に取り組んでいない企業の内、80％は2006年までに倒産するか危機に陥る
　－（「組織成熟度」ではなく、）「プロセス能力」のレベルを上げられない企業の内、80％は2007年までに少なくとも開発予算の50％超過に陥る

－組織が今後管理の対象とする5つの項目の提案

－プロセス改善が実際のビジネス目標にどのように寄与？
－標準のプロセスに準拠している（していない）プロジェクトはどれ？
－標準のプロセスが個別のプロジェクトレベルに活用され効果を出しているか？
－日々の意思決定が適切な測定値をベースになされているか？
－ビジネス目標に優先位が設定され、グループ間衝突は回避されているか？

2.2. The Great Myths About TSP and CMMI （Jim McHale, SEI）

－TSP と CMMI は、同時に実践できない
－CMMI はモデルであり、TSP はプロジェクトレベルのプラクティスの集合で、CMMI が対象とするプロセス分野の殆どをカバー
－TSP はモデルではないので、CMMI と矛盾せず、モデルとしての CMMI からマッピングされるプロジェクトに参加するスタッフのプラクティスを計画化するのに有効

－TSP を実践するためには、現行のプラクティスを捨て去らなければならない
－TSP チームは、以下をベースにチーム独自のプロセスを定義
－組織の標準を含む現行プラクティス
－チーム員が「より効果的な仕事のためには、こうしなければならない」と考える方法
－上記の組み合わせ

－３つの運用レベル
－CMMI：組織としての能力向上のために
－TSP：　経費とスケジュールを考慮した高品質の製品開発のために
－PSP：　各人のスキルと規律を高めるために

－TSP 導入は難しすぎる
－An Introduction to the Team Software Process, Humphrey, 2000)に説明されている用紙とその使用ガイドの多さからのコメント。しかし、
－21 説明ガイド－繰り返し使用されるものは７つに過ぎない
－51 ページの用紙－4～5 種類の用紙が定期的に使用されるに過ぎない
－TSP を実践するためには、成熟度レベル x でなければならない
－TSP を初期導入するチームの多くは、（モデルに準拠していない）というレベル１
－実績を見ると、どのレベルからでも TSP の初期導入はされている

－TSP は十分アジャイルではない

－Balancing Agility and Discipline, Boehm, Turner, A-W, 2004 によれば、アジャイル度評価において XP と TSP は中程度で隣り合っている
－アジャイル信奉者の多くが、アジャイル手法においてプロセス記述が簡略化されていることをカバーするためにプロセスに関して卓越した知識・経験を持つ人が必要であることに気付いていない

－我々は TSP を導入するには大き過ぎる
－TSP はプロジェクトレベルのプラクティス
－CMMI は、プロジェクトを越えてより広い組織として、標準とデータに基づく管理を実施するために記述したもの
－どんなプラクティスも、そのサイズに関係なく組織に応じた解釈がなされ、導入される必要あり

－TSP 信奉者はレベルをスキップする
－TSP は成熟度レベルに関しては言及しないし、TSP 信奉者はプロセスを目的と測定可能性で議論
－レベルをスキップするということは組織評価に関連する表現であるが、TSP はプロジェクトレベルにおいて導入

2.3 Lessons Learned from Adopting CMMI for Small Organizations (Jack Conway, ASI & Sandra Capeda, CSSA)

－Huntsville にある 25～250 人の中小企業を対象にパイロット
－2003 年 8 月～2004 年 4 月における投入時間と経費

チーム会議	718 時間（37%）	$52K（38%）
電話会議	520 時間（27%）	$37K（26%）
プロセス定義	200 時間（10%）	$16K（11%）
訓練：	84 時間（4%）	$6K（4%）
導入：	183 時間（9%）	$13K（10%）
評価準備：	132 時間（7%）	$9K（6%）
評価：	104 時間（5%）	$7K（5%）
合計	1941 時間	$138K

2.4. Adopting Software Product Lines: Getting Leverage from Your Process Improvement (Linda Northrop, SEI)

－CMMI のレベルが高くなれば、「再利用」の比率が高まっていることは調査の結果分かっている。
－Product Lines は「戦略的な再利用」と位置付けられ、ライフサイクルの早い段階で再利用を取り入れればその効果は大きく、以下の分野における「コア資産」に対する投資を活用
　－要求分析
　－ドメインモデル
　－ソフトウェアアーキテクチャーと設計
　－効率を上げるためのエンジニアリング
　－文書化
　－テスト
　－スタッフ
　－プロセス
　－部品
－プロセス改善活動が Product Lines の導入の基礎
　CMMI プロセス分野に対応するプラクティス分野を定義

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