SPI Japan 2012 基調講演

社会とソフトウェア：
あるソフトウェア工学者の経験

ソフトウェアプロセス改善カンファレンス SPI Japan2012
大阪国際交流センター, 2012.10.10

京都大学大学院文学研究科
現代文化学専攻
情報・史料学教授　林晋

SPI Japan 2012, トップページから

- 今年のテーマは「日本の“カイゼン”力の再生　～次の10年への飛躍～」です。

  JASPICが主催するソフトウェアプロセス改善カンファレンスは今年で10回を数えることとなりました。これまでは失われた10年などと呼ばれ、世界的な経済不安の拡大や度重なる自然災害により、社会全体に停滞ムードが広がっております。このような状況を打破し次の10年に向けた活力を生み出すべく、幅広い知恵を結集いたしました。また今年は昨年までよりもさらにインタラクティブ性を高め、発表者・講演者と聴衆・聴講者が議論をする場を設けて、次の10年へ向けた多くの“気づき”と“元気”を参加した全員が得られることを目指しております。

  本カンファレンスに参加されるすべての方が繋がり、さらに拡がりを見せ、**社会を変革**していこうではありませんか。

SPI Japan 2012, トップページから

- 今年のテーマは「日本の“カイゼン”力の再生　～次の10年への飛躍～」です。

  JASPICが主催するソフトウェアプロセス改善カンファレンスは今年で10回を数えることとなりました。これまでは失われた10年などと呼ばれ[redacted]や度重なる自然災害により、社会全体[redacted]ます。このような状況を打破し次の10[redacted]く、幅広い知恵を結集いたしました。[redacted]らにインタラクティブ性を高め、発表者[redacted]論をする場を設けて、次の10年へ向[redacted]を参加した全員が得られることを目指し[redacted]

  ソフトウェア開発、
  システム開発とは、
  たとえ小さくても、
  社会変革である

  本カンファレンス[redacted]に参加されるすべての方が繋がり、さらに拡がりを見せ、**社会を変革**していこうではありませんか。

今日の話
それは林が如何に失敗したかの話

- 20世紀初頭、フランスにフランシス・ピカビアという目まぐるしくスタイルを変えた前衛芸術家がいた。その人が言ったこと:
  - The world is divided into two categories: failures and unknowns.
  - 世界は二つのカテゴリに分類される:失敗と未知だ
  - 林の解釈:成功して止まったら先がない
- 先のない成功(成功への安住)よりは失敗と次への前進の方がましだ
- 失敗ほど良い教訓はない。もっとも身に染みるから
  - 転んだら良い機会だから、周りを見回し何か掴んで立ち上がる
- でも失敗はシンドイ。呆然として放心してしまうこともある
- 失敗の意味を、予め少し知っておくと、失敗からの回復が早いし、より深く学べることもある
- また、人が穴に落っこちるのを見れば、そこは避けて通れる
- そういう風にお役に立つことを期待して、ソフトウェア工学者としての林の失敗談をお話します

林の略歴

1. 最初、数学者（数理論理学）
2. 次にコンピュータ科学者
3. そしてソフトウェア工学者
   - 2の時期も形式的技法用システム開発が主で、3の時期との境界は曖昧
   - 今日、話すIT振興政策研究もしていた。それは4の最初と重なる
4. 今は人文学者。主に歴史学(思想史)と社会学
   - 人文研究が主だが歴史研究用のITツールの開発もしている

- 一番長いのは2と3のIT関連の経歴で合わせて25年余。4の時期の歴史ツール開発も入れると30年を超える

IT関連の経歴　その1

- 1981年までは数理論理学者。数学で学位を取得した後から、数理論理学の形式的技法への応用研究を開始。
  構成的プログラミングシステムPXの開発を始める。
  （筑波大数学研究科時代）
- 1988年、PXシステムに関する著書をMIT Pressより出版。PXシステムのプロジェクトがほぼ終了。
  このプロジェクトの中で形式的技法に対する疑問を抱くも、海外からの評価に甘え本質的問題に気が付かず研究を継続。
  （京大数理解析研究所時代）
- 1995年に出版された教科書「プログラム検証論」の執筆とソフトウェア技術者の教育が契機となり、形式的技法の本質的問題に気付き、プログラムの正しさを保証する技術としての形式的技法の研究を放棄することを決意。（龍谷大学理工-神戸大学工時代）
- 1998年ころから形式的技法を形式的証明の誤りを見つける技術として転用する方法を研究。証明アニメーションなどの技術を提案。
  一方でUML2.0実行可能モデラーSMARTの開発に着手。
  さらに、ソフトウェアの社会的側面に気が付く。（神戸大学工時代）

IT関連の経歴　その1

- 1981年までは数理論理学者。数学で学位を取得した後から、数理論理学の形式的技法への応用研究を開始。
  構成的プログラミングシステムPXの開発を始める。
  （筑波大数学研究科時代）
- 1988年、PXシステムに関する著書をMIT Pressより出版。PXシステムのプロジェクトがほぼ終了。
  このプロジェクトの中で形式的技法に対する疑問を抱くも、海外からの評価に甘え本質的問題に気が付かず研究を継続。
  （京大数理解析研究所時代）
- 1995年に「検証論」の執筆とソフトウェア技術者の[illegible]の本質的問題に気付き、プログラムの正[illegible]式的技法の研究を放棄することを決意。[illegible]）
- 1998年[illegible]用の誤りを見つける技術として転用する方法[illegible]ンなどの技術を提案。
  一方でUML2.0実行可能モデラーSMARTの開発に着手。
  さらに、ソフトウェアの社会的側面に気が付く。（神戸大学工時代）

もちろん、全ての技術は社会的。
しかし、その中でソフトの世界
は
ハードの世界より、
社会的側面の重要度が
より高い、という意味

IT関連の経歴　その２

- 2004年、SMARTシステムをUML2004で発表。文部科学省科学技術政策研究所NISTEP非常勤研究員就任。ソフトウェア業界振興策研究を開始。
（神戸大学工時代）
- 2005年、京都大学文学研究科教授就任。人文学者（歴史学）に転向。
NISTEPでのソフトウェア人材育成の調査・研究を本格化。
（以後、京大文学研究科時代）
- 2007年ころから、SMARTをベースに歴史研究用ツールSMART-GSの開発に着手。
このころからトップダウンのソフトウェア業界振興策の限界を認識し、NISTEPでの活動を実質停止。
- 2010年ころから２０年近いブランクを経て現役プログラマに復帰し、sourceforge.jpでのSMART-GSシステム開発のメイン・プログラマ(Javaプログラミング）となる。
2011年10月にはsourceforge.jpでの月間アクティビティのランクが９位となる。
老眼に苦しむアラカン・プログラマとしては自慢！！
- 現在もJavaプログラマとして活動しつつ、ソフトウェア工学の社会的側面の研究を続けている。

IT関連の経歴　その2

- 2004年、SMARTシステ　　表。文部科学省科学技術政策研究所NISTEP非常勤研究員　　振興策研究を開始。
（神戸大学工時代）
- 2005年、京都大学　　文学者（歴史学）に転向。
NISTEPでのソフトウェ　　究を本格化。
（以後、京大文学研究科
- 2007年ころから、SMARTをベースに歴史研究用ツールSMART-GSの開発に着手。このころからトップダウンのソフトウェア業界振興策の限界を認識し、NISTEPでの活動を実質停止。
- 2010年ころから20年近いブランクを経て現役プログラマに復帰し、sourceforge.jpでのSMART-GSシステム開発のメイン・プログラマ(Javaプログラミング）となる。
2011年10月にはsourceforge.jpでの月間アクティビティのランクが9位となる。
老眼に苦しむアラカン・プログラマとしては自慢！！
- 現在もJavaプログラマとして活動しつつ、ソフトウェア工学の社会的側面の研究を続けている。

トップダウン開発のことではありません！
政府によるトップダウンの産業振興策のこと

ソフトウェア工学における林の変化

- それはITの歴史変化を忠実に追っている

1. 形式的技法 (Formal Methods)の研究
   - **Up-front, top-downの時代**
2. UMLモデラーの開発・研究、証明アニメーションの研究
   - 一見 up-front に見えるが、実は研究の主眼は、アジャイル法、特にケント・ベックのテスト駆動開発(TDD)をUMLモデル構築、形式的証明構築に応用することだった。
   - また、その背景にあったのは、エンタープライズ系のシステムが、テクノだけでなくテクノ・ソシオの存在だという認識だった。
   - **テクノ・ソシオ・システムとアジャイルの時代**
3. IT振興策研究
   - **ソシオ、ソーシャルの時代**
     - **社会という観点が第一と意識する時代： WEB, SNS がITの花形に**

では、ITの歴史変化とは何だったか

- Up-front, top-downの時代 (1980年代中頃まで)
  - アプリ: 数値計算、制御、旧タイプDB
- テクノ・ソシオとアジャイルの時代(1990年代以後, 特にインターネット、WEBの時代)
  - アプリ: ブラウザ、メーラ、WEB DB、ケータイのシステム、B2C、B2B
- ソシオ、ソーシャルの時代
  - アプリ: SNS, ケータイ、スマートフォン、ゲーミフィケーション….

Up-front, top-downの時代

- Up-front, top-downの時代 (1980年代中頃まで。大体、FGCSの第一期が終わるころ)
  - アプリ: 数値計算、制御、旧タイプDB
  - 前提条件
    1. 人間とシステムのインタラクションは少ない。
    2. システムは一度できたら中々変わらない。
    3. システムの要求・要件は開発前に洗い出せる。
- 想定されるユーザは主にプロ(オペレータ)
- 技術・技術者中心の時代

テクノ・ソシオとアジャイルの時代

- 移行期：スパイラルの時代(1980年代中頃以後。バリー・ベームのスパイラル・モデルが1988)
  - ウォーターフォールなどと言っていたが、実はシステムが動き出して使ってみるまではわからない要求・要件がかなりある。そのために開発がうまくいかないので開発法(プロセスモデル)も修正する必要があるという認識が広く浸透。本質的には仕様のバグの問題と同じ
- テクノ・ソシオとアジャイルの時代(1990年代以後, 特にインターネット、WEBの時代)
  - 前提条件
    1. 人間とシステムのインタラクションは多い
    2. システムは出来た後もドンドン変わる
    3. システムの要求・要件を開発前に完全に洗い出せるわけがない。
  - アプリ：ブラウザ、メーラ、WEB DB、B2Cシステム

テクノ・ソシオとアジャイルの時代

技術者は見落とす
クライアントはシステム
を見ると気が変わる

- 移行期：スパ　　代中頃以後。バリー・ベームのスパ　　3)
  - ウォーターフォール　　と言われていたが、実はシステムが動き出して使ってみるまではわからない要求・要件がかなりある。そのために開発がうまくいかないので開発法(プロセスモデル)も修正する必要があるという認識が広く浸透。本質的には仕様のバグの問題と同じ
- テクノ・ソシオとアジャイルの時代(1990年代以後, 特にインターネット、WEBの時代)
  - 前提条件
    1. 人間とシステムのインタラクションは多い
    2. システムは出来た後もドンドン変わる
    3. システムの要求・要件を開発前に完全に洗い出せるわけがない。
  - アプリ：ブラウザ、メーラ、WEB DB、B2Cシステム

ソシオ、ソーシャルの時代

- ソシオ、ソーシャルの時代
  - 前提条件
    1. ITなくして社会は成り立たない。ITは社会インフラだ、社会の不可欠な一部だ。
    2. ITが社会を変える・動かす：スマート・モブズ、アラブの春、ウォールストリート占拠、反日暴動
  - アプリ： SNS, ケータイ、スマートフォン、ゲーミフィケーション….
  - もう、アプリという概念さえ古い
  - アプリからサービスへ
- ユーザは一般大衆
- ユーザ中心の時代

Up-front 時代の失敗談 1

- PXという形式的技法のシステムを開発・構築。PXによるプログラムの生成が可能となったのが、１９８４，５年ころのこと
- 最初の本格的実例として、命題論理シーケント計算系の完全性定理を PX の proof checker を使い形式的に証明
- PXは作成された形式的証明から、命題論理真偽判定兼定理証明系を自動生成した
- それは、当時としては、コンピュータが形式的証明から自動生成した最大級で、飛びぬけて本格的なプログラムであり、同時に、最初の形式的に妥当性が証明され、絶対にバグがない定理証明系(theorem prover)である
- はずだったのだが….

Up-front 時代の失敗談 2

- 大きな達成観に包まれながら、バグが絶対にないはずの、その定理証明系に、恒真な命題論理式を入力し、リターンキーを叩いた林が見たものは…
- **バグ！！？？？？…**
  - 入力した命題式のシーケント計算系での証明を表現する、LISPのS式が出力されるはずなのに、自動合成された定理証明系は、それを恒真でないと判断し「反例」を出力した
- つまり、形式的に妥当性が保証された筈のプログラムにバグがあった！！！！？？？？

Up-front 時代の失敗談 3

- 当初はPXのバグと考えたが、どうしてもバグが見つからない。様々な試みの末、PXではなく、命題シーケント計算系を定義するコードにバグを発見。つまり、「仕様」の間違いだった
- 「形式仕様のバグ」の可能性は知っていたが、そんなに簡単に起きるとは思ってなかった。しかも、やってみると定義・仕様のデバグは大変に面倒だった
- 理論的には指摘されていた「仕様のバグ」が、形式的技法にとって解決すべき現実的問題になることが、この頃から、世界各地の検証プロジェクトで実際に認識され始めていた。論文になったもので、有名なのは、Viper チップ検証プロジェクトについての Avra Cohn の報告: The Notion of Proof in Hardware Verification (1989)

Up-front 時代の失敗談 4

- 林のこの経験は、これらの経験の一つだった
  - Cohn が指摘したように、形式的技法に、テストや仕様のアニメーションなどの旧来のデバグ技法を組み合わせることにより、仕様のバグの問題、つまり、Validation の問題は大幅に低減できる
  - 林も後に「証明アニメーション」の名称で提唱することになった「形式的証明をデバッグする方法」で、この問題を乗り切った。しかし、その時には公表しなかった
- 形式的技法に本質的な現実的障害があることに衝撃をうけつつも、PXプロジェクトが海外で評価されたため、その評価に安住して、この問題を長い間無視してしまった。これは生涯最大の失敗の一つ

テクノ・ソシオとアジャイルの時代「失敗談」1

- PXの時代は、京大数理解析研究所の助手だったので研究が世界的に評価されれば、それでよかった。学生たちも研究者になりたい人が多数派
- その後、龍谷大学理工助教授に転職し、形式的技法は現実世界で実用的たりえるのか、つまり、龍谷の学生たちに教えて意味があるのか。それが重要な問となった
- そのため「プログラム検証論」という教科書の執筆に際して、形式的技法の実用例の調査を行った
- その調査の結果もあり、執筆していく内に、「形式仕様のバグ」の問題が、どんどん大きく意識され、Verification（仕様に対する正しさの検証）よりValidation(要求・要件&仕様の正しさの検証）の問題の方が現実的には大きくなるという結論に到達

テクノ・ソシオとアジャイルの時代「失敗談」2

- 胃潰瘍になるのではないかと思うほど悩んだ挙句、結局、それまでの Verification 中心の形式的技法の研究を放棄し、Validation 中心の研究にシフトすることを決意
- ちょうどそのころ神戸大工学部に転職。形式的証明やその中の定義の validation の技術「証明アニメーション」を考えて提唱。しかし、神戸大の学生も興味を示さない。そこで、そのころ日本でも流行りだしていた UML, アジャイル、要求工学の研究を始める
- その結果、学生たちとSMARTというUML2.0実行可能モデラーを開発。これは、UML モデルを要求仕様と考え、それをK. ベック流のテストケースの実行の失敗例から、Eclipse のクイックフィックスのような仕組みを使って、半自動でインタラクティブに生成していくもの
  - エアコンの赤外線リモコンの制御プログラム程度だと、リターンキーのヒットを続け、シグナルなどの名前を入力するだけで作成できた（Statechart のセマンティカルなクックフィックスを持っていたから）

テクノ・ソシオとアジャイルの時代「失敗談」3

- このころ、偶然に「マクドナルド化する社会」という社会学の本を読み、それが提起する問題と、自分がソフトウェア工学で直面していた問題との類似性に驚く
- 以後、社会学の勉強を始め、社会学における実質合理と形式合理の対立軸と、validation v.s. verification の関係が同一であることに気づくなどして、ソフトウェアを「装置」「機械」とみなす観点から「社会的存在」とみなすようになる
- この時代には開発・研究では失敗していないが、SMARTの開発が佳境に入ったころに京大文学研究科に転職し、開発が続けられず、企業や大学の研究室に無料で提供して開発を続けてもらおうと交渉するも失敗。今でも、こんなシステムは他にはないはずなのに、残念....

テクノ・ソシオとアジャイルの時代「失敗談」3

- この目的を実現するためには、UML2.0アクション・セマンティクスを表現する多値非同期並列プログラミング言語とその実行系、テスト駆動開発システムと連動する常駐デバッガ—風の実行ヒストリの保存機能、要求工学システム、など多くの機能が必要だった。それを実現したのは優秀なプログラマだった神戸大工の学生たちだが、当然ながら文学部にはそういう学生がいない
- この時代には開発・研究では失敗していないが、SMARTの開発が佳境に入ったころに京大文学研究科に転職し、開発が続けられず、企業や大学の研究室に無料で提供して開発を続けてもらおうと交渉するも失敗。今でも、こんなシステムは他にはないはずなのに、残念....

ソシオ、ソーシャルの時代の失敗談 1

- 神戸大時代は、副業で引き受けていた岩波文庫の仕事を通して、学生時代一度はプロになろうとして失敗した数学史の仕事にのめり込み始めた時代でもある
- 結果、昼はソフトウェア工学者、夜は歴史家という風になる。これが京大文への転職の理由となった
- そのころFGCSを通して知っていた黒川利明氏(当時CSK、現在KIT）の紹介で、文部科学省科学技術政策研究所(NISTEP)の客員研究員となり、IT振興策を研究し始める
- この時代に書いたのが、基調講演の概略にPDF版のURLをつけた三つのレポート

ソシオ、ソーシャルの時代の失敗談 2

- 第1レポート: Verification/Validation=形式合理性/実質合理性という見方をもとに、アジャヤイル開発とトヨタ生産方式(TPS)の類似性を指摘し、TPSが生まれた日本では、その知恵や人材をITに流れ込ませることにより、プロセス改善、ひいてはIT産業の振興を一気に行うことが可能ではないか、という提言
  - アジャイル、リーンとTPSの関係を陽に指摘したものとしては、最初の方のひとつで、当時、東京と名古屋で同じようなアイデアのもとに研究会などがたちあがりつつあった。その名古屋の方に誘われて参加
  - 最初は、形態上の同型性に注目しただけだったが、後に、歴史的にもアジャイル・リーンのルーツが、TPSなどの日本の産業力のアメリカ経営学・IEにおける80年代の研究にあることを知る

ソシオ、ソーシャルの時代の失敗談 3

- 第2レポート: 人材・知恵に言及しているものの、第1レポートは、ソフトウェア・プロセスの改善に主眼があったが、このレポートでは、プロセスを担う人の問題に視点が移った
- トヨタでは社員が「洗脳」されいるが故に、TPSが機能するように、工程を担う人たち、つまり、IT技術者のマインドを変化させないと、良いプロセスは実現できない
- そのためには、プロセスを理解するだけでなく、その背景にある「思想」まで理解して身に着けないといけない
- という内容だった

ソシオ、ソーシャルの時代の失敗談 3

- 第3レポート：第１，２レポートのような提言を実現するには、まず人材が第一である
- では、どの様な人材をどうやって、この国で育成するのか、その問題に具体的に踏み込んだのが、このレポート
- これを書くために、内外のIT技術者や経営者、先進的大学・大学院教育などを取材、自腹（実はフライト・ポイント）で Google 本社とスタンフォード大学デザイン・スクールまで取材する
- そして、その取材中に大きな挫折を経験

ソシオ、ソーシャルの時代の失敗談 4

- はこだて未来大における優れた教育システムの発案者を取材中、自分の学生たちが、彼らに教えた考え方を企業で発揮すると、それが上司に理解されず、能力を身に着けていることで却って落ち込んでしまうことがあることを思い出し（神戸大当時、学生の間で一番人気だった家電某S社での事例）、「良い人材を送り出しても理解されないということはありませんか？」と質問してみた。
- その発案者が急に悲しそうな顔になり話してくれたのが、業界でも評判のよい鉄鋼系某Sソリューションズに学生を採用してもらって学生ともども喜んでいたら、その学生のお父さんが「なんだ。Sと言っても鉄を作ってるのではないのか」と呟き、それを聞いた学生が落ち込んだ、という話
- それを聞いたとき脳裡に浮かんだある有名な画像

ソシオ、ソーシャルの時代の失敗談 5

- 太平洋戦争末期、沖縄沖に停泊する米海軍艦隊に勇猛に突っ込みながらも、猛烈な弾幕に次々と撃ち落される特攻機たち
- 幾ら人材を育成しても、それは社会の中の人材なのだから、社会がそれを受け入れないのでは、その人たちを傷つけてしまうだけだ
- 日本のITの問題を、人材を生み出す人材育成の問題だと誤解し、それを改善すればすべてが解決すると安易に思っていたが、これは人材の受け手側の問題でもあり、実は、そちらの問題の方が大きいのだ
- どのように人材育成やプロセス改善で頑張っても、受け手側が水をかけて消してしまう。これでは湿ったマキに火をつけようとする行為と変わらない
- 徒労だ…. ガーン！！！

ソシオ、ソーシャルの時代の失敗談 6

- 第3レポートは、本当は、この考えのもと、社会の問題を指摘すべきだったが、それを指摘すると大臣官房を通らないのではないかという研究員仲間の意見もあり、また、社会を変えよ、日本人のマインドを変えよ、と提言しても、NISTEPのレポートとしてはあまり意味がないこともあって、取材で得た最大の成果はほぼ隠して、第3レポートをまとめる
- レポートは、一部では評価していただけたものの、書いた本人は、これを書くことにより、かなり落ち込んだ
- これは今までの人生で最大の失敗・挫折

ソシオ、ソーシャルの時代の失敗談 7

- しかし、とは言っても進まねばならない。やるべきことはマキを乾かすこと。何十年かかっても、これをやるしかない
- NISTEPの様なトップダウンのルートではこれは難しい。本を書いたり、講演したり、講義したり、で繊毛の一本として流れを作る努力をするしかない
- と、考え第3レポート完成後、月１，２回は行っていたNISTEP通いをパッタリ止めた
- 以後、京大での講義など、ことあるごとに、この話をしているが、大勢のしかも、IT関係者の前で話すのは、そしてPPTスライドでこういう風に公に出すのは、今回が初めて

ここから　まとめ その1

- 林が経験した変化とは、ITシステム制作の仕事が、
  - 工場や鉄工所にいる人（技術者、工員）が行うような仕事から、
  - 営業所、小売店のような場所にいる人が行うような仕事に、
  - 変わってきているということ。　製造→サービス
- 簡単にいえば、ユーザとのコンタクトが増え、また、要求されるレスポンスの速度が上がっている
- また、ユーザ側も、ITシステムにより、その生活が直接に影響をうけ、システムとのコンタクトが増え、それへの対応のレスポンス速度が上がっている
  - 常にバージョンアップに対応していかないと、社会に置いていかれる

まとめ その2

- ITシステムが社会そのものとなりつつある。自動車、家電製品は、確かに社会的に重要だが、ITほどではない。今、ITが突然すべてなくなると、その自動車、家電製品はもとより、医療、金融、経済、通信、交通、軍事、治安、とにかく、殆どのものが麻痺する
- 林が客員となる前、NISTEPでITの未来像を科学技術各分野の専門家ネットワークを使いデルファイ調査したことがあった
- 結果は「2010年代初頭、ITは先端技術でなくなる」という意見が大半だった
- 当時、ITは国の重点分野だったので、NISTEPは驚き詳細な再調査を行った処、そのころには科学技術各分野でITが必要不可欠のあたりまえの重要技術となると、多くの分野の専門家が考えており、だから、それは個別の独立した「先端」技術ではなくなるという判断だとわかり、調査をしたNISTEPも納得した

つまり、ITの変化とは

IT＝電子情報に関する技術

⇓

IT＝社会に関する技術

⇓

IT≒社会を造る技術

という変化だった

IT技術者は消滅する？

- 三つの時代の内、最後の時代は、前二つと異質
- なぜか？
- 前二つはユーザ、社会の要素が大きくなって行ってはいるが、飽くまで技術中心、技術者の目で時代を見ている。
- ところが、最後のソシオ・ソーシャルの時代では、社会、ユーザが前面にでていて、技術、技術者の存在が消えてしまっている。
- まるで、グローバル化時代の「国家の消滅」のような話

しかし、どんなにユーザ中心でも システムは技術者が作る

- Android マーケットなどで、素人さん作成のシステムが沢山ダウンロードされたという話はある
- 優秀なオープン・ソースも多い。(林も作っている)
- しかし、現代の社会を支えているシステム、それがなくなると交通信号がとまり鉄道がとまり病院がとまり、明日から困ってしまうシステムの多くは、その規模からして、まだまだ、みなさんのようなプロのIT技術者集団にしか作れない
- おそらく、この状況は、当分揺るがない
- グローバル化した世界を一番下で支え、グローバル化を可能にしているのが、未だ、国家であることと同じこと
- では、技術者のプレゼンスが消えたかのようなこの時代に、IT技術者と社会の位置関係はどうなっているのか？

ソシオ・ソーシャル時代の IT技術者と社会の関係

- システムは企業、社会、などエンタープライズの一部
- すべてがITシステム中心でなされるようになってきているのだから、現代の事業所では、ITシステムは業務そのものとさえいえる
- つまり、IT技術者は、機械のようなシステムを作っているのではなく、事業所・会社、社会、そういうものの一部、場合によっては、業務の仕組みの殆ど全てを作っているといえる

イノベーションとは何か

- この様な変化に10年近く前から気が付いているのがアメリカ
- 統計機械のメーカだったIBMは、今や巨大ソリューションビジネスに変身しているが、その会長のパルミサーノを中心に、2004年に纏められたのが、有名なパルミサーノ・レポート Innovate America
- このレポートは、イノベーションを「技術を革新すること」ではなく、「技術を使って社会を革新すること」だと理解し、如何にアメリカ社会をイノベートするかが論じられている
- 林は、この見方に全面的に賛成している
- そう考えると….

IT技術者は社会技術者

- ITプロジェクト、特にSIerが行うITプロジェクトは、業務を行うシステムの開発というよりは、そのシステムが日々担う業務を造っているのだ、という視点が必要
- プロセス改善もシステム作成の工程の改善という「送り出す視点」だけでなく、そのシステムを使う事業所の人々の日々の仕事のシステムの改善を提案し作成するのだ、という視点が必要
- IT技術者は、即、社会イノベーションのための社会技術者だ！！

弾幕を賢く掻い潜ろう

- この様に考えれば、みなさんは社会イノベーションのために鹿児島県知覧の特攻隊基地を飛び立つ特攻隊員ということになる
- しかし、撃ち落されてしまっては仕方がない
- 昔から、良いクライアントがいる場所では良いシステムが作られるといわれる
- しかし、残念ながら、ITの意味を理解する良いクライアントは、この国ではまだまだ少ない（確実に増えているが）
- そうならば自分でマキを乾かしていくしかない
- 対象業務のエスノグラフィーを入念に行い、業務担当者に共感を持ち、粘り強く説得していくと、「この業務は昔から、こうなのです！その通り作って下さい！」という弾幕を張るクライアントも、理解してくれ考え方を変えるときがある

弾幕を賢く掻い潜ろう

- この様に考えれば、みなさんは社会イノベーションのために鹿児島県知覧の特攻隊基地を飛び立つ特攻隊員ということになる
- しかし、撃ち落され[illegible]ない
- 昔から、良いクライ[illegible]は良いシステムが作られるといわれる[illegible]
- しかし、残念ながら[illegible]る良いクライアントは、この国ではまた[illegible]増えているが）
- そうならば自分でマネ[illegible]しかない
- 対象業務のエスノグラフィーを入念に行い、業務担当者に共感を持ち、粘り強く説得していくと、「この業務は昔から、こうなのです！その通り作って下さい！」という弾幕を張るクライアントも、理解してくれ考え方を変えるときがある

これはわかっている方には釈迦に説法です。
ご容赦を m(_ _)m

結論：心にかけられた軛（クビキ）を外し、未来のために賢く共に闘おう

- プロジェクトがうまくいかない場合、ソフトの中だけで問題を解消しないといけないと、大変なことになることがある。林の学生の中にも、某Oソフトウェアで、そのために深刻な病になった人がいる
- 自分が、単なるシステム技術者だと思えば、与えられた要求・要件・仕様に忠実にシステムを作ることが責務になる
- しかし、自分は社会技術者だと思えば、要求・要件・仕様の見直しの提言も、というより、それこそが最重要の仕事だということになる
- もちろん、この様なことは現在のゼネコン型のIT業界や、カスタマーの状況では大変難しい
- しかし、少なくとも自分たちだけで難題をすべて抱え込む、抱え込まされるのは間違いだということを理解しておけば、潰される可能性は減るし、うまくいく場合には、地道な社会変革を成し遂げることができる
- これが林が失敗続きのソフトウェア工学者人生の中で最終的に学んだこと