随想

〔知働化研究会誌第２号〕
知働化研究会
Executable Knowledge and Texture Laboratory

知働化の最前線

知働化フォーラム 2015 講演を中心として

〔はじめにの前に〕

本作品は、2015 年 3 月 5 日に開催された「知働化フォーラム 2015」での山田正樹氏（以降「山田さん」と記載）の講演『知働化の最前線』の講演録、かつ、その解説をまとめたものです。本講演の録画ファイルを高橋雅宏氏よりご提供いただき、それに基づき講演内容を拾って記録するとともに、大槻の主観的な解説などを追記（追記部分は左罫線入りインデント表示）しています。

講演を聴講していない方にも内容が伝わるように講演で使用した山田さんのスライド情報を簡易的に掲載するとともに、口述内容をなるべく活かした形で掲載しています。編集方針は、以下の通りです。

- ✓ 山田さん「らしさ」が伝わるように、語った事項を忠実に記載しています。
- ✓ 山田さん、そして、知働化研究会として伝えたいメッセージをできる限りわかりやすくまとめています。
- ✓ 知働化研究会誌 Volume 2 の他の作品、知働化研究会メンバの研究などとの関連について解説・参照しています。

この講演『知働化の最前線』を依頼した意図は、2009 年に設立した知働化研究会も、6 年程経ったことだし、そろそろ次の 10 年くらいを見据えたネタをコンセプトリーダとしての山田さんに提供していただきたいという私（大槻）の下心に端を発しています。本質をとらえ、かつ、皆が「わからねぇ～」っと首をかしげる（否、頭をかかえる）ようなものをというのが私から山田さんへの要求でした。

従って、この講演に関するまとめ資料（つまりこの作品）は、以降何年間かの間、参照に耐えうるようなものにしたいという想いで編集しました。

知働化研究会 運営リーダ 大槻（記）

【はじめに】

メタボリックスの山田です。メタボリックス社では、インデペンデントのソフトウェア開発をやっています。

「知働化」っていう言葉からしてよくわからないかもしれませんが、「(大槻さんが)知働化最前線の話をしろ」ということでやります。あまり外では聞かない話です。「知働化」って何？って考えることが大事です。10 年後 20 年後に思いをはせて、どういう方向へ向かっているのかを考えてみることにしましょう。

【ソフトウェアは溶けていく】

- ソフトウェアは消えていく

- ソフトウェアは見えなくなる

ソフトウェアは見えなくなり、消えていくことになるでしょう。我々にとってソフトウェアははっきりしていて、大きい物かもしれませんが、一般の人から見れば、どんどん消えて、溶けていきます。

- 社会に溶けていく
  - ＊ 溶解 dissolving
  - ＊ 浸潤 infiltration

溶け方には 2 種類あって、水に塩を溶かしてしょっぱくなるような、溶解（dissolving）していくというのと、社会を侵していく（浸潤 : infiltration あるいは invasion）というのがあります。

- かたちなきかたち
  - ＊ anarchitecture アナーキテクチャ

よいイメージの溶解という観点から言うと、かたちなきかたち、アーキテクチャはなくなっていく＝anarchitecture ということに相当しています。つまり、ソフトウェアをうまく溶けていくようにしたいということです。

- ＊ big data
- ＊ deep learning
- ＊ internet of things
- ＊ “social”

悪いイメージの浸潤という観点では、big data, deep learning, internet of things, “social”（facebook などの SNS 的なもの）とか。どうしてもこういう言葉だと受け身になっていく。便利さだけが目的となっていく。スクラッチでプログラミングさせたりして、子供たちに何かやらせるといったことが多く取り組まれているけれど、それで良いのかといった感覚になる。ピアジェのような考察が無い。わからないけど使ってみてという、受け身になってしまっている。

- そこに欠けているソフトウェア「らしさ」
  - ＊ self descriptive 自己記述性
  - ＊ dialogical 対話性
  - ＊ reflective 反映性（メタ）
  - ＊ executable 実行可能性

これ等で足りないものというのは、ソフトウェア「らしさ」です。ソフトウェアは自分が何かが判っている。例えば Emacs というエディタ（ソフトウェア）だと対話的に動いているのは何か、それが何かは Emacs 自身にきけばよい。Smalltalk もそうだけど、対話ができる。それを実現するためにはメタオブジェクトが、さらには、実行可能であるということ、そういうことがソフトウェア「らしさ」なのです。

- 本当にそれでいいのか

- ソフトウェアとは何か

じゃぁ「ソフトウェアとは何か」ということを、考え直したいと思っている。便利さ以外のものがある。知働化研究会の存在意義はそこにあります。イノベーションという言葉も、その方向でよいのかを問わなくてはならない。ソフトウェアをどのように考えなくてはならないかということを今日はお話していこうと思います。

「溶ける」という表現は、とても面白い。ソフトウェアが社会、日常生活の中で見えなくなり、意識されなくなっていくことを表しています。これは、シュレーディンガーが「学習とは無意識化することである」と定義したことを思い出します。無論、「学習」は、能動的な活動ですから、この「溶ける」は、「溶解」に相当することになるでしょう。また、この「溶解」と近いのが、パラダイム論の「通常科学」のように思えます。

「浸潤」の方は、単なる「消費」と言えると思います。情報の消費とは、世の中に広く行き渡るということです。ただ、それだけです。

また、ソフトウェアの「溶解」にともない、「かたちなきかたち」＝anarchitecture になるというのも、とても意外性のある考え方です。もっと広く言うと、ソフトウェアにとって「ソ

フトウェアエンジニアリング」だって「無い」とも言えるでしょう。対象理論にとって、「メタ」理論は、対象理論ができあがってしまえば、捨て去るものかもしれません。Ludwig Wittgenstein も『論理哲学論考』の中で、「登り終えた梯子は、捨て去らねばならない」と哲学自身のことを述べています。

【世界を創る】

- 世界制作の方法

- 実行可能な / 実験可能な / 実現可能な哲学
  - Software as a Philosophy
  - Philosophy as a Software

その一つの鍵が「世界を創る」ということ。ビジネスが既成のものとしてあり、そこでそれを前提にして何かするというのではなく、発想を逆転させてみたい。世界をいっぱいつくる。しかも実行可能で、実験可能で、実現可能な世界をつくる。1980 年代、90 年代では、Artifictial Life（人工生命）というのがありましたね。あれは生命を創ってみるということです。実験的にシミュレートしてみるということ。あれと同様に世界を実行して創ってみることができるのではないでしょうか。

- 世界の知識化のさまざまな潮流
  - Programmer Anarchy (Fred George)
  - ソフトウェア開発の現象学 (Charles Tolman)
  - リアリティ・コンストラクションとしてのソフトウェア開発(Christiane Floyd)
  - 有機体論
  - コトバ・ソフトウェア

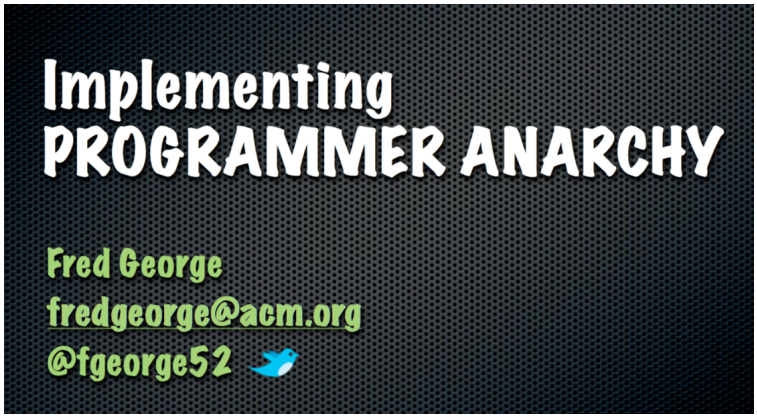
今日は抽象的な話を紹介していこうと思います。Programmer Anarchy というのは、アジャイルの次って言っている人（Fred George）、ソフトウェア開発の現象学（Charles Tolman）、リアリティ・コンストラクション（Christiane Floyd）、有機体論、コトバ・ソフトウェアなどがあります。ある意味アマチュアな人々がやっているところがミソ。今述べた人々も Floyd が大学の先生ですが、他はばりばりの開発者です。

知働化研究会誌 Vol.2 掲載作品の中にも「世界を創る」ことに触れたものがいくつかあります。

- ✓ 濱勝巳氏の『アーキテクトとは』では、「アーキテクトの仕事は世界の「法」をつくることである」と述べています。
- ✓ 私の『知平』に掲載の最後の作品『ソフトウェア参謀』では、「ソフトウェアに関する主要な仕事は、抽象機械による新たな世界を創造すること」これを行う人は「概念操作の専門家」でなくてはならないと述べています。

【Programmer Anarchy】

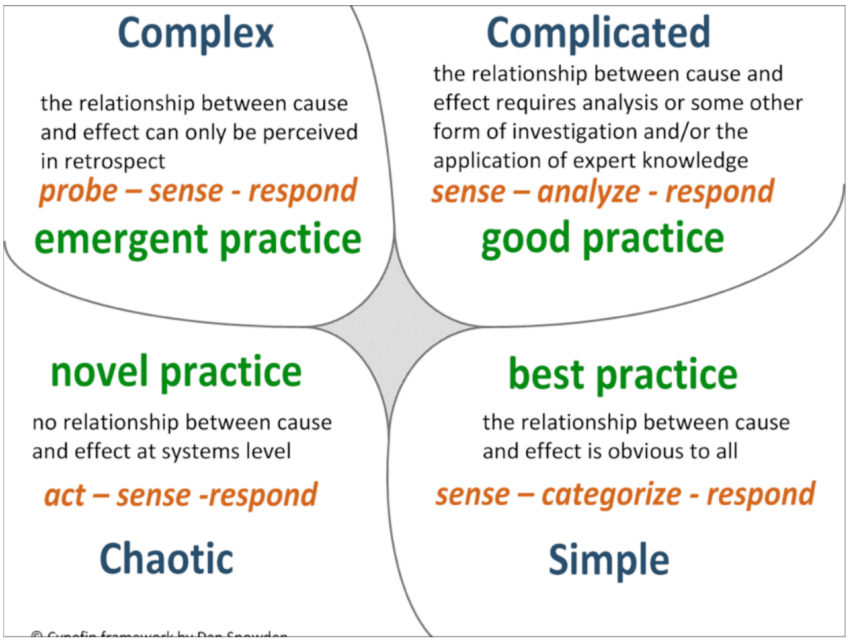
- post-agile



- 開発者は、毎日のスタンダップ・ミーティングで、自分が今日やることを決める
- PM, BA, QA はいない、顧客と開発者のみ
- 一切のルールはない、創造性と生産性がすべて
- 異議と失敗が重要
- 恐れを取り除く
- 信頼
- Agile Manifesto, XP Value は有効だけど、アジャイルのプラクティスは使わない
- ビジネス価値はすべて開発者が負う
- Cynefin Framework

ポストアジャイルの一つです。みんな勝手にいろんなこと言っているという人の一人。Kent Beck だって、smalltalk の世界ではちょっとした有名人でしたが、昔は勝手なことを言っているという一人でした。

Fred George というおじいちゃん。アジャイルなんだけど、もっとつきつめてみよう。顧客と開発者だけ。マネージャも BA（ビジネスアナリスト）も QA（品質保証）もない。ルールもない。創造性と生産性がすべて。SCRUM などがよいといわれるのは、やることがはっきりしていることだけど、Programmer Anarchy ではプラクティスもない。ビジネス価値はすべて開発者が負う。クライアントは負わない。失敗しないようにするのをやめよう。アジャイルでは、コミュニケーションをとり、テストをやり、失敗しないように保険をいっぱいかけていく方向だけど、それとは逆の考え方です。



その背景は、エンジニアリングの対象には図に示したように５つのドメインがある。

- ✓真ん中の穴（グレーのところ）は「むちゃくちゃ」なドメイン。
- ✓Complex は影響がちょっと見ただけではどのように及ぶかわからないくらい複雑なもの。地球は複雑だということ。
- ✓Complicated は因果関係を研究や専門知識によって分析しなくてはならないもの。複雑だけど依存関係はないような領域。
- ✓Chaotic は因果関係がないもの。

✓Simple はそれが自明なもの。

それぞれのドメインでどのようなやり方が役に立つかをまとめています。Programmer Anarchy が対象としているのは、絵の Complex, Complicated のところですかね。

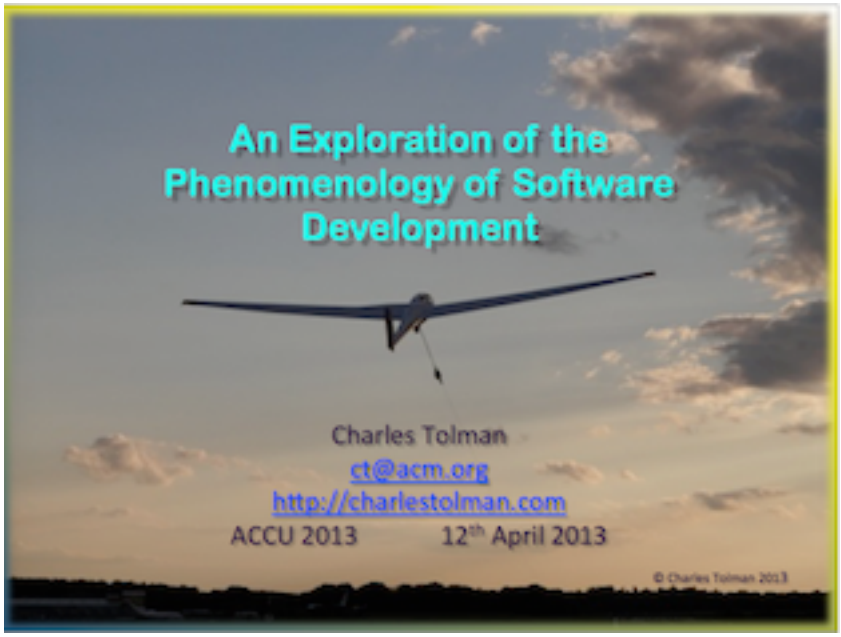
- Programmer Anarchy と整合性のある技術
  - loosely coupling
  - 500 行のサービスを 200 個
  - Entity から Event へ
  - REAL objects
  - 要件無しで作る
  - ビジネス価値に基づいてテスト

精神論はそれはそれであって、技術的な裏付けもある。ちゃんと仕事をしている。Martin Fowler（ThoughtWorks 社）のにおいがしてしまうけど、「マイクロサービスアーキテクチャ」を使っている。500 行のサービスを 200 個。エンティティではなくてイベントが中心。データ中心だとタイトなカップリングになってしまう。Smalltalk で言う本来のオブジェクトのような世界。そういうものがネットワークの上に広がっているという REAL Objects の世界です。

> データ中心かイベント中心かというのは、古来よく議論されてきた事項です。組込み系では「振る舞い」、つまり、イベントの系列が重要ですし、制御系でもプロセス（タスク）が並行に稼働する世界です。私が好みの JSD(Jackson Structured Development)でも、「世界認識の起点はイベントである」と述べられています。
>
> ちなみに、500 行×200 個＝100K 行ですから、そこそこの大きさのものがアナーキーに作ることができるというのは、すご

> いことです。最近、こういった実践的な話をきいていて思うのは、マイクロサービスという従来の部品より小さいものが、上手く機能する理論、適切な抽象度を設定するような指標や基準（従来の工業化の枠ではなく思考の技法として）のようなものがあるような気がしています。

【ソフトウェア開発の現象学】



Charles Tolman という人が唱えています。

- ソフトウェアを作るとは
  - ＊　考え続けることであり
  - ＊　メンタル・モデルを作ることであり
  - ＊　それをソフトウェアにトランスコードすること
- 経済的な側面が強調されすぎているけど、本来それは遊戯なのだ
- よいプログラマはそれに気付いている
- コンピュータとは「思考の鏡」なのだ

もうちょっとややこしい俗流哲学みたいなことになりますが、「ソフトウェアをつくるというのは、考え続けるということ」という話です。一人一人の中に、そのソフトウェアが動く世界は何っていうメンタルモデルをつくっていくと、結果として

ソフトウェア（最終的なコード）ができちゃう。本来ソフトウェアづくりは経済ではなく、遊戯（play）なのだ。まぁそうだよね。

彼の話が面白いのは、彼がブログとかで引用している論文が面白いのでここで紹介して、話を広げてみようと思います。

《Peter Naur》

- 理論構築としてのソフトウェア開発
  - ＊ Peter Naur, 1985
- http://www.dc.uba.ar/materias/plp/cursos/material/programmingAsTheoryBuilding
- 世界のある事象がプログラムによって、どうやって取り扱われるべきか
- 世界のある事象の見方が変わったときに、プログラムをどのように構成的に変えていけばよいか

Naur は、BNF(Backus Naur Form)記法とかで有名な人です。「理論構築としてのソフトウェア開発」。世界のある事象（要求という意味ではなく）がプログラムによって、どうやって取り扱われるべきか。プログラムを部分から全体へ積み上げて作っていく構成的（constructive）な方法を示しています。

僕らが世界に持っているイメージ、あるいは、理論と、コードとの間の関係を説明したもの。

《Boris Wyssusek》

- Peter Naur の「理論構築としてのプログラミング」という考え方を哲学的に再評価する
  - ＊ Boris Wyssusek, 2007
- http://is2.lse.ac.uk/asp/aspecis/20070190.pdf

- プログラミングの本質はコード・ドキュメント・仕様などの成果物ではなく、構築される理論にある

Peter Naur の理論構築としてのプログラミングという考え方を哲学的に再評価する。中身はどちらかと言うと、デザインサイエンスに近い。

《Duane Truex》

- 方法論無しのソフトウェア開発
  - ＊ Duane Truex, et.al, 2000
- http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.2819&rep=rep1&type=pdf

方法論無しのソフトウェア開発。方法論いらないよね。だって我々生きている時にそんなもん使ってないじゃん。

《Christiane Floyd》

- リアリティ・コンストラクションとしてのソフトウェア開発
  - ＊ Christiane Floyd, 1992
- http://swt-www.informatik.uni-hamburg.de/uploads/media/SD_AS_RC.pdf

リアリティ・コンストラクション＝現実感を構成していく過程としてのソフトウェア開発。世界制作の方法。リアリティ・コンストラクションというのはラジカル構成主義という言葉を使う人たち（Ernst von Glasersfeld）の用語。精神医学とか教育・心理学方面の言葉。

ラジカル構成主義

- 知覚は感覚やコミュニケーションを経由して受動的に受け取られるものではない
- 知識とは認知主体によって能動的に構築される
- 認知の機能は、生物学的な意味で適応的なものであり、適合や実行可能性への傾向性を有している

- 認知は主体による経験世界の組織化の役目を果たすであって、客観的な存在論的実在を発見してるわけではない

サイバネティクスや、ピアジェの世界にいく。西垣先生（知働化フォーラム 2015 の午後の講演者）はラジカル構成主義の本 “The radical constructivist view of science”の監訳をされているので、つながりがあると思う。

これは何かというと、ラジカル構成主義とは、知るということは感覚によって受動的に受け取られるものではなく、知識というのは能動的に構築される。認知の機能は適応的なもの。やってみてだめだったら変えるというのが、認知の基本。

ビジネスがあって要求がでてくるのではなく、要求を書くことによってビジネスを創っていくといった考え方に似ている。

- 今までの知働化のキーコンセプトは「実行可能な知識」
  - ＊ でも「知識」という言葉は「いやらしい」
  - ＊ 知識とは実際は
    - ・ 知るという働き
      （仏教用語としての識）
    - ・ 知るに際して有用な(有害な)道具であるコトバ

  として考えた方がよい

知働化との関連では、「実行可能知識」というと、知識という言葉はやらしいので、実際には、知るという働き、「識」というのは働き。知るに際して有用なコトバに分けて考えている。

> 知働化研究会の中で、「仏教」特に「原始仏教」の考え方がソフトウェアづくりに合っていて腑に落ちるところが多いという意見は多いようです。私の『知平』の中の小論『仏教からのヒント』でもこのあたりについて軽く記しています。
>
> 濱勝巳氏の『アーキテクトとは』の「法」の話もそうですし、以前、彼は「ソフトウェアは空（くう）である」ということも言っています。

山田さんも仏教に造詣が深く、今回の講演のような話をされたのだと思います。

仏教の「唯識」「縁起」「法」「行（ぎょう）」「律」などの諸概念は、ソフトウェアの世界を探る際にとても参考になります。

- 製造としてのソフトウェア開発には限界がある

デザインとは、以下のような異なる世界をリンクさせること

- 問題となっている社会的な世界（ビジネスとか）
- 実装の手段としての技術的な世界
- 手法と概念の形式的な世界

デザインとはこんがらがった現実領域を解きほぐすこと

- 製造としてのソフトウェア開発の線形的な方法では無理
- 要求を分析せずに、我々が我々のパースペクティブから構成する
- あらかじめ定義された方法論を使うのではなく、その場で情況に応じて創り出す
- 固定された実装手法を参照するのではなく、既にあるものの中からその場でテスト、選択、補完によって構成する

デザインとは、判断の網の目である

デザインには対話がなければならない

- ひととひとの間
- ひととソフトウェアの間
- ひととセカイの間

デザインとは、こんがらがった現実を解きほぐすこと、製造のようなリニアなアプローチではだめでしょう。要求は分析しない。我々のパースペクティブから構成する。方法論を使うのではなく、創り出す。現実を創り出すという意味でのソフトウェア開発。

デザインでは対話が必要でしょう。

> このあたりの話は、「デザイン論」の展開になっています。私の『知平』の小論『新ソフトウェア宣言その後』の初段に「ソフトウェアは美しい人工物（artifact）である」ということを述べており、その背景には、山田さんの言う「リンク」「編み目」「対話」といった事項と同様の問題意識があると考えています。

《A.N.Whitehead》

- メレオロジ
- 有機体論
- 出来事

Bertrand Russell と共著の『プリンキピア・マテマティカ』で有名な Whitehead のメレオロジという考え方。今の数学は所属する「∋」という関係をベースにしている。このアプローチは数学の世界を構成していくのに成功を収めました。

一方、メレオロジというのは、全体と部分という関係「⊃」だけで数学を構成する。オントロジとかでもでてくる。何か要素を集めると、階段を上がってしまう。概念の世界はもっとフラット。出来事の計算というのを企てた人に Nelson Goodman という人がいる（でも、最終的にはこの再構成の企てはあきらめました）。

前にも言いましたがエンティティではだめでイベントが主体。こういったことを

Whitehead はめんどくさい文体で書いています。

パラレルでコンカレントな世界は、エンティティがあるとボトルネックになってしまう。

> 数学の世界の話なので一見敷居が高くみえますが、メレオロジというのはソフトウェアエンジニアリングではよくでてきます。例えば、UML にもなっている D.Harel の提唱した statechart では、状態を表す○と、遷移を表す→とを自在に繋いでよくて、状態の中に状態を含んで（コ）よいことになっています。こういうグラフをハイグラフと言いますが、これはメレオロジの考え方に従ったものです。

《Joseph Goguen》

ばりばりのコンピュータサイエンティストです。2006 年に亡くなりました。代数的仕様記述言語 OBJ とか、形式手法（institution という圏論・計算機科学の概念提唱）の大御所です。。その一方で認知言語学の仕事もしている。メンタルスペースやブレンディングのような話もしている。

世界の人はどんなことを考えているのかを紹介してきました。さて、最後に僕がどういうことを考えているのか、僕が彼等につけ加えて考えていることをお話します。

【コトバ・ソフトウェア】

- ソフトウェアはコトバで出来ている
- ソフトウェアはコトバという材料（だけ）を使って，人工物を作る唯一のやり方（エンジニアリング）である
- コトバとは，何らかの対象を，その対象とは直接関係のない別の対象をもって，表出（記述や発話，コーディングなど）する試みである
- コトバには，コトバを表出するものの他に，コトバを解釈するものが必ず存在する（あるいはその存在が想定される．少なくとも表出した ものは潜在的に解釈するものでもある）
- 解釈とは，解釈した結果，解釈したものの性質が変化してしまったり，解釈したものが外界に何らかの作用を及ぼしてしまったりすることである
- 蒸気機関が単にお湯を沸かすだけのものではなく，お湯を沸かすことによって「仕事」をすることで，エンジニアリングになり得たように，ソフトウェアはコトバが解釈を引き起こし，解釈が現実世界を変化させることでエンジニアリングとなり得る
- ソフトウェアはあるコトバを別のコトバに連続的に変換していく（エ ンジンがエネルギーを別のエネルギーに変換していくように）．そして最終的には CPU と呼ばれるものが解釈可能なコトバに変換する．CPU は，あるコトバを解釈することによって，現実世界にある変化を 引き起こす仕組みを持っている

ソフトウェアはコトバでできている。ソフトウェアが人工物をつくるエンジニアリングだとしたら、その材料はコトバしかない。ソフトウェアエンジニアリングがあるとしたらコトバを使うこと。そのことがないがしろにされているように思います。

> Michael Jackson は、「ソフトウェアエンジニアリングとは記述の学問である」と述べています。ソフトウェアづくりでは、プログラミング言語、仕様記述言語をはじめ複数の言語の記述

を扱います。無論、無意識や語り得ないものもありますが、表現として表れ、考えを進め、コミュニケーションを図り、世界を変えていくところに本領・本質があります。

言語というのは、社会の中でダイナミックに変化し、新たな概念を創り出していくところに力が宿っています。これを山田さんはあえてカタカナで「コトバ」と表したのだと思います。

- 認知言語学的なコトバ
- から
- 生成文法的なコトバ
- へ

- thing, entity, relation, process
- イメージ・スキーマ
- カテゴリ化
- 喩え

今までのコトバ

- コトバを固定化する
- コトバを明確に区別する
- コトバの算出方法を制限する
- コトバの解釈方法を制限する
- コトバから個問えばへの変換方法を制限する
- 「コミュニケーション」による意味伝達を意図する

コトバが死ぬ

うまく行かない

コトバとは、何らかの対象を、別の対象を使って表すこと。コトバには解釈するものが存在する。計算というのも解釈の一つ。その解釈が現実世界に作用し、変化させるので仕事をしており、エンジニアリングになり得る。認知言語的（頭の中の世界認識）なコトバから、生成文法的（プログラム）なコトバへ。

> 「認知言語」と「生成文法」というのは、歴史的には先に「生成文法」が Noam Chomsky によって 1955 年に提唱され人間が生得的に言語能力を普遍文法という形で持っており、これが生後の学習によって個別文法の習得に至るという理論。いろいろ批判はあるものの、形式言語やプログラミング言語のクラスを定義することができ、計算機科学やコンパイラ技術の基礎になっています。
>
> 「認知言語」は、George P. Lakoff が 1985 年頃から提唱した人間の概念・身体知能に備わっていることがらを起点として、メタファ、イメージスキーマなどの概念操作を用いて言語実態を解明していくアプローチです。
>
> 山田さんは、人間の考えを認知言語的にコトバとしてとらえ、それを生成文法的な計算の世界のコトバに対応させていくといったことを言っていたのではないかと思われます。
>
> 本橋正成氏がこのあたりの対応関係について、講演時に質問をしていました。

認知言語学では、Thing,Entity,Relation,Process がベース。上下関係、移動といった身体的なイメージスキーマ、カテゴリ（概念）、喩え（メタファ）。

今までのソフトウェアエンジニアリングでは、コトバは、固定化、明確化、制限といったコトバが死ぬ方向のことをやっている。

- いきいきと生きるコトバ

- イメージ・スキーマに相当するコトバを作る
- コトバがコトバを生み出すような仕組みを作る

- コトバの体系（言語）を作る
- コトバを媒介とするカップリングを導く課程

- ソフトウェアにおけるアスペルガー

生きるコトバは、イメージスキーマに相当するコトバを作る。コトバの体系を作る。例えば、「まく」。コトバがコトバを生み出すような仕組みが必要。コトバを媒介とするカップリングを導く。

> 「アスペルガー」のスライドは講演時に説明がなかったのですが、ここのところ日本でもアスペルガー症候群の人の優れた集中力を活用する話もでています。偉大な数学者である Alexander Grothendieck もアスペルガーであったと言われており、ソフトウェアの世界でもこのような能力が必要なのではないかという話をされたかったのかもしれません。

【おわりに】

- object, modeling, agile(1994～)
- 実行可能な知識としてのソフトウェア(2004～)
- 知働化研究会（2009～）
- 識とコトバ（2014～）
  - real objects
  - M.E.T.A
  - まく
  - dissolving architecture, anarchitecture

１時間にわたって夢のような話をしました。僕ら、否、僕の話をしました。今は、「識とコトバ」についていろいろ活動しています。メソドロジーでも何でもない、日常、こんな活動をしているということをお話しました。この先、どうなるかはわかりませんが、また５年後くらいに何か変なお話ができるとよいと思っています。

〔おわりにの後に〕

実は、この講演録をまとめたのは、2015 年 10 月のことです。つまり、山田さんの知働化フォーラム 2015 での講演から半年以上経っています。この間、山田さんの講演をかみしめながら、ソフトウェアについて考え、いろいろな人と議論をしました。山田さんの話は、今までのソフトウェアの世界観を覆す新たなパラダイム提唱です。一言で表現するならば、**「ソフトウェアの（識とコトバによる）認知的転回」**とでも呼ぶことができるでしょう。私の周囲で、ソフトウェアのことをよく考えている人は、表現の差はありますが同様の考えに至っているように思えます。

こうして次の 10 年を見据えた話をうかがうと、「アジャイルプロセスとは何か」といった事項も自然と明らかになってきます。山田さんが講演の中で述べていますが、従来のソフトウェアエンジニアリングの延長線上で、俊敏で・軽量な、チーム・ステークホルダー間のコミュニケーションと確認によって、リスクを減らす（失敗しないようにする）ものと言えるでしょう。

一方で山田さんの講演は、「創造」に焦点をあて、人の頭の中、思考や認知の世界に踏み込んだアプローチになっています。**「創造の具現化は、概念・コトバを創ること」**なのです。

2010 年にまとめた『新ソフトウェア宣言』では哲学的な表現にとどまりましたが、具体的な活動の方向も見えてきたように思えます。山田さんの講演中のキーワードを 7 つの宣言に対応するものを並べてみましょう。

月：ソフトウェアは、美しい人工物である　　現実の人工物の創出

火：ソフトウェアは、分解不能な全体である　　Complex における失敗のススメ

水：ソフトウェアは、学びの副産物に過ぎない　溶解、anarchitecture

木：ソフトウェアは、実行可能な知識である　　識とコトバ

金：ソフトウェアは、富を生む資源である　　遊戯、新しい価値観と世界を創る

土：ソフトウェアは、数学理論探求の上に成り立つ　メレオロジ

日：ソフトウェアは、言語ゲームである　　認知科学

はっきりと未来をつくる照準が定まってくるような気がします。「知働化」が「働き」であり、活動である証左ですね。

今回の講演『知働化最前線 2015』が、10 年、20 年後に「知働化研究にとって歴史上転機になったね」と言われるようになることを祈りつつ、筆を置くことにしたいと思います。

2015 年秋
大槻 繁

■