アジャイルより大きいなにか

YAMADA Masaki, masaki@metabolics.co.jp

2016.01.23

今日は次の三つのことについてお話ししたいと思います.

まず,「アジャイル」とは何で **あった** のか, について.

そして,「アジャイルより大きい何か」について.

最後に, 我々はこれからどこに行くか, についてです.

1

「アジャイルなソフトウェア開発」, あるいはその出発点である「軽量手法」(lightweight methodology) や eXtreme Programming (XP) が広く知られるようになってから, 既に長い年月が過ぎました.

伝統的なソフトウェア・エンジニアリングの視点からは, 未だに奇異な目で見られがちな「アジャイル」ではありますが,「アジャイル」は既に伝統的なソフトウェア・エンジニアリングがやろうとしてけっきょくできなかったことを成し遂げ, あるいは成し遂げつつあります.

「伝統的なソフトウェア・エンジニアリングがやろうとしてできなかったこと」とは何でしょう?

それは一言で「ソフトウェアを工業化すること」と言っていいのではないかと思います.

ちゃんと「アジャイル」に実行されているソフトウェア開発の現場をご存知の方ならば分かるでしょうが,「アジャイルな」ソフトウェア開発はかなりの程度, (ソフトウェア的な意味で) 工業化されています.

例えば, ユーザ・ストーリから実際に動くコードが配備されるに至るまでのパイプラインがあります. もちろんハードウェアの生産とは異なる, ソフトウェアの開発である, という前提のもとにですが, それは極めて標準化され, 効率よく, 実効性のあるプロセスを提供しています.

そのようなパイプラインは, 謂わば, ハードウェアの生産におけるベルト・コンベヤのようなものです. そこには

- ユーザ・ストーリを生み出す手法とプロセスがあり,
- ユーザ・ストーリを一定の品質を保ちつつ, 動くコードに変換するよく知られた手法とプロセスがあり,
- 動くコードを配備, 運用, 保守しながら, 進化させていく手法とプロセスがあり,
- しかもそれらの基本的な部分は, Scrum あるいは Lean というような形で標準化され,
- さらにそれらが顧客に対して価値を生む可能性を追求しうるような仕組みを提供し

ています.

伝統的なソフトウェア・エンジニアリングを採用しているプロジェクトのいくつかは, 一見「工業的」であるかのように見えるかもしれませんが, 単に「手続き的」「管理的」であるのに過ぎないことが多いのです.

Scrum や Lean についてある程度理解/体感していれば, プロジェクトやソフトウェア組織を渡り歩いても, 最小限の部分では知識や体験を共有しうるので, それほど困ることもないでしょう.

伝統的なソフトウェア・エンジニアリングでは, ごく一部の先進的で, 資金と能力に恵まれたソフトウェア組織でなければできなかったことが, 多くの, 多様なソフトウェア組織で可能となっています.

もちろん, その背景にはソフトウェア/コンピュータ技術の進歩があることは間違いありません. しかし, ソフトウェア技術がそのように進歩したことと, アジャイルなプロセスが広まったことの間には一方向の因果関係というよりは, 共進化の関係があると思われます.

ある意味では, 皮肉なことですが, 「アジャイルは成功したウォータフォール」ということもできるかも知れません.

そして「アジャイル」がソフトウェアの, ソフトウェアとしての工業化に貢献したことはもちろん悪いことではありません.

しかし, それが, それだけが, 我々の望んでいたことだったのでしょうか?

2

「アジャイル」を生み出し, 作り上げてきた多くの (有名無名に拘わらず) 人たちは, 単に「工業化」や「生産性」や「ユーザ価値」を追求してきたのではなく, その背景に何らかの

理想, ソフトウェアとはどういうものであるべきかという思想, 世界とソフトウェアはどう関わるか, 自分はなぜソフトウェアを作っているかという哲学を持って生きてきたのではないかと思います.

しかしそれらの背景は, 「アジャイル」が成功するにつれ, 希薄になってきていきます.

そもそもなぜ「アジャイル」なのかが, 常に問い直され続けなければ, 「アジャイル」は第二のウォータフォールになり果てるしかないでしょう.

我々も, ソフトウェア開発の現場で具体的な活動や実践を行う一方で, それらの背景となる「考え方の基礎」を探求してきました. そのようなものがなければ, どんな実践もいずれは痩せ細った単なる慣習となってしまうと考えるからです.

山田個人で言えば, それを「実行可能な知識」(Executable Knowledge) という形で, まとめて来ました. その一部は ITmedia (当時は@IT) の連載 「実行可能な知識とソフトウェア」 (2004 ~ 2005), 「ソフトウェア開発をちゃんと考える」 (2005 ~ 2007) に掲載されています.

それらはいくつかの原則 (principle) から成る, ソフトウェアについてのメンタル・モデルのネットワークです.

1. ソフトウェアは実行可能な知識である
2. 機能はただである
3. ソフトウェアは言語を材料とする人工物である
4. ソフトウェアを作る過程は言語の創発と変換の過程である
5. すべてはメタファである
6. プロセス (アクティビティ) がシステムの構成要素であり, ソフトウェア自体は排泄物に過ぎない
7. ソフトウェア・アーキテクチャはなめらかでなければならない (アナーキテクチャ)
8. ソフトウェアは織り紡ぐものであり, 重要なのはテクスチャである
9. 作ることと使うことは一つの動的平衡過程の二つの側面である
10. 事態 (である, がある) より, 出来事 (が起きた (らしい, ことを知っている)) が本質である
11. 世界は集合 - 要素関係ではなく, 全体 - 部分関係からできている

これら原則の一つ一つに正誤はありません. これらのネットワーク全体がぼんやりした心像を作り上げます. 謂わばひとつのパラダイムです. ソフトウェアに関わるあらゆる活動の実践は, これらの心像から導き出されますが, それは個々のプロジェクトによって異なり得ます.

我々の日々ソフトウェア開発の現場での実践は, これらの心像から生み出された一つの, その

時点でのインスタンスです.「アジャイル」のいくつかのプラクティスや考え方とは多くの部分で重なりますが, 教条化された「アジャイル」とはほとんど関係ありません.

これが「アジャイルより大きいなにか」です.

これらの原則は10年ほどの時間を経て, だいぶ成熟してきましたが, 今でも少しずつ更新されて = 揺らいでいます. そして, 2009年頃からは, より多くの人の, より多くの考えが加わり,「知働化」という名前が付けられ, 拡がっています.

「工業化としてのアジャイル」を包含する,「知働化としてのアジャイル」と呼んでもいいかもしれません.

3

では, これから我々はどこに向かうのでしょうか.

厚顔無恥であることは承知の上で, 敢えて言ってみるとすれば...

「ソフトウェア」はだんだん消えていくでしょう. なくなってしまうのではなく, 社会という溶媒の中に溶け込んでいくでしょう. 溶解するソフトウェアです. そして, 見えなくなるかもしれません. 目の前にあったとしても.

社会は「ひと」のネットワークです. ひととひとの間をコミュニケーションがつなぐネットワークです. コミュニケーションと言っても, 特に意味のある何かを伝えることが重要なわけではありません. コミュニケーション ~「つぶやき (tweet)」と呼び替えてもいいくらいなのですが, 次のコミュニケーションを生み出すことだけが重要なのです (コミュニケーションの媒体であることを除けば,「ひと」でさえ, たいして重要ではありません).

実はこのネットワークに参加しているのは「ひと」だけではありません.「もの」(自然物, 人工物) もこのネットワークに参加しています.「もの」と「ひと」はどうやってコミュニケートしているかって?「もの」も「ひと」に向かって, さまざまなメッセージをさまざまな形で発していますよね? 椅子は「ここに座ってみて」と話しかけています [1]. 道は「ここに沿って歩いていいよ」と言っています. それに対して,「ひと」はそれらのメッセージに応じた (あるいは応じない) 行為を取ることで,「もの」たちと対話をしているのです.

このネットワークの参加者のうち,「ひと」をアクタと呼ぶことがあります.「ひと」と「もの」を合わせたネットワークの参加者を「アクタント」と呼ぶことがあります.

実は最近, このネットワークにはまた別の種類の参加者がいることが分かってきました. それを何と呼ぶかは, まだよく分かっていませんが, 私は「こびと」とか「ちいさいおぢさん」と

呼ぶことがあります.「ひと」,「もの」,「ちいさいおぢさん」たちの間のコミュニケーションが織り上げるネットワークが社会, 我々が認知する世界です [2].

我々の仕事は, 多分, この「ちいさいおぢさん」と「ひと」,「もの」の間のコミュニケーションをデザインし, それに基づいて「ちいさいおぢさん」を生み出し, 彼らをネットワークの中に密かに潜り込ませることなのです.

その時, 我々にとって重要な概念は何でしょう? 何をどこから学べばいいでしょうか? 私は, 少なくとも個人的には, 次のようなことについて学ぼうとしています.

- ダイアログ
- メタ (という名の, ある特殊な構造的カップリング)
- メレオロジ
- アスペルガー症候群

実行可能知識が, 認知言語学, 構成主義, 学習理論, 有機体論などから学んだように, あるいはその延長上に, まだ学ぶべきことがあるようです.

もちろん学ぶだけではありません. 我々はそのようなソフトウェア = ネットワーク中のちいさいおぢさんをどうやってデザインし, 生み出せばよいか, 技術的な裏づけも必要です.

小さな「サーバ」を, RaspberryPi でも, Arduino でも, AWS EC2 インスタンスでも, node でも「ひとり」立ち上げてみましょうか. 彼らとは, どんなことばを使って, 何をどうやって話せばいいでしょうか.「ふたり」立ち上げるとどうなるか.「ひと」や「もの」が介入するときにはどうなるか. よい対話とは何か. 途切れのない対話はどうして生まれるか. ネットワークが「社会」になったとき, 階層や抑圧や協調は発生するか.「知識」はどこに発生するか.

「本来の」オブジェクトの考え方 (Smalltalk など) はどこまで使えるでしょうか.「関数型」とは実際には何を意味しているのでしょうか. バズワード “Micro-service Architecture” に似ているでしょうか. “IoT” ? 本質は何で, どこが違うのか.

これは外から見れば一種のゲーム (SimCity:-) ですが, 我々はその中に投げ込まれていて, その中で生きていることを忘れることはできないのですが.

---

1. いわゆるアフォーダンスと呼ばれる考え方です. ↩

2. 実はこのような考え方をする哲学の分野があり, “Actor-Network Theory” とか “オブジェクト指向存在論” と呼ばれています. もっとも, それらの論文を読んでもあまり得るところはないでしょう:-) かつて, ソフトウェアはさまざまな哲学や思想から影響を

受けてきましたが, 今やソフトウェアが哲学や思想に影響を与える時代に成りつつあるのかもしれません. ↩