ソフトウェア工学と人文･社会科学

京都大学文学研究科
情報･史料学専修
林晋

この講演の目的
-本ワークショップのプログラムから-

1. プログラム検証、Agile 法、UML、要求工学、ソフトウェア産業構造、などを例にとり、ソフトウェア工学の問題が、価値、チーム内コミュニケーションなどの問題と深く関連していることを、M. ヴェーバー社会学の合理性理論、生産工学、経営学等の考え方を援用しつつ指摘する。
2. また、それらの問題へのアタックを容易にするためのツールとして構想されたシステム SMARTの現況と問題点を解説し、
3. さらに、SMART のようなツールを、歴史学などの入り組んだ推論を必要とする人文学に応用する可能性について現実の数学史研究を例に論じる。

の予定だったが…

- 昨日いただいた吉田先生の講演資料に刺激されて、1に話題を絞り、そういう研究の背景･動機を説明することに予定変更！
- まだ、試論。是非、是非、ご批判を！！！
- 2,3 を聞きたかった方、すみません。m(_ _)m
  - 2, 3 は、この集会も、その一環の科研費特定領域の報告会 (open、1月24日、学術総合センター) などで…。報告会のスライドは、林のサイトに置きます。

社会と情報化を考えるということ

- 現代社会のある特徴づけ、「高度に情報化されたグローバルな市場経済社会」。
- 市場経済、情報化、グローバル化、を前提にして議論を進めることもできる。しかし、それでは、現実は捉えがたい。
- WHY？:現代社会は、reflective な変化が凄まじい。「前提」が、その「帰結」により簡単に変化してしまう。現代の「資本主義」は、「生き残り」をかけて、そういう柔構造になってしまっている。
- だから、「市場経済、情報化、グローバル化」という「前提」が常に変化している。それを捉えるには、その「原因」、「理由」を問わない限り、その場、その場限りの、あるいは、短命な考察に終わってしまう。

よくある「情報化論・情報社会論」

- しかし、現在、我々が目にする「情報化論・情報社会論」の多くは、「ケータイができました。インターネットができました。だから、社会は、こう変化したのです。こう変化するでしょう」という議論になっている。
- つまり、「ケータイ、インターネット、メール」というような、技術を、数学の公理のように「大前提」としておいて議論している。
- しかし、その「前提」が、どうして、どのように、生じてきたのかの議論は、ほとんどない。つまり、次の図式

社会の一部としての情報技術者、情報科学者

- その情報技術は、神様や悪魔が作っているのではない。我々、情報技術者や情報科学者が作っている。
- そして、我々は、社会の一部であり、社会・文化に大きな影響を受けている。
- つまり、情報技術を作っているのは、実は（当たり前ですが）社会！
- そして、情報技術者の持つ「文化」が、その生産物の性格に大きな影響を与える。また、逆に、その生産物が、その作者・使用者に大きな影響を与える。
  - UNIX**文化**, The UNIX Philosophy ←**本の題名**。

吉田CMC空間論

- だから、先ほどの図の逆も考えなくてはいけない。
- 吉田CMC空間論では？
- 吉田先生のスライドから：
  - **モダニティの徹底化** (①+②+③)＝〈**情報化**〉
    - **情報通信技術が社会の本質的な構成要素として組み込まれ再構築されていくプロセス**
    - **生活世界への浸透 ⇒ CMC空間の出現**
- 「モダニティの徹底化 ⇒ CMC空間の出現」、つまり、吉田CMC空間論には、逆の図式が組み込まれている

Max Weber「プロテスタントの倫理と資本主義の精神」

- 注意：これは、単なる起源論ではない、その driving force である modernity が、情報化により、さらに強化されるという reflection の理論でもある。
- これと、同じように、現代の「市場経済社会」の原理、あるいは、その「先祖」ともいうべき「資本主義」について、およそ100年前に、「どうして、どのように、生じ、どのように維持されているか」を議論をしたのが社会学者の M. Weberであり、有名な「プロ倫」と呼ばれている著作と、それを契機とする Weber の膨大な研究。その発生の図式は、

必要な問い

- 情報技術における、ウェーバーのケースの「プロテスタントの精神」のようなものは何なのか？
- これを意識していないと、未来予測も現実認識も、10年も持たないものになる。現実は、次のような「形」をしているのだから：

- この何かを同定し、それを探求しないと reflective な現代、急速に「情報システム化」する現代社会、そして、その一部として、その社会の現代化という流れに影響をうけつつ変化を続ける情報技術それを drive している情報技術を理解できない。
- たとえば、これを無視した情報科学・技術政策や、情報化投資は大きな誤りを犯すだろう。

「何か」とは？

- 吉田CMC空間論のように、この「何か」を reflective modernity として説明することもできる。再び引用：
  - モダニティの徹底化 (①+②+③)＝〈情報化〉
    - 情報通信技術が社会の本質的な構成要素として組み込まれ再構築されていくプロセス
- 林も、ほぼ同意見。
- しかし、もう一歩踏み込むこともできるのではないか。踏み込んだ方がよいのではないか？ Weber のように。
- 林は、社会学の素人ながら、情報学の専門家という点を生かして、これをやりたと思って、研究を始めている。
  - たとえば、これを通して、日本のソフトウェア工学の弱さの「原因」を解明したい。

その試みの一つ：
B. フランクリンとしての Software engineering

- Weber がフランクリンという個人に、資本主義の具現化をみて、それを通して資本主義（の理想型）を考察したように、「何か」としての合理性や「近代性」の特質を、Soft. Eng. を通して、理解できるのではないか？
- また、それにより、文化的産物としての Soft. Eng. の理解も進む！
- そして、それはさらに、Goguen 講演、黒川講演にあったような、ソフトウェア工学のあり方、ソフトウェア工学の未来予測に、大きな示唆を与えるはずだ。
- すでに、その視点から、いくつか技術政策的な発言をしている（林・黒川 NISTEP論文 etc.）

Software 開発法の歴史

静的、計画経済的、理論的

動的、市場経済的、経験的

- Waterfall
- Iterative developments
- Prototyping
- Agile

Agile の特徴

- ツールに重きを置かない
  - 「要求仕様」はカードに書け
  - ツールはコンパイラだけ
- 個人プレーよりチームプレー
- 早く決めるな。決定は先延ばしにしろ
- 顧客との連携を大事にせよ：「カスタマーは神だ」
- いわば、「ルールより人。形式知より暗黙知。契約よりコミュニケーション」
- Agile という技術は、方法の特性だけで定義されるのでなく、それを行う技術者の価値観が、その定義の中心を占める！！

Waterfall の基本思想 (1)

- 本来は「分析・設計・実装・テスト・運用」だが、運用は開発ではないので簡単化のために、テストまでの工程のみを考えると次のようになる。
- 分析
  - 設計
    - 実装
      - テスト
- 滝が落ちているようなので、Waterfall という。

Prototyping

- システムのプロトタイプを、不完全でよいから作る。
- そして、それを実際に動かしながら、設計上の欠陥がないかどうかをチェックする手法。
- システム要求の解釈の違いを早期に確認できる。
- しかし、最終製品ではないので、プロトタイプが正しくても、最終製品が正しいとはいえない。

プログラムの正しさの検査：Verification と Validation

- これらの方法の重要な目的はV&V
- プログラムの「正しさ」を検査（テスト）するという行為は、次の二つに分類される。これを合わせて V&V ということも多い：
  - Verification（検証）
  - Validation (定着した和訳がない)

Verification & Validation by B. Boehm

- V & V の違いについての完全な共通見解はないが、Spiral model の提案者、B. Boehm の次の言葉は、大抵の人が賛成する。
  - Verification
    - **Are we building the product right ?**
  - Validation
    - **Are we building the right product ?**

Verification & Validation by 林

- Verification
  - **Are we building the product right ?**
  - すでに何らかの方法で明瞭に記述されたシステムの必要条件を仕様というが、その仕様に対して、システムが正しいかどうかを調べること。
  - 要求分析、設計のフェーズの生産物が仕様。要求仕様、設計仕様、詳細仕様などの言葉で呼ばれる。
  - WHAT-HOW の HOW を WHAT に合わせて check すること。
- Validation
  - **Are we building the right product ?**
  - システムが仕様作成以前の要件に合うか、あるいは仕様が、本来の要件にあうかを調べること。
  - WHAT-HOW の WHAT を check すること。

Weber と V&V

- V&Vは説明するのが難しいが、M.Weber の合理性理論を使うと、すっきり説明できる。

Max Weber の合理性の種類

- ドイツの社会学者 Max Weber (1864〜1920)は合理性を様々な角度から考察し、それにより「近代化」、「近代社会」を特徴づけようとしたことで有名。ただし、この理論は未完だった。
  - 参考文献：マックス・ヴェーバーの方法論的合理主義(矢野善郎著)
- Weber の対立する合理性の対
  - 価値合理性 vs 目的合理性
  - 実質合理性 vs 形式合理性
  - 理論的合理主義 vs 実践的合理主義
- 今の話題で関連があるのは、主に2番目の対と、目的合理性という概念。

実質合理性 vs 形式合理性

- Weber の「法社会学」から（矢野：p.78）
  - 法が「形式的(formal)」であるというのは、実体法上も起訴上も、もっぱら一義的で一般的な要件メルクマールのみが尊重されることである。
  - 〈実質 material 合理性〉が意味していることは、まさに、抽象的な意味解明の論理的一般化ではなくて、それとは違った性質の威信をもった規範が、法律問題の決定に対して影響力をもつべきであるということだからである。…中略…これらが外面的なメルクマールの形式主義をも、論理的中小の形式主義をも打破すべきであるというのである。
- メルクマール：Merkmal（独）特徴, 目印, (特性を示す)指標

実質合理性 vs 形式合理性(続き)

- 少し違う「定義」。先ほどは法律について。今度は経済行為(Wirtschaften, economic action)。
- 同じく、Weber の「経済と社会」から(Max Weber: Economy and Society, Vol.1, p.85, Univ. California Press, 林の意訳)
- 形式合理性(formal rationality, formale Rationalitaet)：ある経済的行為のシステムの各経済行為が、どの程度数字で表せて、また、実際に表されているかの程度により、そのシステムが、どの程度形式合理的であるかが決まる。価値が実数で表される「貨幣経済」のシステムは、高度に形式合理的であることになる。
- 実質合理性(substantive rationality, materiale Rationalitaet) 経済行動の選択基準を、今選択されている技術的に最適で可能な経済の方法論に照らして、ゴール志向的な合理的計算により選択されているという、純粋に形式的で、かつ、比較的明瞭な事実にのみ置くのではなく、たとえば、倫理的、政治的、功利論的、快楽主義的、封建的(staendich), 平等主義的、等の究極的基準、と、その経済行動の結果を照らしあわせること。この態度を実質合理的という。

Verification と形式合理性

- 形式論理による「プログラム検証」は形式合理性の典型例と考えられる。
- 形式的仕様が「一義的で一般的な要件メルクマール」、「数字で表現された経済行為」、「貨幣」、「生物的個体としての個」にあたる。
- Weber は、「形式合理的」という言葉を、「要件メルクマール」、「経済行為を表す数字と、その計算」、「貨幣」などの具体的な概念を遣って、法律、経済行為という具体的な領域の場合に説明した。同じように、ソフトウェア開発（ソフトウェアを作ること。

Validation と実質合理性

- Validationは実質合理的な行為である。
- Waterfall 開発では、仕様は開発の規範(norm)であり、それが開発の全体を通して指導原理となる。その最後の段階のテストが、要求仕様、つまり、すでにどの程度であれ形式化された要求、に対する実装の検討しかしなければ、それは形式合理的行為といえるが、最初の段階で割り出された要求（つまり、意識化されて、固定化され、開発チーム内、および、stakeholders 間で意思疎通可能な程度に明文化された要求）以外の要求の存在、あるいは、その割り出された要求の「本来の意図」に照らし合わせた「正しさ」を疑うという意味でのテストは、目的合理的といえる。

形式合理性から実質合理性へ

- 先の V&V=形式合理性/実質合理性、という解釈の視点からは、ソフトウェア工学のトレンドの変化は、形式合理性から実質合理性への移行として説明できる。
- Waterfall 法を厳守すると、実質合理性が、開発の最後にのみ起こる。
- その問題点が指摘され、prototyping や spiral model が提唱され、現在は、さらに agile methods が流行中と考えられる。
- これは、Waterfall, Prototyping, Agile における、サイクルと、その終端におけるカスタマーによる実質合理性への関わりの頻度を分析すると「疎明」できる。

Spiral model

- Wikipedia.com の解説から：
- The **spiral model** is a development model combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts.
- The spiral model was defined by Barry Boehm in his article *A Spiral Model of Software Development and Enhancement* from 1988. This model was not the first model to discuss iteration, but it was the first model to explain why the iteration matters. As originally envisioned, the iterations **were typically 6 months to 2 years long**. This persisted until around 2000.
- Each phase starts with a design goal and **ends with the client (who may be internal) reviewing** the progress thus far.
- Boehm の spiral の図

Agile software development

- Wikipedia.com の解説から：
- **Agile software development** is a conceptual framework for undertaking software engineering projects. There are a number of agile software development methodologies, such as those espoused by the Agile Alliance, a non-profit organization.
- Most agile methods attempt to minimize risk by developing software in short timeboxes, called iterations, which typically **last one to four weeks**. Each iteration is like a miniature software project of its own, and includes all the tasks necessary to release the mini-increment of new functionality: planning, requirements analysis, design, coding, testing, and documentation.
- Most agile teams are located in a bullpen and include all the people necessary to finish software. **At a minimum, this includes programmers and their "customers." (Customers are the people who define the product.** They may be product managers, business analysts, or actual customers.)

形式合理性の重要性

- Agile への流れは形式合理性が軽視されていることを意味しない。
- 現実には、形式合理性のためのツールであった形式的技法(Formal methods)から生まれた semi-formal な方法である、デザイン・パターン、UML などの「標準化」(これが semi-formalの本質) の技法が、猛烈な勢いで進化している。
- 実質合理性重視の傾向が生まれたのは、形式合理性が成功したから。そして、その形式合理性技術の整備は、ますます進んでおり、agile法や lean production methodのように、「実質合理性」の「形式合理化」さえ起きている。
- むしろ、形式合理性が実質合理性の「高速」な運用を可能にしている？

なぜ Weber?

- こういう話をすると、「Weberは、もう古い」という意見を聞く。たとえば、林の同僚から。(林の同僚は人文・社会科学者)
- 答がある。
- アメリカの社会学者 Talcott Parsons は、Weber 理論を mosaic theory と批判した。
- Parsonsは general theory を求めた。
- しかし、ソフトウェア工学・計算機科学の歴史は、これと逆だった。最初に general theory が求められ、今は、それぞれのソフトウェア開発プロジェクトが、Wittgenstein の language game にたとえられる時代。

とりあえずの結論

- ソフトウェア工学は、社会学的理論や考察に多くを学べる。その結論は経営学的・生産工学的な知を提供し、大きな practical な利益をもたらすはず。
- 逆に、ソフトウェア工学は、社会が情報技術に reflect する「入り口」として、社会学的研究における重要な対象になるはず。
- そして、それにより、日本社会の modernization における「位置」が見出せるのでは…