ソフトウェア開発法の新傾向

林　　晋

A New Trend in Software Developments and Ex Post Modularity

Susumu HAYASHI

Agile development method is a new trend in software development methodology. This adaptive method is to overcome increasing complexity of systems and speed-up of businesses. However, the method is capable only when it is based on theoretical and systematic methods. This combination of agility and discipline is related to Chuma's notion of "ex post modularity." These thinking ways would be consequences of the further modernization of society.

Key words: software engineering, agility and discipline, ex post modularity

Waterfall モデル以来[1]，多くのソフトウェアの開発法のモデルが提唱されている．現在のトレンドは，"agile" というキーワードでよばれる方法である．最初の有名なソフトウェア開発モデルである waterfall モデルが計画経済型であったとすれば，agile とは市場経済的であることといえる．この「計画経済」から「市場経済」への変転は，情報システムの複雑化とビジネスのスピードアップとに，ソフトウェア技術が対応した結果とみることができる．この複雑化と高速化という要因は，ソフトウェア技術に限ったことではない．「さらなる近代化」とグローバル化の波を受けて，ソフトウェア産業，ハードウェア産業，教育，行政など，多くの社会的活動が同じ運命にある．

中馬の論文[2]によれば，半導体露光装置における競争力の変化に，この変化の典型的な事例がみられるようだ．この稿では，中馬が半導体露光装置の開発技術の変化を示すために使った ex post modularity の視点とソフトウェア工学における agility の関係を指摘し，さらに，この変化を社会の「近代化」の避けがたい帰結ととらえる．

1. Agility，トヨタ開発法，UNIX

プロジェクトのマネージメントは難しい．現代のように急速に変化する環境では，特に難しい．歴史上，もっとも複雑な人造物とさえいわれるソフトウェアの開発は，当然ながら難しい．組み込みソフトウェアの場合，それが組み込まれるハードウェアの実体が，ソフトウェア開発プロジェクトがスタートする時点ではっきりしていないことは当たり前だ．標準に準拠したソフトウェアを開発する場合も，標準が固まってから開発を始めたのでは遅い．標準が固まるころには，すでに対応商品が市場に投入されてもおかしくない．「すべてを見切ってから走る」のでは目標を達成できず，「走りながら考える」しかないのである．これは技術者の姿勢や倫理の問題ではない．「市場」がそう要求しているのである．ソフトウェアの開発を科学的・工学的にマネージすることを目指すのがソフトウェア工学であり，単純化すれば，その基本目的は「ユーザーの目的に合うソフトウェアを低コストで短期間に作成する方法を提供すること」だといえる．そして，現代では，これに「いかに迅速に変化に対応するか」という重要な要素が加わっているといえるのである．

有名なソフトウェア開発モデルである waterfall モデルでは，開発を，河が上流から下流に流れるように，ソフトウェアの目的を記述した仕様から始めて（これが源流，上流），その仕様に合ったプログラム（これが河口，下流）を産出する非可逆的プロセスと考える．仕様を書いた仕様書

神戸大学工学部（〒657-8501 神戸市灘区六甲台町 1-1）　E-mail: susumu@shayashi.jp

がプロジェクトの計画書であり，すべては最初に立てられた計画どおり整然と進むべきだという思想である．現実の開発で，すべてが整然と進むわけはないが，この流れを乱し，下流が上流に影響を与えることを「悪」としてとらえ，それを少なくすることを「善」とするのが waterfall の思想なのである．

当然ながら，このような方法論では「走りながら考える」ことはできず，現実に適応しない．そのため，新しい方法論が数多く提唱されている．代表的なものとして，waterfall の要素的単位（要求分析，設計，コーディングなど）の終わりごとに見直しを行い，必要なら，いったん上流に引き返し修正を行う螺旋的な開発モデル spiral モデルがある．現実とのぶれを，定期的なアセスメントで修正しようという考えである．人間がすべてを事前に計画してコントロールできるという，決定論的の幻想は捨てるのである．

これが，現在，世界中で流行しつつある agile 法となると，さらに非決定論的で適応的（adaptive）になる．米国工業がドイツ，日本の工業に追い上げられた際に，それに対抗すべく，日本やドイツの競争力の分析を行い，その上に構築した生産法のひとつに agile manufacturing がある．機械工学，医療，生産工学などでおもに使われる，この agile という形容詞が，ソフトウェア工学の agile 法のもとである．この agile 法には多くのバリエーションがあり，waterfall や spiral モデルに比べて，一律に論じることは難しいが，その精神は，次の “Agile Manifesto” [http://agilemanifesto.org/] に集約されている：

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

つまり，（i）体系的な開発プロセスやツールよりは個人技と協調，（ii）完全なドキュメンテーション（仕様）よりは実際に動くソフト，（iii）詳細な契約よりはカスタマーとの協力関係，（iv）計画の遵守よりは変化への対応，だというのである．XP（eXtreme Programming）という極端な意見では，「仕様は悪だ」，つまり「計画書を書くことは悪だ」とさえ主張される．

これをみれば，agile がある意味で「日本流」であることがわかる．この突拍子もない見方は，アメリカの agile 法の推進者たちが唱えはじめた説である．例えば，agile 法の精神と，柔軟さを売り物にするトヨタ生産方式の関連が，米国のソフトウェア・コンサルタント Poppendieck により指摘されており，Poppendieck は，この類比をもとに，トヨタ生産方式の米国版理論化である Lean production の名にちなむ Lean software development というソフトウェア開発法を提唱している[3]．

筆者と黒川は，この事実をキーに，日本のソフトウェア産業を改善する方法について論じた[4]．しかし，その論文でも指摘したように，それがきわめて「日本的」なものであっても，それは日本固有のものではないのである．UNIX 的思考法[5] を知っている人ならば，agility の minimalism 的・適合的性格が，UNIX 的思考法，あるいは，UNIX の哲学[5] とされるものにきわめて似通っていることに気づくだろう．この UNIX ハッカーたちが共有する「UNIX 文化」は，日本的工業生産法とは無縁の存在である．その源流は，現代の Web に代表される思考補完的で interactive なコンピューター環境という，1960 年ころの米軍の研究機関 ARPA をめぐる人たちに求めることが可能なようだ[6]．UNIX 的思考法，agile 法は，その人的系譜からしても，この流れを汲むものと筆者は考えている．

2. Ex post modularity

藤本は，日本産業の特性と，それに基づく日本の競争力を論じる際に，「事前合理性 vs. 事後合理性」という対立軸を用いた[7]．藤本は，鋼板プレス加工のような作業も含めて，生産とは情報転写であるという説をとる．そして，日本が得意とする産業は，情報転写が困難で，それゆえに事前の計画が難しく，結果として，作りこみ・摺り合わせを必要とする自動車産業のような分野であると結論づけた．そして，転写が容易なゆえに，waterfall のように事前の計画立案が機能するソフトウェアなどの分野，つまり「事前合理性」が支配的な分野には日本は弱いとした．藤本によれば，生産計画よりは，顧客の注文を契機として動くカンバン方式や，生産ラインを止めることをいとわないトヨタ生産方式の発想は，これと対比して「事後合理性」とよばれるべきものとなる．つまり，事前の計画に固執せず，とにかくやってみて，その後に合理的に修正・改善をかける方法である．

もうすでにお気づきと思うが，この事後合理性は，agility の方法そのものである．藤本がいうように，コンピューターの記憶は書き込みやすい．しかし，ソフトウェアの生産では，単に目の前に情報を転写するのでなく，それを目的にあうように加工し，または作り出してから記録（転写）しなくてはならない．たとえ，媒体に記録すること自身は簡単でも，記録するデータの用意まで考えれば，特に，それを変転する顧客や市場のニーズに合わせて作ることを考えれば，ソフトウェアの「転写」は，容易どころかきわめて困難なのである．

このように藤本の説には，そのままでは現実を説明できない部分がある．中馬は，半導体露光装置開発の研究を通し，ex post modularity という，事前合理性，事後合理性という単純な図式への alternative を提案している[2]．筆者は，中馬の“modularity”は，むしろ「体系性」という，より大きな概念で置き換えたほうが適当ではないかと考えているが，いずれにせよ，それは modularity あるいは体系（システム）を，予測可能性（マックス・ウェーバーの意味での「計算可能性」）を保証するものとしてでなく，問題点や現実とのずれが発生した場合に，その修正・改善を効率的に行うものとしてとらえるという考え方であろう．同様の考え方は，科学を，その体系性による確実性を保証するための装置としてでなく，むしろ誤りを検出するための装置としてとらえた科学哲学者ポッパーの思想の中に見て取ることができる．ポッパーは，数学者ヒルベルトの公理論以降の数学の演繹体系は，絶対的真理を保証する装置ではなく，仮説の合理的批判のための装置になったと指摘したのである[8]．

中馬は，日欧台の半導体露光装置産業を分析することにより，「複数の組織（企業，大学等）が，同じアーキテクチャーを共有して育てる，オープンでモジュラー型の（欧州型の）開発方式のほうが，内製化傾向の強い（日本型の）開発方式よりよい結果を出しつつあり，それにより日本企業の競争力が急速に低下している．そして，その変化は，（半導体露光装置の）システムの複雑性が，オープン・モジュラー型でないと克服できないような，ある臨界点を超えたからではないか」という視点を提示している．これは，複雑なシステムに対処するには，体系性は，最大限の予測可能性を目指すものでなく，むしろ藤本のいう事後合理性のために使うべきだという意見と理解できるだろう．

ソフトウェア科学者・工学者の立場から，この中馬の説を聞いて思い出すのが，Linux である．一面識もない世界中のボランティア・プログラマーがネットワークにより知識を共有することにより，周到に計画され，大量の資金と労力を投入したプロジェクトより信頼性の高い OS を生み出したという事実は，「事前計画に基づく大聖堂建設型の開発に，自然発生的なバザールのような開発が勝利した」事例として，ソフトウェア技術者・工学者にショックを与えた．Linux は，半導体露光装置の事例より，さらにオープンで非事前計画型のモジュール型開発の典型的な成功例なのである[9]．

3. 変化のための体系性

最終的な結論を下すにはまだ研究が十分ではないが，筆者は，半導体露光装置，agile 法，Linux，UNIX 哲学は，すべて ex post modularity と同様に，「変化のための体系性」という共通性をもつのではないかと考えている．体系というと，固定化したものを連想しがちだ．思想家リオタールが「大きな物語の終焉」としてポストモダン社会を描き出して以来，体系性，近代化，正当性のための物語（narrative）などの諸概念を混同して用い，それにより「体系の無効化」と「近代の終焉」を結論づける論点がある．しかし，これは誤りである．変化のための体系は，その体系自体を否定し「前に進む」ための体系なのである．それは外界から身を守る甲殻でなく，柔軟な体を支えるための骨格なのである．

ある意味できわめて「ポストモダン的」な Linux の成功は，世界中のハッカーたちが，UNIX 的思考法という共通文化をもち，UNIX のアーキテクチャーという言語を共有できたことにある．そこには，事前計画性は希薄だが，UNIX 文化といわれるような共通の価値観と，国籍や母国語を超えた高いコミュニケーション可能性があったのである．つまり，そこにはある種の robust な体系が存在していたのである．それが骨格としての体系である．

アルゴリズム的な決定論ではないが，非決定ながら多数が自然に方向性を共有する共通のスタイルを「文化」とよぶとすれば，われわれが現在目の前にしているレベルの複雑システムに対峙する際に必要なものは「文化としての体系」なのだろう．そして，おそらく，事前計画的に解決可能な問題の多くがすでに解決されてしまった，つまり，あまりに「近代」が成功してしまった後の現在は，その意味においてポストモダンの時代であり，それゆえに，さまざまなレベルの共有文化を武器として，試行錯誤的，適応的なチームワークによって問題解決をしていくしかない時代なのではないだろうか．そして，そのチームワークにとってもっとも重要なことは，コミュニケーションの媒体としての（標準などの）体系性なのである．

そして，この標準は，硬い骨格が機敏な動作を保証するように，自由性を保証するためにあり，自由性は事前計画の不可能性に対処するための戦略的な技術なのであろう．藤本がソフトウェア分野の日本の弱さの原因とした事前合理性の弱さとは，実は，このような「変化するための体系構築能力」の弱さのことなのではないだろうか．もし，この説が正しいとすれば，弱さの原因は「文化的」なものである可能性が高く，中馬の半導体露光装置の事例からしても，日本の競争力を考える際に無視できないことに違いない．

文　　献

1) W. W. Royce: “Managing the development of large software systems: Concepts and techniques,” *Technical Papers of Western Electronic Show and Convention (WesCon)* (Los Angeles, August 25–28, 1970).
2) 中馬宏之：“我が国サイエンス型産業が直面する複雑性と組織限界：半導体露光装置産業の事例から”，一橋ビジネスレビュー 2005 年春季号（東洋経済，2005）.
3) M. Poppendieck and T. Poppendieck: *Lean Software Development: An Agile Toolkit for Software Development Managers* (Addison-Wesley, 2003).
4) 林　　晋，黒川利明：“二つの合理性と日本のソフトウェア工学”，科学技術動向，9 月号（文部科学省科学技術政策研究所科学技術動向研究センター，2004）.
5) M. Gancarz: *The UNIX Philosophy* (Digital Press, 1996). 芳尾桂訳：UNIX という考え方（オーム社，2001）.
6) M. M. Waldrop: *The Dream Machine, J.C.R. Licklider and the Revolution That Made Computing Personal* (Penguin USA, 2002).
7) 藤本隆宏：能力構築競争―日本の自動車産業はなぜ強いか―，中公新書（中央公論新社，2003）.
8) K. Popper: *Realism and the Aim of Science* (Part I, Chapter 4) (Routledge, 1992).
9) 林　　晋：“モジュール化と科学”，特集あなたが考える科学とは（第 2 回）収録，科学，6 月号（2001）800–801.

（2005 年 3 月 14 日受理）