第12章

パラダイム雑感

12.1 科学革命としてのソフトウェア革命

ソフトウェア革命はいつ起こるであろうか。パラダイム論者トーマス・クーンの「従来のパラダイムが行きづまり，危機的状況になったとき，新しいパラダイムが生まれ，科学革命が起こる」という科学史観を借りれば，ソフトウェア革命は，ソフトウェア危機が新しいプログラミングパラダイムによって打破されたときに起こると言える。

トーマス・クーンの著書「科学革命の構造」は示唆的である。彼自身は，科学の発展が他の分野とは決定的に異なることをパラダイムという概念を用いて論証するのが目的であった。したがって，彼のパラダイム論を他の分野へ転用することは彼にとって迷惑には違いないが，あえて，ソフトウェア革命を科学革命のアナロジで論じてみたい。

まず，トーマス・クーンの科学革命，すなわち「専門家たちに共通した前提をひっくりかえしてしまうような異常な出来事」の構造を要約すると以下のようになる。(図 12.1)

① [パラダイムの確立]

1つの科学革命が完成すると，その立役者となったパラダイムに沿った通常科学が確立する。

② [パラダイムの改良]

科学者集団の一人ひとりは，通常の研究活動の一環として，このパラダイム

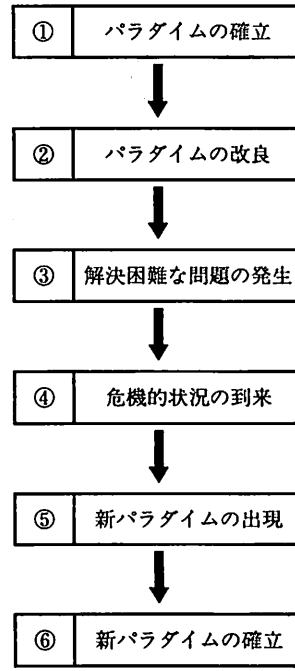

図 12.1 科学革命（パラダイム転換）のプロセス

を用いて残された問題を解決するというパズル解きを行い，パラダイムを整備する。

③［解決困難な問題の発生］

このパラダイムでは解決できない問題が出現し，このパラダイムの変則性の解決が重要課題となる。

④［危機的状況の到来］

このような変則性が多発し，かつこれらの解決困難性が科学の発展の障害となる危機的状況が生じる。

⑤［新パラダイムの出現］

視覚の転換を促す新しいパラダイムが出現し，この危機を打開するパラダイム転換が起こる。

⑥［新パラダイムの確立］

科学者集団が新しいパラダイムへ移行／改宗し，科学革命が完成する。

このような科学革命を繰り返しながら科学は進歩していくというのがトーマス・クーンの科学史観である。トーマス・クーンは，パラダイムを「一般に認められた科学的業績で，一時期の間，専門家に対して問い方や答え方のモデルを与えるもの」と定義している。

さて，トーマス・クーンのパラダイム論の紹介はこれぐらいにして，本題に入ろう。本題とは，臆面もなくこの科学革命の構造のアナロジとしてソフトウェア革命を語ることである。まず，本書では，プログラミングパラダイムを「プログラムの作り方に関する規範，すなわち，プログラムの設計手順やプログラムの構造の決め方，プログラムの記述方法を規定するもの」と定義する。これまで種々のものが提案され，その一部は実用になっている。多くの場合，各プログラミングパラダイムに対応するプログラミング言語や設計ツールが提供される。

先の科学革命の6段階に対応させると，ソフトウェア革命は次のように進展する。

① [パラダイムの確立]

最初のプログラミングパラダイムである手続き型パラダイムの出現は，フォン・ノイマンが提案したプログラム内蔵方式の最初のコンピュータ EDSAC-1 が1949年に完成した時にさかのぼる。コンピュータへの命令を1ステップずつ手続き的に記述するというプログラムの作り方は，その後の半世紀近くの間，まったく変わっていないのである。

② [パラダイムの改良]

その間，手続き型パラダイムは，その時々のプログラムの作り方に関する課題を解決するために発展してきた。最初は，効率良くプログラムを作るためにアセンブラ言語や高級言語が開発され，デバッガやエディタなどのツールも作られた。さらに，信頼性の高いプログラムをつくるために構造化技法が適用され，グラフィカルユーザインタフェースを用いて使い勝手の良いプログラム開発環境が普及し始めた。

③ [解決困難な問題の発生]

従来の手続き型パラダイムでは，新しい社会的ニーズに対応できなくなってきた。ソフトウェアの生産性の向上が重要課題となった。

④［危機的状況の到来］

規模と量と質の面からソフトウェア危機が深刻になってきた。

⑤［新パラダイムの出現］

新しいプログラミングパラダイムが数多く提案され始めた。プログラミングスタイルに関するパラダイムシフト（パラダイム転換）の兆しが見え始めた。

(現在の状況)

⑥［新パラダイムの確立］

新しいプログラミングパラダイムへの移行/改宗が進み，ソフトウェア革命が完成する。

筆者は，第6段階のソフトウェア革命の完成がいつになるかを知らない。新しいパラダイムに移行して革命をもたらすのが伝統的ルールに縛られない若い人たちであることだけは確かである。

12.2 ソフトウェア産業論

12.2.1 ソフトウェア産業

ここでは，第3章で述べたようなソフトウェア危機回避のためのドラスティックなシナリオ案に対応する技術の位置付けを明確にするために，ソフトウェア産業の進化過程を表12.1のような図式でとらえる。この分類では，必ずしも世の中一般の用語の定義と一致しない部分もあるが，あえて単純な図式を作ってみた。

表12.1 ソフトウェア産業の進化過程

ソフト産業の形態	主要な技術職	主 要 技 術
労働集約型産業	プログラマ	自動化 (CASE)
知識集約型産業	設計者	標準化 (パッケージ)
知恵集約型産業	業務専門家	エンドユーザコンピューティングツール

(1)　労働集約型産業

現在のソフトウェア産業の大部分に相当する労働集約型産業においては，生産コストあたりの生産量という生産性の効率向上が重要である。ここで最も重要なメトリックスは，ステップ数／人月あるいはステップ単価である。メーカは，CASE ツール等を用いて自動化率を向上させ，人海戦術からの脱皮の努力をしている。しかし，この尺度の致命的欠陥は，ソフトウェアの価値（質）を規模（量）ではかることである。ソフトウェア産業の未熟さは，ソフトウェアの価値をステップ単価や工数（人月）でしか計算できないというソフトウェアメトリックスの未熟さに起因している。

(2)　知識集約型産業

ソフトウェアの生産性は本来以下のように定義されるべきである。

ソフトウェアの生産性＝生産物の価値／生産コスト

「生産物の価値」はユーザの視点で決まるべきものであり，高品質のソフトやベストセラーのアプリケーションパッケージの価値は高い。このようなアプリケーションパッケージの開発には，業務の知識と情報処理技術の双方が必要である。

いま，大まかに現状を把握するために，生産物の価値の代わりに売上高に着目し，生産コストの代わりにその大部分を占める人件費に対応する従業員数を用いると，従業員一人当りの売上高という指標が得られる。この指標で過去 10 年間を振り返ると，売上高の約 7 倍の伸びに対し，わずかに 2 倍弱程度の伸びである。売上の伸びが従業員数の伸びで支えられているという労働集約型産業の実態がよくわかる。

今後，アプリケーションソフトウェアのビジネスは，システムインテグレーションの方向に発展していくと思われるが，この場合，業務の知識と情報処理の知識をノウハウとして蓄積することが必要である。

(3)　知恵集約型産業

情報処理システムが，業務の効率化ではなく，経営戦略の実現に用いられるようになると，ソフトウェア開発も，効率よりも効果が重要視される。SIS や

BISでは，何を作るかが最も大事であり，それを決めるのは業務専門家である。業務専門家が知恵をしぼって意思決定支援などの非定形業務用ソフトウェアをタイムリーに作っていくためには，エンドユーザ自身が開発でき，かつ保守拡張ができる必要がある。

このようなエンドユーザコンピューティングを実現するためには，そのためのツールや環境を提供しなければならない。自動化ツールや標準パッケージなどの情報処理技術を統合した上に，きめ細かく分類された応用分野対応(domain-specific, application-oriented)の業務の知識に基づいた使い勝手のよい環境の構築が必須である。潜在ユーザとしての業務の専門家の数を考えると，市場規模は現在の情報サービス産業のそれ（数兆円規模）を大きく上回ることになる。

12.2.2 ソフトウェア異質論

ハードウェア産業の華々しさとは対照的に，ソフトウェア産業はいまだに未成熟である。長い間，ソフトウェアの生産技術がハードウェアのそれとのアナロジーで語られることが多かった。「ソフトウェア工場」，「クリーンルーム方式」，「導通試験」，「設計と製造の分離」，「部品化」などの概念や用語が用いられてきた。

しかしながら，実際にはソフトウェアはハードウェアとは異質な特徴をもつ。文化と文明，精神と物質，論理と物理など，いろいろな視点でその違いを語りえるが，現実面での決定的な違いは，ハードウェアに比べてソフトウェアは見えない部分が多いため，定量的なもの差しが少ないことである。ソフトウェア産業を真に「産業」たらしめるために，そして，ソフトウェア工学を真に「工学」たらしめるためには，第一にソフトウェアの生産に関与する諸要因を観測可能にすること，第二にその観測データを科学的根拠のあるもの差しを用いて評価可能にすることが望まれる。ソフトウェアで解決しようとしている問題に関する評価，ソフトウェアそのものの多面的評価，開発者の能力に関する評価，工程管理に関する評価，ツールや環境に関する評価など，ソフトウェアメトリクスの対象はあまりにも多い。

12.2.3　パーセプショントランスファー

日米のハイテク技術比較で5年先，10年先も米国がリードしている分野として，医療，宇宙航空と共にソフトウェアが挙げられているが，ソフトウェア危機の問題は，1つの国の産業の問題としてではなく，いまや地球規模の問題として解決していかなければならない。

この分野のブレークスルー技術を生み出すためには，ソフトウェア科学，ソフトウェア工学，ソフトウェア産業，エンドユーザの協力が必須となろう。しばしばテクノロジートランスファ（技術移転）の難しさが指摘されるが，その前にお互いのパーセプションギャップ（問題認識の溝）を解消することが重要であり，テクノロジートランスファとは逆方向のパーセプショントランスファ（問題認識の移転）が鍵となるように思われる。

12.3　第4のソフトウェア危機

本書の冒頭で，情報化社会は「あかるくらい」社会であると述べた。第2章では，その暗い面として，規模と量と質のソフトウェア危機について言及した。このソフトウェア危機は，生産技術の面からとらえたものであるが，他にも種々の観点がある。

たとえば，コンピュータ関連業界がソフトウェア開発要員として理工系の学生を大量に採用することにより，他の産業界の労働力不足がより深刻になる。これでは，ソフトウェア危機が実は産業の危機ということになる。また，エンドユーザコンピューティングの進展と共にパソコンを中心としてそのソフトウェアの本が急増している。これらの本が本屋の売場をどんどん占領しはじめている。これは，本屋の危機，出版業界の危機にとどまらず，文化の危機と言えなくもない。コンピュータ犯罪も増加している。コンピュータウイルス騒ぎは日常化しているし，磁気カードにからむ不正も多い。国税庁の調べでは，最近の脱税の手口にコンピュータのソフトウェアを悪用したものが増えている。コンピュータウイルスは敵の軍事システムを破壊する兵器として扱われはじめている。

ここでは，規模，量，質に続く第4のソフトウェア危機として，知的所有権の問題と製造物責任の問題を取り上げたい。

12.3.1 知的所有権

Intellectual Property は，知的所有権または知的財産権と訳される。知的所有権は，工業所有権と著作権に分類され，前者には，特許権，回路配置権，トレードシークレット法，商標権などがある。

特にソフトウェアに関する知的所有権をどんな方法で保護するかという問題が注目されている。既存の制度では，特許や著作権がある。従来，ソフトウェアは特許になりにくかったが，最近は方法特許やコンピュータに組み込んだ装置特許として認められることが多くなってきた。一方，ソフトウェアを著作物とみなして著作権で保護することも行われるようになってきた。しかしながら，従来，物としての製品やその製造方法を念頭において作られた特許制度や文芸作品の文章などの表現を対象とした著作権は，そのままソフトウェアに適用すると不都合な面もある。ソフトウェアの知的所有権に関して各国でその保護の方法が異なっているが，米国や欧州ではソフトウェアを言語著作物（リテラリワーク）として著作権法で保護する考えが強い。この問題は，経済のグローバル化と共に，その標準化の必要性に迫られており，関税貿易一般協定・多角的貿易交渉の対象になっており，世界知的所有権機関（WIPO：World Intellectual Property Organization）でも種々の努力がなされている。

知的所有権への関心の高まりと共に，ソフトウェア関連の特許出願が急増している。ソフトウェアに関するトラブルも増加している。特に，最近はワークステーション，パソコンの普及やビットマップディスプレイの普及につれて外部インタフェースの部分が多くなり，ウィンドウやポインティングデバイス関連の類似性が問題になりやすい。今後，マルチメディアデータを簡単にコンピュータで扱えるようになるとその著作権も大きな問題となろう。すでに，他人の持っている特許を買い取って，その特許を侵害していると思われる会社にライセンス料を請求する特許管理会社も出現している。その反対にソフトウェアの権利化の弊害を主張して，良いソフトウェアを普及させるために無料で配布する団体（FSF：Free Software Foundation）もある。

このようなソフトウェアの知的所有権の保護が行き過ぎれば，ソフトウェア産業の発展が阻害される。そればかりか，今日のように日常生活で接しているあらゆるものにコンピュータが使用されるようになると，最も迷惑を受けるのがエンドユーザということにもなりかねない。にもかかわらず，ソフトウェア

の知的所有権の保護はさらに強化されていくと思われる。

この視点で来たるべき21世紀を展望すると，規模，量，質に続く第4のソフトウェア危機が知的所有権によってもたらされる可能性がある。日本の約50倍の弁護士がいるといわれる米国の訴訟社会が最初に本格的に日本にもたらされるのはこの分野かも知れない。その時には，プログラマ不足に代わって弁護士，弁理士不足が問題になるだろう。訴訟に対抗し，違法行為を合法化するためにマネーロンダリングならぬテクノロジーロンダリングがはびこるだろうか。ソフトウェア開発技法の大半が法律問題で占められ，ソフトウェア産業が法律産業と化すことはないとは思うが。

反面，知的生産物の権利が尊重されるということは，この分野の研究者や技術者に大きなビジネスチャンスが生まれることにもなる。第3章のソフトウェア危機回避のシナリオの1つとして，情報処理技術者の自由業化について述べたが，アイデア1つで億万長者も夢ではなくなる。

12.3.2　製造物責任

第3のソフトウェア危機として，第2章で「質」の問題について述べたが，製造物責任（PL：Product Liability）の制度化と共に第4のソフトウェア危機に発展するかもしれない。日本では現在立法化を検討中であるが，米国で1975年に成立した製造物責任法は，ある製品を使用中に事故が起こったとき，製品の製造者に過失がなくてもその責任が問われるものである。有名な例として，濡れた猫を電子レンジで乾かそうとして死なせてしまった事件があるが，予見可能な誤用に対して適切な処置をとっていなければ製造者に賠償責任が生じる。

日常生活の中で，お菓子の袋に「万一変質品がありましたら…郵送料弊社負担でお取り換えいたします」とか，フィルムの箱に「…新しいフィルムとお取り換えいたします。それ以外の責はご容赦願います。」などと断っている文を見かけるが，ソフトウェアに関してはどうであろうか。プログラムにバグがあれば，それを取り除いたものと取替えるのは当たり前と思うが，パソコンのソフトウェアパッケージやゲームソフトあるいは量産品の組込みソフトなどでは取替えは大変である。第2章で述べたように，ソフトウェアの誤りが原因で社会的な影響力の大きなシステム事故が発生したとき，誰がどのような補償をするのか。

製造物責任の観点では，ソフトウェアの誤った使い方で生じた損害も補償することになる。かつて米国の株価の大暴落にソフトウェアによる自動取引が関与したことがある。AIを応用した金融証券分野の商品が投資家に大きな損害を与えたこともある。この責任は，投資家が負うのが当たり前，とは言えなくなるかも知れない。投資家に販売した会社，アプリケーションソフトウェアの開発者，その開発ツールの開発者，そのツールの開発に用いたプログラミング言語のコンパイラ開発者，あるいは言語設計者までさかのぼって無過失責任が問われることになれば，ソフトウェア産業は簡単に壊滅する。ソフトウェアに関しては「誤った使い方」の範囲はどのように限定されるのであろうか。「…健康法」という本の影響を受けて健康を害した人やある小説の手口を真似て犯罪をおかした人に補償することにはならないと思うが。

ただ，ソフトウェアパッケージを購入すると，分厚い説明書が付いていて，最初に長々と知的所有権に関する警告文が書いてあり，その次にまた長々と製造物責任を回避するための使用上の注意が書いてあり，最後の方に使い方が書いてある，ということにはなるかもしれない。

12.4 インテリクローン

12.4.1 究極のプログラミングパラダイム

道具としてのコンピュータを活かして用いるためには，エンドユーザコンピューティングの実現が不可欠である。そのためには，従来の手続き型パラダイムからの脱却はもちろんのこと，プログラミングの概念そのものから脱却できなければならない。そこで，究極のプログラミングパラダイムについて考えてみよう。ここでは，究極のプログラミングパラダイムによって作られる究極のソフトウェアを「自分がやりたいことをやってくれる」知的なクローン (Intelligent Clone) という意味で，インテリクローンまたはソフトウェアクローンと呼ぶ。

一般に，ソフトウェア技術は「モデリング & シミュレーション」技術である。従来のプログラミングではシミュレーション可能なモデルを作るために論理を駆使してきたが，そのためには熟練技術が必要であった。非手続き的アプローチによりシミュレーションのための制御の概念を取り除くことには成功し

たが，図 4.1 でも示したように対象モデルを計算モデルに変換するという作業は残った。

このプログラミング形態を大きく変えたのは，第 10 章で述べた人工知能技術である。知識と推論という人間の知的能力のアナロジーにより，モデリング & シミュレーションを実現するために論理を組み立てるというプログラミング作業がなくなった。その代わりに望ましい推論結果を導くための知識の調整 (adjustment) 作業が必要である。これもある程度の熟練技術は要求される。そのため，本書では人工知能技術を簡易プログラミングと位置付けている。

インテリクローンを実現するためには，もっと簡単にやりたいことが伝わらなければならない。熟年夫婦や有能なベテラン秘書との会話のようにならなければ，自分の分身とはいえない。そのためには，調整に代わって適応 (adaptation) あるいは学習 (learning) 機能が必要である。その可能性を探るためにマルチエージェントとニューラルネットワークについて考察する。

12.4.2 マルチエージェント

自分がやりたいことをやってくれるインテリクローンに近いメタファとして，秘書を考えてみよう。秘書は，マルチパラダイム型の知識を駆使して業務を代行してくれるエージェントである。

筆者は，第 10 章で紹介したマルチパラダイム型の知識表現機能を持つエキスパートシステム構築ツール ES/X90 を用いて，知的秘書システム KISS (Knowledge-based Intelligent Secretary System) のプロトタイプを開発した経験を有する。このシステムコンセプトは，一見したところ非定形業務のように見えるユーザ（筆者自身）の雑多な中間管理業務を小さな定形業務の集まりと見て，可能な限りコンピュータ化し，本質的な意思決定の部分のみをユーザが行う，というものである。まず業務を細かく分類し，フレーム表現を用いて図 9.9 に示したような業務のオブジェクト階層を作る。業務が発生する度にこれを用いて業務のインスタンスを生成する。次に，業務の種類毎に定形的な処理部分をルールメソッドまたは述語メソッド化する。ユーザの判断が必要なところは会話型処理にする。このようなシステムはまだプロトタイプの段階であるが，オフィスにおける業務の電子化と分散コンピューティングが進めば，早晩実用になるものと期待している。

分散コンピューティングの世界では，グループウェアを経由して，このようなエージェント同士がお互いにメッセージをやり取りして分散協調システムを形成することになる。たとえば8.3節で例題に用いたOOOシステムにおける自律的なオブジェクトがこのようなエージェントとして機能すれば，全体がマルチエージェントシステムとなる。この時の1つの課題は，各エージェントにどのように適応機能を持たせることによってシステム全体の適応機能を実現するかということである。そのためには，エージェントにマルチパラダイムの知識表現のみならず，メタ知識やリフレクション機能が必要となろう。

12.4.3 ニューラルネットワーク

自分がやりたいことをやってくれるインテリクローンに近いもう1つのメタファとして，ニューラルネットワークを考えてみよう。これは，人間の脳神経網の生理学的知見に基づく並列計算モデルにより，現在のフォン・ノイマン型アーキテクチャのコンピュータが不得意とする認識・理解機能の実現を試みるものである。学習による自己組織化機能を有し，究極のプログラミングパラダイムとして期待される。

筆者は，かつて思考過程のモデル化にニューラルネットワークを用いたことがある。思考過程を「記憶された概念の想起の列である」と規定すると共に，その基本機能としての連想作用と意識作用に注目し，思考エネルギーというコンセプトを導入して次のように規定した。

①意識とは，大脳における思考エネルギー分布の集中作用であり，十分な集中が概念を明確に想起させる。
②連想とは，集中した思考エネルギーの拡散作用であり，十分な拡散が新たな概念の想起を促進する。

このモデルでは，思考過程は思考エネルギーの絶えざる集中と拡散の中で生じる概念想起の列となる。学習は，思考エネルギーの拡散の方向を決めること，すなわち，特定の方向へ拡散しやすくすることであり，ニューラルネットワークにおけるニューロン間の結合係数の増加に対応する。

このモデルをRAC(Repetition of Association and Concentration)モデルと名付けた。従来のニューラルネットワークモデルにありがちな計算モデルと

しての意味論の曖昧さを避けるため，確信度付きの並列プロダクションシステムとして計算の意味を規定したが，詳細は文献（95）等に譲る。

このモデルを用いて，対話学習による概念学習の可能性を探るために種々のシミュレーションを行った。その方法を**図 12.2** に示す。f は連想に対応する拡散関数で，従来のニューラルネットワークと同じである。g は意識に対応する集中関数で，RAC モデルに固有である。h はある概念の思考エネルギーが閾値を超えたらそれを出力する言語化関数である。β は外部入力を思考過程（fg サイクル）に取り込むときの重み係数で，注意度を表す。ニューラルネットワーク（拡散関数）の結合係数の初期値と学習規則，各種パラメータ，集中関数などの設定の仕方により，種々のモデリング & シミュレーションが可能である。ここでの本題である対話学習の他，人間の思考能力の発達段階，幼児期の自己中心言語の役割，失語症の多種多様な症状，デルファイ法による多数意見への誘導，などの分析に適用した。

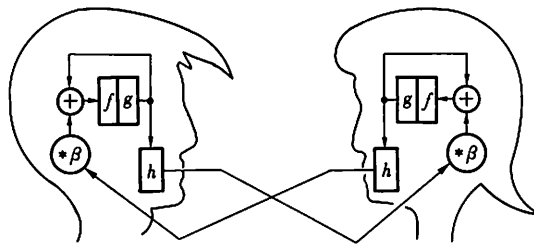

図 12.2　RAC モデルを用いた対話学習のシミュレーション

インテリクローンという観点では，まだ十分な成果を得ていないが，脱プログラミングの 1 つの方向として学習による自己組織化機能は魅力的である。粗忽長屋という落語に，粗忽者が行き倒れの自分を引き取りにいって，「かつがれている奴は確かに俺だが，かついでいる俺はいったい誰だ？」と悩む場面があったが，インテリクローンのユーザが，「考えているこのシステムは確かに俺だが，考えている俺を考えている俺はいったい誰だ？」などと悩む時代の到来を期待したい。