経時的に変化する製品の品質

大　場　　充

（受付　2016 年 10 月 31 日）

1.　序：産業化社会における製品と品質

1.1　製品と効用

20世紀の前半は，産業化社会の時代[1]であった［ドラッカー 1993］。その時代の経済は，資本を集約し，生産を集中させ，自由経済とは言うものの，供給でみると寡占状態が一般的であった。巨大な工場での非常に多数の非熟練労働者の分業による大量生産と，高度に発達した輸送システムによる効率的な製品の配送，巨大店舗やインターネット通販を活用したシステムによる大量生産・大量消費が経済の核心であった。

この大量消費を支えるための大量生産を効率的に実施するためには，巨大な資本と大量の労働力を集中的に投下することが必要である。そのための資本の豊富な蓄積があり，質の高い，均質な労働力が豊富な西欧先進諸国が有利であった。20世紀を通して，先進諸国は年率２パーセント以上で経済成長した。その結果，それらの国々には多くの非熟練労働者から構成される中産階級が生まれた。この中産階級に属する人々は，労働力を供給する供給源であると同時に，工場で生産された製品を消費する最終需要者でもあった。この労働力の供給源が，同時に市場に供給された製品の最終需要者であることが，中産階級の多い先進諸国が経済発展を成し遂げた理由である。

中産階級に属している最終消費者は，自分たちが提供した労働の対価として得た給与所得を，なるべく合理的に配分して自分たちに適した生活を実現しようとする。このため，購入する製品やサービスの効用に対しては，厳しい評価をする傾向にある。この「効用」[2]とは，消費者としてある財（製品）またはサービスを購入するために支払うコストに対して，購入した財またはサービスから得られる満足感との均衡で評価されるものである。その満足感は絶対的な評価ではなく，消費者が購入した財またはサービスから得た満足感と，実際には購

1）　ドラッカーは，生産性革命とマネジメント革命による，産業革命後の資本主義の発展段階に到達した社会を産業化社会（industrial society）としている。また，サローは，第２次産業革命の時代の社会を産業化社会とした。

2）　産業革命以前にイギリスの哲学者ロックが定義し，後にアダム・スミスが国富論において採用した財の価値を表現する概念である。

入しなかったが購入時に検討した代替財または代替サービスから得られたと想定できる効用の期待値との相対的な比較によって決定される［大場，ソフトウェア技術者 2014］[3)]。

競争的な市場における財やサービスの売れ方は，その時点での当該財またはサービスの消費者による評価を反映したものである。つまり，よく売れる製品やサービスほど，購入する消費者の当該製品またはサービスに対する効用の期待値が高い。この効用の期待値を事前評価するとき，一般の消費者は当該製品やサービスを説明した販売促進用カタログを参照する。また，製品の場合，一般の消費者はカタログの写真や店頭で見ることができる現物のデザインから受ける印象も考慮する。一部の消費者は，当該製品やサービスに対する他の購入者の評価なども参考にする。

1.2　効用と品質

消費者が製品やサービスの「効用の期待値」を決定するために利用する情報は，消費者が当該製品またはサービスの事前評価を実施する時点での，当該製品またはサービスの品質に関する客観的または主観的で静的[4)] な情報に限定される。従って，その消費者が当該製品またはサービスの購入後に，長期に渡り供給側の企業から提供される可能性のある有償・無償のさまざまなサービスも考慮に入れた動的で「通時的」[5)] な効用をも考慮に入れた評価ではない。結果として，消費者の購入した財やサービスの効用に対する評価は，時間とともに変化する。

従来型の「製品」の場合，供給企業が販売後に，特にその購入者に対して提供可能な関連サービス[6)] は極めて限定されている。一般的には修理と「リコール」による製品の回収・修理または問題のない新しい製品との交換や，質問に対する回答のみである。さらに，消費者がその製品の使用中に健康を害するなどの損失問題が生じた場合などには，その損失を補償するなどの対応も考えられる。極端な例であるが，購入した製品の利用中に，その製品に隠れていた問題が原因で利用者の生命が失われた場合，その製造物責任に対する損害賠償問題も発生する。

3）1.3節における解説を参照されたい。そこでは，「質」の概念の歴史と，製品の質に関する歴史的な議論がまとめられている。

4）ここでは，時間を捨象し，その時点における限定的で確定的な評価を意味する。その時間の前後における評価の変化や，それまでの歴史的文脈などは無視した評価の方法を言う。

5）通時的と言う表現は，コセリウ著「言語変化という問題」［コセリウ 1973］における用語である。この書においてコセリウは，言語学における言語の変化を見るとき，歴史的に変化する言語の変化に注目した見方を通時的と表現し，歴史的に変化する言語の中に見ることができる普遍的な部分に注目する見方を共時的と表現している。同書において，コセリウは「静的」と「動的」と言う用語も用いている。これらの用語をコセリウは，研究対象である対象言語体系の性質を表現するために用いている。これに対して，「通時的」は，対象の見方を表現する言葉として用いている。

6）購買者が購入した製品の保守サービスなどを意味する。

従来型の製品における品質とは，消費者がその製品を購入した時点で，購入した製品に残存している欠陥問題がなく，利用者が購入した製品の利用によって，その製品に期待されている効用が得られる確率である[7)]。この定義は，古くからある伝統的な製品にも当てはまるが，新しいソフトウェアにも適合する定義である。特にソフトウェアの場合は，プログラム中に残存する欠陥がなく，そのソフトウェアの機能仕様に定義されている機能が実行され，ユーザが期待した出力が，期待されている時間内に得られる確率を言う。

従来型の製品の場合には，出荷した製品に潜在的な設計問題がある場合，それが特定のユーザによって発見されると，その問題を分析して，その原因となった設計の誤りを発見し，その設計の誤りを修正し，設計に誤りがない新しい設計を完成する。さらに，その新しい設計を反映した部品を工場で生産し，ユーザが既に購入し利用している製品を回収し，問題となった部品を新しい部品と交換する。つまり問題の発見から製品の改良までには，かなりの時間（期間）と費用を必要とする。このため，従来型のハードウェア製品[8)] では，「リコール」と呼ばれる制度を利用し，購入者が販売店まで購入した製品を持ち込むなどして，部品交換に対応する方法が採られてきた[9)]。

リコール方式を採用した場合でも，購入者が製品を利用場所から購入店舗まで運搬し，購入店舗または工場で問題の部品を交換し，再び購入店舗で製品を受け取って，それを普段の利用場所まで運搬して利用可能な状態にまで戻す間の時間，利用者は購入した製品を利用できない。このため，購入者が製品に期待した効用は得られないことになる。このような問題が生じた場合，製品の品質評価は著しく低下する。

7） これは，ソフトウェア品質を規定した ISO/IEC 9126の「信頼性」の定義による。同標準規格では，ソフトウェアの品質を説明する特性として，機能性，信頼性，使用性，経済性，保守性，移植性を用いている。このようにソフトウェアの品質を詳細に分析し，体系的に整理する研究としては，マッコールのソフトウェア品質に関する研究が有名である。マッコールは，ソフトウェア品質を11の性質で説明した。

8） ここでハードウェア製品とは，製品が「独立した 1 つの物理的な実体として提供されるもの」を指す。必ずしもコンピュータのハードウェアを表現するものではない。

9） トヨタ自動車が販売したプリウスの ABS が，設計上の問題で，「ブレーキが作動しない」との問題指摘がユーザによってなされ，その後，リコールに至った事例などがある。実際には，ABS が動作しなかったわけではなく，運転者がブレーキペダルを踏んだ瞬間に，それをセンサで検知し，その入力をソフトウェアが処理し，運転者が確実に車を止めようとしていると判断したとき，ABS を動作させる命令を出していたため，ペダルを踏んだ時刻と，ABS が動作し始める時刻との間に時間差が生じたことが原因であった。この時間差があることは，開発後の試験で確認されていたが，開発者たちはユーザのこのブレーキ作動の遅れを問題視するとは考えなかったと，報告されている。

2. 情報化社会におけるソフトウェアとその品質の議論

2.1 ソフトウェアの歴史と開発の問題

1960年頃から商用のコンピュータが普及し始め[10]，企業における会計処理や，在庫管理，生産計画（管理），給与計算などの処理が，それまでのような人手による計算・処理から，コンピュータを使った高速処理に変わった。これらの計算は，ユーザが開発したプログラムによって実行されていた。計算の内容は比較的単純なもので，決まりきった計算・処理を自動化したものであった。とは言え，それまで多数の事務員を雇用して行っていた作業を，コンピュータで置きかえたことにより，事務作業の生産性向上は著しかった。先進国社会は，情報化社会へと移行し始めた。

コンピュータを導入した企業では，事務処理の合理化のため，それまでは人手で実施してきた事務処理を，コンピュータプログラムに置き換える必要があった。人手で実施されていた事務処理は，大枠ではどの企業でも似たようなことを行っていた。しかし，細部ではそれぞれの企業がそれぞれに適した方法を工夫していた。このため，全ての企業が全く同じプログラムを利用することはできなかった。この問題を解決するため，人間が簡単にプログラムを作成することを目的として，高級プログラミング言語[11] が開発された。そして，その言語で書かれたプログラムをコンピュータが実行可能な機械語に翻訳するコンパイラも開発された[12]。

複数種類のコンピュータで稼働可能なプログラムを，同じ高級プログラミング言語のプログラムから生成し，複数種類のコンピュータ上で稼働させるため，プログラムへのデータの入出力の方法を標準化することが必要になった。このことを目的としてオペレーティングシステムが，コンピュータメーカによって開発され，提供されるようになった[13]。この高級プ

10） 特に，1964年に米国市場へ投入されたIBMのS/360によって，多くの企業が事務処理のコンピュータ化を進めるようになった。

11） コンピュータで直接実行が可能な機械語命令は，16進数の並びで書かなければならないので，人間が直接記述することに向いていない。この問題を解決するため，英語に似た表現で，簡単に計算処理の内容を記述できる人工言語が開発された。初期の高級言語としては機械語の命令を英語化して表現したアセンブリ言語，科学技術計算用のFORTRAN，そして事務計算用のCOBOLなどが知られている。

12） IBMによるCOBOL言語の定義とそのコンパイラ（翻訳プログラム）の開発が有名である。COBOLは，事務処理用のプログラムを容易に作成できるように開発された言語で，そのコンパイラも同時に開発された。この言語で書かれた事務処理プログラムは，そのプログラムを稼働させるコンピュータ用に開発されているコンパイラで機械語に翻訳し，実行させることが可能になるように作られていた。

13） IBMによるOS/360の開発が知られている［ブルックス 1982］。OS/360では，複数の処理プログラムを連結して1つの仕事を実行させるジョブ管理，複数の処理プログラムを高速に実行する ↗

ログラミング言語とコンパイラ，そしてオペレーティングシステムなどが導入されたことで，複数のプログラムを連続的につなぎ合わせて，1つの事務処理を完結させる方法が考案された。個々のプログラムではなく，一連のプログラムを全体として考える必要性が生じたことから，ハードウェアにあい対する用語として「ソフトウェア」と言う用語が生みだされた。

ソフトウェアを開発するためには，どのような目的で，どのような機能を実現するのかの仕様を決める必要がある。そして，その仕様をプログラムとして実現するための設計を行わなければならない。設計の妥当性が確認[14]されると，設計に基づいてプログラムを作成する作業が実施される。この作業を「コード化」と呼んだ。コード化が完了すると，プログラムが作成者の意図通りに稼働するかどうかを確認する。それが完了すると2つ以上のプログラムを連結して，仕様に記述された機能が本当に実現されたことを確認する。これを「テスト」と呼んだ。このテストは，連結するプログラムの数を少しずつ増やしてゆき，最終的には全てのプログラムを連結して，最初に決めた仕様通りの機能が実現されていることを確認する。この確認作業が完了すると，ソフトウェアは利用者に引き渡される［大場，ソフトウェアの開発技術 1988］[15]。製品としてのソフトウェアの納入である。

従来型の製品と同じように，「もの」として納入されたソフトウェアにも，テストでは発見できなかった誤りが残存している。ソフトウェアの記述量が多ければ多いほど，残存する誤りの数は多くなる［大場，ソフトウェアプロジェクト実績データの収集と分析 1993］［Jones 1982］[16]。そのような残存している誤りは，プログラムが実際のデータを処理するときに，誤動作として表面化する［Ohba 1984］[17]。例えば，計算結果の誤りなどである。人間が計算結果を見たとき，想定していた計算結果と違う結果が得られると，人間はその計算結果が正しいかどうかを実際に自分の計算で確かめる。そしてプログラムに誤りが残っていたことを

↘ ために同時並行して計算処理を実行させるためのタスク管理，外部記憶装置である磁気テープ装置や磁気ディスク装置への入出力を簡単に実行するためのファイル管理，そして主記憶装置を効率的に利用するためのメモリ管理などを実行する管理プログラム群から構成されていた。これによって，類似のコンピュータであれば，コンピュータハードウェアを変えても，ソフトウェアや外部記憶メディアを変えずにそのまま稼働環境を変えることが可能になった。

14） 設計の妥当性を確認するために実施される作業を「レビュー」と呼ぶ。レビューでは，設計を担当した技術者とは異なる技術者（一人または複数名）がその作業の実施を命じられ，設計を精査し，その設計に問題がないことを第三者として検証する。

15） 2.1節のウォーターフォールパラダイムを参照されたい。この説明で，諸工程の定義，各工程の入出力，問題点が説明されている。

16） 一般的に新規に作成されたプログラムには，記述行で1,000行当り，1から2件の誤りが残存すると言われている。この残存数は，テストへの投入工数が増えると少なくなり，投入工数が減ると多くなる。さらに，設計工程でのレビューに投入する工数にも反比例することが知られている。

17） ソフトウェアの利用時間と誤り発見の関係では，利用時間が長くなればなるほど，残存する誤りを多く発見できることが知られている。この誤り発見過程は，ソフトウェアに残存している誤りを，一定時間間隔で発見できる確率が一定とする時，残存数が多いときほど発見数が多くなる。従って，残存数が少なくなってくると，誤りの発見時間間隔は長くなる。

認識する。

2.2　ソフトウェアの保守

プログラムから得られた出力結果の誤りなどが報告されると，ソフトウェアを開発したグループの技術者たちは，なぜそのような処理誤りが発生したのかを分析する。入力されたデータから誤った結果が計算されるまでの過程を追跡して，プログラムの間違いを探す。こうしてプログラムの間違いが発見されると，正しい（人間が期待している計算結果を得る）計算を実行させるため，プログラムをどう変更すべきかを考え，新しいプログラムを設計する。そして，その新しい設計に基づいてプログラムを書き直し，コンパイルを実行して，新しい機械語プログラムを生成し，その動作が期待しているものと同じことを確認する。この一連の作業を実施することを，「ソフトウェアの保守」と呼ぶ。

多くのユーザが使用しているソフトウェア（製品）の場合，このような問題がユーザから報告されると，開発者たちはその問題が発生した状況に関する情報を収集し，問題発生の条件を絞り込む。そして，プログラムに潜在している誤りを見つけ出し，その誤りを取り除くための修正を設計する。プログラムを修正して，修正したプログラムを，できる限り多くの購入者・利用者に配布し，同じ問題が発生する確率を最小にするよう努力する。このソフトウェアの誤りを修正するための保守作業は，特に近年のように通信回線の利用が容易になってくると，短期間に完了させることも可能である。問題の発見から修正が適用された新しいプログラムの導入までの時間を，数分から，数日程度の時間間隔まで，短縮することも可能である。

このように問題の発見から修正プログラムの導入までの時間を最小にすることで，ほとんどのユーザは，最初の導入の時点ではソフトウェアに残存していた誤りに遭遇することなく，ソフトウェアを利用することが可能となる。このような保守方法を実現したことで，多くのユーザにとって，そのソフトウェアの狭い意味での品質，つまりその故障発生の確率をほぼゼロにまで低減し，品質を高めることができる。この考え方を実践するためには，ソフトウェアの利用者数を可能な限り多数にすることが重要となる[18)]。

特に，現在利用されている多くのソフトウェアは，インターネットなどの通信回線を介して，ユーザが入力したデータを，実時間で処理する機能を提供するものが多い。そのようなソフトウェアの場合，ソフトウェアに残存するプログラムの誤りがあると，その機能の実行

18）特定のソフトウェアに関して，その利用者数が増加すれば増加するほど利用の仕方の多様性が増加する。これによって，プログラムの新しい，それまで実行されたことのない部分が実行される可能性が高まる。したがって，当初ソフトウェアに潜在していた誤りを実行する確率が高くなり，新しい誤りを発見できる可能性も高まる。

が不可能となる。その処理停止が長時間継続すると，ソフトウェアを利用しているユーザへの影響は多大となる。そのソフトウェアが，銀行業務などの国民生活に直結する機能を実行するものである場合，処理の一時停止は，その社会や経済へ悪影響を与える。

ソフトウェア品質の低さは，社会的な問題を引き起こすリスクである。オンライン銀行業務システムの場合，そのソフトウェアのプログラム規模は100万行を超える。100万行を超えるプログラムでは，開発完了直後にプログラム全体に残存する誤りの総数は，1,000件を超える。その誤りのうちの半数の500件程度は，ソフトウェアの使用開始後，短期間（例えば2年）のうちに発見され，修正され，除去されてゆく。しかし，残りの半数の500件程度は，長期に渡り，プログラムに残存することになる。さらに，短期間のうちに発見された誤りを除去するために設計された修正も完全ではなく，そのいくつかには別の新しい誤りが混入する［C. Ohba 1989］[19)]。つまり，全く新しく開発されたソフトウェアでも，全ての誤りを取り除くためには，きわめて長期間を必要とする。残存している誤りの総数を当初の1,000件の100分の 1 の10件にするまでに必要な期間は，利用者数が十分に大きな場合でも10年から15年である。

ソフトウェアの開発が完了し，その実使用が始まると，機能の変更や新しい機能の追加について，様々な利用者から多様な意見・要請が寄せられる。このため，開発完了後のソフトウェアに残存していた誤りの除去に加え，ユーザから寄せられる機能の変更や新しい機能の追加のためのソフトウェアの設計変更が計画され，実施に移される。この追加的な設計変更の作業は，当初の開発作業における設計作業よりも，困難を伴うことが多い。それは，最初の開発で設計された部分が，追加的な設計変更作業を実施するための制約条件になるからである。制約条件が増えれば増えるほど，設計作業は困難になる。さらに，追加的な設計変更作業を進めるためには，既存部分の設計を理解しなければならず，その既存部分の設計に関する完全な知識を獲得することは，保守作業を担当する専門技術者達にとっても容易ではない。

このような背景から，長期間にわたって利用されるソフトウェアの開発は，1 回の開発で完了するわけではない。一般的には，残存する誤りの除去，機能の変更，新機能の追加などを目的とする保守が，そのソフトウェアが利用される期間の間，継続的に実施される。従って，開発と保守の全体を視野に入れたライフサイクルコストを考えると，当初の開発コストは，全体の10分の 1 を超えることはない。つまり，ソフトウェアの保守に投入されるコストが，圧倒的に多い。このことから，大規模なソフトウェアの開発と維持に関わって投入され

19） このプログラムの保守において，問題を解決するために変更を施したプログラムに，この変更のために新しい問題が混入する例がある。この論文では，実際に実施されたソフトウェア製品開発のデータ分析から，そのような問題の混入率は，約10パーセントになることが示された。つまり，規模の大きなソフトウェアでは，10回の修正作業の結果，1 個の新しい間違いが作られている。この確率は，技術者達が想定していたものより，はるかに大きかった。

る資金を考えた場合，その資金投入を効率化する新しい開発と保守の方法を考案することは産業にとって重要である。

3. 産業化社会とグローバル経済社会の時間

3.1 資本主義の発展と経済のサービス化

1980年代の終わりごろから，特に先進国において，経済のグローバル化が始まった。これには，東西冷戦終焉との関係があった。旧東欧諸国の国家体制が崩壊に向かい，旧東欧諸国から EU 諸国，そして米国への人材の移動が始まった。知的な人材の一部が旧東欧諸国を離れ，先進諸国に移動した。この歴史的な高度に専門化した人材の大規模移動によって，EU 諸国を始めとした先進諸国の知的人材が増加した。その結果，高度な専門知識をもつ人材を活用する産業が成長を遂げる契機が生まれた。1989年にベルリンの壁が取り除かれ，さらに旧東西両ドイツが統一されると，この傾向はさらに加速した。

同じころ，米国社会においてインターネットが普及し始めていた。それまでに普及していた複数の独立した付加価値ネットワーク[20]が相互接続された。電子メールの交換やファイル転送が自由になるとともに，新しく SGML[21]を基にした HTML が提案され，インターネットを活用したマルチメディア情報交換が可能となった。特に，ブラウザ[22]の NetScape の登場は，それを利用した情報発信を誰もが自由に行える基盤を提供した。この新しい技術とインターネットを活用して，オンライン銀行である WellsFargo や，ネットワーク書店である AMAZON などが，サービスを開始した［米国商務省 1999］[23]。

これらの新しい事業への取り組みがきっかけで，1990年代の中ごろから，ベンチャー企業によるインターネットを活用した新事業が，特に米国において次々と誕生した[24]。その新事業の中には，Yahoo や Google のような検索エンジンを利用した検索サービス，eBay のようなインターネットオークションサービスなども出現した。そして，「IT 革命」と言う言葉に

20） VAN（Value Added Network）と呼ばれ，同一ネットワークに接続するユーザ間での電子メールやファイルの交換を可能とするサービスである。

21） 電子文書を作成するためのマークアップ言語で，電子文書の各部分が文書のどのような部分に相当するのかを指定するための制御用言語の国際標準規格が SGML である。ホームページ作成用の HTML は，その一部分を選び出して，ホームページの表示指定を可能にした。

22） ホームページをクライアントコンピュータの画面に表示するために利用されるソフトウェアの総称で，現在では，マイクロソフト社が提供する ie や，オープンソースソフトウェアのファイアフォックスなどがある。

23） 236ページのウェルズ・ファーゴ銀行のオンライン事業，259ページのアマゾン・コムを参照されたい。

24） カリフォルニア州サンノゼ市近辺に誕生した小規模な企業が集中した地域であるシリコンバレーは，有名である。

象徴される新しい経済（New Economy）状況が生まれた。米国商務省は，1998年に電子商取引に関する報告書「デジタル・エコノミー」（The Emerging Digital Economy）を発表した［米国商務省 1999］。この報告書は，1997年までの，米国社会におけるインターネットを活用した電子商取引の発展を振り返り，2000年以降の世界を予測したものであった。

米国の社会においては，企業内事務処理業務にコンピュータを活用する情報化社会の発展が飽和した1980年代の中ごろから経済成長は停滞し始めていた。それに代わって，日本経済の成長は著しかった。大型コンピュータの開発・製造においては，特にLSI製造の分野において，日本メーカが，米国のメーカを凌いでいた。米国IBMにおいても，さらにインテルにおいても，米国メーカは，LSIメモリの生産から撤退し，日本製メモリを利用するようになった。IBMもインテルも，CPUの開発と生産に焦点を絞った。このような状況から，米国内においては，21世紀は日本が世界経済を主導するようになるとの認識が芽生え始めていた［ヴォーゲル 1979］。

米国社会においては，1980年代後半以降，日本経済の成長の原動力に関する研究や，日本企業の経営方法に関する研究が進んだ。その結果，日本企業の特に製品生産現場における全社一体となった品質と生産性向上の取り組みが注目された。日本人の製品開発や生産プロセスの改善に対する熱意が注目された。日本企業の研究に基づく報告として出版されたセンジの「第五の組織原理」（The Fifth Discipline）は，米国企業では企業が過去の経験から，組織として学ぶことがない傾向があったことを指摘した。それが日本との競争で，米国企業の弱点になっているとした［Senge 1990］[25)]［大場充，学習する組織 1997］。

この頃，米国内の識者の間で同時に言われていたことは，日本社会における「変化への対応の遅さ」であった。これは，日本の社会がどちらかと言えばボトムアップ式の意思決定法を基本としており，迅速な意思決定ができないという構造的な問題を持っているという分析に基づいていた。また，日本人の多くが「変化を嫌う」傾向が強いことも，知られていた［リンカーン 2004］。これらは，米国社会でもしばしばみられることではあったが，日本社会では，その傾向が著しかった。このような分析から，一部の経済学者は，米国が日本との経済競争に勝つためには，「意識的に変化を創り出す」ことであると主張し始めていた[26)]。

米国社会において，情報化社会への移行がほぼ完了し，それまでの「産業化社会」（Indus-

25） センジは，組織は生物ではないため，経験から学ぶことをしないが，21世紀の企業組織は過去の経験から学ぶ機構を持たなければ，企業間競争に勝てないとした。この組織学習の理論は，センジによる日本企業の研究成果とされている。米国企業は，組織学習の機能を持っていなかったため，企業内における知識の蓄積は，企業に所属している専門家に依存しているとした。

26） 1991年の9月に米国議会上院で開催されたゴア氏が議長を務めた公聴会においてこのことが議論された。公聴会の主題は，米国のソフトウェア産業の日本のそれに対する優位性を維持するための戦略であった。

trial Society）としての発展が止まったことから，経営学者のドラッカーや経済学者のサローらは，21世紀の経済が従来のような資本だけを中心としたものではなく，むしろ知識を中心としたものになると予測していた［ドラッカー 1993］［サロー 1999］[27]。それは，米国社会においては，外国通貨安・ドル高によって，グローバルな競争においては労働コスト高で，製造業の対外競争力が低下していたからであった。それにも拘らず，特に金融業などの高度な知的サービス分野では，他国の追随を許していなかった。つまり，製造業は，労働コストの低い国々に移転してゆくが，最先端のサービスは，先進諸国に留まると予測した。

3.2　経済のサービス化と時間

米国社会は，1980年代の初頭から世界経済はグローバル化せざるを得ないとの認識から，将来のサービスの貿易に備えた国内体制の整備を進めていた。その一つが，政府によるプロパテント（特許推進）政策であった。従来は，自社における技術的優位性を確保するため，特許による技術の公開を嫌っていた米国企業に対して，米国政府は特許収入によって経営を維持できるようにすべきと推奨した。また，GATT ウルグアイラウンドに代表される世界的な規模での貿易協定によって，従来は輸出が困難であったサービスを，国境を越えて輸出可能とするための枠組みの整備にも力を注いでいた。すなわち，米国方式の世界標準化である。これによって，世界各国にあった自国の金融機関を保護するための法律などを撤廃させ，米国の金融機関でも諸外国の金融市場で業務を実施できるように自由化した。

インターネットの普及，米国企業による技術革新の進展，米国政府による米国制度の世界標準化政策などにより，1990年代の中ごろから，米国経済は，グローバル化，サービス経済化，知識化を進展させた。この新しい経済の枠組みの中で，最も重要な要素は，人材である。米国社会は，海外から絶えず流入する人材，特に優秀な人材を活用することで，技術革新を起こし易い体質を持っている。また，米国社会においては，歴史的に人々は変化を受け入れる傾向がある。金融機関は新しい事業創出案に対して，躊躇なく投資をする風土もあった。このため，シリコンバレーに多数のベンチャー企業が生まれ，大きな技術革新の波が生まれた。

米国社会においては，「リバタリアン」[28] と呼ばれる一種の無政府主義者たちが，「小さな

27）21世紀の社会における経済原理を，ドッラッカーは，ポスト資本主義と名付け，サローは，知識主義と名付けた。どちらも，専門家の知識が富を生み出す源泉になることを予測し，その影響について議論している。

28）社会のメンバーである個人の自由を最大限に認め，自由な競争を可能にすることが，社会の発展においては，最も効率的だとする考え方を言う。従って，政府の介入は，それを阻むため，最小限にすべきだとする。例えば，国家による所得税の徴収は，正しくないとする。所得格差是正は，それが重要だとする人々の自由意思による慈善活動や募金によって，低所得者を救済すればよいとする。経済学者であるフリードマンらの新自由主義はこの考え方に基づいている。

政府」を提唱し，「政府の介入を最小限にし，個々人の自由を最大限に拡大すべき」とする運動を展開していた。経済学者のフリードマンらは，新自由主義と呼ぶ，「制約のない市場において，自由競争の原則に基づく」自由経済の有効性を主張した。この主張が社会に認められ，米国においては所得税の累進性が低減され，さらに政府の規制の少ない市場が確立した。結果として，市場において優位な立場にあった企業は，市場占有率を高め，「独り勝ち」[29] の状態が生まれた。このことから，米国社会においては，国民の間での貧富の格差が拡大した［バートレット 1993］。

世界経済を見ると，米国経済の影響を受けて，世界市場はゆっくりであったが，徐々に自由化が進み，経済のサービス化も進み始めた。「独り勝ち」を推奨する新自由主義経済では，巨大な米国市場を制した企業が，世界市場をも制する可能性が高く，有利である。例えば，パーソナルコンピュータ用の基本ソフトウェアを開発・販売している米国のマイクロソフト社は，米国市場を席巻した後，世界市場をも席巻した。

グローバル化，サービス化，高知識化が進展する新世界経済システムにおいては，企業が生き残るためにとるべき戦略は，可能な限り素早く新製品・新サービスを開発し，市場に投入し，先行者利益を勝ち取ることである。そして，知的財産権を活用して，その市場での独占的な地位を確立することである。全ての企業は，製品やサービスの開発だけでなく，その提供プロセスにおいても技術革新を起こし，開発した製品やサービスを他社に先駆けて市場に投入すべく努力している[30]。

そのような技術革新の中核に，ソフトウェアがある。新製品や新サービスの開発においても，その提供プロセスの革新においても，コンピュータとソフトウェアが重要な役割を担う。そのためのソフトウェア開発は，現在でも労働集約的[31] な側面があり，労働コストの高い先進国で実施した場合，開発に投入する資本は多大になる。投入した開発資金を確実に回収するためにも，可能な限り早期に，他社に先んじて新製品や新サービスを市場に投入すること，または新提供プロセスを導入することが企業経営にとって，死活問題になる。新しい資本主義にとって，時間はますます重要になった。

29） 英語では “Winner takes all” と表現され，勝ち残った者が全てを取って行くことを言う。

30） ソフトウェア開発手法として最近，話題になっているアジャイル開発は，そのような目的で提案された実践方法である。

31） ソフトウェア開発とその保守は，知識集約型の労働によって支えられている。しかし，その労働を提供する数多くの技術者を必要とする意味で，労働集約型の産業とも言える。ただし，その労働力は，20世紀型の肉体労働のための労働力でなく，設計やプログラミング，そして試験のための高度に知的な労働力である。

4. ソフトウェアはつねに変化する

4.1 ソフトウェアの非決定性

1987年にブルックスが述べたように，ソフトウェアは，物理的な実体がなく，容易に変えることが可能で，絶対的な制約をもたない，人工的な製作物である［Brooks, No Silver Bullet 1987］[32)]。これは，数学の定理や，法学が対象とする法律に似ている。しかし，ソフトウェアを作成するのは，高等専門教育を受けた少数の学者ではなく，高等教育を受けているとは言え，多数の技術者達である。前述したように，ソフトウェアの記述量が膨大になれば，数学の定理や法律と同じように，人間が犯す錯誤が原因での誤りは混入する［大場，ソフトウェアプロジェクト実績データの収集と分析 1993］[33)]。似たようなことは，コンピュータハードウェアの開発でも生じることである。しかし，実体がなく，容易に変えられるソフトウェアでは一層起きやすい。

物理的な実体がなく，絶対的な制約がないことは，製作のための自由度が高いことを意味する。しかし，それは自由度が高いために，その設計や実現が正しいことを確認することが容易でないことも同時に意味している。設計者やプログラマが何を，どのように考えたかを理解し，その理解に誤りがないことを，本人ではない第三者が確認・検証しなければならない。物理的な実体がある製品の場合，その設計に基づいた試作品を実現して，動作させることで，かなりの正確さで設計に本質的な間違いがないことを，第三者が確認できる。これが，従来の製品テスト（製品試験）である。しかし，ソフトウェアの場合，この方法は有効には機能しない。それは，「非決定性」と言う理論的問題を孕んでいるからである［大場充 2014］［大場，ソフトウェア技術者 2014］[34)]。

32） この論文のタイトルの “silver bullet” は，中世ヨーロッパにおいて，社会を混乱に陥れた悪魔の手先とされた「狼男」を殺すために使った銀の弾丸を指している。これは，神父が胸にかけている銀の十字架を溶かして作成した鉄砲の弾である。銀の弾丸を使うことで，狼男を殺しても悪魔からの仕返しを免れると信じられていたのである。このことから派生して，英語での表現としては，「たった一つのことで世の中を変えるもの」を意味している。そのような銀の弾丸は，ソフトウェア開発には存在しないことをブルックスは主張した。この論文の最初の部分で，ブルックスは，ソフトウェア開発がなぜ困難なのかの理由を分析した。

33） 注16）を参照されたい。

34） 組込みソフトウェア工学第 1 章1.5.2節に副作用を利用した計算過程と非決定性の問題が議論されている。また，ソフトウェア技術者第 2 章2.5節において非決定性がもたらす社会的リスクの問題が議論されている。変数への値の代入操作を使った値の更新は，記憶装置の有効活用が可能になるが，更新後に更新前の値を参照することはできない。この数学的ではない操作によって，計算過程の不可逆性が生じる。この不可逆性のために似たような計算過程であるにもかかわらず，ある変数が保持している値が影響して，実際の計算が変化するため，異なる計算結果を得ることがある。これを非決定性と言う。

非決定性とは，コンピュータが実行する計算を数学的に考えたチューリングが考え出した概念である。チューリングは，現実のコンピュータを抽象化した仮想の機械であるチューリング機械の上で実行されるプログラムが，その無限の記憶域に記憶されている計算過程の情報を，実行する計算手順（部分プログラム）の選択のために参照可能であることを示した。このことで，実際に実行される計算手順が動的に変わる。完全に同じ入力値に対しても，過去の計算過程の影響を受けて，異なる計算結果を出力することもある。つまり，入力されたデータだけでは，単純に出力結果を決められない。これを「非決定性」と呼んだ。

この非決定性の導入は，従来のような物理系として実現されるモノや製品が，入力されるデータや，実行時における外的な諸条件だけで出力を決定するという，単純化された機能の束縛からの解放を可能にした。つまり，プログラムは，自分自身が実行した過去の計算を記憶しておき，入力データや実行時における外的な諸条件に加えて，過去の計算結果も参考にする計算が可能になった。これによって，コンピュータソフトウェアでは，「学習」による計算手順の調整や変更も可能になる。特に，LSI 技術の発展によって，コンピュータが小型化され，さらに低価格化が進んだため，多くの製品にマイクロプロセッサが組込まれた。その制御によって機能を実現できるようになったため，様々な製品に学習機能が付加できるようになった。

4.2　組込みソフトウェアと製品開発の変革

このように組込みソフトウェア[35)] を活用した製品開発の場合，従来の物理系として機能の実装を行っていた製品では，ほとんど不可能[36)] であった学習機能などを，低コストで実装することが可能になった。また，従来の物理系として実現されている製品の生産では，複雑な構造で精密な加工が必要なため，熟練作業者が生産工程で作業を実施しなければならなかった。結果として生産コストが高くなった。マイクロプロセッサと組込みソフトウェアを活用することで，機能の実装を物理系から組込みソフトウェアと単純な物理的構造に置きかえることが可能になった。製品の構造を単純化できたため，熟練作業者による作業の必要性は低

35）組込みソフトウェアとは，従来は 1 つの完結した物理系として実現されていたシステムを，外部からの情報を受け取るセンサ，外部の情報に基づいてシステムの応答を外部に出力するアクチュエータ，そしてセンサから入力された情報を処理して応答を計算し，アクチュエータに出力するコンピュータの 3 つの機能部品に分割し，それらを統合することで，従来のシステムと同様の機能を実現する目的で，制御用コンピュータ上で稼働するソフトウェアを言う。

36）機械式機構や電気式機構を応用して製品を構成している場合，その製品の過去の動作を決定していた入力の値を保持しておくことは困難である。そのため，過去の製品開発では学習機能を実装することができなかった。従来型の製品で唯一，可能だったものが，動作を遅らせる機能である。これも，記憶の一種である。現在の値と少し前の時間の値との差を計算して，その差に比例した分だけ出力を変える方法である。ドアをゆっくりと開ける機構などに利用されていた。

減した。結果として多くの製品を，労働コストの低い地域の工場で生産することが可能になった。

従来の物理系の動作として実現されていた機能の確認を目的としたテストは，非決定性をもつ組込みソフトウェアの活用によって，不十分なものとなった。従来の製品の場合，製品の動作を確認するために必要な条件の組合せは限定されていた[37)]。しかし，組込みソフトウェアを利用した製品の場合，その動作を確認するために必要な条件の組合せは，入力データと実行時点での外的条件だけでなく，記憶装置内部に残されている各変数の値の種類[38)]の組み合わせも考慮したものでなければならない。このような理由から，テストで確認しなければならない条件の組合せは，無限に近いものとなり，実質的に十分なテストの実施は不可能となった。

このような非決定性の問題が原因で，ソフトウェアのテストでは，ソフトウェアの機能を完全に検証することは実質的に不可能となった。このため，開発者たちは，テストで確認すべき条件に優先順位を割当て，優先度の高い順にテストで確認を実施し，テスト期間を満了するまでテストを継続する。テスト期間を満了した時点でも実施できていない確認項目は，未確認項目として残し，テストを終了する。このテストでの確認が完了していない条件の組合せを，実利用時のユーザが実行した場合，プログラムに隠れていた誤りが実行され，問題が表面化する[39)]。

一般的に隠れていた誤りが実利用で実行される確率は小さい。そのため，ユーザがそのような誤りによって不利益を被るリスクは限定されている。そのような隠れた誤りが実行されて，ユーザが不利益を被る事態が発生した場合，ユーザはそのことを開発者たちに報告する。開発者たちはその時点で問題を認知し，その原因である隠れた誤りを分析し，除去するため

37） これは，従来の製品では，入力から直接的に出力が決められる機能を実現していたため，入力を分類し，その組み合わせをすべて試験することで，完全な確認が可能であった。また，入力と出力が直接的に関係していたため，同じ入力に対しては，必ず同じ出力が対応していたためである。しかし，ソフトウェアの場合，出力値を決定するためには，全ての変数の値が関係する可能性がある。その変数のいくつかは，現在の入力値ではなく，過去の入力値や過去の出力値が関係する場合がある。従って，出力を決定するためには，極端な場合，プログラムで使うすべての変数を同じ値にしなければ，同じ出力値を得ることはできない。そのために，過去にさかのぼって，全ての入力を一致させなければならなくなる。従って，試験で確認すべき組合せが膨大になるからである。

38） 計算機による計算処理では，ある変数に格納されている値が特定の範囲にあるとき，特定の計算結果を出力する。そのため，変数に格納されている値の範囲の種類の数によって出力の値が変化すると言える。その「値の範囲」の数が問題になる。

39） 2016年に米国内で実証試験中の自動運転自動車で発生している問題の多くは，このような不十分な開発テストが原因で発生したものである。これに対して，プリウスでリコールとなったABSの問題は，社内で実施された開発テストでも確認されていたが，開発者たちが，「これをユーザが問題にすることはないだろう」と，低い優先度に評価したため，市販の製品に問題として残った事例である。このような優先度の評価を誤る問題は，従来型の製品でも起こっていた問題である。

の対応，回避するための対応を考える。そして，そのような誤りを除去して正しく機能するようにしたプログラムにする。潜在している誤りを除去できない場合でも，その誤りによる問題の発生を回避する処理を追加する。このような対応で，ユーザに被害が及ばないようにする対策を講じたプログラムに修正し，より良い品質のソフトウェアに作り替えてゆく。

経済のサービス化の進展で，新しい機能を一定の時間的制約の中で提供することが求められる。製品の開発者たちに可能なことは，与えられた時間の中で製品を開発し，顧客に提供することである。そして，問題が報告されたときは，素早くその問題を解決する修正を行うことである。従来型の設計変更に時間とコストがかかる製品であれば，より完全な状態にある製品にまで仕上げて，顧客に提供することが重要であった。しかし現在，多くの製品の多くの機能においては，顧客が要求している時期までに製品を完成させ，残った問題に対しては速やかな対応で保守を実施することが求められている。その意味で，ソフトウェアは，時事刻々と変化するものになっている[40)]。

5. 製品の本質，その変化

5.1 製品とその経時的変化

製品とは，特定ユーザの要求，または不特定多数のユーザが持っていると考えられる要求を満たす機能を提供するために，1つの自己完結した物体として，ユーザが入手（購入）可能な実体を言う。ただし，その製品が提供する機能は，ユーザが入手した「自己完結した実体」のみで提供可能とは限らない。つまり，何らかの方法で通信を実行し，別の場所にあるコンピュータシステム等と交信することで，データを交換し，その結果に基づいて機能が実現される場合もある。電話機やテレビは，そのようなオンライン型製品の典型である[41)]。

従来，製品は，1つの物理的な実体として設計・製造されていた。オンライン型の機器の場合でも，通信を実行する部品と，データを送信したり，受信したデータに基づいて処理を実行する部品など，いくつかの機能部品に分割される。それらが個別に設計・製造され，さらに最後に1つの製品に統合される。これは，最近の製品のようにコンピュータが搭載され，それを動作させる組込みソフトウェアの制御で機能を実行する形式の製品でも，同じである。

40） 例えば，マイクロソフト社が供給しているWindowsなどでは，インターネットを介してユーザのコンピュータ上で発生した問題のデータと，その問題が発生した前後でのソフトウェアの内部情報（変数の値）をマイクロソフト社へ送信させ，同社内で解析を実施して，隠れていた問題が発生したと判明したときにはその解決策を作り，問題のプログラムを修正して，自動的にユーザのコンピュータに送り込み，他のユーザでの問題の発生を事前に防止している。

41） インターネット通信を介したコンピュータの処理（情報検索，ネット通販など）や，アプリを使ったスマートフォンの処理もこの例である。

その制御部分が，コンピュータ部品として設計・製造されているに過ぎない。

歴史的に振り返ると，産業革命以前の世界では，市場で売られていたもの（商品）は，一人または数名の職人によって作られていた。したがって，簡単な分業もされていた。刃物であれば，刃の部分，柄の部分，さやの部分を，それぞれ専門の職人が担当するやり方もあった。もちろん，全ての作業を一人の職人がやっていた場合もあった。ただし，刃物を作る職人が，刃物の原料である鉄（鋼）を作ることはなかった。鉄の原料（例えば鉄鉱石）から鉄を作るのは，鉄を作る職人の仕事であった。そうやって作られた粗鋼から，刃物の材料になる鋼を創り出す鍛冶の仕事も，鍛冶職人が担当していた。鍛冶職人が作った刃の素材を削り出して，彫刻を施し，刃物の刃を作ったのである。

このような原始的なものの生産においては，商品の全体像を決め，設計図のようなものを描き，それぞれの部分を担当する職人に指示を出していた親方の役割を担っていた人がいたはずである。個々の部分を担当する職人も，全体をまとめる親方も，先輩の指導の下で長年の訓練を受け，さらに長期間の補助作業経験を経て，一人前になった。ヨーロッパの社会では，早くからギルド制が確立したため，職人や親方は，ギルドの構成員として認められて初めて一人前になった。そして，長期間にわたる職業訓練と補助作業員としての経験を経て，「どうやってものを作るのか」を学んだ。

この先輩から後輩への技術の伝承は，全てが現場での実践経験に基づく伝承であったため，その方法を変えることには，失敗のリスクを覚悟しなければならず，作業方法の変更は容易ではなかった。このような理由から，モノづくりの方法は，長期間にわたって同じ方法が繰り返えされる環境であった。その結果として生産される商品の原型は，きわめてゆっくりとしか変化しなかった。それでも，商品の表面的な形状などは，商品を購入する依頼者の意向などを受け，歴史的には少しずつ変化していた。しかし，作り出すものが果たすべき機能について言えば，その変化はほとんどなかった。

5.2　機械式計時機構と時計製品の経時的変化

中世の教会の塔に取り付けられた大型の時計から，城の大広間に置かれる柱時計が発明され，さらにそれを小型化した置時計が作られた。このように，数百年の単位で見てゆくと，基本的なものの機能に変化はなくても，その利用目的の変化に応じて，商品の形状も作り方も，時間とともに変化していた。さらに，時計の場合，卓上型の置時計を小型化して，ポケットに入れて持ち歩ける懐中時計が18世紀になって作られた。これによって，個人個人が必要な時に，いつでもどこでも時間を知ることが可能になった[42)]。そして，20世紀になると，懐

42）　この懐中時計による時間の個人化が，18世紀に始まった資本主義の発展に大きく貢献した。

中時計はさらに小型化され，腕時計にまで進化した。しかし，教会の大時計から腕時計に至るまで，機械式の計時機構という本質的な部分は変わらなかった。

この歴史的な機械式時計の変化を見ると，長期的には2つの流れが基本に存在した。その一つは，小型化の潮流である。これは，時間の個人化の要求に応えるためのものであった。もう一つは，低価格化の潮流である。これは，時間の個人化を多くの人々が必要としていて，その要請に応えるためのものであった。時計が一部の富裕層の人々だけの持ち物であった時代，時計の機構は同じでも，その表面的な作りは，多くの職人が参加した，ぜいたくなものであった。文字盤の作り，針の作り，竜頭の作り，機構を守るケースの作り，文字盤と針を守るためのガラスの作りなど，全ての部分に一流の職人が関わったのである。

産業革命後の社会で，多くの人々が時計を所持するようになった。すると，そのような装飾的な部分は簡素化され，工場での分業[43]によって時計が作られるようになった。分業を細分化することで，熟練工の必要性を減らし，時計を安く作ることが追求された[44]。時計の生産は，労働コストの高い先進国から，労働コストの低い開発途上国へと，次々と移転した。フランス北部からイギリス，アメリカ，そして日本へと，小型時計の主要生産地は時代とともに変わっていった。特に，アメリカでは，18世紀に懐中時計の生産革命が起こり，大量生産が可能となった。そして，日本に明治維新が起きていた頃，タイメックス社によって，1ドル時計が生産された[45]。

このタイメックス社による1ドル時計の販売は，社会を大きく変化させた。一定以上の収入がある人々に，携帯時計を所持することが可能になったため，資本主義が大きく発展した。その後，工場での分業における作業の時間を簡単に計測することが可能になった。作業の所要時間を計測して，長い時間を要する作業を細分化して，分業の効率化を進める方法がアメリカで開発された［テイラー 2009］[46]。アメリカ社会は産業化社会へ移行した。

5.3　製品変化の様態とその要因

製品の変化は，時として社会の在り方をも変化させる。逆のことも言える。社会における技術革新の要請は，製品の機能や提供方法を変える可能性がある。ただ，従来型の製品（商品）の場合，新しい要求に対応する製品の設計や製造工程の準備，生産と販売の準備に必要

43）アダム・スミスは，富国論において分業による生産性の向上を議論した。
44）テイラーは，全ての熟練労働は，細かく細分化された分業によって除去できると唱えていた。
45）1ドルの価値は，現在の1ドルとは著しく異なる。それは，その後に開発されたシンガー社の機械式家庭用ミシンの通信販売価格が30ドルを少し超える値段であったことから分かる。
46）テイラーは，分業化された作業の実施に必要な時間を計測し，作業者による作業時間の違いを見つけた場合は，それらの作業を比較することで，より短時間に作業を完了できる作業方法を選択することで，作業効率を向上させることができることを示した。

な各種の準備作業の期間と投入する資本が多大であった。そのため，製品の変化を簡単に起こすことはできなかった。たまたま，それらの条件が整っていた場合に，新しい製品の供給が可能となった。そして，その新しい製品の供給によって，社会が変化するという状況が発生した。

このことは，従来型の製品でも，その機能設計や表面的なデザインの変更，生産工程や供給方法の変更が可能であっただけでなく，実際にそのような変化の起こった例が数多く存在したことを示す。特に，高度な工業製品の例である自動車などの場合，見た目のデザインの変更だけでなく，新しいエンジンや駆動系を搭載した新設計への変更，さらに内装の高級化などによる高級志向モデルの追加などやその逆など，さまざまな変更がある。それらの変更を意図的に導入することで，顧客の購買意欲を刺激する方法が採用されていた。つまり，社会における人々の価値観の変化を受けた製品の変更（更新）は，製品マーケティングの一般的な手法として使われてきた。

このような製品の変更の中には，1世代前の製品において，その販売開始後に市場のユーザから寄せられる，不完全な実現が原因で発生する機能的問題[47]の指摘や，ユーザからの改善の要請などを反映した変更などを積極的に取り入れた設計変更の例も存在する。ある企業が売り出した新型車の安全機能に対する顧客の関心が強ければ，他の企業も類似した安全機能を搭載した自動車への設計変更を採用し，企業間競争に勝ち残ろうとする。このようにして，製品は少しずつ，その姿を変えてゆく。とは言え，自動車の設計変更には，短くても1年程度の期間を要している。これは，自動車が1個の独立した物理系として実体化され，大量生産されなければならないという制約のためであった。

利用者の健康や生命に影響を与える可能性のある問題が，製品を市場に提供した後に発見された場合，その製品を提供する企業は，緊急な対応を迫られる。このため，製品の回収やリコールを公表し，実施する。このような対応は，消費者の保護を目的として，多くの先進諸国において法律で定められている。さらに，製品提供企業にとっても，製造物責任を問われる事態に発展する可能性があるため，企業自身を守るための手段として製品回収やリコールを実施する。家庭電器メーカによる，石油ファンヒーターの回収対応は，販売した製品に設計上の問題が隠れていたことが判明し，その対応として実施された[48]。前述したハイブリッ

47）日本の企業においては製品の「不具合」と呼んでいる例も多い。これは，機能的問題が製品の欠陥を意味しており，社内で欠陥の残存を認知していたことを暗に示す言葉の表現を避けるために使われている。

48）この事例は，2005年に発生した4件の石油の不完全燃焼による一酸化酸素中毒事故に対する対応として，当時の松下電器が実施したリコールである。1985年からの7年間の期間に渡り製造された松下電器製石油ファンヒーターの設計上の問題で発生したと考えられた死亡事故をきっかけに実施された。

ド型自動車の ABS に関する不具合の場合，メーカ側はリコール対象として発表し，その自動車に搭載されているマイクロプロセッサを制御する組込みソフトウェアの書換え作業を実施した。

5.4 製品ライフサイクルの変化

新しい「変化する製品」では，そのライフサイクル[49] も変化する。従来型の変化しない製品の場合，ライフサイクルは，企画段階，計画段階，開発段階，試作試験段階，量産段階，保守段階から構成されていた。一般的に製品が保守段階に入ると，次世代の製品企画段階が始まった。しかし，そのような対応では次世代製品の開発から量産開始までの過程が遅くなるため，最近では，製品の開発と，次世代製品の企画が同時に並行して実施されるようになっている。このライフサイクルは，継続的に「変化する製品」の場合，開発段階と保守段階の間の明確な境界が消失し，さらに次世代製品の企画も消失して，現行製品の保守と新製品の企画が混然一体[50] となって実施される。

このことは，従来型の製品開発では，1 つの製品が開発され，市場へ投入された後に，次の世代の製品が開発され，市場へ投入されるという，次世代製品のライフサイクルの最初の部分と旧世代製品のライフサイクルの最後の部分が重なり合っていたことを意味している。しかし，新しい「変化する製品」の場合，開発と保守の境界が消失し，連続的な開発と保守がライフサイクルの最後まで継続することとなる。当然の帰結として，1 つの製品のライフサイクルは長くなる。

この長い製品ライフサイクルを詳細に分析すると，1 つの製品として見ることはできるが，詳細部分では少しずつ違ういくつかの個別製品の小さなライフサイクルの中の開発や保守が，パイプライン[51] 状になって，同時並行的に実施されている。これらのパイプライン状に実施されている複数の開発作業の成果として生み出される製品の小さな改良版の中には，最終的には試作だけに終わり，市場には投入されないものも出現する。これは，生物の進化に似て

49） ライフサイクル（life-cycle）とは，ソフトウェア工学ではソフトウェアの構想から，開発，実使用期間中の保守を経て，ソフトウェアの利用を終えるまでの期間全体を想定し，その各段階で実施される作業に注目して，いつ，何をするのかの大枠に関する知識を整理したものである。一般的に，ライフサイクルを人為的に変更することは，失敗のリスクが高まる。この点が，作業プロセスが作業の方法を意味しており，変更によって生産性向上が可能になる例がある点と異なっている。

50） 旧製品の保守と新製品の企画が，同時に並行して実施されることを意味し，両者が融合した作業に置き換えられることは意味しない。一般的には，異なる人々が，情報交換をしながら，それぞれ独立に作業を進めるのである。

51） パイプラインとは，一連の同じ作業の流れを，複数の実行主体がそれぞれに与えられた課題の解決や使命を全うするため，同時に，それぞれ独立に，並行して実施することを言う。サーフィンで波の下にできた輪状の空間をパイプラインと呼んだことによる。

いる過程で，突然変異によって複数の変異した個体が生み出され，その後の適者生存の原理に基づく自然淘汰の過程を経て，後世に生き残る新種が生み出される過程に類似している。

このような生物の進化に似た過程をたどりながら，「変化する製品」は少しずつ変化（進化）してゆく。そして，社会の要請に最もよく適合した製品の設計だけが選択され，存続してゆく。その製品の発展の歴史を通して通時的に観測を続け，その製品の全体像を共時的[52]に分析すると，1つのまとまった製品としての像が浮かび上がってくる。この像が，製品のブランドに相当するものであると言える。その全体像には，その製品が，「どのような人が，どのような場面で，どのような目的のために，どのように利用するのに適した製品であるか」と言う，利用シーンとの適合性に関係した評価も付随する。

6. 経時的に変化する製品の品質とは

6.1 経時的に変化する製品としてのソフトウェア

これまで議論してきたように，組込みソフトウェアによって機能の本質的な部分を実現している製品でも，完全な物理系として全機能を実現している製品でも，製品が変わることは現代社会の製品の本質である。その変化の要因としては，5.3節で議論したように製品開発の過程で製品設計時に入り込んでしまった「誤り」などがある。また，物理的に完結した実体として提供される製品の場合，生産工程の設計や量産準備段階で混入した人間の誤りなどが原因で発生する問題の解決，製品の出荷後に利用者から寄せられた意見がきっかけで製品の機能を変更したり，追加したりすることを目的として実施される変更などもある。

このように，物理的な実体として提供される製品の場合，外見は同一の製品に見えても，機能や機能以外の面での改善のための変更が行われている例は多い。さらに，利用者の好みに合わせた形状や色彩などを取り入れた，外見上の変更や，利用者が見て，触れてわかるパッケージ素材の変更など，外面的な変更を行うこともある。金属素材のパッケージからプラスチックなどの高分子化合物素材のパッケージへの変更は，製品の軽量化や，形状加工の自由度の高さ，そして低価格化のための手段として実施される。

これに対して，ソフトウェアのような物理的な実体をもたない製品の場合，主たる変更は，製品機能の変更である。ただし，機能の変更とは言え，その製品に本質的な機能の追加や変更の場合もあるが，入出力方法の変更や，ユーザインタフェースの変更など，どちらかと言

52） 共時的と言う表現は，コセリウが変化し続ける言語を，通時的に観測し，そこから時間を捨象した言語の静的な性質や構造を抽象するための見方を言う。ここでは，製品を長期間にわたり観測して，その製品の性質や属性の中で，時間を超越して観測できるものを取り出すための見方を言っている。

えば，本質的ではない表面的な部分を変更する例も少なくない。マイクロソフト社が開発したWindows 3.1は，それまでのMS-DOSの機能と本質的な違いはなかった。しかし，アップル社やSUNマイクロシステムズ社が採用していたウィンド方式を採用し，ユーザインタフェースの改善をした例であった。

このWindows 3.1の後，マイクロソフト社はWindows 95を開発し，市場に提供した。そのWindows 95においては，マルチタスク方式[53]が採用されたため，基本ソフトウェアの機能は大きく変更された。ユーザから見たインタフェースでは，Windows 3.1では，MS-DOSを起動後，Windows 3.1を起動する方式が採用されていた。Windows 95では，最初からWindows 95が起動される方式へと変更された。このため，MS-DOS制御下でのプログラムの実行などは，DOSプロンプトのウィンドを開いて実行する方式に変更された。ただし，ユーザインタフェースの大きな変更はなかった。

このマイクロソフト社におけるパーソナルコンピュータ用基本ソフトウェアの歴史を振り返ると，製品としての基本ソフトウェアは，時間とともに少しずつ変化していたことが分かる。さらに，MS-DOSやWindows 3.1でも，製品を市場に投入した後に発見された問題（誤り）に対応するための修正が，次々と発表され，利用可能であった。つまり，製品は時々刻々と変化していた。

6.2 経時的に変化する製品の品質

このように変化し続ける姿が必然である現在の製品における品質とは何かについて考える。かつての完全に独立した，物理的な実体として提供されていた製品においては，短期間での変化の必然性が小さかった。また，変化が必要な場合，当時，企業側が採用できる方法は，新製品の開発と市場への投入以外には，製品の回収と改良した製品との交換[54]，または代金の返却のみであった。また，従来は製品開発の過程で実施されていた試験も有効に機能していた。市場投入後の製品に設計の誤りなどの問題が発見される例は少なかった。とは言え，製品パッケージの塗装に使われていた塗料の問題で，人体に影響があることが初出荷後，数年の後に判明する例などもあった[55]。

米国の自動車メーカが開発した小型乗用車で，その設計時の燃料タンクの素材選択の誤りと，燃料タンクの搭載位置の選択の誤りが重なり，衝突時に車が炎上するという重大な問題

53） 複数のプログラムが同時並行して実行されているように見えるプログラム実行の管理方式である。中央処理装置（CPU）の処理速度が，入出力装置の実行速度に比較すると，はるかに早いという時間の差を利用して，入出力実行時に他のプログラムを実行されることで，1つまたは数少ない中央処理装置をそれよりも多い数のプログラムが共有して利用しているように見える。

54） 製品の「リコール」対策を意味する。

55） このような事故事例が多いため，国際標準化機構においては，国際規格ISO8124を制定している。

が判明した例があった［サンデル 2010］。この事例では，当初，自動車メーカはその自動車のリコールを実施しなかった。同社では，燃料タンクの材質を変更し，リコールを実施して，燃料タンクの取り換えを行うことも検討したが，結論として多大なコストを必要とすることが判明したため，実施しなかった。この自動車メーカの対応は，そのことが報道によって周知された後，ユーザから非難の的となった[56)]。

物理的な実体として提供される製品の場合，シューハートによる古典的な品質の定義は，『製品がある利用者の利用目的にそった製品の利用で，その製品の仕様に定義された機能を成功裏に実行させ，所期の目的を達成できる確率』となる［大場，ソフトウェア技術者 2014］[57)]。品質に直接かかわる問題は，製品の仕様，利用者による製品利用の目的，そしてその利用目的を達成できる確率である。利用者が製品の仕様に定義されていない機能や方法で，その製品を利用する場合に，利用者の利用目的が達成されるかどうかは，古典的な品質の概念の範疇では問題にならない。

この狭義の品質の定義は，ソフトウェア品質の定義としても，1970年代の中ごろまで，一般的に使われてきた。これに対して，1980年代に入って，新しい，広義の品質の定義が議論されるようになった［大場充 2014］[58)]。それは，ユーザの利用目的だけでなく，利用状況における諸条件も考慮するものである。例えば，ISO/IEC 9126に定義されたソフトウェア品質では，信頼性，使用性，経済性，保守性，移植性と，相互運用性を含む機能性が定義されている。つまり，利用者にとって操作が自然で使いやすいものであるかどうか，利用者が想定していた時間範囲の中で処理を完了できたかなども問題にされている。古典的な狭義の品質は，このISO/IEC 9126の定義では，信頼性の範疇の問題とされる。

従来の品質管理の議論では，ソフトウェアの場合も含めて，品質では，製品の開発が完了し，市場に投入された後に，機能の仕様や要求仕様に対して，製品がそれらの仕様に適合し，利用者の利用目的を達成するために寄与し，さらにその結果が利用者の期待に合致する確率

56） サンデルは，この問題に対するフォード社の経営陣の対応について，フォード社が損害賠償額と，設計変更に伴う製造コストの上昇による販売価格の上昇の影響で，同車の販売が落ち込むことを懸念して，製造物責任保険によってこの問題に対応した態度が，倫理的に問題があったと批判している。実際に，フォード社経営陣のこの対応は，一般消費者のひんしゅくを買ったとされている。

57） ここでは，第 3 章3.2.3節に記述されている統計的品質管理の提唱者であるシューハートの品質の定義を参照した。シューハートは，アダム・スミスが述べた製品の効用が測定不可能であった問題を，正常に仕様通りの機能を実行できる確率と再定義できることを示し，品質を基礎的な統計学を応用して計測可能なものとした。

58） 第 3 章3.1.1節において，ISO/IEC 9126の国際規格で定義された講義の品質について解説している。さらに，ISO/IEC 9126で定義された 6 つの品質特性が，機能性，信頼性，使用性，効率性と，保守性，移植性の 2 つのグループに分割できることを示した。さらに後者の 2 つの品質特性が，現時点でのソフトウェアの品質を問題にしているのではなく，将来において発生しうる問題を先取りしたものであることを述べている。

を，評価時点における静的な全体像として表現することが問題とされてきた[59]。これは，製品を完成したものとして静的に見ることが基本となっている。しかし，前節で議論したように，現代の製品は変化し続けるものであり，一時点の静的な姿[60]で，製品全体の効用を把握することはできない。

製品の本当の姿を動的に把握し，表現しなければ，製品の品質を評価したとは言えないのである[61]。したがって，経時的に変化する製品の品質とは，製品開発段階を経て市場に提供された『製品が，その後の長期に渡る保守段階において，社会の変化に対応し，継続的に新しい利用者の要求を反映し，利用者の期待に応えられる確率』を言う必要がある。

6.3 変化の要因と重要な品質特性

変化する製品の品質を評価する場合に重要なことは，その変化が外的な要因によって起こるものであるのか，または内的な要因によって起こるものであるのかを明確に分離して考えることである。ここで，外的な要因によって発生する変化とは，例えば，i）社会の変化が原因で，利用者の利用目的に新しい目的が追加されることになったこと，ii）利用者の利用方法に変化が生じたこと，iii）利用者が利用目的を達成するために製品機能の利用手順などを変える必要が生じたことなど，製品を利用する利用者側の要請による変化である。

また，内的な要因によって発生する変化とは，例えば，i）製品の設計に誤りがあったことが判明したため，その誤りを除去するために設計を変更しなければならないこと，ii）将来に予想される外的要因による新しい機能の追加や既存機能の変更要請に対応するため製品全体の骨格を決めている基本設計を変更する必要があること，iii）社会の変化に伴って製品と利用者の間に存在するインタフェースを刷新して利用者の利便性を向上させる必要があることなど，長期的な視点で製品そのものの延命のために実施される変更である。

これらの変更は，ソフトウェアの用語では保守や移植[62]と呼ばれていた。1970年代末，

59） シューハートの品質の定義も，ISO/IEC 9126の基礎となったマッコールのソフトウェア品質の定義も基本的に同じ視点に立っている。

60） これは，製品の歴史を見たとき，時間の経過とともに変化してゆく製品を，評価の時点で固定し，スナップ写真のような静止した動きを見ているかのように，品質を評価しようとしていることを言っている。これでは，製品の本当の姿を見ることはできないのである。

61） つまり，製品の隠れた問題が原因で，利用者が利用目的を達成できない例が発生したとしても，その事実だけで品質が悪いとは言えない。その問題を製品供給企業がどれだけの期間で解決し，利用者が利用目的を達成できるようにできるかが問題になる。さらに，利用者が仕様の細部の解釈から，ある目的のためにも利用可能だと考えたことが，実際には不可能な場合，それを新しい要求として提供企業側に要請したとき，それを可能にすべく製品の機能を改善できるかどうかなども品質の評価に影響する。

62） 「移植」（porting）とは，ソフトウェアにおいてその動作環境を変えることを言う。移植性とは，そのような移植がどの程度し易いかの程度を言う。

レーマンは，IBM で開発されたオペレーティングシステムのデータを分析し，ソフトウェアに S 型のソフトウェアと E 型のソフトウェアがあることを指摘した[63]［Lehmann 1980］。当時，S 型ソフトウェアは，組込みソフトウェアのような仕様を厳密に記述することが可能で，その仕様が変化しない種類のソフトウェアを指して言った。これに対して，E 型ソフトウェアは，時間とともに仕様が変化し，その仕様の変化に合わせてソフトウェアそのものが変わってゆくものとした。オペレーティングシステムを E 型ソフトウェアの典型とした。

現代社会の製品は，ソフトウェアで機能を実現するものも，そうでないものも，この E 型の製品である。つまり，長い歴史を通してみれば，具体的な個々の仕様は時間とともに変更してゆき，その実現も仕様の変更に合わせて変えられてゆく。これは，1980年代には，S 型と考えられていた組込みソフトウェアでも1990年代に入って起こった変化であり[64]，今やほとんどのソフトウェアは E 型である。さらに，物理的な実体として提供される他の製品でも，S 型から E 型への変化が見られるようになっている[65]。

E 型のソフトウェアの場合，長期的な視点からそのソフトウェアの品質を左右する品質特性は，保守性と移植性である［大場充 2014］[66]。それは，E 型のソフトウェアでは，新機能の追加や機能の変更が絶対的な条件であり，さらに，長期間の利用を支えるためには，あるコンピュータハードウェア上での稼働から，新しいコンピュータハードウェア上で稼働可能なように移植することが絶対的な条件になるからである。特に，新しいコンピュータハードウェアの場合，利用可能な記憶域が拡大し，処理速度も高速化する。従って，そのような実行の環境条件の変化にも対応できるようにしておかなければならない。

マイクロソフト社の MS-DOS から Windows 10までの変化を考えると，その稼働環境としてのプロセッサは，初期の Intel 8088から現在の Pentium まで変化してきており，そのクロック周波数を見ても，KHz 単位から，GHz 単位まで，100万倍に近い差が生じている。さら

63） S 型は，仕様書を厳密に定義できることを意味し，specification の頭文字をとって，S 型とした。E 型は，一時的に仕様を確定できても，その仕様に永続性が保証できないため，仕様自身が変化することを意味し，進化を意味する evolution の頭文字をとって，E 型とした。

64） 日本でも，1990年代の後半に入って，携帯電話の組込みソフトウェアにおいて，それまでの通信機能に加えて，インターネットとのやり取りを可能にするソフトウェアが付加され，それまでの小型の通信ソフトウェアでは発生しなかった問題が発生した。

65） 自動車用車載組込みソフトウェアの分野では，個々の機構部品の制御に限定されていたソフトウェアの機能が，車内 LAN に接続されたマイクロプロセッサ間での通信によって，協調的に動作をするようになってきており，組込みソフトウェアの規模は，既に1,000万行を超えている。このソフトウェア規模の増大によって，車載組込みソフトウェアでは，S 型から E 型への遷移が起きた。これに対応して，国際標準化機構では，車載組込みソフトウェアの安全性（品質）を担保するため，ISO 26262（組込みソフトウェアを利用した自動車開発・生産の機能安全規格）を制定した。

66） 第 3 章3.1.1節において，保守性と移植性が将来のソフトウェア品質の維持に関係する問題として重要であることを述べた。これは，長期に渡って利用されるソフトウェアでは，新機能の追加や新しい入出力装置への対応，そして新しい稼働環境への適応が問題になるからである。

に，コンピュータに接続される表示装置も，600×400ピクセル程度の解像度から，2,000×1,000ピクセル程度の解像度まで，100倍程度の差が生じている。このような差を吸収しながら，ソフトウェアは進化してきている。

社会の変化や技術の進歩を反映して，さらにWindowsは変化し続けてきた。特に，インターネットの普及と，無線通信技術の進歩から，高速のwifi無線通信，近距離通信のためのブルートゥース（bluetooth）無線通信などの機能が追加され，外部機器の接続のためにも使用できるようになっている。また，従来型の有線接続でも，USBの高速化に対応してきている。逆に，従来のパラレルポート接続インタフェースやキーボード接続インタフェース，232Cシリアル接続インタフェースなどは，除去されてきた。

このように目まぐるしく変化する稼働環境に対応して，Windowsは変化に適応することを強いられてきたのである。さらに，社会的な問題としてパーソナルコンピューティングの一般化に伴い，情報セキュリティが問題視されるようになり，そのための対応も迫られた。特に，起動時のログイン時におけるユーザ認証，さらに外部との通信における情報の暗号化，そしてウィルス対策などの機能も追加されてきた。特に，全世界的に利用者の多いWindowsの場合，ハッカー達の標的にされやすく，セキュリティホール[67]に対する対応も要求された。

これらの変化は，主として稼働環境の変化に対応するための保守や移植の問題への対応と，社会や技術の変化に伴う新機能の追加のためのものであった。ただし，セキュリティホールへの対応は，セキュリティの視点から見た製品の機能的問題を改善するための特殊な保守と言えるものであった。さらに，ユーザから見た製品の使いやすさを改善するための改良や，競合製品（Windowsの場合はLinux）との比較で問題視される低速な処理の改善のための変更，さらに既に市場に提供されている製品に見つかった設計の誤りへの対応などもあった。これらの使用性，経済性，信頼性などに関わる製品改善のための変更は，上述した保守や移植の問題を解決するためではなく，製品の仕様そのものの改善・改良のための変更である。

6.4 保守作業の効率と構造設計

従来のほとんど設計変更がなかった時代の製品では，この保守や移植のための設計変更を目的とした変化・変更がなく，使用性，経済性，信頼性，価格性能比を改善することを目的とした変化・変更が一般的であった。つまり，製品仕様を改善して，より現在の顧客の満足

67） ソフトウェア技術者達が開発時においては全く懸念を持っていなかった機能の実現において，開発終了後にハッカーらがその製品を足掛かりにしてコンピュータシステムに打撃を与える目的で，インターネットなどの通信機能を利用して標的となっているコンピュータへ侵入するために利用する，「ネズミの穴」のようなソフトウェアの部分をセキュリティホール（セキュリティの穴）と言う。

を得られる製品にするための改良を目指す変更であった。これは，その時代の製品のライフサイクルが長く，長期間をかけた開発が可能であったからと言える。経済のグローバル化が進み，時間の進みが早くなった今日では，開発期間の短縮が製品開発戦略にとって重要な課題となった。短期間の開発で製品を世に出し，ユーザからの意見を取り入れて製品の仕様を改良し，新しい仕様に対応する製品を短期間で開発して，世に出すという短い製品開発・保守サイクルを繰り返しながら，長期間にわたり製品を市場に供給し続ける方法が採用されるようになった。

そのような戦略での製品供給に対応できる製品の基本設計が求められるようになってきている。そして，その基本設計の上に，個々の開発活動で開発される仕様の派生製品が実現され，市場に供給されるようになっている。この場合，製品の試験期間も短期間にならざるを得ず，従来のような十分に時間をかけた試験を実施することは不可能になった。その結果，実使用において試験で未確認の機能の組合せの実行が発生しやすくなり，そのような組合せで問題が発生するリスクも高くなる。これに対応するため，ユーザからの意見を素早く吸収し，保守や次の製品開発にフィードバックすることが重要になってきている。

このフィードバックを高速化することは，製品を提供する企業にとって極めて重要になっている。このフィードバックの高速化のために，製品の基本設計において，将来の設計変更や機能追加，さらに稼働環境の変更に対応させるための設計変更に対して柔軟に対応できるような構造を採用することが重要になる。これは，設計者たちが，よく「美しい設計」と言う言葉で表現するようなことである。そのような「美しい設計」，すなわち「良い構造設計」にすることは，ユーザの視点からすると，長期的に見て「質の良い」製品を提供することにつながる。

このような考え方は，従来のS型の製品にはなかった。つまり，従来製品では，利用者が製品を購入した時点での製品の質の良さが，その製品の確定的な「品質」であった。しかし，現代の変化し続ける製品においては，その製品が利用者の変化する要請や，変化し続ける社会の制約[68)]に対応できなければ，製品としての寿命は終わるのである。ソフトウェアなどでは，さらに稼働環境の変化などにも対応できなければならない。例えば，自動運転の自動車であれば，道路交通法が変わった場合でも，それに対応することが求められる。さらに，ヨーロッパなどでは，ある国から別の国に入った時にも，異なる法律や習慣に従った自動運転ができなければならない。自動車が，長い場合には数十年間利用されることを考えると，自動運

68） 例えば，法律の改正に伴う製品仕様の変化が考えられる。消費税が導入されたとき，販売管理のソフトウェアは，新たに消費税を計算して売上金額を計算するように，ソフトウェアの仕様を変更した。そして，さらに消費税が3パーセントから5パーセントに上がった時，プログラムの一部を変更した。同様なことは，西暦が1900年代から2000年代に移行したとき，ソフトウェアの各所に散らばっていた日付チェックなどの計算処理を変更した時にも起こった。

転を制御するマイクロプロセッサの供給が途絶えても，同一のソフトウェアを新しいマイクロプロセッサ上で，利用者が満足できる速さで稼働できるようにしておかなければならない。

6.5 通時的品質と長期的期待効用

以上，変化する製品の「通時的品質」[69] においては，従来型の「静的」[70] な製品品質とは異なり，仕様の良さや製品の質感の良さなどよりも，社会の変化や時間の経過に従って発生する変化への製品利用者の要請に対応できる保守性や移植性が重要になる。このことは，製品を市場に提供する企業が，その社会的責任として，企業の時間的・空間的な安定性を維持することも重要になる。企業が社会の中で長期にわたり，さらに地理的にも広範囲にわたって必要な保守サービスを提供できなければ，その製品の利用者にとっては意味がないからである。このことは，独立した実体としての製品そのものの静的な良さや効用よりも，それを社会（市場）に提供している企業の経営実態の方が重要になることも意味している。

その提供企業の時間的経営安定性のためには，経営の継続性に対する利用者（購入者）の期待が重要になる。それは，古い表現を用いれば，「ブランド力」と表現されていたものに近い。長期的な企業の経営戦略や，経営哲学に基礎を置いた，企業と社会との関係性評価に基づく長期的期待効用[71] である。また，提供企業の空間的安定性のためには，経営思想における多文化への対応や，多地域における製品利用への配慮や考察が十分になされており，製品の販売や保守サービスの提供を，様々な地域において，多様な文化的背景を持った人々を対象として実施できるかどうかの能力が問題になる。その企業の大量生産能力が問題になり，グローバルな企業としての信用が重要になる[72]。

69） コセリウの「言語変化と言う問題」における「通時的」と言う用語の意味を，製品の「品質」をどうみるかの見方を修飾する表現として採用した。従来の品質の概念が，静的で変化しない製品の質を議論するのに対して，ここでは動的に，時間とともに変化する製品の動的な質を問題にするため，「通時的品質」とした。

70） コセリウの用語を用いれば，「共時的」な製品品質と言える。

71） 「長期的期待効用」とは，長期的な視点から評価される，将来に得られると利用者が期待する平均的な効用（価値）を意味する。

72） 大野佝郎氏によれば，世界的な高級ホテルチェーンのリッツカールトンでは，リッツカールトンホテルが提供するサービスの真髄を「ホスピタリティ」として，最高のホスピタリティを顧客に提供することを従業員全員の目標としているとのことである。これは，サービスが製品と比較すれば，その無形性のゆえに変化しやすい側面が強いことに着目し，その変化するサービスの質を維持するための指針として提示された表現であると言える。リッツカールトンホテルは，ブランドであり，そのブランドサービスのコンセプトを表現する言葉が「ホスピタリティ」であると言える。共時的な意味におけるリッツカールトンホテルチェーンのサービスの根底にある基本コンセプトは，長期的な視点で顧客に最高のサービスを提供しようとする「ホスピタリティ」である。そして，「共時的」な意味でのサービス品質として実現しているものが，顧客が支払う料金に見合った利用時点での適正な物理的な滞在環境である。さらにその時点で顧客がホテル利用経験から期待するサービス水準を超えた，顧客個人の満足感を与えることができる，その時代のその社会の文脈で提供できる特別なサービスを問題にしている。

このような背景から，近年の新製品開発，特に新機能を提供する新製品の開発においては，従来は市場との関係の薄かったベンチャー企業などによる製品開発と，新製品の市場への投入が実施される事例が増加している。しかし，そのような新製品が，一旦，市場に受け入れられると，長期に渡り様々な地域において大量に販売され，さらに保守サービスを提供することが必要になる。このため，ベンチャー企業から巨大資本が関係した大企業による製品の販売と保守サービスの提供に変わってゆく例が多い。特に，大規模生産能力があり，販売能力が高い巨大資本が，製品を開発したベンチャー企業を買収し，本格的な事業に成長させる例が見られる。

そのような例の 1 つとして，アップル社によるスマートフォンの開発と，iPhone の販売は，典型的と言える。当初，高価格で高機能な iPhone は市場に受け入れられたが，iPhone の成功が明確になると，韓国のサムスン電子社は，iPhone に似た低価格のスマートフォンを開発し，生産を開始した。巨大資本のサムスン社は，大量生産が得意であり，量産効果によって大幅な価格低減を成し遂げた。さらに，このサムスン社のスマートフォンは，米国のグーグル社が開発しオープンソースソフトウェアとして提供したアンドロイド[73] を基本ソフトウェアに採用している。そのため，ソフトウェアの開発コストを必要としなかった。このことによって，サムスン社のスマートフォンは，アップル社の iPhone を市場の片隅に追いやったのである。

このスマートフォンの開発において重要なことは，サムスン社がスマートフォン用基本ソフトウェアにグーグル社が提供しているアンドロイドを戦略的に採用したことである。このサムスン社が採用した，スマートフォン用基本ソフトウェアとハードウェアを分離して基本ソフトウェアの開発を放棄した戦略は，世界中に拡散している膨大な数の利用者から見れば，自分たちの利用しているスマートフォンは，グーグル社が提供するアンドロイド端末であることを意味する。すなわち，サムスン社以外のメーカが提供する，より低価格なスマートフォンハードウェアへの乗り換えも将来選択可能にする利点を生んだ。これは，一時的にはサムスン社を有利にしたとしても，長期的に考えれば，より低価格のハードウェアを提供できるメーカのスマートフォンの方が有利になる可能性を孕んでいた。

現実に，サムスン社は一時的にスマートフォン市場で世界第一位の市場占有率を誇った[74]。

73） アンドロイドは米 Google 社が開発し，無償提供しているスマートフォン用基本ソフトウェアである。多くのスマートフォンがアンドロイドを利用している。これに対して，iPhone は，アップル社が独自開発し，保守している iOS を利用している。そのため，ソフトウェアの利用者数を見ると，世界的にはアンドロイドが圧倒的に多い。

74） 2015年の IDC による調査では，サムスンの世界市場占有率は，約24パーセントであるのに対して，アップルのそれは，16パーセントで伸び悩んでいた。2013年には，サムスンの市場占有率は，30パーセント程度であった。これに対して，中国系メーカ 3 社が市場占有率で 5 パーセントから 7 パーセントで，この上位 2 社に急速に迫っていた。

その後，より低価格でハードウェアを提供できる中国系企業が出現したため，市場占有率一位の座を手放す結果となった[75)]。さらに，サムスン社はアップル社との新機種開発競争で失敗し，経営的に大打撃を受けた[76)]。このサムスン社のスマートフォン販売中止問題は，サムスン社が市場に投入したいくつかの機種のうちの特定の 1 機種で発生した問題ではあるが，この問題のサムスン社の経営に対する打撃は甚大であり，1 つの製品の品質問題が，同じ企業が提供している製品群全体の販売に影響を与える事例となっている。

サムスン社としては，1 機種に限定された問題でも，同社の開発体制や，同社の経営体質に関係した問題と解釈され，将来の同社の他製品への影響を考慮したとき，企業イメージの毀損を最小限にとどめることを優先し，当該製品の製造販売の中止を決定したと考えられる。しかし，世界中に分散している利用者が，この問題によって，サムスン社の製品全体に対してどのような評価を行うかは，現時点では未知数である。変化する製品の通時的品質の視点から考えると，特定の企業が提供している数多くの製品群の中のたった 1 つの製品の静的品質の問題であっても，その企業のイメージが大きく毀損され，結果としてその企業が提供している全製品の通時的品質の評価を低下させる原因となる可能性もある。

7. 結語：今後の課題と専門家の職業倫理

7.1 解決されるべき課題

21世紀の時代に生きる専門家，技術者達には，自分達が直接関わっている製品の開発や保守において，20世紀型製品における従来の「静的」な品質の問題もさることながら，新しい「変化する製品の通時的品質」の維持・改善が，きわめて重要な命題となる。これは，時間の制約の中で，短期間で開発を実施し，短期間にそのテストを行い，最も適切な時点に市場投入することだけでは，設計上の誤りを完全に除去できないからである。さらに，長期的に見れば，人間社会は常に変化し，製品の利用者は変わってゆく。その変化に対応できるように製品を継続的に変更してゆくことが求められる。

今日の我々には，その「経時的に変化する製品」の本質が何であるかの知識はない。そのような製品がもつ属性の一部を知ったに過ぎない。従って，それを「どう実現してゆくべきか」についての答えは知らない。さらに，その「経時的に変化する製品」の「通時的品質」がどのようなものであり，どのように測定できるかについても，我々は十分な知識を持って

75） 1 社としては依然として世界第一位の市場占有率を誇っているが，中国系の低価格スマートフォンメーカをまとめると，サムスン社は，中国メーカに次いで第二位になっている。

76） これは，2016年10月にサムスン社が発表した，S7 ノートの電池部品と回路基盤との関係から発火するという問題（欠陥）によって，製品の販売停止に追い込まれた問題である。

いない。ただ，従来の知識に基づけば，保守性や移植性として表現されていた問題に近いことだけは理解できる。今後の専門家の活動においては，これらの問題を解明してゆくことが重要になる。

ソフトウェアの開発においては，最近，「アジャイル開発」と言う言葉が使われている[77]。これは，単に言葉の遊びである可能性も否定はできない。例えば，1970年代の末に提唱された「段階的開発（incremental development）」[78] と本質的に何が違うのかは，説明されていない。両者とも，ウォーターフォール開発に対する対案として語られているだけである。しかし，そのような言葉が流行している背景には，従来の開発では考慮されていなかった重要な「何か」が，明確には認識されないまま，そして定義されないままに議論されている可能性もある。

ここでは，そのような議論の一つとして，「経時的に変化する製品」を考えた。「変化する製品」であれば，その製品の仕様を確定し記述することは現実的に不可能である。仮に，ある時点で要求されている仕様を確定し，記述しても，それはその時点における記述としての意味しかなく，通時的な視点で見れば，将来，変わるべきものである。そのような記述に通時的な意味も，共時的な意味もない。ただ，一時的なものとして記述された機能仕様を，プログラムとして実現することは可能である。これを一時的なものとして実現し，利用者たちに提示し，その批判を仰ぐことには意味がある。そして，利用者たちの批判に基づいて，仕様を変更し，新しい仕様に記述しなおし，それを再度実現するのである。この試行錯誤を繰り返すことによって，真に利用者たちの要求に応えられる仕様を決定し，それを実現することは可能であろう。これも，「アジャイル開発」である。

7.2　次世代技術者育成の課題

「経時的に変化する製品」の「通時的品質」を高く維持するために，技術者たちは，その時その時で，自分たちが最善と考える選択をすることが求められる。ただ，その「最善な選択

77）2016年 9 月に東京で開催された日本科学技術連盟主催の SQiP シンポジウム（ソフトウェア品質シンポジウム）においても，アジャイル開発における品質評価をテーマとしたパネルセッションが開催された。

78）1970年代の末にミルズが提案し，ブルックスがソフトウェア開発のリスクを合理的に管理できる開発方法として実験した。それまでのウォーターフォール型開発は，要求を分析し，それに適合した設計をまとめ，その設計に基づいてプログラムを実現した後，テストによって機能が正しく実現されていることを確認していた。この方法では，つねに開発する対象がソフトウェア全体であるため，巨大なソフトウェアの場合，複雑性が管理できないほどに難しくなる問題を持っていた。段階的開発は，要求の分析までは全体を対象とするが，それ以降は，重要な部分から少しずつ選び出し，その部分だけを対象として設計し，実現し，テストを行ってゆく。これによって，問題を管理可能な複雑性に限定するとともに，投入する工数も限定して，問題が発生した場合でもすぐに開発を止められるようにする。

が何か」は，個々の技術者がよって立つ思想によって異なるものであろう。仮に，最善な選択に関する意見が違っていたとしても，多くの場合，間違った選択についての意見は違わない。また，「なぜ自分がこの選択を最善と考えたか」の理由を十分に議論すれば，他の技術者の考えとの違いを明確に認識し，弁証法的プロセス[79]を経て互いに納得できる「より良い選択」に到達できる可能性も高い。

つまり，複数の技術者が互いの選択に関する意見を率直に述べ合い，真摯に議論できなければ，「変化する製品の通時的品質」を高めることはできない。このことに関して，日本人技術者が置かれている現状を振り返ると，状況は楽観できない。それは，我々日本人技術者は，小学校から大学まで，自分たちの考えを明確に説明し，議論することを学んでいないからである。これは，コミュニケーション不足や，コミュニケーション能力不足と言う単純化された言葉で，安易に説明できるものではない。家庭内においても，学校の教育現場においても，我々は話を聴き，理解することばかりに重点を置いた教育訓練を受けてきたのである。

これまでの教育の方法や制度は，産業化社会での，個々に独立した製品の「静的」な品質だけで競争が成り立った時代の，古いシステムでは有効に機能した。しかし，グローバル化が進み，サービス化が進展しつつある新しい社会において，時間の重要性が増し，製品の開発や保守作業の短時間化[80]が，競争に生き残るための条件として著しく重要になっている現代社会においては，長期的に見れば通時的な品質の「良さ」の方が重要になる。そのための対応策として，思考にバイアスがあり，誤りを起こし易い我々人間にとって，その誤りを未然に防止するため，個々人の考えを率直に述べあい，議論することが肝要となる。

21世紀に生きる技術者達はそのことを肝に銘じ，自分の職務を全うし，自分の仕事の成果を限りなく最善の選択に近づけるための努力を惜しんではならない。さらに，次の世代の技術者達を育成するため，自分たちが率先してそのことを実践するだけでなく，次の世代の技術者のために，幼児期からの人間教育・基礎教育に始まり，大学・大学院の専門教育までの期間において，自分たちのアイデアや考えを率直に表明し，互いに真摯に議論する態度を訓練させるように活動しなければならない。

このような仕事への姿勢は，技術者達の生き方の問題でもある。つまり，自分たちがこの世に生を受けた理由が何であり，自分たちが一生をかけて成し遂げるべきものが何であるか，そのため我々は日々をどう過ごしてゆくべきかについて，自分なりの考えを持つことが要求

79） 弁証法的プロセスとは，対立する2つの方法を，独立した複数の属性別にそれらの特徴を列記し，比較評価する。その過程を通して，両者の優劣を比較し，その両者よりも優れた方法を見出す手順である。

80） ここでは，作業を完了するために投入する総労働時間の短縮も問題にするが，さらに作業開始から終了までの期間（どれだけの時間で終わらせられるか）の短縮も問題にしている。

されている。一語で言えば，「倫理観」の問題である[81)]。そのような倫理観なしに，高等教育で何を教え学んでも，社会が技術者達に求めるものは，何も成就できない。我々には，20世紀の日本人にはあったような，向かうべき目的地への地図はないのである。日々をどう生きるかを，自分たち自身の頭で考え，実践することが要求されている。

謝辞

本小論を執筆するにあたり，ソフトウェア業界の大先輩でもある大野侑郎氏に草稿をお読みいただき，重要なご意見を頂戴した。ここに記して謝意を表する次第である。

参 考 文 献

Brooks. Mythical Man-Month. 1975.
————. “No silver bullet,” IEEE Computer, Vol. 20, No. 4, 1987.
Jones. “Measuring programming quality and productivity,” IBM Systems Journal, Vol. 17, No. 1, 1978.
JTC1ISO/IEC. ISO/IEC 9126.
Lehmann. “Programs life cycle and laws of software evolution,” Proc. of the IEEE, Vol. 68, issue 9, 1980.
McCall. “Factors in software quality,” National Technology Information Service, Vol. 1–3, 1977.
Musa. “Software reliability,” Proc. of the IEEE, Vol. 68, issue 9, 1980.
Ohba. “Software reliability analysis models,” IBM Journal of Research and Development, 1984.
Ohba, Basili. “Large scale software testing,” Proc. of the 4th NASA SEW. 1991.
Ohba, Chou. “Does imperfect debugging affect softwarte reliability growth?” Pittsburgh, PA., U.S.A.: Proc. of the 1989 IEEE International Conference on Software Engineering, 1989.
Senge. The Fifth Discipline. 1990.
アダム・スミス．国富論 1 － 3．岩波書店．2000.
ヴォーゲル．ジャパンアズナンバーワン．ティービーエス・ブリタニカ，1979.
コセリウ．言語変化と言う問題．岩波書店，1973.
サロー．富のピラミッド．TBS ブリタニカ，1999.
サンデル．これからの正義の話をしよう．早川書房，2010.
センゲ．最強組織の法則．1995.
テイラー．科学的管理法．ダイヤモンド社，2009.
ドラッカー．ポスト資本主義．ダイヤモンド社，1993.
バートレットスティール．アメリカの没落．ジャパン タイムズ，1993.
フィリップソン．アダム・スミスとその時代．2014.
ブルックス．ソフトウェア開発の神秘．1982.
リンカーン．それでも日本は変われない．日本評論社，2004.

81） これを「コンピテンシー」と表現することもある。1970年代の米国で実施された「マシュマロテスト」では，目前に出されたマシュマロをすぐに食べずに一定時間我慢をし，その後でより多くのマシュマロを得た児童たちが，その後の人生においても成功した事例が多く，高度な知識の獲得よりも，自分が置かれた状況を認識し，その状況に適合した行動を実践できる基本的な「生きる姿勢」の方が，長い人生のためにはより重要であることが示された。コンピテンシーや倫理観として述べられる実践的な能力や態度が問題である。このような能力や態度は，知識を獲得するための教育が開始される前に，教えられ，訓練されなければ，ほとんど身につくことはないと言われている。

大場　充．ソフトウェアの開発技術．1988.
————．ソフトウェアプロジェクト実績データの収集と分析．SRC 出版，1993.
————．「学習する組織」，解説，情報処理　第38巻 5 号，1997.
————．ソフトウェア技術者．日科技連，2014.
————．組込みソフトウェア工学ハンドブック．2014.
東　基衛編．ソフトウェア品質評価ガイドブック．1995.
米国商務省．ディジタル・エコノミー．東洋経済新報社，1999.