2014 年 3 月 15 日

コンピュータ・ソフトウェアにおける
階層介入戦略の研究

－Java と VMware の事例を通じた仮説的推論－

早稲田大学
博士学位論文

2014 年 3 月

加藤 和彦

早稲田大学大学院
商学研究科博士後期課程
商学専攻

目次

序章

―プラットフォーム戦略の探求―

第１節　プラットフォーム戦略研究の動機と関心

企業における事業戦略と IT 戦略の統合分野が、自身の研究領域である。具体的には、開発基盤としての OS（オペレーション・システム）などのソフトウェア拡販戦略に関しての探求に興味がある。自分自身が米国シリコンバレー発の国際的な IT 企業であるサン・マイクロシステムズ社やシスコシステムズ社のマーケティング職に携わり、そのテクノロジーに身近に接したことが研究の動機となっている。

国際的な普及を成功させているソフトウェアは、そのアーキテクチャ（設計思想）に何らかの優位性が存在していると考えていたが、研究を進めるうちに、そこにはプラットフォーム・リーダーの補完業者に対する巧みな誘引戦略や、階層間でのネットワーク効果（外部性）の活用によるプラットフォーム（以降 PF と記すことがある）拡販戦略があることが理解できる。

世界的な普及を果たすソフトウェアが出現して 20 年余り、未だ体系的な理論化がなされていない分野であるコンピュータ・ソフトウェア製品の普及戦略において、特に介入ソフトウェア製品（上位もしくは下位階層のプラットフォームの種類に縛られないオープン性をもつプラットフォーム製品）の普及に関する戦略に、大きな関心をもつに至る。

ソフトウェア産業ならびにネットワーク・コンピューティング業界においては、米国や欧州企業がそのビジネスのイニシアティヴを占有している。日本は元来、半導体や自動車や電機などの「ものつくり」を強みとして国際市場のなかでそのプレゼンスを示してきた。しかし残念ながらパッケージ・ソフトウェアやサービスの国際的市場

ではアップル社、マイクロソフト社、オラクル社、SAP 社、グーグル社、ヤフー社、アマゾン社など外国勢の独断場である。例えば、マイクロソフト社をはじめオラクル社や SAP 社のような業務用パッケージ・ソフトウェア、またアップル社やグーグル社などのスマートフォン携帯端末に関連するソフトウェア、アマゾン社のキンドルなどの電子書籍の台頭がある。

これに対し、日本企業は未だソフトウェア受託開発の労働集約型ビジネスの価格競争に汲々としている。また日本発のイノベーションともてはやされた i-mode も、今ではガラパゴスと称される独自サービスの失敗例として扱われている現状である。本研究にはソフトウェア・プラットフォーム戦略の探究が、日本のソフトウェア産業の国際的競争力を高め、グローバル規模のイノベーションを誘発できる戦略の策定ならびに学術的貢献の一部を担うことができればとの思いがある。今後の日本のソフトウェア業界の国際的な競争力を高めるため、ソフトウェアやサービスのプラットフォーム戦略論を探求することに大きな意義があると考える。

第２節　本論文の主旨

本論文の目的は、先行研究レビュによって補完的な後発プラットフォーム製品のドミナント化のメカニズムに関する仮説的推論をおこない、Java[1]と VMware[2]の事例を確認し、後発プラットフォーム製品の一種である階層介入型プラットフォーム製品の戦略上の仮説の導出をおこなうことにある。本論文でのドミナント・プラットフォーム製品とは、ソフトウェア・レイヤースタック内において、稼動台数で他に大きな差をつけ、強い市場支配力をもつ、階層毎に存在し得るひとつのプラットフォーム製品のことと定義する。

サーバー市場を観ると、市場の誕生期に基盤となる先発のプラットフォーム製品が存在し、そのプラットフォーム製品の上位階層に多くの後発プラットフォーム製品が補完製品として乱立し、市場が成長していくと共に先発プラットフォーム製品がドミナントの地位を築いていく傾向が強い。しかし本論文では、レイヤースタック内で一度形成されたドミナント・プラットフォーム製品の支配力が、後から参入してくる階

---

1 サン・マイクロシステムズ社が開発したプログラミング言語。詳細は後述。
2 ヴイエムウェア社が開発した仮想化ソフトウェア。詳細は後述。

層（ならびにプラットフォーム製品）によって削がれ、補完的な後発プラットフォーム製品へのドミナントの移行を誘発するというメカニズムを仮説提起する。言い換えれば、先発だけでなく後発のプラットフォーム製品もドミナント・プラットフォームになり得るのか。また後発プラットフォーム製品の一種である階層介入型プラットフォーム製品にはドミナント化に関して、どのような戦略上の示唆があるのか。という問題意識を始点としている。

階層構造化が進んだコンピュータ・ソフトウェア市場では、ソフトウェアのレイヤースタックが形成されている。プラットフォーム製品提供者が提供するソフトウェアは、ユーザーが使う製品全体の一階層に位置しながら、他の階層にあるプラットフォーム製品と補完関係をもちながら機能し、完成品において、レイヤースタック内の一部を担っている。レイヤースタック内では、プラットフォーム製品提供者は、他の補完製品提供者と共存しながらも、その優位なポジションをめぐって熾烈な駆け引きをおこなっている。

その狙いは、プラットフォーム製品が、ドミナントとなることで、価格コントロール力や、業界団体や業界標準化プロセスでの発言力、ならびに販売パートナーへの影響力を強化することなどにある。

先行研究では、コンピュータ・ソフトウェアの競争戦略を論じている領域から、1）コンピュータ・ソフトウェア産業の階層的構造変化に関する研究、2）コンピュータ・ソフトウェア企業の分野を特定しない経営戦略に関する研究、3）コンピュータ・ソフトウェア企業の分野を特定する経営戦略に関する研究の３点に分けてレビュをおこなった。またプラットフォーム製品の競争戦略を論じている領域として、1）プラットフォーム・リーダーシップに関する研究、2）プラットフォーム製品の階層戦略に関する研究、3）プラットフォーム製品のドミナント化ならびに WTA に関する研究の３点からレビュをおこなった。

これまでの先行研究の貢献点としては、プラットフォーム製品の普及に関するネットワーク効果に関して、大量かつ広範囲の研究蓄積がある。しかし、戦略の成否に関する研究では、市場における同一レベル階層での競合関係にあるプラットフォーム製品間の先発優位や後発優位に関する研究が比較的主である。加えて、先発プラットフォーム製品がドミナントの地位を築いていく傾向が強いなか、階層介入型プラットフォーム製品を含む補完的な後発プラットフォーム製品の形勢を逆転させるような研

究は十分にされていないと思われる。また、階層介入型プラットフォーム製品のドミナント化のメカニズムに関して詳述する論文も十分とはいえない。

よって、本論文では始めに先行研究レビュにより、補完的な後発プラットフォーム製品のドミナント化のメカニズムの仮説的推論をおこなう。その上で Java と VMware の事例を確認することにより、補完的な後発プラットフォーム製品の一種である階層介入型プラットフォーム製品の戦略上の知見を導出するという方法をとる。このような方法論を選択する理由は、事例対象が極めて少ないという制約のなか、個々の事例の詳細な分析[3]をおこない、可能な限り蓋然性のある仮説を推論する仮説構築型の論文手法をとることで、理論的かつ実践的なインプリケーションを導出するのが適していると思われるからである。

先行研究レビュによりドミナント化要因の仮説的推論をおこない、プラットフォーム製品のドミナント化要因として、要因 A:階層間ネットワーク効果の効用力、要因 B:ブリッジングの影響力、要因 C:プラットフォーム製品排除に対する抵抗力、の３つを提起する。

また、後発プラットフォーム製品提供者の操作項目として、操作項目①：アクセス可能ユーザー数の増加、操作項目②：マルチホーミングコストの低減、操作項目③：隣接対象プラットフォーム製品の多数選定、操作項目④：持続的収益確保モデルの遂行を抽出し、後発プラットフォーム製品のドミナント化要因との関係を仮説として提起する。仮説中の（＋）は促進、（－）は抑制を示す。

〇プラットフォーム製品のドミナント化とドミナント化要因における仮説

仮説 1-1：要因 A・要因 B・要因 C はプラットフォーム製品のドミナント化要因となる

仮説 1-2：仮説 1-1 を前提として、要因 A・要因 B・要因 C の３つの要因が、それぞれ高く（＋）なる場合にドミナント化の可能性が高まる（＋）

3　各事例の詳細情報は、アナリスト・カンファレンス等の公開資料、アニュアル・レポート等の情報ならびにアナリストによる CEO 等とのインタビュ記事より得ている。

○後発プラットフォーム製品のドミナント化要因と後発プラットフォーム製品提供者の操作項目における仮説

仮説 2-1：後発プラットフォーム製品のドミナント化において、項目①の促進（＋）が要因 A を高める（＋）

仮説 2-2：後発プラットフォーム製品のドミナント化において、項目②の促進（＋）が要因 A を高める（＋）

仮説 2-3：後発プラットフォーム製品のドミナント化において、項目③の促進（＋）が要因 B を高める（＋）

仮説 2-4：後発プラットフォーム製品のドミナント化において、項目④の促進（＋）が要因 C を高める（＋）

階層介入戦略は、隣接するふたつの階層間に全く新たなプラットフォーム製品として後から介入する。後から介入するためには、上位層もしくは下位層に対し、オープンなインターフェイスを保持することが必要である。仮に上位層にも下位層に対してもオープンなインターフェイスがない場合、後からの介入は困難になる。また階層介入型ではない後発プラットフォーム製品の参入では総階層数の変化がないのに対して、階層の介入では論理上の総階層数はレイヤースタック内で増加することが特徴である。

よって、階層介入型プラットフォーム製品は、以下のような特徴をもつ。

① （OS などの先発プラットフォーム製品に対し）後発プラットフォーム製品である。

② 階層を形成する最初のプラットフォーム製品である。（レイヤースタック内の総階層数は増える）

③ オープンなインターフェイスを持ち、上下いずれかの隣接階層に複数のプラットフォーム製品を保持し得る。

レイヤースタックの階層間に「介入（Intervention）」し、「橋渡し（Bridging）」をおこなう機能は、既存の階層間関係やプラットフォーム製品間関係を、変化させてしまう可能性をもつ。介入による影響は、各階層のプラットフォーム製品が保有するアクセス可能ユーザーの流動性を高め、同一レベル階層での各プラットフォーム製品

の選択必然性を弱める。アクセス可能ユーザーの流動性の高まりは、相互接続で増加するアクセス可能ユーザーが必ずしも自社プラットフォーム製品の使用に結び付かない可能性につながる。よって、自社以外の隣接プラットフォーム製品にユーザーの多くを横取りされ、結果として他プラットフォーム製品が選択されてしまうことが起こり得る。また、このプラットフォーム製品の選択必然性の弱まりは、同一階層レベルでの各プラットフォーム製品のコモディティ化を誘発する。加えて、介入以前の上下階層をセットにした垂直統合の収益モデルを変化させる可能性が生じる。

次に、ドミナント化のメカニズムを、階層介入型プラットフォーム製品のケースで具体的に理解すること、ならびに階層介入型プラットフォーム製品特有の新たな戦略上の仮説の導出を企図し、Java と VMware のふたつの事例研究をおこなった。Java の事例と VMware の事例の操作項目の観点での確認ならびに分析から、導出される階層介入型プラットフォーム製品特有の戦略に関する仮説は以下である。ちなみに仮説 3-1 は項目①から、仮説 3-2 は項目②から、仮説 3-3 と仮説 3-4 は項目③から、仮説 3-5 と仮説 3-6 は項目④から導出された。仮説中の（＋）は促進、（－）は抑制を示す。

○階層介入型プラットフォーム製品の戦略上の効果（具体的には『既存（先発）の隣接プラットフォームの支配力を介入によって減じる効果』）における仮説

仮説 3-1：階層介入型プラットフォーム製品は、非階層介入型プラットフォーム製品と比較して、隣接（n、n+2）階層のプラットフォーム製品のコモディティ化を誘発し易い

仮説 3-2：階層介入型プラットフォーム製品は、非階層介入型プラットフォーム製品と比較して、既存の隣接（n、n+2）階層のプラットフォーム製品のレイヤースタック内での延命を助長し易い

仮説 3-3：階層介入型プラットフォーム製品は、非階層介入型プラットフォーム製品と比較して、プラットフォーム包囲に対して、それ自体が包囲されにくい防衛的役割をもち易い

仮説 3-4：階層介入型プラットフォーム製品は、非階層介入型プラットフォーム製品と比較して、上下階層セットの垂直統合やバンドルを分断し、既存の隣接プラットフォーム製品の収益モデルにダメージを与え易い

○階層介入型プラットフォーム製品の戦略上の課題（具体的には『階層介入型プラットフォーム製品の普及と収益確保のトレードオフに関する課題』）における仮説

仮説 3-5：階層介入型プラットフォーム製品は普及を優先する（＋）と提供者の収益確保が困難となる（－）

仮説 3-6:階層介入型プラットフォーム製品は提供者の収益確保を優先する（＋）と普及が困難となる（－）

第３節　本論文の構成

本論文の構成は、以下のとおりである。

序章　　―プラットフォーム戦略の探究―
　第１節　プラットフォーム戦略研究の動機と関心
　第２節　本論文の主旨
　第３節　本論文の構成

第１章　プラットフォーム戦略の黎明
　第１節　はじめに
　第２節　コンピュータ・ソフトウェアの階層化の時系列整理
　（１）　プログラムの起源とコンピュータ
　　１）　コンピュータにおけるプログラムの起源
　　２）　プログラム内蔵式コンピュータの誕生
　（２）　プログラミング言語の発達ならびに OS の誕生
　　１）　プログラム技術の進歩とアセンブリ言語
　　２）　高級言語としての FORTRAN と COBOL
　　３）　OS の誕生と OS/360
　（３）　コンピュータ・ソフトウェア産業の創生と産業構造
　　１）　IBM 社の動きとソフトウェア・ビジネスの誕生
　　２）　IBM System/360 による産業構造変化
　（４）　小括
　第３節　プラットフォームの定義
　（１）　基盤機能とメディア機能
　（２）　基盤機能とメディア機能の統合
　第４節　プラットフォーム戦略における階層化の概念
　（１）　上位下位階層の特徴
　　１）相互依存性と一方向依存性
　　２）モジュール化との違い
　　３）階層化による下部隠蔽の役割
　第５節　おわりに

第１章

プラットフォーム戦略の黎明

第１節　はじめに

プラットフォーム戦略は、近年、経営戦略のひとつとして注目されてきた。その理由は、NTT ドコモの i-mode や楽天などのビジネスの隆盛により、それに関する学術的な研究がおこなわれ始めたためである。

例えば、クレジットカードやオークションサイト、イエローページにショッピングモールなどの戦略は、プラットフォーム提供者が補完業者を誘引することによってエコシステムを成長させプラットフォームの拡販を目指す共通点がある。

本章では、「プラットフォーム戦略の黎明」と題して、本論文の議論の前段階とでもいうべき、コンピュータ・ソフトウェアの階層化の時系列整理、プラットフォームの定義、プラットフォーム戦略における階層概念について論じる。

第２節では、はじめにコンピュータ・ソフトウェアの階層化の時系列整理をおこなう。今日でこそコンピュータは生活にも業務にも標準的に使われているが、現在のようなソフトウェアの階層構造を形成するまでの起源と変遷に関してまとめてみたい。プログラムの誕生からプログラム内蔵式コンピュータの誕生、そしてプログラミング言語の発達ならびに OS の誕生へと繋がり、プログラム技術の進歩とアセンブリ言語や高級言語としての FORTRAN と COBOL について説明する。やがて OS が IBM 社の OS/360 の発売が現在の OS の誕生の契機となり、コンピュータ・ソフトウェア産業の創生とソフトウェア・ビジネスの誕生に伴って、産業構造変化へと繋がっていく経過を説明する。その上で、コンピュータ・ソフトウェアが、OS の誕生ならびにコンピュータ・ソフトウェア産業の水平型構造の発展とともに、階層化してきた歴史を整理する。

第３節では、プラットフォームの定義について論じる。根来・加藤（2010）の製品・サービスにおけるプラットフォーム研究のふたつの流れと、基盤機能とメディア機能を統合する定義を論じる。

第４節は　プラットフォーム戦略における階層概念について、上位下位階層の特徴や相互依存性と一方向依存性の概念、モジュール化との違いや階層化による下部隠蔽の役割について論じる。そこでは本論文の全体を通じて議論されるプラットフォーム製品戦略では、上位下位階層の考え方、ならびに相互依存性と一方向依存性の違いがプラットフォーム戦品戦略上の重要な概念となる。

第２節　コンピュータ・ソフトウェアの階層化の時系列整理

誕生から半世紀あまり、コンピュータは今日まで金融、製造、流通、運輸、公共、医療、教育、軍需など様々な産業で大きな役割を担ってきた。そしてこれらの産業の発展に大きく貢献したものにプログラム言語をはじめとするOS、アプリケーションなどのソフトウェアがある。

本節ではコンピュータ・ソフトウェア階層化の時系列整理をおこなう。具体的には、コンピュータ・ソフトウェアがOSの誕生ならびにコンピュータ・ソフトウェア産業の水平型構造の発展とともに、階層化してきた歴史を整理する。

コンピュータ・ソフトウェアは階層化されることにより、ひとつ若しくは複数の階層の機能に該当するソフトウェア製品を提供するプラットフォーム提供企業によって、階層を梃子（Leverage）にした企業戦略が遂行されてきた歴史がある。例えば階層化されたソフトウェアにおいて、ソフトウェアを提供する企業の施策は、競合他社に対していかに自社の優位を獲得するかという戦略に置き換えられる。そこではネットワーク効果を導き出し、プラットフォーム製品上での補完業者・補完製品の獲得と創出を促すメカニズムを誘引する階層施策が、企業戦略としての重要な意味を持つ。

本節では、まずソフトウェアの階層化のプロセスとして、コンピュータの歴史におけるOSの誕生がもたらした階層化に関して整理をおこなう。次にソフトウェアの階層化に追随して起こった現象として、コンピュータ・ソフトウェア産業の勃興と産業構造の成り立ちの歴史を整理する。

（1） プログラムの起源とコンピュータ

本項にて、ソフトウェアの階層化プロセスの整理として、コンピュータの歴史における OS の誕生に関して整理をおこなう。階層化の代表的な例としてコンピュータ・ソフトウェアのプラットフォーム製品としての OS[4]に関し、その起源を時系列的に整理する[5]。OS の誕生を説明するためにはソフトウェアの起源から説明する必要がある。そしてコンピュータにおけるソフトウェアの歴史はプログラム（計算作業手順）の起源にまで遡ることになる。

1）コンピュータにおけるプログラムの起源

1944 年、5 年間の歳月を費やしてハワード・エイケン(Howard H. Aiken)の率いるハーバード大学の計算研究所と IBM 社との合同チームが約 3,000 個のリレー（継電器）[6]を使い電気式計算機“Harvard Mark Ⅰ”を完成させる[7]。Harvard Mark Ⅰではパンチカード（穿孔(せんこう)カード）によって計算手順を与えるという考えが取り入れられた[8]。

同じ頃、米国アバディーンの陸軍試射場の弾道研究所では、真空管を使った高速計算機の研究がおこなわれていた[9]。その主たる目的は「大砲の弾道の落下位置を、装薬量（火薬の量）、砲身長、雷管の形状などの基礎データを元に計算し、弾道表を作る」ことにあった[10]。この研究では ON/OFF の 2 つの値を表す真空管を大量に搭載した機械が考えられていた。ペンシルバニア大学の J. モークリ(John W. Mauchly)、と J. P. エケット(J. Presper Eckert)、陸軍兵器局の H. ゴールドスタイン(Herman

---

4 ただし、コンピュータ・ソフトウェアにおいてのプラットフォーム製品という言葉が常に OS を指す、若しくは OS に限定されるということを意味している訳ではない。具体的には隣接階層に補完製品を持つものはソフトウェアでは OS に限らずアプリケーションも、ミドルウェアもプラットフォーム製品である。

5 整理に関しては、主として長谷川（2000）ならびにケリー・アスプレイ（2006）からの引用による。

6 ジョセフ・ヘンリー（Joseph Henry）が発明した電力機器。動作スイッチ・物理量・電力機器の状態に応じて、制御用の電気信号を出力する。

7 長谷川（2000）, p. 32 参照。

8 長谷川（2000）, p. 33 参照。

9 長谷川（2000）, p. 34 参照。

10 同上。

H.Goldstine)らは、大量の真空管を使った巨大な電子計算機を開発する[11]。開発された電子式数値統合計算機（Electronic Numerical Integrator And Calculator）は頭文字をとって“ENIAC”と呼ばれ、1946 年に公開された[12]。ENIAC は 17,468 本の真空管とコンデンサ 1 万個、スイッチ 6,000 個を用い、全長 30m、総面積 170 ㎡、重量 30 t という現在のコンピュータと比べて、とてつもなく巨大なシステムであった[13]。コンピュータの起源はこのように軍事目的からはじまった。

ENIAC がそれまでの自動計算機と異なるのは、計算手順の制御方法であった。これまでの電気・計算式計算機ではパンチカードまたは穿孔テープから命令を読み取り、演習装置に送られる。この方法では、処理速度がカードやテープを送る機械的な速度に依存してしまう[14]。一方 ENIAC では、演算の開始と終了の指示が、電気信号として送られるようになっていた[15]。たくさんの接点穴（ジャック）を設けた配線盤に、両端にプラグの付いたジャンパーケーブル（プログラム線）を差し込む。この「線の配列」が ENIAC のプログラムであった[16]。

ENIAC のプログラミング手法は一度プログラムを組めば機械的な読み取りは入力データだけとなるため、演習速度はカード送り機械の速度に依存しなくなるという利点はあるものの、再プログラムの作業は重労働であった[17]。その理由はプログラムが「ケーブルの配線」であったという点である[18]。ENIAC でのプログラミングとは、ケーブルの配線やスイッチの ON/OFF という物理的な作業そのものであった。ひとつの計算をおこなうためには、膨大な数のケーブルをつなぐ必要があり、さらに別の計算のためにはその配線をその都度差し替える必要があった[19]。当時は過去の配線を記録しておき即座に元の配線に戻すというようなプログラムの再利用性もなかった[20]。よってその問題を解決するために次に説明するプログラム内蔵式コンピュータの誕生へとつながっていく。

---

11 同上。
12 長谷川（2000）, p. 35 参照。
13 同上。
14 同上。
15 同上。
16 同上。
17 長谷川（2000）, p. 36 参照。
18 ケリー・アスプレイ（2006）, p. 90 参照。
19 同上。
20 同上。

2）プログラム内蔵式コンピュータの誕生

このような問題の解決に貢献した科学者がENIAC開発チームに途中から参加した、ジョン・ノイマン(John von Neumann)である[21]。ノイマンは記憶容量を増やすことでプログラムをコンピュータ本体のなかに組み込むことができると主張し、計算手順そのものを電子の記憶とすることで、プログラミングの手間の軽減とプログラムの再利用を可能にした。これが現在実用化されているほとんど全てのコンピュータであるプログラム内蔵式[22]のコンピュータである。1944年頃からノイマンと開発チームはEDVAC（Electric Discrete Variable Automatic Computer）と名付けたプログラム内蔵型コンピュータの開発をおこない、1950年に完成させる[23]。

しかしそれ以前の1949年、ノイマンらのアイデアを知った英ケンブリッジ大学のM.V.ウィルケス(M.V.Wilkes)らがプログラム内蔵型コンピュータであるEDSAC（Electric Delay Stored Automatic Calculator)を発表していた[24]。

プログラム内蔵型は現代のコンピュータにまで受け継がれている計算機の基本的な概念である。プログラム内蔵型とは「計算機に与える命令をコード（記号）で表記し、計算の対象となる数値データと同じ記憶装置上に記憶させる方式」である[25]。つまり機械式計算機で用いられていた紙テープや穿孔カードの情報、あるいはENIACのプログラム線の配列に相当する部分を、コンピュータの記憶装置、即ちメモリ内に読み込ませ、処理手順も計算結果もすべてメモリの中に置いて処理するという仕組みである[26]。言い換えれば、それまで計算のためだけに使われてきた記憶装置にその機能を広げ、「命令」を読み込ませようという目的である。現在使われている情報の置き場としてのメモリの概念はこの時生まれた[27]。

プログラム内蔵式にはふたつの大きな利点が考えられる。第一の利点は命令がメモリ内にあるため、テープやカードより早く読み出せる[28]。もうひとつの利点は命令の

---

21 ケリー・アスプレイ（2006）, p. 88 参照。
22 EDVAC開発に加わっていたノイマンの名前にちなんで「ノイマン型」と呼ばれる。
23 ケリー・アスプレイ（2006）, p. 91 参照。
24 ケリー・アスプレイ（2006）, p. 102 参照。
25 長谷川（2000）, p. 53 参照。
26 同上。
27 同上。
28 長谷川（2000）, p. 54 参照。

実行順序を切り替えるということで、既に実行された（別の場所にある）命令を瞬時に再生できるという点である[29]。

（２） プログラミング言語の発達ならびに OS の誕生

前項にて、コンピュータにおけるソフトウェアの歴史をプログラム（計算作業手順）の起源から説明してきた。本項では、ソフトウェアの階層化プロセスの整理として、コンピュータの歴史における OS の誕生に関して、プログラム言語の発達と OS の誕生までを時系列的に整理する[30]。

１）プログラム技術の進歩とアセンブリ言語

プログラムをデータと同じメモリに読み込ませるプログラム内蔵式の登場によって、コンピュータは柔軟な制御が可能になり、コンピュータとそれを動かすためのプログラム技術が進歩することになる。

EDSAC では、２進数の機械語を符号と対応させた命令体系を用い、コンピュータにおこなわせることを記号列として書き表せるようになった[31]。これがアセンブリ言語の原形である。

また EDVAC では開発陣がサブルーチン[32]のライブラリ化[33]やサブルーチンの結合というプログラミングの効率化手法を提案した。サブルーチンをライブラリとして記憶装置（メモリ）内に蓄積することで、既存の処理を再利用する効率的なプログラムを記述できた[34]。当時この手法を用いたシステムはコンパイラ[35]（compiler:翻訳者）と呼ばれた。符号を機械語に翻訳するという意味である。またサブルーチンの存在と再利用は、今日のプログラミング言語の基盤である開発環境の基礎概念を生んだ[36]。

---

29 同上。
30 整理に関しては、主として長谷川（2000）ならびにケリー・アスプレイ（2006）からの引用による。
31 長谷川（2000）, p. 67 参照。
32 随時利用される、目的の決まった小さなプログラムのこと。
33 ある特定の機能を持ったプログラムを、他のプログラムから利用できるように部品化し、複数のプログラム部品をひとつのファイルにまとめること。
34 長谷川（2000）, p. 69 参照。
35 プログラミング言語で記述されたソフトウェアの設計図（ソースコード）を、コンピュータが実行できる形式（オブジェクトコード）に変換するソフトウェア。
36 長谷川（2000）, p. 69 参照。

プログラム内蔵式の登場によって、条件分岐や繰り返し処理が可能となり、さらにサブルーチンや浮動アドレス記法[37]によってアセンブリ言語を使ったプログラミングが容易となった[38]。これによりプログラム技術は大いに進歩することとなった。しかし CPU の持つ 2 進数の基本命令に直接記号を割り当てるという形態は、機械語からアセンブリ言語になっても基本的には変わることはない。そのためより人間が扱いやすい、人間側寄りの思考をコンピュータに抽象化した形で伝える利便性が望まれ、プログラム言語はアセンブリ言語から高級言語と呼ばれるものに進化していくことになる[39]。

2）高級言語としての FORTRAN と COBOL

高級言語の代表的なものとして FORTRAN がある。FORTRAN が生まれる以前にも人間の言葉に近いソースコードで記述をおこなえる言語はいくつか存在した。FORTRAN は度重なる改編を経て半世紀後の現在まで受け継がれている。

1953 年 MIT で J.H.ラニング Jr. (J.H.Lanning,Jr.)らが通常の数式に似た形で計算式を書けるシステムを開発した[40]。1954 年 J.バッカス(John W.Backus)をはじめとする IBM 社の開発チームがこれに既存の技術を注ぎ込み FORTRAN の開発を始めた[41]。FORTRAN は FORmula TRANslation（数式変換）の文字を取ったもので、この名前も示すように数式を分かりやすく記述する目的のために考案された言語である[42]。

FORTRAN 登場当時の入力デバイスは穿孔カードであったが、＝による代入、四則演算、()による優先順位の指定、SIN,COS といった三角関数など、一般的な数学の記述がそのまま利用可能であった[43]。FORTRAN が開発された理由は、機械語あるいはアセンブリ言語など「機械寄り」の言語によるプログラミングでは、手間と経費がかかり過ぎて効率が向上できないという背景があった[44]。

---

37 符号化された命令に仮のアドレスを与えておき、メモリに読み込まれたときそれを具体的な実アドレスに展開するという手法。
38 長谷川（2000）, p. 69 参照。
39 長谷川（2000）, p. 70 参照。
40 長谷川（2000）, p. 76 参照。
41 同上。
42 同上。
43 長谷川（2000）, p. 77 参照。
44 同上。

当時、機械語はもちろんアセンブリ言語同士でも互換性などは全くなく、機械が違えば作業環境は製品ごとに異なっていた。そのような状況のなか、FORTRAN は FORTRAN コンパイラさえ動けば、異なるコンピュータでも同じプログラムを使うことができた。いわゆる「移植性」という恩恵をもたらした[45]。

もうひとつの代表的な高級言語として COBOL が生まれた。 COBOL は Common Business Oriented Language（一般実務用言語）の略である[46]。その名が示すように FORTRAN のもつ科学プログラミングの機能では不可能な、一般事務処理のためのプログラムが簡単に書けることを目指してつくられた言語であった。

COBOL は英数字とファイルの扱いが得意な言語であると言われる[47]。また、書式化されたカードからの一括入力や帳票出力など、事務分野で多用される入力形式に強かった。COBOL のデータ形式は「標準データ形式」と呼ばれ、大量の定型情報を一括処理する仕組みとしてデータベースの概念と深く関わっている[48]。COBOL は様々なデータ形式や複雑な入出力形態をファイルの概念に抽象化させたことで、コンピュータ技術にそれほど精通していないユーザーでもプログラムを組める恩恵をもたらした[49]。

COBOL の登場は FORTRAN によって始まったコンピュータの一般の利用を加速させることになる。「科学技術計算の FORTRAN、事務計算の COBOL」[50]と言われるように COBOL はそれまでは数学的な知識が不可欠であったプログラミングというものを一般の事務に持ち込めるようなより簡易なものとすることで普及の加速をもたらした。

３）OS の誕生と OS/360

アセンブリ言語で書かれたソースコードをコンパイルするには、穿孔カードの束を積んで、カードリーダーを動かし、テープを掛け替え、とオペレータは毎回同じ作業を繰り返していた[51]。この定型化された作業を効率よく進め、一連の処理を自動化し

---

45 長谷川（2000）, p. 77 参照。
46 ケリー・アスプレイ（2006）, p. 193 参照。
47 長谷川（2000）, p. 98 参照。
48 同上。
49 長谷川（2000）, p. 99 参照。
50 長谷川（2000）, p. 100 参照。
51 長谷川（2000）, p. 84 参照。

オペレータを助けるためにOSの原型となるモニタ（monitor）が1955年に開発された[52]。

FORTRANで書かれたプログラムのコンパイル作業を自動化する目的で、1950年代後半にFortran Monitor System（FMS）と呼ばれるOSの原形が出来上がる[53]。その後FMSの概念は1962年に登場するIBM社のIBSYS[54]に受け継がれ、1966年IBM社のSystem/360[55]用のOSであるOS/360として結実する。この時のOS/360は、入出力サポートや処理の自動実行と制御機能を備えたOSとして確立された[56]。

OSの登場によって開発環境が整備され、FORTRAN処理系が様々なOSの上で動くようになるとユーザーの拡大に拍車がかかった。またコンピュータがそれまでの一部の専門家だけが扱う難しい機械から、ごく普通の業務をおこなうための装置へと徐々に姿を変えていった。その後、1960年代前半にはCOBOLをはじめとするALGOL[57]、BASIC[58]、PL/I[59]など分野別の様々な言語が多数登場する。

OS/360の開発は当時としては最大規模のプログラミング業務であった。1963年から66年に約5,000人・年の工数（1,000人以上のプログラマー）と100万行以上のコード、当初予算の4倍の5億ドルが投入された[60]。1967年OS/360は予定より丸1年遅れで完成したが、リリース後、数十個ものエラーが発生し、それを根絶するのに何年も必要であった[61]。IBM社の直面した困難の主たる原因は、当時としては野心的と言える何百ものプログラム・コンポーネントの全てが遅れなく、互いに強調して働か

---

52 同上。

53 同上。

54 各種サブシステムと言語サポートを備えた重装備の1962年頃のOS。

55 1964年にIBM社が発表した最初の汎用メインフレーム。当時、他社を圧倒してメインフレーム市場をほぼ独占する。当時のコンピュータでは同一メーカーであっても設計仕様が異なり、それぞれの専用OSとアプリケーションを使っていたので、同一メーカー内でもある機種から他の機種にソフトを移植することが非常に困難であった。System/360ファミリーであれば、同一のOSが走るので、ユーザーはアプリケーションや周辺機器をある機種からある機種へと自由に移せるようになった。

56 ケリー・アスプレイ（2006）, p. 199参照。

57 1960年頃にヨーロッパの学者グループによって開発されたプログラミング言語の総称である。ALGOL 58、ALGOL 60、ALGOL 68などがある。仕様が巨大かつ複雑であったために広く普及することはなかったと言われている。

58 ダートマス大学のジョン・ケリー（John G. Kemeny）とトーマス・カーツ（Thomas E. Kurtz）によって開発された、初心者向けの対話型言語。

59 IBM社によって開発され、1970年代には同社のメインフレームの標準的言語として利用された。ALGOLを基礎に事務処理用言語COBOLや科学技術計算用言語FORTRANの機能を取り込み、あらゆる用途に耐える汎用の言語として、すべての言語を置き換えるべく開発された。

60 ケリー・アスプレイ（2006）, p. 202参照。

61 同上。

ねばならないものであったこと、またマルチプログラミングという、いくつかのプログラムを同時に実行できる新しい技術を開拓しなければならないことであった[62]。

IBM 社のこういった経験は、ソフトウェアの開発と大きなプロジェクトを管理するには特別な努力が必要とされることを世の中に広く認知させる結果となった。

1960 年代後半から 70 年代初めにかけて「ソフトウェア工学」という新しい分野がつくられた。これによりソフトウェア開発に関する業界内のルール、手法、ツールなどが確立されていくことになる。

その後 1990 年代に入り様々な OS が登場した。OS のメリットは階層化プラットフォーム製品として、多くのアプリケーションの開発とその利用に貢献してきた。現在 OS と呼ばれるものは Windows、Unix 系、Linux が代表的なものとして存在する[63]。

（3） コンピュータ・ソフトウェア産業の創生と産業構造

本項ではソフトウェアの階層化に追随した現象として、コンピュータ・ソフトウェア産業の勃興と産業構造の成り立ちの歴史を整理する[64]。前項にて最初の OS として OS/360 の誕生の経緯に触れたが、OS/360 はソフトウェアを階層化させる要因としての位置付けに加え、ソフトウェア・ビジネスの勃興においても重要な役割を担うこととなった。

1）IBM 社の動きとソフトウェア・ビジネスの誕生

初期のコンピュータのハードウェアメーカーは、ハードから独立したソフトウェア製品でビジネスを構築する気持ちはもっていなかった。プログラミング技能も十分ではなかったし、コンピュータ・システムの大型ユーザーは皆コンピュータ・メーカーと同等、若しくはそれ以上のプログラミング技能を自社内に有していた[65]。1950 年代末期から 60 年代初期は、今日我々がソフトウェア製品と呼ぶものに、起業家たちは

---

62 ケリー・アスプレイ（2006）, p. 200 参照。

63 具体的には 2013 年 9 月の時点で、Windows（サーバー側：Windows Server 2012、クライアント側：WindowsXP, WindowsVista, Windows7, Windows8 など）、Unix 系（FreeBSD, OpenBSD, NetBSD, Mac OSX, Solaris, HP-UX, AIX など）、Linux（ディストリビューションによって、Debian 系、RedHat 系, Slackware 系など）、国産 OS（TRON など）、これら以外にも多くの OS が存在する。

64 整理に関しては、主として Cusunano（2004）ならびにハジウ（2006）からの引用による。

65 Cusumano（2004），（邦訳）p. 138 参照。

商機を見出しておらず、当時開発された製品のほとんどは、特定業界の特定機能向けのニッチなソフトだった[66]。ソフトウェア製品のビジネスは、起業家が特定のアプリケーションに標準的なユーザー機能を見出し、多数の顧客用にパッケージ化することが出来るようになった時に活況を呈しはじめた。

製品ビジネスを促すことになったひとつの契機は、1964 年以降、IBM 社が System/360 ファミリーという互換性メインフレーム・コンピュータのモデル群を発表しはじめたことである[67]。各モデルでは、異なるニーズを抱えるユーザーのために、様々なレベルのプロセッサの処理能力やメモリ容量が用意された。この新しいコンピュータ群がきっかけとなり、IBM 社や他の多くの企業が、新しいシステム・ソフトウェアやアプリケーション・ソフトウェアを開発した。この頃、前項にて触れた OS としての OS/360 が生まれた。IBM 社はこの巨大なソフトウェアをハードウェアに一体化して、一見「無料」の形で販売した[68]。

メインフレーム市場の約 80%を握っていた IBM 社が、ハードウェアの販売とソフトウェアサービスの販売をある程度分離するまでは、ソフトウェア製品ビジネスはあまり発展しなかった[69]。IBM 社は 1968 年 12 月にはじめてソフトウェアの一部に独自の価格を設定し、1970 年代初頭にハードから独立して販売を開始すると発表した[70]。OS/360 の開発を陣頭指揮したワッツ・ハンフリー(Watts S. Humphrey)によると、IBM 社の経営陣が分離販売の決断を下した理由は、米国司法省が（ハードウェアとソフトウェアの）一体販売を競争阻害要因と考える恐れがあったからであるとしている[71]。当時 RCA は、IBM 社のソフトウェアが動く互換コンピュータの発表を計画しており、司法省は、互換機メーカーが市場で生き残るチャンスを確保したいと考えていた[72]。実際に当時の IBM 社は、ソフトをハードから分離しないことによって、圧倒的に有利な立場を得ていたためである[73]。

---

66 同上。
67 Cusumano（2004），（邦訳）p. 140 参照。
68 同上。
69 同上。
70 同上。
71 Cusumano（2004），（邦訳）p. 141 参照。
72 同上。
73 Cusumano（2004），（邦訳）p. 140-141 参照。

IBM 社のソフトウェアとハードウェアの分離販売の方針は、IBM 互換用ソフトウェアを開発したいと考えていた新しい世代の起業家たちの意欲を掻き立てることとなった。米国では 1972 年には 81 社[74]ものソフトウェア企業が新たに出現した。IBM 社は主にアプリケーション・ソフトを分離販売したので、ソフトウェア業界への新規参入者にとっての新しい商機はこの分野が最も多かった。

ハジウ（2006）は「（IBM 社による自社メインフレームのハードウェアとソフトウェアを）切り離し、別の産業とするアンバンドル化が進んだため、IBM 社用にプログラムを提供するソフトウェア産業が成長した。その後、マシンが小型化しワークステーションや PC の開発が進むにつれ、デジタル・リサーチ社、アップル社、マイクロソフト社、ノベル社などの独立系ベンダーによって供給される OS が産業のカギとなる（後略）」と指摘している。

2）IBM System/360 による産業構造変化

System/360 は単にコンピュータの在り方のみならず、産業構造を劇的に変えていくこととなった。デザイン・ルールの劇的な変更によって独立した「モジュール」に挑戦する小さなチーム（その多くはスピンアウト）が誕生した[75]。1960 年代以降にこうしたスピンアウトをはじめとする数多くのスタートアップ企業が群生し、次第にコンピュータ関連の周辺機器産業が複数のクラスターとして形成されていった。

國領(1999)は「コンピュータ産業は「垂直囲い込み型」から「水平展開型」への経営構造の転換を最も劇的に経験した。」という見方を示し、コンピュータ産業のビジネス展開が水平プラットフォーム型へ移行してきたとの考えを示している。その上で國領(1999)は、水平展開型戦略の適合する業界の必要条件として、モジュール化が可能であり、水平展開をおこなうニーズが高いことが挙げられると説明している。そして、そのようなニーズが高まるのは付加価値におけるソフトウェアの比重が高く、開発コストが大きい割に開発済み製品の追加供給の変動費が低いことなどの特質をもっている場合で、（この現象は）デジタル情報技術に立脚した知識経済における典型的な知識集約型産業の特質であるとも説明している。

---

74 Cusumano（2004），（邦訳）p. 143 参照。
75 青木・安藤（2006），p. 126 参照。

（4） 小括

コンピュータ・ソフトウェアの階層化の時系列整理のこれまでのまとめとして以下に整理する。

初期のコンピュータにおいて、プログラミングの再利用の観点からノイマン式コンピュータが生まれ、同時にアセンブリ言語や高級言語が誕生する。OS は、本来オペレータがあらゆる処理をおこなうプログラムをすべて作成し、マシン語（機械語）でプログラムを記述し、スイッチを使って入力していくという面倒な定型作業を自動化する目的から誕生した。OS 階層のアプリケーション階層へのインターフェイスの公開により、連携した多くのアプリケーションが生まれ、ユーザーには選択の幅がもたらされた。同時に OS は開発環境の基盤の役割も持つことになった。このようにしてコンピュータ・ソフトウェアの階層化が OS とアプリケーションの分離という形でおこなわれると、OS はソフトウェアにおけるプラットフォームの役目を担うこととなる。コンピュータ・ソフトウェアのプラットフォーム機能とは主としてふたつあり、ひとつはアプリケーションの創発を促す基盤を提供すること、そしてもうひとつは下部階層を抽象化する（隠蔽する）ことである。

一方、OS の誕生と時期を同じくして IBM 社がソフトウェアとハードウェアの分離販売を始めると、多くのスピンアウトした小規模なベンチャー企業が OS や多くのアプリケーション・ソフトウェアを開発し販売を始めた。こういったベンチャー企業の目標は、独立したモジュールに挑戦する小さなチームとして、イノベーションの質とスピードで勝負するモジュール内競争に勝つことであった。言い換えれば、もはや IBM 社がおこなってきたような垂直的なアーキテクチャで独自の優位性を築くことを目指すのではなく、専門化した特定の領域でのベストプレイヤーになることであった。

このようにして、産業を構成する多くのベンチャー企業による水平展開型ビジネスの遂行は、それまでの IBM 社や他のメインフレームメーカーが築いた垂直統合型の産業構造にも変化をもたらしていくこととなった。こういった産業構造の変遷は、コンピュータ・ソフトウェアの階層化に追随して起こった現象のひとつとして考えられる。

そして、これらの潮流の背景には、プラットフォーム製品提供者、補完製品・補完業者、ならびにユーザー全ての間に間接的なネットワーク効果が作用しており、プラ

ットフォーム普及の正のフィードバックが生じる環境が整っていたことは注目に値する。

第３節　プラットフォームの定義

本節ではプラットフォームの定義について論じる。プラットフォームという用語は、日常の一般用語としては「駅などで、乗客が乗り降りする一段高くなった場所」（広辞苑）の意味として使われることが多い。一方、コンピュータでの用語は「アプリケーション・ソフトを稼動させるための基本ソフト又はハードウェア環境」と言う意味で使われている。

IT 用語辞典 e-Words[76]には、コンピュータ関連の用語として以下の説明がある。プラットフォームとは、アプリケーション・ソフトを動作させる際の基盤となる OS の種類や環境、設定などのこと。Windows や Unix、Mac OS は、それぞれ異なるプラットフォームである。また、OS にとっては、自らを動作させる基盤となる PC/AT 互換機、Macintosh などのハードウェアの種類がプラットフォームである。アプリケーション・ソフトにせよ OS にせよ、対応しているプラットフォームはあらかじめ決まっており、それ以外のプラットフォームでは動作しない。例えば、Mac OS プラットフォーム上で動作する文書作成ソフトは、Windows を搭載したパソコンでは動作しない。ただ、複数のプラットフォームに対応するために、「Macintosh 用」「Windows 用」などのように、それぞれのプラットフォームに対応した同じアプリケーション・ソフトを用意することはある。

上記のコンピュータ用語から派生して、経営学の文献では、製品の構造を階層的に捉えて表現する場合や、それに対応した産業構造の階層性を前提にして、ある条件を満たす階層（部分）をプラットフォームと呼んでいる。

例えば出口（1996）は「階層的に捉えることの出来る産業や商品において、上位構造を規定する下位構造（基盤）」という意味でプラットフォームという言葉を使用している。同様に、竹田・國領（1996）は、以下のように述べている。「産業や商品は、しばしば階層的に捉えることができる。例えば、パソコンは、ハードウェア、OS、ア

---

76 株式会社インセプト(Incept Inc.)が運営するウェブ上の IT 用語辞典　http://e-words.jp/ 2008/12/17

プリケーション・ソフトといった異なる階層の商品が組合わさることによって機能を果たす。通信販売会社は、電話会社、運送会社、クレジットカード会社などのサービスを基盤として、消費者に対し統合的なサービスを提供している。プラットフォームという用語は、このように階層的に捉えることの出来る産業や商品において、上位構造を規定する下位構造（基盤）」を意味する。

プラットフォームを製品構造の議論の文脈ではなく、ある業種・業態を指す言葉として使っている例もある。プラットフォームを「プラットフォーム・ビジネス」として理解する立場である。今井・國領（1994）、國領（1995）、根来・木村（1999）などである。今井・國領（1994）によると、プラットフォーム・ビジネスとは「誰もが明確な条件で提供を受けられる商品やサービスの供給を通じて、第三者間の取引を活性化させたり、新しいビジネスを起こす基盤を提供する役割を私的なビジネスとして行っている存在」である[77]。根来・木村（1999）は、「インターネットコマースにおける介在型プラットフォーム・ビジネス」をとりあげ、プラットフォーム・ビジネスを「第 3 者間のコミュニケーションに介在し、インターネットコマースを活性化させる私的ビジネス」と定義している。

以下に挙げる幾つかのプラットフォームの定義は、コンピュータ・ソフトウェアのプラットフォームに限定されたものだけではなく、コンピュータ・ソフトウェアを含むプラットフォーム広義の定義である。また定義される際のコンテクストも一様でないが、「参加者の創発を促す基盤」という意味を共通に持つと考えられる。

國領（1999）は「プラットフォームとは、第三者間の相互作用を促す基盤を提供するような財やサービスのことであり、それを民間のビジネスとして提供しているのが、プラットフォーム・ビジネスである」と主張している。一方、イアンシティ・レビーン（2007）はプラットフォームとは「エコシステムのメンバーがアクセスポイントやインターフェイスを介して利用可能となる、一連のソリューションである」と定義している。Cusumano（2004）は、プラットフォームと言う言葉は「ひとつのシステムが一社またはそれ以上の企業が製造するパーツで成り立っているとき、このようなシステムの核として機能し、そのときにこそ価値が最大化するような基盤製品のこと」を意味するとしている。また根来・加藤（2006）では、プラットフォームとは、「階層

---

77 國領（1995）には、今井・國領（1994）と同じ定義の他に、次の説明もある。プラットフォームとは、「広義には第三者間の相互作用を活性化させる物理基盤や制度、財、サービス」を意味する。

的構造を持つ製品やサービスの中に存在するあるコア製品（ハードウエア・ソフトウェア）・サービスやその製品を成立させるコア技術（テクノロジー）のことである」と定義している。Rochet & Tirole (2001)と Eisenmann, Parker & Alstyne (2006)や Hagiu(2006)に代表される複数（マルチ）サイド・プラットフォーム理論では、プラットフォームを「仲介役として複数のユーザー・グループを結びつける役割」として定義している。アンドリーセン（2007）は自らのブログ[78]の中で「プログラムできるならプラットフォームである。できないなら、違う（プラットフォームではない）。」とプラットフォームの定義としてプログラミング可能であることを挙げている。このように、プラットフォームの定義は一様ではないが、基盤機能もしくはメディア機能の意味を含意していると考えられる。

本論文でのプラットフォームの定義は、根来・加藤 (2010)のプラットフォーム製品・サービスを「各種の補完製品・サービスや補完コンテンツとあわさって顧客の求める機能を実現する基盤になり、プレイヤーグループ間の意識的相互作用の場となる製品やサービス」[79]とする。この定義は製品・サービスにおけるプラットフォーム研究のふたつの流れを統合しようとするものである。以下に、説明をおこなう。

（1） 基盤機能とメディア機能

プラットフォーム製品論は、ふたつの側面を持って発展してきたという歴史をもっている。ひとつは基盤型プラットフォーム論と分類されるもので、補完製品が存在する製品を議論の対象にしてきた。例えばゲームには補完製品としてのゲームソフトが存在し、サーバーの OS にはアプリケーションが存在するので、ゲームや OS はプラットフォーム製品ということになる。

もうひとつはメディア型プラットフォーム論と分類されるもので、仲介、決済、コミュニティ機能を保有するサービスを議論の対象にしてきた。この場合は、異なるユーザーを出会わせる、コミュニケーションを媒介する、取引を媒介するなどの機能を持つサービスがプラットフォームということになる。

---

78 マーク・アンドリーセン(Marc Andreessen) は Web ブラウザー「Mosaic」や「Netscape Navigator」などを開発したことで知られる米国のソフトウェア開発者。
http://blog.pmarca.com/2007/09/the-three-kinds.html 2008/12/17

79 根来・加藤（2010）, p. 3 ならびに根来・足代（2011）, p. 14 参照。

本論文は、前者の製品論をプラットフォームの基盤機能的定義と呼び、後者のサービス論をプラットフォームのメディア機能的定義と呼ぶ。プラットフォームの基盤機能的定義は、「各種の補完製品やサービスとあわさって顧客の求める機能を実現する基盤になる製品やサービス」であり、プラットフォームのメディア機能的定義は、「プレイヤーグループ内やグループ間の意識的相互作用の場を提供する製品やサービス」である[80]。ここで、「意識的」とは、当事者が別グループの大きさや質を「意識」しているがゆえに生まれる相互作用が、少なくともひとつのグループから別のグループに対して存在するということである。

（2） 基盤機能とメディア機能の統合

基盤機能を持つ製品では、定義上補完製品が存在し、その多様性と質が該当プラットフォーム利用者にとって重要な選択要因となる。逆に利用者の数や質が補完製品提供者（補完業者）の当該プラットフォームへと惹きつける。つまり、これらのふたつのプレイヤーグループ（利用者と補完業者）は、プラットフォームを媒介に相互作用する。一方、メディア機能型プラットフォームにおいては、そのサービス自身が、異なるプレイヤーグループの相互作用を媒介することで成立している。例えば、クレジットカードにおいては、加盟店の数と質が加入者の数と質に直接影響する（相互に意識しあってプラットフォームを選択する）。実は、「異なるプレイヤーグループの相互作用」の存在は、上記した基盤型プラットフォームとメディア機能型プラットフォームに共通する性質だと考えられる。

オークションサイトは、仲介機能を持つサービスとして一般にメディア機能型プラットフォームとされる。しかし、オークションサイトにおいても、例えば出品製品の情報は定められたフォーマットでそのサイトに掲載される。この情報は、仲介機能を果たすための前提となる、補完業者（出品者）が提供する補完製品（情報）であると考えられる（出品製品自身はプラットフォームと一緒に利用されるわけではないので補完製品ではない。プラットフォームサービスの対象製品である）。実は、メディア機能型プラットフォームにおいても補完製品は存在しているのである。

---

80 同上。

意識的相互作用を可能にすることをメディア機能、製品だけでなく情報の基盤となることも基盤機能として拡張して考えれば、両機能のうちどちらかの機能がより強いことはあるが、プラットフォーム製品・サービスは、必ず基盤機能とメディア機能の両方の機能を持っているといえる[81]。

以上の考察から、本論文では、プラットフォーム製品・サービスを「各種の補完製品・サービスや補完コンテンツとあわさって顧客の求める機能を実現する基盤になり、プレイヤーグループ間の意識的相互作用の場となる製品やサービス」と定義する。

第４節　プラットフォーム戦略における階層化の概念

本節では、本論文の全体を通じて議論されるプラットフォーム製品戦略における上位下位階層の考え方、ならびに相互依存性と一方向依存性の違いが、プラットフォーム戦品戦略を論じるための重要な概念となる。よってプラットフォーム戦略における階層化の概念[82]について説明する。

コンピュータ・ソフトウェアはストレージ階層、データベース階層、アプリケーション階層、OS 階層、ネットワーク階層など階層構造を成す。階層構造とは、ある事象や認識対象の構造が、層から層へと順に積み重ねて全体を構成している状態である。階層構造を特徴づける性質は、ある階層の隣接上位階層にはふたつ以上（複数）のアイテムが存在する（もしくは存在可能である）。作成されたコンテンツが存在するものは全てプラットフォーム製品と呼び、階層を形成する。

---

81 同上。

82 ソフトウェアにおける階層化の概念や考え方は、それほど斬新なものではない。例えばソフトウェア工学（Software Engineering）において、ソフトウェアの設計や開発プロセスにおける階層的な考え方や手法は以前から存在する。階層化の利点は複雑な構成要素が独立したサブセットに分割されることにより、例えばひとつの階層の改良が他の階層に影響してしまうことを防ぐことができる。またソフトウェアの開発者が、モジュール単位でデザインや開発作業を進めることができ、ソフトウェア提供者間で互換性を提供する標準インターフェイスを定義できるなどである。
ひとたび標準インターフェイスが定義されると、それに応じた様々なソフトウェア製品が複数の開発企業から提供されるようになり、ユーザーの需要の多様性に応えることが可能になる。いわゆる「モジュール化」のメリットがもたらされる。同時にソフトウェア産業のモジュール化は、階層単位でのプラットフォーム製品を提供する企業の増加をもたらし、技術的な革新が促される。またユーザーに対しての多様性、バラエティーに富んだ要求やニーズというものに応えていくことが比較的容易となる。

（1）　上位下位階層の特徴

コンピュータ・ソフトウェアの階層構造を特徴付ける性質には、階層同士の関係が入れ子[83]（nested）になっているという点がある。仮に下方から上方に順に積み重ねた階層構造の場合、下位階層は上位階層を入れ子にしている。逆に上位階層は下位階層によって入れ子にされているといえる。

もうひとつの特性として「上位・下位階層間の非対称依存特性」がある。これは下位階層に及ぼされた影響は上位階層に及ぶが、その逆は起こらないという性質である。言い換えれば、上位階層は下位のダメージを受ける。例えば下位階層がなんらかの理由で機能しなくなると、上位階層も機能不全となる。逆に下位階層は上位階層のダメージの影響を受けることはない（図１－１）。

またコンピュータ・ソフトウェアでの階層化は、下部階層を抽象化する（隠蔽する）特性を持つ。これにより直接 CPU やハードウェアのことを気にすることなく、開発者は OS とのインターフェイスに集中することが可能になる。

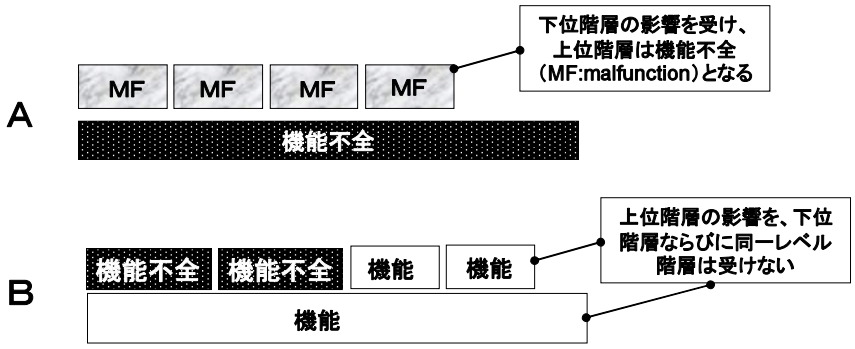
１）　相互依存性と一方向依存性

階層は特性として「上位・下位階層間の非対称依存特性」を持つ。この非対称依存特性を端的に表わす状況として、上下どちらかの階層が機能不全になった際の隣接階層への影響を取り上げ、以下に説明する。以下の図１－１においてＡは下位階層が機能不全であり、その影響で隣接する上位階層のプラットフォーム製品はすべて機能しなくなる。言い換えれば、上位階層にあるプラットフォーム製品は下位階層のプラットフォーム製品の影響を直接受ける。よって下位階層は上位階層をコントロールできるという意味で、優位な立場にある。一方、Ｂでは上位階層にある一部のプラットフォーム製品が機能不全になっても下位階層は直接の影響は受けない。引き続き機能することが可能で、加えて同一レベルの他の独立したプラットフォーム製品も問題なく機能できる。この状況を考えても、下位階層は上位階層に対し優位な立場にあるといえる。

---

83 ある構造の内部に、別の構造が包含されていること。本論文では、アプリケーションの階層が OS という別の機能の階層上で動く場合、上位階層が下位階層に入れ子にされている状態と考える。この時、アプリケーションは OS の掌上でしか機能できない。ちなみに構造化プログラミングにおけるプログラムを構築する手法のネスティングの概念とは異なる。

図１－１：上位・下位階層間の非対称依存特性



これらのことから、ソフトウェアは補完製品に対して、階層性ならびに一方向性依存性をもつ。依存とは下位階層がないと上位階層が動かない（機能しない）ということであり、逆は不成立の場合が「一方向的依存」である。 この一方向依存性は、本論文で取り上げる様々な階層戦略を論じるベースとなる。例えばプラットフォーム包囲論（後述）における上位階層と下位階層の関係など、この一方向依存性を前提とした戦略の議論となる。

２）モジュール化との違い

類似のものに「モジュール化」がある、モジュールと階層の関係は対称依存特性の点で以下のように表される。

・階層はモジュール構造の部分概念
・モジュール同士の関係の特殊ケースが「階層」
・階層とは上位・下位階層間の非対称依存特性を備えたモジュール

言い換えれば、モジュールはそれが動くための依存関係を特定しない概念である。ちなみにモジュール化とはそれぞれ独立に設計可能で、かつ、全体として統一的に機

能するより小さなサブシステムによって複雑な製品や業務プロセスを構築する[84]こととする。

3）階層化による下部隠蔽の役割

またコンピュータ・ソフトウェアでの階層化は、下部階層を抽象化する（隠蔽する）特性を持つ。例えばOSの第一の目的はアプリケーション・ソフトウェアを動作させることであるが、このためのインターフェイスはAPI（アプリケーションプログラミング・インターフェイス）[85]とABI（アプリケーションバイナリ・インターフェイス）[86]である。OSのAPI/ABIはアプリケーション開発者から下部階層であるCPUやハードウェアを抽象化する役目があり、これにより直接CPUやハードウェアのことを「隠された情報」として気にすることなく、開発者はOSとのインターフェイスに集中することが可能になる[87]。

ちなみに抽象化とは、具体化の反意語で、対象から注目すべき要素を重点的に抜き出し、それ以外は無視する方法である。言い換えれば、関心対象に焦点を当て特定の形式で表現し、本質的なものだけを強調して抜き出すことである。

第５節　おわりに

本章では、「プラットフォーム戦略の黎明」と題して、本論文の議論の前段階とでもいうべき、コンピュータ・ソフトウェアの階層化の時系列整理、プラットフォームの定義、プラットフォーム戦略における階層概念について論じてきた。

本論文の全体を通じて議論されるプラットフォーム製品戦略とは、プラットフォーム製品提供者が、補完業者がそのプラットフォーム製品向けの補完製品やサービスの提供をおこなってくれるよう促し、自社プラットフォーム製品の市場での普及を図る戦略である（後述）。例えば、OSやゲームマシンで大きなシェアの獲得に成功すれば、その上で動く補完製品としてのアプリケーション・ソフトが出現する。またその豊富

---

84 Baldwin & Clark（1997）, p. 84 参照。
85 OSやミドルウェア向けのソフトウェアを開発する際に使用できる命令や関数の集合のこと。また、それらを利用するためのプログラム上の手続きを定めた規約の集合。
86 命令体系が同じマイクロプロセッサを搭載したコンピュータ同士で、アプリケーション・ソフトを書き換えなくても動作するように、機械語レベルでの互換性を保証するための規約。
87 Baldwin & Clark（2000），（邦訳）p. 76 参照。

なアプリケーションの出現によって更に「プラットフォーム＝基盤」となる製品の普及が進むなど、相互にグループ間のネットワーク効果を引き出しながら普及を推し進める戦略が、プラットフォーム提供者がとる代表例である。

プラットフォーム戦略は、ソフトウェア・ビジネスに限ったことではない。例えば、クレジットカードやオークションサイト、イエローページにショッピングモールなど戦略の応用範囲は広い。また、ソフトウェアのなかでも、PC やサーバーだけでなく、携帯デバイスなど目的や用途別によってさまざまな形態のプラットフォームが存在している。その共通点は、プラットフォーム提供者が補完業者を誘引することによって、エコシステムを成長させプラットフォームの拡販を目指す点である。

本論文ではコンピュータ・プラットフォームに対象を絞った議論を展開する。そのためサービスのプラットフォームとは区別するためソフトウェアによって創られたプラットフォーム製品に議論の対象を限定する。

第２節では、はじめにコンピュータ・ソフトウェアの階層化の時系列整理をおこなった。今日でこそコンピュータは生活にも業務にも標準的に使われているが、現在のようなソフトウェアの階層構造を形成するまでの起源と変遷に関してまとめた。 プログラムの誕生からプログラム内蔵式コンピュータの誕生、そしてプログラミング言語の発達ならびに OS の誕生へと繋がり、プログラム技術の進歩とアセンブリ言語や高級言語としての FORTRAN と COBOL について説明した。やがて IBM 社の OS/360 による OS の誕生により、コンピュータ・ソフトウェア産業の創生とソフトウェア・ビジネスの誕生、それにより産業構造変化へと繋がっていく経過を説明した。また、コンピュータ・ソフトウェアが、OS の誕生ならびにコンピュータ・ソフトウェア産業の水平型構造の発展とともに、階層化してきた歴史を整理した。

第３節では、プラットフォームの定義について論じた。根来・加藤（2010）の製品・サービスにおけるプラットフォーム研究のふたつの流れと、基盤機能とメディア機能を統合する定義を論じた。

第４節は　プラットフォーム戦略における階層概念について、上位下位階層の特徴や相互依存性と一方向依存性の概念、モジュール化との違いや階層化による下部隠蔽の役割について論じた。本論文の全体を通じて議論されるプラットフォーム製品戦略では、上位下位階層の考え方、ならびに相互依存性と一方向依存性の違いがプラットフォーム戦品戦略上の重要な概念となることを説明した。

第２章

プラットフォーム戦略の先行研究レビュと課題

第１節　はじめに

本章では、コンピュータ・ソフトウェアのプラットフォーム戦略論に関する先行研究を俯瞰し、主な論文の概要を紹介するとともに、先行研究の課題を指摘する。

第２節では、先行研究レビュとして、コンピュータ・ソフトウェアの競争戦略の領域と、プラットフォーム製品の競争戦略の領域からレビュをおこなう。第３節では、第２節のレビュによって導出した課題について整理し、階層介入戦略ならびにプラットフォーム製品戦略の先行研究の課題の所在を指摘する。

第２節　先行論文レビュ

本節では先行研究として、コンピュータ・ソフトウェアの競争戦略の領域から、１）コンピュータ・ソフトウェア産業の階層的構造変化に関する研究、２）コンピュータ・ソフトウェア企業の分野を特定しない経営戦略に関する研究、３）コンピュータ・ソフトウェア企業の分野を特定する経営戦略に関する研究の３点に分けてレビュをおこなった。またプラットフォーム製品の競争戦略の領域として、１）プラットフォーム・リーダーシップに関する研究、２）プラットフォーム製品の階層戦略に関する研究、３）プラットフォーム製品のドミナント化ならびに WTA に関する研究の３点からレビュをおこなった。

（1）コンピュータ・ソフトウェアの競争戦略の領域からの示唆

コンピュータ・ソフトウェアの競争戦略の領域における、コンピュータ・ソフトウェア産業の階層的構造変化に関する研究では、国領(1999)、Cusumano(2004)、ハジウ(2006)がある。そこでは、主に1960年代のIBM社のソフトウェア分離販売から産業が階層化してきたプロセスを論じている。コンピュータ・ソフトウェア企業の分野を特定しない経営戦略に関する研究では、末松・ベネット(1996)、山田(2000)、Evans, Hagiu & Schmalensee(2006)、Foley(2008)、Yoffie, Hagiu & Slind(2009)などがある。コンピュータ・ソフトウェア企業の分野を特定する経営戦略に関する研究での、オープンソース・ソフトウェア（以降OSS）ビジネスに関する研究では、O'Reilly(1999)、DiBona, Ockman & Stone(1999)、Young(1999)、Raymond(1999)、Torvalds(1999)、佐々木・北山(2000)、末松(2002)、末松(2004)、Cusumano(2004)などがある。そこでは、開発者にとって開発のインセンティブやコミュニティの力、OSSがもたらすビジネスインパクトに関して論じられている。

1）コンピュータ・ソフトウェア産業の階層的構造変化に関する研究

コンピュータ・ソフトウェア産業の階層的構造変化に関する研究では、国領(1999)、Cusumano(2004)、ハジウ(2006)がある。コンピュータ・ソフトウェア産業の階層的構造変化の観点で先行研究をレビュする理由は、プラットフォーム戦略理論の前提として、基盤となるプラットフォーム製品と、その上にある補完製品がそれぞれ階層構造を成していることが必要であるからである。仮に市場で独占状態の１社のみが、ハードウェアや各種機能をもつソフトウェアの全てを統合型で提供する場合、産業内での階層構造は形成されにくい。各プレイヤーから階層毎にプラットフォーム製品がオープンな環境において提供され得るという前提が、産業の階層構造化を促進すると考えられる。

国領(1999)は、オープン・アーキテクチャ戦略[88]のなかで、モジュール化とオープン・アーキテクチャの採用で、独立した会社が開発した製品を次々と結合していける

---

88 企業が自社のもつ情報を積極的に社外に公開していくことで顧客の利便性を上げ、最終的に自社の利益を上げようとする新しい企業戦略のことを指す。製品アーキテクチャを公開して事業の構成要素をモジュール化し、外部との効率的な垂直ネットワークを構築するビジネスモデルである。国

仕組みが社会に出来上がったと指摘している。それにより小さな企業でも大きな投資を避け、真に付加価値を高められる部分にだけ集中する戦略をとることによって、自社製品をデファクト・スタンダード化することに成功できる。それにより急速に大企業に成長していけることが、モジュール化がもたらす効果として挙げられる。こういったプロセスは情報産業を超えて広く見られるトレンドである。加えて、コンピュータ産業が「垂直囲い込み型」から「水平展開型」への経営構造の転換を劇的に経験したと指摘し、水平展開型の戦略においては、全顧客のニーズの一部を満たすことが求められ、従来の垂直囲い込み型経営のタテ方向の展開に対して、ヨコ方向への展開が重要であるという関係についても論じている。

Cusumano(2004)によれば、IBM 社のソフトウェアとハードウェアの分離販売の方針が、IBM 互換用ソフトウェアを開発したいと考えていた新しい世代の起業家たちの意欲を掻き立てることとなった。米国ではそれまでほとんど存在しなかったソフトウェア企業が、1972 年には 81 社[89]が新たに出現した。IBM 社は主にアプリケーション・ソフトを分離販売したので、ソフトウェア業界への新規参入者にとっての新しいビジネスチャンスはアプリケーション・ソフトの分野が多かったと説明する。

ハジウ(2006)は「（IBM 社による自社メインフレームのハードウェアとソフトウェアを）切り離し、別の産業とするアンバンドル化が進んだため、IBM 社用にプログラムを提供するソフトウェア産業が成長した。その後、マシンが小型化しワークステーションや PC の開発が進むにつれ、デジタル・リサーチ社、アップル社、マイクロソフト社、ノベル社などの独立系ベンダーによって供給される OS が産業のカギとなる（後略）」と指摘している。また、ソフトウェア・プラットフォームが何を起源として進化していったかについて論じている。コンピュータ関連業界では、ソフトウェアのプラットフォームがベースになり、その進化を促進させてきた。その経済的・技術的な要因を３つ提示している。ひとつは、モジュール化（製品やサービスの構成要素を分割し、分業すること）。ふたつめは、デバイスといった製品自体の複雑性が今後も増していくような技術的な革新や進歩、３つめは消費者の多様性に富んだものへの

領（1999）は、自社や自社の取引先にとって有利な情報だけでなく、不利な情報までオープンにすることで、顧客の信用や信頼を得ることが重要であると主張する。

89 Cusumano（2004），（邦訳）p. 143 参照。

要求の高まりを挙げている。これら３つの要因により、産業界はこれまでの垂直統合から垂直非統合へと移行してきている。今後もこの方向性は続くであろうと説明する。

２）コンピュータ・ソフトウェア企業の分野を特定しない経営戦略に関する研究

コンピュータ・ソフトウェア企業の分野を特定しない経営戦略に関する研究では、末松・ベネット(1996)、山田(2000)、Cusumano(2004)、Evans, Hagiu & Schmalensee(2006)、Foley(2008)、Yoffie, Hagiu & Slind(2009)などがある。末松・ベネット(1996)、山田(2000)がJavaやサン・マイクロシステムズ社（以降サン社）の研究、Cusumano(2004)、Evans, Hagiu & Schmalensee(2006)がソフトウェア企業の競争戦略を分析、 Foley(2008)、Yoffie, Hagiu & Slind(2009)らが、本論文での重要なプレイヤーであるマイクロソフト社やヴイエムウェア社について分析している。コンピュータ・ソフトウェア企業の企業戦略は他にも広範囲に研究があるが、レビュでは本論文に強く関係するものだけを取り上げる。

末松・ベネット(1996)は、1995年に発表されたコンピュータ言語Javaに関して注目し、その誕生をインターネット市場におけるゴールド・ラッシュの幕開けと唱え、その想定されるIT業界への影響の大きさに注目している。そこでは、JavaがWWW、WWWブラウザー、電子貨幣、電子決済などの要素技術を統合することにより、デジタル通信革命がおこりつつある。これまで、一過性のブームともとられていたインターネットならびに情報ネットワークのばらばらな方向性を、ひとつの方向に確定させた。その代表的な牽引役がJavaであると説明している。

山田(2000)は、サン社のコンピュータ言語Javaにおけるオープンの考え方と、マイクロソフト社の企業戦略とを対比してクローズドと定義し、ふたつの企業の相容れない普及政策や経営思想について論じている。通常、自前主義の経営環境では市場はひとつの技術に収斂する。しかしサン社はオープン・スタンダードによってルールに基づいた自由な競争が起きるようにすることを戦略としていた。そしてこのオープン対クローズドの争いは、単なる企業戦略や製品の競合というものにとどまらない思想的な戦いであり、争いは永遠に終わることはないとしている。

Cusumano(2004)は、ソフトウェア企業の競争戦略を、成功のための方策や開発のベストプラクティスの観点から広く分析している。そこでは、ソフトウェアは、一度完

成すれば限界コストは非常に小さい。顧客に一旦購入されれば、ハードのように摩耗したり老朽化することはないため、ソフトウェア提供者は、製品に新しい機能を追加するなどして需要を喚起し続ける必要がある。今日のソフトウェア企業において、徐々に売上高に占めるサービス（コンサルティング、カスタマイズ、メンテナンス、サポート）の割合は、今後高くなっていくと予測される。サービスは、ソフトウェアのライセンスに比べると労働集約的で利幅が低いが、継続して安定した収入が入るというメリットがある。ソフトウェア提供企業は、今後「ライセンスとサービス」の組合せで商売をする「ハイブリッド型」になっていくと説明している。

Evans, Hagiu & Schmalensee(2006) は、様々なソフトウェアを分析し、その潜在的な力に着目している。消費者側から見えるものではハードウェアやコンテンツがあるが、それを駆動しているのはソフトウェアのプラットフォームであり、消費者側からは見えにくいが重要な働きをするエンジンである。そしてそのソフトウェアが形成するプラットフォームがパソコンや携帯電話のビジネス世界においてすべての受け皿としての役割を持ち、これまでの発展の重要な要素であったことを指摘している。

Foley(2008) はビル・ゲイツが退任した後のマイクロソフト社の様々な製品やそれに関する技術について、将来的な可能性に関して予測している。 Yoffie, Hagiu & Slind(2009)はヴイエムウェア社の経営や製品の戦略について、インタビュや内部情報を交えながらケース分析をおこなっている。

3）コンピュータ・ソフトウェア企業の分野を特定する経営戦略に関する研究

コンピュータ・ソフトウェア企業の分野を特定する経営戦略に関する研究のひとつとして OSS ビジネスを取り上げる。OSS の IT 業界への影響に関する研究では、O’Reilly(1999)、Raymond(1999)、末松(2004)、Cusumano(2004)が、プログラマーの開発インセンティブに関する研究では、O’Reilly(1999)、DiBona, Ockman & Stone(1999)、末松(2002)が、OSS のビジネス上の商機に関する研究では、Torvalds(1999)、Young(1999)、佐々木・北山(2000)、Cusumano(2004)がある。

OSS の IT 業界への影響について は、O’Reilly(1999)、Raymond(1999)、末松(2004)、Cusumano(2004)が論じている。O’Reilly(1999)によると、オープンソースやウェブは、コンピュータ業界全体のパラダイムを変貌させつつある。これにより OSS に関係

するビジネスの形態も変化する。例えばIBM社が支配的な地位にあった頃はハードウェアこそが最有力商品であり、コンピュータ・ビジネスに参入しようとする際の敷居は高かった。当時は大半のソフトウェアはハードウェア・ベンダーか、その傘下にあるソフトウェア・ベンダーだけが開発していた。しかし、PCが開発のプラットフォームとして利用できるようになると、Unixのようなオープンなシステムが開発環境として選択され、業界のルールが変わった。突如として業界参入への敷居は低くなった。このような変化と同じようなことが、OSSの普及によるコンピュータ業界で起こりはじめており、業界参入への道を広げていると説明する。

Raymond(1999)は、ソフトウェア業界は、OSS普及によって大きな変化をおこしていると指摘する。Raymond(1999)は「ブルックスの法則」[90]を取り上げ、プログラマーの人数がN倍に増えると、こなせる作業量もN倍となり、複雑さとバグの発生のしやすさはNの２乗となる。つまり大勢の人間が開発に関わると、その開発の成果物は不安定になって収集がつかなくなるという考えを、Linuxのコミュニティはみごとに打ち破った。そして、自らのLinuxのコミュニティの観察から「伽籃とバザール[91]」という論文を発表する。Raymond(1999)はOSSをいかにビジネスのツールとして活用していくかの提言をおこなっている。また、今後のOSSの将来性についても説明している。そこでは、オープンソース方式で開発する人たちは今後、爆発的に増加し続ける。Linuxがオープンソースの主流となり、商用Unixはとって代わられると予想している。

末松(2004)は、オープンソースが世界に及ぼす影響について論じている。次世代ITがもたらす社会像、知的情報によりリードされる世界像が、オープンソースの隆盛を通じてその本質を見定めることができるとしている。そこでは、孤立でも排他でもない、支配や独占でもない、相互依存、相互理解、相互協調が主流となり、知識や情報、ノウハウをインターネットを介して相互に活用し統合し、創発、共創する世界となると説明している。

---

90 フレデリック・ブルックス（Frederick Phillips Brooks, Jr.）によって唱えられた、遅れているソフトウェア・プロジェクトへの要員追加は、さらにプロジェクトを遅らせる結果になるという、ソフトウェア開発のプロジェクトマネジメントに関する法則。ケリー・アスプレイ（2006），p. 201参照。

91 伽藍(がらん)とバザールは、レイモンドによって書かれたOSS開発に関する論文。OSSの成功例としてLinuxの開発手法（バザール方式）と、これまでOSSでよく利用されてきた開発手法（伽藍方式）を、Fetchmailというソフトウェアを自身がバザール方式で開発した経緯をベースに両方式の特徴を考察したもの。

Cusumano(2004)は、OSS の肯定的な影響として、OSS のような活動が、世界中の多くのプログラマーが広く分散し、漫然とした協調性を保ちながらイノベーションのプロセスに参加できるという行動基準を広めてきたと指摘する。

プログラマーの開発インセンティブについて、O’Reilly(1999) 、DiBona, Ockman & Stone(1999)、末松(2002)が論じている。O’Reilly(1999)は、OSS の開発ならびに開発者のメリットを以下のように指摘する。OSS は無料で試せる。自分だけのカスタムバージョンを無料で作る。自由な立場にある多くの人たちがソースコードを入手して吟味できる。ある機能が気にいらない場合には誰でも修正・変更・削除できる。実装が気にいらなければ、別の方法で実装することもできる。変更された部分のソースコードがコミュニティに還元されれば、多数の人がそれをすばやく採用して使用できるなどのメリットがある。

開発者たちは、ビジネス的な野心でソフトウェアを開発しているわけではなく、目の前の問題を解決することに専念している。インターネット上の分散型開発というパラダイムゆえに、OSS はユーザーの手で書き加えられる新しい機能によって、設計されるのと同じ速度で「進化する」のであると説明している。

DiBona, Ockman & Stone(1999)は、O’Reilly(1999)と同様、プログラマー開発インセンティブについて触れている。そこでは OSS の広がりによって、プログラマーの労働と報酬の関係において、新しい経済モデルや考え方が登場していると指摘する。プログラマーという人種は、直接的な見返りだけを追求しているのではなく、そのプロジェクトや仕事にやりがいを感じるときに忠誠心を発揮する。プログラミングをすることを通じて自分の仕事が正しく評価されることを望んでいる。プログラマーは自分の書いたプログラムが他のプログラマーと分かち合えなければ、自分のやった仕事の本当の価値を見出すことはできない。プログラマーの業績はそのプログラマーが書いたプログラムが共有されればされるほど、価値のあるものとなる。DiBona, Ockman & Stone(1999) によれば、オープンソースのプログラムを作成している多くのプログロマは、プログラミングをすることは、ある種の達成感・充実感を味わうための手段であると考えている。知的作業をおこなうことで得られる究極の満足感や、完璧なルーチンを書きあげたときに感じる高揚感を得るために、プログラマーはプログラミン

グをする。言い換えればプログラマーは楽しくてプログラミングしている人達であると説明している。

末松(2002)は、OSS のボランティアの求心力について論じている。末松(2002)は、Linux を Windows に対抗する OS のボランティアによる世界的な制作活動であると指摘している。それには、いくつかの強い求心力が働いている。そこでは、①マイクロソフト社に対する反発による求心力、②若年層のアンチ体制的、アンチ大企業的な価値観による求心力、③新しいボランティア活動に対する連帯感、④価値観をひとつにする仲間による世界的かつ壮大な創作活動の４つを挙げている。Linux 開発活動の成功は、全世界の見知らぬ４万人以上の人間によるボランティアの協調活動である。そこでは、なぜ４万人以上の見知らぬプログラマーが混乱することなく協調して活動ができたのか、なぜ無報酬で自発的に開発に貢献したかという疑問が生まれる。末松(2002)は、前者は、モジュール&インターフェイス方式を採用し、グループをモジュールごとに管理する。これにより、同時並行的な開発と、複数のグループによる開発の競争原理を利用することが可能となったこと。後者は、アンチマイクロソフトと、開発者の間の新たな価値観が生まれていることが理由であったと説明している。

OSS のビジネス上の商機に関する研究では、Torvalds(1999)、Young(1999)、佐々木・北山(2000)、Cusumano(2004)が論じている。Torvalds(1999)は Linux の創始者として周知であるが、OSS のビジネス上の商機に関して、Linux の事例を取り上げ、成功の理由について以下のように論じている。Torvalds(1999)は Linux の成功の理由は、優れた設計理念と開発モデルに基づいて開発され、それゆえに移植性に優れ、利便性のあるシステムとして構築することが容易であったことだと論じている。一方、Young(1999)は Red Hat Linux の事例を参照し、その成功をマーケティング観点からとらえている。Linux のビジネスでは、自分で自由にできる OS が提供されることにより、自分の使う OS を自分でコントロールしたいという欲求に応えることで、大きな利益を生み出してきたと説明する。言い換えれば、Linux は信頼性や使いやすさ、システムの堅牢さ、ツールの多さではなく、ソースコードがオープンであることが支持されている。それによりソースコードを都度ベンダーの承認を求めることなく、自由に自分の好きな目的のために使用できる点が支持されてきた一番の理由であると指摘する。

佐々木・北山(2000)はOSSのLinuxの事例を始めとして、OSSがもたらすビジネス的側面に関してコミュニティ・アライアンス戦略という言葉を使って説明している。そこでは、企業側がコミュニティ側を意のままにコントロール可能と考えるのは適切ではなく、互いを尊重した対等な関係を基盤として、相互の利益という意味を込めた関係を構築していくアライアンス戦略が重要であると説く。そういった意味では、OSSに象徴されるコミュニティという、ともすると制御困難な大きな力といかに上手に付き合っていくかが、企業にとってのビジネスの戦略上重要な事であることを示唆している。

Cusumano(2004)は、OSSが新たなビジネスの商機を生みだしてきたと説明する。Red Hat社、VAソフトウェア社、SCOグループ、ターボ・リナックス社はLinux市場で新たに誕生した企業である。これらの企業は特別なインストールプログラムや開発ツールなどのユーティリティとアプリケーションなどを組み合わせ、サービスやパッケージとセットで販売する機会を得たと指摘する。

（2） プラットフォーム製品の競争戦略の領域からの示唆

プラットフォーム製品の競争戦略の領域において、プラットフォーム・リーダーシップに関する研究ではGawer & Cusumano (2002)やIansiti & Levien(2004)ならびに、根来・加藤(2006)がある。そこでは、プラットフォーム提供者の補完業者へのインセンティブを論じるエコシステム論が論じられている。プラットフォーム製品の階層戦略に関する研究では、Katz & Shapiro(1985, 1986)ならびにShapiro & Varian(1999)は、階層間の相互運用性がもたらすネットワーク効果の理論を展開している。Rohlfs(2001)は間接ネットワーク効果を「補完的なバンドワゴン効果」と同義として、その効果に注目している。また、３者間構造をとるプラットフォーム仲介ネットワーク（platform-mediated networks）の考え方が存在し、Rochet & Tirole(2003)、Eisenmann, Parker & Alstyne(2006)やHagiu(2006)は、プラットフォームを仲介役として複数のユーザー・グループ（階層間）を結び付ける役割として定義している。加えて、Eisenmann, Parker & Alstyne(2007)では、階層バンドルの概念で「プラットフォーム包囲論」を論じている。プラットフォーム製品のドミナント化ならびにWTAに関する研究では、Eisenmann(2010)、根来・加藤(2010)がある。

1） プラットフォーム・リーダーシップに関する研究

Gawer & Cusumano(2002)によると、プラットフォーム・リーダーシップ[92]においてリーダーは４つのレバーを用い、補完業者をコントロールすることが重要であると説いている。４つのレバーの概要とは、レバー１：企業の範囲、レバー２：製品化技術、レバー３：外部補完業者との関係、レバー４：内部組織である。プラットフォーム・リーダーシップのめざすものは、自らの産業において、イノベーションの方向性に多大な影響を及ぼし、それゆえに補完業者を生み出し活用する、企業と顧客のネットワーク、すなわち「エコシステム（産業生態系）」にも強い影響力をもつことであると説明する。自社の主力製品に対する需要が、数ある他社製の補完製品に依存しているということ、それゆえ自社の命運は他社の意思決定や行動に委ねられているということ、これらを再認識することが、プラットフォーム・リーダーになるために、まず考えなければならないことである。プラットフォーム・リーダーと予備軍は、サードパーティ企業が補完的イノベーションを追求したくなるようなインセンティブを維持しつづけなければならない。結論として、プラットフォーム・リーダーシップの本質は、１企業の事業展開、１製品あるいは１部品の技術仕様といったことをはるかに超えて広がるビジョンからはじめることである。もし企業が協働しリーダーに追随するなら、産業生態系の全体はその部分の合計より大きくなるといえるようなビジョンである。言い換えれば、プラットフォーム・リーダーがする意思決定および、しないという意思決定によって、補完業者がおこなうイノベーションの程度と種類に大きく影響を与える。Gawer & Cusumano(2002)によれば、このことこそ、プラットフォーム・リーダーシップが何かについての全てであると論じている。

Iansiti & Levien (2004)が唱えるキーストーン戦略とは、エコシステム全体の健全性を高めることを通じて、キーストーン企業自らの持続的なパフォーマンスを高め

---

92 Gawer & Cusumano (2002)によれば、プラットフォーム・リーダーシップとは「広範な産業レベルにおける特別な基盤技術の周辺で、補完的なイノベーションを起こすように他企業を動かす能力である。」としている。「プラットフォームとはさまざまな企業によって生産された製品やサービスの１つのシステムの中に存在する、あるコア製品。」「コア製品とは、1) それ自身が進化するシステムの一部、2) 補完的な製品あるいはサービスがなければそれ自身では意味がないものである。」Gawer & Cusumano (2002)は、上記をもって「定義」としているわけではないが、上記はGawer & Cusumano (2002)におけるプラットフォーム定義に相当する説明だと考えられる。この定義にみられるように、エコシステムを通じたイノベーションの創出の中心にはコア製品（技術・サービス）がある。詳しくは Gawer & Cusumano (2002), (邦訳) p.165 を参照。

る戦略を意味する。ビジネス・ネットワークを生態系になぞらえて、エコシステム間で競争優位を獲得するためには、キーストーンとしてのプラットフォームをどのように進化させるべきか、またどのようにニッチを創出すべきかという点について議論を展開している。

そこでは、これまでのような自社の視点だけで収益モデルを考えているだけでは不十分であり、実効的な事業戦略や事業システムを描くことは困難になってきている。従来の企業戦略論で事業環境とされてきた産業構造や市場を、Iansiti & Levien (2004)は企業の内外がシームレスに結びついた「ビジネス・ネットワーク」あるいは「ビジネス・エコシステム」(ビジネス生態系)という概念で捉え、新しい視点や戦略パターンを示している。加えて、ビジネス・エコシステムを運営するキーストーン戦略や、エコシステムの価値創造に貢献するニッチ企業の戦略パターンを明らかにしている。

Gawer & Cusumano(2002)と Iansiti & Levien (2004)は、エコシステムをいかに牽引していくかという点は共通している。具体的には、Gawer & Cusumano(2002)はコアテクノロジーをテーブルの真ん中に据えてプラットフォーム提供者が４つのレバーをベースに、いかにコントロールしていくのかを主に論じている。これに対し、Iansiti & Levien (2004)は初めから役割をもったプレイヤーがそれぞれの役割を演じることによって、エコシステムの健全性が保たれるという立場を主張する。

根来・加藤(2006)は Gawer & Cusumano(2002)の研究に対し、批判的発展を加えている。具体的には、Gawer & Cusumano(2002)の事例の分析において、上位層と下位層の考え方を加えることがひとつである。それにより、プラットフォーム業者が働きかけるべき相手がより明確に検討でき、プラットフォームのオープンとクローズドの戦略がもたらすリーダーの産業生態系にもたらす影響力とコントローラビリティ（操作可能性）を可視化しやすいなどの利点が得られる。もうひとつ、５つ目のレバーとして「収益モデル」を加えることで、成果指標の一つとなる利益と直接に関係する検討ができる。収益モデルは補完業者のインセンティブを左右する大きな要因であるという意味でも、検討すべき項目となる。特に階層構造において戦略上どの部分をクローズドにしてどの部分をオープンにするのか、またどの部分で収入を確保するのかを明確にする意味のある製品・サービスにおいて有効と考えられると主張する。

２）　プラットフォーム製品の階層戦略に関する研究

一般にネットワーク効果は、「ある製品から得られる便益が、当該製品のユーザーが増えるに従って増大する性質」[93]と把えられている。ちなみに、Katz & Shapiro(1985)は、ネットワーク効果に関して「製品を使う他人の数が増加することによってもたらされるユーザーの便益」[94]と説明している。また、Kats & Shapiro（1985）同様、補完製品を含めたネットワーク効果の研究をおこなっている Rohlfs (2001)は、ネットワーク外部性（Network Externalities）を「これら（ネットワーク外部性の概念）は電気通信ネットワークを利用する製品とサービスに適用される。利用者の集合が拡大するにつれ、各利用者は、（製品やサービスの利用者となった）より多くの人とコミュニケーションが可能になることから便益が得られる。」[95]と説明している。

Kats & Shapiro（1985）[96]は、ネットワーク効果を２分類し、以下のように説明している（以下筆者訳）。

- 直接的ネットワーク効果：製品によってはその性質上、購入者数の直接的効果によって消費の外部性が生みだされる。例えば電話やファクシミリを購入することによる消費者の効用は、電話ネットワークに加入した他の人の数に全く依存している。こういったネットワーク外部性は、他のコミュニケーション技術にも同様に見受けられる。
- 間接的ネットワーク効果：一方で消費の外部性を生じさせる間接的効果というものがある。例えばPCを購入する人は同じハードウェアを購入する他の購入者数に関心を示すであろう。なぜならソフトウェアの総量と多様性は、すでに販売されたハードウェアのユニット数と相関して供給されると思われるからである。このようなハードウェアとソフトウェアの関係は、ビデオゲームやビデオプレイヤーなどにも見られる。

Katz & Shapiro(1986)では、これらの理論を踏まえ、技術の普及に関する戦略論を展開している。

---

93 経営学大辞典　第２版（1999）中央経済社, ｐ.676 参照。

94 Katz and Shapiro（1985）, p. 424 筆者訳。ネットワーク効果の定義として明示されているわけではないが、例えば以下のような説明がある。There are many products for which the utility that a user derives from consumption of the good increases with the number of other agents consuming the good.

95 Rohlfs（2001）, p. 8 筆者訳。

96 Katz & Shapiro（1985）, p. 424 参照。

Shapiro & Varian(1999)では、ほとんどの企業は他の企業と協力しなければ、標準を確立し互換性のあるユーザーのネットワークをまとめ上げることはできないとし、強力なネットワーク外部性を前にした場合、相互接続とネットワーク接続の戦略の立て方しだいで一大勢力を築くかそれとも失敗して苦しむかが決まってしまうと指摘している。Shapiro & Varian(1999)は、そのための標準化をとる戦いをどのようにおこなうかに関しての提言をしている。両者あるいは提携企業間で保持している資産によって、①避けようのない標準化戦争、②チキン（きも試し）ゲーム。どちら側も相手のテクノロジーよりも優れていると主張するが最後には戦いを避けるため譲歩する、③勝負にならないゲーム。勝ってもよいと考える強いチームと休戦交渉を望む弱いチームとの戦い、の３つの基本形態のうちのどれかの形になる。標準化ゲームにおける手に入る報酬＝業界全体の付加価値×業界の価値に対するシェア(Your reward = Total value added to industry × your share of industry value)であるとし、各企業にとってもっとも大切な課題は、その価値の増加分からどれだけを手に入れられるかであると説明している。

Shapiro & Varian(1999)によれば、標準化戦争では、開放化対コントロールのトレードオフが発生し、自分のおかれている状況によって開放的な方法をとるのか、それともシステムを独占してコントロールを続けようとするのかを判断しなければならない。そして標準化戦争を勝ち抜くための能力は、次に挙げる７つの中核資産を持っているかどうかで決まる。①ユーザー設置ベースのコントロール、②知的財産権、③革命を起す能力、④創業者利益、⑤製造能力、⑥補完製品分野での影響力、⑦ブランド名と名声、であるとしている。

Rohlfs(2001)は、本論文のプラットフォーム製品戦略に関しての示唆をもたらすと考えられる多くの点を論じている。相互連結（interconnection)という用語はｅメールを含むデータ・コミュニケーション・システムの相互接続にも用いられ、その場合の重要な問題はネットワークの加入者が、別のネットワークの加入者とコミュニケーションできるかどうかということである。これに対し、コンピュータ・システムの相互連結の重要な問題は互換性である。具体的には、あるシステムのために書かれたアプリケーション・プログラムが、他の全てのシステムにおいて十分に等しく動作するということだと説明している。バンドワゴン市場の鍵となる概念は相互連結（interlinking)である。相互連結がある場合、各ユーザーはすべてのユーザーが等

しく享受するバンドワゴンの便益を受け取ることができる。相互連結のないバンドワゴン市場はひとつの供給者へと収束する傾向を持つ。相互連結されているバンドワゴン市場では、そうしたひとつの供給者に収束していく傾向を持たないと指摘する。

Rohlfs(2001)は、相互連結の短所もいくつか指摘している。ひとつは、相互連結が非常に費用のかかる可能性がある点である。また、新技術開発のインセンティブを弱める点である。すでに新技術をもつ供給者は相互連結しないことによって、競合上の優位性を得ることが出来るが、相互連結を前提としてしまえば、企業は技術的なイノベーション開発に十分なインセンティブを持たないだろう。加えて、スタートアップ問題解決のインセンティブを弱める点である。相互連結を前提とした場合、どの供給者もスタートアップ問題を解決するために、それに必要な初期の損失を引き受けるインセンティブを十分に持たないと説明している。

Rohlfs(2001)は供給者の相互連結インセンティブについて、バンドワゴン市場における既存の供給者は、新規参入者と相互連結するインセンティブをほとんど持たない。相互連結を拒否することによって、既存の供給者は市場において追いつかれることのない競争上の優位性を保持できなくなるかもしれないためだ。そして興味深いのは、相互連結が便益を与えるにもかかわらず、ある供給者はその製品について他の供給者と相互連結しないインセンティブを持つかもしれない。相互連結してしまえば、全ての供給者が完全なバンドワゴンの便益を全顧客に提供できるが、今度は顧客を巡って必死に競争しなければならないからだと指摘している。

根来・加藤(2008)は、プラットフォーム製品におけるネットワーク効果の概念について再検討をおこなっている。根来・加藤(2008)によると、ネットワーク効果の研究においては多くの場合、通信産業のネットワークの大きさ(ネットワーク加入者の数)がユーザーに与える影響を意識して議論がおこなわれてきた。根来・加藤(2008)ではまず、このネットワーク加入者の「数」だけでなく、特定個体間のアクセスの「価値」(アクセスの頻度と重要度)を考慮したネットワーク効果の把え方を提案している。ネットワーク効果の研究においては、Kats & Shapiro (1985)やRohlfs (2001)にあるように、通信産業のネットワークの大きさ(ネットワーク加入者の数)がユーザーに与える影響を意識した議論がなされてきた。このネットワーク加入者の「数」とともに、根来・加藤(2008)ではネットワーク効果を評価するための追加概念として、特

定個体間のアクセスの「価値」（アクセスの「頻度（frequency）と重要度(weight)」）を考慮すべきことを提案している。詳しくは付録 C を参照のこと。

ネットワーク効果概念の再検討のもうひとつは、階層構造をもつプラットフォーム製品におけるネットワーク効果の分類である。プラットフォーム製品（プラットフォーム製品）に関連するネットワーク効果の代表的研究として、Katz & Shapiro（1985,1986）がある。根来・加藤(2008)が提案する分類は、Katz & Shapiro（1985）における２分類と異なるものとなる。Kats & Shapiro（1985）の２分類においては、補完製品（補完業者）とユーザーを明確に階層化しては把えていない。この２階層を明確に区別することで、根来・加藤(2008)では４分類を提起している。根来・加藤(2008)の４分類と Katz & Shapiro（1985）の２分類を比較すると、①ユーザー間ネットワーク効果：直接ネットワーク効果、②補完製品間ネットワーク効果：Katz & Shapiro（1985)では把握されない、③ユーザーの対補完製品ネットワーク効果：間接ネットワーク効果、④補完業者の対ユーザーネットワーク効果:Katz & Shapiro（1985）では把握されない、の４つとなる。

これは、プラットフォーム製品、補完製品、ユーザーという階層構造に立脚したネットワーク効果の分類である。根来・加藤(2008)は、上記の４分類によって、階層構造を持つエコシステム（プラットフォーム製品、補完製品、最終消費者からなるシステム）内に働くネットワーク効果がより明確になると主張している。

また、３者間構造をとるプラットフォーム仲介ネットワーク（platform-mediated networks）の考え方が存在し、Rochet & Tirole(2003)は、Two-Sided Markets をふたつ以上の異なるタイプの顧客を対象とするプラットフォームを持つ製品があって、その顧客が相互に依存しあい、共同で関与することでプラットフォーム価値を増大させるものと定義している。その理論を発展させるかたちで、Eisenmann, Parker & Alstyne(2006)がツーサイド・プラットフォーム戦略を唱えている。ツーサイド・プラットフォーム戦略論では２面的な市場のネットワーク効果が働き、「収穫逓増」となる。数が増えれば、利益は乗数的に増え、ユーザーはより規模の大きいネットワークに多くの対価を払う傾向があるため、ユーザー基盤の拡大につれ、収益も拡大可能である。２面的な市場ではこのように収穫逓増が約束されていることからプラットフォーム

間では激しい競争が展開される。収穫逓増が働く市場では、成熟化が進むと少数のプラットフォームによる「独占」が発生しやすいと説明している。

また、ネットワークの２面性を特徴とする市場における取引には、常に三者関係が必要となる。サイド内ネットワーク効果とはユーザーの数が増えると、そのユーザーが属するグループにとって、プラットフォームの価値が向上あるいは下落する現象を表す。またサイド間ネットワーク効果とは片方のユーザーが増加すると、もう片方のユーザー・グループにとってプラットフォームの価値が向上あるいは下落する現象であると論じている。

市場の２面性にまつわる課題にどのように取り組めばよいかについて、Eisenmann, Parker & Alstyne (2006)のツーサイド・プラットフォーム戦略では３つの課題を提起している。第１の課題は「プライシング」で、第２の課題は「１人勝ちの力学」、そして第３の課題は「包囲の危機」である。詳細は付録 B を参照のこと。

Hagiu(2006b)は、クレジットカード、ビデオゲーム、PC の OS、eBay などのオンラインマッチング、Amazon、 NTT ドコモの i-mode の事例などマルチサイド・プラットフォームの事例を数多く分析し、マルチサイド・プラットフォームはリサーチコストと複数サイドで共有する取引コストを低減する機能のどちらか、もしくは両方があることを主張する。

Eisenmann, Parker & Alstyne (2007)のプラットフォーム包囲戦略とは、Eisenmann, Parker & Alstyne (2006)にて提示された３つの課題である。第１の課題「プライシング」、第２の課題「１人勝ちの力学」、第３の課題「包囲の危機」のなかで、第３の課題「包囲の危機」をさらに論じたものである。

Eisenmann, Parker & Alstyne (2007)のプラットフォーム包囲戦略とは、プラットフォーム・プロバイダーがプラットフォームのコンポーネントの共通する部分と重なり合うユーザーの関係をテコ（leverage)にすることによって、自分自身の機能にターゲットのプラットフォームの機能をバンドルの状態で結合させ、ターゲットのプラットフォームの市場に入っていく戦略である。このプラットフォームの包囲戦略は、飛躍的イノベーションやシュンペーターの創造的破壊を必要としないプラットフォーム・リーダーシップの交代のメカニズムである。強いネットワーク効果と高いスイッチングコストによって、スタンドアローン競合の参入からは隔離された状態にある支配的な企業も、隣接したプラットフォーム・プロバイダーの複数の階層をバンドル

させる包囲の攻撃に対しては脆弱である場合があると論じている。詳しくは本論文付録 B に詳述する。

3）プラットフォーム製品のドミナント化ならびに WTA に関する研究

Eisenmann(2007)は、ネットワーク市場における WTA（Winner-Take-All）の現象について論じている。多くのネットワーク市場、例えば Fax、DVD、PC の OS などは、ただひとつのプラットフォームによって提供されている。一方、クレジットカードやオンラインゲームのように複数のプラットフォームによって提供されているものもある。その上で、プラットフォーム提供者が市場参入における戦略を策定するにあたり、単一の提供者によるプラットフォームと、複数の提供者によるプラットフォームが将来向き合うと思われる利害の結果を考慮しつつ、戦略策定するための考慮すべき項目に関して論じている。また、Eisenmann(2007)は、プラットフォーム提供者が陥りやすい、スイッチングコストとネットワーク効果の混同、マルチホーミングコストとスイッチングコストとの違いについて説明している。

根来・加藤（2010）はプラットフォーム製品の市場における WTA の現象について論じている。「技術が優れていることによって競争相手を圧倒する」という戦略は、伝統的な製品においてよく見られる。しかし、技術要因以外の要因がシェア獲得や逆転をもたらしたケースも存在する。技術要因以外の要因の働きは、プラットフォーム製品により見られると考えられる。なぜなら、プラットフォーム製品にはより多くの WTA 要因が働くからである。

根来・加藤（2010）は、プラットフォーム製品特有の「技術以外の要因」を含む WTA のメカニズを明らかにしている。そのなかで、まずは非「技術」的要因とし、以下の要因を説明する。

あらゆるビジネスの継続のための前提である「収益モデルの確立」がある。また、「先発性」の有無、「規模の優位・収穫逓増」性があること、「隔離されたニッチ市場」が存在しにくいことがある。加えて、プラットフォーム製品特有の要因として、「ネットワーク効果」が働いていること、「マルチホーミングのコストとメリット」をあげる。ネットワーク効果は、サイド内ネットワーク効果とサイド間ネットワーク効果を分けて考える。最後に、対象製品以外に WTA の製品を持つ企業が利用できる要因として、「製品シナジーの利用」をとりあげている。また、これらの要因に対して

撹乱要因がいくつか存在する。「市場の成長」や「スイッチングコスト、政府の規制」である。

根来・加藤（2010）は、次に、そのメカニズムのモデルを前提に、技術以外の要因で、格差縮小あるいは逆転を図る対抗戦略について論じている。そこでは、前述の WTA の要因に対して、逆転の戦略（WTA 要因を遮断する戦略）が存在し、以下の４つの逆転戦略に関して論じている。

ⅰ） Platform Envelopment　プラットフォーム包囲：後発企業が、先発企業のサイド間ネットワーク効果を無効化するための戦略で階層の異なる製品・サービスによる「包み込み」をおこなう。

ⅱ）Platform Bridging　プラットフォーム間橋渡し：クロスプラットフォーム製品・サービスを投入しそれまで繋がりのなかったプラットフォーム間を橋渡しすることで、ユーザーのマルチホーミングコストを下げる（あるいはゼロにする）。

ⅲ）Platform Compatibility　プラットフォーム互換：先発企業のプラットフォームのコンテンツやアプリケーションなどをそのまま使えるようにする戦略で、クローン戦略とも呼ばれる。

ⅳ）Platform Alliance　プラットフォーム連携：水平連携と越境連携の２種類がある。水平連携とは、同じ機能をもつプラットフォームが連携して、顧客基盤や補完業者基盤を共有する。越境連携とは、ネットワーク効果を持つ他のプラットフォームの顧客基盤を利用する。

非技術決定論としての、「WTA 形成」とリーダー企業への「対抗戦略」の両者を、「WTA 要因のメカニズムのモデル」を使って、関係付けて論じている。

（３）　小括

コンピュータ・ソフトウェアの競争戦略の領域における、コンピュータ・ソフトウェア産業の階層的構造変化に関する研究では、国領(1999)、Cusumano(2004)、ハジウ(2006)がある。そこでは 1960 年代の IBM 社のソフトウェア分離販売から産業が階層化してきたプロセスを論じている。コンピュータ・ソフトウェア企業の分野を特定しない経営戦略に関する研究では、末松・ベネット(1996)、山田(2000)、Evans, Hagiu

& Schmalensee(2006)、Foley(2008)、Yoffie, Hagiu & Slind(2009)などがある。コンピュータ・ソフトウェア企業の分野を特定する経営戦略に関する研究での、OSS ビジネスに関する研究では、O'Reilly(1999)、DiBona, Ockman & Stone(1999)、Young(1999)、Raymond(1999)、Torvalds(1999)、佐々木・北山(2000)、末松(2002)、末松(2004)などがある。そこでは、開発者にとって開発のインセンティブや、コミュニティの力、OSS がもたらすビジネスインパクトに関して論じられている。

そこでは、IBM 社のソフトウェア分離販売により、ソフトウェアならびにソフトウェア産業の階層化が進んできた。また、OSS の誕生により、IT 業界やソフトウェア・ビジネスは大きく変わろうとしている。OSS の開発者たちは、報酬や見返りにこだわらない労働に対する新しい価値観をもっており、ソフトウェアの拡販にそのモチベーションを効果的に利用することが、プラットフォーム製品提供者にとって戦略上の鍵になると、先行研究では示唆している。

プラットフォーム製品の競争戦略の領域において、プラットフォーム・リーダーシップに関する研究では Gawer & Cusumano (2002)や Iansiti & Levien(2004)ならびに、根来・加藤(2006)がある。そこでは、プラットフォーム提供者の補完業者へのインセンティブを論じるエコシステム論が論じられている。プラットフォーム製品の階層戦略に関する研究では、Katz & Shapiro(1985,1986)ならびに Shapiro & Varian(1999)は階層間の相互運用性がもたらすネットワーク効果の理論を展開している。Rohlfs (2001)は間接ネットワーク効果を「補完的なバンドワゴン効果」と同義として、その効果に注目している。根来・加藤(2008)は、ユーザーにとってのアクセス価値やネットワーク効果の分類に関する発展理論を論じている。また、３者間構造をとるプラットフォーム仲介ネットワーク（platform-mediated networks）の考え方が存在し、Rochet & Tirole(2003)、Eisenmann, Parker & Alstyne(2006)や Hagiu(2006)は、プラットフォームを仲介役として複数のユーザー・グループ（階層間）を結び付ける役割として定義している。加えて、Eisenmann, Parker & Alstyne(2007)では、階層バンドルの概念で「プラットフォーム包囲論」を論じている。プラットフォーム製品のドミナント化ならびに WTA に関する研究では、Eisenmann(2010)、根来・加藤(2010)を説明した。

プラットフォーム・リーダーシップでは、補完的な製品を提供する補完業者を惹きつけるためのプラットフォーム提供者の振る舞いについての理論が展開され、補完業者のインセンティブをいかにコントロールするかが戦略の成否に大きく関係する。また、プラットフォーム製品戦略においては、一貫してネットワーク効果の重要性が説かれており、グループ内ならびにグループ間ネットワーク効果を引きだすことが要となる。加えて、WTA のメカニズムにおいて、プラットフォーム製品提供者が取り得る「プラットフォーム包囲」「プラットフォーム橋渡し」「プラットフォーム互換」「プラットフォーム連携」の対抗戦略がある。

第３節　課題の所在

前節では、先行研究レビュとして、コンピュータ・ソフトウェアの競争戦略の領域と、プラットフォーム製品の競争戦略の領域からレビュをおこなった。

これまでの先行研究の貢献点は、ネットワーク効果に関しては、大量かつ広範囲な研究蓄積がある。しかし、後発の補完的プラットフォーム製品の普及戦略ならびにドミナント化のメカニズムに関するものや、後発プラットフォーム製品の一種で、レイヤースタック内の階層数変化を伴う施策である「階層介入戦略」の戦略上の示唆に関して、十分に論じられていないと思われる。本論文ではこの点を先行研究の課題の所在として指摘する。

以下、簡単に先行研究の課題をまとめると以下となる。

① 市場における同一レベル階層での競合関係にあるプラットフォーム製品のシェア争い関する先発優位や後発優位に関する研究が主である。
② 後発プラットフォーム製品から補完的な位置付けで形勢を逆転させるような研究は十分にされていない。
③ 階層介入型プラットフォーム製品のドミナント化のメカニズムに関して詳述する論文も十分でない。

第４節　おわりに

本章では、コンピュータ・ソフトウェアのプラットフォーム戦略論に関する先行研究を俯瞰し、主な論文の概要を紹介した。そこでは、先行研究として、コンピュータ・ソフトウェアの競争戦略の領域から、１）コンピュータ・ソフトウェア産業の階層的構造変化に関する研究、２）コンピュータ・ソフトウェア企業の分野を特定しない経営戦略に関する研究、３）コンピュータ・ソフトウェア企業の分野を特定する経営戦略に関する研究の３点に分けてレビュをおこなった。またプラットフォーム製品の競争戦略の領域として、１）プラットフォーム・リーダーシップに関する研究、２）プラットフォーム製品の階層戦略に関する研究、３）プラットフォーム製品のドミナント化ならびに WTA に関する研究の３点からレビュをおこなった。

その上で、先行研究の貢献点と課題を指摘し、補完関係にある隣接階層に位置する後発プラットフォーム製品のドミナント化に関する研究が十分でない点を指摘した。

第３章

プラットフォーム製品のドミナント化要因（Dominant factors）

第１節　はじめに

本章では、プラットフォーム製品のドミナント化要因について論じる。先行研究レビュにより、プラットフォーム製品のドミナント化を誘発する要因として、階層間ネットワーク効果の効用力、ブリッジングの影響力、プラットフォーム製品排除に対する抵抗力の３つを提示する。第２節で論じる階層間ネットワーク効果の効用力は、階層構造をもつプラットフォーム製品の普及拡大のための必要な要因であり、第３節で論じるブリッジングの影響力はドミナント化のための構造的な要因である。また、第４節で論じるプラットフォーム製品排除に対する抵抗力は、補完製品を提供するプラットフォーム製品提供者がレイヤースタック内に留まるための必要な要因である。またこれらの要因が、どのような先行研究からもたらされたのかを関連付けて論じる。

第２節　階層間ネットワーク効果の効用力（Utility force in network effects between the layers）

階層間におけるネットワーク効果が促進されると、ユーザーにとっての効用が高まりプラットフォーム製品としての魅力が増す。また開発者や補完業者が充実した開発者コミュニティからの支援に促され、もしくは個人的な興味や開発スキルの市場価値に刺激され、プラットフォーム製品の開発インセンティブを高める。こういったプラットフォーム製品のドミナント化を誘発する要因として「階層間ネットワーク効果による効用力」を取り上げる。

この要因の設定に関しては、顧客と開発業者のふたつのユーザー・グループがユーザー内、ならびにユーザー間ネットワーク効果を発揮することや、マルチホーミングコストの低減によるユーザーの購入の促進に依拠する。例えば、OSS コニュニティなどの開発者コミュニティに関して DiBona, Ockman & Stone (1999)、Torvalds (1999)、Perens (1999)、Raymond(1999)、佐々木・北山 (2000) らの先行研究がある。また、補完業者を含むプラットフォーム・ビジネスに関して國領 (1999)、Shapiro & Varian (1999)、ユーザーのプラットフォーム製品選択のインセンティブに関しては、Katz & Shapiro (1985, 1986)、Rohlfs (2001) のネットワーク効果やバンドワゴン効果を指摘している。加えて、Hagiu (2006)、Eisenmann, Parker & Alstyne (2006) のツーサイド・プラットフォーム理論のふたつのユーザー・グループの相乗的増加がある。マルチホーミングコストに関しては、Eisenmann, Parker & Alstyne (2007) や根来・加藤 (2010) が論じている。これにかかわる後発プラットフォーム製品提供者の操作項目 (後述) としては、プラットフォーム製品が保有するアクセス可能ユーザー数の増加やマルチホーミングコストを低減することで階層間ネットワーク効果が高まると考えられる。階層間ネットワーク効果による効用力が高いと、プラットフォーム製品がドミナント・プラットフォーム製品になる可能性が高まると推論される。

(1) 開発業者とユーザーのネットワーク効果の因果ループ

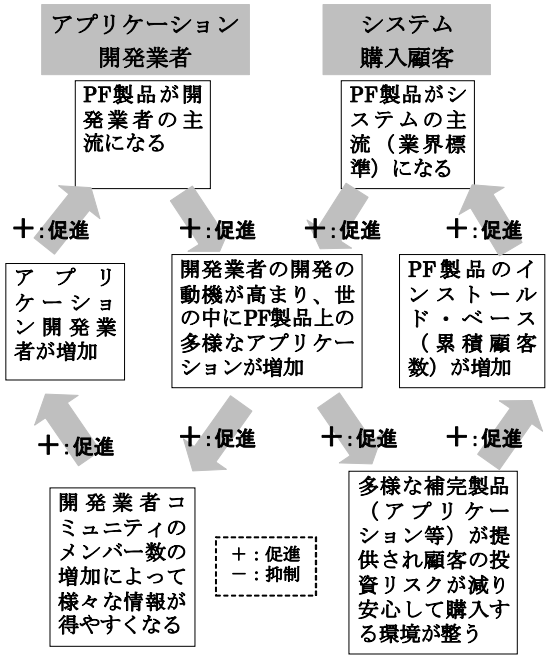
アプリケーション開発者とシステム購入ユーザーにとって、特定のアプリケーションが世の中に増加すると正のネットワーク効果の因果ループが生まれ、相乗的に特定のプラットフォーム製品を選択する動機が高まるという現象が起きる。

以下の図３-１は、本節で論じるアプリケーション開発者とシステム購入顧客のネットワーク効果の因果ループを図にしたものである。世の中にプラットフォーム製品上の多彩なアプリケーションが増加することを始点に、左側にアプリケーション開発者、右側にシステム購入顧客、のそれぞれがネットワーク効果の因果ループにて相乗効果を発揮しながら促進させていくことを表現している。ちなみに、抑制要因としてはスノッブ効果や独占の弊害が考えられるが、この図では表記していない。

アプリケーション開発者：世の中に特定のプラットフォーム製品上の多様なアプリケーションが増加すると、開発者コミュニティのメンバーが増加し、メンバー同士の開発にかかわる様々な情報が得やすくなる。これによりアプリケーション開発者がますます増加し、プラットフォーム製品が開発者の主流となる。それにより開発者の開発の動機が高まり、世の中に特定のプラットフォーム製品上の多様なアプリケーションが更に増加することとなる。

システム購入ユーザー：世の中に特定のプラットフォーム製品上の多様なアプリケーションが増加すると、ユーザーがプラットフォーム製品選択の際、「投資したものが無駄にならない」という特定のプラットフォーム製品を安心して購入する環境が整う。これによりプラットフォーム製品のインストールド・ベース（累積顧客）がますます増加し、プラットフォーム製品が主流（業界標準）となる。それにより、特定のプラットフォーム製品上の多様なアプリケーションが更に増加することとなる。

図３－１：開発者とユーザーのネットワーク効果の因果ループ



1）開発業者にとってのプラットフォーム製品としての魅力

開発業者（開発者）にとって数あるプラットフォーム製品のなかで、習得する開発スキル（言語やAPIの習熟）を選択するのは、大きくふたつの理由がある。ひとつは、開発業者が言語やAPIの習熟自体に興味をもち探究心を擽られるということ。もうひとつは将来的に主流になる開発スキルを習得することで、エンジニアとしてのキャリアを形成していくことに意欲を感じるためである。

この要因の促進のためのプラットフォーム製品提供者がとり得るアクションとしては、開発業者が開発意欲を掻き立てられるようなコミュニティの支援や安価な開発ツールの提供、開発者同士が意見交換できるようなイベントやシンポジウムなどの場を提供するなどがある。こういったことを通じてアクセス可能ユーザー数の増加を図ることが重要である。

また、開発スキルの習得におけるマルチホーミングコストの低減を図ることも重要である。具体的には競合する既存の言語などに似たものとすることで、開発業者の習熟の負担を軽くすることなどがある。

2）ユーザーにとっての効用力

ユーザーにとって、階層間ネットワーク効果の効用が増すことで、利用にかかわる様々な情報が得やすくなる、具体的には多くの補完製品（例えば参考書籍や利用スキルを高めるようなトレーニング）の充実などの効用が増し、利便性が高まる。

この要因の促進のためのプラットフォーム製品提供者がとり得るアクションとしては、ユーザーが購買意欲を掻き立てられるような訴求広告、ユーザー同士がベストプラクティス情報をシェアできるような顧客向け感謝イベントなどの場を提供するなどがある。こういったことを通じてアクセス可能ユーザー数の増加を図ることができる。

また、ユーザーのマルチホーミングコストの低減を図ることも重要である。具体的には複数のプラットフォーム製品を保有するコスト負担を軽くすることである。加えて、他のユーザー・グループとの相互接続により、新たなサービスを追加で利用できるような状況を創り出すことも大切である。

（2） 販売チャネルにとってのメリット

販売チャネルは、プラットフォーム製品を搭載してハードウェアをシステムとして構築し顧客に販売する。BtoB としてのプラットフォーム製品のユーザーである販売チャネルにとってのメリットは、開発者やシステム購入顧客が増加すれば、プラットフォーム製品を搭載したハードウェアの売り上げが増加するなどがある。インストールド・ベースの顧客が増えることで、保守メンテナンスやハードウェア増設などのビジネス機会も増える。トレーニングや書籍などの補完製品を提供している場合は、その売り上げも増加するなどのメリットも期待できる。

第３節 ブリッジングの影響力（Influential force in bridging scope）

プラットフォーム製品は、その隣接階層に多くの補完製品としてのプラットフォーム製品を配することで、ドミナント・プラットフォームとなる可能性がある。その際、一方向依存関係[97]（入れ子関係）が構築され、多くの入れ子にされたプラットフォーム製品を保持することで、プラットフォーム製品が隣接階層のプラットフォーム製品の非ドミナント化を誘発し、自らが新たにドミナント化可能なプラットフォーム製品になる。ここで、隣接する上位階層プラットフォーム製品数を N（多数）、下位階層プラットフォーム製品数を１として、多数対１の関係において、下位階層プラットフォーム製品の隣接階層で入れ子状態にするプラットフォーム製品数Nが多いか少ないかが、ドミナント化に大きく影響する。こういった要因を「ブリッジングの影響力」と呼ぶことにしたい。

この要因の設定に関しては、先行研究における Eisenmann, Parker & Alstyne (2007) や加藤（2008）、根来・加藤（2010）が指摘する、プラットフォーム製品がその隣接階層上のプラットフォーム製品を「入れ子」にする状態をどの程度まで広げられるかが要となることに依拠する。これに関わる操作項目としては、隣接対象にするプラットフォーム製品を多数選定することで、ブリッジングの影響力を高めることができる

97 階層は特性として「上位・下位階層間の非対称依存特性」をもつ。依存とは下位階層がないと上位階層が動かない（機能しない）ということであり、逆は不成立の場合が「一方向的依存」である。一方、モジュールは依存関係を特定しない概念である。詳しくは第１章の第４節を参照のこと。

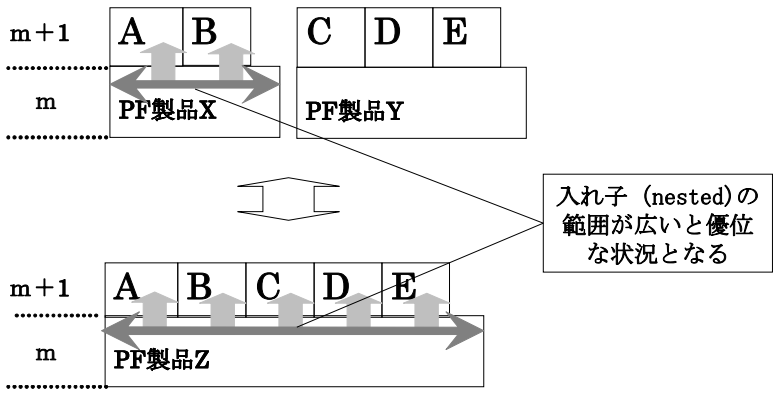
と考えられる。ブリッジングによって、多数の隣接プラットフォーム製品を入れ子関係にすることにより、プラットフォーム製品がドミナント・プラットフォーム製品になる可能性は高まると推論される。

（1） 階層間の入れ子対応関係

ブリッジングの影響力を大きく左右するものとして、隣接する上位階層プラットフォーム製品数をN（多数）、下位階層プラットフォーム製品数を１として、多数対１の関係において、下位階層プラットフォーム製品の隣接階層で入れ子状態にするプラットフォーム製品数Nの多少（多いか少ないか）が関係する。多数の補完的プラットフォーム製品を入れ子の関係にすることによって、優位な状況を作り出すことが可能となる（図３－２）。

図３－２において、m階層にあるプラットフォーム製品Xとm階層にあるプラットフォーム製品Zを比較した場合、補完的なプラットフォーム製品であるA,B,C,D,Eをできるだけ多く入れ子にすることによって、ブリッジングの影響力を高めることが可能となる。具体的な事例としては、プラットフォーム製品統合がこれに該当する。詳しくは付録Aにて詳述。

図３－２：階層間入れ子関係



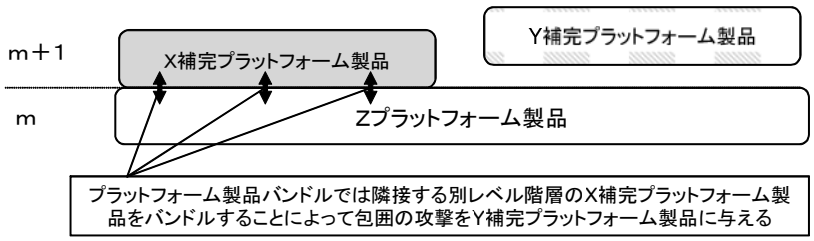
第４節　プラットフォーム製品排除に対する抵抗力　(Detention force against platform exclusiveness)

市場参入したプラットフォーム製品が、プラットフォーム包囲の施策により、レイヤースタック内から排除される可能性が存在する(図３-３)。その際、例えばレイヤースタック内の別のレベルの階層でプラットフォーム製品を保有し、収益を確保することは排除に対する抵抗力を強めることとなる。また、プラットフォーム製品提供者の企業存続の能力も排除に対する抵抗力を左右する。このような要因を「排除に対する抵抗力」とする。

この要因の設定に関しては、Eisenmann, Parker & Alstyne (2007) が提示するプラットフォーム包囲の攻撃に対して、どの程度の抵抗力があるかによって左右されることに依拠する。これに関わる後発プラットフォーム製品提供者の操作項目（後述）としては、持続的収益確保モデルの遂行により、排除に対する抵抗力を高めることができると考えられる。プラットフォーム製品において排除に対する抵抗力が高いと、レイヤースタック内に居座りドミナント・プラットフォーム製品になる可能性が高まると推論される。

図３-３において、ｍ＋１レベル階層にあるＸプラットフォーム製品とＹプラットフォーム製品は両者ともｍレベル階層にあるＺプラットフォーム製品上の補完製品である。Ｚプラットフォーム製品上のＸプラットフォーム製品は隣接階層間のバンドルの戦略を実行したために、Ｙプラットフォーム製品は駆逐されてしまう状況を図にて表わしている。

図３-３：プラットフォーム製品バンドル



（１） プラットフォーム包囲攻撃に対する反撃と防御

プラットフォーム包囲の事例として、以下のようなものがある。

①攻撃側：マイクロソフト社の Windows、ターゲット側：リアルネットワークス社のストリーミング・メディア・プレイヤー、②攻撃側：eBay 社のオークション・マーケットプレイス、ターゲット側：PayPal 社の e-mail ペイメントサービス、③攻撃側：マイクロソフト社のオフィス、ターゲット側：アドビ社のアクロバット・PDF ライター・ソフトウェアの３つが Eisenmann, Parker & Alstyne (2007)では提示されている。ちなみにコンピュータ・ソフトウェア産業での事例は①と③で、両者とも攻撃者はマイクロソフト社である。

これらプラットフォーム包囲の戦略に対し、どのようにして反撃するか、または防御するかについて、以下レイヤースタック外とレイヤースタック内で説明する。

１）レイヤースタック外での包囲に対する反撃（Counteroffensive）

包囲に対してまったく打つ手がないわけではない。単体プラットフォーム・プロバイダーが包囲されないためには、どうすればよいのか３つの対抗策がある。それは①ビジネスモデルを変える、②心強い仲間と手を組む、③法的手段に訴える、の３つである[98]。これらを実践して生き残りをはかっている企業が

98 Eisenmann, Parker & Alstyne (2006), (邦訳) p.101 参照。

存在する。ストリーミング・ソフトの草分け的存在であるリアルネットワークス社である。

2）レイヤースタック内での包囲に対する防御（Defensive action）

レイヤースタック内でのプラットフォーム包囲に対する防御では、入れ子にされる側のプラットフォームのオープン性の堅持が、包囲されない階層として存在できる可能性を高めると考えられる。前述の、入れ子状態にある隣接関係のプラットフォーム間において、入れ子にする側から入れ子にされる側のプラットフォーム包囲の実行は、入れ子にされるスタンドアローン企業側からは打つ手がない状態となる可能性が高い。よって、これを回避するために出来る限り多くの階層との隣接関係をもつことが、重要である。仮にひとつの隣接プラットフォーム製品から包囲をされても、他の隣接プラットフォームを多く抱えることにより、その包囲の影響を限定的にすることができる。

第５節　おわりに

本章では、プラットフォーム製品のドミナント化を誘発する要因として、階層間ネットワーク効果の効用力、ブリッジングの影響力、プラットフォーム製品排除に対する抵抗力の３つを先行研究レビュとの関係を示しながら提示した。

具体的には、階層間ネットワーク効果の効用力の要因の設定に関しては、顧客と開発業者のふたつのユーザー・グループがユーザー内、ならびにユーザー間ネットワーク効果を発揮することや、マルチホーミングコストの低減によるユーザーの購入の促進に依拠している。この分野の先行研究としては、OSS コニュニティなどの開発者コミュニティに関するもの、補完業者を含むプラットフォーム・ビジネスに関するもの、ユーザーのプラットフォーム製品選択のインセンティブに関して、ネットワーク効果やバンドワゴン効果を指摘しているものがある。加えて、ツーサイド・プラットフォーム理論のふたつのユーザー・グループの相乗的増加やマルチホーミングコストに関して論じているものがある。

ブリッジングの影響力の要因の設定に関しては、先行研究におけるプラットフォーム製品が、その隣接階層上のプラットフォーム製品を「入れ子」にする状態をどの程

度まで広げられるかが要となることに依拠する。この分野の先行研究としては、上下階層間の一方向依存性による下位階層の優位性に関するものがある。

プラットフォーム製品排除に対する抵抗力の要因の設定に関しては、プラットフォーム包囲の攻撃に対してどの程度の抵抗力があるかによって左右されることに依拠する。この分野の先行研究としては、プラットフォーム包囲とそれに対する対抗策に関して論じているものがある。

第４章

後発プラットフォーム製品提供者の操作項目

第１節　はじめに

本章では、後発プラットフォーム製品提供者の操作項目について論じる。前３章と同じく先行研究レビュにより、プラットフォーム製品のドミナント化の要因に影響をもたらす後発プラットフォーム製品提供者の操作項目として、アクセス可能ユーザー数の増加、マルチホーミングコストの低減、隣接対象プラットフォーム製品の多数選定、持続的収益確保モデルの遂行の４つを提示する。それぞれ、第２節ではアクセス可能ユーザー数の増加、第３節ではマルチホーミングコストの低減、第４節では隣接対象プラットフォーム製品の多数選定、第５節では持続的収益確保モデルの遂行を論じる。

これらは前３章で提起したプラットフォーム製品のドミナント化要因に対し、後発プラットフォーム製品提供者の操作項目を梃子のようにレバレッジ（Leverage）することによって要因を高める影響を及ぼすものとして提示する。

第２節　アクセス可能ユーザー数の増加（Increasing the number of accessible users）

アクセス可能ユーザー数を増加させるためには、ふたつの方法がある。ひとつは上下階層でプラットフォーム製品を隣接させることにより、後発プラットフォーム製品は既存の複数のプラットフォーム製品の既に保持しているネットワーク効果（アクセス可能ユーザーの総和）を横取りすることが可能になる。この項目に関する先行研究

では、Shapiro & Varian（1999）による相互運用性の効果や、Rohlfs（2001)による補完的なバンドワゴン効果によって論じられている。もうひとつは、後発プラットフォーム製品提供者が開発者や補完業者のコミュニティなどを積極的に支援し、より多くのアクセス可能ユーザーを惹きつけることも重要な施策となる。先行研究では、Gawer & Cusumano（2002)による補完業者の誘引や、DiBona, Ockman & Stone(1999)、O’Reilly(1999)によるプログラマーの開発インセンティブ、佐々木・北山(2000)のコミュニティ、末松(2002)のボランティアの求心力などが論じられている。

こういったアクセス可能ユーザー数の増加は階層間のネットワーク効果を促進する。

第３節　マルチホーミングコスト[99]の低減　(Saving multi-homing costs)

ホーミングコストとは、プラットフォーム製品の導入から運用、さらにはその機会コストに至るまで、ユーザーがプラットフォーム製品を使用し続けるための総コストを指す。利用する「家：Home」の数が増えれば、それだけユーザーの総コストは増える。例えば階層介入型プラットフォーム製品（後述）投入時のホーミングコストのマネジメントとしては、マルチ（複数の）ホーミングコストを低く設定して、ユーザーに抵抗なく受け入れてもらえるようにすることが要となる。特に参入期のクリティカルマスに達しないスタートアップ問題を、これにより回避する可能性を高めることができる。この項目に関する先行研究では、Eisenmann(2010)、根来・加藤(2010)によるWTAメカニズムにおけるマルチホーミングコストの重要性が論じられている。

マルチホーミングコストの低減により、参入する隣接プラットフォーム製品が増加することで、階層間のネットワーク効果が促進される。

---

99 マルチホーミングコストは複数の商品やサービスを同時に利用する際に生じるコストであり、これに対する用語として、単一の商品やサービスを利用するモノホーミングコストがある。また、スイッチングコストとは、ある商品から他の商品、あるいはあるブランドから他のブランドに切り替えることに伴って発生する費用である。

第４節　隣接対象プラットフォーム製品の多数選定 (Wide selecting the targeted neighboring platform products)

参入期の後発プラットフォーム製品提供者にとって、短期間のうちに自社プラットフォーム製品の市場普及を図ることが必要である。普及に時間がかかりすぎると、既存の先発プラットフォーム製品の巻き返しにより駆逐されてしまうリスクが高くなるためである。従って、上下どちらかのオープン性をもつ階層に、できるだけ多くのプラットフォーム製品を選定するか、もしくは既に高いシェアを有する先発プラットフォーム製品をできる限り多く隣接対象として選定することが肝要となる。この項目に関する先行研究では、Eisenmann,Parker & Alstyne(2006,2007)のプラットフォーム包囲ならびに加藤(2009,2013)のブリッジングの効果が論じられている。

多数のプラットフォーム製品を隣接階層に配することで、後発プラットフォーム製品のブリッジングの影響力は高まる。

第５節　持続的収益確保モデルの遂行 (Implementing sustainable profit model)

持続的収益確保モデルの遂行は、プラットフォーム製品提供者の存続において不可欠な項目である。ホーミングコストとの兼ね合いで、スタートアップ期、プラットフォーム製品を無料もしくは廉価に提供する場合、なんらかの方法で持続的に収益を確保する手段が必要となる。具体的には、複数のプラットフォーム製品をレイヤースタック内にもち、一方もしくは両方の階層で収益を確保していくことや、ハードを含めた製品エコシステム全体の中で、持続的に収益を確保できる収益モデルを構築できるかが重要となる。この項目に関する先行研究では、根来・加藤(2006,2010)の収益モデルの重要性について論じられている。

加えて、複数階層でのプラットフォーム製品の提供は、「一方向的依存」状態による包囲のリスクの回避策ともなる。

第６節　おわりに

本章では先行研究レビュにより、後発プラットフォーム製品のドミナント化の要因に影響をもたらす後発プラットフォーム製品提供者の操作項目として、アクセス可能ユーザー数の増加、マルチホーミングコストの低減、隣接対象プラットフォーム製品の多数選定、持続的収益確保モデルの遂行の４つを提示した。これらは次章にてドミナント化要因との関連について論じる。

第５章

推論による仮説の提示

第１節　はじめに

前章までで先行研究レビュにより取り上げた、プラットフォーム製品のドミナント化要因と、後発プラットフォーム製品提供者の操作項目は、それぞれどのような関連があるのか。これまで先行研究レビュによりプラットフォーム製品のドミナント化要因として、要因Ａ：階層間ネットワーク効果の効用力、要因Ｂ：ブリッジングの影響力、要因Ｃ：プラットフォーム製品排除に対する抵抗力、の３つを提示してきた。後発プラットフォーム製品提供者の操作項目として、操作項目①：アクセス可能ユーザー数の増加、操作項目②：マルチホーミングコストの低減、操作項目③：隣接対象プラットフォーム製品の多数選定、操作項目④：持続的収益確保モデルの遂行を抽出した。本章ではその関連を第２節で推論による仮説として提示する。

ここで、プラットフォーム製品のドミナント化要因と後発プラットフォーム製品提供者の操作項目との関係は、プラットフォーム製品提供者が操作項目をレバー（操作てこ）として働かせることで、各ドミナント化要因が高まり、その作用によって後発プラットフォーム製品のドミナント化の可能性を高めるという関係になる。言い換えれば、要因の度合いは、４つの操作項目をコントロールすることで、高めることが可能である。

従って本論文では、後発プラットフォーム製品のドミナント化は４つの操作項目をコントロールすることで、可能性を高めることができる可能性を示唆している。

第２節　後発プラットフォーム製品のドミナント化の仮説的推論

（１）　プラットフォーム製品におけるドミナント化の可能性とドミナント化要因の仮説

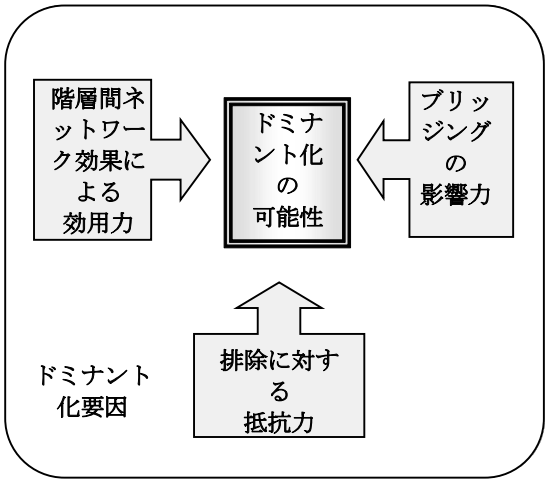
プラットフォーム製品におけるドミナント化の可能性とドミナント化要因との関係において、以下のような仮説を提起する。仮説中の（＋）は促進、（－）は抑制を示す。

仮説 1-1：要因 A・要因 B・要因 C はプラットフォーム製品のドミナント化要因となる

仮説 1-2：仮説 1-1 を前提として、要因 A・要因 B・要因 C の３つの要因が、それぞれ高く（＋）なる場合にドミナント化の可能性が高まる（＋）

上記、仮説 1-1、仮説 1-2 を図で表現すると以下のようになる（図５－１）。矢印はドミナント要因からドミナントの可能性（中央）に向けての「作用」を示す。

図５－１：プラットフォーム製品におけるドミナント化と要因の関係



（2）後発プラットフォーム製品におけるドミナント化要因と操作項目の仮説

後発プラットフォーム製品におけるドミナント化要因と操作項目の関係において、以下のような仮説を提起する。仮説中の（＋）は促進、（－）は抑制を示す。

仮説 2-1:後発プラットフォーム製品のドミナント化において、項目①の促進（＋）が要因 A を高める（＋）

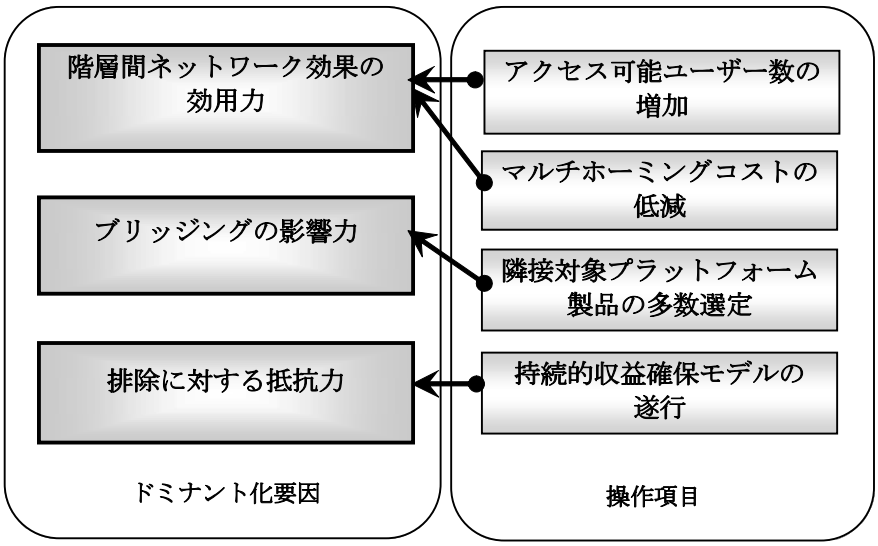
仮説 2-2:後発プラットフォーム製品のドミナント化において、項目②の促進（＋）が要因 A を高める（＋）

仮説 2-3:後発プラットフォーム製品のドミナント化において、項目③の促進（＋）が要因 B を高める（＋）

仮説 2-4:後発プラットフォーム製品のドミナント化において、項目④の促進（＋）が要因 C を高める（＋）

上記、仮説 2-1、仮説 2-2、仮説 2-3、仮説 2-4 を図で表現すると以下のようになる（図５－２）。実線矢印は操作項目（右）からドミナント化要因（左）への「主たる影響」を示す。

図５－２：後発プラットフォーム製品におけるドミナント化要因と操作項目の関係



1） 階層間ネットワーク効果の効用力と操作項目

アクセス可能ユーザー数を増加させるためには、ふたつの方法がある。ひとつは、プラットフォーム製品提供者が自らの魅力を高め、多くのアクセス可能ユーザーを惹きつけることである。もうひとつは、隣接階層のアクセスユーザーを奪い取ることによって、増加させることができる。よって、より多くのアクセス可能ユーザーを既に保有している階層と隣接することが重要である。

マルチホーミングコストを低く設定することは、ユーザーにとって、追加コストを低く抑えることができるという理由で、より導入しやすい環境となる。これにより多くのユーザーがプラットフォーム製品を選択する可能性が高まる。

アクセス可能ユーザー数の増加、ならびにマルチホーミングコストの低減は階層間のネットワーク効果の効用力を高めるという影響を及ぼす。

2） ブリッジングの影響力と操作項目

隣接対象のプラットフォーム製品の選定においては、オープンのインターフェイスをできるだけ多く、且つ広範囲にもつことで、多くのプラットフォーム製品の隣接となる。現実にはドミナント・プラットフォーム製品から、オープンのインターフェイスを阻害するような行動をとられることもある。そのため、常にオープン性の堅持が重要である。隣接対象のプラットフォーム製品を多数選定すること、これによりブリッジングの影響力を高めるという影響を及ぼす。

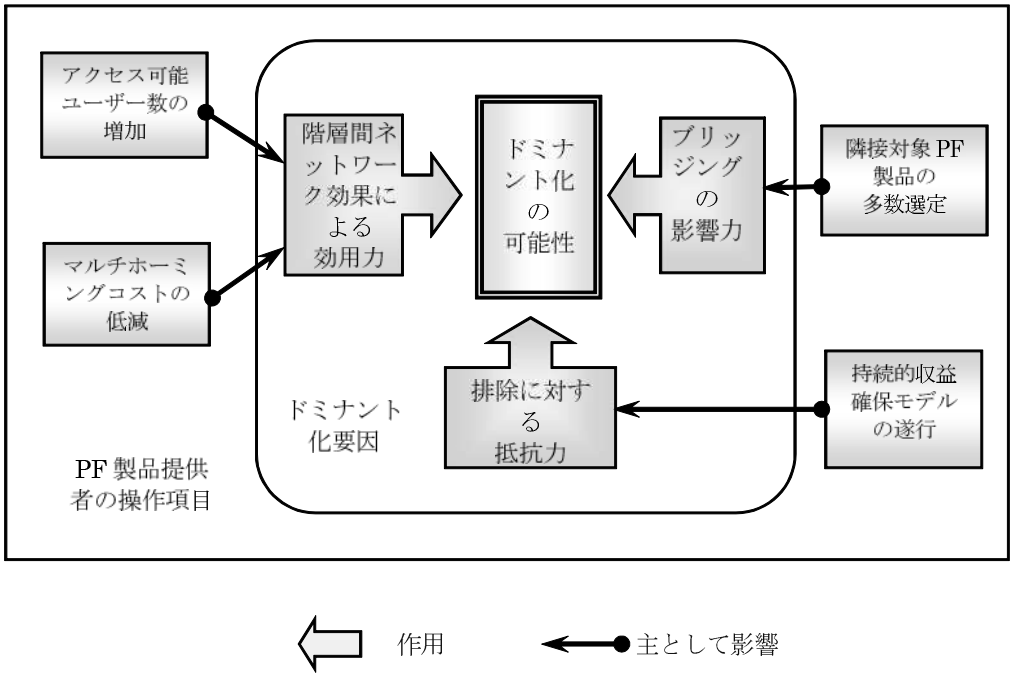
3） プラットフォーム排除に対する抵抗力と操作項目

持続的な収益モデルの遂行は、企業の存続能力を高める。これによりレイヤースタック内に居座りドミナント・プラットフォーム製品になる可能性を有することができる。なぜなら、後発プラットフォーム製品のなかには、既存の下位階層のプラットフォーム製品に収益源を断たれ、レイヤースタック内から排除されてしまう状況が想定されるためである。持続的な収益モデルの遂行は、包囲に対する抵抗力を高めるという影響を及ぼす。

（3） 小括

これまで、後発プラットフォーム製品におけるドミナント化の仮説的推論をおこなってきた。本章のドミナント化の可能性とドミナント化要因ならびに後発プラットフォーム製品提供者の操作項目を図にすると以下のようになる（図５−３）。実線矢印は主となる影響を示し、太い矢印は作用を示す。実線矢印を主となる影響と表記したのは、かならずしも矢印の方向のみに影響を与えるのではなく、他の要因へも少なからず影響することもあるということを意味している。

図５−３：後発プラットフォーム製品におけるドミナント化のモデル



第３節 おわりに

本章では、先行研究レビュにより取り上げた、プラットフォーム製品のドミナント化要因と、後発プラットフォーム製品提供者の操作項目の関連を論じた。これまで先行研究レビュによりプラットフォーム製品のドミナント化要因として、要因Ａ：階層

間ネットワーク効果の効用力、要因Ｂ：ブリッジングの影響力、要因Ｃ：プラットフォーム製品排除に対する抵抗力、の３つを提起してきた。また、後発プラットフォーム製品提供者の操作項目として、操作項目①：アクセス可能ユーザー数の増加、操作項目②：マルチホーミングコストの低減、操作項目③：隣接対象プラットフォーム製品の多数選定、操作項目④：持続的収益確保モデルの遂行を抽出した。本章では仮説的推論より、その関連を後発プラットフォームのドミナント化のモデルとして提示した。

第６章

階層介入の戦略と位置付け

第１節　はじめに

本章では階層介入戦略と、その位置付けについて論じる。第２章で「階層介入」(Layer Intervention)の戦略を定義し、プラットフォーム戦略のひとつとして、どのように位置付けられるかに関して考察する。まず、階層介入はプラットフォーム戦略上の後発プラットフォームの一種であると考えられる。後発のプラットフォーム製品とは、レイヤースタック内において、エコシステムの中心に位置する、いわゆるコアとなるプラットフォーム製品が先発として存在し、その後そのコアのプラットフォーム製品向けの補完的なプラットフォーム製品が後発で誕生する際、後者を指す。加えて、階層介入型プラットフォーム製品は、その特徴として隣接階層に対してオープンなインターフェイスを確保している。また、先行研究レビュでも触れたように、「同一階層内でのプラットフォーム製品統合」と「隣接階層プラットフォーム製品バンドル」とは異なる、全く新たな階層が介入して、具体的には階層数が変化（増加）して、それによって階層間関係やポジションに影響をもたらすというものである。

サーバー市場を俯瞰すると、市場の誕生から１社もしくは数社のプラットフォーム製品が市場シェアの大部分を占め、市場の成長と共にドミナントの地位を築く傾向が強い。その際、隣接する補完製品の階層は、数多くの提供者が乱立し存在するようになり、コモディティ化が進行し、隣接する補完製品の階層へのプラットフォーム製品提供者のマージンは圧迫される。

また第３節では、階層介入の機能と効果について分析する。階層介入は、その機能として、「介入（Intervention）」と、「橋渡し（Bridging）」がある。この機能が

もたらす効果として、①アクセス可能ユーザーの流動性の高まり、②同一レベル階層のコモディティ化の促進、③上下階層間の相互インターフェイスの制御がある。

第２節　階層介入戦略の位置付け

本節では、階層介入型プラットフォーム製品を定義した上で、階層戦略の位置付けを競合関係ならびに補完関係の視点で比較する場合と、プラットフォーム製品統合ならびにプラットフォーム製品バンドル戦略と比較する場合で、階層介入戦略の位置付けに関して論じる。

（１）階層介入型プラットフォーム製品の定義

本論文では、階層介入型プラットフォーム製品を以下のように定義する。階層介入型プラットフォーム製品は、以下の３つの特徴を同時にもつ製品である。

① （OS などの先発プラットフォーム製品に対し）後発プラットフォーム製品である。
② 階層を形成する最初のプラットフォーム製品である。（レイヤースタック内の総階層数は増える）
③オープンなインターフェイスを持ち、上下いずれかの隣接階層に複数のプラットフォーム製品を保持し得る。

（２）競合関係と補完関係での比較

本項では競合関係にあるプラットフォーム製品間の戦略を同一レベル階層戦略と呼び、補完関係にあるプラットフォーム製品間の戦略を複数レベル階層戦略と呼ぶ。それぞれの戦略を比較することにより、本論文での階層介入戦略の位置付けを明示する。

ふたつの戦略の大きく異なる点は、プラットフォーム製品間の関係が、競合関係であるか、補完関係であるかということである。それにより、競争の場が対立するふた

つのレイヤースタックであるか、ひとつのレイヤースタック内でおこなわれるかという分類となる。以下、表にする（表６－１）。

表６－１：階層関係視点での分類

階層関係の視点	同一レベル階層戦略	複数レベル階層戦略
PF 間の関係	競合関係	補完関係
主戦場	対立するふたつのレイヤースタック	ひとつのレイヤースタック内
具体例	同一階層レベルの既存 OS と新 OS、 既存アプリと新アプリ	下位階層の OS と隣接する上位階層のアプリ
事例	Windows vs Solaris など	Windows and Netscape, SMP, Java, VMware など

（３）補完関係における先発・後発での比較

次に、補完関係にあるプラットフォーム製品間で、市場参入の時期にて分類をおこなう。OS などの先発プラットフォームに対して、アプリケーションは後発となる。そのなかで、階層介入型プラットフォーム製品は、隣接階層に対しオープンなインターフェイスをもっていることによって区分される。以下、表にする（表６－２）。

表６－２:補完関係のプラットフォーム製品の分類

参入時期	先発	後発	後発
対隣接階層	オープン	クローズド	オープン
PF 間の関係	補完関係	補完関係	補完関係
主戦場	ひとつのレイヤースタック内	ひとつのレイヤースタック内	ひとつのレイヤースタック内
具体例	OS	アプリ	階層介入型 PF 製品

事例	Windows など	Real Player の SMP など	Java, VMware など

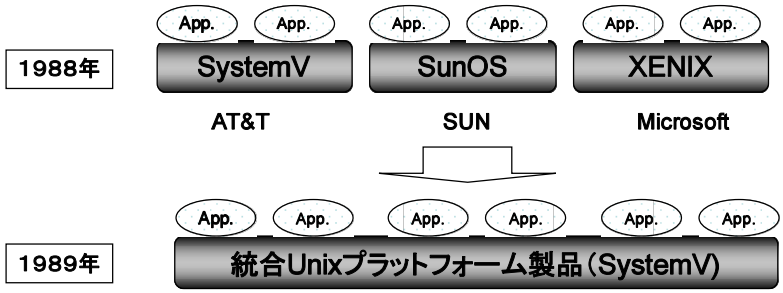
（4）プラットフォーム製品統合とプラットフォーム製品バンドルと階層介戦略の比較

プラットフォーム製品統合ならびにプラットフォーム製品バンドル戦略との比較をおこなう。

1）プラットフォーム製品統合の概要

コンピュータ産業では 1980 年代後半から AT&T 社が中心となった UI（Unix インターナショナル）と IBM 社が中心となった OSF（オープン・ソフトウェア・ファンデーション）の標準化の２大陣営が生まれることとなる。1980 年代後半には各社の独自 Unix が普及していたため、アプリケーションはそれぞれの Unix 用に作り込む必要があった。これら亜種の Unix を統合するため実行された施策は同一レベル階層内でプラットフォーム製品を統合する戦略と理解できる。UI は OSF に対抗して、自らが自社 OS を中心に据えて統合化を推し進めた。言い換えれば、クリティカルマスに達しないその他の独自商用 Unix を一掃する動きに出た。UI 陣営は 1989 年 11 月、「Unix システムＶリリース 4.0」を完成させ発表する。これは「システムＶ」「バークレー4.3」「XENIX」を統合したものである（図６－１）[100]。詳しくは付録 A を参照。

図６－１：同一レベル階層でのプラットフォーム製品統合



100 ホール・バリー（1991）, p. 160 参照。

２）プラットフォーム製品バンドルの概要

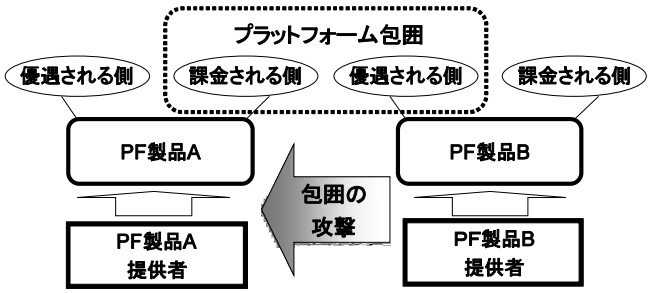
1999 年から 2000 年にかけて、ストリーミング・ソフトのプラットフォーム製品提供者として草分け的存在であったリアルネットワークス社は、消費者に SMP（Streaming Media Player）を無償配布し、コンテンツ企業にサーバーを販売していた[101]。リアルネットワークス社は、これによりストリーミング・メディア市場を短期間でほぼ独占し、ドミナント・プラットフォーム製品としてかなりの利益を上げていた。しかしそれ以前の 1998 年にはマイクロソフト社からの攻撃によって、レイヤースタック内から駆逐されそうな時期があった[102]。当時のマイクロソフト社は市場があると判断すると、水平的に補完業者の市場に侵入することを公然と明言していた。リアルネットワークス社の SMP は Windows の補完製品としてパートナー関係をもちながらも、マイクロソフト社によって駆逐される対象となった。隣接階層の Windows はリアルネットワークス社の SMP を入れ子（nested）の状態にしていた。施策は自分自身の機能にターゲットのプラットフォーム製品の機能をバンドルの状態で結合させる（隣接階層とのバンドリング）プラットフォーム包囲である（図６－２）。詳しくは付録 B を参照。

プラットフォーム包囲を、ツーサイド・プラットフォームのそれぞれのプラットフォーム製品に繋がる課金されるユーザー・グループと優遇されるユーザー・グループで考える場合、図６－２において、プラットフォーム製品 A の課金される側のユーザー・グループはプラットフォーム製品 B の優遇される側のユーザー・グループによる包囲によりダメージを受け、プラットフォーム製品 A はレイヤースタックから排除されるリスクが発生する。

---

101 Eisenmann, Parker & Alstyne（2006），（邦訳）p. 100 参照。
102 同上。

図６－２：ユーザー・グループの観点からのプラットフォーム包囲



プラットフォーム製品統合は単一レベルの階層戦略で、レイヤースタック内の階層数は変わらない。また、プラットフォーム製品バンドルは複数レベルの階層戦略で、レイヤースタック内の階層数は変わらない。これに対し、階層介入戦略は単一レベルの階層戦略もしくは複数レベルの階層戦略で、階層の数が変化するものである。これら、ふたつの階層戦略と階層介入戦略を分類すると以下のような表にまとめることができる（表６－３）。

表６－３：統合・バンドル・介入の分類

	階層数不変	階層数変化
単一レベル階層戦略	プラットフォーム製品統合	階層介入
複数レベル階層戦略	プラットフォーム製品バンドル	

（５）小括

同一階層内での「プラットフォーム製品統合」は、具体的には上下の階層を意識せず、同じレベルの階層間での統合を図るものである。また、複数レベルの階層戦略である「プラットフォーム製品バンドル」は、飛躍的なイノベーションやシュンペーターの唱える創造的破壊を前提としないプラットフォーム・リーダーシップの交代のメカニズムを、バンドルを利用した戦略で捉えたものである。階層介入戦略とプラットフォーム製品バンドル（プラットフォーム包囲）の類似点は、同一レベルの階層間の

プラットフォーム製品競争において、隣接する別の階層を利用し攻撃を仕掛ける点にあるが、プラットフォーム包囲は同一レベル階層間プラットフォーム製品競争において、隣接する既存の別レベルの階層とバンドルする（階層自体を増加させるものではないが、ふたつの階層を束ねる）ことによって包囲の攻撃を与えるものである。これらに対し、階層介入戦略は、隣接する別レベルで新たな階層を階層外から介入させる（階層数を増加させる）ことにより、階層間のプラットフォーム製品競争へ影響を与えることが可能な戦略である。

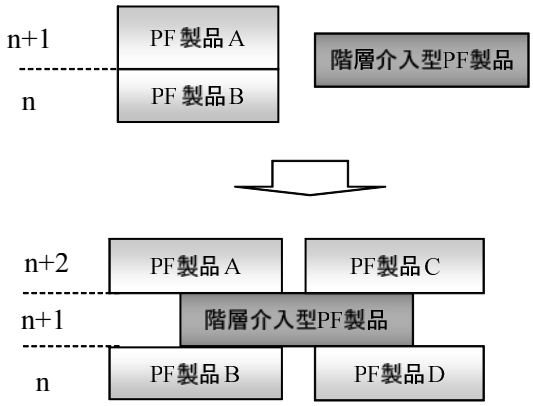
まとめとして、本節ではプラットフォーム製品統合やプラットフォーム製品バンドルとの比較を通じて、階層介入戦略の位置付けをおこなった。具体的にはプラットフォーム製品統合は単一レベルの階層戦略で、プラットフォーム製品バンドルは複数レベルの階層戦略であるが、レイヤースタック内の階層数は変わらない。これに対し、階層介入戦略は単一レベルの階層戦略もしくは複数レベルの階層戦略で、階層の数が変化するものであることを指摘した。

第３節　階層介入とその効果

本節では、階層介入による「介入（Intervention）」と「橋渡し（Bridging）」の機能が、どのような効果をもつか論じる。

階層介入の戦略は、上下に隣接するふたつのプラットフォーム製品の間に新たなプラットフォーム製品として後から介入する（図６－３）。後から介入するためには、上位層もしくは下位層にオープンなインターフェイスを確保し続けることが必要である。仮に上位層にも下位層にもオープンなインターフェイスがない場合、後からの介入は困難となる。また介入によって論理上の総階層数は、レイヤースタック内で増加することが特徴である。

図６－３



＜図６－３の説明＞

図６－３は、ｎレベル階層にあるプラットフォーム製品Ｂとｎ＋１レベル階層にあるプラットフォーム製品Ａの間に介入階層が介入する前と後を図として表現したものである。ｎ＋１レベル階層への介入階層の介入により、それまでｎ＋１レベル階層にあったプラットフォーム製品Ａはｎ＋２レベル階層へ変更となる。同時に、ｎレベル階層にあるプラットフォーム製品Ｂは同一レベル階層にあったプラットフォーム製品Ｄと介入階層により橋渡しされた。また、ｎ＋２レベル階層のプラットフォーム製品Ａは同一レベル階層にあるプラットフォーム製品Ｃと橋渡しされた。

ソフトウェア・レイヤースタックの階層間に「介入（Intervention）」し、「橋渡し（Bridging）」をおこなう機能は、既存の階層間関係や階層内でのポジションを変化させてしまう可能性をもつ。介入による影響は、各階層が保有するアクセス可能ユーザーの流動性を高め、同一階層レベルでのプラットフォーム製品の選択必然性を弱める。アクセス可能ユーザーの流動性の高まりは、相互接続で増加するアクセス可能ユーザーが必ずしも自社プラットフォーム製品の使用に結び付かない可能性につながる。よって、自社以外の隣接プラットフォーム製品にユーザーの多くを横取りされ、結果として他プラットフォーム製品が選択されてしまうことが起こり得る。また、このプラットフォーム製品の選択必然性の弱まりは、同一レベル階層でのプラットフォーム製品のコモディティ化を誘発する。加えて、上下階層をセットにした垂直統合の収益モデルを変化させる可能性が生じる。

以下、それぞれの効果に関して詳しく説明する。

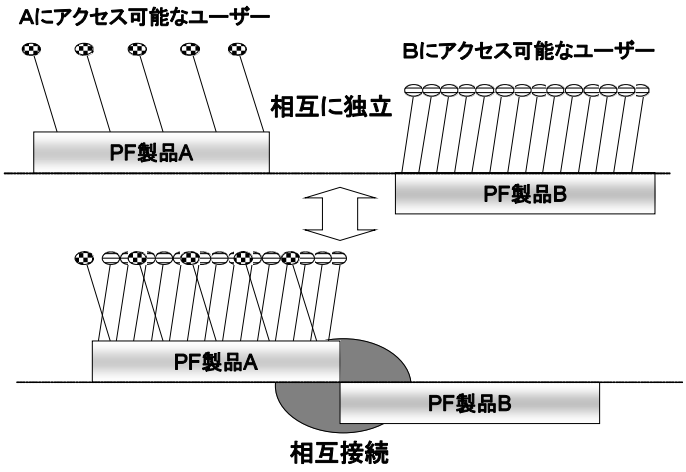
（1） アクセス可能ユーザーの流動性の高まり

ひとつめの効果は「橋渡し」機能が起因し、各階層が保有するアクセス可能ユーザーの流動性が高まることである。

一般に、ネットワーク効果は、「ある製品から得られる便益が、当該製品のユーザーが増えるに従って増大する性質」と把えられるが、その性質はエンドユーザー（B to C ユーザー）に適用される。しかし、補完製品（補完階層）を考慮した間接的なネットワーク効果を考える場合では、より魅力的なプラットフォーム製品に惹かれユーザーのアクセスが、特定のプラットフォーム製品に偏ってしまうことにより、相互接続で増加するアクセス可能ユーザーが必ずしも自社プラットフォーム製品の利益に結び付かない現象が起こり得る。

言い換えれば、介入階層の相互接続が必ずしも自社プラットフォーム選択に利益をもたらさず、自社以外の隣接プラットフォーム製品にユーザーの多くを横取りされてしまう。結果として他プラットフォーム製品が選択されてしまうことが起こり得る（図６－４）。

図６－４：アクセス可能ユーザーの流動性



＜図６－４の説明＞

図６－４はプラットフォーム製品Aのアクセス可能ユーザーとプラットフォーム製品Bのアクセス可能ユーザーが、相互接続する前と後でどのような変化をするのかを図として表現したものである。プラットフォーム製品Aのアクセス可能ユーザーとプラットフォーム製品Bのアクセス可能ユーザーは、相互接続する前はそれぞれのプラットフォーム製品にのみアクセス可能であった。相互接続後は、プラットフォーム製品Aのアクセス可能ユーザーはプラットフォーム製品Bに、プラットフォーム製品Bのアクセス可能ユーザーはプラットフォーム製品Aにもアクセス可能となる。これによって、極端な例としては図６－４で示されるように、両プラットフォーム製品すべてのアクセス可能ユーザーがプラットフォーム製品Aに集中してしまうことも起こり得る。

（２） 同一レベル階層のコモディティ化の促進

ふたつめの効果も、主に「橋渡し」機能が起因し、同一レベル階層でのユーザーのプラットフォーム製品の選択必然性を弱める。ひいては、同一レベル階層に存在する全てのプラットフォーム製品がコモディティ化を誘発される(図６－５)。

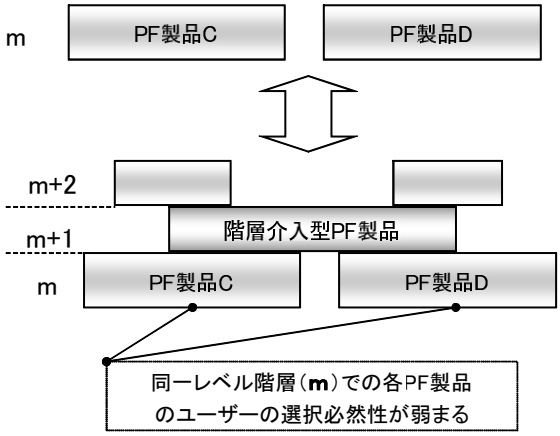
また、隣接階層で排他性が強いプラットフォーム製品は、その排他力を弱められてしまう。

これにより、介入階層の橋渡し機能によって、WTA（勝者総どり）[103]の傾向が弱められ、複数の隣接プラットフォーム製品が共存（延命）する現象が起こり易くなる。

---

[103] Winner-Take-All の略。詳しくは、根来・加藤（2010）を参照されたい。

図６－５：同一階層レベルでのコモディティ化



＜図６－５の説明＞

図６－５は、ｍレベル階層にあるプラットフォーム製品Ｃとプラットフォーム製品Ｄの上位階層に介入階層が介入する前と後を図として表現したものである。ｍ＋１レベル階層への介入階層の介入により、ｍレベル階層にあるプラットフォーム製品Ｃとプラットフォーム製品Ｄはコモディティ化する状況となる。同様にｍ＋２レベル階層にあるプラットフォーム製品もコモディティ化の状況となる。

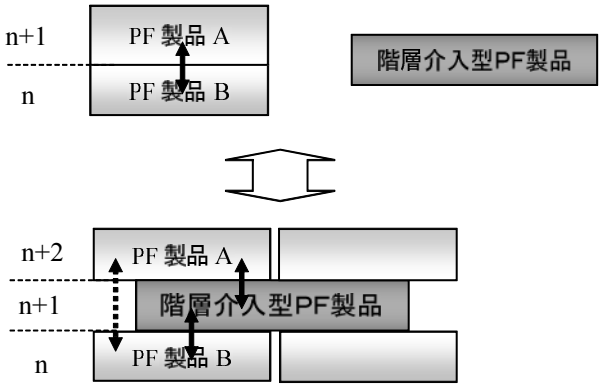
（３）　上下階層間の相互インターフェイスの制御

３つめの効果は、主として「介入」機能が起因し、既存の上下階層の直接的な情報のやりとりを制御することである。

通常、隣接する上下階層間ではふたつの階層間で直接の情報のやりとりがおこなわれている。しかし、階層が介在することで、それまでの上位階層と下位階層の直接のやりとりは出来なくなる。つまり上位階層は介入階層と、下位階層は介入階層とのやりとりとなる。言い換えれば、常に介入階層を通じなければ情報のやりとりは困難となる。これにより、介入階層が上下の階層とのインターフェイスを制御することが可能となる（図６－６）。例えば、隣接するプラットフォーム製品の一部に多くの情報を与えたり、与えなかったりというような制御があり得る。

加えて、仮に既存の上下階層の垂直統合状態が存在する場合、それらを分断し、競合にダメージを与えるようなことが可能となる。

図６－６：上下階層のインターフェイスの制御



＜図６－６の説明＞

図６－６は、ｎレベル階層にあるプラットフォーム製品Ｂとｎ＋１レベル階層にあるプラットフォーム製品Ａの間に、介入階層が介入する前と後のそれぞれのインターフェイスの変化を図として表現したものである。ｎ＋１レベル階層への介入階層の介入により、それまでｎレベル階層にあったプラットフォーム製品Ｂとｎ＋１レベル階層にあったプラットフォーム製品Ａの間で直接のやりとりが可能であった。しかしｎ＋１レベル階層への階層の介入により、プラットフォーム製品Ａはプラットフォーム製品Ｂとの直接のやりとり（破線両矢印）は不可となり、介入階層を介してのやりとりとなる。

（４）　小括

これまで論じてきた「介入（Intervention）」と「橋渡し（Bridging）」による効果、ならびに想定される現象をまとめると以下のような表となる（表６－４）。

表６－４：階層介入の機能・効果・想定される現象

機能	効果	想定される現象
介入（Intervention） 橋渡し(Bridging)	アクセス可能ユーザーの流動性の高まり。	PF 製品選択限定性の回避。 アクセス可能ユーザーの横取りによる他 PF 製品の選択。
	同一レベル階層のコモディティ化の促進。	PF 製品提供者のマージンの減少。 既ドミナント PF 製品の排他性の抑制。 隣接 PF 製品の延命。
	上下階層間の相互インターフェイスの制御。	隣接する特定 PF 製品の優遇もしくは冷遇。 既垂直統合状態の PF 製品間の分断。

第４節　おわりに

これまで論じてきたように、階層介入は、オープン性の保持とレイヤースタック内の階層数の増加という点で、後発プラットフォーム製品のなかでも特殊な性質をもつものであり、後発プラットフォーム製品の一種と位置付けされる。具体的には以下の３つの特徴を同時にもつ製品である。①(OS などの先発プラットフォーム製品に対し)後発プラットフォーム製品、②階層を形成する最初のプラットフォーム製品（レイヤースタック内の総階層数は増える）、③オープンなインターフェイスを持ち、上下いずれかの隣接階層に複数のプラットフォーム製品を保持し得る。

比較の対象として、プラットフォーム製品統合とプラットフォーム製品バンドルのふたつの階層戦略との比較を示した。

また、階層介入にはその機能として、「介入(Intervention)」と「橋渡し(Bridging)」がある。そして、この機能がもたらす効果として、①アクセス可能ユーザーの流動性の高まり、②同一レベル階層のコモディティ化の促進、③上下階層間の相互インターフェイスの制御が考えられる。本章では、これら３つの効果がもたらす想定される現象についても考察した。

第 7 章

階層介入の事例研究

第 1 節　はじめに

本章では、階層介入の戦略を具体的に理解し、先行研究レビュで導出した仮説的推論を確認する。加えて、プラットフォーム製品提供者の操作項目における共通点・相違点を明らかにした上で、戦略上の示唆を得ることを企図し、ふたつの事例研究をおこなった。

ひとつめのプラットフォーム製品による階層介入の事例として、コンピュータ言語の Java を取り上げる。1995 年からサン社は Java を市場に投入する[104]。Java は下位層（ここでは OS 階層）に対してオープン（WORA：Write Once Run Anywhere「一度書けばどこでも動く」）を標榜し、積極的に共同開発やライセンス契約という形態で仲間づくりを推進する。それにより、オラクル社やモトローラ社や IBM 社などとの間で Java 陣営を形成することに成功した[105]。Java はサン社の商用 OS の Solaris（Unix）やオープンソースの Linux などの OS と、マイクロソフト社の Windows（サーバー/クライアント）の間を、階層で橋渡しをおこなった。サン社は Java 自体からの収益はほとんど期待せず、その宣伝効果によるブランド力の向上や、賛同してくれる企業数のアピールによるマーケティング効果などによって、商用 OS やハードウェアなど補完製品の売上から収益を確保した。

---

104 サウスウック（2000）, p. 164 参照。
105 サウスウック（2000）, p. 201 参照。

第２節では、Java の事例の考察として、最初に多面的に事実内容の説明をおこない、事例として選択した理由ならびに前述したプラットフォーム製品提供者の４つの操作項目の観点で分析をおこなった。加えて、階層介入前と後に関して、階層間関係とポジションの変化について考察をおこなった。

もうひとつのプラットフォーム製品による階層介入の事例として、ヴイエムウェア社によって 2001 年から提供されている VMware （具体的には VMware ESX・ESXi[106] 、ベアメタル型[107]のハイパーバイザー[108]でサーバーの仮想化[109]階層を形成するソフトウェア）を取り上げる。VMware によって創られる仮想化階層上では、Windows や Linux といった異なる種類の OS を走らせることができる。サーバー仮想化の歴史は古く、IBM 社のメインフレーム用にリリースした IBMSystem/360[110]向けの OS[111] にまで遡る。スタンフォード大でヴイエムウェア社の創業者の一人でもあるメンデル・ローゼンブラム（Mendal Rosenblum）准教授のグループが、メインフレームでおこなっていた技術を x86CPU[112] のシステムに応用した。この技術が確立し製品化の目処が立って、ヴ

106 ヴイエムウェア社による、コンピュータを仮想化するためのソフトウェア。VMware ESX は仮想化ソフトウェア製品パッケージの一部として有償で販売。サービス・コンソールなどの機能を制限した VMware ESXi は無償で提供されている。仮想化の機能自体は両者で同じ。VMware ESX はハードウェア上で直接動作し、仮想的に構築されたコンピュータ上で様々な種類の OS を動作させることができる。

107 OS などのソフトウェアが何も搭載されていない状態のハードウェアのことを指し、裸のままの金属という意味。

108 ハードウェアの BIOS 上に直接、仮想ソフトウェアを動かし、その上で仮想マシンを動作させる技術。ホスト OS 層を介さずに仮想マシンが稼働するため、ホスト型よりもパフォーマンスが高く、サーバー仮想化の主流となってきている。

109 サーバーの仮想化の機能として、パーティショニング 、隔離 、カプセル化 などがある。これらの機能により、１台の物理サーバーマシン上で複数の OS を稼働させたり、障害（クラッシュ、ウィルス感染、パフォーマンス低下など）を仮想マシンに隔離し、他の仮想マシンへの影響を防いだりすることが可能である。また、ファイルの移動とコピー同様、仮想マシンを容易に移動およびコピーができるなどのメリットを享受できる。サーバー仮想化には仮想マシン方式があり、仮想マシン方式はホスト OS 型とハイパーバイザー型に大別される。詳しくは後述。

110 IBM 社が 1964 年 4 月に発表したメインフレームのシリーズ名称。多用途をカバーするファミリーを形成し、商用から科学技術計算まで幅広く使われた。商用では初めて OS や仮想機械が登場した。

111 世界初のメインフレーム用仮想化 OS（＝ハイパーバイザー）である CP-67 で、現在は z/VM（ゼットブイエム）に続いている。

112 インテル社のパソコン用 CPU の総称。加えて、アスロンなどのインテル互換 CPU も含まれる。8086、80286、i486 といったように下二桁は「86」として数字を製品名にしていたためこう呼ばれる。

イエムウェア社は1998年シリコンバレーでスタンフォード大とUCバークレーの5人の研究者によって設立された[113]。

第３節では、VMware の事例の考察として、多面的に事実内容の説明をおこない、事例として選択した理由、プラットフォーム製品提供者の４つの操作項目の観点で分析をおこなった。その上で、階層介入前と後に関して、階層間関係とポジションの変化について考察をおこなった。

第４節では、操作項目の観点で両事例をまとめ、共通点、相違点について整理をおこなった。

第２節　Java 事例の考察

Java は、クロスプラットフォーム機能を有するプラットフォーム製品である。クロスプラットフォーム製品とは、通常、複数のプラットフォーム製品上で使用できる製品やサービス[114]を指すが、Java の場合は Java 言語によって開発されたアプリケーション（Java アプリケーションや Java アプレット[115]）がそれにあたる。

Java を世に出すという方向性をサンが決定したのは、1994 年の春にタホ湖の近くでおこなわれた会議[116]であったと言われている。その会議の参加者は当時のサン社の幹部 6 人であったが、それまでグリーン・プロジェクトとして開発してきた OAK[117] を Java として再デビューさせることとした。この時点で開発のために既に 1 億 7500 万ドル[118]を投資していた。サン社内には全く利益を生み出していないプロジェクトに対する風当たりは強かったが、そこにいたメンバーはインターネット・ブームが現実になることを読み取って勝負に出ることにした。

Java は、全ての OS に対して Java Virtual Machine（Java 仮想マシン）という環境を備えることで、どの OS の仮想マシンを使っていても、その上で同一の Java のアプ

---

113 出所：http://biz.bcnranking.jp/article/serial/siliconvalley/0910/091013_120511.html 2013/09/23

114 ソフトウェア以外のものとしては、前者については複数電子マネーに対応した読み取り機、後者については複数キャリアに対応した携帯電話サイトなどがある。

115 ネットワークを通じてダウンロードされ実行される Java で書かれたプログラム。ただし、VM による起動速度が遅いなどの理由で、開発の主流は徐々にサーブレットに移行していく。

116 岩山（2001）, p. 58 参照。

117 プログラミング言語 Java の前身（後述）。

118 岩山（2001）, p. 58 参照。

リケーションが動くようにする仕組みをもっている。この特徴を標榜し、Java 陣営はこれまで多くの企業と共同で言語開発をすすめ、システムの実装では競合関係にある企業同士であっても協力し発展させてきた歴史がある。

（1） Java とは

Java は、サン社が開発した言語である。オブジェクト指向[119]を備え、ネットワーク環境で利用されることを強く意識した仕様になっている。Java で開発されたソフトウェアは特定の OS やチップに依存することなく、基本的にはどのような OS 上でも動作する。Java で記述されたソースコードは、コンパイル時に Java バイトコードと呼ばれる中間コードにいったん変換される。ソフトウェアはバイトコードの状態で配布され、実行時には Java 仮想マシン（JVM：Java Virtual Machine）と呼ばれるソフトウェアによって、実行するプラットフォーム製品に対応した形式（ネイティブコード）に変換され、実行される。プラットフォーム製品間の違いは Java 仮想マシンが吸収するため、開発時にはプラットフォーム製品の違いを意識しなくて良い。

利用するユーザーにとって Java により可能となるものは、アプリケーション・オン・デマンド[120]とも呼ぶべき形態である。必要なときにだけアプリケーションをネットワークを介してダウンロードし、自動的に実行するという仕組みは、ユーザーにメリットがあるだけでなくソフトウェア提供にとって低コストで製品やサービスを提供できる有力な手段となる。通常はユーザーにネットワークを介して製品やサービスを提供するには、システムの違いが障害となり容易に提供できないことが想定される。しかし、この Java を利用することにより、そういった障害は減少する。

Java を開発したサン社は、Java に対してオープンな互換性を堅持していくというコミットメントを宣言した。その結果、発表当時から一貫して、提携先であり、また補完業者であるソフトウェア企業各社に対して約束したことを守り続けることになった。もっとも重要な約束は「Write Once, Run Anywhere.（一度書けば、どこでも

---

119 ソフトウェアの設計や開発において、操作手順よりも操作対象に重点を置く考え方。関連するデータの集合と、それに対する手続き（メソッド）を「オブジェクト」と呼ばれるひとつのまとまりとして管理し、その組み合わせによってソフトウェアを構築する。

120 末松・ベネット（1996）, p. 8 参照。

動く）」という特性である。言いかえれば、「Java は OS の種類に縛られないものであり、ネットワークを介してどこでも動く」という約束である。

この約束履行の一環としてサン社は 1996 年「100% Pure Java 認定プログラム」[121]を発表し、翌年より認定制度をスタートさせた[122]。一方、補完業者であるソフトウェア会社は、Java 関連製品を推進していくにあたって、人員や開発ツールなどを先行投資するというリスクを負っている。その背景には Java に対する大きな期待があった。それは将来サン社が描くようなネットワーク・コンピューティングの世界が実現し Java への投資が自社の成長につながる、投資に対するリターンが将来返ってくるという期待である。この認定プログラムは長期的に Java のオープン性を守り、パートナーであり補完業者であるソフトウェア会社が、将来自分たちが活躍するフィールドを狭められることのないようにする上で重要な役割をもっていた[123]。

Java 言語の利用による主なメリットとして、以下があげられる[124]。

①Java 仮想マシンがインストールされていれば Java アプレットはどの OS でも実行することができる。クロスプラットフォームの実現。

②コンポーネント・レベルのオブジェクト・プログラミングに対応した言語である。コンポーネントの再利用と組み合わせによる開発の生産性向上が見込まれる。

③ネットワーク上の任意の位置から動的にアプリケーションをダウンロードして利用可能。アプリケーションの集中管理による管理コスト削減。クライアントサイドではシンクライアント[125]の利用によるセキュリティ向上と管理コストが低減できる。

---

121 サン社が Java で開発されたアプリケーションに対しておこなっている認定プログラムで、特定の OS の機能の使用を許可せず、どのような環境でも動作することを保証したもの。Java には各 OS の独自機能を呼び出す仕組みなどが備わっているが、100% Pure Java では、このような OS 依存の機能を一切使用せず、純粋な Java API のみを使用することが求められる。また OS によって異なる定数をベースにした実装なども許可されていない。

122 松下・臼井（1998）, p. 134-135 参照。

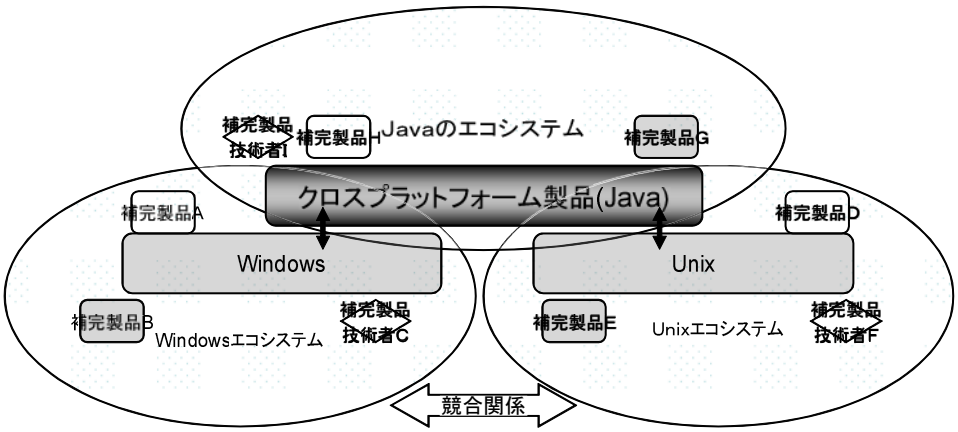
123 同上。

124 出所：http://www.weblio.jp/content/Java%E8%A8%80%E8%AA%9E 2013/9/23 をもとに筆者作成。

125 ユーザーが使うクライアント端末に必要最小限の処理をさせ、ほとんどの処理をサーバー側に集中させたシステムアーキテクチャ、またはその端末機の呼び名。

図７－１は、Java のクロスプラットフォーム機能により Windows と Unix の間を橋渡ししている状況を図にしている。それぞれのプラットフォーム製品は補完業者を誘引し、エコシステムを形成している。競合関係にある Windows と Unix を Java が橋渡しすることによって、両エコシステムが結びつく状態を表わしている。

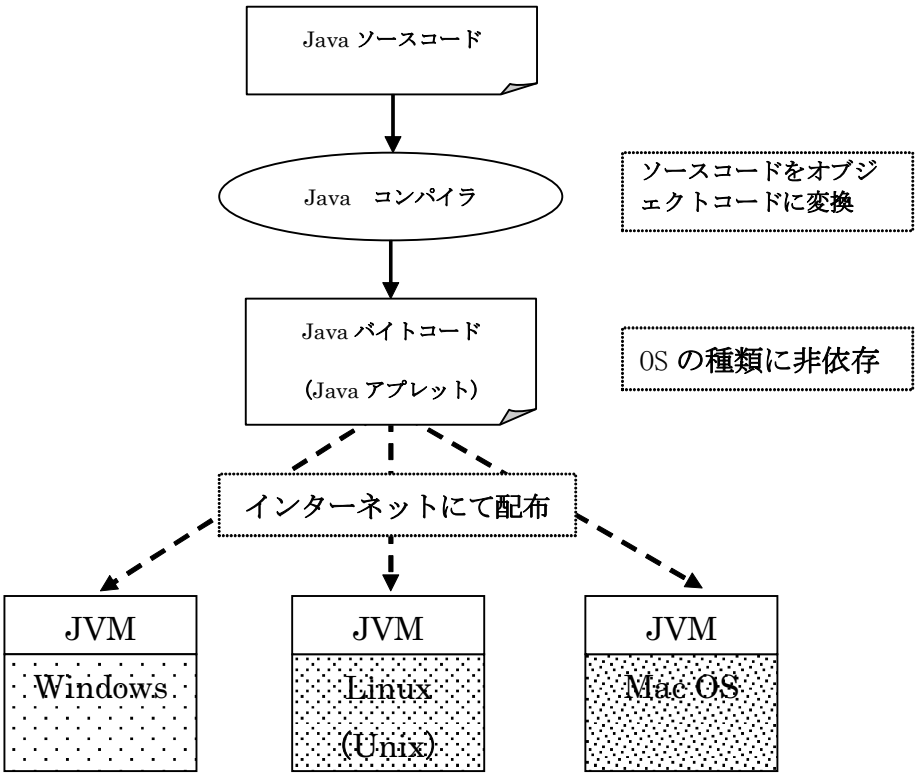
図７－１：Java によるクロスプラットフォーム



Java の起動するしくみ（クライアント側でのバイトコード・JVM の関係）を以下のような図で表現する（図７－２）。

Java をコンパイルするとそのソースコードが OS の種類に依存しない形式の中間コードに変換される。その中間コード（バイトコード）を Java 仮想マシンと呼ばれるインタプリタで実行する。この Java 仮想マシン（JVM）が機種の違いを吸収して Machintosh や Unix、Windows といった異なる OS 上でも同じアプレットの実行が可能となる。

図７－２　Java アプレットの起動するしくみ[126]



1995 年秋、米国コロラド州アスペンで会議がおこなわれ、1997 年 2 月に登場する Java 開発キット（JDK）1.1 のアーキテクチャと製品戦略について打ち合わせがおこなわれた[127]。この会議は Java をホームページのアニメ言語から様々な世界でソリューションを促すプラットフォーム製品に成長させる機会となった。内容はビジネス市場を明確に意識したもので、ネットワーク、セキュリティなどに対応する課題に加えて Java をしっかり定義するためのコアとなる API[128]の開発プロセスを他社と共同しておこなうことになった[129]。

126 出所：http://ossforum.jp/en/node/604　2013/9/23 をもとに筆者作成。
127 岩山（2001）, p.75 参照。
128 Application Program Interface の略語で、OS やミドルウェア向けのソフトウェアを開発する際に使用できる命令や関数の集合のこと。
129 岩山（2001）, p.75 参照。

具体的にはGUI[130]に関してはアップル社やネットスケープ社と共同で、2次元グラフィックスに関してはアドビ（Adobe）社と共同で、3次元グラフィックスに関してはSGI社[131]と共同で、データベースのストアードプロシージャーのJavaによる標準化はオラクル社、IBM社、タンデム社、インフォミックス社、サイベース社と共同でおこなうことに合意[132]した。また当時影響力が強い競合の１社であったIBM社と提携してJavaへの共同の取り組みをおこなった。IBM社は異なるサーバー用のプラットフォーム製品を抱えサンと競合する立場であったが、クロスプラットフォームとインターネットを即戦力化できる技術の点でJavaを評価し、同社のe-ビジネスの戦略的土台に据えた[133]。自社が開発した技術でないJavaをIBM社が率先して採用したことは、当時としては画期的な出来事であった。

2000年にIBM技術アカデミーのプレジデントになったイアン・ブラッケンベリー（Ian Brackenbury）は「（Javaの）活動を始めた頃は、殺されるかも知れないと思いましたが、今はもうJavaが当たり前です。IBM社にとってJavaとCORBA[134]を連携させることは重要であり、JavaのAPIはIBM社のミドルウェア戦略の一環になりました。」と語っている[135]。

加えて、サン社の創業者スコット・マクニーリ（Scott MacNery）はオラクル社のラリー・エリソン（Larry Ellison）を早速Javaの仲間に引き入れた。エリソンは1995年頃からネットワーク・コンピュータ構想を提唱し、それ以後Webブラウザーからデータベースを利用可能にするため、Javaを自社のデータベースに統合する努力を一貫して続けていくこととなった。ラリー・エリソンは「オラクル社はJavaのデバッグで最大の貢献をしている」と、その取り組みをユーザー・カンファレンスで皮肉交じりにしばしば語っている[136]。

---

130 Graphical User Interfaceの略語で、ユーザーに対する情報の表示にグラフィックを使用し、大半の操作をマウスなどによって簡単に行うユーザー・インターフェイスのこと。
131 Silicon Graphics, Inc. ソフトウェアやソリューションを手がけるアメリカの大手コンピュータメーカー。
132 岩山（2001）, p. 75参照。
133 岩山（2001）, p. 63参照。
134 業界団体OMGによって策定された、様々なプラットフォームでの分散処理の連携を実現するための基本仕様。
135 岩山（2001）, p. 63参照。
136 岩山（2001）, p. 62参照。

オープンな経営を標榜していたサン社は、IBM 社やオラクル社以外にも多くの補完業者と積極的に手を組み、補完製品としての Java の普及を推進することで新たな市場創造の戦略を選択した。主要な補完業者と提携した分野には、ビジネス・アプリケーション、デスクトップ・アプリケーション、システム・インテグレーション、OS、スマートカード、通信機器、情報家電など多岐に渡った。Shapiro & Varian (1999) は消費者が新しいテクノロジーに手を出すか出さないかを決める重要なファクターに期待感があると論じているが、Java に関して「サン社は Java を支持してくれる味方の集め方が、例えば Java を支持する企業をリストアップした全ページ広告を出すなど、非常に明快だった。サン社のこのやり方は、ネットワーク外部性が強力な市場において、期待感の醸成がいかに重要かを教えてくれる。」と指摘している[137]。

また「オープンな経営」を標榜していたサン社は多くの補完業者と積極的に手を組み、Java の普及を推進していった。表７－１は Java における陣営形成のため主要補完業者（ISV[138]s・SIer）との提携などを表にしたものである。

表７－１：主要補完業者（ISVs・SIer）との提携[139]

分野	補完業者	提携分野
ビジネス・アプリケーション・ソフト	オラクル SAP	Java ベースに対応した ERP パッケージの提供。
	IBM	Enterprise Java Beans の開発と提供。
デスクトップ・アプリケーション・ソフト	ネットスケープ	Java に対応したブラウザーの提供。
	ロータス	表計算・文書作成ソフトの開発と提供。
SI（システム・インテグレーション）	NTT データ 日本総研	Java を使ったビジネスソリューションの提供。

表７－２は主要補完業者（チップベンダー・通信機器ベンダー・情報家電メーカー・スマートカードベンダー）との提携などをまとめたものである。

137 Shapiro & Varian（1999），（邦訳）p. 483 参照。
138 特定のハードメーカーと関係をもたない独立系ソフト開発会社のこと。「Independent Software Vendor」の略。
139 松下・臼井（1998），p. 112 より筆者編集。

表７－２：主要補完業者（チップベンダー・通信機器ベンダー・情報家電メーカー・スマートカードベンダー）との提携[140]

分野	補完業者	提携分野
OS	NCR インテル 富士通	64 ビット MPU への Solaris の採用ならびに共同開発。
	IBM	NC 向け JavaOS の開発ならびに製品の提供。
スマートカード	VISA	スマートカード規格への Java の採用。
	シーメンス	Java を採用したスマートカードの生産。
通信機器	モトローラ NEC 富士通 IBM	Java 規格を利用した各種端末の開発と製品化ならびに販売。
情報家電	NEC 三菱電機	Java を利用した情報家電の開発ならびに提供。
	SONY	情報家電分野での Java の採用。
その他	松下電送	カラーファクシミリへの Java の採用と製品提供。

１）Java デビューの経緯と歴史

Java が誕生するきっかけは、1990 年に入社 3 年足らずの 1 人の 25 歳のプログラマーであったパトリック・ノートン（Patrick Naughton）がサン社を辞めたいと申し出たとき、CEO のマクニーリが「辞める前にサン社のまずいところを書き出してくれないか。それに、問題点を挙げるだけではなくて解決策も教えて欲しい。自分が神様になったつもりで、君だったらどうするか教えてくれないか。」[141]と依頼した一言が始まりと言われている。翌朝、この問いに答える 12 ページに及ぶ改善提案のメールを彼はマクニーリに送ったところ、マクニーリはそのメールを管理部門の全員に送り、新たなプロジェクトを、そのメールの指摘する問題意識からスタートさせた[142]。ただし、初期のサン社のビジネス的な成功により生まれてきた社内の保守的な気運が、この革新的なアイデアを潰してしまうことを恐れ、プロジェクトの使命はサン社の最高

140 同上。
141 松下・臼井（1998）, p. 155 参照。
142 同上。

首脳以外には秘密にされた[143]。このプロジェクトは「グリーン」というコードネームで呼ばれ、サン社の敷地の外にオフィスを借りてプロジェクトルームとした。初年度の予算は 100 万ドル[144]であった。

その後、グリーン・プロジェクトから「Beyond the Green Door（緑のドアを越えて）」という企画書が 1990 年 12 月に出された[145]。サン社の幹部であったジェームズ・ゴスリング（James Gosling）が開発した Java の前身となるプログラムが OAK[146]と命名されたのは 1991 年 8 月であった[147]。OAK の当面の売り込み先は三菱電機とフランス・テレコム社とした。三菱電機は産業オートメーションからチップまで製造販売をおこなっており、フランス・テレコム社の電話ベースのマルチメディア端末の制御に OAK は最適であると思われた[148]。しかし、これらの企業は自社製品の付加価値を高める１部品に、数ドル以上のコストをかけることには積極的ではなかった。次に OAK の売り込み先として、インタラクティブ・テレビ用のセットトップ・ボックス事業に目をつけた。ワーナー社のサーバーと家庭のセットトップ・ボックスが生み出すネットワーク上を行き来するデータを、総合的に OAK がコントロールするのが適していると考えた[149]。しかし、ワーナー社はサン社のライバルであったシリコン・グラフィクス社に白帆の矢を立ててしまった[150]。その次に接近したのは 3DO 社であった、3DO 社が松下電器と組んで家庭用ゲーム機を発売するも不振であったために、ゲーム機をセットトップ・ボックスとして利用し、OAK を利用しようと話が進んだ。ところが最後になって 3DO 社が OAK の独占権を要求し、サン社側がそれを拒否し交渉は破談となった[151]。

グリーン・プロジェクトはここまで全く良いところが無く、このまま終わってしまいそうな雰囲気であった。それを救ったのは経営幹部の OAK への信頼とリスクをとって挑戦する姿勢であった。数百万ドルを投資したにもかかわらず、重なる失敗を報告

---

143 サウスウック（2000）, p. 170 参照。
144 松下・臼井（1998）, p. 156 参照。
145 同上。
146 Object Application Kernel『オフィスの窓の外の右側に樫(OAK)の木があったから』と後にゴスリングが説明。岩山（2000）, p. 36 参照。
147 松下・臼井（1998）, p. 156 参照。
148 松下・臼井（1998）, p. 159 参照。
149 サウスウック（2000）, p. 174-175 参照。
150 同上。
151 サウスウック（2000）, p. 175-176 参照。

にくるプロジェクト・メンバーをマクニーリは決して怒ったり責めたりしなかったと言われている。そしてプロジェクト・メンバーに、こう言った「きっとうまくいく」[152]。そんな折 1994 年の春、ビル・ジョイ（Bill Joy）らのサン社の幹部がタホ湖のロッジに集まった際に、インターネットに OAK を持ち込んだらどうかというアイデアが、誰からともなく持ち上がった。その頃ジョイは「ただでばら撒くんだ。フランチャイズを作るんだよ。」と言っていた[153]と伝えられている。当時、インターネットは破竹の勢いで普及拡大しつつあった。シェアを可能な限り早い段階で拡大するための方策は、プログラムのソースコードをインターネットで発表することである。ただし、それにより当面は利益を確保することは困難となる。そればかりでなく、そのような思い切った市場投入戦略は大きなリスクを伴う。なぜなら上手くいかない場合は、開発に費やした資金を回収できないばかりでなく、重要な技術情報が競合に流れ出し、自社にとって取り返しのつかない大きな損害となることは明らかであるためである[154]。このような大きな賭けに出ることは、当時の幹部のひとりであったエリック・シュミット（Eric Schmidt）でさえ自信がなかった。シュミットは、「４年に及ぶ開発の末、数百万ドルもの価値のあるソフトウェアの無料サンプルをばら撒くというアイデアに、サン社の経営幹部は尻込みするのではないかとエンジニアたちは心配していた。（中略）それを最初にスコットに話しをした。彼はすぐさま理解を示してくれた。」と後に説明している[155]。

1995 年 1 月に、OAK はインターネット仕様に作り変えられ、名前も Java と命名された。その 3 か月後にはネットスケープ社のマーク・アンドリーセンにコピーが送られた。こうして Java がインターネットに登場した。Java はサン社のホームページで公開され、個人であれ法人であれ必要とする人は誰でも、自由にダウンロードすることができるように提供された[156]。デビューに至るまで、丸５年に及ぶ試行錯誤のなか、遂にサン社は大きな賭けに出た。

アップル社から移籍してきたエンジニアで、Java が企業のシステムをどう変革させるかについて体系化したといわれているバド・トリブル（Bud Tribble）は Java の成

152 松下・臼井（1998）, p. 162 参照。
153 松下・臼井（1998）, p. 164 参照。
154 同上。
155 松下・臼井（1998）, p. 165 参照。
156 同上。

功に関して、「Java の開発当初から、今のような成功をイメージしていたわけではない。実は苦闘続きだった。最初は、機械制御の言語にいいと思ったのだがビジネスにならなかった。それでも、資金をつぎ込んだ。次に CATV のセットトップ・ボックスに最適だと思ったのだが、これも駄目だった。けど、資金を止めなかった。これはいいものだとみんな信じていた。そしてインターネットと出会って爆発的に市場が立ち上がった。（後略）」とコメントしている[157]。

Java のあゆみ（1991 年－2000 年頃まで）

以下では、Java というプログラム言語が誕生した初期の経緯を時系列で簡単に説明する[158]。

1991 年　OAK 誕生

- ✓ サン社で開発されたプログラム言語で、もともとは家庭用電気製品のプログラムの開発言語であった。
- ✓ 「Green」と呼ばれたプロジェクト・チームが Java のコンパクト性に注目し、家電に埋め込まれたコンピュータ・チップ上での開発をおこなった。

1995 年　Java に名前を変える。「Java」正式発表

- ✓ それまでは専門的なプログラムは C++と言う言語を使うのが一般的であったが、OS の種類に依存しない、信頼性の高いプログラム言語 Java に注目が集まる。
- ✓ Netscape Navigator バージョン 2.0 から Java(アプレット）実行環境を組み込む。インターネットの発展とともに急速に普及が始まる。
- ✓ IBM 社、Java ライセンス供与受ける。
- ✓ 富士通、Java ライセンス供与受ける。

157 松下・臼井（1998）, p. 158 参照。
158 末松・ベネット（1996）, p. 38-41 ならびに松下・臼井（1998）, p. 288-290 をもとに筆者加筆。

1996 年　Java ソフト社設立

- ✓ OS ベンダー10 社が Java を採用する。
- ✓ 20 社が JavaOS のライセンス供与を受ける。
- ✓ Java チップが発表される。
- ✓ 100％ Pure Java プログラムを発表する。
- ✓ 三菱電機、Java ライセンス供与を受ける。

1997 年　Enterprise・JavaBeans、PersonalJava、JFC などを発表

- ✓ IBM 社がサンフランシスコ・プロジェクトを発表。
- ✓ マイクロソフト社を契約不履行で提訴する。

1998 年　サン社が Java コミュニティプロセス（JCP)を導入

- ✓ Jini プロジェクトを発表。
- ✓ アップル社 Quick Time for Java を発表。
- ✓ モトローラ社、Java ライセンス供与受ける。

1999 年　サン社が各種用途向け API J2SE[159], J2EE[160], J2ME[161] を発表

2001 年　NTT ドコモ社が Java の技術をベースとした「i アプリ」のサービスを開始

- ✓ J-フォンでは「Java アプリ」KDDI(au)社では「ezplus」を開始。

2006 年　Java SE と Java ME をオープンソース化（GPL 2）

Java は、以下に示すように広く普及している（2010 年データ）[162]。全世界に多数の開発者をかかえプログラム言語としての地位を築いている。

- ➢ 4.5M Java Developers：450 万人の Java の開発者が存在する

159 J2SE：Java2 Platform Standard Edition （現在の JavaSE 後述）
160 J2EE：Java2 Platform Enterprise Edition（現在の JavaEE 後述）
161 J2ME：Java2 Platform Micro Edition（現在の JavaME 後述）
162 サン社公式ウェブ参照 http://jp.sun.com/java/everywhere/ 2010/05/26

- 950 JCP Members：950 人の JCP の登録メンバーが存在する
- 180 Authorized Licensees：180 社のライセンシ企業が存在する
- 635 Handset Models：635 機種の Java 対応電話モデルが発表されている
- 700M Desktop Computers：7 億台の Java 対応デスクトップが使われている
- 1.0B Java-powered Phones：10 億台の Java 搭載の通話端末が存在する
- 1.25B Java Smart Card deployed：12 億 5 千枚の Java スマートカードが世に出ている

2）サン社の生い立ち

以下、サン社の創立期について記述する[163]。

米国スタンフォード大学で校内のネットワーク用のワークステーションを独自に組み立てたアンディ・ベクトルシャイム（Andy Bechtolsheim）が、２年早く別の会社を設立していたビノッド・コースラ（Vinod Khosla）、同大学の MBA を取得して別の小企業にかかわっていたスコット・マクニーリ（後の CEO）らとともに会社を 1982 年に創立したのがサン社の始まりである[164]。サン社の名前は、Stanford University Network の頭文字 SUN から由来する。その後、カルフォルニア大学バークレー校で BSD Unix を開発していたビル・ジョイを創立メンバーに加える[165]。このようにして、ソフトウェアの開発はジョイ、ハードウェア開発はベクトルシャイム、製造部門はマクニーリがそれぞれ担当し、社長としてコースラが経営全般にあたる。1982 年、全員が 26 歳と言う若いベンチャー企業が誕生する[166]。

シリコンバレーに位置するスタンフォード大学は、1960 年代に当時学部長のフレデリック・ターマン(Frederick Terman)教授のインダストリアル・パークの企業誘致活動による、ウイリアム・ショクレー(William Shockley)の研究所やインテル社、AMD 社を始め、後にヒューレット・パッカード（HP）社、アップル社、シスコシステムズ社、ヤフー社、グーグル社など多くの IT 関連企業の創発の舞台となる大学である。

---

163 記述に関しては、主としてホール・バリー（1991）、根本・松岡（1992）ならびに松下・臼井（1998）からの引用による。
164 根本・松岡（1992）, p. 21 参照。
165 同上。
166 同上。

1982 年、同大学の博士課程の大学院生であったベクトルシャイムは、高機能で高速な自分の理想のコンピュータを自由に使いたいと考えていた。理由は自らの博士論文のための研究において 1 台のコンピュータを、研究者が共同で使用しなければならない不便さを感じていたからである[167]。ベクトルシャイムの望むようなコンピュータは、当時ひとりの人間が独占して使えるような環境ではなかった。何人もの研究者で共有する汎用コンピュータは、気難しいシステム・オペレータに管理され、自分のプログラムを走らせるためには長い待ち行列に並ぶしかなかった[168]。デジタル・イクイップメント（DEC）社やデータゼネラル社の提供するミニコンピュータは、同時にプログラムを走らせると極端に処理スピードが低下した[169]。パーソナルコンピュータは価格面では安価で入手しやすいものであったが、なにぶん性能や処理速度において満足のいくものではなかった[170]。

性能的に満足できるワークステーションは、メーカーが CPU などの部品や搭載するソフトウェアを独自に開発するため、当時はかなり高価なものとなってしまっており、一介の大学院生が個人的に購入するには手が出ない代物であった[171]。そこで、ベクトルシャイムは既に市場に出回っている部品を使って、自分の好む性能を持ったワークステーションを安価で短期間で組み立てることを考えた。「スタンフォード大の学生として、私にはそれほど選択肢はなかった。」と述べている[172]。既成部品の組み合わせでコンピュータを作るという発想は、後にデル・コンピュータを創業するマイケル・デル（Michael Dell）の考え方と同類のものと言えるが、ベクトルシャイムはそういった商機よりも一介の学生として博士論文を完成させることに興味があったようである。

ベクトルシャイムの考案したワークステーションは研究仲間の間で注目を集め、同様に高性能で安価なワークステーションの必要性に賛同するもうひとりのスタンフォード大の卒業生であるコースラは、事業家としての道に強い興味があり、ベクトルシャイムを説得し、一緒にビジネスを興す約束を取り付ける[173]。「僕が欲しいのは金

---

167 同上。
168 松下・臼井（1998）, p. 51 参照。
169 同上。
170 同上。
171 同上。
172 根本・松岡（1992）, p. 22 参照。
173 根本・松岡（1992）, p. 24 参照。

の卵ではなく、金の卵を産むガチョウだ。二人で大きな会社を築いて、数百万ドルの利益を上げることだってできる。会社の設立者は君だ。」と博士号取得にしか興味が無かったベクトルシャイムを説得した[174]。コースラはインド工科大学、カーネギーメロン大学を経てスタンフォード大学の MBA を取得、その後デイジー・システムズ社という工作機械メーカーで働いていた[175]。コースラはビジネスを展開していくにあたり、３つのものを用意しようと考えた。ひとつめは資金である。先ず、ベンチャーキャピタルから資金を調達する合意を得た[176]。ふたつめに製造部門を任せられる人物の登用であった。コースラは旧知でスタンフォード大の MBA 修了生であったマクニーリを当時勤めていたオニックス・コンピュータ社の製造部門より引き抜いた[177]。最後に OS である。ベクトルシャイムとコースラとマクニーリの３人は OS （特に Unix）開発に卓越した能力を有するビル・ジョイをヘッドハントするために訪ねた[178]。ジョイはその当時から、Unix の魔術師と呼ばれるほど、その分野の第一人者であったと言われている。ジョイはカリフォルニア大学バークレー校に部屋を持ち、初期の Unix を改良して BSD 版 Unix を完成させていた[179]。ジョイは訪ねてきた３人をコンピュータのキーボードから目を上げてしばらく見るが、再びキーボードをたたく仕事に戻ってしまったという。マクニーリは後のインタビュで「なぜ我々を無視したのかとビルにたずねたところ、彼が待っていたのは自分と同じジーパン姿の人間ではなく、いかにもトップの経営者という風貌の人間だったというのです。」と語っている[180]。結果的にはジョイは３人のオファーを受け入れサン社の創業メンバーとなる。

ベクトルシャイムの考案したワークステーションはハードウェアだけであり、ハードを制御する OS に何を採用するかは未だ決まっていなかった。そのため、創業メンバーはいちから開発に時間と資金が多く必要な自前の OS ではなく、当時多くの研究者に既に受け入れられている Unix を自分たちのワークステーションに搭載することにした[181]。当初ワークステーションの OS は、BSD Unix を採用し SunOS と呼ばれてい

174 同上。
175 松下・臼井（1998）, p. 52 参照。
176 根本・松岡（1992）, p. 25 参照。
177 同上。
178 根本・松岡（1992）, p. 28 参照。
179 同上。
180 根本・松岡（1992）, p. 29 参照。
181 松下・臼井（1998）, p. 55 参照。

たが、後に当時 Unix を所有し Unix System V[182]をライセンスしていた AT&T 社と共同して、BSD Unix と System V を一部合流し商用 OS の Solaris として販売を開始した[183]。サン社の Unix はジョイの考え方を具現化したものであり、OS に限らず Java の開発においてもジョイの貢献は計り知れないものがある。

ジョイがサン社に加わったことは、サン社にとって大きなマーケティング効果を発揮することとなる。まずサン社＝Unix というイメージが定着し、Unix 分野で主導権をにぎることに成功した[184]。それと同時に、ジョイが率いるサン社に加わりたいと考える優秀なエンジニアが集まった。1988 年 11 月サン社が従業員に見せたビデオの中で、マクニーリはジョイのことを「サン社の頭脳であり教祖だ。」と紹介している[185]。

ネットワーク・コンピューティング（ネットワーク上の様々な要素に機能を分散し、分散された情報に容易にアクセスできること）は、当時まだ登場したばかりの新しい技術であったが、サン社はその潜在的能力を同社のマシンに常に取り入れてきた。サン社の経営は企業理念でもある“The network is the computer”[186] というフレーズを唱え、ネットワークとコンピュータは一体のものという思想に基づいていた。

サン社は数ある IT ベンダーの中でも、オープンな経営を強力に推進する企業となる。自社の技術を公開したりライセンスしたりすることを厭わず、オープンな戦略を特徴としていた[187]。例えば、多額の投資によって開発した先進的な技術を独占的に使用しようとするのではなく、「業界全体の進歩のため」という理由で惜しげもなく公開してしまう極めてオープンなスタンスを取っていた[188]。加えて、オープンソース系のコミュニティに対する支援にも積極的であり、そのような姿勢が多くのソフトウェア開発者を含む多くの技術者から支持を受けていたことが、同社の経営上の大きな資産でもあった[189]。

1990 年代前半、いち早いインターネットに向けた Unix サーバー群の販売により、米国を中心とする世界市場において、サン社は一人勝ちの様相を示した。インターネ

---

182 AT&T 社が、同社（当時）のベル研究所で開発された初代 Unix から派生する形で開発した商用の Unix システム。
183 松下・臼井（1998）, p. 55 参照。
184 同上。
185 ホール・バリー（1991）, p. 93 参照。
186 ホール・バリー（1991）, p. 228 参照。
187 松下・臼井（1998）, p. 92-94 参照。
188 同上。
189 同上。

ットの時代にビジネス展開を考える企業が、こぞってサン社のハードウェアならびにソフトウェアを購入したためである[190]。日本市場では富士通・東芝・CTC などと提携し、当時、通信系や企業基幹系に浸透しつつあった Unix 市場において、その価格性能比と知名度で大きくシェアを伸ばした。結果として、HP 社と日本電気の陣営や、IBM 社と日立の陣営などと並び、有力な商用 Unix 系ベンダーのひとつとなった[191]。

3）Java のプログラム言語としての設計上の特性

Java のプログラム言語としての特徴としては、コンパイルしたファイルは中間言語に変換され、Java 仮想マシン上で実行されるため、OS 環境に依存しないアプリケーションが作れるということであった。これ以外にも、ジェームス・ゴスリンやビル・ジョイの理想とする言語として、設計上の特徴として以下が挙げられる[192]。

まず始めに、オブジェクト指向言語であり、コードの生産性が高いという点である[193]。オブジェクト指向言語とは、データとそれを操作する手続きをオブジェクトと呼ばれるひとまとまりの単位として一体化し、オブジェクトの組み合わせとしてプログラムを記述するプログラミングの技法である。これによりプログラムの部分的な再利用がしやすくなるなどのメリットがある[194]。代表的なオブジェクト指向言語としては、Java 以外では C 言語にオブジェクト指向的な拡張を施した C++言語や、Xerox 社の Smalltalk、NeXT 社(現アップル社)が自社の OS である NeXT STEP 向けアプリケーション・ソフト開発用に開発した C 言語ベースの Objective-C などがある[195]。

Java のプログラミングのシンタックス（構文）は C 言語に極めて近い[196]。C 言語のプログラマーが C++以上に徹底したオブジェクト指向でプログラムできるように設計されている。世間の Java に対する関心の高さと ISV の積極的な対応は、Java が C 言語の普及を引き継ぎ、オブジェクト指向の理想をさらに徹底して追求したことに由

---

190 出所：http://www.yano.co.jp/opinion/090701.html　2013/9/23
191 出所：http://www.itmedia.co.jp/enterprise/articles/0711/27/news079.html　2013/9/23
192 特徴の記述に関しては、主として岩山（2001）からの引用による。
193 岩山（2001）, p.48 参照。
194 同上。
195 同上。
196 同上。

来する[197]。オブジェクト指向に開発者の関心が集まるのは、ソフトウェア開発を迅速化することが可能になるためである。

次に挙げられるのが、ガーベジ・コレクション（自動ゴミ集め機能）を備えており、メモリ管理の手間が非常に少ないという点である[198]。ガーベジ・コレクションとは、OS やプログラミング言語の実行環境などが持つメモリ管理機能のひとつで、実行中のプログラムが使用しなくなったメモリ領域や、プログラム間の隙間のメモリ領域などを集めて、連続した利用可能なメモリ領域を確保する技術である[199]。ガーベジ・コレクションがうまく機能しないと一度確保されたメモリ領域が再利用できず、次第に利用可能なメモリが減ってゆくため、OS やアプリケーション・ソフトの再起動などを強いられることになる[200]。Java は、こうしたメモリ管理が JVM によって自動的におこなわれる。JVM は、Java プログラムのどこからも参照されなくなった不要な Java オブジェクトを見つけ出し、そのメモリ領域を自動的に解放する機能をもっている[201]。

加えて、ガーベジ・コレクションに関連して、ポインタを廃止したことにより、不正なメモリ参照が起きにくくなった[202]。ポインタとは、ある変数のメモリ上のアドレスを記憶しておくために使う変数であるが、開放し忘れたり、開放の指定をすると他のバグがそれに関連して発生したりする。ガーベジ・コレクションは、オブジェクトがメモリを利用した後、そのメモリの領域を自動的に開放し、ポインタ演算やメモリ割り当てに関する関数を排除した[203]。

プログラムが予想外の振る舞いをする可能性を徹底的に排除すること、これが Java の設計で目指したもののひとつである[204]。Java が安全な言語であるといわれるのは、暴走など人間が意図していないプログラムの振る舞いの原因を取り除いたアーキテクチャになっているからである。ガーベジ・コレクションによるメモリ・マネジメントの自動化はバグが発生する可能性を減らし、プログラマーの開発における生産性を向上させただけでなく、記述されたプログラムの安全性を強化した[205]。これらのこと

197 同上。
198 岩山（2001）, p. 47 参照。
199 同上。
200 同上。
201 同上。
202 同上。
203 同上。
204 同上。
205 同上。

により、インターネットなど不特定多数の人によって接続されるネットワーク・システムで安全に使われる言語として普及していった。

ジョイはゴスリングの採用したガーベジ・コレクションを高く評価し、「ジェームズがガーベジ・コレクションを OAK そして Java で導入して、プログラムをモジュール化することが出来ないという問題から開放してくれました。Java の歴史的な意義を考えると、ガーベジ・コレクションは大きな成果のひとつとして評価されると考えています。ガーベジ・コレクターによって構築されたシステムであるが故に、信頼性が驚くほどに向上したのです。Java のコードはどこでも走り、数多くのデバイスに組み込むことができます。そして、信頼できるコンポーネントのライブラリを、自信を持って開発することができるようになりました。」と語っている[206]。

こういったこと以外にも、多重継承を禁止しているため、メソッド呼び出しの際のあいまいさが排除された[207]。静的な型づけを採用しているため、変数に違う値などが入って起こるバグを最小限に抑えられる[208]。などのメリットが備わっていた。

Java は、デスクトップはもちろん、サーバー、携帯端末など、さまざまな用途で使われているため API の数も膨大な数となる。そのため、Java の API は次の 3 つのエディションを規定している。それぞれの用途は以下である（以下、該当記載部分の抜粋）[209]。

○Java SE（Java Platform Standard Edition）

Java の標準的な規格。一般の学習用や簡易なプログラムの実装から、ワークステーション、PC やサーバーなどの機器で、汎用的な用途でも使用されている。Java のプログラムで役立つ基底の API を含んでいる。実行環境や開発環境である JRE や JDK がインターネット上から簡単に入手可能であり、情報も充実している。

---

206 岩山（2001），p. 48 参照。
207 出所：http://ossforum.jp/en/node/604　2013/9/23
208 同上。
209 出所：http://www.atmarkit.co.jp/fjava/rensai4/java_dotnet01/01.html　2013/9/20 より抜粋。

○Java EE（Java Enterprise Edition）

Java の中でもエンタープライズ・システム開発に対応したもの。サーバーサイド機能を備えた実行環境、開発環境の規格。Java SE にあるクラスすべてを含み、ワークステーションよりもサーバー上でプログラムを動かすのに、より役立つ機能や、大規模システムを開発するための、さまざまな API が追加されている。

○Java ME（Java Platform Micro Edition）

携帯電話、PDA、テレビのようにリソースが制限されたデバイスに載せるための Java のエディション。一般的に PC などで利用する環境より、より小さな環境で動作するための Java の小型のセット。

4）JCP（Java Community Process）と SDC（Sun Developers Connection）

サン社は Java の独立性と公平性を保つために、1998 年 Java コミュニティプロセス（JCP）を導入した[210]。導入の背景には、競合からのサン社の Java に対する強すぎる関与を批判する動きへの対応という側面があった。Java 発表当時の 1995 年頃にサン社がもっていたシステム仕様などに関する管理を JCP に移管した。JCP においては、Java 言語を拡張する必要を感じる人々は誰でも、JSR（Java Specification Request）を申請することができ、その必要性が認められると有識者からなるエキスパート・グループの編成が呼びかけられる。個々の活動は、JCP 配下の各専門グループである JSR でおこなわれており、そこで Java のスペックの標準化と拡張が推進されている[211]。JSR はシステム構築の対象分野ごとに 100 以上のグループに分かれており、それぞれのグループで、対象分野に関する拡張スペックの仕様の作成、最終的な仕様の一般公開の準備活動をおこなっている[212]。

具体的には、JCP には JCP メンバーの投票によって選任されたエグゼクティブ・コミッティが存在する。エグゼクティブ・コミッティは「デスクトップ/サーバー」担当と「組み込み」担当に分かれている。提案された仕様は、担当のエグゼクティブ・コミッティの承認を得て、JCP のプロセスに乗せるか否かの承認を得る。承認された仕様には JSR が振られ、次の２段階のステップに進む[213]。

---

210 出所：https://www.jcp.org/aboutJava/communityprocess/background.html 2013/9/20
211 出所：http://jcp.org/aboutJava/communityprocess/JCPoverview-ja.pdf 2013/9/20
212 同上。
213 同上。

最初のステップは「コミュニティドラフト」である。ここではJCPのコミュニティメンバーとエグゼクティブ・コミッティによりレビュがおこなわれる。エグゼクティブ・コミッティの承認を受けて、次のステップである「パブリックドラフト」に移行する[214]。この段階では、仕様がインターネットに公開され、不特定多数の人がその仕様に対するレビュをおこなう。よって多数のフィードバックがあり、仕様にはそれらを盛り込んでつくられていく。こうして２段階で多くの人のレビュを経て、最後のエグゼクティブ・コミッティの承認を得てリリースされる。最終版のリリース後も、それ以降に発見されたバグはその都度修正し、エグゼクティブ・コミッティの承認を受けてリリースされる[215]。

J2EE（Java 2 Platform, Enterprise Edition）、J2ME（Java 2 Platform, Micro Edition）は、このJCP/JSRから生まれた代表的な規約群である。

2000年の6月にJCP2.0が発足する。そこでは22社によるエグゼクティブ・コミッティが形成された[216]。

加えて、JDCのようなコミュニティを利用する特徴として、サン社傘下にSDC（サン・デベロッパーズコネクション）という独立組織を発足させ、開発者や利用者のためのサポートをおこなった[217]。背景としてはJava技術に対する需要が急速に高まり、サン社がそれに対応しきれなかったということがある。そのためソフトウェア特有の品質（バグの修正）の問題や開発者の意見交換についてSDCという組織を発足させて開発者間での情報の交流によってサポートすることにしたのである。この活動によってJavaの品質を高めることに成功した。

以下は、SDCのウェブにつくられたJava関連コミュニティのためのサイトのいくつかを記載する（以下、該当記載部分の抜粋）[218]。

○Java フォーラム：Javaに関する情報交換をおこなうことを目的としたコミュニケーション・フォーラム。

---

214 同上。
215 同上。
216 出所：http://www.nikkeibp.co.jp/archives/105/105894.html 2013/9/23
217 出所：http://itpro.nikkeibp.co.jp/as/sun_jirei/sun_solution/86/03.html 2013/9/23
218 出所：http://otn.oracle.co.jp/technology/global/jp/sdn/java/community/index.html 2013/9/20 より抜粋（一部筆者修正）。

○日本 Java ユーザー・グループ：Java 技術の向上・発展、開発者の支援を目的とした任意団体。

○java.net（英語）：サン社のエンジニア、研究者、技術者、エバンジェリストなど、さまざまなメンバーで構成されるグループで、Java テクノロジーに関する議論や開発プロジェクトのための共有サイト。

○NetBeans.org：日本語ポータル：幅広く利用されている NetBeans IDE を開発することを目的とした、オープンソース・コミュニティー。

○Bug Database（英語）：バグの調査・報告のためのサイト。Java のバグ情報の検索、報告をおこなうことができる。

5）サン社の経営方針と Java の普及

以下、サン社の経営方針と Java の普及に関する方針について記述する。

サン社は補完製品として自社の OS を有償で提供し、この OS に関するサポートを収入源のひとつとしている。また自社製 OS と親和性の強い RISC[219]チップならびにその周辺機器や筐体などのハードウェアを有償で提供しており、収入の大部分はここで確保している。同時に自社製 OS のライセンスやチップを他の補完業者に OEM として提供し収入をあげている。

サン社は提携などによって多くの ISV などの補完業者の支援をとりつけているが、サン社自身も Java をサポートするソフト会社を増やすためにいくつかの手を打った。1996 年 8 月サン社・伊藤忠・米大手ベンチャーキャピタルのクライナー・パーキンス・コーフィールド・アンド・バイヤーズ（KPCB）は、IBM 社と組んでアメリカに 1 億ドルのベンチャー投資基金を設立している[220]。目的は Java を業界標準とするために、Java で作成したソフトなどを開発するベンチャー企業に投資することである。

1996 年、サン社は既存のソフトウェア部門とは別に Java ソフト社を発足させる[221]。しかし、サン社は必ずしも社内の意見を統一できてはいなかった。Java は急速に普及していったが、サン社内の意見の相違により部門間の対立が生まれていたと言われて

---

219 Reduced Instruction Set Computer の略、縮小命令セットコンピュータ個々の命令を簡略化することにより、並行して複数の命令を処理する方式で、効率を高め、処理性能の向上をはかっている。

220 松下・臼井（1998）, p. 130 参照。

221 サウスウック（2000）, p. 192 参照。

いる。 Java ソフト社は Java のライセンス普及推進におけるジョブ１をサン・ソフト社[222]はジョブ２をそれぞれのミッションと捉えていた。このジョブ１とジョブ２が互いにぶつかり合うことがしばしばあった[223]。

ジョブ１＝より多くの Java のライセンスを普及させること＝Java ソフト社はこの技術を可能な限り多くの企業や開発者が利用できるようにして、広い基盤をもつ標準技術として普及させたいと考えていた。

ジョブ２＝ Java 事業自体とは関係しないサン社の法人市場での売上を伸ばすこと＝サン・ソフト社は Java をサン社の既成技術、例えば Solaris などとシームレスに動作するようにして、法人市場での売上を伸ばしたいと考えていた。

「目標はサン社の売上と利益を最大化することなのか、それとも Java のライセンス普及を推進することなのか」この時点でサン社は社内的に統一した結論を出せなかった。このふたつの意見の対立は、その後もサン社内部で大きな議論の的となっていたと言われる[224]。

６）サン社の経営とマイクロソフト社の経営

サン社とマイクロソフト社の経営手法における違いは、Java の思想に代表されるサン社のオープンな経営思想と、クローズドと呼ばれるマイクロソフト社との対立を招いている[225]。

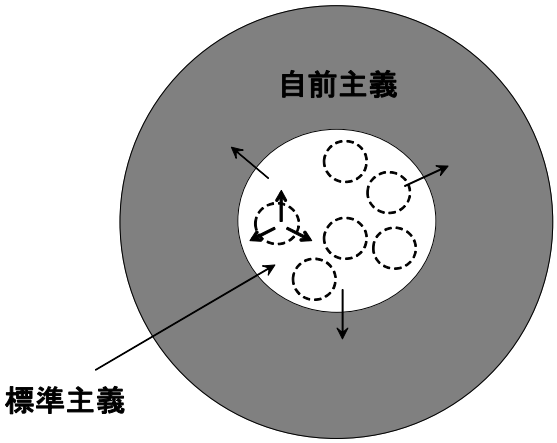
山田（2000）は、以下（図７－３）のような図を用いて、両者の経営思想の違いを論じている。ドーナッツ状になっているグレーの円の部分は、マイクロソフト社などプラットフォーム製品を自社単独で支配したいと目論む自前主義の従来型の多くの企業が属している。その内側にオープン・スタンダードでやろうとする企業が属する。当時、内側に属している企業はサン社１社だけであった[226]。当時はサン社のような型

---

222 サン社のソフトウェア部門で 1991 年分社化、後に統合によりサン社内部の部門となる。
223 サウスウック（2000）, p. 205 参照。
224 同上。
225 対立の記述に関しては、主として山田（2000）からの引用による。
226 山田（2000）, p. 130 参照。

破りな考え方で経営を進める企業は他にほとんど存在しなかったため、サン社のマネジメント層はサン社の製品を単に宣伝して売っていくというよりは、自社の経営方針や Java のような製品の思想を啓蒙することに力点を置いていた[227]。つまり、図７－３で表される内側の白い部分のエッジを、グレーの部分に向けて広げる活動を、経営に織り込みながら販売を伸ばしていくことになる。従来型のマーケットを変えていこうとしているのであるが、サン社１社でできることは知れている。なぜなら、サン社を除くほとんどすべての企業はグレーの自前主義に属しているのである。よって、共感する仲間を募って、グレーに属している企業を内側に取り込んで進めていくことになる。IBM 社やモトローラ社やオラクル社は、もともとグレーの自前主義に属する企業であったが、Java を契機としてサン社と Java 陣営を結成した[228]。サン社のやり方は、まず事業化し、技術をオープンにし、周囲に周知させ、それにより内側のエッジを広げて白い部分を大きくすることでパイが大きくなり、自分の取り分も増やしていくというものである[229]。

図７－３：オープン・スタンダードの陣営形成[230]　（山田（2000）より）



サン社とマイクロソフト社は水と油のごとく考え方が違うので、コンソーシアムのような共同作業を進めていくことは、かなり困難である。

227 同上。
228 同上。
229 同上。
230 山田（2000）, p. 131 図 8-1 の抜粋。

この白とグレーの鬩ぎ合いはなくならない。じわじわと広がっていく白の力と、グレーの押し戻す力がぶつかり合う[231]。これまでのデファクト・スタンダードの話は、グレーのもつ自前主義の論理のなかで考えられていた。言い換えれば、グレーに属する各企業の勢力争いのなかでのスタンダードであった。これに対し、内側の企業はスタンダードをつくっていくこと自体が生きていくための土台となっている。いわば本気のスタンダード[232]といえる。

また、山田（2000）は、サン社の標準主義とマイクロソフト社の自前主義の対立について、「今考えられているネットワークは、地球上に存在するすべての人のみならず、家畜なども含めて、繋がるモノはすべて繋げようとしている。このような流れと、可能であれば100%自分で市場を独占したいという従来の自前主義の経営精神が結びつくと、ひとつのアーキテクチャに収束せざるをえない。マイクロソフト社は独占に関して批難されることが多いが、それはビル・ゲイツ（Bill Gates）会長の意志に原因があるというよりも、全部繋げるときは、ひとつしか残らないメカニズムが働き収斂することに原因があるとみた方がよい。」と説明している[233]。

Java のシェアが高まり出すと、マイクロソフト社にとって、Java は目の上の瘤となる。末松（2002）は、「Java はあらゆる OS の上で動作し、（中略）ここで重要なのは、あらゆる OS の上で機能するという点で、システムは Windows でなくてもよいことになる。つまりプラットフォームとして強大な地位を築いてきた Windows を One of them にしてしまい、サン社自らがプラットフォームの地位を奪い取ろうとする（中略）。Windows でなければならないというプラットフォームの優位性で市場を支配してきたマイクロソフト社にとって、重大な脅威（後略）。」と説明している[234]。ことあるごとに市場ではマイクロソフト社が Java の普及に対し様々な妨害をおこなってくる。オープン性を唱え、開発者コミュニティを活発化する Java に対して、マイクロソフト社は「Java はオープンではない」、「Java はサン社の自社利益独占のためにある」と攻撃した。また、「（Java の）ライセンス料は将来、普及した後で大幅に吊り上げられるかもしれない。」と誹謗した[235]。加えて、マクニーリは開発者のイベントやユ

---

231 山田（2000）, p. 131 参照。
232 同上。
233 山田（2000）, p. 128 参照。
234 末松（2002）, p. 68 参照。
235 末松（2002）, p. 69-79 参照。

ーザー会での発言のたびに、マイクロソフト社批判を繰り返した。このような様子に、業界ではサン社とマイクロソフト社は犬猿の仲と呼ばれるようになっていく。

7）マイクロソフト社の対抗製品 ActiveX Control[236]

1996 年にマイクロソフト社が発表した ActiveX は サン社が開発した Java に対するマイクロソフト社の対抗戦略である。ActiveX Control は機能的には、Java アプレットとほぼ同じものと考えられる[237]。ActiveX 環境で実行するプログラムを作成するときに中心になるのはコンポーネントとなる。これは、ActiveX ネットワーク（現状では、Windows と Macintosh のシステムで構成されるネットワーク）内のどこからでも実行できるプログラムである[238]。このコンポーネントを ActiveX Control とも呼ぶこともあるが、単に ActiveX と呼ぶ場合も多い。

ActiveX Control は、ActiveX の登場以前に OLE（Object Linking and Embedding）[239]と呼ばれていた技術群をインターネット技術に適応させたものである[240]。もともと OLE はコンピュータ上のアプリケーション間で機能の連携をおこなうための技術であったが、ActiveX Control ではインターネットを通じて、サーバーが提供する機能モジュールをブラウザーで実行できるようになっている[241]。

しかし、ActiveX control は Java のアプレットと異なり Windows 上でしか使えない。ActiveX Control を採用するサイトでは、Internet Explorer(もしくはプラグインを導入した Firefox)以外のブラウザーでは、アクセス出来ない。また Unix 系の OS ではサポートされないケースが多い[242]。具体的には、マイクロソフト社の製品以外では「プラグイン」が必要になり、JVM も「アクティブ X 用 JVM」にしなければならない。問題はその両方ともマイクロソフト社が厳しく管理しているという点である[243]。

---

236 記述に関しては、主として末松・ベネット（1996）, p. 119-123 ならびに http://www.weblio.jp/content/ActiveX からの引用による。
237 末松・ベネット（1996）, p. 119 参照。
238 同上。
239 OLE とは、あるアプリケーションで作成している文書の中に、別のアプリケーションで作成した情報を埋め込んだり、別のアプリケーションの機能をあたかも自分の機能であるかのように提供することができる技術。末松・ベネット（1996）, p. 119 参照。
240 末松・ベネット（1996）, p. 119 参照。
241 同上。
242 末松・ベネット（1996）, p. 122 参照。
243 同上。

ActiveX Control の他にも、Web ブラウザーで Office ドキュメントを表示できる ActiveX Document や、ActiveX Control やスクリプト言語などの機能を統合するための ActiveX Scripting などがある[244]。ActiveX Control の例としては、Adobe Flash[245] や Shockwave[246]、Quicktime[247]、などが挙げられ、Internet Explorer でそれらを再生するためのプラグインとして利用される[248]。

ActiveX Control に関しては、マイクロソフト社側とサン社側の見解にはかなりの相違がある。マイクロソフト社側は「ActiveX Control は Java の全てのメリットを保有すると同時に、Windows の資産である OCX（OLE ベースのコントロール）を活用できる。ActiveX Control はクロスプラットフォームである。」と主張する[249]。これに対し、サン社側の主張は「ActiveX Control は（Windows）OS に対する依存が強く、オープンではない。」ということである[250]。このように、どちらのプラットフォームがよりオープンであるか、クローズドであるかを判断することは互いの主張が真っ向から対立している。サン社は Java を OS に依存しない点をオープンと解釈し、Windows を必須とした ActiveX Control はクローズドであると主張する。マイクロソフト社側は、Windows を必須とするものの、その上では Java を含む様々なアプリケーションを走らせられる、よってオープンであるとの主張である。

ActiveX Control はセキュリティに問題をもつといわれている[251]。そのことが普及を伸ばす事のできなかった理由と指摘する声もある。ActiveX Control は Web ページの表示に変化をもたらし、インタラクティブ性を提供することでウェブサイトを閲覧する楽しさや利便性を向上させるメリットがある。しかし、Windows Vista より以前のバージョンでは ActiveX Control の動作に制限が掛けられていないために、セキュリティ上しばしば問題となった[252]。例えば、ActiveX Control を用いれば現在ログオンしているユーザーが、アクセスできるコンピュータ内のファイル全てに自由にアク

---

244 出所：http://support.microsoft.com/kb/879760/ja 2013/9/23
245 アドビシステムズ社が開発している動画やゲームなどを扱うための規格。
246 マクロメディア社（現在はアドビシステムズ社が買収）が開発した、音楽や動画といったマルチメディアのデータを再生するためのプラグイン。
247 アップル社が開発するマルチメディア技術。音楽、動画、画像、テキストデータなどを取り扱うことができる。
248 出所：http://msdn.microsoft.com/ja-jp/library/hh537942(v=office.14).aspx 2013/9/23
249 末松・ベネット（1996）, p. 119 参照。
250 同上。
251 出所：http://secure.blog.ocn.ne.jp/column/2010/11/49activex_5bdf.html 2013/9/23
252 出所：http://www.ipa.go.jp/security/ciadr/vul/20090707-ms-activex.html 2013/9/23

セス可能となる。これにより、マルウェア[253]として動作する ActiveX Control が不正にユーザーのファイルにアクセスし、情報を盗み取るリスクが生じる。実際 ActiveX Control のセキュリティホールを攻撃する不正なデータを受信することで、被害を受ける可能性が多数指摘されている[254]。

ActiveX Control は、例えばマイクロソフト社の Windows Update にも使われている[255]。それにより、ユーザー側のマシンを調査し、それに対する最適なアップグレードを提供する。便利な機能ではあるが多くのセキュリティホールが見つかっており、悪用されると、パソコンのハードディスクからファイルを削除したり、フォーマットしたり、悪質なプログラムを実行させる事が可能となる[256]。

8） Java における成果の限定性

Java による階層介入戦略はその成果に限定性をもつ。Java の事例では、マイクロソフト社が独自の JVM ならびにそれに対応する Java アプリケーションをつくり、Java の本来の思想である WORA の能力を限定的なものにした経緯がある。これにより Java はネットワーク効果の規模が限定され、特定 OS 対応のアプリケーションと同じ位置付けとなってしまう。具体的には、サン社は 1996 年「100% Pure Java 認定プログラム」を発表し、認定制度をスタートさせていたが、マイクロソフト社は独自に改良を加えた Java を 1997 年 9 月末にリリースしたブラウザーの「インターネット・エクスプローラ(IE) 4.0」に搭載する[257]。これによって、サン社が提唱する 100% Pure Java で書いたソフトが IE4.0 上で走らなくなるという事態が起こった。末松(2002)では、1999 年 11 月の連邦地裁の事実認定の記述より、「マイクロソフト社は他のプラットフォームとは非互換の Windows 用 Java を開発し、それを普及させることによって Java のポータビリティ（移植性）を妨害した。（中略）標準のほんのわずか一部分を自社仕様に変更することにより、それを使用するユーザーには、標準に準拠し

---

253 不正かつ有害な動作を行う意図で作成された悪意のあるソフトウェアや悪質なコードの総称。
254 出所：http://www.ipa.go.jp/security/ciadr/vul/20090707-ms-activex.html 2013/9/23
255 出所：http://www.microsoft.com/ja-jp/security/resources/activex-whatis.aspx 2013/9/23
256 出所：http://technet.microsoft.com/ja-jp/security/bulletin/ms13-090 2013/9/23
257 松下・臼井（1998）, p.133 参照。

ているような誤解をあたえつつ、結局は自社製品しか使えないようにする、つまり自社内に囲い込むことを目的とする（後略）。」を紹介している[258]。

サン社は開発パートナーとのWORAの約束(コミットメント)を守れないとして1997年10月、Javaのライセンス契約違反でマイクロソフト社を米国連邦地裁に提訴する[259]。その後和解に合意するが、マイクロソフト社はその後リリースするOSではJVMを手動によるダウンロードが必要な状態に変えることとなる。

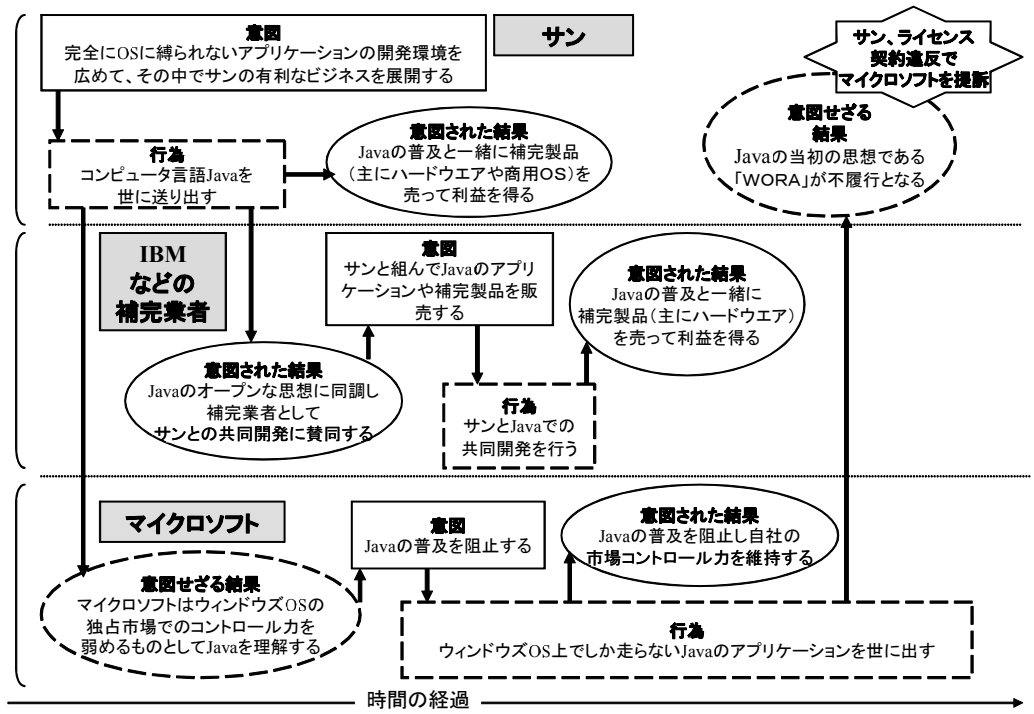
以下にサン社とIBM社などの補完業者、ならびにマイクロソフト社のJavaにおける意図・行為・結果を図に簡単にまとめる（図７－４）。

図７－４では、サン社とIBM社などの補完業者、ならびにマイクロソフト社の３つのプレイヤー間で、サン社によるJavaの発表の意図と行為が、意図された結果ならびに意図せざる結果をもたらす経過を示している。ここで注意が必要であるのは、マイクロソフト社もサン社にとっては補完業者のひとつであるにも関わらず、他のIBM社などの補完業者とは異なる解釈をJavaに対しておこなったということである。ネットワーク効果の観点で考える場合、補完製品であるJavaはマイクロソフト社にとって、ユーザーの便益を増加させる好ましいものとして受け止められるはずである。しかし、マイクロソフト社はJavaを「Windowsの支配力を無力化させる脅威」と受け止めた(詳しくは後述のJava裁判での反対尋問でのマイクロソフト側の内容を参照)。結果として、Javaの普及を阻止するため、WORAの機能を不履行にし、ネットワーク効果の及ぶ範囲を限定させてしまった。これにより、JavaはWindows上で走るひとつのクローズドなアプリケーションの位置付けとならざるを得ない状況となった。

図７－４：Javaに関するサン社、IBM社などの補完業者、マイクロソフト社の意図・行為・結果

をそれぞれ示す。

258 末松（2002），p.75参照。
259 出所：http://itpro.nikkeibp.co.jp/free/ITPro/USNEWS/20020311/1/ 2012/11/03



9）マイクロソフト社との Java 裁判と顛末

マイクロソフト社は 1992 年 2 月の RMI[260]の発表に触発されて、Java の普及を妨害する。Web ブラウザーの IE(Internet Explorer)4.0 から RMI と JNI(Java Native Method Interface）を意図的にはずす[261]。

1997 年 10 月、サン社がマイクロソフト社を Java に関するライセンス侵害で訴える[262]。マイクロソフト社が Windows 固有の技術を Java に組み込んできたためである。この問題が法廷に持ち込まれた結果、両社は 2001 年 1 月に和解[263]。2000 年 4 月にワシントン連邦地裁のトーマス・ジャクソン（Thomas P. Jackson）判事が公開した独占禁止法の事実断定において詳しく分析されている。命令書の中で「Java の脅威との闘い」と項目だてされた部分では、「マイクロソフトはまた、自身の Java 開発ツールを、開発者がパフォーマンスよりポータビリティを選択しようとしているにも関わらず、はからずも Windows 上でしか走らないアプリケーションを記述するように故意

260 Remote Method Invocation の略。ふたつの別のマシン上で動作する Java プログラムの一方のオブジェクトのメソッドを他方のプログラムから呼び出す機能を実現するための仕組み。
261 岩山（2001），p.195 参照。
262 同上。
263 出所：http://www.atmarkit.co.jp/news/200101/25/msun.html　2013/9/23

に設計した。（中略）最後にマイクロソフトは Windows に関して同社に技術情報と認定を依存している ISV（独立ソフトウェアベンダー）にサン社のバージョンに準拠したものでなく Windows の JVM を使用し配布するように仕向けた。これらの行動はメリットに基づいた競争とはいえず、消費者の利益になるものではなかった。（後略）」と結論づけている[264]。

結果として、マイクロソフト社は古いバージョンの JVM のみを利用することで合意している。これに伴いマイクロソフト社は、2001 年早々に Internet Explorer（IE）6 ベータ版から JVM を削除し、Windows XP に Java を標準搭載しないと表明する[265]。Java が必要な場合、ユーザーは Windows Update サイトから手動でダウンロードする必要がある。

この措置に対し 2001 年 8 月初め、サン社は New York Times や Wall Street Journal などの大手新聞にフルページの広告を掲載し、「消費者は、サン社の Java 技術が Windows XP に搭載されることを望んでいる」とアピールした[266]。同時に同社は、主要な PC メーカーに対し、Windows XP プリインストール・マシンへ JVM をインストールするよう要請した。「ユーザーの使い勝手を向上させる目的で、Java アプレットは 600 万以上の Web ページで利用されている。（例えば、）社内システムに対するセキュリティを確保する手段や Web ベースのアプリケーションを容易にアップグレードする手段、Web アーキテクトがサービスを運用する手段としても使われている。マイクロソフト社には最新の Java 2 プラットフォームをライセンスするという選択肢もあった。しかし、それはなされなかった。」と、サン社はこの件に対する報告書で述べている[267]。マイクロソフト社の措置に対して、サン社はその不便さをユーザーに向けてアピールするかたちの行動に出たのである。

これに対し、マイクロソフト社の広報担当者は、「皮肉なものだ。我々は、Windows の中に Java を入れることをやめさせようとしたサン社と、3 年間も法廷で争った。その同じ会社が、今度は Windows XP に Java が標準搭載されていないと不平を言っている。」と皮肉をこぼしている[268]。しかし、この広報担当者の表現は正確ではなく、

---

264 岩山（2001）, p. 196 参照。
265 出所：http://itpro.nikkeibp.co.jp/free/NT/NEWS/20010816/1/ 2013/9/21
266 同上。
267 同上。
268 同上。

Windows の中に入れさせないようにしたものはマイクロソフト独自 Java であり、Windows XP に標準搭載させようとしているものは 100% Pure Java である[269]。

Windows XP から Java を削除することで生じた一連の争いは、マイクロソフト社が IE や Windows を通して、Java にとっての最大の流通業者であるという事実から生じたことであるといえる。補完的な関係でありながら競合するという構造は、利害における様々な軋轢を生む。そこには自らのプラットフォーム製品の普及は、対立しながらも補完し合う相手にその根っこを握られているというジレンマから生じている。

これらの係争や両社の CEO の批判合戦において、当時の IT 業界では「サン社とマイクロソフト社は天敵」と評する向きもあったが、当時 Java ソフトの統括者であったアラン・バラッツ（Allan Baratz）は「マイクロソフトの独自 Java を黙認したのではサン社が Java の利用者に対しておこなった約束を守れない。それを避けるために敢えてマイクロソフトに対する訴訟に踏み切った。」と述べている[270]。

１０）サン社とマイクロソフト社との和解

2004 年 4 月、マイクロソフト社とサン社が包括提携をし、長年にわたる全ての係争に対する和解に合意する[271]。両社の製品間の連携を強化するための広範な技術提携、および両社間の全ての係争において全面的に和解することで合意したとの発表をおこなう。両社はまた、特許を含む他の案件での協力関係にも合意する。

サン社ならびにマイクロソフト社のコメントは以下である（以下、該当記載部分の抜粋）[272]。

サン社 CEO（最高経営責任者）のマクニーリは、「今回の合意により、サン社とマイクロソフト社は、双方が顧客の選択肢を維持しながら相互に協力していくという新たな関係の構築に向けて大きく踏み出すことになります。この合意は、サン社とマイクロソフト社双方の顧客に大きな利益をもたらすでしょう。すなわち、新たな製品開発が促進されることで、複数ベンダーのサーバー製品を結びつけ、異機種混在環境で

---

269 同上。

270 松下・臼井（1998）, p. 133 参照。

271 出所：
http://www.atmarkit.co.jp/fwin2k/insiderseye/20040526sunmicrosoft/sunmicrosoft_01.html
2013/9/21

272 2004 年 4 月 2 日に米国で発表されたリリースの抄訳をベースにしたマイクロソフト社のリリース http://www.microsoft.com/japan/presspass/detail.aspx?newsid=1887 2013/9/21 より抜粋（一部筆者修正）。

のシームレスなコンピューティングを実現しようとする顧客に、大きな選択肢を新たに提供することになります。我々は、今回の合意が、サン社とマイクロソフト社との将来の協力関係の枠組みを固める好機となるものと期待しています。」と述べている。

マイクロソフト社 CEO のスティーブ バルマー（Steve Ballmer）は、「両社は今後とも厳しい競争関係を継続していきますが、今回の合意は、両社の協力関係の基礎を新たに構築することによって、双方の顧客にさらなる利益をもたらします。今回の合意は、最先端の研究開発および知的財産の保護が我々の業界の成長と成功には欠かせぬものであるという認識に基づいていて締結されたものです。この合意は、サン社およびマイクロソフト社双方にとって大きな前進ではありますが、本当の意味の勝者は、我々の製品と革新技術を必要としている顧客であり開発者にほかなりません。」と述べている。

本合意より、マイクロソフト社はサン社に対して、独占禁止法関連問題の和解金として 7 億ドル、特許関連問題の和解金として 9 億ドルを支払う。両社はさらに、テクノロジーの相互利用についても合意する。これに基づき、マイクロソフト社はサン社のテクノロジーを使用するための前払い金として 3 億 5 千万ドルをサン社に支払い、サン社はマイクロソフト社のテクノロジーを同社のサーバー製品に組み込む場合にマイクロソフト社に対して使用料を支払うこととなる。

署名された合意は、以下の内容が含まれている[273]。

①技術連携（The Technical Collaboration Agreement）：技術連携契約により、両社のサーバー関連技術を相互に利用し合うことが可能となる。本契約により、両社は、双方の製品間の連携機能を強化した新しいサーバー製品を開発することができる。第一段階として、Windows サーバーと Windows クライアントを中心にして進められるが、最終的には、電子メールやデータベース ソフトウェアを含む他の重要な分野も含まれる。例えば、大規模なコンピューティング環境には欠かせない、ユーザーの ID、認証、許可を管理するためのソフトウェアなどがある。この合意の結果、サン社とマイクロソフト社のエンジニアは、マイクロソフト社の Active Directory とサン社の Java System Identity

273 同上。

Server 間のユーザーID 情報の共有が簡単にできるように協力し、複雑さが軽減されたよりセキュアなコンピューティング環境の実現を目指すこととなる。

②Java と.NET における将来的な提携：両社は、Java と.NET 間の技術連携強化についても協力していくことに合意した。

１１）サン社のその後の合併と Java の行方

サン社は 2000 年 9 月には、インテル系の CPU と Linux の組合せのサーバーを販売していたコバルトシステム社を買収[274]。インテル系の CPU と Linux を組み合わせたサーバーが 2001 年からサン社より販売された。2005 年には StorageTek(STK)社、2008 年 1 月には MySQL 社の買収を発表した[275]。

1995 年の発表以来、Java は順調にその開発者とダウンロード数を伸ばしていくが、インテル・アーキテクチャの性能の向上による安価なチップ上の Windows や Linux の台頭により企業としての存続が脅かされ、サン社はオラクル社に買収されることとなる。Java ならびに Solaris はオラクル社のブランドで引き続き存続するが、Java の普及はサン社の直接の収益向上に十分働くことはなかった。

2003 年頃から収益や株価がいっそう低迷するなか、サン社はなぜ Java から直接収益を得ることをしないのかという声が聞かれるようになる。2004 年 2 月 Java Technology Conference 2004 でのインタビュで、エグゼクティブバイスプレジデント（後に CEO）のジョナサン・シュワルツ（Jonathan Schwartz）は Java の将来性について懐疑的な声もあることを自ら明かした[276]。「ウォールストリートではサン社が Java に力を入れている訳を理解してもらえず、事業の売却や中止を勧める声もある」と述べる。しかし「サン社には確固たるビジネスモデルがある」[277]と断言し、これからも Java の研究開発を継続することを表明している。加えて、シュワルツは現在 200 種類以上の携帯電話に Java が載っており、日本の NTT ドコモや Vodafone をはじめ世界中で Java コンテンツの配信が始まっている。コンシューマ・ユーザーだけでも Java 携帯電話に年間 30 億ドルを使っており、今後も増えていくという巨大な市場である。

---

274 出所：http://www.atmarkit.co.jp/news/200009/21/sun.html 2013/9/21

275 出所：http://japan.cnet.com/news/biz/20368251/ 2013/9/21

276 出所：http://cloud.watch.impress.co.jp/epw/cda/topic/2004/02/19/1426.html 2013/9/21

277 同上。

PC においても Java は広く活用されており、全体の 6 割で Java が載り月間 700 万のクライアント・プログラムがダウンロードされていることを説明した[278]。このような Java の実績と将来性を展望されたにも関わらず、その後サン社自体の収益や株価が大きく好転することはなかった。

2009 年 4 月、オラクル社はサン社を約 74 億ドルで買収することについて、サン社と合意したと発表した[279]。後に、サン社株主の合意などを経て正式に決定された。サン社株式を、1 株当たり 9.5 ドルで買収。「最高のエンタープライズソフトウェアとミッションクリティカルなコンピューティングシステムが融合する」[280]とし、顧客側の作業負担なしで両社のシステムを統合するとしている。

オラクル社の企業サイトトップページには、「Oracle Buys Sun」と、サン社のサイトには「ORACLE TO BUY SUN」書かれた大きなバナーが掲載された。サン社をめぐっては、IBM 社や HP 社も買収提案したと報じられていた。HP 社とオラクル社でサン社を 2 分割する案もあったとされる[281]。

オラクル社はサン社を傘下に収めることにより、市場規模 170 億ドルとされる企業向けハイエンド Unix サーバーの分野で HP 社を抜き、IBM 社に次ぐ２位のメーカーになる[282]。オラクル社はまた、サン社がもつ Java と OS の Solaris も手に入れた。

技術の進歩は日進月歩であるが、Java をとりまくネットワーク環境も大きく進歩し、当初の新しかった技術も徐々に時代遅れとなるものも多い。Java はリリース初期には Web ブラウザー上で動く、アプレットという動的な Web ページを実現するアプリケーションとしての使われ方が主流であった。しかし、このアプレットとしての役割は 2000 年以降、徐々に Adobe Flash などに取って代わられた[283]。その後、Java SE といった基本的な開発環境に加え、Java EE といったサーバーサイドのアプリケーション開発に特化した開発環境や、Java ME といった組み込み機器用の開発環境がリリースされた。それにより、今日では主にサーバーサイドの Web アプリケーションや、携

278 同上。
279 出所：http://www.itmedia.co.jp/news/articles/0904/20/news110.html 2013/9/21
280 同上。
281 同上。
282 同上。
283 出所：http://ossforum.jp/node/604 2013/9/23

帯電話等に搭載されている組み込み機器技術など、幅広い分野で利用されている[284]。Java プログラムの実行形態にはコマンドラインで実行されるコンソールプログラム、Java アプレット、Java サーブレットといった種類がある。Java アプレットは Adobe Flash などの普及によりあまり利用されなくなったが、Java サーブレットは Web サーバー上で実行される Java プログラムである。Perl[285]や Ruby[286]などスクリプトで処理される CGI[287]と比べると、様々なオーバーヘッドが少ないために動作が速いというメリットがあり、現在も多くの開発者の支持を得ている[288]。

１２）オラクル社に引き継がれた Java のサポート

オラクル社は、2010 年のサン社の買収に伴って、Java の所有権と管理責任を引き継いだ。現在では 900 万人を超える開発者が Java を使用して、スマートカードやスマートフォン、企業向けサーバー、さらにはクラウドまで、幅広いプラットフォームを対象にアプリケーションを開発している[289]。Java は 4,500 種類を超えるブランド製品の基盤となっており、企業向けデスクトップ・コンピュータの 97%、ブルーレイ・デバイスの 100%で動作している[290]と同社のウェブで説明している。

Eclipse Project の最高責任者であるマイク・ミリコウビッチ(Mike Millinkovich)は、オラクル社傘下後の Java に関して、「サン社の管理下にあった最後の数年間は、Java への投資が目に見えて減らされていました。」「Java Community Process の実行委員会レベルでも Java 7 の開発が行き詰まっていました。オラクルが管理を引き継いだことで、ようやく息を吹き返しました。」とコメントしている[291]。また、JCP の委員長であるパトリック・カーラン(Patrick Curran)は、「Java コミュニティのすべてのメンバーが新しい JSR の作成状況を把握し、その進展に携わることができるようにするため、新しい JCP 2.8 プロセスでは専門家グループがオープンに活動するこ

---

284 同上。
285 ラリー・ウオール（Larry Wall）が開発したプログラミング言語。
286 まつもとゆきひろによって開発されたオブジェクト指向のスクリプト言語。
287 Web サーバーが、Web ブラウザーからの要求に応じて、プログラムを起動するための仕組み。
288 出所：http://ossforum.jp/node/606　2013/9/23
289 出所：
http://www.oracle.com/jp/corporate/citizenship/introduction/java-in-action-1988938-ja.html?iframe=true&width=800&height=600　2013/9/21
290 同上。
291 同上。

とが求められます。これにより、開発者にわかりやすく、実装または利用する人が迷わない仕様が策定されることを期待しています。」と語っている[292]。

過去 2 年間、オラクル社は Java の管理に関して 3 つの目標に重点を置いてきた。それは、Java テクノロジーの前進、Java 標準策定へのコミュニティ参加の拡充、開発者とエンド･ユーザーを取り巻く Java エコシステムの拡大であると説明している[293]。今後も、サン社同様オラクル社も引き続き Java の思想を継承し、サポートを続けていくことが想定される。

１３）Java と.NET の開発環境と開発者コミュニティの比較

2004 年 4 月、Java と.NET における将来的な提携により、両社は、Java と.NET 間の技術連携強化についても協力していくことに合意した。そのふたつの開発環境の比較を簡略に記す。あくまでもそれぞれの得意な分野を尊重した中立的な立場でのコメントである（以下、該当記載部分の抜粋）[294]。

〇J2EE（Java 2 Enterprise Edition）

Java プラットフォームのベースをなすアプリケーション・サーバー規格が J2EE（Java 2 Enterprise Edition）である。サン社を中心に、Java Community Process の標準化グループによって規格化された。J2EE では、デスクトップ・アプリケーション向けの J2SE（Java 2 Platform Standard Edition）の仕様に加え、分散コンポーネント・モデル（EJB）や Web アプリケーション・サーバー機能（サーブレット、JSP、JSF）、Web サービス対応機能（JAX-M/JAX-RPC）、データ・アクセス機能（JDBC）などといった、いわゆる「サーバーサイド Java」機能が追加されている。

サン社の Sun One Application Server に加え、IBM 社の Web Sphere、BEA 社の Web Logic、オラクル社の Oracle Application Server、フリー・ソフトウェアの JBoss など、さまざまなベンダーから J2EE に対応したアプリケーション・サーバーが提供されている。

---

292 同上。
293 同上。
294 出所：http://www.atmarkit.co.jp/ad/ms/interoperability/interoperability02.html 2013/9/21 より抜粋（一部筆者修正）。

○.NET（Microsoft .NET）

.NET は、マイクロソフト社が提供するソフトウェア・プラットフォームの総称である。ISO（国際標準化機構）や JIS（日本工業規格）で標準化された CLI（Common Language Infrastructure）をベースに開発された.NET Framework ランタイム・ライブラリ上にサーバー、クライアント双方のアプリケーション環境を構築している。このうちサーバー・アプリケーション向けには、Web アプリケーションのユーザー・インターフェイス設計を支援する ASP.NET、Web サービス/データ・アクセスを支援するサービス・コンポーネントの ADO.NET などが提供されている。

.NET と Java は機能や仕様だけでなく、根本的な違いがある。

Java 対応ソフトウェアは、J2EE などサン社がリードして作成した仕様（ドキュメント）に従って、さまざまなソフトウェア・ベンダーやオープンソース・グループが実装をおこなう。ユーザーは、こうして実装されたアプリケーション・サーバー製品などから選択する。Java は様々な分野に精通している。大規模企業システムや携帯電話などのモバイルはもとより、小規模システム、デスクトップ、オープンソース・コミュニティー支援、そしてコンシューマー市場や組み込み製品市場向けまでカバーしており、多くの実績がある[295]。

これに対し、.NET 対応ソフトウェアは、マイクロソフト社が仕様を策定するだけでなく、実装も並行しておこない、実際に Windows 環境で稼動するソフトウェアとしてユーザーに提供される。開発基盤としては後発である.NET であるが、こちらも急速に普及が広まっている。小規模な SOHO レベルから、大規模なエンタープライズ環境に至るまで、さまざまなビジネスの要求に応えられるシステム構築のための要素が網羅されている。その理由としては.NET Framework によるテクノロジーの一貫性、統合性が考えられる。さらに、高度なサービスやフレームワークが提供されていることで、アプリケーションの品質を高い水準で維持できるということも大きい要因である。

当分は、この 2 つのプラットフォームが市場をけん引すると予想される[296]。

---

295 出所：http://www.atmarkit.co.jp/fjava/rensai4/java_dotnet01/04.html 2013/9/21 より抜粋（一部筆者修正）

296 同上。

Java と.NET のそれぞれの開発者コミュニティ活動の面から両者を比較すると以下のようになる（以下、該当記載部分の抜粋）[297]。

〇Java のコミュニティ

Java の開発者コミュニティは、オープンソースであることも手伝って、数多く存在していた。その中でも代表的なものを以下に記す。

- ＊ OpenJDK：JDK をオープンソース化しようとするプロジェクト
- ＊ java.net：技術者から研究者などさまざまな有識者で構成されているコミュニティ
- ＊ GlassFish：フリー、かつオープンソースでエンタープライズ向けのアプリケーション・サーバーやツールの提供、リファレンス実装をしているコミュニティ
- ＊ Java Community Process（JCP）：Java の技術の開発や標準仕様の策定をおこなっている国際的機関
- ＊ Apache Software Foundation：数多くのプロジェクトを保持するオープンソースのソフトウェア製品を展開するコミュニティ

他にも世界中に数多くのコミュニティが存在しており、オープンソースの成果とともにコミュニティは拡大している。一方、.NET のコミュニティに関しては以下である（以下、該当記載部分の抜粋）[298]。

〇.NET のコミュニティ

.NET のコミュニティも充実している。.NET には、マイクロソフト社が提供する「Microsoft Developer Network（MSDN）」というエンジニア向けのサポートサービスがある。この MSDN の中でマイクロソフト社の製品や技術ごとのコミュニティが紹介されており、マイクロソフト社の社員による回答も得られる。

---

297 出所：http://www.atmarkit.co.jp/fjava/rensai4/java_dotnet03/04.html 2013/9/21 より抜粋（一部筆者修正）

298 同上。

日本における両社のコミュニティの状況としては、どちらの技術も言語仕様、ドキュメント、コミュニティともに日本語のサポートはかなり充実している。日本でのコミュニティも数多くある。その中でも Java では「日本 Java ユーザー・グループ（JJUG）」「java-ja」、.NET では「Visual Studio User Group（VSUG）」、そして双方をカバーする「The Seasar Project」などが有名である[299]。

（２） Java 事例採用の理由

階層介入戦略の事例として Java を取り上げる理由は、1995 年に誕生した Java が、それまでの C 言語などとは違う完全なクロスプラットフォーム指向（プラットフォーム OS 非依存）を標榜し、誕生から僅か 10 年で 450 万人以上[300]の開発プログラマーを世界に育てた実績と、PC を初めとして携帯電話やスマートカードなど様々なデバイスで利用され続けている[301]点が階層介入事例の代表的な存在と認められるためである。

実際に Java が投入されたのは 1995 年からであったが、90 年代後半にサン社は IT バブルに乗じて Unix 市場で IBM 社や HP 社と比肩して大きなシェアを獲得していた。しかし、コンシューマー市場で大きなシェアを獲得しつつあったマイクロソフト社は、Unix が主流を占めるエンタープライズ市場でのシェア獲得を目論んで[302]おり、サン社にとっては脅威であった。サン社の Java による階層介入戦略は、シェアを伸ばし始めたマイクロソフト社の Windows に対する攻撃と、既存の Unix 市場を Windows から守るふたつの目的を持ち、また新たに Java のプラットフォーム製品でのエコシステムを形成させるための切り札として位置付けられていたと考えられる。

この介入階層に隣接する OS を提供するマイクロソフト社は Java の投入をどのように認識していたかについては、米国マイクロソフト社のホームページに掲載されている Java の訴訟でのコメント内容の抄訳[303]から推察できる。

---

299 同上。

300 JavaOne 東京 2005 でのサン社発表による。

301 世界中の 8 億 5000 万台を超える個人用コンピュータや、世界中の何十億台ものデバイス（モバイルデバイスや TV デバイスなど）で動作している。出所：
http://www.java.com/ja/download/faq/whatis_java.xml 2013/09/23

302 サウスウイック（2000）, p. 212 参照。

303 出所：http://www.microsoft.com/japan/presspass/trial/120298_doj_java.htm
2012/11/03

「（抄訳の抜粋）またバート（Tom Burt：マイクロソフト社の弁護士）のゴスリング博士（サン社の幹部の１人で Java の創案者）に対する反対尋問の中で、1995 年にサン社がインテル社のチップとマイクロソフト社のオペレーティング・システムを無力にするような別のプラットフォームを Java を用いて開発して、マイクロソフト社とインテル社に対し激しい攻撃をかける計画をしていたことが明らかになりました。（中略）その中でシュミット氏（サン社の元最高技術責任者）は、Java 言語をベースとした新しいプラットフォームをつくるという攻撃的な計画を示しています。バートはまた、新しい完全なプラットフォームのベースに Java 言語を用いるというサン社の計画を具体的に示す他の証拠も紹介しました。（後略）」。この内容から、マイクロソフト社が Java に対して相応の脅威を感じていたことが理解できる。

（３） Java における操作項目の事例整理

次に、Java においてアクセス可能ユーザー数の増加、マルチホーミングコストの低減、隣接対象プラットフォーム製品の多数選定、持続的収益確保モデルの遂行の４つの項目の観点から事例を整理する。

１）アクセス可能ユーザー数の増加

Java は Unix や Linux や Windows の間の橋渡しをおこなった。これにより、下位階層の複数のプラットフォーム製品それぞれがもっているアクセス可能ユーザー、ならびにサイド内ネットワーク効果を奪い取ることとなった。

Rohlfs（2001)は「Java はバンドワゴンの点でオペレーティング・システムを相互連結させるという長所を持ち、複数のオペレーティング・システムの利用者は Java で書かれたプログラムから完全なバンドワゴンの便益を受け取ることができる。（後略）」と指摘している[304]。

この階層介入により、新たな階層として Java の階層[305]が形成された。利用者は Java が動作するチップと OS であればどれを使おうと同じ便益を受けることができる。そ

---

304 Rohlfs（2001），（邦訳）p. 163 参照。

305 正確には新たな介入階層は JVM（Java 仮想機械）であるが、JVM は Java アプリケーション（含むアプレット）の起動時のみ機能を発揮するため、本論文では JVM 階層とは呼ばず Java 階層と表現する。

のため、Java を開発言語としてアプリケーションを開発しようという開発者のインセンティブが高まる[306]こととなる。これにより Unix や Windows など単一の OS 上でしか走らないアプリケーション開発者の多くが、Java での開発に興味をもつことになり、開発環境における主流となるプラットフォーム製品の階層が、それまでの特定 OS 階層から Java の階層へと移行するインセンティブが生まれた。

サン社が Java のソースプログラムを公開して、3 年間で 100 万人以上[307]のプログラマーの賛同を得ることができたのは、プログラマーのエコシステムを機能させたからだと言われている。会社という事業体の枠を超えてプログラマーのエコシステムを形成し、他社の開発成果や導入事例を参照しながら次期システムの在り方を考えた。エンジニアは優れたアイデアを事業化しようと企業が動くときには安心して仕事に邁進する。そうでなくなると存在意義を失い転職してしまう。1996 年に 30 人でスタートした Java は 1999 年に 1500 人のスタッフを擁する開発部隊[308]となった。

2）マルチホーミングコストの低減

この点において、Java は無償[309]で提供し、誰もがインターネットでダウンロードできる仕組みをとった。これにより、ユーザーのマルチホーミングコストを低く抑え、普及を促していくことが可能であった。急速に普及を果している Java に対し、マイクロソフト社は「ライセンス料は将来、普及した後で大幅に吊り上げられるかもしれない」と批判した[310]。また、将来的に課金されるのではないかというユーザーや開発者の不安に対し、その懸念を払拭している。1998 年 3 月サン社のチーフ・サイエンティストであったジョン・ゲイジ（John Gage）は、5 年後あるいは 7 年後のサン社のビジネスモデルはどのようなものになっているのか、という質問に対し「（サン社が）Java のライセンスで儲けることを考えないのは今と同じだ。」と答えている[311]。

---

306 サウスウイック（2000）, p. 196 参照。
307 岩山（2001）, p. 77 参照。
308 岩山（2001）, p. 79 参照。
309 ちなみに Java のソースコードの商用ライセンスの公表価格は、WIRED 誌の記事によると前金で 125,000 ドル（日本円で約 1,000 万円）プラスコピー1 本につき 2 ドル と言われている。松下・臼井（1998）, p. 100 参照。
310 末松（2002）, p. 70 参照。
311 松下・臼井（1998）, p. 140 参照。

３）隣接対象プラットフォーム製品の多数選定

介入先の対象とする隣接階層に関して、Java は広く普及した（もしくは普及しそうな）階層内のプラットフォーム製品をターゲットに選定して介入した。具体的には、既に広く普及している Windows や Linux や Unix などの隣接階層内のプラットフォーム製品をとらえ、それらを梃子に介入した。Java の場合は、アプリケーション階層と OS 階層間に下位階層に対してオープンなインターフェイスをもつ製品として介入している。OS 階層で大きなプレゼンスを有する Windows をターゲットとして選定したことは、補完関係にあるプラットフォーム製品を利用して短期間のうちに拡大を図ることを可能にした。そして結果として Windows の市場支配力を削ぐことになった。この点について末松・ベネット（1996）は「Java は OS に依存しないプラットフォームであり、その普及は、当然のことながら、OS の市場を独占している Windows の相対的パワーを弱めることになる。しかも Java は、今後大きく開かれようとしているコンピュータ家電に最適なプラットフォームであるから、その市場での普及を許せば、マイクロソフト社の発展性は断たれてしまうと言っても過言ではないだろう。（後略）」と説明している[312]。

加えて、オープンなインターフェイスを堅持することで、多数のプラットフォーム製品と隣接する状況を維持した。互換性を保証するためにサン社は 140 社以上のパートナー企業とライセンス契約を結び、各社にはソフト出荷時に１万件以上の診断テストをおこなうことを約束してもらっている[313]。また、マイクロソフト社の WORA を阻害する動きには断固として対抗してきた。

４）持続的収益確保モデルの遂行

サン社は Java を無償で提供し、Java 自体からの収益はほとんど期待せず、その宣伝効果によるブランド力の向上や、賛同してくれる企業数のアピールによるマーケティング効果などによって、商用 OS やハードウェアなど補完製品の売上から収益を得た。それゆえ Java 普及の成功とは裏腹に、その開発と普及活動に相当の投資をしてきたにもかかわらず、サン社自体はその恩恵を間接的な形でしか享受できなかった。

---

312 末松・ベネット（1996）, p. 118 参照。
313 松下・臼井（1998）, p. 133 参照。

Java はその後、JCP [314]にその運営を任せることとなるが、サン社は持続的な収益確保の構築と実現に大きな課題を残すこととなった。

事実、Java は 1995 年のデビュー当時から明確な事業化の成算があったわけではない。松下・臼井（1998）によると、シュミットは Java をインターネットでただでばら撒くという意思決定をサン社の幹部がおこなった時も「周囲の雰囲気は『お前はわが社のテクノロジーをマイクロソフト社をはじめとする競合会社に渡すつもりか。どうやって儲けを出そうと言うのだ。』当時、私はこれに答えられなかった。どうにかなるとは思ったけれど。嘘をつこうかとさえ思った。」とコメントしている[315]。

（4） Java 介入による階層間関係とポジションの変化

階層介入型プラットフォーム製品は、既存のレイヤースタックの階層間に後から「介入」することにより階層間関係やプラットフォーム製品間関係に影響を及ぼし、変化させてしまう可能性をもつ（図７－５）。

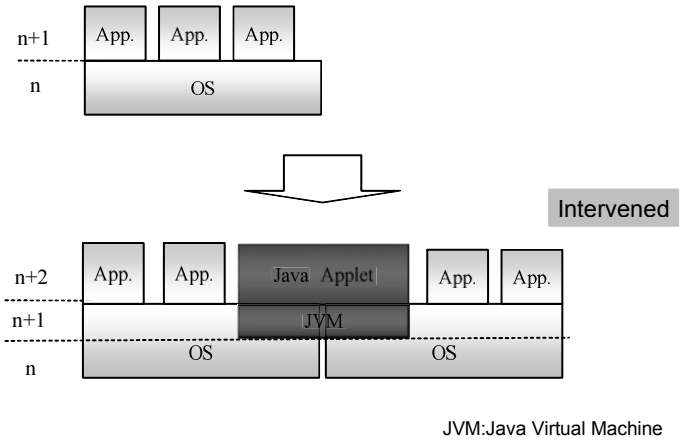
図７－５は、Java が介入する前と後での階層構造を表わしている。介入前はｎレベル階層に OS、ｎ＋１レベル階層にアプリケーションがある。介入後は、ｎ＋１レベル階層に JVM が介入し、ｎ＋２レベル階層に Java Applet がダウンロードされる。介入階層は JVM であり、JVM は Java Applet がダウンロードされる時にのみ機能する。JVM は事前に OS にインストールされていることが条件となる。介入する階層は JVM であるが、本論文では Java Applet とセットで Java 階層と表記する。

---

314 サン社は Java の独立性と公平性を保つために 1998 年 Java コミュニティプロセス（JCP）を導入した。1995 年に Java 発表当時はサン社がもっていたシステム仕様などに関する管理を JCP に移管した。JCP においては、Java 言語を拡張する必要を感じる人々は誰でも、JSR（Java Specification Request）を申請することができ、その必要性が認められると有識者からなるエキスパート・グループの編成が呼びかけられる。JCP は分野ごとの専門家集団を広げてゆくことになった。詳細は本章の第 2 節の(2)の 4）を参照。

315 松下・臼井（1998）, p. 165 参照。

図７－５：Java による階層介入図



具体的に、Java の投入によってそれらの隣接階層にどのような変化をもたらしたのかに関して、階層間の対応関係、各階層のプラットフォーム製品戦略、ソフトウェア・レイヤースタック内でのドミナント状況を投入以前と投入後を比較しながら以下にまとめる（表７－３）。

表７－３：Java における介入前後の変化

階層	隣接下位階層	隣接上位階層	介入階層（中位）
該当製品	サーバーOS（Unix・Linux・Windows 他）	業務用アプリケーション	JVM（Java 仮想機械）
参入時期	先発	後発	後発

【階層介入前】

上下隣接階層間のプラットフォーム製品対応関係	ひとつの OS に対し複数のアプリケーションを走らせることができる。OS とアプリケーションは１対多の対応である。	ひとつの OS に対し複数のアプリケーションを走らせることができる。OS とアプリケーションは１対多の対応である。	
各階層のプラットフォーム製品	隣接することで、階層間ネットワー	隣接することで、階層間ネットワ	

戦略	ク効果を発揮し、プラットフォーム製品戦略を遂行する。OS 提供者はより多くの普及が進んだ（もしくは進む見込みのある）アプリケーションと手を組むことが肝要となる。	ーク効果を発揮し、プラットフォーム製品戦略を遂行する。	
ソフトウェア・レイヤースタック内でのドミナント状況	特定の OS がドミナント・プラットフォーム製品になる可能性がある。	数多くの特定 OS 専用のアプリケーションが濫立しアプリケーションが OS を凌駕し、ドミナント・プラットフォーム製品になる可能性はかなり低い。	

【階層介入後】

上下隣接階層間のプラットフォーム製品対応関係	ひとつの OS に対し Java アプリケーションや Java Applet を含む複数アプリケーションが走る。OS とアプリケーションは1対多の対応であり、階層介入前と変わらない。また、OS と JVM は1対1の対応である。	複数アプリケーションが OS 上を走る。OS とアプリケーションは1対多の対応であり、階層介入前と変わらない。	ひとつの JVM に対してひとつの OS が対応する。JVM と OS は1対1の対応である。JVM により OS に依存せず、その上で様々な Java アプリケーションや Java Applet が走るため、Java アプリケーションや Java Applet と JVM は多対1の対応となる。
各階層のプラットフォーム製品戦略	理論的には Java の普及により OS の普及も少なからず促進される可能性はあるが、自社 OS のみを普及させることはできない。	Java アプリケーションや Java Applet の増加により既存の非 Java 業務用アプリの需要は相対的に減少する可能性がある。	Java アプリケーションや Java Applet が正常に走るように、インストールド・ベースの OS に JVM をインストールする。 Java 開発者を増加させる。

ソフトウェア・レイヤースタック内でのドミナント状況	コモディティ化に陥るリスクが高まる。	数多くの特定 OS 専用のアプリケーションが濫立する。アプリケーションが OS を凌駕するようなドミナント・プラットフォーム製品になる可能性はかなり低い。	JVM 上にダウンロードされるアプリケーションや Java Applet により、ドミナント・プラットフォーム製品になる可能性がある。

上記の表により、以下がわかる。

- ✓ JVM（Java 仮想機械）階層の介入は、階層介入以前の上位階層と下位階層の階層間での対応を変化させる。
- ✓ 各階層のプラットフォーム製品戦略は、２階層間もしくは３階層間で階層間ネットワーク効果を利用し拡販を続けるという点では、階層介入前後で変化はない。
- ✓ レイヤースタック内でのポジションは Java 階層の介入によって、下位階層の OS はコモディティ化に陥るリスクが高まるのに対して、Java 階層自体はドミナント・プラットフォーム製品になる可能性が高まる。

（５） 小括

本節では、Java 事例の考察として、Java の概要、Java の事例選択の理由、Java の操作項目による事例の整理、階層介入による階層間関係とポジションの変化について論じた。Java の概要では、Java デビューの経緯と歴史、サン社の生い立ち、Java のプログラム言語としての設計上の特性、JCP（Java Community Process）と SDC(Sun Developers Connection)、サン社の経営方針と Java 戦略、サン社の経営とマイクロソフトの経営、Java における成果の限定性、マイクロソフト社の対抗製品 ActiveX Control、マイクロソフト社との Java 裁判と顛末、サン社とマイクロソフト社との和解、サン社のその後の合併と Java の行方、Java と.NET の開発環境と開発者コミュニティの比較、オラクル社に引き継がれた Java のサポートなど多面的に事実内容の説明をおこなった。その上で、事例として選択した理由について論じ、前述したプラットフォーム製品提供者の４つの操作項目の観点で確認をおこなった。加えて、階層介入前と後に関して、階層間関係とポジションの変化について考察をおこなった。

第３節　VMware 事例の考察

次に階層介入の事例として VMware を取り上げる。ヴイエムウェア社は仮想化ソフトを始めとして、多様な仮想マシン環境構築用ソフトウェアやサービスを提供している。通常「VMware」と表記する場合、ヴイエムウェア社の社名とヴイエムウェア社が提供する VMware ファミリー製品を指すが、本論文では社名はヴイエムウェア社とし、VMware と表記する場合は、本論文のなかで取り上げる VMware ESX や VMware ESXi を指すものとする。

ヴイエムウェア社（VMware, Inc）は、コンピュータの仮想化用ソフトウェアを製造・販売する、米カリフォルニア州パロアルト市に本拠を置く会社である。ヴイエムウェア社は 1998 年に設立され、2004 年 1 月に EMC コーポレーションによって買収された[316]。2007 年 8 月にはニューヨーク証券取引所で株式公開した[317]。また 2003 年には日本法人であるヴイエムウェア株式会社（VMware K.K.）が設立され、現在（2013 年 9 月 23 日）の所在地は東京都港区となっている[318]。

（１）　VMware とは

VMware はヴイエムウェア社が提供する仮想マシン構築のためのソフトウェアで、主力製品として VMware ESX、ESXi がある[319]。

ヴイエムウェア社は、x86 ベースのアーキテクチャにおいて、仮想化の仕組みを初めて実現した企業である。ヴイエムウェア社は、1999 年に x86 ベースのコンピュータ上に仮想マシンを構築する VMware Workstation を発表し、PC の仮想化を実現した[320]。

---

316 Yoffie, Hagiu & Slind（2009）, p. 5-6 参照。
317 同上。
318 出所：http://www.vmware.com/jp/company/　2013/9/23
319 製品概要情報の出所：http://www.vmware.com/jp/products/esxi-and-esx/overview
http://www.vmware.com/jp/products/esxi-and-esx/compare.html
http://www.atmarkit.co.jp/fwin2k/operation/vmcomp01/vmcomp01_02.html
http://thinkit.co.jp/article/127/1　すべて 2013/9/23　製品の機能ならびにアーキテクチャの記述に関しては、主として上記 URL からの引用による。
320 出所：http://www.atmarkit.co.jp/fwin2k/operation/vmcomp01/vmcomp01_02.html
2013/9/23

この技術をサーバーに発展させたものが、2001年にリリースされたVMware GSX Serverおよび2002年に発表されたVMware ESXである[321]。

VMware ESXは、現在仮想化ソフトウェア製品パッケージであるVMware vSphereの一部として有償で販売されており、サービス・コンソール[322]などの機能を制限したVMware ESXiは無償で入手して利用することができる。仮想化の機能自体は両者で違いはない[323]。

VMware ESXはハードウェア上で直接動作し、仮想的に構築されたコンピュータ上で様々な種類のOS(ゲストOS)を動作させることができる。ゲストOSに仮想化のための修正を必要としない「完全仮想化」型のハイパーバイザーで、通常のOS製品を仮想マシンにそのままインストールして利用することができる。

VMware ESXのアーキテクチャの中心は、VMkernelと呼ばれる専用のハイパーバイザーである。VMkernelは、プロセッサ、メモリ、ストレージおよびネットワークといったハードウェア資源を抽象化して管理し、要求に応じた資源の割り振りをおこなう[324]。VMware ESXは、完全仮想化を実現するハイパーバイザー方式の仮想化ソフトウェアであるため、WindowsやLinuxといったような一般的なホストOSを必要としない。ハイパーバイザーが直接x86ハードウェア上で動作する[325]。VMkernelでは64bitのゲストOSを稼働させるときのみIntel VT[326]やAMD-V[327]などのプロセッサの仮想化支援機能が必要となる。32bitのゲストOSだけを稼働させている環境ではこれらのプロセッサ側の支援機能の対応は必要としていない[328]。VMware ESXには、VMkernelのほかにLinuxベースのサービス・コンソールと呼ばれる管理用のOSがある。なお、無償で提供されているVMware ESXiには、サービス・コンソールが含まれていないため、

---

321 同上。

322 ヴイエムウェア社がLinuxを利用して、開発した管理用のOS環境。VMkernelへの管理インターフェイスを提供するサービス。VMkernelとは別物。

323 出所：http://www.atmarkit.co.jp/fwin2k/operation/vmcomp01/vmcomp01_02.html 2013/9/23

324 同上。

325 同上。

326 インテル社のマイクロプロセッサに採用されているシステム仮想化技術。1台のコンピュータで複数のOSを並行に動作させることを可能にする。

327 AMD社が製造販売するCPUに搭載された、ひとつのCPUの上で複数のOSを動作させるためのハードウェアレベルでの仮想化技術。切り替えなどにかかるオーバーヘッドが少ない利点がある。

328 出所：http://www.atmarkit.co.jp/fwin2k/operation/vmcomp01/vmcomp01_02.html 2013/9/23

リモートの管理端末から Remote CLI[329]（コマンドライン・インターフェイスによるリモート管理ツール）を使用することによって一部の操作をおこなうことができる[330]。

ゲスト OS がインストールされるのは、VMkernel の上で動作する仮想化された x86 ハードウェアである。ゲスト OS は仮想マシン上で動作していることを特に意識する必要はない。言い換えれば、Windows や Linux を含む x86 に対応した OS を、特別な変更を必要とせずにそのまま稼働させることができる[331]。それぞれの仮想マシンは独立性をもっており、どれかひとつの仮想マシンの障害が他に対して影響を及ぼすことがないようになっている[332]。

また VMware VMotion という製品を使うと、ゲスト OS を稼働させたまま仮想マシンをほかの物理サーバーに移動させるライブマイグレーションの実行が可能になる[333]。VMotion は仮想マシンの移動にダウンタイムを必要としないため、サーバーのメンテナンス時の計画的な停止の時間帯に有効活用できる。加えて Storage VMotion の機能により、同様に仮想マシンの実行中に仮想ディスクを、別のストレージに移動させることも可能である[334]。Storage VMotion はストレージの増設にともなう移行などの際に、ダウンタイムなしに作業をおこなうことができ、VMware VMotion と同様にデータセンター全体の利用に有効とされる[335]。

1）ヴイエムウェア社が提供するソフトウェア製品

ヴイエムウェア社が提供する製品には、VMware Workstation、VMware ESX、VMware ESXi、VMware Infrastructure、VMware Server（現在配布終了）、VMware Player などがある。いずれも x86 および x64 プロセッサを搭載するコンピュータで動作する仮想マシン環境構築用のソフトウェアである。製品は、有償で提供しているものと、無償提供しているものがある。以下一覧のかたちで整理する[336]。

---

329 CLI とは、キーボードからコマンドと呼ばれる命令語を打ち込んでパソコンを操作すること。
330 出所：http://www.atmarkit.co.jp/fwin2k/operation/vmcomp01/vmcomp01_02.html 2013/9/23
331 同上。
332 同上。
333 同上。
334 同上。
335 同上。
336 主として VMware 徹底入門 第２版（2011）翔泳社からの引用をもとに筆者加筆。VMware に関する製品情報は 2013 年 8 月 11 日のデータをもとに作成しており、その後、製品の機能や名称の変更ならびに有償・無償の改変などの情報は織り込まれていない。

＜有償提供製品＞

VMware Workstation：ワークステーション用の仮想化ソフトウェア。

VMware Infrastructure（VI）：データセンターにおけるサーバー統合と集中管理をおこなうもの。仮想マシン機能を提供する VMware ESX と、管理機能を提供する VMware Virtual Center を中心に、多数の製品やオプション機能から構成される組み合わせ製品。

VMware Virtual Center：複数の仮想化システム（ESX）を一元的に管理運用するもの。

VMware vCenter LabManager：開発・テスト環境で利用される仮想マシン群の集中管理をおこなうもの。

VMware Fusion：個人ユーザーをターゲットとした Intel Mac 用仮想マシンで、Windows ゲストにおいて DirectX をサポートするもの。

VMware View：WindowsXP や Vista を仮想デスクトップとして使用するための組み合わせ製品。

VMware ThinApp：アプリケーションの仮想化をおこなうソフトウェア。

vSphere4：VMware Infrastructure（VI）3 の後継となる組み合わせ製品。世界初のクラウド OS として 2009 年 5 月より出荷されている。後継は vSphere5 へと続く。

＜無償提供製品＞

VMware Server：コンピュータ・サーバー用の仮想化ソフトウェア。当初は GSX Server という名称にて有償で提供されていた。2011 年 1 月をもって配布終了。

VMware Player：PC 用の仮想化ソフトウェア。

VMware vSphere Hypervisor ：従来の VMware ESXi。

VMware ESX からコンソール OS を除き、さらに単独での運用に特化したクラスター機能を含まないもの。マイクロソフトの Hyper-V に対抗するために無償公開される前は Infrastracture の一部として提供されていた。ハイパーバイザー型の仮想化 OS で、VMware のサイトから無償ダウンロードできるほか、多くのハードウェアメーカーから USB メモリなどで組み込み済みの形でも提供されている。サービス・コンソールは別のコンピュータで実行し、ネ

ットワーク経由で操作する。なお、本製品は無償製品であるが、ユーザー登録をおこなわないと 60 日間の試用期間後には使用できなくなる仕組みがとられている。

VMware vCenter Converter：既存のシステムを仮想マシンに変換・移行するユーティリティ。

VMware ESX、ESXi はベアメタル型のハイパーバイザーであり、ホスト OS が存在せず、VMkernel と呼ばれる専用のホストカーネルが直接ハードウェア上で動作し仮想マシン環境を構成する。VMware ESX、ESXi がサポートする OS は以下である[337]。現在使われているほとんど全ての OS がサポートされている。また、ヴイエムウェア社が公式にサポートを表明していないものであっても、PC/AT 互換機で動作する OS の多くが実行可能である[338]。

Windows operating system

Unix and Other operating systems

Mac OS X server, eComStation, FreeBSD, IBM OS/2 Warp, Netware, Solaris

Linux operating systems

Asianux Server, CentOS, Debian, Fedora Desktop Edition, Mandrake Linux, Mandriva, Oracle Enterprise, Red Hat Enterprise Linux, Red Hat Linux, Sun Java Desktop System, SUSE Linux Enterprise

VMware の場合、仮想化階層によって、すでに OS のメーカーがサポートを終了してしまっている OS でも、VMware ESX、ESXi 上では動作させることが可能となる。例えば Windows をみると、以下のような新旧の OS が VMware ESX、ESXi ではサポートされている[339]。

---

337 VMware ESXi/ESX 3.5 and later, VMware Workstation 4.0 and later, VMware Fusion1.0 and later, VMware ACE 1.0 and later がサポートする OS。出所はヴイエムウェア社の公式サイト。http://partnerweb.vmware.com/GOSIG/home.html 2013/9/23

338 出所：http://business.acrovision.jp/vmware/words/34.html 2013/9/23

339 出所：http://partnerweb.vmware.com/GOSIG/home.html 2013/9/23

Windows Server 2012, Windows 8, Windows Server 2008 R2, Windows 7, Windows Server 2008, Windows Vista, Windows Server 2003, Windows XP, Windows 2000, Windows NT 4.0, Windows ME, Windows 98, Windows 95, MS-DOS 6.22 and Windows 3.1x

2）ヴイエムウェア社の x86 仮想化における貢献

ヴイエムウェア社は自社のホームページでx86仮想化における貢献を以下のように説明している[340]。

仮想化の技術は古くはメインフレーム時代に既に存在したものであった。しかし、メインフレームの場合とは異なり、x86 コンピュータは完全な仮想化をサポートするようには設計されていなかった。ヴイエムウェア社は、x86 コンピュータで仮想マシンを作成するために、大きな課題を克服する必要があった[341]。

メインフレームと PC に内蔵されている多くの CPU の基本的な機能は、格納されている一連の命令 （ソフトウェア・プログラム） を実行することにある。 x86 のプロセッサには、仮想化において問題となる 17 種類の特定の命令がある。これらの命令により、警告が表示されたり、アプリケーションが停止したり、すべてがクラッシュすることがある。x86 コンピュータ上へ最初に仮想化を実装する際、これらの 17 種類の命令が大きな障害となった[342]。

問題を引き起こすこれらの命令を x86 アーキテクチャで処理するため、ヴイエムウェア社は問題となる命令が生成されると、それを「トラップ」[343]し、仮想化が可能な安全な命令に変換するという仮想化手法を開発した。ほかのすべての命令は、この処理の影響を受けずに実行される。これにより、ホストハードウェアに適合し、ソフトウェアの全体的な互換性を維持する、仮想マシンを実現することができた。ヴイエム

---

340 記述に関しては、主として
http://www.vmware.com/jp/virtualization/virtualization-basics/history.html 2013/9/23
からの引用による。
341 同上。
342 同上。
343 コンピュータで例外が発生した時、それを捕捉し特定の処理を行わせる機能。

ウェア社はこの手法をいち早く実用化した企業である[344]。その貢献により、仮想化テクノロジー開発の先駆的リーダーとして知られている。

3）ヴイエムウェア社設立の歴史

サーバー仮想化の歴史は古く、IBM 社のメインフレーム用にリリースした IBMSystem/360 向けの OS にまで遡る。スタンフォード大でヴイエムウェア社の創業者の１人でもあるローゼンブラム准教授のグループが、メインフレームでおこなっていた技術を x86CPU のシステムに応用した。この技術が確立し製品化の目処が立って、ヴイエムウェア社は 1998 年シリコンバレーでスタンフォード大と UC バークレーの 5 人の研究者[345]によって設立された。

以下では、ヴイエムウェア社設立後の初期の経緯を時系列で簡単に説明する[346]。

1998 年 1 月　　ヴイエムウェア社創立（米国カリフォルニア州パロアルト市のビレッジ・チーズ・ハウスと呼ばれていたみすぼらしいオフィスにて）

2000 年 5 月　　デル社ならびにベリタスソフトウェア社による企業投資家と、アズレ・キャピタル・パートナーズ社、チェース H&Q 社ならびにゴールドマン・サックス社による機関投資家から 2000 万ドルを調達[347]。シリコンバレー投資銀行コミュニティから上場を投げかけられるもこの時点では非公開を保持。

---

344 出所：http://www.vmware.com/jp/virtualization/virtualization-basics/history.html 2013/9/23

345 ダイアン・グリーン（Diane Greene），主任科学者:メンデル・ローゼンブラム （Dr. Mendel Rosenblum），チーフ エンジニア:スコット・ディバイン（Scott Devine），チーフ エンジニア:エドワード・ウォン （Dr. Edward Wang），エドワード・バグニオン （Edouard Bugnion） 出所：http://www.vmware.com/jp/company/leadership/ 2013/9/23

346 Yoffie, Hagiu & Slind（2009）をもとに筆者加筆。

347 “Dell Leads $20 Million Strategic Investment Round in VMware,” http://www.vmware.com/mena/company/news/releases/financingpr.html 2012/8/28

2001年　　ホストOSのいらないハイパーバイザー型　VMware ESX をリリース。

2002年 11月　　マイクロソフト社から企業買収の提案を受けるが、拒否[348]。

2003年 12月　　ストレージベンダーのリーダー企業EMC社に6,350万ドルで買収され100%子会社[349]となることを発表する。EMC社と直接競合関係にあるHP社やIBM社に配慮し、ヴイエムウェア社は独立した子会社として運営されることとなる。

2007年8月　　ニューヨーク証券取引所で保有株の一部を新規株式公開（IPO）[350]。その直前、シリコンバレーの２大企業（インテル社とシスコシステムズ社)がそれぞれ2.5%と1.5%の株を保有[351]し、EMC社の株保有率は86%となる。この年、売り上げは前年度の80%以上の増加で13億2581万ドル、利益マージンは17%[352]となる。また、歳入の22%をR&Dに投資[353]。

---

348 “VMware Prepares New Products and Plans IPO,”
http://www.thefreelibrary.com/VMware+Prepares+New+Products+and+Plans+IPO.-a0106123697 2012/8/28

349 “EMC to Acquire VMware for $635 Million,”
http://www.crn.com/news/storage/18840273/emc-to-acquire-vmware-for-635-million.htm 2012/8/28

350 “EMC Plans to Sell Public 10% Slice of VMware Software Unit,”
http://online.wsj.com/article/SB117089134669901506.html　2012/8/28

351 “Intel to Invest in Virtualization Leader,
http://online.wsj.com/article/SB118402275833561461.html　2012/8/28
“Cisco Investment Reflects Rush into Virtualization,”
http://online.wsj.com/article/SB118554086561980271.html　2012/8/28

352 Yoffie,Hagiu & Slind (2009) “VMware,Inc.,2008 ” Selected Financial Information,2004-2008 p.17

353 VMware,Inc.,Form 10-K, February29,2008,p.17

2008年7月　　マイクロソフト社の元幹部ポール・マリッツ(Paul Maritz)[354]をCEOに迎える。この年の中頃、従業員が6000人を超える[355]（2003年時は370人）。

4）VMwareユーザ会（VMUG）の設立とコミュニティ

ヴイエムウェア社の日本法人は、世界の各国で展開しているVMware User Group (VMUG)[356]の日本組織として「VMwareユーザ会」[357]を2011年に設立した。そこでは、VMware製品に関する戦略、技術などの情報の共有、VMware及びパートナー製品の活用方法の研究と同時に、会員相互のコミュニケーション、親睦などのネットワーク構築をおこなえる「場」の提供を目指している。

具体的には、会員企業間で、現場にあるノウハウや技術情報などの相互共有できるように、定期的に「仮想インフラ最適化部会」、「クラウド部会」、「デスクトップ仮想化部会」などの部会を開催し、会員のヴイエムウェア社の製品及び関連ソリューションに対する理解の深化をおこなっている[358]。

会員資格は、VMware ｖSphereやVMware vCenter Serverなど企業向け製品を導入し、利用している団体およびその事業所、部門とし、会費は無料である[359]。

また、Japan Online Forum[360]というコミュニティサイトでは、ウェブ上でユーザーがヴイエムウェア社が提供するソフトウェア製品の使用法に関する相談やトラブル

---

354 ポール・マリッツは2008年7月にヴイエムウェア社にCEOとして入社。 1978年にロンドンのBurroughs社でそのキャリアをスタートしたマリッツは、1981年にインテル社に入社、5年間ソフトウェア開発ツールの分野に携わる。1986年から2000年までの14年間は、5人で構成される幹部委員会メンバーの1人として マイクロソフト社の運営に関与し、Windows95、Windows NT、およびWindows2000 などのシステム・ソフトウェア製品、Visual Studioなどの開発ツール、SQLServerなどのデータベース製品、およびExchange製品ラインの開発とマーケティングを指揮。
http://www.vmware.com/jp/company/leadership/paul-maritz.html より引用。2012/8/28
ちなみにヴイエムウェア社は、CEOのポール・マリッツが2012年9月をもって退任、後任がパット・ゲルシンガー（Pat Gelsinger）、現EMC情報インフラストラクチャ製品部門のプレジデント兼CEOになることを発表。マリッツはヴイエムウェア社の取締役会に残り、EMC社でニューテクノロジーストラテジストの役職につく予定。
“VMware Announces Changes in Executive Leadership and Preliminary Second Quarter Financial Results” http://www.vmware.com/company/news/releases/vmw-exec-change-07-17-12.html 2012/8/28

355 Yoffie, Hagiu & Slind（2009）, p.6 参照。

356 出所：http://www.myvmug.org/ 2013/9/23

357 出所:http://www.vmware-usergroup.jp/outline.html 2013/9/23 2013年12月の時点で、250団体（企業）の会員が所属しているとのこと。

358 同上。

359 同上。

解決法などを投稿し、それに別のユーザーが答えるという仕組みがつくられている。様々な製品のカテゴリに分けて、多くの投稿が寄せられている。

５）サーバー仮想化とその方式

仮想化（Virtualization）とは、コンピュータにおいて物理リソース[361]の抽象化を指す用語である。仮想化技術によって、単一の物理リソースを複数の論理リソースに見せかけたり、複数の物理リソースを単一の論理リソースに見せかけたりすることが可能となる。言い換えれば、この技術によってコンピュータ・リソースを物理的構成にとらわれずに論理的に統合や分割、または交換することができるようになる[362]。

サーバーの仮想化[363]の機能として、パーティショニング[364]、隔離[365]、カプセル化[366]などがあり、これらの機能により、１台の物理サーバーマシン上で複数の OS を稼働させたり、障害（クラッシュ、ウィルス感染、パフォーマンス低下など）を仮想マシンに隔離し、他の仮想マシンへの影響を防いだり、ファイルの移動とコピー同様、仮想マシンを容易に移動およびコピーができるなどのメリットを享受できる[367]。

サーバー仮想化のソリューションとして仮想マシン方式があり、仮想マシン方式はホスト OS 型とハイパーバイザー型に大別される[368]。ホスト OS 型は、ホスト OS 上に仮想ソフトウェアをアプリケーションとしてインストールし、その上で仮想マシンを動作させる技術である（図７-６）。

図７-６では、仮想化を利用しない場合、ホスト OS の上でアプリケーションが動作する（図の左側）。仮想化を利用する場合、ホスト OS 上に仮想レイヤーが形成され

---

360 出所：https://communities.vmware.com/community/vmtn/vmug/forums/asia_pacific/japan 2013/9/23

361 「サーバー」「OS」「アプリケーション」「ハードディスク」など。

362 出所：http://www.infraexpert.com/study/virtual.html 2013/9/23

363 仮想化には、他にデスクトップの仮想化、ストレージの仮想化、ネットワークの仮想化などがあるが、本論文ではサーバーの仮想化に焦点をあてる。

364 １台の物理サーバーのリソースを分割することにより、同時に複数の仮想マシンを実行する機能。

365 同じハードウェア上の仮想マシンどうしを完全に独立した状態で稼働させる機能。

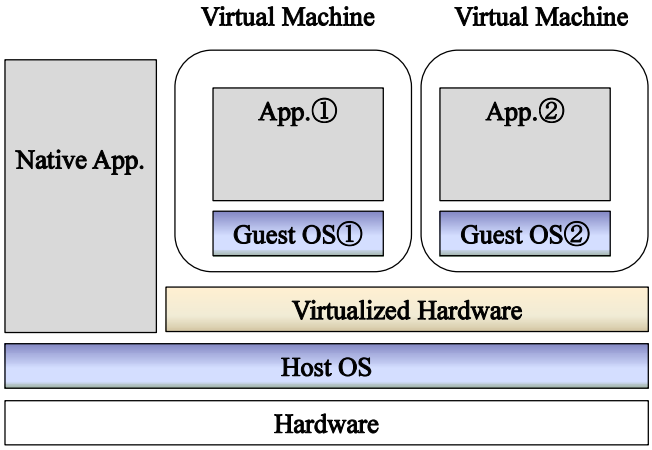
366 ハードウェア構成、BIOS、ディスクの状態など仮想マシン全体を、物理ハードウェアからは独立した少数のファイルに保存する機能。

367 出所：http://www.infraexpert.com/study/virtual.html 2013/9/23

368 同上。

る。その上に仮想マシンがつくられ、ゲスト OS とアプリケーションが載る（図の右側）。

図７－６：ホスト OS 型仮想化階層図[369]



ホスト OS 型の仮想化は、Windows や Mac OS X、Linux といった OS をホスト OS として、その上で仮想化ソフトを実行し、そこで仮想マシンを作成して実行する方法である。仮想化のための専用の OS を用意する必要がなく、あたかもひとつのアプリケーションと同じように手軽に実行できるため、別のマシンを用意せずに他の OS を新たにインストールしたい場合や、実験・検証環境などの用途に向いている。また、無償で利用できるソフトも多い[370]。

このホスト OS 型は手軽であるというメリットがある一方、デメリットとして実行時のオーバーヘッド[371]が大きいため、仮想マシンの動作速度が遅くなってしまう点がある。仮想化ソフトによっては、専用のドライバやツール類が標準で用意されているので、これらをインストールすることによって多少パフォーマンスが改善されることもある[372]。

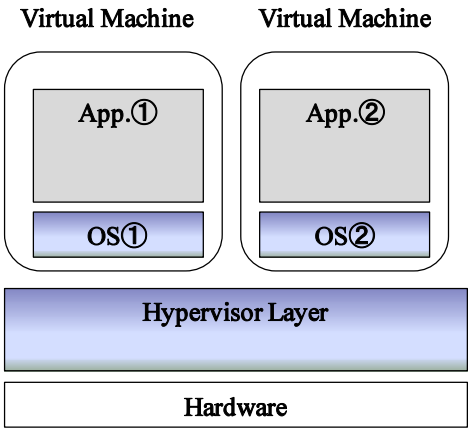
---

369 渡辺・川添（2012）, p. 17 をもとに筆者作成。
370 出所：http://www.infraexpert.com/study/virtual.html　2013/9/23
371 間接費という意味。コンピュータの分野では、何らかの処理を進める際に、間接的・付加的に必要となる処理とそれにより発生する負荷の大きさのことを指す。
372 出所：http://www.infraexpert.com/study/virtual.html　2013/9/23

一方のハイパーバイザー型は、ハードウェアの BIOS[373]上に直接、仮想ソフトウェアを動かし、その上で仮想マシンを動作させる技術である（図７－７）。ホスト OS 層を介さずに仮想マシンが稼働するため、オーバーヘッドが少なくて済む。それによりホスト型よりもパフォーマンスが高く、サーバー仮想化の主流となってきている[374]。

図７－７では、ハードウェア上にハイパーバイザーによる仮想化レイヤーが形成され、その仮想化レイヤー上に仮想マシンがつくられ、ゲスト OS とアプリケーションが載る。

図７－７：ハイパーバイザー型仮想化階層図[375]



ハイパーバイザー型の場合、仮想マシンの実現方法に「完全仮想化」と「準仮想化」の 2 通りの方法がある。完全仮想化は、物理マシンを完全に仮想マシンとして実現する方法で、Windows などの OS に手を加えることなくそのまま仮想マシンで実行することが可能である。VMware ESX、ESXi はこの方法である[376]。

---

373 BIOS（バイオス）Basic Input/Output System とは：コンピュータに接続されたディスクドライブ、キーボード、ビデオカードなどの周辺機器を制御するプログラム群。これらの機器に対する基本的な入出力手段を OS やアプリケーション・ソフトに対して提供する。

374 出所：http://www.infraexpert.com/study/virtual.html 2013/9/23

375 渡辺・川添（2012），p.18 をもとに筆者作成。

376 出所：http://www.plathome.co.jp/solution/virtualserver/introduction/02.html 2013/9/23

一方の準仮想化は、OS のカーネル内でおこなわれる物理マシンに対するシステムコールに手を加え、ハイパーバイザー固有の「ハイパーバイザーコール」[377]を呼び出す形に修正して実行する方法である。ハイパーバイザーの機能を直接利用するため、オーバーヘッドを抑えて仮想マシンを高速に実行することができる。この方法でのソフトウェアは、シトリックス社の Citrix XenServer やノベル社といったベンダーから提供されている[378]。

６）サーバー仮想化の歴史と背景

仮想マシンの歴史は古く 1972 年に IBM 社がメインフレーム用にリリースした System/370 まで遡る[379]。その当時の仮想化は、高価なコンピュータ・リソースを無駄なく余すことなく使用することが主たる役目であった。それから 26 年後、スタンフォード大学のメンデル・ローゼンブラム准教授のグループはメインフレームで使っていた仮想化技術をベースに、x86 CPU のシステムで仮想化を実現する研究をおこなっていた。この技術が確立され製品化の目処が立って、ヴイエムウェア社はシリコンバレーで 1998 年に設立され、翌 1999 年にまず Linux で仮想マシンを動かす VMware Workstation が、その後 Windows で仮想化をおこなう製品をリリースした[380]。ヴイエムウェア社は、設立当初はテクノロジー・カンパニーで、シリコンバレーによくある「こんなテクノロジーを作ってみたけど、何かに使えないかな」的な存在であった[381]。

ホスト OS 型の仮想化ソフトウェアの場合には、仮想化ソフトウェアはホスト OS のアプリケーションとして動く。その場合、仮想マシンへの CPU 割り当てやメモリアロケーション、ディスク/ネットワークの処理はすべてホスト OS がおこなうこととなる。また、ホスト OS 型の仮想化ソフトウェアでサーバーOS を仮想マシンで動かす場合には、他のアプリケーションを一緒に動かすことはほとんどないという現状があった[382]。言い換えれば、業務では仮想化ソフトウェアだけをホスト OS で動かすことが一般的であった。その際、仮想マシンはオーバーヘッドが大きくなり処理が重いことや、仮

---

377 同上。
378 渡辺・川添（2012）, p. 45 参照。
379 歴史の記述に関しては、主として http://enterprisezine.jp/iti/detail/362?p=2　2013/9/23 からの引用による。
380 出所　http://enterprisezine.jp/iti/detail/362?p=2　2013/9/23
381 同上。
382 同上。

想マシンに最適なリソース制御をおこなえないことが、データセンターでの使用での課題となっていた。また、ホストOSのメンテナンスやホストのクラッシュなどのトラブル時は、その上で動いている仮想マシンがすべてダウンしてしまうことも大きな懸念事項であった[383]。

そこで、仮想マシンのための専用OSの必要性が高まることとなった。この専用OSの働きをするものをVMwareではVMkernelと呼び、現在ハイパーバイザーと呼ばれるものとなる[384]。基本的な仮想化アーキテクチャはVMware Workstationのものを引き継ぎ、それまでホストOSでおこなわせていたリソーススケジューリングやディスク/ネットワークに対する入出力を新たにVMkernelでおこなわせた。この製品は、2002年、ESX Serverと名づけられ、その後継は現在のヴイエムウェア社の仮想化製品の主流となっている[385]。

ヴイエムウェア社がESX Server 1.0をリリースしたのとほぼ同時期、イギリスのケンブリッジ大学でイアン・プラット（Ian Pratt）教授のグループが新しいハイパーバイザーの開発を開始した[386]。当初からハイパーバイザーを前提とした設計で、仮想化のCPUオーバーヘッドを小さくし仮想化におけるパフォーマンスの損失を最小限に抑えるようにしたことが特徴である。このため仮想マシンで動くOSは、ハイパーバイザーと協調して動くように修正する。これが、準仮想化OSと呼ばれるものである[387]。

2005年初頭、オープンソースのソフトウェアパッケージとしてリリースされていたXenはLinuxディストリビュータ、CPUベンダーだけではなくサン社やHP社、IBM社などから、サポート表明を受けた[388]。これによりXenを取り巻く環境は大きく変わることとなる。Linuxはカーネルを準仮想化し、Xen上で仮想マシンとして動くようにし、Xenのプロジェクトリーダであるイアン・プラットはヴイエムウェア社と同じシ

383 同上。
384 同上。
385 同上。
386 同上。
387 同上。
388 同上。

リコンバレーで 2005 年に、Xen ベースの商用ソフトウェア・ビジネスのため XenSource 社（後にシトリックス社に買収される）を設立する[389]。

Xen はカーネルの修正を必要とするため、当初は Linux 用仮想マシンソフトウェアという位置付けであったが、インテル社と AMD 社が CPU で仮想化の支援をおこなう機能を発表したことで、Xen は準仮想化 OS だけでなく、Windows のように修正していない OS もゲスト OS として動かすことができるようになった[390]。

サーバーの仮想化の歴史を考える際、ソフトウェアと同様にハードウェア性能のトレンドも考慮することが必要である。具体的には、新たなアーキテクチャによるチップの開発などにより、価格対性能比の大きな向上がある。ムーアの法則[391]はチップの性能に関する向上であるが、それによりハードウェアの処理能力は日進月歩で向上している。渡辺・川添（2012）によると、メインフレーム当時の仮想化は、高価なコンピュータ・リソースを無駄なく余すことなく使用することが仮想化の主たる役目であったが、現在の x86 CPU のシステムで仮想化はその趣旨とはだいぶ異なっている。具体的には、IA(Intel Architecture)サーバーの場合は、メインフレームと比較すると、ハードウェアの価格自体が格段に安いために、リソースの不足はサーバーの数を増やすことで対応されている。アプリケーションごと、用途ごとに別のサーバーを購入することは運用の安定性とシステム管理の単純さからもメリットがある。サーバーのハードウェアの価格の低下も、この風潮を後押しした[392]。

しかし、サーバーの台数が多くなるにつれて、新たな問題が浮上してきた。具体的には管理対象の増加による管理負担の増大、設置スペース、消費電力、発熱量などの増大である。加えて、地球温暖化への懸念や電力消費削減の圧力が高まり、原油高による電力コストの増大などに対して最小限のエネルギーで高効率に運用する風潮が強くなってきた[393]。これまでのようなアプリケーションごと、用途ごとにサーバーを用意する形態では、個々のサーバーの利用率は低くなる。平均的なサーバー利用率は

---

389 同上。
390 同上。
391 世界最大の半導体メーカーであるインテル社の創設者の一人であるゴードン・ムーア（Gordon Moore）博士が 1965 年に経験則として提唱した、「半導体の集積密度は 18～24 ヶ月で倍増する」という法則。
392 渡辺・川添（2012）, p. 14 参照。
393 同上。

10%～20%程度[394]にとどまっているという調査データもある。よって、サーバーが本来発揮できるはずの処理能力の使われていない 8 割 9 割を、サーバーの仮想化によって集約し、複数のサーバーを 1 台のサーバー・ハードウェア上に集約すれば、利用率を高めることが可能となる。

さらに、このトレンドを後押しする要因となったのは、プロセッサのマルチコア化の進展がある[395]。従来のソフトウェアはマルチコアプロセッサ[396]に最適化された構造ではなかったため、急速に並列度を増したプロセッサの演算リソースを使い切れないという課題があった[397]。サーバーの仮想化は、マルチコアプロセッサの演算性能を使いこなすための効果的な手法のひとつとして考えられた[398]。このように、2000 年代中旬からサーバーの仮想化が急速に拡大したのは、効率化の必要性と、ビジネスにおける柔軟性、そして IA サーバーのマルチコア化が背景として考えられる。

7）サーバー仮想化のユーザーメリット

次に、ユーザーにとってのサーバー仮想化のメリットに関して説明する。

多くの企業ではコスト削減のため IT コストの見直しに取り組んでいる。なかでも、サーバーの台数を適正化するなどのリソースの有効活用や、消費電力や運用管理のコスト低減などは重要課題のひとつとなっている[399]。また一方で、顧客獲得の戦略的なツールとして、ビジネスの変化に迅速かつ柔軟に対応できる IT システムを必要としている。これらの観点で、サーバー仮想化のメリットを説明する[400]。

サーバー仮想化とは、1 台のサーバー（物理サーバー）を複数台の仮想的なサーバー（仮想サーバー）に分割して利用する仕組みである。それぞれの仮想サーバーでは OS やアプリケーションを実行させることができ、あたかも独立したコンピュータのように使用することができる。社内でビジネスや業務の変化に対応すべく新たなシステ

---

394 同上。

395 出所：http://thinkit.co.jp/article/972/1/page/0/2　2013/9/23

396 Multiple core は、ひとつのプロセッサ・パッケージ内に複数のプロセッサ・コアを封入した技術であり、マルチプロセッシングの一形態である。外見的にはひとつのプロセッサでありながら内部的には複数のプロセッサとして認識されるため、主に並列処理をする環境下においては、プロセッサ・チップ全体での処理能力を上げ性能向上を果たすためにおこなわれる。

397 出所：http://thinkit.co.jp/article/972/1/page/0/2　2013/9/23

398 同上。

399 渡辺・川添（2012）, p. 14 参照。

400 メリットの記述に関しては、主として渡辺・川添（2012）からの引用による。

ムを構築する際にも、仮想化環境ではハードウェア等を新たに購入しなくても新サーバーを容易に追加することができるので、変化にすばやく対応できる[401]。

渡辺・川添（2012）によると、サーバー仮想化のメリットは、コスト、統合化、管理・運用の柔軟性、高可用性（HA）、ディザスター・リカバリ（DR）を指摘している。また、現時点での最も確実なメリットは、ハードウェアに依存しなくなるという点からもたらされるものであるとしている[402]。

以下、サーバー仮想化のユーザーにとってのメリットを５点説明する。

① サーバー台数の集約／運用の効率化

大企業では、サーバールームに無計画に設置された多くのサーバー群、部門別に管理されたサーバーなど、数百、数千台ものサーバーが稼働している事例がある。こういった大量のサーバーを管理するだけでも、大変なコストと時間がかかり、サーバーのハードウェアの故障などのリスクもサーバーの台数が増えるほど増大する。さらに問題なのは、常時稼働している大量のサーバー全てが、有効活用されていないことが多い。業務アプリケーションごと、部門ごとに導入したサーバーの CPU 使用率をみると、各サーバーとも意外と低く、サーバーの性能を十分に発揮されていないことがある[403]。サーバー仮想化によって、複数台のサーバーを 1 台に集約することで、サーバー・リソースの無駄を減らし有効に活用できるようになる。また、サーバーの台数が増えることで、管理が複雑になり、設置スペースも広くする必要がある。この点も、IT コストを膨れ上がらせる原因となっているため、管理を容易にし、管理コストを削減することも可能である[404]。

② 過去の IT 投資資産の保護

新しいOSに対応しない既存の古いアプリケーションも、仮想化環境を構築すれば、最新のサーバー上で運用することができる。これにより、これまで投資してきた IT 資産をハードや OS の更新によって無駄にすることを防ぎ、引き続き利用することが

401 渡辺・川添（2012）, p. 14 参照。
402 同上。
403 同上。
404 同上。

可能となる[405]。仮想サーバーは、物理サーバー上では OS とアプリケーションを一体となった 1 つのファイルとして見なすため、ある物理サーバーから別の物理サーバーへと簡単に移動可能となる。通常はサポート切れとなる OS 上のアプリケーションは、何らかのかたちで新バージョンの OS の上に載せ換えるか、もしくは諦めるしかない。しかし、仮想化階層があることによって古いバージョンの OS も、同じハード上で引き続き利用し続けることができる[406]。

③ 事業の継続と災害時対策

地震などの災害が発生してメイン業務の継続が困難になった時、仮想サーバーのシステム部分とデータ部分を、バックアップサイトの仮想化環境にコピーするなどして、同じシステム環境を速やかに立ち上げることにより、短期間で業務を再開することができる[407]。いわゆるディザスター・リカバリーならびにビジネス・コンティニュイティ（事業継続）用としての役目をもつ。仮想化技術により、例えば実際のシステムと同じ設定情報をソフトウェアで管理・保管することで、災害時にそれまで別目的で使用していた予備システムの環境を即座にシャットダウンし、実際のシステムの設定情報を管理・記録したソフトウェアを予備システムのハードウェアで立ち上げることが可能となる[408]。

④業務アプリケーション開発の迅速化

開発環境に仮想化の技術を取り入れることで、新たな物理的な開発マシンを用意するよりも時間を節約できるというメリットが生まれる。一般的に仮想化による開発時のメリットとしては、サーバーの物理的な構造に依存しない環境を柔軟に提供できることが挙げられる。そして、開発環境の基盤部分を仮想化で一度構築してしまえば、要求に応じて新たな開発のソフトウェア環境を迅速に提供できる[409]。

---

405 出所：http://www.atmarkit.co.jp/fserver/articles/fivemin/virtualization/04.html 2013/9/23
406 同上。
407 出所：http://www.itmedia.co.jp/im/articles/0903/09/news093.html 2013/9/23
408 同上。
409 出所：http://www.atmarkit.co.jp/ad/hp/vse0704/vse01.html 2013/9/23

⑤開発におけるコストの低減化

コスト面からも開発環境に仮想化技術を取り入れるメリットは大きい。通常大規模なシステムを開発する際、開発は複数のチームに分かれ、同時並行的に作業が進められる。そのため、複数のチームごとにサーバーが用意され、他のチームの作業に影響しないようにする。それぞれに相当なスペックのサーバーを複数用意することになるので、開発に入る時点で大きな費用が発生することになる。こういった状況において、大規模なシステム開発に仮想化技術を採用すれば、効率的に開発プラットフォームを用意できる。十分なパフォーマンスの物理サーバーを 1 台用意し、その上に仮想化環境を構築する。仮想化を使って複数のバーチャルマシンを作り、各チームの開発ソフトウェア環境として利用する[410]。

開発時には、作業の内容によってチーム毎に開発環境に要求される処理性能にばらつきが生じる。仮に、あるチームの作業負荷が高くリソースが足りなくなったならば、ほかの余裕のあるバーチャルマシンからリソースを借りて再配置することも可能となる。この方法ならばリソースの無駄も少なく、必要最低限のサーバーを導入するだけで済ませることができるなど柔軟な運用が可能となる[411]。

また、サーバーや OS 環境が異なれば、新たなシステムの開発時に前回用意した開発マシンが転用できない状況も発生する。開発現場では、新しい開発サーバーをその都度要求することになるが、むやみに物理的なマシンを新たに購入することもコスト面で課題がある。仮想化環境であれば、こういった新たな開発環境の要求にも十分に応えることが可能である[412]。

8）サーバー仮想化からクラウドへの移行

仮想化のこれまでの用途としては、サーバー仮想化は、主として「IT リソースの効率向上」を目標に進化・発展を続けてきた。現状の低いサーバーの利用率を前提に、仮想化によるサーバー統合でこの利用率を大きく引き上げられる、というのが初期の仮想化技術導入の大きなメリットとしてアピールされてきた[413]。いわゆるコスト削減の視点である。

---

410 同上。
411 同上。
412 同上。
413 出所：https://www.impressrd.jp/idc/story/2011/08/22/1843　2013/9/23

次いで、新しいサービスやアプリケーションを稼働させるためには、まずハードウェアの選定・調達から始まるプロセスを経る必要があり、従来は期間も数週間から数カ月を要していた。しかし仮想化されたインフラを前提にすれば、標準化された仮想サーバー環境を、既存の仮想化インフラのリソースを割り当てて稼働させるという作業なら、数分から長くても数時間程度でできてしまう[414]。これによって生じるメリットも、コスト削減である。

しかし、仮想化技術はコスト削減のためだけの手段ではない。民間調査会社のガートナー社（Gartner）によると、仮想化を単なる集約プロジェクトやコスト削減プロジェクトと見なすべきではなく、クラウド・コンピューティングの採用に向けたロードマップの出発点としても認識することが重要であると説明している[415]。その上で、仮想化からクラウド・コンピューティングへのロードマップには５つのステージがあると提起している。そこでは、ステージ１：サーバーの仮想化、ステージ２：分散仮想化、ステージ３：プライベート・クラウド、ステージ４：ハイブリッド・クラウド、ステージ５：パブリック・クラウドという、仮想化のステージで社内 IT インフラの仮想化は移行していくと説明している[416]。

このように仮想化技術を、単に IT のコスト削減と効率向上のための手段としてではなく、より積極的にビジネス上のメリットに直結する存在として位置付けていく、という考え方が今後主流となっていくと考えられる。

ちなみに、これまで仮想化インフラの構築を強力に推進してきたヴイエムウェア社では、仮想化インフラをクラウド実現のための重要な前提条件と位置付けている。

パブリック・クラウドサービスの場合は、不特定多数が「サービスとして」利用できる点が重要であるが、自社内に保有する IT インフラを前提としたプライベート・クラウドの場合は、社内向けに「サービスとして」提供される。言い換えれば、社内の IT 環境を仮想化するプロセスは、社内の IT 環境をプライベート・クラウド化するということの前段階になっていると理解できる[417]。

加えて、必要とされるサービスレベル、セキュリティ、運用ポリシーといった要件は、企業によってさまざまに異なる。サービス化された IT のリソースは、ユーザー

414 同上。
415 出所：http://www.gartner.co.jp/b3i/research/120223_inf/ 2103/9/23
416 同上。
417 出所：https://www.impressrd.jp/idc/story/2011/08/22/1843 2013/9/23

の需要に応えるための適切な要件を満足させる必要がある。言い換えれば、自社の需要に応じて、プライベート・クラウドを適切に設計していく必要が生じる[418]。これまで社内ネットワークインフラなどの進化に関しては、初めはコストの削減からはじまり、やがて戦略的な IT ツールとして、ビジネスの成果に直接貢献させるといった目標があった。そして近い将来、プライベート・クラウドやパブリック・クラウドの採用に対しても、同じ目標が期待される可能性は高いと考えられる。

9）マイクロソフト社の仮想化への取り組みと Hyper-V

「マイクロソフト社は、既存のビジネスモデルによって数十億ドルを稼いできたし、今も稼いでいる。（中略）しかし、その未来についてもっとも当たり障りのない見方をしている支持者でも、彼らが相手にしている市場で、まもなくすべてがひっくり返ろうとしていることは、否定できないだろう。ソフトウェアは、サービスに道を譲ろうとしている。（中略）そして、ユーザーは、本当の意味で他社製品との相互運用を実現しようとしないベンダーに封じ込められるのを否定するようになってきている。」と Foley (2008)はマイクロソフト社がクラウドの普及により、プロプライエタリなオンプレミス型のビジネスモデルから変化せざるを得ない状況を指摘している[419]。Foley (2008)はまた、マイクロソフト社は、その状況に対応すべく、すでに多くの仮想化ソフトウェアをリリースしていると以下を説明している（以下、該当記載部分の抜粋）[420]。

① バーチャル PC:Windows 上で、古いアプリケーションや互換性をもっていないアプリケーションをホストするためのデスクトップ環境。

②Hyper-V:Windows Server 仮想化、ハイパーバイザーとも呼ばれる。（後述）

③システムセンター・バーチャルマシンマネジャー：ホスト・コンフィグレーション、仮想マシンの作成、ライブラリ管理、インテリジェント VM プレースメント、モニタリング、急速復旧、セルフ・プロビジョニング、オートメーションを管理する製品。

---

418 同上。

419 Foley（2008），（邦訳）p. 256-257 参照。

420 Foley（2008），（邦訳）p. 309-310 からの抜粋（一部筆者修正）。

④バーチャルサーバー：サーバーベース製品で、ホストの Windows Server 上でゲストを実行できるようにする。

⑤ソフトグリッド：アプリケーションの仮想化テクノロジー。

⑥ターミナルサービス：「プレゼンテーション仮想化」ソリューションとして販売できるよう強化されている。

このようなマイクロソフト社の動きは、自社ならびに競合による仮想化製品によって、Windows とマイクロソフト Office がこれまで築いてきた容易に達成できない高いシェアが蝕まれるのを防ぐために、現在と将来の仮想化製品をいつ、どのようにして、どのようなライセンシング形態で顧客に提供するかについて、しっかりと統制しようとしていると、Foley (2008)は指摘している。

前述のマイクロソフト社が提供する仮想化ソフトウェア製品のなかで、ヴイエムウェア社にとってもっとも脅威となる Hyper-V について以下に説明する。

Hyper-V は Windows Server 2008 x64 とセットになって動作するソフトウェアである。Hyper-V は、OS とハードウェアの間に Windows ハイパーバイザーと呼ばれる薄い層を形成する[421]。以前の仮想化のようなエミュレーション層を介すことなく、ゲスト OS の命令がハードウェアに伝わるため、パフォーマンスが改善された[422]。

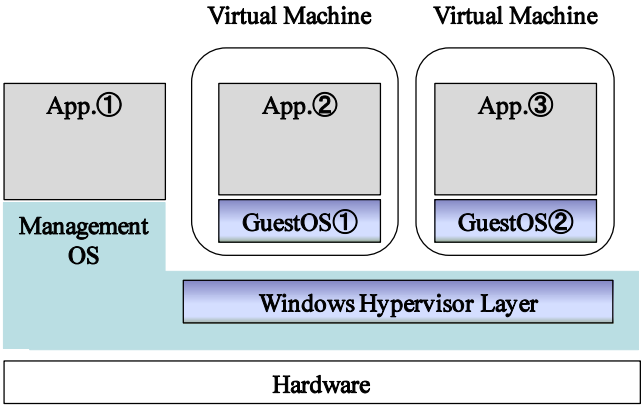
OS として標準搭載されているので、有償である Windows Server 2008 x64 以外の追加コストなしで無償かつ簡単に導入できる[423]。マイクロソフト社のサーバー・アプリケーションと互換性が高く、障害発生時もトータルにマイクロソフト社がサポート可能であることを同社は導入のメリットとして挙げている[424]。

以下は、 Windows Server 2008 x64 での Hyper-V のしくみを階層構造として図示したものである。ハードウェア(BIOS)階層と管理 OS（この場合は Windows Server 2008 x64）を介してのアプリケーションもしくはゲスト OS との間に、Windows Hypervisor 階層が管理 OS に含まれる形態で介在する（図７－８）。

---

421 出所：http://www.atmarkit.co.jp/ait/articles/0809/03/news125_2.html 2013/9/23
422 同上。
423 同上。
424 同上。

図７－８　Windows Server 2008 x64 での Hyper-V の仮想化階層図[425]



サポートする OS は、

Windows Server 2008 x86 Standard/Enterprise/Datacenter/Web, Windows Server 2008 x64 Standard/Enterprise/Datacenter/Web, Windows Server 2003 SP2 x86 Standard/Enterprise/Datacenter/Web, Windows Server 2003 SP2 x64 Standard/Enterprise/Datacenter, Windows 2000 Server SP4 x86 Standard/Advanced Server, Windows Vista SP1 x86 Business /Enterprise / Ultimate, Windows Vista SP1 x64 Business /Enterprise / Ultimate, Windows XP SP3 x86 Professional, Windows XP SP2 x86 Professional, Windows XP SP2 x86 Professional, SUSE Linux Enterprise Server 10 SP1, SP2　x86, SUSE Linux Enterprise Server 10 SP1 SP2 x64[426]である。Linux 系では、リリース中の Windows とシトリックス社やノベル社との協業による SUSE Linux がある[427]。サポート対象は主として、新しい Windows の OS であり、すでにメーカーとしてユーザー・サポートを打ち切った古い OS は含まれていない。この点は、VMware ESX とは異なっている。

１０）Hyper-V の VMware ESX の追い上げ

マイクロソフト社は Hyper-V の VMware ESX に対する優位性を自社のウェブや新聞広告、ユーザー向けのカンファレンスなどで、アピールしている。「VMware vSphere 5 ではなく Microsoft Hyper-V R2 SP1 を選択すべき 5 つの理由－サーバー仮想化に

425 出所：https://www.tis.jp/service_solution/hyper-v/　2013/9/23　をもとに著者作成。
426 出所：http://ascii.jp/elem/000/000/176/176563/#eid176573　2013/9/23
427 同上。

おける VMware に対するマイクロソフトの優位性ー」[428]と記して、自社ウェブでは詳細資料の PDF をダウンロードできるような形態を導入している。そのなかでは、業界内での高い評価と導入顧客、組み込みによるサーバーソフトとの親和性、他のマイクロソフト社製品との連携、可用性の高い管理機能、価格性能比の５点[429]に関して Hyper-V の VMware ESX に対する優位性についてアピールしている。

また、仮想化に関する Gartner Magic Quadrant という、民間調査会社ガートナー社が定期的に発行している業界内のプレイヤーの区分において、後発であったマイクロソフト社をサーバー仮想化インフラストラクチャの「リーダー」[430]として評価している。

こういった、包括的な訴求効果によってか、民間調査会社 IDC 社によると、2012 年第 1 四半期以降、日本のサーバー仮想化市場の四半期売上において、Hyper-V はシェア 1 位の座を保持している。国内のシェアは Hyper-V が３四半期連続でトップとなっている。具体的には、2008 年第４四半期のシェアで、Hyper-V が 3.9% VMware ESX が 29.2%とその差が 25.3%もあったにもかかからず、その後差が徐々に縮小し、2012 年第４四半期のシェアでは、Hyper-V が 41.9% VMware ESX が 38.7%とその差が 3.2%の逆転となっている[431]。

2007 年 9 月、サンフランシスコにて開催されたヴイエムウェア社主催の仮想化イベント「VMworld 2007」での当時の CEO のグリーンは、競合のマイクロソフト社が次期サーバーOS の「Windows Server 2008」に仮想化機能を搭載することに関して、「x86 ベースの仮想化においては、VMware の仮想化ソフトが一番信頼性の高いものだと確信しています。マイクロソフト社の仮想化機能は、ユーザーがどうしても必要だと考える一部分の機能しか提供していません。例えばマイクロソフト社は、仮想マシンを止めることなくライブで移行させることが可能な「V-Motion」のような機能は提供していません。ただ、仮想化の世界はこれからもっと発展していくため、ヴイエムウェア社にとってもマイクロソフト社にとっても戦いの余地はあるでしょう。」と語ってい

---

428 出所：https://www.microsoft.com/ja-jp/server-cloud/windows-server/hyper-v.aspx 2013/9/23

429 同上。

430 同上。

431 出所：http://www.atmarkit.co.jp/ait/articles/1303/14/news004.html 2013/9/23

た[432]。しかし、圧倒的なインストールド・ベースを保有しているヴイエムウェア社であるが、マイクロソフト社の急速な追い上げに四半期ベースの出荷台数では、逆転を許す結果となってしまった。

１１）仮想化のための必要支援機能：VMware ESX、XenServer、Hyper-V の比較

それぞれの仮想化製品を仮想化の方式で比較をすると、VMware ESX、Citrix XenServer、Microsoft Hyper-V のいずれもホスト OS 型ではなく、ベアメタル型のハイパーバイザー方式を採用している。ただし、ゲスト OS を稼働させる仕組みにはそれぞれの製品によって違いがあり、ハードウェア階層やネットワーク管理ツールなどの支援機能がないと稼働できないものもある。

以下に、それぞれ３者の製品の支援機能の説明をする[433]。

VMware ESX では、ゲスト OS に修正を加えることなくそのまま稼働させるために、完全仮想化の仕組みを全面的に採用している。完全仮想化は、64bit のゲスト OS を稼働させるとき以外は Intil VT や AMD-V などのプロセッサの仮想化支援機能に依存しない[434]。

Citrix XenServer では Linux のゲスト OS に対して準仮想化の仕組みを取り入れることにより、より少ないオーバーヘッドでゲスト OS を稼働させることを可能にしている。準仮想化に対応しない Windows などのゲスト OS を稼働させる場合は、プロセッサの仮想化支援機能を利用した完全仮想化によって動作する[435]。

Microsoft Hyper-V では、プロセッサの仮想化支援機能が動作の前提となっており、Windows Server 2003/2008 などのゲスト OS は完全仮想化によって稼働する。ただし、ゲスト OS に統合サービス・コンポーネントを導入することにより、一部において準仮想化の仕組みを取り入れている。また、Linux のゲスト OS を稼働させるために Xen 対応の準仮想化カーネルを使用することもできる[436]。

---

432 出所：http://japan.zdnet.com/interview/20356452/ 2013/9/23
433 比較に関しては主として、出所：http://www.atmarkit.co.jp/fwin2k/operation/vmcomp03/vmcomp03_01.html 2013/9/23 からの引用による。
434 出所：http://www.atmarkit.co.jp/fwin2k/operation/vmcomp03/vmcomp03_01.html 2013/9/23
435 同上。
436 同上。

また、いずれの製品も複数サーバーを集中管理する仕組みを持っているが、その方式には違いがある。例えば、VMware ESX では VMware VirtualCenter、Microsoft Hyper-V では System Center Virtual Machine Manager を集中管理用のサーバーとして使用するため、専用のサーバーが必要となる。一方、Citrix XenServer では 1 台の XenServer がマスターとして動作することによって、XenCenter による集中管理をおこなう際に専用のサーバーを必要としない[437]。

３者の違いを仮想化の方式、完全/準仮想化、仮想化支援機能（Intel VT/AMD-V）、管理コンソール、複数サーバーの集中管理ソフトの観点で下表にまとめる（表７－４）。

表７－４：仮想化製品３者の必要とされる支援機能の一覧[438]

	VMware ESX	Citrix XenServer	Microsoft Hyper-V
仮想化方式	ベアメタル型ハイパーバイザー	ベアメタル型ハイパーバイザー	ベアメタル型ハイパーバイザー
稼働の仕組み 完全/準仮想化	完全仮想化	準仮想化/完全仮想化	準仮想化/完全仮想化
仮想化支援機能（Intel VT/AMD-V）	64bit ゲストのみ必須	必須	必須
必要とされる管理コンソール[439]	VMware Infrastructure Client	Citrix XenCenter	Hyper-V マネージャ
複数サーバーの集中管理をおこなう管理ソフト	VMware VirtualCenter	Citrix XenCenter	System Center Virtual Machine Manager

この表が示すように、VMware ESX、Citrix XenServer、Microsoft Hyper-V の 3 者ともハードウェア側の支援機能が必要（VMware ESX は 32bit ゲストの場合は不要）となる。また、管理するためのコンソールや複数サーバーの集中管理をおこなう管理ソフトも必要となる。これらは全て有償で提供されているものを購入する必要があり、オペレーションのスキルの習得のためのトレーニングなどを必要とする。このように

437 同上。

438 出所：http://www.atmarkit.co.jp/fwin2k/operation/vmcomp03/vmcomp03_01.html 2013/9/23 の表を一部筆者修正。

439 コンピュータを操作するために使う入出力装置のセット。制御卓、操作卓などとも呼ばれる。

仮想化を実現するためには、３者とも単なるひとつのソフトウェアの導入だけではなく、システム全体の支援環境を整えねばならない。このことは、導入ならびにインストール後の他者への乗り換えなど、スイッチングコストに大きく関わる要素である。

１２）VMware における成果の限定性

VMware の事例では、マイクロソフト社は Windows に併設する仮想化階層を形成する製品である Hyper-V を、VMware の対抗製品として自社サーバーソフトにバンドルするかたちで、2008 年 6 月に市場に投入してきている[440]。いわゆるプラットフォーム包囲攻撃である。Hyper-V はヴイエムウェア社の主力製品である VMware ESX と、直接対抗する製品となる。Hyper-V の仮想化機能は、「Windows ハイパーバイザー」と呼ばれる仮想化技術をベースとしている。Windows Server の 64 ビット版の機能のひとつとして、例えば Windows Server 2008 x64 などに組み込まれて提供されているほか、同社ウェブサイトで「Hyper-V Server」として単体で無償提供されている[441]。

本来 Windows を含む Linux や Unix ならびに MacOS に対してオープン性をもつ VMware にとって、数多くあるなかの一種類の OS からプラットフォーム包囲の攻撃を受けても、理論的にはそれほどのダメージを受けることはない筈である。しかし、Windows は他の OS と比べて、かなりの大きなシェアをもつため、ヴイエムウェア社にとっての相応のダメージとなったと理解できる。

１３）ヴイエムウェア社の戦略と今後の方向性

仮想化技術は、数十年前から存在する。しかし、x86 チップを搭載した主流のコンピュータに搭載されたことにより、同技術は一気に表舞台に飛び出し、技術ならびに企業の将来性から投資の対象として注目を集めた。EMC 社の子会社であるヴイエムウェア社は、2007 年 8 月に新規株式公開（IPO）をおこなった[442]。

ヴイエムウェア社の将来の方向性に対して、CEO のダイアン・グリーンは 2007 年当時、以下のようにコメントしていた。2007 年 9 月、サンフランシスコで開催されている同社主催のイベント VMworld 2007 で開かれた記者会見で、「現在、売り上げの 8

---

440 出所：http://technet.microsoft.com/ja-jp/evalcenter/dn205299.aspx 2013/9/23
441 同上。
442 Yoffie, Hagiu & Slind（2009）, p.6 参照。

割以上をハイパーバイザー以外で上げている。」と発言した上で、「われわれはこれまで、ユーザーにとっての仮想化の価値を明確にする製品の開発という大変効果的な仕事をおこなってきた。」とコメントした[443]。これは、当時、マイクロソフト社が Viridian という開発コード名で呼ばれるハイパーバイザーを、Windows Server の将来版に搭載する計画の発表を意識しての発言である。

その後のマイクロソフト社のハイパーバイザーへの本格参入により、それまでの VMware ESX から無償版の ESXi にシフトせざる得なかったが、最新製品の vSphere5 では、ヴイエムウェア社はあえて ESXi のみを提供している[444]。ESXi はもともとサーバー組み込み用のハイパーバイザーで市場投入されたものであり、USB メモリなどにインストールして、サーバーベンダーが出荷時に稼働環境構築済みとして販売するための「特殊な軽量版」であった[445]。ヴイエムウェア社がこれに絞り込んだのは、パートナー製品との抱き合わせによるデリバリーにより普及を加速すると同時に、仮想化ソフト自体からは収益を期待しないということであると考えられる。言い換えれば、ESXi を IBM 社、Dell 社、HP 社、NEC 社、富士通のサーバーに搭載し広く普及させることで、付随するサポートソフトや管理ツールを販売し収益源を得ようとする計画であると思われる。

また、これだけでなく、数多くの企業買収を通じて収入源の確保をおこなっている。具体的な例として、ヴイエムウェア社は 2010 年 3 月、RTO Software 社の資産買収とともに同社の親会社である EMC 社の複数の資産の取得も発表した[446]。この 2 億ドルの買収には、いずれも EMC Ionix 社のインフラ管理ポートフォリオの部品である「Server Configuration Manager」(Configuresoft 社)、「FastScale」、「Application Discovery Manager」（nLayers 社）、および「Service Manager」（Infra 社）が含まれている[447]。これにより、もはや仮想化（そしてクラウド・コンピューティング）会社のヴイエムウェア社ではなく「基盤からサービスおよび買い取り型のアプリケーションまでに対

443 出所：http://japan.cnet.com/news/biz/20356303/ 2013/9/23
444 出所：https://my.vmware.com/jp/web/vmware/evalcenter?p=free-esxi5&lp=default 2013/9/23
445 出所：https://www.impressrd.jp/idc/story/2012/02/27/2050 2013/9/23
446 出所：http://virtualization.info/jp/news/2010/03/vmwareemc-ionix20100303-3.html 2013/9/23
447 同上。

応するインフラ管理会社」[448]と変貌することで、新たなビジネスモデルを構築することになる。ヴイエムウェア社は、これまで、1 台のコンピュータ上で複数の OS を同時に稼動することを可能にするハイパーバイザーと呼ばれるコア仮想化ソフトウェアの販売で利益を上げていた。しかし、今や同社の事業は、ハイパーバイザー販売だけにとどまらない。

グリーンは、向こう 10 年間の見通しのなかで、仮想化は一部の人々の間で受け入れられているが、今後は単に受け入れられるだけにとどまらず、至るところで見られるようになると指摘した。「将来、仮想化はハードウェアの随所に見られるようになると考えている。そして、3 台～3000 台のサーバーを利用している企業や組織の自動化されたデータセンターでは、仮想化はごく当たり前のことになるだろう。」と述べている[449]。

（２） VMware 事例採用の理由

今後成長が見込まれる x86 サーバー仮想化市場では、ヴイエムウェア社をはじめとして、マイクロソフト社、シトリックス社、レッドハット社、オラクル社、ノベル社などがそれぞれのサーバー仮想化ソフト製品を提供している。このようななか、階層介入戦略の事例として VMware を取り上げる理由は、ヴイエムウェア社は仮想化ソフト市場において、2008 年までに Fortune100 企業の 100%と Fortune1000 企業のおよそ 90%の顧客をもち、ISV との協業で 700 以上の仮想アプライアンス[450]を市場に投入している。また、ハイパーバイザー型のサーバー仮想化市場における出荷台数で、８割を超える大きなシェアをもつリーダー的存在[451]であるためである。

Yoffie,Hagiu & Slind (2009)は「ヴイエムウェア社のリーダー達は、マイクロソフト社が仮想化市場で興味を示していなかった時代には、自分の会社のポジションを

---

448 同上。

449 出所： http://japan.cnet.com/news/biz/20356303/ 2013/9/23

450 特定の機能に特化したコンピュータのこと。具体的には単機能サーバー、Web 閲覧・メール送受信専用端末などがある。

451 Yoffie,Hagiu & Slind (2008) ,p.22 より VMware ESX のマーケットシェアは 86%、マイクロソフト社の Hyper-V は 3%、シトリクス社の Xen は 9%（2008 年 3 月期データによる 2009 年末までの予測）

仮想化市場でのドミナントなプラットフォームの持ち主として確保し拡張する野心的で複合的な戦略を発展させてきた。」と説明している[452]。

（３） VMware における操作項目の事例整理

次に、VMware においてアクセス可能ユーザー数の増加 、マルチホーミングコストの低減、隣接対象プラットフォーム製品の多数選定、持続的収益確保モデルの遂行の４つの項目の観点から事例を整理する。

１）アクセス可能ユーザー数の増加

VMware は、既に普及している Windows や Linux や Unix が、それぞれに保有しているユーザーが創りだすサイド内ならびにサイド間ネットワーク効果を譲り受け、それら OS を上位階層としてその種類に縛られない介入階層として下位階層に介入した。言い換えれば、介入によってアクセス可能ユーザーを横取りしていると言える。

初期のころヴイエムウェア社とマイクロソフト社は、良き協業としてのパートナーの関係であった。2000 年 3 月には両社は OEM の同意[453]を結んでいる。内容は VMware 上に Windows がプレインストールされたいくつかのバージョンを、顧客に購入できるようにしたものであった。これは、抱き合わせによってアクセス可能ユーザーを増加させ、階層間ネットワーク効果の増大を目論んだものであると理解できる。

また、VMware は積極的に開発者や販売チャネルと手を組み、2007 年終わりまでに、30 パートナー企業の 350 人以上の開発者が VMware ESX ソースコードにアクセスするプログラムを扱えるようになった[454]。そのソースコードにアクセスすることは、ヴイエムウェア社との共同開発プロジェクトに参加したり、開発者自身でアプリケーションをデザインしたりすることを可能にした。この開発者の育成により、階層間ネットワーク効果を引き出す目的があった。

---

452 Yoffie, Hagiu & Slind（2009）, p. 13 参照。

453 “VMware Signs OEM Agreement with Microsoft”
http://www.vmware.com/company/news/releases/microsoftpr.html 2012/08/25

454 VMware, Inc.（2008）, p. 6 参照。

2）マルチホーミングコストの低減

VMware は、その巧みなマーケティング戦略において、第３者のアナリストにサーバーの低い稼働実態や、IT 設備の複雑さによる高い管理コストをホワイトペーパー（技術白書）等でアピールさせるなどして、ユーザーの需要を煽った。その上で「仮想化によるひとつの筐体上での複数 OS 利用によるコスト削減」と「コンピュータ・システムの複雑性の回避による管理コスト削減」を掲げ、参入障壁を下げるためにユーザーのマルチホーミングコストの削減を強くアピールした。仮想化のメリットを標榜するヴイエムウェア社のユーザー向けウェブサイトでは、「サーバー統合によって設備投資コストを削減し、自動化によって運用コストを低減できます。（中略）運用効率を向上し、必要なハードウェアを減らす（後略）」と説明している[455]。

3）隣接対象プラットフォーム製品の多数選定

ターゲットとする隣接階層の選択の点において、VMware も広く普及した（もしくは普及しそうな）階層内のプラットフォーム製品をターゲットに選定して介入した。具体的には、既に広く普及している Windows や Linux や Unix などの隣接階層として、それらを梃子に介入した。VMware の場合は、OS 階層とバードウェア（BIOS）階層間に上位階層に対してオープンなインターフェイスをもつ製品として介入している。

加えて、VMware は、2007 年終わりまでに、500 以上のハードウェアならびに搭載された 60 以上の OS との実装テスト[456]を実施している。この OS との相互運用性の確立によるオープン性の保持により、多くのプラットフォーム製品と隣接することが可能となった。

4）持続的収益確保モデルの遂行

VMware に関しては、根の深い衝突が、コンピュータ・アーキテクチャ内の仮想化の役割において顕在化している。マイクロソフト社は VMware の対抗製品となる Hyper-V が 2010 年度の日本国内の単年度出荷ライセンス数では、VMware を追い越したとのアナウンス[457]をおこなった。世界市場に圧倒的な数のインストールド・ベースの顧客を

---

455 出所：http://www.vmware.com/jp/virtualization/ 2012/11/03

456 VMware, Inc.（2008）, p.7 参照。

457 マイクロソフト社のウェブサイトで、日本国内では「2010 年上半期の導入数で Hyper-V が VMware を上回っています。（中略）Hyper-V が 2010 年上半期 x86 サーバー用仮想化ソフトウェア

抱えているとはいえ、順風に普及をしてきたVMwareも大きな課題に直面してきている。レイヤースタック内に主力となる収益階層を仮想化階層以外にもたないヴイエムウェア社にとって、有効な収益モデルの必要性が生じている。

Yoffie,Hagiu & Slind(2009)はインタビュでのリチャード・サーワル（Richard Sarwal：当時のヴイエムウェア社の幹部）のコメントとして、「我々（ヴイエムウェア社）は、ハイパーバイザーはハードウェアの一部であると信じている。しかし、マイクロソフト社はそれをOSの一部だと信じている。これは根幹的な違いだ。」と紹介している[458]。

Yoffie,Hagiu & Slind(2009)はインタビュでの、ラグー・ラグラム（Raghu Raghuram：ヴイエムウェア社の担当上席副社長）のコメントとして、「ヴイエムウェア社では、（ネットスケープに見られたような）ブラウザー戦争の（顛末の）類似がどのくらい密に仮想化にもあてはまるかという問題が議論される。しかし、変化に対する顧客の抵抗はネットスケープの場合より、かなり高い。VMware ESXそれ自体を交換することは難しくはない。しかし、もし仮に管理ツールを使っていたり、それの周りでオペレーションを構築していたりすると、切り替えコストはとても高くつくためだ。」とコメントしている[459]。

（４） VMware介入による階層間関係とポジションの変化

階層介入型プラットフォーム製品は、既存の複合製品の階層間に後から「介入」することにより階層間関係やプラットフォーム製品関係を、変化させてしまう可能性をもつ（図７－９）。

図７－９は、VMware が介入する前と後での階層構造を表わしている。介入前はｎレベル階層にハードウェア（BIOS）、ｎ＋１レベル階層にOS（ｎ＋２レベル階層にアプリケーション）がある。介入後は、ｎ＋１レベル階層に仮想化階層としてVMware が

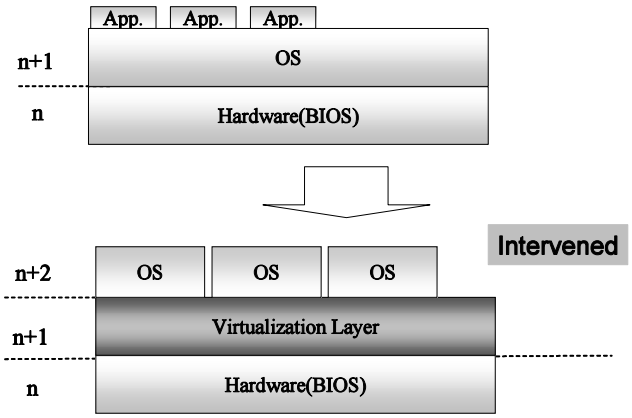
として導入率 No.1 を獲得。」との内容が大手調査機関のレポート引用として掲載されている。
http://www.microsoft.com/ja-jp/business/industry/gov/hyperv/column01.aspx 2012/11/03

458 Yoffie, Hagiu & Slind（2009）, p.4 参照。

459 Yoffie, Hagiu & Slind（2009）, p.15 参照。

介入し、ｎ＋２レベル階層に OS が移行する。介入階層は VMware であり、その上に多くの OS を配することが可能となる。

図７－９：VMware による階層介入図



BIOS:Basic Input/Output System

具体的に、VMware の投入によってそれらの隣接階層にどのような変化をもたらしたのかに関して、階層間の対応関係、各階層のプラットフォーム製品戦略、複合製品内でのポジションを以下にまとめる（表７－５）。

表７－５：VMware 階層介入前後の変化

階層	隣接下位階層	隣接上位階層	仮想化階層（中位）
該当製品	（CPU やチップセットから構成される）ハードウェア物理階層	サーバーOS（Unix・Linux・Windows 他）	VMware ESX, ESXi
参入時期	先発	先発	後発

【階層介入前】

上下隣接階層間のプラットフォーム製品対応関係	ひとつの筐体（ハードウェア）に対し複数の OS を走らせることはで	原則ひとつの OS に対してひとつの筐体（ハードウェア）が必要とな	

	きない。OS と筐体は１対１の対応となる。	る。複数の OS を走らせるためにはその同数の筐体が必要となる。OS と筐体は１対１の対応となる。	
各階層のプラットフォーム製品戦略	隣接することで、グループ間ネットワーク効果を発揮し、プラットフォーム製品戦略を遂行する。ハードウェアメーカーはより多くの普及が進んだ（もしくは進む見込みのある）OS と手を組むことが肝要となる。	隣接することで、グループ間ネットワーク効果を発揮し、プラットフォーム製品戦略を遂行する。	
ソフトウェア・レイヤースタック内でのドミナント状況	（従前から）コモディティ化が進行している。特定のハードウェア物理階層がドミナント・プラットフォーム製品になる可能性は、かなり低い。	隣接階層に数多くのアプリケーションを配することで、特定の OS がドミナント・プラットフォーム製品となる可能性がある。	

【階層介入後】

上下別レベル隣接階層間のプラットフォーム製品対応関係	ひとつの筐体に対し複数の OS を走らせることが可能となる。筐体と OS の対応は１対多となる。また筐体と仮想化製品は１対１の対応となる。	複数の OS をひとつの筐体上で走らせることが可能となる。OS と仮想化製品は多対１の対応となる。また、OS と筐体の対応も多対１となる。	原則ひとつの仮想化製品に対してひとつの筐体が必要となる。仮想化製品と筐体は１対１の対応となる。仮想化階層の上では複数の OS が走る。仮想化製品と OS は１対多の対応となる。
各階層のプラットフォーム製品戦略	理論的にはハードウェアの需要は減るが、ハードウェアメーカー	理論的には既存 OS の新規需要は減るが、新たに VMware 上で走る	OS メーカーやハードウェアメーカーと協業して、グループ間ネットワーク

	[460]は仮想化階層との間でグループ間ネットワーク効果を発揮し、ヴイエムウェア社と協業して拡販している。	特定用途の OS が ISVs 開発者などによって開発されアプライアンスとして拡販される。	効果を発揮し、自社製品を拡販している。
ソフトウェア・レイヤースタック内でのドミナント状況	（従前から）コモディティ化が進行している。特定のハードウェア物理階層がドミナント・プラットフォーム製品になる可能性は、かなり低い。	コモディティ化に陥るリスクが高まる。	ドミナント・プラットフォーム製品になる可能性がある。

上記の表により、以下がわかる。

- ✓ 仮想化階層の介入は、階層介入以前の上位階層と下位階層の階層間での対応を変化させる。
- ✓ 各階層のプラットフォーム製品戦略は、２階層間もしくは３階層間でグループ間ネットワーク効果を利用し拡販を続けるという点では、階層介入前後で変化はない。
- ✓ レイヤースタック内でのポジションは仮想化階層の介入によって、上位階層のサーバーOS はコモディティ化に陥るリスクが高まるのに対して、仮想化階層自体はドミナント・プラットフォーム製品になる可能性が高まる。

（５） 小括

本節では、VMware 事例の考察として、VMware の概要、VMware の事例選択の理由、VMware の操作項目による事例の整理、階層介入による階層間関係とポジションの変化について論じた。VMware の概要では、ヴイエムウェア社が提供するソフトウェア製品、ヴイエムウェア社の x86 仮想化における貢献、ヴイエムウェア社設立の歴史、VMware ユーザ会（VMUG）の設立とコミュニティ、サーバー仮想化とその方式、サーバ

460 ヴイエムウェア社が共同開発を進めたハードウェアメーカーは IBM 社、HP 社、Dell 社、NEC 社、Fujitsu 社、Fujitsu-Siemens 社とサン社などのコンピュータ・メーカーやインテル社、AMD 社のような半導体メーカーとシスコシステムズ社のようなネットワーク機器メーカーも含まれる。

ー仮想化の歴史と背景、サーバー仮想化のユーザーメリット、サーバー仮想化からクラウドへの移行、マイクロソフト社の仮想化への取り組みと Hyper-V 、Hyper-V の VMware ESX の追い上げ、仮想化のための必要支援機能:VMware ESX、XenServer、Hyper-V の比較、VMware における成果の限定性、ヴイエムウェア社の戦略と今後の方向性など様々な角度から事実内容の説明をおこなった。その上で、事例として選択した理由について論じ、前述したプラットフォーム製品提供者の４つの操作項目の観点で確認をおこなった。加えて、階層介入前と後に関して、階層間関係とポジションの変化について考察をおこなった。

第４節　操作項目における両事例の整理

Java と VMware のプラットフォーム製品提供者の操作項目のこれまでの確認を、共通点・相違点の観点でまとめると、以下が分かる。

（１）　共通点

Java はアプリケーション階層と OS 階層間に介入、VMware は OS 階層とハードウェア（BIOS）階層間に介入という違いはあるが、両事例ともアクセス可能ユーザー数の増加、マルチホーミングコストの低減、隣接対象プラットフォーム製品の多数選定の３つの項目に関して共通の動きを遂行している。

具体的には、アクセス可能ユーザー数の増加では、ふたつの方法でアクセス可能ユーザー数の増加を図っている。ひとつは、広く普及した Windows の上や下に介入し、すでに Windows が保有しているアクセス可能ユーザーを奪い取っている。また、もうひとつの方法として、開発業者を含む補完業者を積極的に支援し、Java の場合は多くのアプリケーションを、VMware の場合は多くの既存 OS との隣接や目的別の専用 OS を開発してもらうことで、自らが魅力的なプラットフォーム製品となりアクセス可能ユーザーを惹きつけている。マルチホーミングコストについては、Java も VMware もユーザーのマルチホーミングコストの削減もしくは負担増加がないことを強くアピールし、選択における障壁を低下させている。隣接対象プラットフォーム製品の多数選定では、両事例ともオープン性を堅持することで、多くの隣接階層上のプラットフ

ォーム製品と隣接関係を維持している。これらのことにより、それまで Windows がもっていたドミナント・プラットフォーム製品の地位を脅かし、自らが大きな支配力をもつ可能性を高めている。

（２） 相違点

持続的収益確保モデルの遂行に関しては、Java と VMware には相違点が存在する。Java においてサン社は別階層に自社 OS を有償で提供し、収益階層を保有している。しかし、Java のオープン性は、Java の普及が自社 OS の直接の売り上げ増には結びつけられない状況にならざるを得ない。これによりサン社自体は収益モデルに課題を残すこととなる。VMware は仮想化ソフトウェア階層以外に収益階層をソフトウェア・レイヤースタック内にもたないため、ヴイエムウェア社は VMware で築いたインストールド・ベース顧客を囲い込むため、レイヤースタック外での製品やサービスの提供に経営の重点をシフトさせてきた。

（３） 小括

これまでの整理で、Java はアプリケーション階層と OS 階層間に介入、VMware は OS 階層とハードウェア（BIOS）階層間に介入という違いはあるが、両事例ともアクセス可能ユーザー数の増加、マルチホーミングコストの低減、隣接対象プラットフォーム製品の多数選定の３つの項目に関して共通の動きをおこなっていることが分かった。また、持続的収益確保モデルの遂行に関しては、Java と VMware には相違点が存在し、階層介入型プラットフォーム製品は提供者の収益確保に関して課題があることも理解された。

第５節 おわりに

本章では、階層介入の戦略を具体的に理解し、先行研究レビュによる仮説的推論を確認すること、ならびにプラットフォーム製品提供者の操作項目における理解と共通点・相違点を明らかにするために、ふたつの事例研究をおこなった。ひとつめのプラ

ットフォーム製品による階層介入の事例として、コンピュータ言語の Java を、ふたつめの事例として VMware を取り上げた。それぞれの事例において、多面的に事実内容の説明をおこなった。その上で、事例として選択した理由について論じ、前述したプラットフォーム製品提供者の４つの操作項目の観点で確認をおこなった。加えて、階層介入前と後に関して、階層間関係とポジションの変化について考察をおこなった。

第８章

仮説的推論の確認と戦略上の示唆の導出

第１節　はじめに

本章では、先行研究レビュによって推論した後発プラットフォーム製品のドミナント化メカニズムを、第２節にて Java と VMware の事例によって確認する。その上で、第３節では、新たな戦略に関する示唆の導出をおこなう。そこでは、仮説的推論の確認にて十分に論じられていない理論的かつ実践的な示唆の提起をおこなう。

第２節　仮説的推論の確認

Java と VMware の場合において、仮説的推論の確認をおこなう。

（１）　Java 事例によるドミナント化のメカニズムの確認

提起された後発プラットフォーム製品のプラットフォーム製品提供者の４つの操作項目の観点で、Java 事例においてドミナント化のメカニズムを確認する（表８－１）。

表８－１：Java 事例での分析

操作項目	Java 事例での現象	ドミナント化要因	ドミナント化の可能性
アクセス可能ユーザー数の増加	開発者コミュニティの支援。パートナーとの共同開発・ライ	階層間ネットワーク効	高める

	センス契約。 先発ドミナント・プラットフォーム製品（Windows）との隣接。	果の効用力	
マルチホーミングコストの低減	Java の無償ダウンロード。		高める
隣接対象プラットフォーム製品の多数選定	WORA（オープン性）の堅持。	ブリッジングの影響力	高める
持続的収益確保モデルの遂行	（プラットフォーム製品提供者としてのサン社は）不履行。	排除に対する抵抗力	関係せず

Java の事例では、４つの操作項目の観点からの分析において、それぞれの項目がドミナント化要因に影響を与え、３つのドミナント化要因の高まりによってドミナント化の可能性が高まることが概ね確認された。

ただし、持続的収益確保モデルの遂行においては、プラットフォーム製品提供者としてのサン社は不履行である。にもかかわらず、Java がドミナント・プラットフォーム製品であるひとつの理由として、Java は 1998 年より JCP というオラクル社、IBM 社、HP 社、Nokia 社、インテル社など多くのパートナー企業による共同管理による運営に切り替えられたため、サン社１社のみの「持続的収益確保モデルの遂行」の履行・不履行の影響は極めて小さいためであると考えられる。尚、JCP はサン社合併後は現在、オラクル社のサポートによって運営されている。

ここで、更にメカニズムの蓋然性を確認するため、不成功事例としてマイクロソフト社が Java に対抗してリリースした Active-X について、４つの操作項目の観点で、分析する（表８－２）。

表８－２：Active-X 事例での分析

操作項目	Active-X 事例での現象	ドミナント化要因	ドミナント化の可能性
アクセス可能ユーザー数の増加	開発者コミュニティの支援。 パートナーとの共同開発・ライセンス契約。 セキュリティの脆弱性によるユーザーならびに開発者離れ。	階層間ネットワーク効果の効用力	高められない
マルチホーミング	インターネットでの無償提供。		高める

コストの低減			
隣接対象プラットフォーム製品の多数選定	OS は Windows に限定される。	ブリッジングの影響力	高められない
持続的収益確保モデルの遂行	OS でドミナントの地位にあるマイクロソフト社が提供者。	排除に対する抵抗力	高める

Active-X の場合、ドミナント化のメカニズムにおける、マルチホーミングコストの低減と持続的収益確保モデルの遂行の履行によって、ドミナント化の可能性を高めることとなる。一方、アクセスユーザー数の増加と隣接対象プラットフォーム製品の多数選定に不履行の要素があるために、結果としてドミナント化は高まらず、Java の対抗製品として比肩するシェアをもつに至らなかった。

このように操作項目の不履行により、一部のドミナント化の要因を高められず、結果としてプラットフォーム製品のドミナント化の可能性を高められないという状況に鑑みても、ドミナント化のメカニズムは是認されると考えられる[461]。

（2） VMware 事例によるドミナント化のメカニズムの確認

提起された後発プラットフォーム製品のプラットフォーム製品提供者の４つの操作項目の観点で、VMware 事例においてドミナント化のメカニズムを確認する（表８-３）。

表８-３：VMware 事例での分析

操作項目	VMware 事例での現象	ドミナント化要因	ドミナント化の可能性
アクセス可能ユーザー数の増加	開発者コミュニティの支援。 パートナーとの共同開発・ライセンス契約。 先発ドミナント・プラットフォーム製品（Windows）との隣接。	階層間ネットワーク効果の効用力	高める
マルチホーミング	ひとつのハードウェア上での		高める

461 ただし、ひとつの事例での分析であり、他の要素も含めた厳密な分析や検証もおこなっていないため、普遍的な結論として認めたものではない。本論文では、あくまでメカニズムを肯定する一例として挙げ、検証等については今後の課題としたい。

コストの低減	複数 OS の稼働。 複雑な管理コストの低減。 VMware ESXi 無償版の提供。		
隣接対象プラットフォーム製品の多数選定	オープン性の堅持。	ブリッジングの影響力	高める
持続的収益確保モデルの遂行	インストールド・ベース顧客に対し、レイヤースタック外での収益確保。	排除に対する抵抗力	高める

VMware の事例において、４つの操作項目の観点からの分析においてそれぞれの項目がドミナント化要因に影響を与え、３つのドミナント化要因の高まりによってドミナント化の可能性が高まることが確認された。

ここで、更にメカニズムの蓋然性を確認するため、別の事例として、マイクロソフト社が VMware に対抗してリリースした Hyper-V について、４つの操作項目の観点で分析する(表８-４)。

表８-４ ：Hyper-V 事例での分析

操作項目	Hyper-V 事例での確認	ドミナント化要因	ドミナント化の可能性
アクセス可能ユーザー数の増加	開発者コミュニティの支援。 パートナーとの共同開発・ライセンス契約。 先発ドミナント・プラットフォーム製品（Windows）とのバンドル。	階層間ネットワーク効果の効用力	高める
マルチホーミングコストの低減	既存の有償 OS に付随して無償で提供するため追加費用なし。		高める
隣接対象プラットフォーム製品の多数選定	OS は Windows と一部の Linux に限定される。	ブリッジングの影響力	高められない
持続的収益確保モデルの遂行	OS でドミナントの地位であるマイクロソフト社が提供者。	排除に対する抵抗力	高める

Hyper-V の場合、ドミナント化のメカニズムにおける、アクセス可能ユーザー数の増加、マルチホーミングコストの低減、持続的収益確保モデルの遂行の履行は、ドミ

ナント化の可能性を高めることとなる。一方、隣接対象プラットフォーム製品の多数選定に不履行の要素があるために、ドミナント化の可能性の高まりは限定的である。しかし、最も多く普及しているサーバーOS は現状では Windows であり、サポートが必要な OS は Windows と一部の Linux で十分であると考えるユーザーも多い。2012 年頃からの Hyper-V の四半期ベースでの出荷台数での VMware の逆転は、このようなことが起因しているのかも知れない。

このように操作項目の不履行により、一部のドミナント化の要因を高められず、結果としてプラットフォーム製品のドミナント化の可能性を高められないという状況に鑑みても、ドミナント化のメカニズムは是認されると考えられる[462]。

（3） 小括

後発プラットフォーム製品としてのドミナント化のメカニズムに関し、Java と VMware はその操作項目とドミナント化要因の関連において同様の関係があり、また３つのドミナント化要因の高まりがプラットフォーム製品のドミナント化の可能性を高めていることが概ね確認された。加えて、マイクロソフト社の Active-X と Hyper-V の事例の分析により、ドミナント化のメカニズムにおける操作項目の不履行が、一部のドミナント化要因を高められず、ひいてはドミナント化の可能性を高めることができない状況を指摘した。この点からもメカニズムは肯定されていると考えられる。

しかし同時に、本論文での問題意識である「後発プラットフォーム製品はドミナント・プラットフォーム製品になり得るか」そして「どうして階層介入型プラットフォーム製品はドミナント・プラットフォーム製品になり得たのか」という問いに応えるべく、後発の非階層介入型（階層介入型ではない）プラットフォーム製品と階層介入型との違いから生じる戦略上の示唆を明らかにする。よって、次節ではその点において議論をおこなうこととする。

---

462 同上。

第３節　新たな戦略に関する示唆の導出

Java の事例と VMware の事例の操作項目の観点での分析ならびに確認から、導出される階層介入型プラットフォーム製品特有の戦略に関する仮説は以下である。ちなみに仮説 3-1 は項目①から、仮説 3-2 は項目②から、仮説 3-3 と仮説 3-4 は項目③から、仮説 3-5 と仮説 3-6 は項目④から導出された。仮説中の（＋）は促進、（－）は抑制を示す。

○階層介入型プラットフォーム製品の戦略上の効果（具体的には『既存（先発）の隣接プラットフォームの支配力を介入によって減じる効果』）における仮説

仮説 3-1：階層介入型プラットフォーム製品は、非階層介入型プラットフォーム製品と比較して、隣接（n、n+2）階層のプラットフォーム製品のコモディティ化を誘発し易い

仮説 3-2：階層介入型プラットフォーム製品は、非階層介入型プラットフォーム製品と比較して、既存の隣接（n、n+2）階層のプラットフォーム製品のレイヤースタック内での延命を助長し易い

仮説 3-3：階層介入型プラットフォーム製品は、非階層介入型プラットフォーム製品と比較して、プラットフォーム包囲に対して、それ自体が包囲されにくい防衛的役割をもち易い

仮説 3-4：階層介入型プラットフォーム製品は、非階層介入型プラットフォーム製品と比較して、上下階層セットの垂直統合やバンドルを分断し、既存の隣接プラットフォーム製品の収益モデルにダメージを与え易い

○階層介入型プラットフォーム製品の戦略上の課題（具体的には『階層介入型プラットフォーム製品の普及と収益確保のトレードオフに関する課題』）における仮説

仮説 3-5：階層介入型プラットフォーム製品は普及を優先する（＋）と提供者の収益確保が困難となる（－）

仮説 3-6:階層介入型プラットフォーム製品は提供者の収益確保を優先する（＋）と普及が困難となる（－）

（1） 仮説 3-1

仮説 3-1 については、いやおう無しに垂直統合やバンドルされたプラットフォーム製品を、ユーザーが選択せざるを得ない状況がある場合でも、介入階層によるアクセス可能ユーザーの流動化のおかげで、同一階層レベルの「他のどのプラットフォーム製品を選択しても同様の便益を享受できる」こととなり、プラットフォーム製品選択における限定が弱まる。言い換えれば、介入階層の隣接（介入階層をｎ＋１レベルとした上で、ｎとｎ＋２レベルにある）階層の、ユーザーのプラットフォーム製品の選択必然性は弱まり、それらのコモディティ化を促す。

ちなみに既に多くのインストールド・ベース顧客をもつ Windows にとって、自らのユーザー・グループからのプラットフォーム製品選択の必然性が弱められてしまうことは、現状のドミナントの地位を危うくする原因となる。Java や VMware の介入階層自体がレイヤースタック内でのドミナントとなる可能性を有し、それまで Windows がもっていたドミナントの地位を脅かす可能性を高めることとなった。

この点について Rohlfs （2001）は、マイクロソフト社の市場支配力の低下を Java 介入のケースで、「もしアプリケーション・プログラムが主に Java で書かれるようになるならば、マイクロソフト社もインテル社もその市場支配の多くを失うことになるだろう。アプリケーション・ソフトの利用可能性に絡むバンドワゴンの便益を受けるために、それらふたつの会社の製品をもはや購入する必要がなくなるからだ。利用者は Java が動作するマイクロプロセッサとオペレーティング・システムであればどれを使おうと同じバンドワゴンの便益を受けることができるようになるからだ。」と指摘している[463]。

この点について末松（2002）は「Java はあらゆる OS の上で動作し、ネットワークに関係する機能を提供する。ここで重要なのは、あらゆる OS 上で機能する点で、システムは Windows でなくてもよいことになる。つまりプラットフォームとして強大な地位を築いてきた Windows を One of them にしてしまい、サン社自らがプラットフォームの地位を奪い取ろうという戦略であった。（後略）」と説明する[464]。

---

463 Rohlfs（2001） ，（邦訳）p. 163 参照。
464 末松（2002），p. 68 参照。

また Foley（2008）は、VMware を含む仮想化ソフトの普及について、「マイクロソフト社のみならず、ソフトウェア市場のすべての企業に対して、事業の根本的な見直しを迫る可能性がある最大の激震の震源は仮想化である。（中略）仮想化は OS の選択が参入障壁になる場面も減らす。Windows ユーザーの中には、Windows でなければ動作しないアプリケーションを使わなければならないために、MacOS や Linux に切り替えることを躊躇している人々がいるが、そういう懸念はなくなる。」と説明している[465]。

（２） 仮説 3-2

仮説 3-2 については、低く設定されたマルチホーミングコストのおかげで、特定のプラットフォーム製品の勝者総どり（WTA：Winner-Take-All）の傾向が抑制され、その隣接階層のプラットフォーム製品の延命を助長することが可能となる。具体的には、Java の事例では、隣接階層（n、n+2 レベル）にある商用 OS の Solaris の延命が助長され、VMware の事例では、本来なら新バージョンに移行を余儀なくされる古いバージョンの OS 上の業務用アプリケーションを、引き続き利用することを可能にする。それにより古いバージョンの OS の延命を図ることができるようになった。Windows のようなドミナント・プラットフォーム製品が、繰り返し階層バンドルの施策を上位補完業者に対しておこなうと、プレイヤーの多様性の減少が起こり自然に独占状態に近付いていく傾向をもっている。Java や VMware にはこの動きを阻止する働きがあった。

（３）仮説 3-3

仮説 3-3 については、Java の事例では下位階層に位置する Windows からのプラットフォーム包囲、VMware の事例では上位階層に位置する Windows のプラットフォーム包囲に対する防衛的役割を担っているといえる。言い換えれば、両プラットフォーム製品とも包囲困難な階層として存在し得る。

---

465 Foley（2008），（邦訳）p. 308-309 参照。

サン社にとって Java の投入は Windows の台頭に対する防衛の意味があった。2005 年 5 月の IDG News Service のインタビュの中で、スコット・マクニーリ（当時の会長兼 CEO）は、Java の恩恵を、「もしも 10 年前に Java がなかったら、サン社は今どうなっているだろうか」との問いかけに対し、「すべてが Windows になり、われわれは終わっているだろう。開発者が Java Web サービスを書いていないのなら.NET 向けにサービスを書いていることになり.NET 向けに書くということは、Windows 向けに書くことになる。Windows 向けに書くのであれば、サン社の機器向けには書かないということだ。」と説明している[466]。

（4）仮説 3-4

仮説 3-4 については、隣接対象に選定されたプラットフォーム製品は、介入により介入階層を挟んだ上下に隔てられる。例えば Java の事例の場合、Windows とワード・エクセルならびに C#言語[467]によって開発された業務用アプリケーションのバンドル状態が、また VMware の事例の場合、Windows とインテル社製チップの連合（ウィンテル連合）の関係が分断される。これにより、介入プラットフォーム製品が上下階層セットの垂直統合やバンドルを分断し、既存の隣接プラットフォーム製品間での収益モデルにダメージを与えることができる。また競合プラットフォーム製品の収益力の弱体化と、自社プラットフォーム製品の収益力強化を目論むことが可能である。

Rohlfs (2001)は、この両者のバンドル状態に対する Java の介入について、マイクロソフト社への影響を、「マイクロソフト社は Java でアプリケーション・プログラムを書こうとするインセンティブを明らかに持っていない。（中略）ワード・エクセル・アクセスが産業標準であり、それだけで（マイクロソフト社は）大きなバンドワゴンを得ているのだから、（マイクロソフト社にとって）Java を通じてオペレーティング・システムを相互連結させる可能性は非常に限定されるだろう。」と説明している[468]。

---

466 出所：http://www.itmedia.co.jp/enterprise/articles/0505/06/news023.html 2013/9/14
467 マイクロソフト社が 2000 年に提唱した.NET のアプリケーションの開発に使用する言語。
468 Rohlfs（2001），（邦訳）p.163 参照。

Foley(2008)は、VMwareについて、「(前略)仮想化に対するアプローチの中には(現在の収益の流れを寸断してしまうという意味で)危険なものがある。ユーザーは、仮想化によって比較的簡単に他の環境への切り替えを検討できるようになるのである。(中略)ハイパーバイザーは直接ハードウェアの上でゲストを実行し、ホストOSを必要としない。」と指摘する[469]。

(5)仮説3-5と仮説3-6

仮説3-5と仮説3-6については、階層介入型プラットフォーム製品は普及と提供者の収益確保に関してトレードオフが発生する。Javaの事例では補完業者とのコミットメントの故に、VMwareの事例ではWindowsによるバンドル攻撃の故に、プラットフォーム製品の拡大と収益の確保にジレンマが生じている。言い換えれば、普及を目指せば収益の確保は難しく、収益を確保しようとすると普及に支障が生じるという状態である。JavaもVMwareもこのジレンマを克服するため、サン社はレイヤースタック内の別階層に自社OSを有償で提供し、収益のためのプラットフォーム製品を提供している。一方、VMwareは仮想化ソフトウェア製品以外に主となる収益プラットフォーム製品がレイヤースタック内には存在しない。よって、インストールド・ベース顧客を囲い込むため、買収によるハードウェア階層の垂直統合管理ツールの充実、プライベート・クラウドへの誘導など、レイヤースタック外での収益確保を進めている。

(6)小括

先行研究レビュから得られた後発プラットフォーム製品の仮説を補足する戦略上の知見として、仮説3-1 、仮説3-2、仮説3-3、仮説3-4、仮説3-5、仮説3-6を提起し、隣接プラットフォーム製品のコモディティ化や延命、既存バンドル状態の分断や普及と収益のトレードオフについての知見を提示した。それぞれ仮説3-1 は項目①から、仮説3-2は項目②から、仮説3-3、仮説3-4は項目③から、仮説3-5、仮説3-6は項目④から導出された。これまでを、表にすると以下となる(表8-5)。

---

469 Foley (2008), (邦訳) p.310参照。

表８－５：両事例での戦略上の示唆

操作項目	両事例において現れる効果や現象で階層介入型 PF 製品特有のもの	Java	VMware
項目①：アクセス可能ユーザー数の増加（＝アクセス可能ユーザーの流動化）	隣接階層の PF 製品のコモディティ化を誘発する。＜仮説３－１＞	隣接階層にある Windows を他の同一レベル PF 製品のなかのひとつ（one of them）としてしまう。	隣接階層にある Windows を他の同一レベル PF 製品のなかのひとつ（one of them）としてしまう。
項目②：マルチホーミングコストの低減	既存のレイヤースタック内での他の PF 製品の延命を助長する。言い換えれば、一社独占の傾向を抑制し、隣接階層の PF 製品の延命を助長する。＜仮説３－２＞	隣接階層の商用 OS である Solaris の延命。	隣接階層の古いバージョンの OS の延命。
項目③：隣接対象 PF 製品の多数選定（＝入れ子構造の創出）	階層介入型 PF 製品はそれ自体が PF 包囲されにくい防衛的役割を担う。＜仮説３－３＞ 階層介入の場合は、オープンなインターフェイスをもつために、階層間の上下階層をセットにした PF 製品の垂直統合やバンドルを分断する。＜仮説３－４＞	Windows（下位階層）からの包囲の脅威に対する防衛的役割。 Windows と C# 言語で開発された業務用アプリのバンドル状態の分断。	Windows（上位階層）からの包囲の脅威に対する防衛的役割。 Win-Tel のバンドル状態の分断。
項目④：持続的収益確保モデルの遂行	階層介入型 PF 製品は普及と提供者の収益確保に関してトレードオフが発生する。 ＜仮説３－５、仮説３－６＞	Java 自体の無償での提供。	VMwareESXi という無償版の普及に移行。

第４節　おわりに

本章では、後発プラットフォーム製品のドミナント化のメカニズムを Java と VMware の事例、ならびに不成功や別の事例として Active-X や Hyper-V によって確認した。その上で、階層介入戦略特有の戦略上の示唆として、プラットフォーム製品提供者の４つの操作項目での確認を通じ、階層介入型プラットフォーム製品の戦略上の効果における４つの仮説と、階層介入型プラットフォーム製品の戦略上の課題におけるひとつの仮説を提起した。そこでは、隣接プラットフォーム製品のコモディティ化や延命、防衛的役割、既存バンドル状態の分断や普及と収益のトレードオフについての知見が得られた。

終章

―階層介入戦略と知見の理論的含意―

第１節　ビジネス機会と階層介入戦略の有効性

今後、クラウドの普及による階層構造の変化や、新たなソフトウェア製品の出現などによって、既存の階層間関係やドミナント状況が変化する可能性がある。このことは、介入ソフトウェア製品を提供するプラットフォーム製品提供者にとって、大きなビジネス機会となる。本論文の Java や VMware の事例の紹介のなかで説明したことであるが、Java も VMware も誕生当初からビジネス上の戦略的な活用を目的として開発されたわけではない。シリコンバレーのテクノロジー・カンパニーが偶然に生み出した革新的な技術を、経営的手腕をもつ経営幹部によって競合戦略上のエポック・メイキングとして活用することでドミナント化につながってきたといえる。Java や VMware に次ぐドミナントを目論む新たな介入ソフトウェア製品が出現する可能性が存在する。

第２節　階層介入戦略の適用可能性

ここまで介入階層のドミナント化について考えてきたが、本当にこれらの要因が実際の介入階層のドミナント化とどの程度関係していたかについては、本論文で厳密に検証したわけではない。それについては今後の課題[470]となるであろう。しかし、本論

---

470 更なる検討項目として、以下も考慮する必要がある。そのひとつとして、根来・加藤（2008）が指摘しているアクセス価値の考慮である。本論文では、アクセス可能ユーザーの個々の価値（アクセス価値）が介入階層のドミナント化において影響するという理論を論じていない。つまり、ひとつのレイヤースタック内では、そのなかで価値の高いアクセス可能ユーザーの存在の有無にかか

文が示唆することは、決してアプリケーション階層と OS 階層間、OS 階層間とハードウェア（BIOS）階層間という特定の階層間でしか成立しない議論ではないと考える。例えば、ソフトウェア・レイヤースタック内のアプリケーション階層や OS 階層を含むストレージ階層、データベース階層、ネットワーク階層などの隣接階層間や、階層を形成する他のソフトウェア・レイヤースタック（例えばスマートフォンなどの携帯型デバイス等）の隣接階層間においても、このような介入階層のドミナント化が誘発される可能性が十分ある。

加えて、本論文での分析により、Java と VMware は両事例ともプラットフォーム製品の普及と収益の確保において、ジレンマを有していることが新たな知見として示唆された。階層介入型プラットフォーム製品は、その性質上の問題として上記ジレンマを有している、ということに関わる精緻な仮説の構築と検証を今後の研究の方向性としたい。

第３節　日本のソフトウェア産業とドミナント化の機会

ソフトウェア産業ならびにネットワーク・コンピューティング業界においては、米国や欧州企業がそのビジネスのイニシアティヴを占有している。日本は元来、半導体や自動車や電機などの「ものつくり」を強みとして国際市場のなかでそのプレゼンスを示してきた。しかし残念ながらソフトウェアやサービスの国際的市場ではアップル社、マイクロソフト社、オラクル社、SAP 社、グーグル社、ヤフー社、アマゾン社など外国勢の独断場である。例えばマイクロソフト社をはじめオラクル社や SAP 社のような業務用ソフトウェア、またアップル社やグーグル社などのスマートフォン携帯端末に関連するソフトウェア、アマゾン社のキンドルなどの電子書籍の台頭がある。

わらず、ドミナント階層が存在するという立場をとっている。しかし実際は、レイヤースタック同士を比較してみると、アクセス価値の高いユーザー数の多い方が、プラットフォーム製品としての魅力がより高いという理由で、ドミナントの地位を築くことが現実にはある（詳しくは、付録 C を参照）。例えば、ドミナントのなかのドミナント的な存在や根来・加藤(2010)にて取り上げる WTA：Winner-Take-All のような状態がこれに該当する。また、レイヤースタック内での階層においては、ひとつのドミナント階層と多数の非ドミナント階層が混在し、非ドミナント階層はコモディティ化する階層となる視点で本論文では論じている。しかし、ドミナント階層はレイヤースタック内にひとつしか存在し得ないかについては、異論もあると思われる。例えば、トップドミナントとサブドミナントというような状態やドミナント化移行過渡期がこれに該当する。このように本論文でのドミナント化の捉え方には、曖昧さが残る点もあり、今後の課題である。

これに対し、日本のソフトウェア産業は未だソフトウェア受託開発の労働集約型ビジネスの価格競争に汲々としている。また日本発のイノベーションともてはやされた i-mode も今ではガラパゴスと称される独自サービスの失敗例として扱われている現状である。本研究にはソフトウェア・プラットフォーム製品戦略の探究が、日本のソフトウェア産業の国際的競争力を高め、グローバル規模のイノベーションを誘発できる戦略の策定ならびに学術的貢献の一部を担うことができればとの思いがある。

付録 A

プラットフォーム製品統合の事例

第１節　はじめに

単一レベル階層のプラットフォーム製品戦略の事例としてプラットフォーム製品統合を詳説する。具体的には、1980 年代後半に Unix 市場で起こったふたつの陣営の対立（俗称：Unix 戦争）の事例を取り上げる。説明は、加藤（2006）の先行研究を基にする[471]。

第２節　Unix 標準化の背景と時系列整理

1980 年代のコンピュータ業界は、自社コア製品（技術）を業界の標準に据えようとするプラットフォーム戦略によって、複合製品であるコンピュータのレイヤースタックでの CPU 階層や OS 階層などの標準化にネットワーク効果のメカニズムを利用して自社のシェア拡大を図り、業界での主導権をにぎろうとするベンダー間の思惑のぶつかり合いの歴史であった。特に CPU・OS・アプリケーションの３階層での互換性や相互運用性の確立は、ネットワーク効果を利用した拡販戦略としてプラットフォーム・リーダーにとっての戦略上の重要なレバー（施策）であった。本論文では 1980 年代後半に起こった Unix の統合による標準化における２大陣営の対立の時系列整理を通じて、単一レベル階層のプラットフォーム製品戦略を理解する。

---

471 時系列整理の記述に関しては、主としてホール・バリー（1991）、根本・松岡（1992）ならびにサウスウイック（2000）からの引用による。

（1）Unix 標準化の背景と時系列整理

Unix 普及以前のシステムは IT ベンダーの顧客の囲い込みによるプロプライエタリのメインフレームやミニコンのシステムが主流であった。1980 年代後半からの Unix の普及によるシステムのオープン化の流れの中で、Unix リーダーはいかに自らが水平的なプラットフォームの業界標準の主導権を握るかという覇権争いに明け暮れるようになる。その中で 1980 年代後半に起こった UI（Unix インターナショナル）と OSF（オープン・ソフトウェア・ファンデーション）の Unix 標準化の２大陣営対立に着目し、それが起こった背景と時系列整理をおこなう。

1）Unix の誕生

Unix は現在でこそオープンシステムを構築するための中核 OS として IBM 社、HP 社、サン社、NEC 社、富士通など世界のほとんどのコンピュータ・メーカーに採用されているが、AT&T 社のベル研究所から最初に登場した 1969 年当時はもちろん 1980 年代初めでも業界標準 OS と言えるようなものではなかった[472]。それまでの OS は特定のハードウェアに固有の機械語で書かれていて、他のハードに移植することは困難なものであった。1970 年代、最初は Unix もアセンブリ言語で書かれていたが後にプログラムを「C 言語」に書き換えたことで異なるプラットフォームに対する移植性が高くなり、可読性が高いため機能の改変をおこなうことが比較的容易となった[473]。

最初に Unix を開発したのは 1960 年代後半ベル研究所のケン・トンプソン（Ken Thompson）とデニス・リッチー（Dennis Ritchie）で、DEC 社のミニコンピュータ上での使用に向けて開発された[474]。AT&T 社は 1974 年からこの Unix を世界の大学や研究機関に無料で配布していく。当時は米国の独占禁止法によって AT&T 社は通信分野以外への事業進出はできない事情があった[475]。1982 年には裁判所による AT&T 社の分割命令により、AT&T 社は７つの地方電話会社と、ひとつの中央電話会社に分割される[476]。中央電話会社である新生 AT&T 社はコンピュータ事業への参入が許され、1983 年

472 根本・松岡（1992）, p. 80 参照。
473 同上。
474 同上。
475 根本・松岡（1992）, p. 81 参照。
476 同上。

にはUnix のライセンス業務をベル研究所からAT&T 社本体に移管し「Unix システムV（ファイブ）」を発表する[477]。

Unix は“ユニークで統一”されたOS の概念で生まれたにもかかわらず、1980 年後半には多くのバージョンが派生し互換性を保つことができなくなってしまった（表10-1）。このようにUnix のバージョンが増えてしまった理由は大きくふたつある。ひとつはメーカーがゼロからOS を開発するコストを抑えるため、AT&T 社からUnix のライセンスを取得し、自社の商用OS として変更を加える方法で取り入れることを選択したこと[478]。もうひとつはAT&T 社がソースコードの使用権を許可したが、バイナリ・コード(CPU ごとに異なるプログラム・コード)を許可しなかったことである[479]。

表10-1：1980 年代のUnix バージョン[480]

Unix バージョン	OS の専用・非専用区分	提供企業	説明	備考
システムV	非専用	AT&T 社	1983 年にAT&T 社が発表する。2 年後にはシステムV のスペックを含んだマニュアルであるSVID（システムV インターフェイス仕様書）を発表する。	
バークレー4.3BSD	非専用	サン社	SunOS の基本となるもので、生みの親のAT&T 社のオリジナルに様々な改良を加えている。特に科学分野・技術分野のユーザーに好まれる傾向が強い。システムV とともに当時の２大Unix 業界標準である。	
XENIX	非専用	マイクロソフト社	マイクロソフト社のUnix でAT&T 社のSVID に準拠。マイクロソフト社はこれとは別にIBM－PC 用とPS/2 用にそれぞれMS-DOS とOS/2 を開発しているが、これらとXENIX	XENIX は後にSCO へ売却される

477 根本・松岡（1992），p. 82 参照。
478 ホール・バリー（1991），p. 153 参照。
479 同上。
480 ホール・バリー（1991），p. 155-158 をもとに筆者作成。

			は多少の互換性を持つ。	
AIX	専用（プロプライエタリ）	IBM 社	自社提供のコンピュータに合わせて様々なバージョンの AIX を提供している。IBM 社は PC/AT 用に XENIX も販売する。	
HP/UX	専用（プロプライエタリ）	HP 社	システム V と同時に動き、バークレー版の特徴も持つ。特に商用アプリケーションに役立つ機能（リアルタイム処理、ロック機能）をもつ。	
A/UX	専用（プロプライエタリ）	アップル社	アップル社は他社にくらべかなり後発で参入した。マッキントシュⅡ用の Unix を発表した。	
ドメイン/IX	専用（プロプライエタリ）	アポロ社	システム V とバークレー版の両方が入っており、ユーザーは使用するアプリケーションにあわせて選択可能。	アポロ社は 1989 年 HP 社によって買収
ウルトリクス	専用（プロプライエタリ）	DEC 社	バークレー版とも互換性があり、システム V にも適合する。	DEC 社は 1998 年コンパック社（現在の HP 社）によって買収

2）Unix の２大陣営対立

前述のように Unix は 1960 年代後半に AT&T 社のベル研究所で開発され、AT&T 社の寛大なライセンス供与のおかげで大学や研究所で広く利用される[481]。しかしながら AT&T 社自身の Unix とは別に、当時サン社の共同創業者でテクノロジー部門責任者の一人であるビル・ジョイが UC バークレー在学中に開発したバークレー版 Unix が、もうひとつの大きな系統であった[482]。そして当時 Unix にはこのふたつの系統をルーツとする多くの変種が存在し、市場ではどちらを選べばいいのかと混乱が生じていた[483]。

このような経緯のなか 1987 年 10 月サン社のスコット・マクニーリと AT&T 社のコンピュータ製造事業部の責任者であるヴィットリオ・カッソーニ（Vittorio Cassoni）は、両社の Unix システムの統一版に共同で取り組むことに合意する契約に署名する

481 サウスウイック（2000）, p. 110 参照。
482 ホール・バリー（1991）, p. 150 参照。
483 サウスウイック（2000）, p. 110 参照。

[484]。後の 1990 年にサン社の取締役会を去ったボブ・カヴナー（Bob Kavner）は、「Unix でサン社と AT&T 社が協力しあって単一のバージョンを完成し、AT&T 社のブランドを高めると同時にサン社の市場浸透力を促そうとする考えであった」と語っている[485]。

しかしながらサン社と AT&T 社にとっての想定外であったのは、Unix 業界の他の競合である HP 社や IBM 社や DEC 社の強い反発である。ジオパートナーズ・リサーチ社（GeoPartners Research Inc.）の社長ジム・ムーア（James F. Moore）はこういったサン社と AT&T 社の Unix の動きが、低価格で大量販売可能な Unix の統一版の開発を中心に位置付けることで、Unix 市場支配目的のために業界の価格の高止まりを打破しようとしているかのような印象を与えていると指摘した[486]。

このような動きに対抗する同盟が HP 社、DEC 社、IBM 社を中心に結成される。OSF（オープン・ソフトウェア・ファンデーション）と命名されたグループには 30 社以上のベンダーとコンサルティング会社が参加した[487]。一方、当初の発表どおりサン社と AT&T 社はインテル社、東芝、ユニシス社、モトローラ社、富士通その他十数社と Unix インターナショナルを設立する（表 10-2）[488]。

表 10-2: UI と OSF 参加企業[489]

	UNIX インターナショナル（UI）	オープン・ソフトウェア・ファンデーション（OSF）
海外	AT&T、サン・マイクロシステムズ、ユニシス、ゼロックス、ICL、アムダール、オリベッティ、NCR、コントロール・データ、インテル、モトローラ、プライムコンピュータ、データゼネラル、ネクストコンピュータ　など	IBM、HP、DEC、アポロコンピュータ、シーメンス、ニクスドルフ、ブル、フィリップス、シリコングラフィックス、データゼネラル　など
国内	富士通、NEC、東芝、沖電気工業、富士ゼロックス、リコー、ソニー、オムロン、キャノン　など	日立製作所、三菱電機、ソニー、キャノン、オムロン　など

484 同上。
485 サウスウイック（2000）, p. 111 参照。
486 同上。
487 サウスウイック（2000）, p. 112 参照。
488 同上。
489 根本・松岡（1992）, p. 95 参照。

1988 年の OSF の結成発表後、対立する両陣営の間では和解にむけた交渉がおこなわれた。OSF 側からの提案内容は IBM 社の Unix である「AIX[490]」と「システム V」との融合を提案、同時に AT&T 社とサン社に OSF への加入を呼びかけた[491]。AT&T 社にとっては各 OS メーカーに Unix のライセンスを与えてきたという関係もあり、“お客様”である IBM 社、DEC 社、HP 社などメーカーとの対立は利益にならないという観点から、サン社に OSF への加入を進めた[492]。また両陣営が分裂して“統合 Unix”をつくっても枝分かれした状態の解消にはならないため、日本の富士通や日立など別々の陣営に所属することとなったメーカーからも分裂状態の解消を望む声が上がっていた[493]。サン社の社内では OSF へ参加すべきか否か議論がたたかわされたという。しかし AT&T 社とサン社連合の出した最終的な結論は IBM 社の「AIX」の存在を認めないというものであった（表 10-3）[494]。

表 10-3:UI 結成までの時系列整理[495]

	UI 連合(サン社と AT&T 連合）の動き	競合(IBM 社, DEC 社, HP 社など）の動き	業界の反応
1985 年 9 月	サン社と AT&T 社は３段階計画で Unix を統合してゆくと発表。	IBM 社や DEC 社は当初は危機感が薄かった。	Unix はまだ一部の顧客が利用しているにすぎない。サン社の企業規模も小さい。
1987 年 10 月	サン社と AT&T 社が新たな提携に調印。---サン社の開発した RISC プロセッサ「SPARC」搭載コンピュータ向けの統合 Unix を共同開発する。	Unix に関して中立的立場の AT&T 社のサン社との蜜月関係に対し不満を表明。	Unix をサン社が中心に市場を開拓してきたので無視できない OS となってきている。
1988 年 1 月	AT&T 社がサン社に資本参加する。---- AT&T 社がサンの全株式の 20%を取得。	大きな危機感をもつ。-----自社独自 OS 以外にオプションで Unix を提供するようにな	AT&T 社とサン社は本気で Unix の覇権を握ろうとしていると業界では判断しはじめ

490 AIX とは IBM 社の開発した Unix 系 OS の名称。AIX は IBM 社のメインフレームなどとの連係に優れているとされる。
491 根本・松岡（1992）, p. 93 参照。
492 同上。
493 同上。
494 根本・松岡（1992）, p. 94 参照。
495 根本・松岡（1992）, p. 81-95 をもとに筆者作成。

		る。	る。
1988 年 5 月		世界を代表するコンピュータ・メーカー７社が OSF を設立。 ---IBM 社、DEC 社、HP 社、アポロ社など。	
1988 年 10 月	世界のコンピュータ関連会社 17 社はサン社と AT&T 社が共同開発する「UNIX システム V4.0」を支持すると発表。---アムダール社、ユニシス社、NCR 社、インテル社、富士通、東芝など。	OSF とサン社と AT&T 社連合間で和解の話し合いがおこなわれる。	IBM 社と AT&T 社の米国巨大企業の代理戦争との見方。
1988 年 12 月	結局 IBM 社の「AIX」は認めないと結論。正式に UI を旗揚げする。		

1989 年 11 月、UI は「Unix システム V リリース 4.0」を完成させ発表する。これは「システム V」「バークレー版 Unix[496]」「XENIX[497]」を統合したものである[498]。これに対抗し OSF は 1990 年 10 月「OSF/1」を発表。OSF は IBM 社の「AIX」をカーネルにすることで進めていたが、DEC 社や HP 社の反対によって米カーネギーメロン大学の開発した「Marc(マーク）」を採用する方針に切り替えた[499]。「OSF/1」は発表の遅れや陣営内での不協和音によって搭載新機種を出せない状態が続き大きく出遅れた[500]。

一方 UI 側では 1990 年に入り AT&T 社が NCR 社の買収を発表[501]。これにともない 1991 年 6 月「Unix システム V リリース 4.0」の共同開発により、当初の目的は達成されたとして、AT&T 社とサン社は資本提携関係を解消した[502]。結果としてサン社と AT&T 社の企業文化の違いもあり、UI 結成をきっかけに両社の関係がその後の Unix 分野で発展的に構築されることはなかった。

---

496 バークレー版 Unix とは米カリフォルニア大学バークレー校の CSRG(Computer Systems Research Group）が機能拡張を行なって配布した Unix。
497 XENIX とは 1980 年に AT&T 社からライセンスを購入してマイクロソフト社が SCO とともに開発・販売していた PC 用 Unix。
498 根本・松岡（1992）, p. 95 参照。
499 根本・松岡（1992）, p. 96 参照。
500 同上。
501 同上。
502 根本・松岡（1992）, p. 97 参照。

Unix の２大陣営の対立は、その後のコンピュータ・ベンダー勢力図に大きな影響を残すことになる。この Unix 標準化の２大陣営の争いは、コンピュータ市場の大きな転換期であったとの見方がある。両派が Unix の縄張りを争っている間に、デスクトップ・コンピュータの市場支配をもくろんでいたマイクロソフト社に、市場参入の機会を与えてしまうこととなったと、当時 AT&T 社のコンピュータ製品担当社長のボブ・カヴナーは指摘している[503]。もしこの時点で Unix 業界が互いに争わず団結して統一 Unix を実現していたら、より強いネットワーク効果の働きにより、マイクロソフト社の新しい独自 OS であった WindowsNT の台頭を阻止することもできたかしれないという見方が存在するからである。

（２） 階層戦略としての位置付け

コンピュータ産業では 1980 年代後半からのシステムのオープン化の流れのなかで、様々な Unix の派生版（亜種）が市場に存在した。このような状況のなか Unix の OS プラットフォーム製品提供者は、いかに自らが OS 階層での業界標準の主導権を握るか、という争いをおこなった。そのなかで AT&T 社が中心となった UI（Unix インターナショナル）と IBM 社が中心となった OSF（オープン・ソフトウェア・ファンデーション）の標準化の２大陣営が生まれることとなる。

1980 年代後半には各社の独自 Unix が普及していたため、アプリケーションはそれぞれの Unix 用に作り込む必要があった。仮に商用 Unix 間のバイナリ互換性[504]が統一されれば、アプリケーション開発者は OS の違いを気にすることなくアプリケーションを開発し、また転用や再利用など開発生産性は飛躍的に向上し、コストも大きく削減することが可能となる。顧客はこれにより切り替えのスイッチングコストを低く抑えることや、既に開発したアプリケーションの無駄を防ぐことができるなど、大きな恩恵を受けることが可能となる。

施策の内容は以下である。攻撃側プラットフォームとして、UI のシステム V（ファイブ）が OSF の AIX をターゲットとし、実行された施策は水平的に同一レベルの階層

---

503 サウスウイック（2000）, p. 113 参照。

504 バイナリ互換性とはふたつのプラットフォーム上で共通のバイナリコードが利用できること。例えば片方のプラットフォーム上でコンパイルしてできたプログラムをもう片方のプラットフォーム上でそのまま動作させられること。

間でのプラットフォーム製品を統合する施策と理解できる。UI は OSF に対抗して、自らが自社 OS を中心に据えて統合化を推し進めようと、CPU ならびに Unix 搭載機メーカーの囲い込みをおこなった。UI 陣営は 1989 年 11 月、「Unix システム V リリース 4.0」を完成させ発表する。これは「システム V」「バークレー4.3 」「XENIX」を統合したものである。本論文ではこれを「プラットフォーム製品統合」の戦略と呼ぶことにする。

これに対抗し OSF は 1990 年 10 月「OSF/1」を発表するが、発表の遅れや陣営内での不協和音によって搭載新機種を出せない状態が続き、大きく出遅れた。OSF は当初 IBM 社の「AIX」をカーネルにすることで進めていたが、DEC 社や HP 社の反対によって米カーネギーメロン大学の開発した「Marc(マーク）」を採用する方針に切り替えた。

UI としては統合するシステム V にほとんどの Unix を統合する目論見であったが、結果として Unix の標準化は２大陣営の分裂と対立という結果を招くこととなった。

（３） 戦略の意図と解釈

Katz & Shaprio（1985）が指摘する間接的なネットワーク効果の効用を引き出すためには、バイナリ互換をもった Unix の統一をおこなうことはプラットフォーム・リーダーがとりうる有効な戦略のひとつであり、当時米国の２大企業の AT&T 社と IBM 社の代理戦争とまでいわれた Unix 標準化の２大陣営の争いも、まさにこの戦略の遂行の結果表面化した争いであったと理解できる。これをプラットフォーム製品戦略としてプラットフォーム・リーダーの施策の視点で考えた場合、1980 年代までそれまで不統一でばらばらであった商用 Unix のバイナリ互換を共通化し統合することは、結果として２大陣営の分裂を招き、真の“統合” は実現しなかったにせよ、顧客と補完業者にとっての間接的効用の増大をもたらすことを標榜し、自社陣営に Unix 搭載機メーカーを惹きつけ組織化する意図であったと理解することができる（表 10-4）。

表 10-4:統合による顧客と補完業者に対する間接的効用

対象の種類	間接的効用の対象	間接的効用の内容
顧客	システム導入顧客	インストールド・ベースの増大による補完製品の多様化・低価格化。
補完業者	アプリケーション開発業者	OS のインストールド・ベース増大による補完製品（アプリケーション）開発インセンティブの向上。
	CPU 提供メーカー	OS のインストールド・ベース増大による補完製品（CPU）生産インセンティブの向上。

統合前の時点で一定のインストールド・ベースを確保していた「システム V」と「バークレー版 Unix」の搭載メーカーは、ネットワーク効果の間接的便益を利用して自らがその自社 OS を中心にすえて統合化を推し進めようと、CPU ならびに Unix 搭載機メーカーの“仲間つくり”をおこなった。言い換えれば、クリティカルマスに到達しないその他の独自商用 Unix を一掃する動きに出たのである。

この統合化の目的はネットワーク効果が、システム導入顧客とアプリケーション開発業者に対して正の因果ループをもたらすことであり、この点を CPU ならびに他の Unix 搭載機メーカーにアピールして“仲間つくり”を推進することにより、正の因果ループのメリットをもたらそうというメカニズムの遂行であったと考えられる。

（4） 小括

Unix 戦争の事例では示唆されるいくつかのプラットフォーム製品戦略上のポイントが存在する。Rohlfs（2003）はネットワーク効果を引き出す概念のひとつとして相互連結（interlinking）を指摘している。また相互連結の“費用（コスト）”に関して、相互連結の費用は新製品が製造される場合は少なくて済むが、すでに製品化された互換性をもたないハードウェアの場合は大きいかもしれないと論じている。

本事例は、ゼロから市場に製品のシェアを築いていくのとは異なり、既に一定数のインストールド・ベースの顧客が市場に存在している環境での「統合」（＝相互連結）の事例である。このように多くの専用・非専用プラットフォーム製品が乱立する環境

においては、統合の費用をできるだけ少なくするため、プラットフォーム・リーダーは自らのプラットフォーム製品を統合の中心に据えて費用の負担を少なくして統合化を図ろうとするという戦略が重要となる。結果としてUIにとって対立陣営OSFの主幹であるIBM社のAIXを中心OSに据える統合は、自陣営にとってコストが大きすぎると判断し容認しなかったと解釈できる。

もうひとつの戦略上のポイントは“利益コミュニティ”の考え方である。Rohlfs（2003）は相互連結のない状態では市場はたったひとつの供給者に引き寄せられる傾向をもつと論じている。しかしこの傾向にもかかわらず、小規模な供給者は市場のニッチを捉えることによりしばしば生き延びるとも指摘している。

既に大きなインストールド・ベースをもつ供給者が、新規参入の供給者との相互連結を拒むという「独り占め」の戦略は、現在まで多くの研究事例として取り上げられてきている。しかし利益コミュニティを前提とするニッチ市場が存続可能であるならば、「統合」という相互連結をあえてしないという戦略の選択が、たとえ大きなインストールド・ベースをもたないプラットフォーム・リーダーにも有効な戦略としてとり得るということである。UIに与することのなかったOSFのAIXやHP/UXが2006年時点でも市場に存在し続けている事実は、このニッチにおける利益コミュニティの考え方を肯定しているかも知れない。

付録 B

プラットフォーム製品バンドルの事例

第１節　はじめに

複数レベル階層のプラットフォーム製品戦略の事例として、プラットフォーム製品バンドルを詳説する。具体的には、プラットフォーム包囲の概要、ならびに 1990 年代後半から 2000 年にかけて、リアルネットワークス社の SMP（Streaming Media Player）とマイクロソフト社の WMP（Windows Media Player）との間で起こったプラットフォーム包囲の事例を取り上げる。加藤（2008）と加藤（2009b）にて先行研究として取り上げた Eisenmann, Parker & Alstyne(2006)と Eisenmann, Parker & Alstyne(2007)のプラットフォーム包囲からの引用を基にする。

第２節　Eisenmann, Parker & Alstyne(2006)ならびに Eisenmann, Parker & Alstyne(2007)の先行研究

Eisenmann, Parker & Alstyne（2006）のツーサイド・プラットフォーム理論で興味深いのは、複数のユーザー・グループに対してサイド内ならびにサイド間ネットワーク効果を引き出す観点からの施策が有効であると論じている点である。加えて Eisenmann, Parker & Alstyne（2007）では、ツーサイド・プラットフォームの「包囲の脅威」をさらに深く論じる「プラットフォーム包囲論」を提示している。プラットフォーム包囲は、飛躍的なイノベーションやシュンペーターの唱える創造的破壊を必要としないプラットフォーム・リーダーシップの交代のメカニズムを、プラットフォーム製品階層バンドルの概念で提供している。特にプラットフォーム製品バンドル

に見られるプラットフォームの包囲の概念に関しては、レイヤースタック内の下位階層のプラットフォーム製品による上位階層のプラットフォーム製品の包囲という視点を提供している。

（１）Eisenmann, Parker & Alstyne（2006）の研究

ここでは、Eisenmann, Parker & Alstyne（2006）のツーサイド・プラットフォーム戦略の概略を説明する。

１）収穫逓増の法則[505]

１面的な市場をターゲットにした製品やサービスは、事業が拡大していくにつれ、ある時点から「収穫逓減」の法則が働く。しかし２面的な市場のネットワーク効果が働く場合は、「収穫逓増」となる。数が増えれば、利益は乗数的に増えていく。ユーザーはより規模の大きいネットワークに多くの対価を払う傾向があるため、ユーザー基盤の拡大につれ収益も拡大可能である。２面的な市場ではこのように収穫逓増が約束されていることからプラットフォーム製品間では激しい競争が展開される。また収穫逓増が働く市場では、成熟化が進むと少数のプラットフォーム製品による「独占」が発生しやすい。このような２面的なネットワークは多くの業界で見られる。

２）ネットワークの２面性の力学[506]

ネットワークの２面性を特徴とする市場における取引には、常に三者関係が必要となる。またネットワーク効果はサイド内とサイド間の２種類を考慮する必要がある。サイド内ネットワーク効果とはユーザーの数が増えると、そのユーザーが属するグループにとって、プラットフォーム製品の価値が向上あるいは下落する現象を表す。またサイド間ネットワーク効果とは片方のユーザーが増加すると、もう片方のユーザー・グループにとってプラットフォーム製品の価値が向上あるいは下落する現象である。

---

505 Eisenmann, Parker & Alstyne（2006）, p. 94 参照。
506 同上。

３）提起する３つの課題[507]

プラットフォーム製品戦略上の目下の問題として、プラットフォーム製品提供者が２面的なネットワークの構築と維持に躓いていることがある。その根底に共通する問題とは、戦略を立案するにあたって、ネットワーク効果が働かない市場の仮説やパラダイムの下、２面的な市場の経済構造を無視して意思決定してしまっている。このため市場の２面性にまつわる課題にどのように取り組めばよいかについてツーサイド・プラットフォーム戦略では３つの課題を提起している。第１の課題は「プライシング」、第２の課題は「１人勝ちの力学」、そして第３の課題は「包囲の危機」である。

４）第１の課題：異なるユーザー・グループへのプライシング[508]

ビジネスモデルを構築するうえで考慮すべき要素として、戦略策定上の第１の課題はプライシングである。２種類のユーザー・グループにそれぞれ異なるプライシングをする際、どちらのプライシングにおいてもユーザーの増加率と購買意思を考慮する必要がある。一般的なのが「優遇される側」と「課金される側」である。一方のユーザーが増加すれば、もう一方のユーザーに大きな価値をもたらす場合、前者が優遇される側となり、後者は課金される側となる。狙いは「サイド間ネットワーク効果」を生み出すことである。

５）第２の課題：１人勝ちの力学[509]

第２の課題として１人勝ちの力学がある。２面的な市場では収穫逓増の法則が見込めるため、プラットフォーム製品提供者同士の戦いは、唯一の勝者が決まるまで続いていく可能性が高い。したがって、プラットフォーム製品提供者はどこかの時点で決断を迫られる。つまり、最後まで戦い続けるのか、それとも競合他社とプラットフォーム製品を共有するのか決定しなければならない。

---

507 同上。
508 Eisenmann, Parker & Alstyne（2006）, p. 95 参照。
509 Eisenmann, Parker & Alstyne（2006）, p. 99 参照。

６）第３の課題：包囲の危機[510]

新しいプラットフォーム製品の確立に成功しても、隣接市場から新たなプラットフォーム製品提供者が参入し、１人勝ちしたはずのプラットフォーム製品を包囲する脅威が発生する可能性がある。各プラットフォーム製品のユーザー基盤には重なり合う部分があり、ここをテコにすれば、あるプラットフォーム製品提供者が別のプラットフォーム製品を丸ごと飲み込むことがある。実際の脅威は隣接市場のプラットフォーム製品提供者がこちらのプラットフォーム製品機能を含め、製品やサービスをバンドルして提供する際に起こる。このような多機能のバンドル製品が低価格で提供されれば、単体のプラットフォーム製品を提供する側はお手上げの状態となる。敵のバリュー・プロポジション（提供価値）に対抗しようにも、課金される側のユーザー設定価格を引き下げる余裕はない。また同じくバンドル製品で対抗することもできない。

２面的な市場、なかでも技術進歩が速い市場では、このように包囲される状況が起こりやすい。その結果、市場の境界線があいまいになる「収斂（コンバージェンス）」が起こる。単体のプラットフォーム製品を提供する企業が包囲の危機に直面したとき、選択の余地はほとんどない。身売りするか、撤退するかのどちらかである。よってプラットフォーム製品に特化した提供者は、常に包囲の危機を警戒しなければならない。市場の境界があいまいになればなるほど、どの方向から攻撃されるか予測がつきにくいからである。

７）リアルネットワークス社の包囲の危機への対抗策[511]

しかし、まったく打つ手がないわけではない。単体プラットフォーム製品提供者が包囲されないためには、どうすればよいのか３つの対抗策を提示している。それは①ビジネスモデルを変える、②心強い仲間と手を組む、③法的手段に訴える、の３つである。ストリーミング・ソフトの草分け的存在であるリアルネットワークス社は、これらを実践して生き残りをはかっている。

リアルネットワークス社は消費者にストリーミング・メディア・プレイヤーを無償配布し、コンテンツ企業にサーバーを販売することで 1990 年と 2000 年にはストリーミング・メディア市場を短期間で独占し、かなりの利益を上げていた。しかしそれ以

---

510 Eisenmann, Parker & Alstyne（2006）, p. 100 参照。
511 同上。

前の 1998 年にはマイクロソフト社からの攻撃によって市場から駆逐されそうな時期もあった。マイクロソフト社はリアルネットワークス社同様ストリーミング・メディア・プレイヤー（マイクロソフト社の場合は Windows Media Player/WMP）を無償配布した。またストリーミング・サーバーを NT サーバー[512] に標準装備した。

コンテンツ企業は喜んでマイクロソフト社の提案を受け入れた。NT サーバーを買えばストリーミング・サーバーが無料でついてくるからである。また消費者も NT サーバーから配信されたデータを受信するには WMP が必要であったため追随することになった。

このようなマイクロソフト社の包囲の脅威に対しリアルネットワークス社は以下のような対抗策を選択した。

①ビジネスモデルを変える[513]

マイクロソフト社の攻撃を受けて、リアルネットワークス社は「課金先を変える」という戦術にでた。2003 年、リッスン・ドットコム社のラプソディを買収し、且つこの財産をテコに会員制音楽配信サービス＜リアルラプソディ＞に乗り出し、全 50 万曲のなかから無制限に PC にストリーミングできるサービスを開始した。

②心強い味方と手を組む[514]

リアルネットワークス社はケーブル・テレビ会社や携帯電話会社と提携しその力を借りる戦術をとった。ケーブル・モデム・サービスに会員制の音楽配信サービスを組み込むことで、他の音楽配信サービスに乗り換えるとなると別のメディア・プレイヤーが必要となったり、リストの作り直しが必要だったりとスイッチングコストを高めることによって乗り換えにくくするなどの手段を講じた。

---

512 マイクロソフト社が提供するファイル・サーバー、プリント・サーバー、メール・サーバー、ウェブ・サーバーなどの機能を備えた多目的サーバーOS。

513 Eisenmann, Parker & Alstyne（2006）, p. 101 参照。

514 同上。

③法的手段に訴える[515]

マイクロソフト社を反トラスト法違反で訴え 2005 年に 4 億 6000 万ドルの和解金のほか、3 億 100 万ドルの支払いと、リアルラプソディへの販促協力を取り付けた。

以上が Eisenmann, Parker & Alstyne (2006)のツーサイド・プラットフォーム戦略の概略である。

（２）Eisenmann, Parker & Alstyne (2007)の研究

ここでは、先行研究のひとつである Eisenmann, Parker & Alstyne (2007)のプラットフォーム包囲論の概要と、そのなかで論じられる３つの攻撃に関して簡単に記述する。Eisenmann, Parker & Alstyne (2007)のプラットフォーム包囲論の独占権行使攻撃(後述)は、既にいくつかの適用事例によって明らかにされた、隣接する階層間に対する有効な戦略のひとつとして位置付けられる。

１）プラットフォーム包囲論の概要[516]

Eisenmann, Parker & Alstyne (2007)のプラットフォーム包囲戦略とは、プラットフォーム製品提供者がプラットフォーム製品のコンポーネントの共通する部分と重なり合うユーザーの関係をテコ（leverage)にすることによって、自分自身の機能にターゲットのプラットフォーム製品の機能をバンドルの状態で結合させ、ターゲットのプラットフォーム製品の市場に入っていく戦略である。

このプラットフォーム製品の包囲戦略は、飛躍的イノベーションやシュンペーターの創造的破壊を必要としないプラットフォーム・リーダーシップの交代のメカニズムである。強いネットワーク効果と高いスイッチングコストによって、スタンドアローン競合の参入からは隔離された状態にある支配的な企業も、隣接したプラットフォー

---

515 同上。

516 Eisenmann, Parker & Alstyne（2007）, p. 1 参照。

ム製品提供者の複数の階層を結合させる包囲の攻撃に対しては脆弱である場合があると論じている。

２）包囲の３つの攻撃の概要[517]

プラットフォーム包囲の攻撃の選択は攻撃側とターゲット側のプラットフォーム製品間の関係に依存する。２者のプラットフォーム製品の関係は以下の４つに分類されるとしている。

①非関連プラットフォーム製品関係：ふたつのプラットフォーム製品は機能的に非関連で、いわゆる根本的に異なる目的でデザインされたものの場合である。

②代替品プラットフォーム製品関係：ふたつのプラットフォーム製品が似通った機能を提供し、代替品関係がある場合である。しかし強い代替品関係は包囲の攻撃の対象外としている。弱い代替品関係で同じ基本的な目的を持っているが異なるタイプの取引に則した技術を用いているような場合、包囲の攻撃の候補となる。例として Monster.com と LinkedIn.com の関係などがある。

③入れ子状態にされたネットワーク・ユーザー関係：プラットフォーム仲介ネットワークが幾層からなる階層状態にある場合である。結果として同じ構成要素がひとつのネットワーク階層のプラットフォーム製品提供者であると同時に、別のプラットフォーム製品のユーザーであったりもする。例えば eBay と WorldWideWeb の関係である。

④入れ子状態にされた部分コンポーネンツ関係：プラットフォーム製品提供者はそのプラットフォーム製品とコンタクトするネットワーク・ユーザーの最初のコンタクトポイントとなる。しかしそれらは必ずしもすべてのプラットフォーム製品のコンポーネンツを供給していない。例として eBay とクレジットカードの関係などである。

---

517 Eisenmann, Parker & Alstyne（2007）, p.5 参照。

上記の攻撃側のプラットフォーム製品とそのターゲット側のプラットフォーム製品との関係によって選択される包囲の攻撃は、以下の３つをそれぞれ適用するとしている。

①複合事業化攻撃（conglomeration attack）[518]：攻撃者のバンドルの対象は非関連事業であり、いわゆるコングロマリット化の攻撃である。プラットフォーム製品同士は機能的には関連していないが、共通のユーザーとコンポーネンツを共有している場合はバンドリングによりかなりの範囲の経済（economies of scope）を得ることができる。

②複合モード攻撃（intermodal attack）[519]：攻撃者のプラットフォーム製品とターゲットのプラットフォーム製品は弱い代替品関係である。それらプラットフォーム製品は同じ基本的な目的を持っているが、異なるユーザーのニーズを満足させるもので、異なるテクノロジーと独特なモード（提供の形態）を有している場合に適用する攻撃である。

③独占権行使攻撃（foreclosure attack）[520]：ターゲットのプラットフォーム製品は攻撃者のプラットフォーム製品の入れ子の状態になっているユーザーか、または入れ子状態になっているコンポーネンツのどちらかである場合に選択される攻撃である。攻撃者は自社のプラットフォーム製品にターゲットのプラットフォーム製品をバンドルする。この攻撃では、プラットフォーム製品同士は補完関係である。隣接階層に足掛りを持っていないスタンドアローンの競合プラットフォーム製品提供者は、この攻撃で市場での力を弱めさせられる。

特にコンピュータ・ソフトウェア製品における施策として適用性が高いものとしてこの独占権行使攻撃が挙げられる。独占権行使攻撃の過去の適用事例[521]として、①攻撃側：マイクロソフト社の Windows、ターゲット側：リアルネットワークス社のストリーミング・メディア・プレイヤー、②攻撃側：eBay 社のオークション・マーケットプレイス、ターゲット側：PayPal 社の e-mail ペイメントサービス、③攻撃側：マイ

518 Eisenmann, Parker & Alstyne（2007）, p. 13 参照。
519 Eisenmann, Parker & Alstyne（2007）, p. 16 参照。
520 Eisenmann, Parker & Alstyne（2007）, p. 17 参照。
521 Eisenmann, Parker & Alstyne（2007）, p. 14 参照。

クロソフト社のオフィス、ターゲット側：アドビ社のアクロバット・PDF ライター・ソフトウェアの３つが Eisenmann, Parker & Alstyne (2007)では提示されている。ちなみにコンピュータ・ソフトウェア産業での事例は①と③で、両者とも攻撃者はマイクロソフト社である。

３）独占権行使攻撃の整理[522]

以下、独占権行使攻撃に関して階層間に対する施策の影響を考察するために、４項目で整理をおこなう。４項目とは①攻撃の際のコンテクスト、②ターゲットの選定、③攻撃（施策）の選択と実行、④もたらされる効果とする。

① 攻撃の際のコンテクスト

ターゲットのプラットフォーム製品が大きな市場シェアを持っている（または持とうとしている）状態で、ユーザーとコンポーネンツの重複関係が相互重複(Reciprocal)や一方向的重複(Unilateral)になっている場合は、攻撃を仕掛けた際の成功の可能性が高い。

②ターゲットの選定

ターゲットは同一レベル階層での競合のプラットフォーム製品で入れ子にされた側のプラットフォーム製品であり、攻撃側は同一レベル階層での競合のプラットフォーム製品で入れ子にしている側のプラットフォーム製品となる。

③攻撃（施策）の選択と実行

自分自身の機能にターゲットのプラットフォーム製品の機能をバンドルの状態で結合（隣接階層とのバンドリング）させる独占権行使攻撃となる。この場合、攻撃者であるプラットフォーム製品提供者は隣接する下位階層にもプラットフォーム製品を提供している。そのため両プラットフォーム製品をバンドルすることによって、プラットフォーム製品市場での市場独占権を行使し、下位階層にはプラットフ

---

522 Eisenmann, Parker & Alstyne（2007）, p. 1-5 参照。

ォーム製品を提供していないプラットフォーム製品提供者を攻撃（駆逐）することが可能となる。

④もたらされる効果

結果として、攻撃者のプラットフォーム製品がターゲットのプラットフォーム製品を捕り込むということになる。また、攻撃者の成果としてはターゲットのプラットフォーム製品の市場に入っていくことができる（ターゲットのプラットフォーム製品を捕り込み消滅させる）。一方、ターゲットの被る影響は自分のプラットフォーム製品が攻撃者のプラットフォーム製品に捕り込まれてしまう。自社プラットフォーム製品は駆逐されてしまう。

以上がEisenmann, Parker & Alstyne (2007)のプラットフォーム包囲戦略の概略である。

付録 C

ユーザーにとってのアクセス価値

第１節　はじめに

ネットワーク効果を考える際に重要な概念であると思われるユーザーにとってのアクセス価値について説明する。これまでの議論では、普及やドミナント化の議論において、アクセス可能ユーザーの数による影響の考慮をおこなってきたが、ここでアクセス価値を取り上げる理由は、アクセス価値が普及やドミナント化に影響するという問題意識に依拠する。説明は、根来・加藤（2008）の先行研究の要約を基にする。

第２節　アクセス可能ユーザーにとってのアクセス価値

これまでのネットワーク効果の研究においては多くの場合、通信産業のネットワークの大きさ（ネットワーク加入者の数）がユーザーに与える影響を意識して議論がおこなわれてきた。ここでの提案は、このネットワーク加入者の「数」だけでなく、特定個体間のアクセスの「価値」（アクセスの頻度と重要度）を考慮したネットワーク効果の把え方を提案する。

（１）特定個体間のアクセスの価値：ネットワーク効果を評価するための追加概念

ネットワーク効果の研究においては、Kats & Shapiro（1985）やRohlfs（2001)にあるように、ネットワークの大きさ（ネットワーク加入者の数）がユーザーに与える

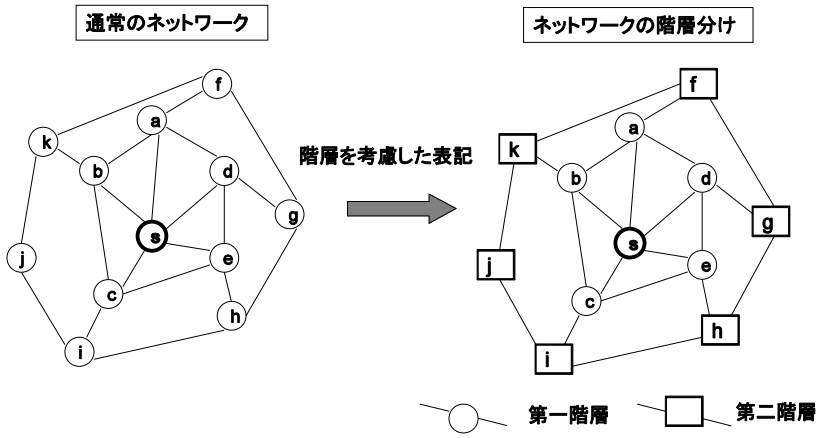
影響を意識した議論がなされてきた。このネットワーク加入者の「数」とともに、ここではネットワーク効果を評価するための追加概念として、特定個体間のアクセスの「価値」（アクセスの「頻度（frequency）と重要度(weight)」）を考慮すべきことを提案する。

１）表記法：ネットワークの階層分け

追加概念の詳細を説明する前に、ここで論ずる階層構造を前提にしたネットワークの考え方を示すための表記法について示す。これは通常のネットワークの表記に階層分けを考慮した表記となる。図 12-1 は第 1 階層と第 2 階層の 2 階層に分けた場合の図である[523]。

それぞれの個体間を結ぶ実線は「アクセス可能」を意味する。個体 s を中心にした場合、a から k までの個体間アクセスは左図のように表記される。この図において a から e までを第一階層（s と直接アクセスする個体）、f から k までを第二階層とした場合、右図のように表記できる（図 12-1）。

図 12-1：ネットワークの階層分け表記



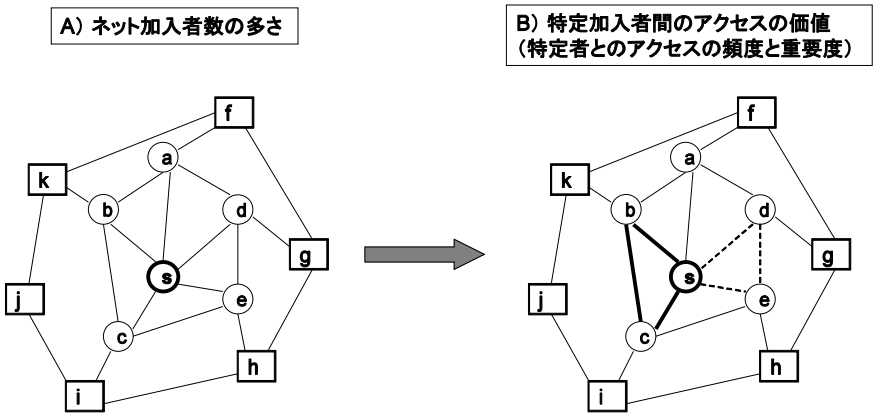
523　ここでの議論は、２階層の製品システムを前提にしておこなわれるが、３階層以上の多階層の製品システムにも拡張可能であると思われる。

２）特定加入者間のアクセスの価値

上記階層構造の表記を使って、ここでは特定個体間のアクセスの価値を加味したネットワーク効果の概念を論じる。以下では、アクセスの価値の大小を表す表記法として個体間を結ぶアクセスの実線を、そのアクセス価値の高さに応じて太線や普通線や破線（この順番で価値が低くなる）で区別して表記する[524]。

図 12-2 は図 12-1 にアクセス価値の追加概念を加味して表記したものである。図 12-2 ではｓを中心とした場合ｂ、ｃとのアクセス（関係）は、そのアクセス価値がｓにとって高いものであるため他の個体間を結ぶ線より太い線で表記する。また逆にｄ、ｅはアクセス価値が低いため破線で表記する（図 12-2）。

図 12-2：アクセス価値の加味



ネットワーク効果の過去の研究においては多くの場合、全体ネットワークの大きさ（ネットワーク加入者の数）がユーザーに与える影響を意識した研究がおこなわれてきた[525]。前述したように、本論文はこのネットワーク加入者の「数」とともに、「特定個体間のアクセス価値」を考慮したネットワーク効果の把え方が、ネットワーク効果を考える際に必要であると考える。

---

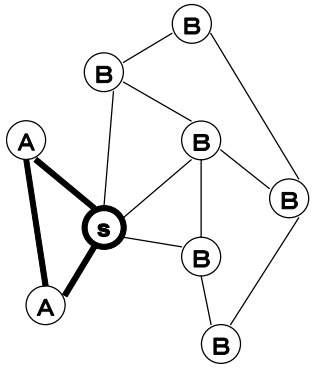
524 ここでは、頻度と重要度を区別した議論はおこなわない。これはアクセスの価値を頻度×重要度の一種の「積」と把えていることになる。もちろん頻度と重要度を区別したより詳細な議論も可能だろう。

525 例えば、Shapiro & Varian（1999）で引用されている電話のネットワークやカラーテレビの事例。Rohlfs（2001）の Fax, Early Telephone のケーススタディなどがある。

例えば特定個体間のアクセスの価値を考慮することにより、最近携帯電話会社で導入されている家族間無料通話などの特定個体間アクセスの需要を狙ったサービスが、ユーザーにとって携帯電話の購入のインセンティブのひとつになっている現象をうまく説明することができる。従来のネットワーク効果の把え方である「製品の普及の進展によってもたらされるユーザーの便益」や「多くの人とコミュニケーションが可能になることから得られる便益」に共通の「アクセス可能な加入者の数」の観点だけでは、例えばほとんど通話することのない地域の加入者数の増加は、ユーザーにとってプラットフォーム製品選択の大きなインセンティブとはならない。言い換えれば、アクセス可能な数が増加してもユーザーにとってメリットが薄い場合、選択のインセンティブになることはない。しかし例えば家族間など頻繁にしかも重要な相手とのアクセスの便益が向上する（例：無料サービスが提供される）ことは、ある携帯電話のキャリア（プラットフォーム製品）を選択するインセンティブとなり得る。

図 12-3 は上記の家族間無料通話の例を図にしたものである。仮に A という携帯電話網と B という携帯電話網があったとする。B の既存加入者数は A の加入者数より多いので s がどちらかを選択する場合、「アクセス可能な加入者の数」だけがインセンティブとして働くならば、加入者の数が多い B の携帯電話網を選択することとなる。しかし実際にはAのネットワークに所属する特定相手(家族)との通話頻度を考えて、B ではなく A を選択するインセンティブが働き得る。

図 12-3：家族間無料通話[526]の例



526　図 12-3 で使われている太い線はアクセスの価値が大きいことを示す。

以上が、アクセス可能ユーザーにとってのアクセス価値の説明である。本論文のなかでは、特にアクセス価値に関する議論はおこなっていない。今後の研究の課題として検討する事項であると考える。

謝辞

本論文は、著者が早稲田大学大学院商学研究科博士後期課程に外資系 IT 企業に勤めながら在職の社会人ドクターコースの学生として入学し、その後、現職の大学教員となるまでの約８年間の研究をまとめたものです。本論文の作成にあたり多くの方のご指導とご支援を賜りましたことを、ここに深く感謝いたします。

主査である根来龍之先生には科目履修生時代から長年にわたりご指導とご助言をいただきました。根来龍之先生のご支援がなければ本論文は完成し得ませんでした。副査の黒須誠治先生（早稲田大学）と坂野友昭先生（早稲田大学）ならびに國領二郎先生（慶応義塾大学）の先生方には論文指導などを通じて貴重なご助言をいただきました。深く感謝し、お礼申し上げます。

そして学部時代の明治大学政治経済学部経済学科のゼミの指導教員であられた故楓元夫先生、オーストラリア大学院留学時代のボンド大学大学院の Dr. Andrew Crouch には、現職の大学教員の職に就くにあたり大きな影響を受けました。

加えて、妻や家族これまで様々な局面で支えてくれた親しい方々に心より感謝いたします。

2014 年 3 月

加藤 和彦

参考文献

[1] Baldwin, C. Y. and Clark, K. B. (1997) Managing in an Age of Modularity, *Harvard Business Review*, Vol. 75 No. 5, pp. 84-93.

[2] Baldwin, C. Y. and Clark, K. B. (2000) *Design Rules, Vol. 1: The Power of Modularity*, Cambridge, MA: MIT Press. (安藤晴彦訳『デザイン・ルール　モジュール化パワー』東洋経済社、2004 年)

[3] Cusumano, M. A. (2004) *The Business of Software*, New York, NY: Free Press. (サイコム・インターナショナル監訳『ソフトウェア企業の競争戦略』ダイヤモンド社)

[4] DiBona, C., Ockman, S. and Stone, M. (1999) Introduction, *Open Sources: Voices from the Open Source Revolution*, Sebastopol, CA: O' Reilly.

[5] Eisenmann, T. (2007) Winner-Take-All in Networked Markets, *Harvard Business School*, Note. 9-806-131.

[6] Eisenmann, T., Parker, G. and Alstyne, M. W. V. (2006) Strategies for Two-Sided Markets, *Harvard Business Review*, Vol. 84, No. 10, pp. 96-101. (「ツー・サイド・プラットフォーム戦略」『Diamond ハーバード・ビジネス・レビュー』ダイヤモンド社、2007 年 6 月, 68-81 ページ)

[7] Eisenmann, T., Parker, G. and Alstyne, M. W. V. (2007) Platform Envelopment, *Harvard Business School Working Paper*, No. 07-104.

[8] Evans, D. S., Hagiu, A. and Schmalensee, R. (2008) *Invisible Engines*, Cambridge, MA: MIT Press.

[9] Foley, M. J. (2008) *Microsoft 2.0: How Microsoft plans to Stay Relevant in the Post-Gates Era*, Indianapolis, IN: John Wiley & Sons. (長尾高弘訳『マイクロソフト　ビル・ゲイツ不在の次の 10 年』翔泳社)

[10] Gawer, A. and Cusumano, M. A. (2002) *Platform Leadership*, Boston, MA: Harvard Business School Press. (小林敏男監訳『プラットフォーム・リーダーシップ』有斐閣、2005 年)

[11] Hagiu, A. (2006a) How Software Platforms Revolutionize Business, *Harvard Business School Working Knowledge*, <http://hbswk.hbs.edu/item/5482.html> 2012/11/05

[12] Hagiu, A. (2006b) Multi-Sided Platforms: From Microfoundations to Design and Expansion Strategies, *Havard Business School Working Paper*, No. 09-115. <http://www.people.hbs.edu/ahagiu/TSP_microfoundations_and_strategies_01062007.pdf> 2007/05/01

[13] Hagiu, A. (2006c) New Research Explores Multi-Sided Markets, *Harvard Business School Working Knowledge*, <http://hbswk.hbs.edu/item/5237.html> 2007/05/01

[14] Katz, M. L. and Shapiro, C. (1985) Network Externalities, Competition, and Compatibility, *American Economic Review*, Vol. 75, No. 3, pp. 424-440.

[15] Katz, M. L. and Shapiro, C. (1986) Technology Adoption in the Presence of Network Externalities, *The Journal of Political Economy*, Vol. 94, No. 4, pp. 822-841.

[16] Kato, K. and Negoro, T. (2007) A Theoretical Review of Network Effects on Platform Products, THE INTERNATIONAL SOCIETY FOR THE SYSTEMS SCIENCES, <http://journals.isss.org/index.php/proceedings51st/article/view/595> 2013/09/16
[17] O'Reilly, T. (1999) Hardware, Software, and Infoware, *Open Sources: Voices from the Open Source Revolution*, Sebastopol, CA:O'Reilly.
[18] Perens, B. (1999) The Open Source Definition, *Open Sources: Voices from the Open Source Revolution*, Sebastopol, CA:O'Reilly.
[19] Raymond, E. S. (1999) The Revenge of the Hackers, *Open Sources: Voices from the Open Source Revolution*, Sebastopol, CA:O'Reilly.
[20] Rochet, J. and Tirole, T. (2003) Platform Competition in Two-Sides Markets, *Journal of the European Economic Association*, Vol. 1, 2003, pp. 990-1029.
[21] Rohlfs, J. H. (2001) *Bandwagon Effects in High-Technology Industries*, Cambridge, MA:The MIT Press. (佐々木勉訳『バンドワゴンに乗る：ハイテク産業成功の理論』NTT 出版、2005 年)
[22] Shapiro, C. and Varian, H. R. (1999) *Information Rules*, Boston, MA:Harvard Business School Press. (千本倖生監訳『ネットワーク経済の法則』IDG ジャパン)
[23] Torvalds, L. (1999) The Linux Edge, *Open Sources: Voices from the Open Source Revolution*, Sebastopol, CA:O'Reilly.
[24] VMware, Inc. (2008) Form 10-K, February29, 2008, Annual Report. <http://ir.vmware.com/annuals.cfm> 2012/11/05
[25] Yoffie, D., Hagiu, A. and Slind, M. (2009) VMware, Inc., 2008, *Harvard Business School*, Case. 9-790-435.
[26] Young, R. (1999) Giving It Away:How Red Hat Software Stumbled Across a New Economic Model and Helped Improve an Industry, *Open Sources: Voices from the Open Source Revolution*, Sebastopol, CA:O'Reilly.
[27] 青木昌彦・安藤晴彦(2006)『モジュール化　新しい産業アーキテクチャの本質』東洋経済新報社。
[28] 今井賢一・國領二郎(1994)『プラットフォーム・ビジネス』情報通信総合研究所。
[29] 岩山知三郎(2001)『ネットワークをコンピュータにした人々　ビル・ジョイの冒険』コンピュータ・エージ社。
[30] M. イアンシティ・R. レビーン(2007)（杉本幸太郎訳）『キーストーン戦略』翔泳社。
[31] 加藤和彦(2006)「コンピュータ OS のネットワーク外部性に関する考察―1980 年代後半の Unix 標準化の２大陣営対立の事例を通じて―」『商経論集』第 91 号, pp. 13-24。
[32] 加藤和彦(2008)「プラットフォーム戦略論における「包囲の危機」のフレームワークに関する適用可能性の一考察」『商学研究科紀要』第 66 号 早稲田大学大学院商学研究科, pp. 63-75。
[33] 加藤和彦（2009a）「コンピュータ・ソフトウェアのプラットフォーム戦略における階層間施策の考察」組織学会全国大会発表予稿集, pp. 123-126。
[34] 加藤和彦(2009b)「コンピュータ・ソフトウェアにおける階層化の時系列整理と考察」『商経論集』第 96 号, pp. 1-13。
[35] 加藤和彦(2009c)「コンピュータ・ソフトウェアにおけるプラットフォーム階層間施策の考察」『商学研究科紀要』第 68 号 早稲田大学大学院商学研究科, pp. 43-55。
[36] 加藤和彦(2009d)「階層構造をもつコンピュータ・ソフトウェアにおけるプラットフォーム戦略としての階層介入施策の考察」『日本経営学会誌』第 23 号, 千倉書房, pp. 75-86。
[37] 加藤和彦(2010)「コンピュータ・ソフトウェアのクロスプラットフォーム製品における競生の考察」, 経営情報学会全国大会。
[38] 加藤和彦(2011)「スタートアップ期のコンピュータ・ソフトウェア企業におけるク

ロスプラットフォーム製品戦略の考察」日本経営学会第 85 回全国大会報告要旨集, pp. 348-352。
[39] 加藤和彦(2012a)「コンピュータ・ソフトウェアの階層戦術の考察」早稲田大学 IT 戦略研究所, WP No. 47, pp. 1-26。
[40] 加藤和彦(2012b)「コンピュータ・ソフトウェアの階層戦術の考察－VMware の仮想化ソフトの事例を通じて－」経営情報学会全国研究発表大会。
[41] 加藤和彦(2013a)「コンピュータ・ソフトウェアの階層介入戦略におけるドミナント化の演繹的仮説の構築」, 経営システム学会全国大会発表予稿集, pp. 122-125。
[42] 加藤和彦(2013b)「コンピュータ・ソフトウェアの階層介入戦略の先行研究レビュと課題の考察」, 経営情報学会全国大会。
[43] 加藤和彦(2013c)「コンピュータ・ソフトウェアのプラットフォーム戦略論における課題と発展」, 日本経営学会全国大会発表予稿集, pp. 97-100。
[44] 加藤和彦(2013d)「コンピュータ・ソフトウェアにおける階層介入戦略の考察」『日本経営学会誌』 第 32 号, 千倉書房, pp. 19-29。
[45] M. C. ケリー・W. アスプレイ(2006)(山本菊男訳)『コンピューター200 年史』海文堂。
[46] 國領二郎(1995)『オープン・ネットワーク経営』日本経済新聞社。
[47] 國領二郎(1997)「プラットフォーム・ビジネスの構造」『DIAMOND ハーバード・ビジネス・レビュー』1997 年 11 月号, pp. 34-41。
[48] 國領二郎(1999)『オープン・アーキテクチャ戦略』ダイヤモンド社。
[49] K. サウスウイック(2000)(山崎理仁訳)『世界ハイテク企業の痛快マネジメント サン・マイクロシステムズ』早川書房。
[50] 佐々木裕一・北山聡(2000)『Linux はいかにしてビジネスになったか』, NTT 出版。
[51] A. シュエン(2008)(上原裕美子訳)『Web2.0 ストラテジー－ウェブがビジネスにもたらす意味』オライリー・ジャパン。
[52] 山田博栄(2000)「ネットワーク言語 Java の思想と日米の差」『デファクト・スタンダードの本質』有斐閣。
[53] 末松千尋(2002)『京様式経営－モジュール化戦略』日本経済新聞社。
[54] 末松千尋(2004)『オープンソースと次世代 IT 戦略』日本経済新聞社。
[55] 末松千尋・K. ベネット(1996)『Java 革命』ダイヤモンド社。
[56] 竹田陽子・國領二郎(1996)「情報技術が企業間関係に与える影響に関する試論」『慶應経営論集』, Vol. 13, No. 2, pp. 169-183。
[57] 出口弘(1996)「自律分散型組織の戦略的設計」『DIAMOND ハーバード・ビジネス・レビュー』, pp. 44-53。
[58] 出口弘(2005)「プラットフォーム財のロックインと技術革新」『経済論叢』 Vol. 175, No. 3, pp. 18-44。
[59] 根来龍之・足代訓史(2011)「経営学におけるプラットフォーム論の系譜と今後の展望」早稲田大学 IT 戦略研究所 WP, No. 39, pp. 1-24。
[60] 根来龍之・加藤和彦(2006a)「クスマノ＆ガワーのプラットフォーム・リーダーシップ「４つのレバー」論の批判的発展」早稲田大学 IT 戦略研究所 WP, No. 18, pp. 1-41。
[61] 根来龍之・加藤和彦(2006b)「クスマノ＆ガワーのプラットフォーム・リーダーシップ「４つのレバー」論の批判的発展」経営情報学会・オフィスオートメーション学会合同全国研究大会予稿集, pp. 570-573。
[62] 根来龍之・加藤和彦(2008)「プラットフォーム製品における「ネットワーク効果」概念の再検討」『国際 CIO ジャーナル』2008 Vol. 2, pp. 5-12。
[63] 根来龍之・加藤和彦(2009)「プラットフォーム間競争の技術「非」決定論」, 経営情報学会全国大会。
[64] 根来龍之・加藤和彦(2010)「プラットフォーム間競争における技術『非』決定論のモデル」早稲田大学国際経営研究, 第 41 号, pp. 79-94。
[65] 根来龍之・木村誠(1999)『ネットビジネスの戦略』日科技連出版社。

[66] 根本英幸・松岡功(1992)『巨人 IBM に挑むサン・マイクロシステムズの戦略』にっかん書房。
[67] A.ハジウ(2006)「マルチサイド・ソフトウェア・プラットフォーム」『日本のイノベーション・システム』東京大学出版会。
[68] 長谷川裕行(2000)『ソフトウェアの 20 世紀』翔泳社。
[69] M.ホール・J.バリー(1991)(アスキー書籍編集部監訳)『UNIX ワークステーションを創った男たちサン・マイクロシステムズ』アスキー出版局。
[70] 松下芳生・臼井淳(1998)『機心なきサン・マイクロシステムズの挑戦』コンピュータ・エージ社。
[71] 渡邉利和・川添貴生(2012)『仮想化するインフラを構築する技術』インプレスジャパン。