MMRC-J-152

製品アーキテクチャの選択プロセス
—デジタル複合機における ファームウェアの開発事例—

東京大学大学院経済学研究科　博士課程
福澤光啓

2007 年 3 月

東京大学21世紀COE〔センター・オブ・エクセレンス〕
ものづくり経営研究センター

.

製品アーキテクチャの選択プロセス

―デジタル複合機におけるファームウェアの開発事例―

東京大学大学院経済学研究科　博士課程

福澤光啓

2007 年 3 月

1. はじめに

近年、情報家電や携帯機器、自動車、産業機器などをはじめとした多くの製品分野において、製品に組み込まれているソフトウェア[1]によって実現される機能、および、機械部品や半導体デバイスなどのハードウェアとファームウェアとの連動によって実現される機能が増大している。これにともなって、ファームウェアの大規模化や複雑化が進み、製品開発コストの多くの部分を当該ファームウェアの開発コストが占めるようになり、製品開発を行う上でファームウェアの開発をいかに上手く行うのかが重要な課題となっている。

例えば、機械製品の代表であった自動車においても、マイコン（マイクロコンピュータ）とファームウェアからなる電子制御ユニット（ECU）が上級車では 100 個程度搭載されるようになっており、これらによって、車両走行制御やパワートレーン制御、ABS やエアバックに代表される安全装置、テレマティックスなどといった多数の機能が実現されるようになっている（福澤・立本・新宅、2006）。さらに、これらの機能を高い品質で提供するためには、複数の ECU をいかに上手く統合的に制御していくのかが重要であり、自動車メーカーや部

[1] 「組み込みソフトウェア」とか「ファームウェア」と呼ばれるが、本論文ではファームウェアと称する。ファームウェアとは、狭義では、ROM に格納されたマイクロ・プログラム（MPU の動作を直接制御するための言語（マイクロ・コード）によって記述されるプログラム）を指して用いられる（新宅・小川・善本、2006）。ファームウェアの果たす代表的な役割として、製品の高性能化、耐久性の向上、多機能化、低コスト化などが挙げられる（福澤他、2006）。

品サプライヤー各社にとって喫緊の課題となっている。また、携帯電話端末では、一台で通話や電子メール、写真撮影、ネットブラウザ、音楽プレーヤ、動画鑑賞などといった、多数の機能が提供されており、このような事例は、DVD レコーダやデジタルスチルカメラ、カーナビゲーションシステムなどといった多くの製品分野において観察される。

このような製品の多機能化は、製品の付加価値を高めるための手段の一つであり、これを上手く行うためには、どのような機能を織り込むのかということや、機能間での調整・統合をいかにして行うかが重要になる。デジタル化された製品において多機能化を行うためには、それぞれの機能に対応したアプリケーションソフト[2]やミドルウェア、OS などが必要になるのだが、これら複数のファームウェア・コンポーネント間の調整・統合をどのように行えば、製品の多機能化をうまく行うことができるのだろうか。さらに、これらのファームウェア・コンポーネント間の調整や統合を、上手く行う上で適した組織のあり方とはどのようなものなのだろうか。

従来の経営学では、どのようにして製品を構成部品（コンポーネント）に分割し、そこに製品機能を配分し、それによって必要となる部品間のインタフェースをいかに設計・調整するかに関する基本的な設計構想を製品アーキテクチャと呼び（藤本、2001）、それに関する研究が行われてきたのだが、そこで注目されているコンポーネントのほとんどが、ハードウェア・コンポーネントであった(青島・武石、2001; Baldwin and Clark、2000; Henderson and Clark、1990; 藤本、2001; Ulrich、1995)。そのため、ハードウェアとファームウェアの両方から構成される製品のアーキテクチャについては、十分に研究されてこなかった。また、ソフトウェア工学の分野においては、ソフトウェア開発モデルに関する研究は進められている（立本、2002）のだが、製品に組み込まれているソフトウェアが製品アーキテクチャにどのような影響を与えているのか、ということについては十分に研究されていない。

しかしながら、加藤（2002）や新宅他（2006）において述べられているように、製品に用いられるファームウェアの増大によって、それが製品アーキテクチャに与える影響が非常に大きく、かつ重要になっている。ハードウェアとファームウェアによって構成される製品において、多機能化を進めている企業の多くは、製品開発活動において一連のファームウェアをどのような構想の下で設計するのかということに取り組んでおり、各社とも困難に直面しているのである。上述の疑問に答えるために、本論文では、①製品機能とファームウェア・コンポーネントとの対応関係や②ファームウェア・コンポーネント間の対応関係、③ファー

[2]携帯電話機を例に取れば、電子メール閲覧・作成用ソフトや音楽再生用ソフト、ゲームソフト、写真撮影用ソフトなどがこれに該当する。

ムウェア・コンポーネントと CPU[3]との対応関係から記述される製品システムの特性を「ファームウェア・アーキテクチャ」と定義し、多機能化を長年にわたって進めてきた、株式会社リコー（以下、リコーと称する）のデジタル複合機におけるファームウェアの開発事例を取り上げて考察を行う。その際には、特に、①リコーの代表的なデジタル複合機におけるファームウェア・アーキテクチャや、②それらのアーキテクチャが生み出されて選択されていく一連のプロセスに注目していく。

2. 製品アーキテクチャに関する既存の議論

2.1 製品アーキテクチャの概念

そもそも、製品の「機能」がどのような「構造」によって実現されているのかということについては、製品アーキテクチャに関する議論が行われてきた(青島・武石、2001; Baldwin and Clark、2000;藤本、2001; Ulrich、1995)。Simon(1996)では、複雑なシステムを効率的に設計するためには、当該システムを多数の機能的部分に対応した半独立の構成要素に分解するための適当な方法を見出すことが必要であると述べられている。このように、半独立の構成要素に分解することのできるシステムは準分解可能（nearly decomposable）なシステムと呼ばれている。Ulrich(1995)では、製品アーキテクチャとは、①一連の機能要素と、②それらの機能要素と物理的コンポーネントとの対応関係、③相互関係にある物理的コンポーネント間のインタフェースの特徴によって定義されるとしている。あるシステムのアーキテクチャとは、「構成要素間の相互依存関係のパターンで記述されるシステムの性質である[4]」と定義される。さらに、藤本（2001）では、製品アーキテクチャとは、どのようにして製品を構成部品に分割し、そこに製品機能を配分し、それによって必要となる部品間のインタフェースをいかに設計・調整するかに関する基本的な設計構想のことであると述べられている。アーキテクチャを把握する視点としては、①モジュール化/統合化という視点と、②オープン化/クローズ化[5]という視点の二つがある(青島、武石、2001)。

まず、モジュール化とは、「システムを構成する要素間の相互関係に見られる濃淡を認識して、相対的に相互関係を無視できる部分をルール化されたインターフェースで連結しよう

[3] CPU は半導体デバイスのひとつであるが、製品機能を提供する上でファームウェアと密接に関わっているので、ファームウェア・アーキテクチャの構成要素のひとつに含めている。
[4]青島・武石 (2001, p.33)
[5] オープン化とは、システムの構築、改善、維持に必要とされる情報が公開され、社会的に共有･受容される動きのことであり、逆に、クローズ化とは、情報の社会的な共有･受容が制限される動きのことであるとされている（青島･武石、2001）。

とする戦略[6]」である。また、モジュールとは、「半自律的なサブシステムであって、他の同様なサブシステムと一定のルールに基づいて互いに連結することにより、より複雑なシステムまたはプロセスを構成するもの[7]」であると定義される。一方、統合化とは、「要素間の複雑な相互関係を積極的に許容して、相互関係を自由に解放して継続的な相互調整にゆだねる戦略[8]」である。モジュール化によって、各モジュール内部での進化のスピードは速くなるが、システムが達成することのできる最大のパフォーマンスには一定の制約がかかる。逆に、統合化されたシステムでは、全ての構成要素に自由な相互作用が許されているので、実現可能な最大のパフォーマンスは限りなく高くなるが、構成要素間の調整が複雑であるためそのシステムを進化させるのには多大な時間がかかる(青島、武石、2001)。このようなモジュラー型アーキテクチャの下では、「ミックス・アンド・マッチ」で製品を作ることができるので、市場や技術の変化に素早く対応できるという意味で、「戦略的柔軟性（strategic flexibility）」が高まるとされている（Sanchez and Mahoney、1996）。

2.2　製品アーキテクチャと組織との適合に関する研究

Thompson(1967)や Galbraith (1973)、von Hippel(1990)では、部門間調整のあり方は、当該組織が処理すべきタスクの相互依存性によって決まるということが示されている[9]。Clark and Fujimoto(1991)では、製品開発組織のあり方の理念型として、内的統合（部品どうしがぴったりとはまりうまく動作すること）と外的統合（製品体験がユーザーの期待と一致していること）の程度の違いによって、四つのタイプ（機能別組織、軽量級プロダクト・マネジャー、重量級プロダクト・マネジャー、プロジェクト実行チーム）が示されており、自動車のように「製品の首尾一貫性」が重要視されるような製品において、効果的な製品開発を行う上では、内的統合と外的統合の両方をうまく行うために重量級プロダクト・マネジャーを設ける開発組織のタイプが適しているということが示されている。

[6]青島･武石(2001), 前掲, p.33
[7]青木 (2002, pp.5-6)
[8]青島･武石(2001), 前掲, p.33
[9] Thompson(1967)では、組織における諸活動間の相互依存関係のタイプに応じて適した調整方法が異なるとされているが、このことを、ある製品システムにおける構成要素間の相互依存関係にも適用すると、製品システムにおける構成要素間の相互依存関係に応じて、必要となる調整方法が異なってくるということになる。例えば、von Hippel(1990)では、製品開発を行う際に、相互依存関係の強い問題解決活動を一緒にするようなタスクの分割(task partitioning) を行うことによって、効率的に製品開発を行うことができると述べられている。このようなタスクの分割を行う上で重要なことは、製品アーキテクチャとタスク分割とをうまく連携させることであると考えられる。von Hippel(1990)では、タスクの分割と製品アーキテクチャとが適合していることによって、効率的な製品開発を行うことができるとされているが、分割したタスク間での調整や統合を行う必要性が生じた場合に、組織がそれにどのようなプロセスをたどって対応していくのかということについては十分に議論されていない。

Sanchez and Mahoney(1996)においても、構成部品間の相互依存性が高い場合には、それぞれのコンポーネントを開発している組織あるいは企業間での緊密な相互調整が必要になると述べられている。一方、構成要素間の相互依存性が緩やかな場合には、それぞれのコンポーネントを開発している組織や企業間での相互調整の必要性が低いということが述べられている。同様に、楠木･チェスブロウ（2001)では、製品アーキテクチャが統合型であれば、「さまざまな要素の相互作用が不明確なので、市場メカニズムに基づく活動の調整は効率的ではなくなる[10]」ため、組織内部に活動を統合化するような「統合型の組織戦略」が適しているが、逆に、製品アーキテクチャがモジュラー型であれば、各構成要素間での相互作用については明確なので組織内部に活動を統合化するのではなくて、むしろ特定の活動分野に特化する「バーチャル型の組織戦略」が適していると述べられている。

さらに、藤本(2001)では、組織能力と製品アーキテクチャとの適合関係について、日本企業が得意とするのは、「擦合せ」重視、つまり「部品間の微妙な相互調整、一貫した工程管理、緊密な社内部門間調整、取引先との濃密なコミュニケーション、顧客との接点の質の確保など、社内外の擦合せが競争力を決めるタイプの製品･産業[11]」であり、米国企業が得意とするのは、「事前に『つなぎ』の部分(インターフェース)を標準化し、擦合せがそもそも不要となる工夫をした上で、自由自在に部品や製品設計や事業自体を連結し、ビジネスの急速展開に結びつけるというタイプの仕事[12]」であると述べられている。

以上の議論をまとめると、製品アーキテクチャのタイプとそれに適した組織のあり方（部門間調整や企業間での分業形態）との間には、ある一定の適合関係が存在しているということが示されてきたのである。すなわち、製品の部品間の相互依存性が高い場合（統合型アーキテクチャ）には、それを開発する組織における部門間調整は緊密に行われる必要があり、部品間の相互依存性が相対的に低い場合（モジュラー型アーキテクチャ）には、部門間調整はそれほど緊密に行われる必要はないのである。

2.3　製品アーキテクチャの変化に関する既存研究

製品アーキテクチャは、一般的には統合型からモジュラー型へとシフトする[13]（Abernathy 、1978; Baldwin and Clark、2000）が、逆にモジュラー型から統合型へとシフトするというよう

[10]楠木･チェスブロウ (2001, p.265)
[11]藤本(2001, p.11)
[12]同上, p.11
[13] Baldwin and Clark(2000)では、複雑なシステムに対して、「分離」、「交換」、「追加」、「削除」、「抽出」、「転用」という六つの「モジュール化オペレータ」が用いられることによって、当該システムのモジュール化が進められていくプロセスについて考察されているが、これはモジュラー化が進むという一方向的な現象のみに注目されるにとどまっている。

に、時間の経過とともにダイナミックに変化していく(Fine、1998; 楠木･チェスブロウ、2001)と考えられている。既存の議論では、製品アーキテクチャの変化を引き起こす主な要因として、①実現しようとする製品機能の変化と、①製品に用いられている技術の変化の二つに焦点が当てられてきた。

2.3.1 製品機能の変化による製品アーキテクチャの変化

製品アーキテクチャの変化を引き起こす要因として、当該製品で実現しようとする機能の変化（小型化や軽量化、高性能化が主な目的）が挙げられる。Henderson and Clark(1990)では、構成部品に用いられている技術には変化はないのだが、相互依存関係にある構成要素構成要素の組み合わせ（つまり製品アーキテクチャ）に変化が生じるような製品イノベーションをアーキテクチャル・イノベーション[14]（architectural innovation）と呼び、このようなイノベーションに既存企業が対応できない理由について議論されている。その理由として、Henderson and Clark(1990)では、既存の製品に関するアーキテクチャ知識[15]が、組織における①コミュニケーション・チャネル（comunication channel）や②情報フィルタ(information filter)、③問題解決のあり方（problem solving strategy）に内面化されてしまうので、それを変更することが困難であるということが挙げられている[16]。既存企業は製品アーキテクチャの変化を既存のアーキテクチャ知識に基づいて解釈するのに対して、新規企業は、従来のアーキテクチャ知識にとらわれることなくアーキテクチャの変化を新たな視点で解釈することができるという大きな違いがあり、この違いによって、アーキテクチャの変化への対応の違いが生じていると主張されているのである。しかし、Henderson and Clark(1990)では、新たな製品アーキテクチャが、企業内でどのようなプロセスを経て生み出されてくるのかということについては十分に議論されていない。

[14] Henderson and Clark(1990)では、このイノベーションは、しばしば、特定の構成部品における変化（形状や大きさなどに関する変化であって、当該部品に用いられている技術自体は基本的には変化しない）によって引き起こされるとされている。

[15] Henderson and Clark(1990)では、製品をいくつかの物理的な部品から構成されるシステムであると捉えて、製品開発活動を通じて生み出される知識が、①構成部品（コンポーネント）に関する知識（component knowledge）と②アーキテクチャ知識（architectural knowledge）の二つに分類されている。前者は、各構成部品に用いられている知識のことであり、後者は各構成部品をいかにしてひとつの製品へとまとめあげていくかに関する知識のことである。

[16] Henderson and Clark(1990)では、組織における知識や情報処理のあり方が、構成要素間の結びつき方（つまり、製品アーキテクチャ）を反映するようになる一方で、企業がそれまで辿ってきた歴史や文化によって、アーキテクチャ知識やコンポーネント知識の形成が影響を受けるとされている。

2.3.2 技術の変化による製品アーキテクチャの変化

楠木・チェスブロウ（2001）では、HDD におけるコア・コンポーネントであるヘッドの要素技術の変化に注目して、コンポーネントの相互作用に関するインテグラルな知識の蓄積（文脈依存的で試行錯誤によって漸進的に蓄積されるもの）が進むことによってモジュラー化が引き起こされ、一方、コンポーネントに用いられている技術が大きく変化することによって、製品アーキテクチャは統合型へとシフトするということが述べられている。しかしながら、「企業が製品アーキテクチャを変化させる」プロセスそのものは分析されていない。

さらに、楠木・チェスブロウ（2001）では、製品アーキテクチャが統合型からモジュラー型へとシフトするような場合には、組織は統合型であり続けようとするけれども、製品アーキテクチャはモジュール化していくため、組織戦略と製品アーキテクチャとの間で不適合が生じてしまうとされており、これは統合型組織の陥りやすい罠であると述べられている。逆に、製品アーキテクチャがモジュラー型から統合型へとシフトする場合には、「バーチャル組織のままであり続ける結果、製品を構成する要素間の新しい相互依存を理解する知識や経験を欠いてしまい、重要なイノベーションの利益機会をみすみす逃してしまう[17]」ことになるが、これは「モジュラリティの罠[18]」(楠木･チェスブロウ、2001)と呼ばれている。

2.4 製品アーキテクチャに関する既存の議論における限界

既存の製品アーキテクチャに関する議論では、主として、①実現した「機能－構造」関係としての製品アーキテクチャ（Ulrich、1995）や、②コンポーネント技術の変化によって引き起こされる製品アーキテクチャの変化（Abernathy、1978; 楠木・チェスブロウ、2001; 柴田他、2002; 新宅、1994）、③提供しようとする製品機能の変化によって生じる製品アーキテクチャの変化（Henderson and Clark、1990）、④製品アーキテクチャのタイプに適合するような組織のあり方（Sanchez and Mahoney、1996; Fine、1998; 藤本、2001; 楠木・チェスブロウ、2001）という四つの論点が示されてきた。

これらの議論では、確かに、製品アーキテクチャは企業の主体的な設計活動を通じて生み出されてくるものであるという前提が置かれているけれども、実際に研究する際には、「製品アーキテクチャの変化」を組織が適応するべき一種の「環境の変化」として捉えているのだと考えられる。例えば、楠木・チェスブロウ（2001）における、「モジュラー型の製品ア

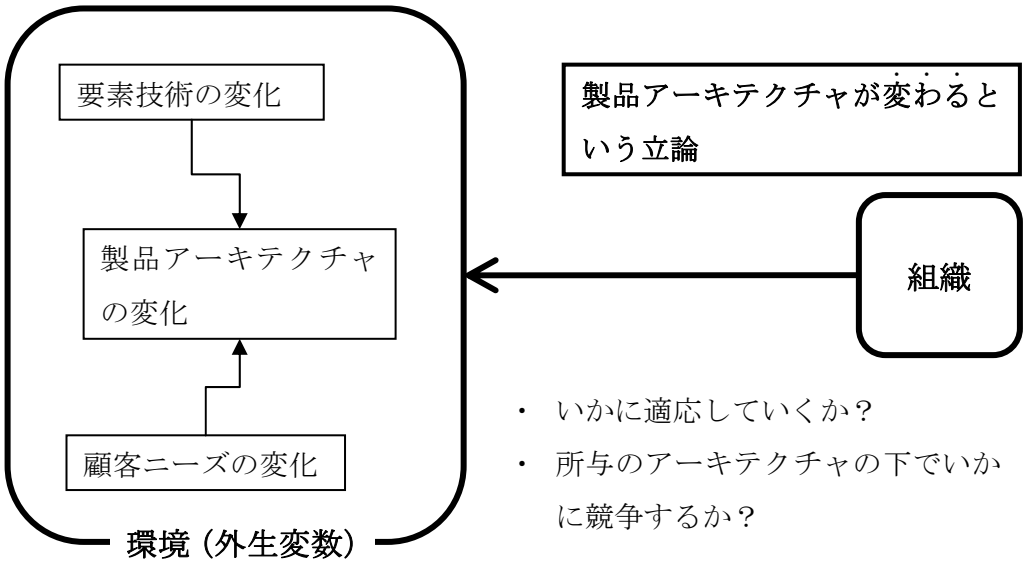
[17]楠木・チェスブロウ(2001), 前掲, p.270

[18] これが統合型企業の罠よりも「悪質」である理由として、バーチャル組織にとっては、①イノベーションの機会が事前に明確であること、②モジュラーイノベーションを体現したコンポーネントや要素技術を市場を通じてすぐに入手できること、③統合型アーキテクチャへのシフトが、モジュラー型へのシフトに比べて一般的に急速に起きるということが挙げられている。

ーキテクチャに対しては、モジュラー型の組織戦略が適している」という主張は、製品アーキテクチャを、組織が適応するべきある種の「環境」として認識した上で、それがモジュラー化した状況下で、どのような組織のあり方が適しているのかという観点からなされているものであり、製品アーキテクチャをモジュラー化するためには組織においてどのような相互調整が行われる必要があるのかということについては明らかにされていないのである。このことを突き詰めて言えば、既存の製品アーキテクチャに関する議論では、製品アーキテクチャにおいてどのようなタイプの変化が生じたときに、どのようなタイプの組織が適応できるのかということについて言及されているのであり、これは個体群生態学モデル（population ecology model）と類似した思考様式であると考えられる（Hannan and Freeman、1977）。

図 1 に示されているように、既存の製品アーキテクチャに関する議論においては、製品アーキテクチャが変わるといった場合には、その変化を所与として、その下で企業がどのような行動をとっているのかということや、そのような状況に適した行動とはどのようなものなのかということに関する考察がなされている。これらの議論では、「モジュール化した製品分野においていかにして競争していくのか」という問いかけや、「製品アーキテクチャの変化に企業はいかにして適応すればよいのか」という問いかけに対して、多くの有益な解答が示されてきた。しかしながら、この観点の下では、製品アーキテクチャと組織を両方とも変えるということを行っている実際の企業が、一体どのように行動すればよいのかということについて参考となる示唆を得ることが難しい。したがって、本論文では、製品アーキテクチャとは企業が主体的に変えていくものであるという観点のもとで、製品アーキテクチャを創出・選択するプロセスに注目して議論を進めていく。それによって、製品開発活動をより有効かつ効率的に行うための一つの視座を提供することができるのではないかと考えられる。

図 1：既存の製品アーキテクチャに関する議論における焦点



出所）筆者作成

3. 事例研究：デジタル複合機におけるファームウェアの開発

3.1 事例研究の方法

本事例研究は、2005 年に複数回にわたって行われたリコーのデジタル複合機のソフトウェア技術者に対するインタビュー[19]やリコー社内資料、『Ricoh Technical Report』、リコー社史編集委員会編（1996）等に基づいている。本論文における研究方法として技術者へのインタビューに基づく事例研究を採用した理由は、この分野での研究の蓄積がそもそも少ないので、まずは実際の現象を観察して情報を収集し、そこから何らかの示唆を得ることを通じて、今後、一般的な仮説の構築や議論を展開していくための足がかりを得ることを志向したためである。また、デジタル複合機を事例研究の対象とした理由としては、①ハードウェアとファームウェアの両方を用いていながらも、ファームウェアの果たす役割が重要な製品であること、②アナログ方式からデジタル方式へと変化し、多機能化を進めていった代表的な製品であり、増大していく機能を一台でまとまりよく提供するための取り組みが長年行われてきたこと、③1990 年代前半において既にファームウェアの規模の増大が顕著な製品であることが挙げられる。

[19] 2005 年 9 月 20 日（3 時間）、2005 年 10 月 11 日（3 時間）、2005 年 10 月 24 日（4 時間）、2005 年 11 月 29 日（同日付で回答いただいた電子メールによる、質疑応答および内容確認）の計 4 回にわたって行われたインタビュー。

事例の考察期間は、リコー初の普及型デジタル複合機である「IMAGIO 320/420」が発売された 1987 年から、「IMAGIO Neo 350/450」が発売された 2001 年までとする。その間に、「IMAGIO 320/420」(1987 年)、「IMAGIO MF530」（1991 年）、「IMAGIO MF150」（1993 年）、「ASAP アーキテクチャ」（1994 年頃）、「NAD アーキテクチャ」（1998 年頃）、「GW アーキテクチャ」（2001 年）というように、六つのファームウェア・アーキテクチャが開発された。

3.2　デジタル複合機の製品アーキテクチャ[20]

今となっては、デジタル複合機といった場合、複写機能や FAX 機能、印刷機能、読み取り機能、ネットワーク機能などといった多くの機能が一台で提供されている。しかし、当初からこれらの機能が一台で提供されていたのではなくて、差別化競争の過程において、どのような機能をどのような構造で実現するかということに関する試行錯誤を通じて、実現可能な機能を増大させてきたのである。その際には、これまで別々の製品で提供されていた機能を一台でまとまりよく提供するために、各機能を担っている一連のファームウェア・コンポーネントをどのようにつないでいくのかということが重要な課題となったのである。

複写機業界では、カラー化や多機能化が必要となるということについては 1980 年代半ばには明らかになっていた。リコーにおいて、従来異なる製品によって提供されていた機能を、一台でまとまりよく提供できるようなデジタル複合機（「IMAGIO Neo 350/450」）が実現されるまでには、1987 年から 2001 年までの間に、五度にわたってファームウェア・アーキテクチャが変化してきた。ファームウェア開発の困難さが日に日に増大していくという、デジタル化された製品を開発している多くの企業が近年直面している問題に、リコーは既に 1990 年代前半に直面して、それを解決するための試みを行ってきたのである[21]。

3.2.1　アナログ方式からデジタル方式への転換

1970 年代初期の複写機には光源の光量制御にアナログ回路、シーケンス制御にリレーが使われていた。その後、システムの高精度化・複雑化にともなって、それらの制御はマイコンとファームウェアによる制御に取って代わられた。マイコンを用いた制御の特徴は、①複雑な制御が可能なこと、②プログラムを組む事により要求に応じた制御が容易であること、③システム設計の後半段階で発見されたハードウェアの不具合を、ファームウェアによって修正可能なことなどである。これらの利点から、マイコンとファームウェアを用いた制御が多用されるようになったので、ファームウェア開発の重要性および困難性が増大することに

[20]筆者インタビューやリコー社内資料、リコー社史編集委員会編(1996)に基づく。
[21] リコーにおける複写機事業の略歴については補論 1 に示されている。

なった。

1990 年代半ばまでは、アナログ複写機が主流であったが、その後、1990 年代後半からデジタル複写機・複合機が急速に普及し始めた[22]。デジタル化が行われる際には、カラー化や多機能化を進めていくことが重要課題とされていた。アナログ方式では、カラーコピーする場合、十分な品質を出すことができず複写速度も遅かったが、デジタル方式ではこれらの問題点を改善することができたのである。

アナログ方式における電子写真プロセスは、光源から原稿に対して光を直接当てて、反射光を屈折させて感光体ドラムに像を形成しトナーを吸着させて、それを紙の上に熱を加えて圧着することによって、複写物を作るというものである[23]。一方、デジタル方式の場合には、感光体ドラムに吸着したトナーを熱で紙に圧着するというプロセスについてはアナログ方式と同様であるが、原稿からの反射光を CCD で読み込んで画像データとしてメモリ（またはＨＤＤ）に蓄積して、そのデータに基づいて半導体レーザによって感光体ドラムに照射して像を形成するという部分が異なっている。デジタル化によって、従来はなかったコントローラ部分が設けられて、製品全体がエンジン[24]部分とコントローラ部分の二つに大別されるようになった。

以上のように、デジタル複合機は、①ハードウェア（機械と電子回路）と②ファームウェア（デジタル複合機に組み込まれているソフトウェア）の両方によって構成されていて、両者が一体となってさまざまな機能が提供されている。アナログ複写機との主な違いとして、①ファームウェアによる処理が格段に増えていることと、②FAX やプリンタ、スキャナなどといった複数の機能が一台で提供されている（複合化）こと、③エンジン（駆動部分）とコントローラの二つに大きく分離されていることが挙げられる。

このように、ファームウェアの利用される程度が高まることによって、製品アーキテクチャのモジュール化が進むということに関しては、これまでいくつか議論がなされてきている。柴田他(2002)では、NC システムにおいてソフトウェアが用いられることによって、製品アーキテクチャのモジュール化が進んだということが述べられているが、ファームウェアのアーキテクチャがどのように変化してきたのかということについては十分に分析されていな

[22] 1999 年には、デジタル複写機・複合機の国内出荷台数とアナログ機の国内出荷台数とが逆転した（経済産業省『機械統計年報』各年版）。

[23]原稿に光を当てて反射光を屈折させる光学系の部分や、帯電→露光→現像→転写→定着という一連のプロセスを担う機構系の部分とが絶妙なタイミングで機械的に連動することによって実現される。これらのプロセスを実現するためには、光学、化学、物理学、電磁気学などの様々な分野の技術が必要であり、高度な技術蓄積が必要とされる。

[24]特に、印刷機能をつかさどっているプリンタを指して「エンジン」と呼ばれることもあるが、さらに、動作する部分としてスキャナについても「スキャナエンジン」と呼ばれることもある。

い。また、加藤（2002）では、HDD の事例に基づいて、デジタル化されている製品においてファームウェアが果たしている役割として、ハードウェア・コンポーネント間の相互依存関係にまつわる問題を自らに集中させて問題解決を行うことによって、各ハードウェア・コンポーネントの独立性を維持・促進しているということが挙げられており、このことによって製品アーキテクチャが統合型へとシフトしてしまうことが防がれている[25]と述べられている。さらに、新宅他（2006）では、多くのエレクトロニクス製品において MPU とファームウェアが用いられて、そこに部品間のすり合わせノウハウが閉じ込められることによって製品アーキテクチャのモジュラー化が進展すること、および、それが企業間で共有されることによって、後発企業のキャッチアップが誘発されるということが指摘されている。

これらの議論では、ファームウェアが製品アーキテクチャのモジュール化を促進する役割を担っているということが示されているが、そのようなファームウェアがどのようにして開発されてきたのかということについては分析されていない。また、個別のハードウェア・コンポーネントに一対一対応したファームウェアについて分析されているにとどまっており、製品に組み込まれているファームウェア・コンポーネント間のつながりがどうなっているのか、そしてファームウェア自体がどのような設計思想の下で設計されているのかという、「ファームウェア・アーキテクチャ」という観点からの議論は行われていない。したがって、本事例研究では、ハードウェアとファームウェアの両方に焦点を当ててデジタル複合機の製品アーキテクチャについて考察する。その際には、①ファームウェアを利用することによって、製品設計活動において解決しなければならない、機能間連携や部品間連携といった相互依存関係の問題が、ファームウェアの設計活動に集中していくことや、②当該ファームウェア内部における相互依存性のあり方が、一連の製品開発活動を通じてどのように変化してきたのかということに注目していく。

3.2.2　ハードウェアに注目した場合の製品アーキテクチャ

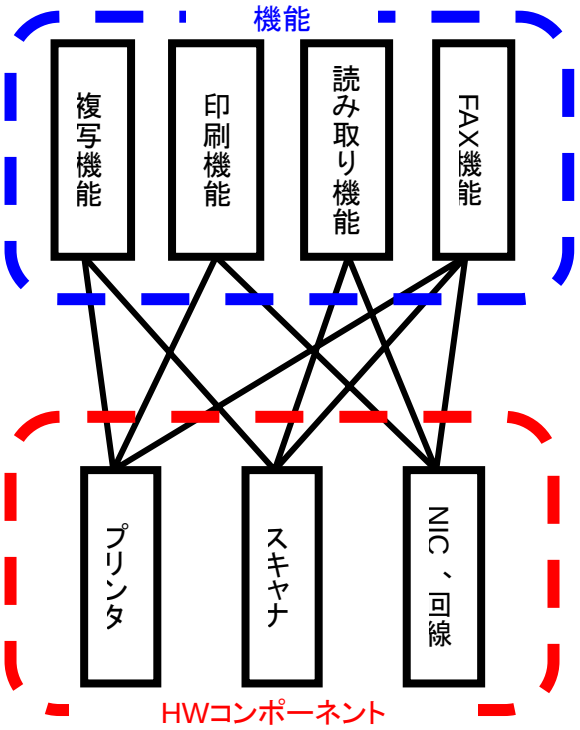
デジタル複合機の製品アーキテクチャをハードウェアに注目して記述すると、図 2 に示されているように、複写機能についてはプリンタとスキャナの連動、FAX 機能についてはプリンタやスキャナ、電話回線などの連動、読み取り機能についてはスキャナ、印刷機能についてはプリンタ[26]というように、各々の製品機能とハードウェア・コンポーネントとが一対多

[25] 加藤(2002)では、ファームウェアのこのような役割のことを「モジュラリティ・ドライバ」と呼んでいる。

[26] ネットワークスキャナやネットワークプリンタとして利用する場合には、NIC（Network Interface Card）も必要となる。

対応している[27]。しかし、これらのハードウェア・コンポーネントのみでは複写機やプリンタ、スキャナとして機能するわけではない。実際に印刷機能や読み取り機能を実現するためには、これらのハードウェア・コンポーネントを制御するためのファームウェア・コンポーネントが必要である。

図 2　デジタル複合機のハードウェア・アーキテクチャ



出所）筆者作成

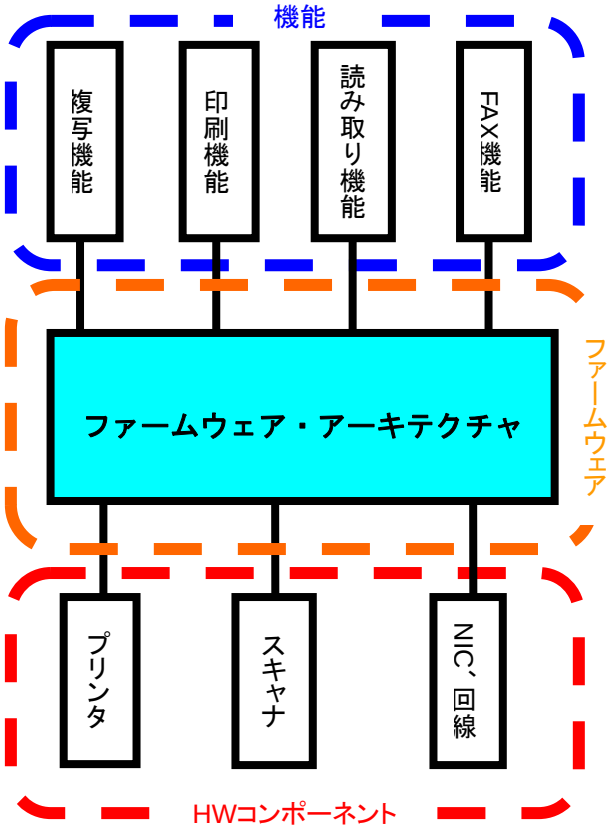
3.2.3　ファームウェアにも注目した場合の製品アーキテクチャ

デジタル複合機のファームウェア・コンポーネントは、図 3 に示されているように、複写機能に対してはコピーアプリケーション、FAX 機能に対しては FAX アプリケーション、印刷機能に対してはプリンタアプリケーション、読み取り機能に対してはスキャナアプリケーションが一対一対応している。これらのアプリケーション間に共通している機能をまとめたものとしてミドルウェア（サービス）層があり、それぞれのタスク間での調整を担うファームウェア・コンポーネントとして OS（主にリアルタイム OS）が利用されている。

[27] アナログ単機能の複写機の製品アーキテクチャは、単純化すれば、製品自体が複写という機能を提供するための専用機であることから、製品と機能が一対一対応していると見ることもできる。しかし、構成部品レベルで見た場合には、多数の部品が機械的に複雑に連動することによって複写機能が提供されるということから、アナログ単機能機の製品アーキテクチャの統合度は高いと考えられる。

図 3 で示されているように、ファームウェアにも注目して製品アーキテクチャを記述すると、ファームウェアを利用することによって、製品全体としてモジュラー化が進んでいるように見える。しかし、デジタル複合機のアーキテクチャについて考える場合には、ファームウェアのアーキテクチャがどうなっているのかについても考慮しなければ、製品アーキテクチャを十分に捉えることはできないと考えられる。その理由は、一連のファームウェア・コンポーネントをどのように設計するのか（ファームウェア・アーキテクチャ）によって、どの機能をどのコンポーネントでどのようにして実現するのかが決まり、それによってデジタル複合機のアーキテクチャが大部分決まるからである。

図 3　デジタル複合機におけるファームウェアの役割



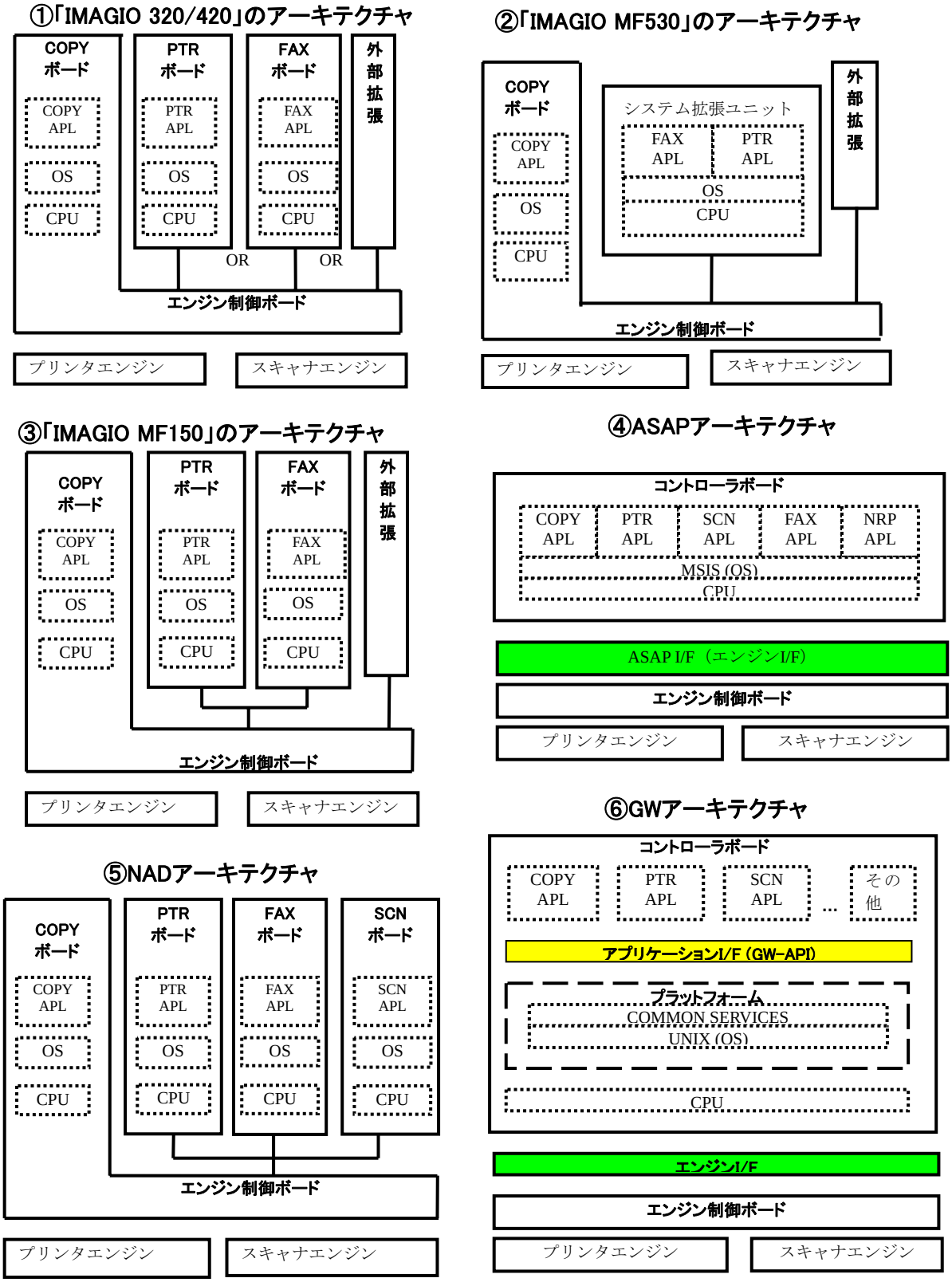
出所）筆者作成

3.3　株式会社リコーにおけるファームウェア開発への取り組み[28]

リコーのデジタル複合機におけるファームウェア・アーキテクチャの変遷を模式的に示すと図 4 のようになる。以下本節では、リコーのデジタル複合機におけるファームウェア開発の取り組みを四つのフェーズに分けて順に考察する。

[28] それぞれのファームウェア・アーキテクチャの詳細な特徴については補論 2 に示されている。

図 4　リコーにおけるデジタル複合機のファームウェア･アーキテクチャの変遷（簡略図）



出所）筆者インタビューおよびリコー内部資料に基づき筆者作成

3.3.1　ASAP アーキテクチャ開発以前

ASAP アーキテクチャを開発する以前（「IMAGIO 320/420」や「IMAGIO MF530」、「IMAGIO MF150」）は、コピーや FAX、プリンタを開発する事業部があり[29]、ファームウェアの開発は別々に進められていた。ファームウェアを開発する際には、これらの部門間での相互調整がほとんど行われておらず、部門ごとに使われている言葉が違うといわれるほどにまで、部門特有の文化が形成されていた。

また、「IMAGIO 320/420」や「IMAGIO MF530」、「IMAGIO MF150」におけるファームウェア・アーキテクチャの開発を進めていたのは複写機部門であり、複写機部門出身のエンジニアがリーダーとなってファームウェア・アーキテクチャが開発されていたのである。そのため、これらの製品のファームウェア・アーキテクチャは、コピーをベースとしたものになっていたのだと考えられる。

3.3.2　ASAP アーキテクチャの開発と失敗

一方、ASAP アーキテクチャは、プリンタのエンジニアをリーダーとして、コピー機能を必ずしも中心とせずに各機能を並列に据えるという、従来とは異なる志向性に基づいて開発が行われた。ASAP アーキテクチャにおいては、従来の機種では考慮されてこなかった、①ファームウェア・コンポーネント間での共通化と、②複数機種間でのファームウェアの共通化という、二種類の共通化が目指されていた。これによって、ファームウェアの開発リードタイムや開発コストおよび品質が向上すると期待されていた。

しかしながら、これまで各機能を実現するためのファームウェアは、先述のように、他の部門との相互調整をほとんどすることなく開発されてきたので、それらを一つの OS 上で適切に機能するように統合することが困難であった。そのような統合作業を行おうとしても、部門間の調整を上手く行うことができなかったのである。特に問題となったのは、ASAP アーキテクチャにおけるリコー独自の OS（MSIS）は、各部門との連携を十分にとらずに開発されたため、各部門に対して OS のインタフェースに関する情報の開示が遅れてしまったことである。その結果、各部門がそれぞれ独自に OS の開発まで含めてアプリケーションの開発を行ってしまい、ひとつの OS の上に複数の機能を並列させるという当初の目論見が破綻

[29] 株式会社リコー『有価証券報告書』各年版や『リコー・ファクトブック』各年版に記載されている組織図、および『Ricoh Technical Report』各年版に記載されている論文執筆者の当時の所属部署から、複写機（複合機）、FAX、プリンタ（スキャナも含む）を開発する組織は、1986 年から 2000 年までは、それぞれ別の事業部であった。2001 年には複写機事業部とファクシミリ事業部は統合されたが、依然としてプリンタ事業部は別であった。その後、2005 年 4 月 1 日付けで、MFP 事業本部、LP 事業部、GJ 事業部という組織体制となっている。

してしまったのである。例えば、FAX 部門は OS まで含めて独自に開発を進めてしまい、MSIS との調整を行うために開発工数が増えてしまった。また、各機能がバラバラに完成して、それに伴って各機種も異なるタイミングで発売されることになったので、全体のシステムチェックが不十分であり、それぞれ異なる時期に発売した機種間で OS までもが異なってしまった。システムチェックによってバグが発見された場合に、それを修正するためのデバッグツールも独自で開発しなければならなかったので、さらに開発工数が必要となってしまった。結局、最初に製品化できたのは、コピー機能のみを搭載したベーシック機であり、複数機種を同時に開発して発売することはできなかった。さらに、機種間での OS 共通化やアプリケーションソフトの互換性も実現することができなかったのである。

このような失敗が生じた原因としては、①開発すべきファームウェアの規模の増大に比して、それを開発するためのエンジニアや資金といった開発資源が不足していたことや、②ASAP アーキテクチャを開発するためには、部門間の緊密な相互調整が必要であったにもかかわらず、それが十分に行われなかったこと、③当時のコントローラに用いられていた CPU の処理速度が不足していたということが考えられる。

3.3.3 過去のアーキテクチャへの回帰

このような ASAP アーキテクチャにおける失敗を受けて、NAD アーキテクチャの開発を行う際には、コピー部門がリーダーとなって開発が進められた。そこでは、ファームウェア・コンポーネント間の共通化を行うよりも、むしろ、既に実際の機種で用いられたことのあるファームウェア・アーキテクチャ（「IMAGIO MF150」）に回帰して、各部門において OS からアプリケーションまで開発するということが行われた。ASAP アーキテクチャから NAD アーキテクチャへの変化は、旧来のアーキテクチャに回帰することによって、新たにファームウェア・アーキテクチャを開発するために発生する部門間での相互調整タスクを低減しようとするための対応であった。

しかし、このような対応は重大な問題を招くことになった。すなわち、NAD アーキテクチャが開発された時点では、既にコピーや FAX、プリンタ、スキャナといった機能を製品に取り入れただけでは不十分であり、オフィスなどのネットワークにつながって、データや情報のやり取りを行う上で重要な役割を果たす事務機器として機能するという「ネットワーク化」の要求に対応することが求められていたのだが、NAD アーキテクチャでは、その要求には十分に対応できなかったのである。さらに、このアーキテクチャの下では、利用するハードウェア資源（たとえば CPU の数）が多くなるため開発コストが増大したり、複数機種を開発する際には、必要となるファームウェアをその都度開発しなければならず、過去に開

発されたファームウェアを流用したり、機種間で共通のファームウェアを利用することも困難であるという問題点があった。

3.3.4　GW アーキテクチャの開発

GW アーキテクチャは、1998 年に GW-PT（プロジェクトチーム）の発足とともに開発が開始されたが、このような、ファームウェア・アーキテクチャを開発するためのプロジェクトチームが、リコー社内で設置されたのは初めてである。GW アーキテクチャの開発を行う際には、社内から広くエンジニアを連れてきて、ASAP アーキテクチャの失敗を生かしつつ、以前よりも多くの開発資源が投入された。

NAD アーキテクチャの下では、各機能を実現するためのファームウェアを別々に開発していたので、各機能で必要となるアプリケーションや OS について、それぞれの組織が独自に開発を進めてしまい多くの重複部分ができていた。これに対して、GW アーキテクチャの開発を行う際には、各部門を統合するために GW-PT が設けられて、これにより共通部分（プラットフォーム）の発見と切り出しが可能となったのである。

さらに、GW アーキテクチャをスクラッチから完成させるには、ソースコード量が多いので、NAD アーキテクチャの下で開発されたファームウェアの一部が GW アーキテクチャでも再利用された。GW アーキテクチャの開発に際しては、10 人以上のグループで OS の開発が行われて、部門間での調整も行われていたので、共通部分の洗い出しをうまく行うことができた。また、オープンな UNIX OS に準拠しているので、オープンなデバッグツールを利用することも可能になった。このように、GW アーキテクチャは、ASAP アーキテクチャにおける共通化の失敗と、NAD アーキテクチャにおける新規機能への拡張性が乏しく機種間での共通化が困難であるという問題を踏まえたうえで、それらを解決するために開発された。

3.4　事例の小括

リコーにおいては、1980 年代後半以降、デジタル複写機・複合機の高性能化や多機能化を推進したことによって、製品に用いられるファームウェアの規模や複雑性が増大し、ファームウェア開発の難しさが露呈した。これを受けて、1992 年から ASAP アーキテクチャの開発が行われたが、前節で述べられているように失敗に終わった。その理由として、ファームウェアの開発体制に注目した場合、製品の多機能化を行う上で中心的な役割を果たしているのが、ファームウェアであるということがリコー社内で十分に認識されていなかったので、ASAP アーキテクチャの開発に必要となる開発資源が十分に投入されなかったということが考えられる。このことは、当時におけるファームウェアの実際の重要度と、その重要性に関

する組織の認識との間にギャップが生じていたということを意味している。

ASAP アーキテクチャ開発の失敗を受けて、次に採られた方策は、ファームウェア開発体制の強化ではなくて、既存のアーキテクチャ（「IMAGIO MF150」におけるもの）への回帰であり、それによって開発されたのが NAD アーキテクチャであった。そこでは、依然として、ファームウェア・アーキテクチャが問題であるということが十分に認識されておらず、対応も不十分であった。このアーキテクチャは、多機能化を進めて他社と競争していく上では不利なアーキテクチャであった。このように、ASAP アーキテクチャと NAD アーキテクチャにおける、二度にわたる失敗を経て、ようやくファームウェアの重要性が社内でも認識されるようになった。実際に、ファームウェア・アーキテクチャを開発するための組織として、GW-PT を設けて対応するようになり、GW アーキテクチャの開発に成功したのである。ASAP アーキテクチャと GW アーキテクチャは、類似した構造をしているが、両者における開発の取り組みは異なっており、それを比較すると表 1 のようにまとめられる。

なぜ、デジタル複合機の開発を行う際には、既にハードウェアとファームウェアの両方に十分な注意を払う必要があったにも関わらず、それができなかったのだろうか。その理由として、リコー社内では「士農工商：メカーエレキーソフト」という風潮が支配的であり、ファームウェアを開発する部隊は、リコーの中ではマイナーな部隊として扱われていたということが挙げられる。これは、デジタル複合機の開発組織がハードウェア主体であるということを意味している。実際に、ファームウェアの開発コストは製品の開発コストとして換算されず、たとえされたとしても製品に搭載される ROM のコストとして扱われていた。

また、一連のファームウェア・コンポーネント間の関係性であるファームウェア・アーキテクチャを決めるという、複数のファームウェア・コンポーネントに関わる決定であるにも関わらず、それを決めていたのはコピーのファームウェアを開発する部隊であった。これによって、部門間の相互調整が希薄となってしまい、無駄なソースコードが開発されることになった。さらに、ファームウェア・コンポーネント間での統合テストを行うためには、多くの開発工数が必要となるので、実際に市場で動作することが確認されているファームウェア・アーキテクチャを、わざわざ変える必要はないという考え方が支配的となり、ファームウェア・アーキテクチャを根本的に見直すことが十分に行われていなかったのである。このように、デジタル複合機の開発組織はハードウェア主体であるため、ファームウェア開発部隊に十分なリソースが供給されなかった。そのため、ファームウェア開発部隊では、ただでさえ人手が足りないのに、部門間の相互調整にまで貴重な人材を割く余裕がないという問題が生じていたのである。

以上本節で考察してきたように、リコーにおけるデジタル複合機のファームウェア・アー

キテクチャの開発において重要であったのは、各ファームウェア・コンポーネント間の共通部分を洗い出して、新たなファームウェア・コンポーネントとして「抽出[30]」（Baldwin and Clark、2000）するということであると考えられる。そのための作業をうまく行うためには、部門間での相互調整が緊密に行われる必要があったのである。この「抽出」作業を最もうまく行うことができたのが、ファームウェア・アーキテクチャを開発するための組織を設けて緊密な部門間調整を行うことが試みられた、GW アーキテクチャのときであった。

表 1 ASAP アーキテクチャと GW アーキテクチャにおける取り組みの比較

	ASAP アーキテクチャ	**GW アーキテクチャ**
開発資源投入量	少量	大量
開発体制	一事業部内での 小規模な取り組み	GW プロジェクトチーム
開発リーダー	プリンタ	コピー
OS 開発	単独で開発 独自 OS 独自デバッグツール	グループ開発（10 人以上） UNIX OS パブリックなデバッグツール
社内での 情報共有 （部門間の やり取り）	遅れた FAX 部隊が OS の機能まで作ってしまった。 →機能の共通化の失敗 →同時リリース不可能	積極的に進めた →ドキュメント化 →ファンクション毎にグループ制採用
アプリケーションと OS との切り分け	アプリケーションと OS が渾然一体	アプリケーションと OS をインタフェースで分離

出所）インタビューに基づき筆者作成

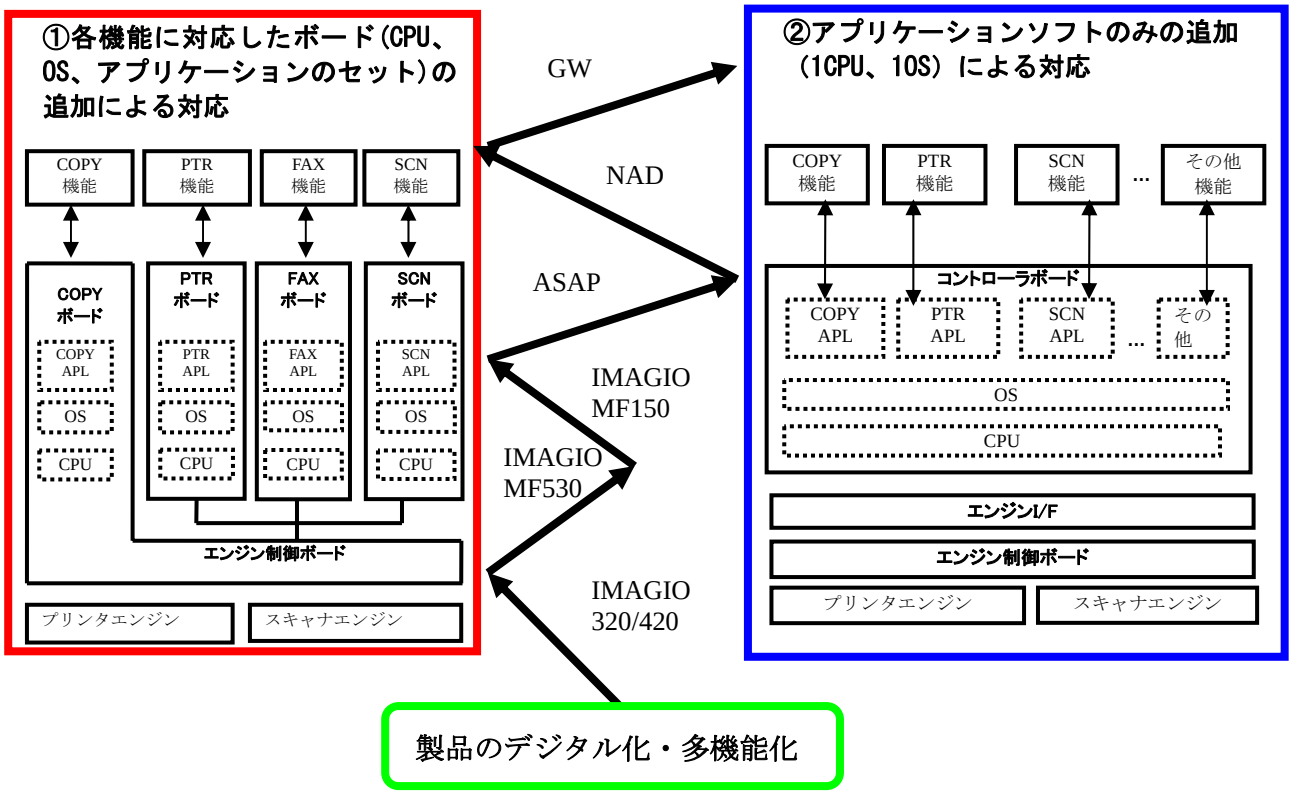
[30] Baldwin and Clark（2000）では、モジュラー型の製品アーキテクチャのメリットとして、①複雑なシステムを、複数のモジュールに「分離」することによって、それらのモジュール間での組み合わせが増えるため、製品のバリエーションを増やすことができる、②複数のモジュールに分離されることによって、あるモジュールにおける改良を他のモジュールとは独立して行うことができ、その改良の成果を既存のモジュールと「交換」することによって、製品の全体としての性能を向上させることができる、③既存のシステムに対して新たなモジュールを「追加」したり「削除」することによって、顧客のニーズの変化に事後的に対応可能となる、④複数のモジュール間で共通している部分を「抽出」することによって冗長部分のコストを削減できる、⑤あるシステムにおけるモジュールを他のシステムに「転用」することによって、全てのシステムごとにスクラッチから設計しなくても良いので設計コストを削減することが可能になる、ということが挙げられている。

４．議論：製品アーキテクチャの選択プロセス

4.1　デジタル複合機におけるファームウェア・アーキテクチャの揺れ動き[31]

前節における事例研究から、リコーにおけるデジタル複合機のファームウェア・アーキテクチャは、図 5 に示されているように、「多機能化」を進めていく際に①アプリケーションや OS、CPU をひとつのセットとして、そのボードの追加によって対応するか、②ひとつの OS 上で複数のアプリケーションを動作させることによって、機能の追加や削除の要求にアプリケーション・レイヤにおいて対応するという、二種類のアーキテクチャ間での揺れ動きが起きていた。

図 5 ファームウェア・アーキテクチャの揺れ動き



出所）福澤・立本・新宅（2006）の図 5 を加筆修正

前者のアーキテクチャは、各機能の「部分最適」を志向するものであり、このアーキテクチャでは CPU の処理速度への要求が小さく、各機能のパフォーマンスは最適化されるのだが、冗長性が大きいため部材費用が高くなったり、デジタル複合機としての「統合機能」を実現する上では不向きであるという問題があった。一方、後者のアーキテクチャは、各機能の「全体最適」を志向するものであり、デジタル複合機に求められる統合機能を実現するの

[31]筆者インタビューやリコー社内資料、リコー社史編集委員会編(1996)に基づく。

には向いており、部材費用を削減することもできるが、高速処理が可能な CPU が必要とされたり、各ファームウェア･コンポーネントを開発する際に、機能間における多くの相互調整が必要とされるという問題があった。このように、製品の多機能化に対応するために二つのファームウェア･アーキテクチャの間で揺れ動きが生じていたのだが、最終的には後者のアーキテクチャに落ち着いた。このことは、デジタル複合機においてある機能を実現するためには、各機能に対応したボードをいちいち追加していくというアーキテクチャよりも、アプリケーション・レイヤにおいてある機能を実現するためのアプリケーションソフトを「追加」したり「削除」できるアーキテクチャのほうが、機能の拡張性や開発コスト（ハードウェアおよびファームウェア)、統合機能の実現可能性の点で優れていたということを意味していると考えられる。

これら二つのアーキテクチャの間での揺れ動きは、リコーがデジタル複合機の製品アーキテクチャとしてより適したものを創出して選択していくための一連のプロセスであったと考えることができる。まず、アナログ方式からデジタル方式への転換によって、ハードウェアの側面から記述される製品アーキテクチャのモジュラー化が進んだことにより、製品アーキテクチャの決定領域の大部分が、ハードウェア領域からファームウェア領域へと移行し、ファームウェアのアーキテクチャが、製品システムの挙動や性能の大部分を決めるようになった。デジタル化された当初は、複数の機能を単に寄せ集めたものが複合機として提供されていたのだが、デジタル複合機での競争が厳しくなるに伴って、各機能をいかにうまくまとめあげてひとつの製品に仕上げていくのかという点で、差別化が行われるようになったのである。このように、多機能化や機能の統合化を進めるためにファームウェアの大規模化や複雑化が進んだことによって、一連のファームウェア・コンポーネントをいかにして設計していくのかということが重要な問題となったのだが、このことは、製品性能や機能の大部分を、ファームウェア・アーキテクチャや個々のファームウェア・コンポーネントの出来の良し悪しが決めるようになってきたということを意味していると考えられる。つまり、製品開発における問題解決活動（相互調整タスク）の大部分がファームウェアの開発に集中するようになり、「ファームウェア・アーキテクチャの選択≒製品アーキテクチャの選択」という関係が強く成立するようになったのである。

本研究の事例は、事後的に見れば、リコーが採りうるアーキテクチャとして、既に上記の二種類のアーキテクチャが存在していたにも関わらず、ハードウェアの性能不足によって単に後者のアーキテクチャに到達するのが遅れたかのように解釈することもできる。しかし、むしろ、各アーキテクチャを開発していた当時のリコーでは、どちらが優れているのかということを明確に決める基準は存在しておらず、二つのアーキテクチャの間を揺れ動きながら

各々のアーキテクチャのメリットとデメリットについて学び、さらには、それぞれのアーキテクチャを成功させるための知識を学習してきたという解釈のほうが妥当であると考えられる。その理由は、両方のアーキテクチャを実際に開発する経験を経ることによって、はじめて、どちらのアーキテクチャを選択すればよいのかということが明らかになると考えられるからである。以下では、ファームウェア・アーキテクチャの選択に影響を与える要因について、事例に基づいて考察する。

4.2 ファームウェア・アーキテクチャの選択への影響要因

デジタル複合機における製品アーキテクチャは、ハードウェア（機構および電子回路）の側面から記述すると、モジュラー化が進んだものとして記述することができる。このような、モジュラー化の促進はファームウェアが用いられたことに大きく起因している。しかし、それら一群のファームウェアをどのようなアーキテクチャで設計するのかということは、以前として解決すべき課題として残されていた。このような、ファームウェア・アーキテクチャの選択に対して影響を与えていた要因として、①ハードウェア・コンポーネントの性能（特に、CPU の処理速度）と②製品の捉えかた、③開発組織における部門間調整のあり方の三点が考えられる。

4.2.1 ハードウェアの技術的限界

まず、ハードウェア（CPU）の技術的限界については、デジタル複合機に用いられていたコントローラ CPU のクロック数は、当初数十 MHz 程度であったため、必要とされる性能を維持しながら、複数のファームウェアを並行処理することができなかったのである。その後、CPU のクロック数は向上し続けて、GW アーキテクチャで用いられている CPU のクロック数は数百MHz となり、当時の約十倍まで向上したため、複数の機能を実現するための大規模ソフトウェアであっても並行処理することが可能となった。

しかし、今後も、このようなハードウェア技術の向上によって、ある特定のアーキテクチャに収斂するとは必ずしも断定できない。その理由は、①当該製品で処理したい情報量とハードウェア（CPU）の性能とが適合しているかどうかということが重要であり、両者の追いかけっこは依然として続いていく可能性が高いことや、②仮に両者が適合したとしても、狙ったアーキテクチャを実現するための組織能力が低ければ、当該アーキテクチャの開発に成功することは難しいと考えられるからである。

4.2.2　製品の捉え方

デジタル複合機において、各社の競争の焦点となっているのが、「一台でどれだけ多くのことができるのか」という「多機能化」である。このような多機能化を進めることによって、開発すべきファームウェアのソースコード量も増大していく。一言に「デジタル複合機」といっても、それぞれの製品について細かく見ると、①どのような機能を、②どのようにして一台にまとめていくのかという点で異なっている。そして、この違いがファームウェア・アーキテクチャの違いとして表われているのだと考えられる。このように、製品で提供しようとする機能によって必要となるファームウェア・アーキテクチャが異なるし、逆に、ファームウェア・アーキテクチャによって、当該製品において提供できる機能が影響を受けるということが起きていたのである。

「IMAGIO 320/420」は一台で二つの機能を提供することによって、「多機能化」を行うための第一歩となったが、まだこの時点では、「コピー」に何かもう一つ機能をくっつけたもの、すなわち、「○○もできるコピー機」として捉えられていたのである。「IMAGIO MF530」は、多数のアプリケーション機能を備えたマルチファンクション機であり、当時リコーでは、複合機から「融合機」への発展を遂げるための機種として位置づけられていた。つまり、単にコピーに対してFAXやプリンタといった機能がくっつけられているというのではなくて、各機能の「融合」を行うということが目指されたのである。このことは、システム拡張ユニットにおいて、各アプリケーションがひとつのOS上に並列に配置されているということに表れている。しかしながら、「IMAGIO MF530」のファームウェア・アーキテクチャは、このシステム拡張ユニットをメインボードに接続するというものになっているので、依然として、他の機能はあくまでも「コピーの付随機能」に過ぎなかったのだと考えられる。このように、「IMAGIO MF530」では、各機能の「融合」ということがいわれているけれども、実際には、依然として「○○もできるコピー機」として捉えられていたのである。

「IMAGIO MF150」では、自社のデジタル複合機の中で、FAX機能を中心とした拡張機能がひとつだけの複合機（一複合形態）が圧倒的に多く売れているということを受けて、ターゲット市場として省スペース化が必要な一般小規模オフィスを選択した。そのため、「IMAGIO MF150」では、「IMAGIO MF530」におけるように多くの機能をシステム拡張ユニットとして提供するのではなくて、「顧客の求めない冗長な機能については省く」という考えの下で、コピー機能をベースとして、これにFAXボードやプリンタボードなどの個別の拡張機能を顧客の要求に合わせて追加して販売していた。つまり、「IMAGIO MF150」も「○○もできるコピー機」として捉えられていたのである。

ASAPアーキテクチャでは、各機能の「融合」がもう一度志向された。実際に、「非コピ

ー(NRP: Non-Reprographic)」機ということがいわれており、「○○もできるコピー機」というものから「多機能の事務機器」へという製品の捉え方の変化が起きたのではないかと考えられる。ASAP アーキテクチャでは、エンジン部分とファンクション・コントローラとのインタフェースについては、比較的しっかりと設定されているのだが、アプリケーションと OS とのインタフェースについては、依然としてうまく設定されておらず、デジタル「融合機」のファームウェア・アーキテクチャとしては、依然として不十分であった。NAD アーキテクチャでは、一台で多機能を提供するという、「多機能化」を志向していることには変わりないが、「IMAGIO 320/420」や「IMAGIO MF530」、「IMAGIO MF150」におけるように、「○○もできるコピー機」へと回帰している。NAD アーキテクチャは、当時重要になっていた「ネットワーク化」への対応をする上では不適切なものであり、他の機能への拡張性も低かった。

GW アーキテクチャは、ASAP アーキテクチャと同じように、非コピーベースのアーキテクチャであり、各アプリケーションが並列に配置されている。また、GW アーキテクチャは複数の機種間で共通利用されている。GW アーキテクチャでは、NAD アーキテクチャにおけるように、拡張アプリケーション毎にオプションボードが必要ではなくて、機能の追加や変更についてはアプリケーション・レイヤで対応可能である。この GW アーキテクチャによって、「ネットワーク化」も含めた「多機能化」が最もうまく実現されるようになった。

以上のように、リコーにおいて、デジタル複合機という製品の捉えかたが、「○○もできる 複写機」(「IMAGIO 320/420」や「IMAGIO MF530」、「IMAGIO MF150」、NAD アーキテクチャ)というものから、「オフィスで生じる様々な情報を処理するための機能を搭載した事務機器」(ASAP アーキテクチャ、GW アーキテクチャ)というものへと洗練されてきたと考えられる。しかしながら、後者のような捉え方に基づく製品をうまく実現するためには、ASAP アーキテクチャは依然として不十分なものであり、GW アーキテクチャにおいて一応の完成を見せたのである。従来は、コピー部門がファームウェア・アーキテクチャを開発していたので、デジタル「複合機」と言いながらも、実際には「○○もできる複写機」として製品が捉えられており、コアとなる機能に縛られたファームウェア・アーキテクチャの開発が行われてしまったのである。製品の多機能化を進めていく際には、このようなコア機能を最適に動作させられるようなファームウェア・アーキテクチャが、むしろ足枷となっていたのである。

4.2.3 開発組織における部門間の相互調整のあり方

開発組織における部門間の相互調整のあり方としては、従来は、コピーや FAX、プリンタ、

スキャナといった各部門で必要となるOSやアプリケーションを開発するというように、各部門でバラバラにファームウェアの開発が行われてきたが、最終的には、ファームウェア・アーキテクチャの開発を行うための専門組織を設けて、緊密な相互調整が行われるようになった。しかし、このような部門間の相互調整が行われるのが遅れた要因として、①そもそも部門間の相互調整をうまく行えなかったということや②ファームウェア・アーキテクチャ開発の重要性について社内で十分に認識されていなかったということが挙げられる。これらの問題は、①そもそもハードウェア主体の開発組織であるということと、②既存のコア部門（複写機部門）がファームウェア・アーキテクチャの開発を行っていたので、他の機能までを含めた「全体最適」の観点からの設計を行うことが困難であったということから生じていたのである。

ASAPアーキテクチャを開発する際には、CPUの処理能力が足りないながらも、ひとつのOS上で複数のアプリケーションを動作させるアーキテクチャを採用した。そのようなアーキテクチャを可能にするためには、部門間の緊密な相互調整が必要であったにもかかわらず、それが不十分であったため、ファームウェア・コンポーネントの「抽出」をうまく行うことができずに失敗してしまった。このように、CPUの処理速度が遅いことと、部門間の調整を行うのが困難であるということから、部門間の緊密な相互調整を行うことによってファームウェア・コンポーネントの抽出を行うよりもむしろ、その相互調整を必要としなくても良いようなアーキテクチャに回帰したのが、NADアーキテクチャであった。しかし、これは当時要求されていた水準の機能の拡張性には乏しいアーキテクチャであった。ASAPアーキテクチャとNADアーキテクチャの失敗は、Lawrence and Lorsch (1967)で言われているような部門間での「分化」の程度が高いのだが、それらをうまく「統合」できなかったということを意味していると考えられる。つまり、ASAPアーキテクチャを開発する以前において採られていた、複写機を中心としたハードウェアの開発を行うのに適した組織の分化と統合のあり方に引きずられてしまったため、ASAPアーキテクチャを開発する際には、ファームウェアまでをも視野に入れた組織のあり方を実現できず、さらにNADアーキテクチャの開発の際には従来の組織のあり方が強化されたのだと考えられる。

これらの二回の失敗を受けて、GWアーキテクチャにおいては、①複数のアプリケーションを並列処理するに足るだけのCPUの処理速度が実現されたことと、②部門間の緊密な相互調整を行うための専門のプロジェクトチームが設けられて、ファームウェア・コンポーネントの抽出がうまく行われたことによって、多機能化の要求にアプリケーション・レイヤで対応できるようなファームウェア・アーキテクチャの開発に成功したのである。

4.3 事例研究から得られた教訓

リコーは、アナログ複写機の時代から大きな市場シェアを占めていて、複写機を本業としてきたので、ファームウェア・アーキテクチャを設計する際には、本業である複写機能を中心としたアーキテクチャが志向されており、開発組織における部門間調整のあり方も複写機開発部隊が中心となっていた。リコーの事例は、ある製品においてコアとなっている機能を開発している一部門が、既存のコアとなる機能を中心に据えたファームウェア・アーキテクチャを開発したため、多機能化により適したアーキテクチャの開発に失敗したということを示している。このような傾向は、既存のコア機能をうまく満たすことのできるようなファームウェア・アーキテクチャを開発するための知識や能力が蓄積されていればいるほど顕著になると考えられる。このようなコアとなる機能に縛られてアーキテクチャを開発するという体制から抜け出して、狙ったアーキテクチャを開発するために必要となる知識を獲得するための試行錯誤のプロセスが、本事例で観察されたファームウェア・アーキテクチャの揺れ動きとして表れているのだと考えられる。本論文で取り上げたようなリコーが直面していた問題は、デジタル化された製品において多機能化を進めている他の多くの企業においても重要になっており、それを解決するために各社とも苦労していると考えられる。ファームウェア・アーキテクチャの開発に何度も失敗しないために、これらの企業が採るべき施策はどのようなものなのか。

リコーの事例研究から示されたように、従来のコアとなる機能に縛られて、それを中心としたファームウェア・アーキテクチャを開発していたのでは、多機能化を行う上で適切なアーキテクチャを開発することができなくなってしまうので、そのような機能に縛られないアーキテクチャづくりが不可欠である。リコーの事例から、多機能化に適したファームウェア・アーキテクチャを開発していくためには、既存の企業内部において、それぞれの機能についての知識を保有したエンジニアが集まって部門間の相互調整を緊密に行うことのできるような組織を設けて、そこで当該アーキテクチャを開発していく必要があるということが示唆された。これを実行するためには、①ファームウェア・アーキテクチャの開発を行うための資源を大量に投入し、②従来のコア機能を開発してきた部門の意見に縛られることなく、当該製品に対する捉え方の見直しを図り、③ハードウェアを開発してきた組織体制を見直して、ハードウェアとファームウェアの両方を開発することを見越した上で、複数部門間での相互調整を緊密に行うことが必要であると考えられる。

5. 結論

5.1 まとめ

本論文では、デジタル複合機におけるファームウェアの開発事例に基づいて、事後的に観察される製品アーキテクチャの変化という現象を生み出すプロセスを、製品アーキテクチャの創出・選択プロセスとして捉えた上で、特にファームウェアを用いた製品における固有の問題として、①ハードウェア主体の開発組織による弊害と、②組織内に散らばっているファームウェア開発のための知識を統合するための組織の取り組みに焦点を当てて議論してきた。

本論文の貢献は、製品アーキテクチャと組織のあり方との静態的な適合関係およびその関係が製品アーキテクチャの変化によって遷移するということに加えて、そもそも、企業がいかにして製品アーキテクチャを変化させてきたのかという、製品アーキテクチャの創出・選択プロセスに焦点を当てて分析されていることにある。既存の議論では、製品におけるファームウェアの利用度が増大することによって、製品アーキテクチャのモジュール化が進むことが示されているが、そこでは、ハードウェア間の相互依存関係がファームウェアというひとつのコンポーネントによって吸収されているということを主張するにとどまっており、ファームウェアのアーキテクチャをいかにして設計していけばよいのかということについては考察されていない。これに対して本論文では、デジタル化された製品の代表例としてデジタル複合機のファームウェアの開発に焦点を当てることによって、もともとアナログ単一機能製品であった「複写機」から、デジタル化を進めながら、従来とは異なるさまざまな機能を取り込んで行くことによって、デジタル「複合機」へと製品が進化していくプロセス、および、当該プロセスにおいてファームウェア・アーキテクチャが揺れ動いていたということが示された。これによって本論文では、製品アーキテクチャの決定領域が、ハードウェアレベルからファームウェアレベルへと移行しているということに加えて、ファームウェア・アーキテクチャを創出・選択していくプロセスや、ファームウェアの開発において企業が直面している問題とその解決策のひとつが明らかとなったのである。

5.2 製品アーキテクチャの概念の再解釈

既存の製品アーキテクチャに関する議論では、製品アーキテクチャの変化を所与として、その下で企業がどのような行動をとっているのかということや、そのような状況に適した行動とはどのようなものなのかということついて、多くの有益な知見が示されてきた。しかしながら、これらの議論では、従来とは異なる製品アーキテクチャを組織が選択した結果として、製品アーキテクチャの変化が観察されるという観点が十分に考慮されていないため、製

品アーキテクチャがどのように変わってきたのかということについては考察されてきたのだが、そのような変化をどのようにして企業が引き起こしたのかということについては考察されてこなかったのである。

それに対して、本論文では、「組織が主体的に製品アーキテクチャを決めていく」という観点に立って、製品アーキテクチャの変化について観察することによって、当該組織が「どのような意図に基づいてそのような変化を引き起こしたのか」ということと、「そのような変化を起こすのに適した組織のあり方とはどのようなものなのか」という問いかけを行うことの重要性が主張されている。この観点に立って製品アーキテクチャの概念を再解釈すると、図 6 に示されているように、①観察される製品アーキテクチャの背後では、目的とする機能を実現するためにどのような構成要素の組み合わせで対応するのかに関する一連の意思決定が行われていること、②製品アーキテクチャの変化は市場において事後的に観察可能なものに過ぎず、そのような変化の背後には、製品アーキテクチャの選択肢が創出され、それらの中から特定のアーキテクチャが選択されるという一連のプロセスが時間の経過とともに進行していること、③特定のアーキテクチャの選択と製品化の結果を受けて、さらに異なる製品アーキテクチャが創出・選択されているということが考えられる。さらに、これらの一連のプロセスに対しては、企業の戦略や組織のあり方、利用可能な経営資源、製品の捉え方、組織能力が影響を与えていると考えられる。特に、本論文で取り上げた事例では、組織のあり方や製品の捉え方、利用可能な経営資源からの影響について考察されている。

既存の議論のように製品アーキテクチャを外生変数として捉えるのではなくて、本論文のように、製品アーキテクチャの創出・選択プロセスに注目して企業が製品アーキテクチャを変える（内生変数として捉える）という立論をすると、①なぜ変える必要があるのか、②どのようなものに変えるのか、③変えるためにはどのような組織が必要なのかという問いかけが有効になり、各々に答えていくことによって、実際の企業活動についてより有益な示唆を得ることができるようになると考えられる。

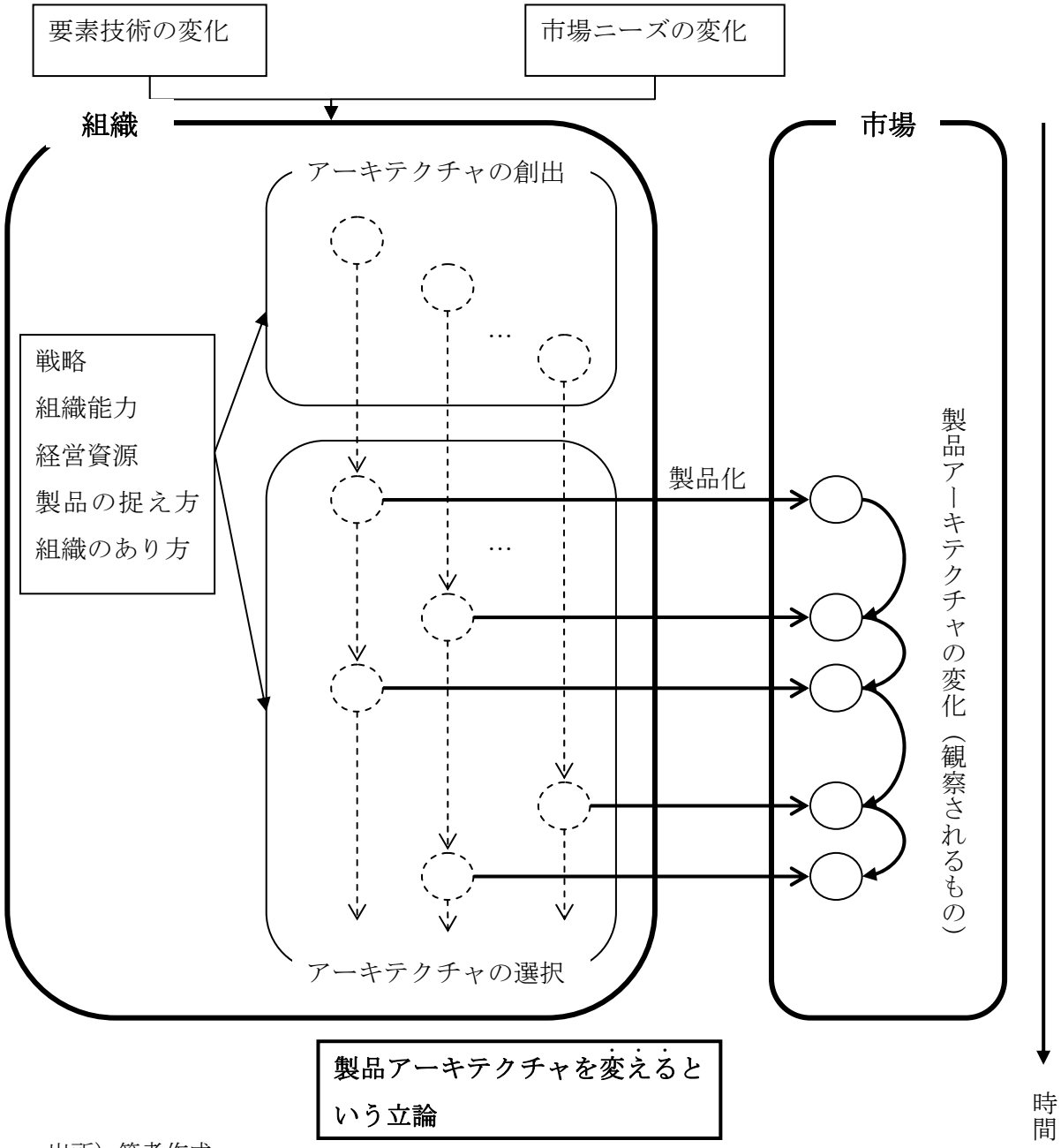
5.3　ファームウェア・アーキテクチャの構築能力に基づく競争へ向けて

本論文で考察してきたように、デジタル化された製品においては、ハードウェア・コンポーネントとファームウェア・コンポーネントの両方が揃って初めて、狙った製品機能を実現できる。そして、ハードウェア・コンポーネントをどのように使って、どのような機能を実現するのかということを決めているのが、ファームウェア・アーキテクチャである。既存の製品アーキテクチャに関する議論では、ファームウェアやファームウェア・アーキテクチャに注目した研究が十分に行われてきておらず、デジタル化された製品分野において競争して

いる企業が直面している問題に対して、効果的な処方箋を提示できていない状態にある。デジタル化が進んだ製品分野においては、各社の競争の焦点はハードウェアレベルでのアーキテクチャの優劣を競う段階から、ファームウェアレベルでのアーキテクチャの優劣を競う段階へと移行してきているのではないかと考えられる。この状況下では、既存のハードウェアを所与とした上で、いかにしてファームウェアを設計していくのかということ、さらには、ファームウェアのアーキテクチャをうまく設計することによって、ハードウェアレベルでのアーキテクチャが変化するということも起きうる。したがって、ファームウェア・アーキテクチャを戦略的に決定していくことが非常に重要になるのである。さらに重要なこととして、ハードウェアとファームウェアの両方について考慮しなければ、よりよい製品アーキテクチャを開発することができないという点で、従来のようにハードウェア・アーキテクチャのみに注目していた製品開発では生じていなかった問題が起きる。すなわち、従来はハードウェアの開発ごとに組織が形成されていて、それぞれにおまけとしてファームウェアの開発が行われていたのだが、ファームウェアの開発を大規模に行おうとした場合には、各部門に散らばっていたノウハウや知識を統合する必要性が生じるのである。つまり、優れたファームウェア・アーキテクチャを創出し選択していくためには、各部門に蓄積されている知識をうまく統合できるかどうかが重要であり、そのための組織能力が必要になるのである。本論文の事例で示されたように、組織の分化に伴って製品に関する知識も分化していくのだが、各部門が保有している知識をうまく統合することができなければ、優れたファームウェア・アーキテクチャを構築することは困難である。このような統合活動を行う範囲が社内で完結しておらず、サプライヤーにまで及んでいる場合には、さらに困難になると考えられる。

このように、ある製品に用いられているファームウェア・コンポーネント全体の相互依存関係を変化させる試みである、「ファームウェア・アーキテクチャの構築」という活動を行うためには、ファームウェア・コンポーネントやそれらの結びつき方に関する深い知識を保有している必要がある。したがって、多機能化に適したファームウェア・アーキテクチャを開発していくためには、既存の企業内部において、それぞれの機能についての知識を保有したエンジニアが集まって部門間の相互調整を緊密に行うことのできるような組織（例えば、リコーにおけるような GW-PT）を設けて、そこで当該アーキテクチャを開発していく必要があると考えられる。さらに、観察可能な製品アーキテクチャの変化にいかに対応していくかということだけではなくて、製品アーキテクチャの創出・選択プロセスにも焦点を当てて分析することによって、製品アーキテクチャを主体的に変えることによって自社にとって有利な競争状況をいかにして作っていくのかという、製品アーキテクチャの構築能力に基づいた競争戦略についてより深い考察を行えるようになると考えられる。

図 6：製品アーキテクチャの創出・選択プロセス



出所）筆者作成

5.4　今後の課題と展望

本論文における議論は、一社の事例研究に基づいたものであり、そこから得られた知見が他の企業や他の産業においても適用可能なのかということについては明らかではない。そのため、今後の研究活動を通じて、同一産業における競合他社についても同様の事例研究を行

い、それらを比較することによって、ファームウェア・アーキテクチャや開発組織、および製品戦略における企業間での差異や、それを生み出す要因について明らかにしていく必要があると考えられる。さらに、自動車など他の製品分野へと分析対象を広げることによって、産業間での比較分析も行っていきたい。また、本論文では、ファームウェア・アーキテクチャと製品開発パフォーマンスとの関係については十分に明らかにされていないが、これについても今後の研究で明らかにしていきたい。

補論１　株式会社リコーにおける複写機事業の略歴[32]

複写機産業全体の歴史の大まかな時代区分を示すと次のようになる。

①ジアゾ式全盛期（1950 年代～60 年代前半）

②複数方式並存期（1960 年代）

a.ゼログラフィー：米国 Xerox「914」(1959 年)

b.ジアゾ式：リコー（リコピー）

c.EF（Electro Fax）方式：リコー（BS1）

③PPC（Plane Paper Copier）競争期（1970 年代～）

a.ゼログラフィーの導入（富士ゼロックス、リコー他）

b.NP 方式の開発（キヤノン）

④デジタル競争開始（1984 年～）

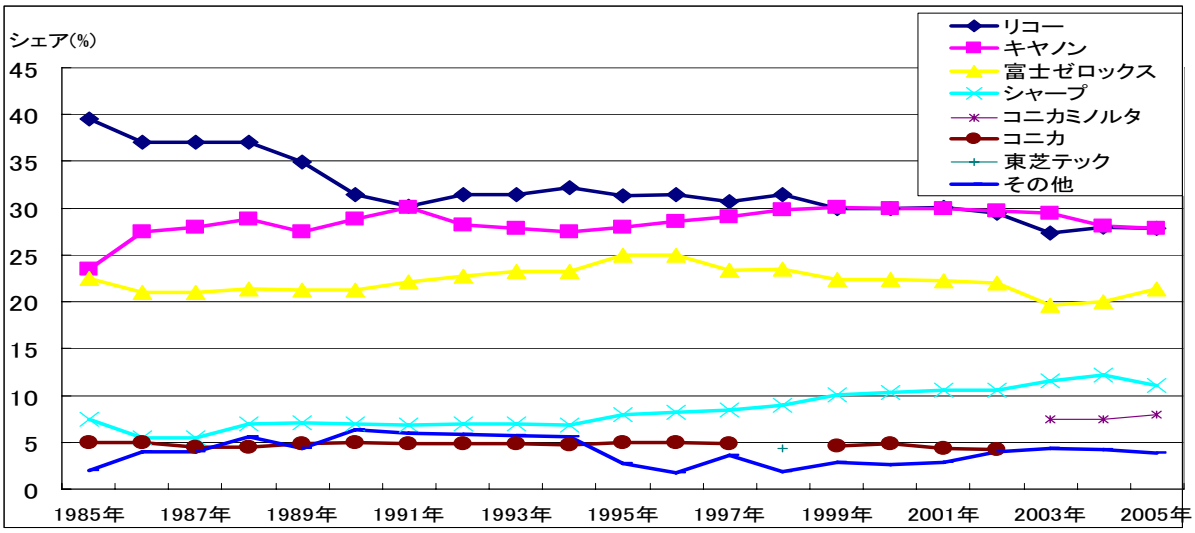
⑤デジタル複合機における競争（1990 年代～）

複写機市場における競争状況としては、表 2 に示されているように、リコーやキヤノン、富士ゼロックスという上位三社によって市場シェアの大半が占められており、これら三社間で熾烈なシェア争いが繰り広げられている[33]。この状況には、アナログ複写機 PPC 全盛の時代からデジタル化の進んだ現在に至るまで大きな変化は見られない。

[32] リコー社史編集委員会編(1996)やキヤノン史編集委員会編（1987）、リコー社内資料、吉原(1992)に基づく。

[33] 表 2 では 1985 年から 2005 年までの複写機の市場シェアの推移について示されている。例えば、2005 年の出荷台数ベースのシェアは、リコー（27.9%）、キヤノン（27.8%）、富士ゼロックス（21.4%）、シャープ（11.1%）、コニカミノルタビジネステクノロジーズ（8.0%）、その他（3.8%）となっている。

表 2：複写機メーカーの市場シェアの推移



出所）日経産業新聞編『市場占有率』各年版より筆者作成

リコーはもともと、1955 年にジアゾ式複写機「リコピー101」を発売して事務機分野へと参入した企業である。この複写機は当時建築分野で多用されることになり、用紙の特徴的な色から「青焼き」と呼ばれていた。リコーは、その後もジアゾ式複写機の分野で国内トップシェアを維持することになった。その後、1965 年には RCA からライセンス供与を受けた EF 方式の複写機である「電子リコピーBS1」を発売して、普通紙複写（PPC）の分野に参入した。1967 年には、「リコピーBS2」を発売し、これが記録的なヒットとなり EF 方式での地位を確立することになった。

1970 年にゼログラフィーの基本特許が切れたことを受けて、キヤノンを除いた複写機メーカーは、ゼログラフィー技術に基づいた PPC の開発・生産・販売に着手し始めた。キヤノンは、1970 年に独自の普通紙複写方式である「NP 方式」に基づいた複写機である「NP-1100」を開発・発売している。リコーは、1972 年に乾式 PPC である「PPC900」を発売し、次いで 75 年に発売した「ニューリコピーDT1200」は、発売後半年で 25,000 台を売上げて PPC 分野でも台数シェア一位を獲得するという大成功を収めた。1976 年には、アナログカラー機である「リコーCR-1000」を発売した。

1979 年には、スキャナやプロセッサ、プリンタの三要素で構成される「デジタルコピア開発プロジェクト」が発足し、1980 年に今で言うところの最初のデジタル複写機である「GT1000」が開発された。1980 年代の複写機ビジネスは、コピーカウンタで複写枚数をカウントして、その枚数（コピーボリューム）によって課金するというものであり、富士ゼロックス、キヤノン、リコーといった上位三社間（いわゆる御三家）での熾烈な競争が繰り広げられていた。当時、既にモノクロ複写機だけでは、他社との差別化を図ることが難しくな

ってきたため、「カラー化」と「ネットワーク化」を行なうことによって付加価値を高めていくという方向へと各社進んでいった。さらにその後、一台で複数の機能を搭載した「デジタル複合機」が登場することになった。

1982 年には、「GT1000」を商品化したモデルである「リコア 3000」が発売された（定価は 990 万円）。この「リコア 3000」での経験が、その後の「IMAGIO」シリーズの企画・開発に生かされることになった。1980 年代初めには、デジタル化やカラー化、ネットワーク化に対応した新製品の開発がリコーにとって最大の課題となっていた。

1987 年には、リコー初の普及型のデジタル複合機である「IMAGIO 320」が発売され、成功を収めた。この製品の特徴は、低価格かつ二機能（コピー+オプション）を実現できることであり、リコーにおけるデジタル複合機の先駆けとなった。1991 年には、「IMAGIO MF530」が発売されたが、本製品は、高性能システムコントローラを搭載して、本格的なマルチファンクション（MF）機を目指したものである。1993 年 6 月には、本格的な普及タイプのデジタル複合機である「IMAGIO MF150」が発売された。1994 年には、「IMAGIO DA250/350」が発売され、アナログ複写機からデジタル複写機への本格的な転換が図られた。2001 年には、「IMAGIO Neo 350/450」が発売され、現在にいたるまでデジタル複合機市場へ新製品が投入され続けている。

リコーは、1991 年に営業赤字に転落したけれども、1992 年以降は大幅に業績を回復し続けており、この業績改善の要因として、それまでの多角化路線から複写機事業への本業回帰が功を奏したということが言われているが、そこでの主力製品として位置づけられていたのがデジタル複写機・複合機であった（藤原、2005; 島田・石崎、2005）。当時、競合他社が既存製品の好調な売り上げによってデジタル分野への参入が遅れたことを利用して、リコーは他社に先駆けてデジタル複写機・複合機事業へと参入し、有利なポジションを確立することに成功したのである（藤原、2005; 島田・石崎、2005）。

補論 2　各ファームウェア・アーキテクチャの特徴[34]

補 2.1「IMAGIO 320/420」のファームウェア・アーキテクチャ

最初に開発されたファームウェア・アーキテクチャは、1987 年に発売されたリコー最初の普及型デジタル複合機である「IMAGIO 320/420」に用いられた。このデジタル複合機を開発するにあたっては、コピー機の開発部隊に対して、FAX やプリンタを開発していた技術

[34]筆者インタビューやリコー社内資料、『Ricoh Technical Report』 (2001)、リコー社史編集委員会編 (1996)に基づく。

者が入ってきた。既に 1984 年に、キヤノンと富士ゼロックスから相次いでデジタル複写機が発売されたが、その価格は 198 万円から 800 万円もするものであり、複写機として普及するには無理があった。これを受けてリコーは、普及型のデジタル複写機の開発を行うことにしたのである。「IMAGIO 320/420」の開発コンセプトは、①スタンドアロンの複写機として今までにはない多彩な画像編集機能で特徴を出すこと、②将来を見据えて他の OA 機器との連動（つまりネットワーク化）を可能にする複合機能を持たせること、③デジタル複写機として本格的に普及させるために 100 万円以下の価格にするというものである。これらのコンセプトを実現するために、FAX 機能やプリンタ機能、電子ファイルへの入出力機能などの複合化を行うためのファームウェア・アーキテクチャが開発された。それとともに、アナログ複写機との部品の共通化を図り、画像処理のための LSI を ASIC にするなどの低コスト化が徹底的に進められた。

このアーキテクチャを開発する際には、コピーや FAX、プリンタ、スキャナを開発する事業部において、ファームウェアをそれぞれ開発する組織があり、別々に開発が進められていた。これらの部門間では、相互の調整もほとんど行われていなかった。また、「IMAGIO 320/420」のファームウェア・アーキテクチャの開発を進めていたのは、複写機部門であり、複写機部門出身のエンジニアがリーダーとなってファームウェア・アーキテクチャが開発されていた。「IMAGIO 320/420」のファームウェア・アーキテクチャは、図 4①に示されるとおりである。

「IMAGIO 320/420」では、コピーをベースとして、それに ANITA インタフェースを介して、FAX ボードかプリンタボードか外部拡張ボードのうちどれかひとつを顧客の要求に合わせて差し込み、一台で二機能が提供されていた。このように、「IMAGIO 320/420」は一台で二つの機能を提供することによって、「多機能化」を実現していくための最初の一歩となったが、まだこの時点では、コピー機に何かもう一つ機能をくっつけて販売するにとどまっており、現在における「デジタル複合機」からはほど遠い製品である。この「IMAGIO 320/420」におけるファームウェア・アーキテクチャでは、一台で三役ないしは四役の多機能を実現できないという限界があった。

補 2.2 「IMAGIO MF530」のファームウェア・アーキテクチャ

前世代機である「IMAGIO 320/420」のファームウェア・アーキテクチャの下では「一台で三役ないし四役」を提供することは不可能であったが、それを可能にするデジタル複合機として開発されたのが、1991 年に発売された「IMAGIO MF530」である[35]。このアーキテク

[35]名称の MF とは Multi‐Function の略である。

チャを開発する際には、コピーや FAX、プリンタ、スキャナを開発する事業部において、ファームウェアをそれぞれ開発する組織があり、別々に開発が進められていた。これらの部門間では、相互の調整もほとんど行われていなかった。また、「IMAGIO MF530」のファームウェア・アーキテクチャの開発を進めていたのは、複写機部門であり、複写機部門出身のエンジニアがリーダーとなってファームウェア・アーキテクチャが開発されていた。

この製品は、コピーをベースとしたファームウェア・アーキテクチャであることは「IMAGIO 320/420」と同じだが、ひとつの OS 上に、FAX アプリケーションやプリンタアプリケーションなどが並列して配置されているという点で異なっている。「IMAGIO MF530」のファームウェア・アーキテクチャは図 4②のとおりである。「IMAGIO MF530」におけるファームウェア・アーキテクチャでは、外部拡張機能とメインボードとの間に ANITA インタフェースを設けて接続している。また、メインボードに対してシステム拡張ユニットが、C4 インタフェースと呼ばれるリコー独自のインタフェースを介して接続されるようになっている。

この「IMAGIO MF530」は、多数のアプリケーション機能を備えたマルチファンクション機であり、当時リコーでは、これを複合機から「融合機」への発展として位置づけていた。この「融合機」では、例えば、フロッピーディスクや光ディスク内の文書を直接 FAX する機能などのように、ハードコピーに出力することなしに他のアプリケーションとデータのやり取りを行えるということを特徴としている。これを可能にするリアルタイム処理とマルチタスク処理をするためのリアルタイム OS として i-TRON 準拠のオリジナル OS が開発された。「IMAGIO MF530」では、一つの CPU 上で三種あるいは四種の機能を処理しようとするため、処理速度に限界があり、さらに、一台のマシンを様々なアプリケーションで使えるというメリットがある反面で、実際のオフィス業務中に、例えば複写出力中に FAX 出力が重なってしまう場合に、複写業務を中断せざるを得ないなどの、使いにくさが表面化するケースが多かった。つまり、アプリケーションからのジョブを終了または中断してからでなければ、異なったアプリケーションへの切り替えができなかったのである。そこで、次世代機には、パフォーマンス（スピード）の向上およびマルチタスク制約の解決が求められた。

補 2.3　「IMAGIO MF150」のファームウェア・アーキテクチャ

前世代機である「IMAGIO MF530」において課題となっていた、パフォーマンス（処理スピード）の向上およびマルチタスク制約の解決をめざして次に開発されたのが、1993 年に発売された「IMAGIO MF150」である。このアーキテクチャを開発する際には、コピーや FAX、プリンタ、スキャナを開発する事業部において、ファームウェアをそれぞれ開発する組織が

あり、別々に開発が進められていた。これらの部門間では、相互の調整もほとんど行われていなかった。また、「IMAGIO MF150」のファームウェア・アーキテクチャの開発を進めていたのは、複写機部門であり、複写機部門出身のエンジニアがリーダーとなってファームウェア・アーキテクチャが開発されていた。

「IMAGIO MF150」は本格普及タイプのデジタル複合機であり、ターゲットとした市場は、一般小規模オフィスであった。このときまでに、FAX 機能を中心とした拡張機能がひとつだけのケース（一複合形態）が圧倒的に多いという販売実績があったので、この一複合形態に焦点を合わせるとともに、マルチファンクション化も可能となるシステム構成とすることにした。そのため、FAX コントローラやプリンタコントローラなどの個別の拡張機能は独立した形態とし、「IMAGIO MF530」のような全拡張機能を統合管理するシステムコントローラは設けられていない。また、メインボードと各コントローラ（FAX やプリンタなど）とのインタフェースには 1 対 N の同時接続が可能となるバス形式（C2 インタフェース）が採用された。バスをどの拡張機能で使用するかということについては、メインボード内にある OS によって使用要求コマンドに応じてコントロールされた。

これらによって、各拡張機能の独立性が高く、相互の干渉や拡張機能の増設による性能低下が生じず、各ユーザーが別々のアプリケーションで一台の機械を占有しているかのように利用できる操作環境が提供されるようになり、マルチファンクション機としての操作性を各段に向上することができた。

本製品におけるファームウェア・アーキテクチャも、コピーをベースとしたものであるが、FAX とプリンタのそれぞれに一つずつ CPU を用いることによって、「IMAGIO MF530」におけるハードウェアリソースの制約を解決して、処理スピードを向上させることに成功した。本製品のファームウェア・アーキテクチャは図 4③のとおりである。このように、「IMAGIO MF150」のファームウェア・アーキテクチャの下では、それぞれの機能を実現するためのハードウェアとファームウェアが自己完結している。実際に販売する際には、コピー機能だけで良いという顧客に対してはそれだけで売っていた。さらに追加機能が欲しいという顧客に対しては、各機能をオプションボードとして買ってもらうようにしていた。

このファームウェア・アーキテクチャによって、確かに製品パフォーマンスの向上が図られたが、機能ごとに一つずつ CPU が使われているため部材コストが高くなるという問題が生じた。さらに、このアーキテクチャの下では、コピーや FAX、プリンタといった各機能を実現するためのファームウェアを別々の組織で開発していたので、各々スクラッチから OS やアプリケーションのプログラムを作成する必要があった。そのため、それぞれのファームウェア間で重複する部分が出るといったムダが生じたり、各ファームウェアの完成するタイ

ミングがずれてしまうという問題が生じた。加えて、デジタル複合機に対して、リコー社内で作っていたレーザープリンタとの共通化への対応も求められることになった。したがって、これら全ての問題を解決できるようなファームウェア・アーキテクチャを構築することが必要となった。

補 2.4　ASAP アーキテクチャ

前世代機である「IMAGIO MF150」において課題となっていた、部材コストの問題の解決やファームウェアの同時開発、リコー製レーザープリンタとの共通化を実現するための新たなファームウェア・アーキテクチャとして開発されたのが「ASAP アーキテクチャ」である。

このアーキテクチャは、1992 年に開発が開始され 1994 年に完成したのだが、最初は「IMAGIO DA250/350（AD2/3）」（以下「DA250/350」と称する）に用いられた。本製品はコピー機能のみを搭載した機種であり、アナログ複写機からデジタル複写機への本格的な転換を図る目的で開発された。その他には、本製品をベースとして、FAX 複合機である「IMAGIO CF」やスキャナ/プリンタ複合機である「IMAGIO MF-P」が開発された。ASAP アーキテクチャは図 4④のとおりである。開発の際の基本コンセプトは、「モジュールアーキテクチャ」（各アプリケーションを独立させる）、「オープンアーキテクチャ」（ASAP インタフェースをサードパーティーにもオープンにする）、「シングルアーキテクチャ」（用いるプラットフォームはひとつ）であった。

ASAP アーキテクチャでは、デジタル複合機をファンクション・コントローラとエンジンに分けて、この間のインタフェース（ASAP インターフェイス）を規定することによって、ファンクション・コントローラとエンジンの同時開発が目指された。また、共通のシステムソフトウェア（OS）上にファンクション・モジュール（ファンクション 1～4）を選択的に積み上げることで異なる製品群を同時開発することを目的としており、その製品群としてコピーのみのベーシック機、CF（Copy-Fax）機、SP（Scanner-Printer）機、NRP（Non- Reprographic ）機が設定された。このように、複数の製品にまたがって、ひとつのファームウェア・アーキテクチャを用いるというように、ファームウェア・アーキテクチャの共通利用という視点が、ASAP アーキテクチャ開発の際から加わったのである。OS としては、リコー独自のリアルタイム OS（MSIS）を開発して、これをベースにマルチファンクション化を図り、ひとつの CPU とひとつの OS 上で全ての機能が動作するというファームウェア・アーキテクチャが志向された。それによって、一つの CPU ですべての機能を処理することができるようになるので、部材コストを抑えることが可能になった。また、コントローラ用 CPU として今後のパフォーマンス要求の増大に備えて基本的には RISC の CPU が採用されが、プリンタで実現

しようとしていたメカのスピードに対応できるだけの性能を持った CPU は当時なくて、依然として CPU の能力が不足していた。

このように、ASAP アーキテクチャでは、①ファンクション・コントローラとエンジンの同時開発と、②四分類の同時開発という、二種類の同時開発が目指されていたが、実際には各機能の同時開発を行うことはできず、コピー機能が先に完成して、他の機能はそれに続いて完成するということが起きた。さらに、同一の機能を実現するためのファームウェアが機種ごとに開発されてしまい、これらの調整を行うだけでも多くの開発工数が必要とされた。

補 2.5　NAD アーキテクチャ

前項で見たような ASAP アーキテクチャにおける失敗を受けて、ASAP アーキテクチャから「IMAGIO MF150」のファームウェア・アーキテクチャへと回帰する形で、コピーベースのファームウェア・アーキテクチャとして「NAD（New AD）アーキテクチャ」が開発された。このアーキテクチャが、どの時期に開発されたのかということと、どの製品に用いられたのかという具体的な情報については明らかにされていない。ただし、ASAP アーキテクチャの開発が完了した 1994 年から、NAD アーキテクチャの次に開発された GW アーキテクチャの開発が開始される 1998 年までの間に、NAD アーキテクチャの開発が行われたのだと推察される。NAD アーキテクチャは図 4⑤のとおりである。

このアーキテクチャの開発を行う際には、コピー部門がリーダーとなって開発が進められた。そこでは、ファームウェア間の共通化を行うよりも、むしろ、既に実際の機種で用いられたことのあるファームウェア・アーキテクチャ（「IMAGIO MF150」）に回帰して、各部門において OS からアプリケーションまで一貫して開発するということが行われた。

NAD アーキテクチャの下では、ベースエンジン側にコピーアプリケーションが位置しており、コピー機能だけならば最小コストで構成できるが、例えば紙幣複写の防止や複写時に地紋を挿入したり、複写と同時に E-Mail を送信するといった機能要求の発展には対応不可能、あるいは大幅なシステム構成の変更が余儀なくされてしまう。したがって、結局のところ、システムが保有する様々なリソースをアプリケーションが自由に活用できる構成にはなっておらず、デジタル複合機のアーキテクチャとしては失敗事例であった。

NAD アーキテクチャの下では、拡張アプリケーション毎にオプションボードが必要になる。また、メモリや HDD、NIC といったハードウェア資源がうまく共有されておらず効率的に利用できないので、ハードウェア・コンポーネントにおけるムダの多いファームウェア・アーキテクチャである。さらに、それぞれのアプリケーション機能ごとに、ひとつの OS とひとつの CPU が用いられており、「IMAGIO MF150」のときと同じように部材コスト

が高くなるという問題が生じた。

以上のように、NAD アーキテクチャでは、①部材コストが高いことや②新機能（ネットワーク化など）の追加に対応できないという問題があった。特に後者の問題は、オフィスなどにおけるネットワークの中で様々な機能を一台で提供可能な「デジタル複合機」を実現する上で大きな制約となっていた。このような NAD アーキテクチャの問題点を解決するべく、次世代のファームウェア・アーキテクチャが開発されることになった。

補 2.6　GW アーキテクチャ

NAD アーキテクチャの限界を克服する形で開発されたファームウェア・アーキテクチャが、2001 年以降に発売されているデジタル複合機に用いられている GW アーキテクチャである。このアーキテクチャは、1998 年に GW-PT（プロジェクトチーム）の発足とともに開発が開始された。GW-PT は、GW アーキテクチャを開発するためのプロジェクトチームである。このような、ファームウェア・アーキテクチャを開発するためのプロジェクトチームが、リコー社内で設置されたのは初めてである。

GW アーキテクチャの開発を行う際には、社内から広くエンジニアを連れてきて、ASAP アーキテクチャの失敗を生かしつつ、以前よりも多くの開発資源が投入された。OS の開発は、10 人以上のグループで行われており、部門間での調整も行われていたので、共通部分の洗い出しをうまく行うことができた。さらに、オープンな UNIX OS に準拠しているので、オープンなデバッグツールを利用することも可能になった。

GW アーキテクチャを開発した狙いとして、①LP および MFP（Multi-Function Printer）における最適プラットフォームの確立、②共通プラットフォーム化による機能向上、コスト低減、開発・評価期間短縮（国内/海外やモノクロ/カラー、LP/MFP を共通化）、③既存ノウハウ（エンジン、メモリ、ネットワークなど）の活用、④生産・サービスの効率化（製造および検査用アプリケーションの設計）、⑤ネットワーク標準によるソリューションの提供（ネットアプリケーションの提供）、⑥オープン性の向上（標準部品の採用、UNIX OS の採用、PCI の採用、リコー標準 I/F の制定と公開）、⑦ユーザーへのアプライアンスの向上（機種間での同一機能、同一操作の提供）が挙げられている。

GW アーキテクチャは、図 4⑥に示されているように、ASAP アーキテクチャに回帰する形で開発されたもので、非コピーベースのアーキテクチャであり、各アプリケーションが並列に配置されており、全てのアプリケーションがひとつの CPU かつひとつの OS 上で動作するようになっている。ASAP アーキテクチャとの相違点は、OS とアプリケーションとの間に、アプリケーションインタフェースが設けられていることと、各アプリケーション間で

共通している部分を共通サービス（Common Services）として切り出していることである。

GW アーキテクチャには、PC に用いられている技術（オープンソース UNIX OS、PCI、HTTP、XML、IEEE1394 インタフェース）が多く取り入れられており、ネットワークへの親和性が高められている。他の機器やネットワークと接続するための仕様は PC と同じように外部に公開されている。また、アプリケーションインタフェースとして社内標準 API（Application Programming Interface）である「GW-API」を開発して、アプリケーションの拡張はソフトウェアおよび最小限のハードウェア（NIC、FAX 通信ボードなど）の追加によって実現できるようになっており、全ての機能を実装したときにコスト最適となるように設計されている。また、オープンソース UNIX OS を採用することによって、高機能デバッガを利用することが可能となった。

GW アーキテクチャでは、エンジンやメモリ、ネットワーク I/F、FAX I/F 等のハードウェア制御を行うためのファームウェアモジュールが単一の CPU 上に実装されている。これによって、NAD アーキテクチャではオプションボードごとに実装されていた制御ソフトウェアモジュールを共通利用することができるようになった。従来機種では、個別オプションボードに実装されていた、プリンタや FAX、スキャナ機能を単一のコントローラ上に移したことによって、機種間での機能・操作性の共通化や、多機種同時開発における大幅な開発期間の短縮が実現した。

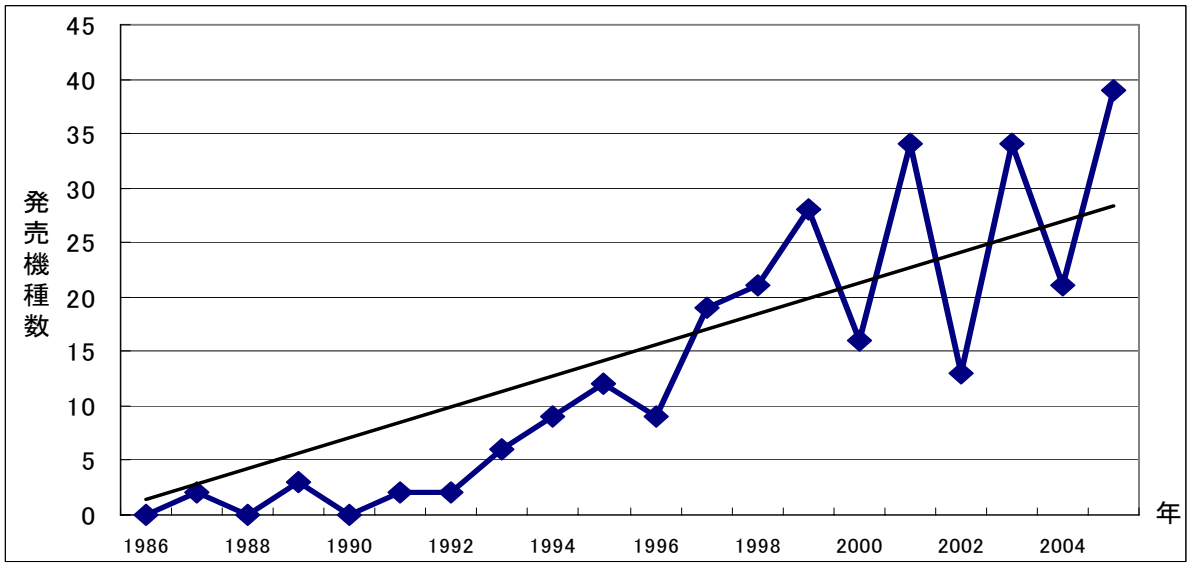
NAD アーキテクチャにおけるアプリケーションプログラム全体のソースコード量は、コピー（150K ステップ）、FAX（150K ステップ）、プリンタ（150K ステップ）を合計して 450K ステップあったが、そのうち共通コントロールサービスに相当する部分が 250K ステップあった。つまり、ファームウェア・コンポーネント間で重複している部分が 250K ステップもあったのである。一方、GW アーキテクチャでは、コピー（50K ステップ）、FAX（50K ステップ）、プリンタ（50K ステップ）、共通コントロールサービス部分（100K ステップ）を合計して 250K ステップまでアプリケーションプログラムのソースコード量を削減することに成功した。それによって、GW アーキテクチャの下では、共通コントロールサービス相当部分のソースコード量を 40%削減（100K/250K）することができた。

補 2.7　各アーキテクチャと新機種投入数との関係

リコーにおいて行われた、ファームウェア・アーキテクチャ開発への取り組みを通じて、開発効率に対してどのような効果が見られるのだろうか。このことについて、表 3 に示されているリコーにおけるデジタル複写機・複合機の年間の新機種投入数と関連付けて考察する（1986 年から 2005 年までを対象）。最初にデジタル複合機が発売された 1987 年から 1995

年にかけて発売機種数は年々増加しているが、1996 年に一度減少し、その後 1999 年まで再び増加しており、2000 年以降は、一年おきに増減を繰り返している。このように年毎の数量の増減はみられるものの、全体的に見て年間新機種投入数は増加傾向にある（1980 年代は最大でも 3 機種、1990 年代前半は十機種前後、1990 年代後半は 12～28 機種、2000 年以降は 34～38 機種）ことから、デジタル複合機市場の拡大に伴う新機種投入数の増大もさることながら、より優れたファームウェア・アーキテクチャを開発することによっても、投入機種数が増大したのではないかと推測される。このようなファームウェア・アーキテクチャと製品開発の効率との関係性に関する研究は、別稿にて詳細に行うことにする。

表 3：リコーにおけるデジタル複写機・複合機の新機種発売数量推移



出所）2000 年までは OA ライフ(1996、1999、2001)、2001 年以降はリコー　ニュースリリースより筆者作成

謝辞

本論文の作成にあたっては、多くの方々のご支援をいただいた。なかでも、本論文の事例研究で分析対象とさせていただいた、株式会社リコーの技術者の方々には、日々の業務でご多忙のところ、貴重なお時間を割いて私のインタビュー調査にご協力いただき、加えて貴重な資料も提供していただいた。ここに記して感謝の意を表したい。また、立本博文氏（東京大学ものづくり経営研究センター）には、インタビュー調査にご同行いただくなど大変お世話になり、有益な助言をいただいた。さらに、新宅純二郎先生（東京大学大学院経済学研究科助教授）ならびに、東京大学ものづくり経営研究センターの小川紘一氏、善本哲夫氏から

は有益な助言をいただいた。心から感謝申し上げます。

参考文献

Abernathy, W. J. (1978). *The Productivity Dilemma*. Baltimore: The Johns Hopkins University Press

青木昌彦 (2002) 「産業アーキテクチャのモジュール化―理論的イントロダクション」 青木昌彦,安藤晴彦編著 『モジュール化』1 章, 東洋経済新報社

青島矢一, 武石彰(2001)「アーキテクチャという考え方」藤本隆宏, 武石彰, 青島矢一編著 『ビジネス・アーキテクチャ』2 章, 有斐閣

Baldwin, C.Y. & Clark, K.B. (2000). *Design Rules, vol.1: The Power of Modularity*, Cambridge, MA: MIT Press. 邦訳, C・ボールドウィン, K・クラーク (2004)『デザイン・ルール―モジュール化パワー』 安藤晴彦訳 東洋経済新報社

Clark, K.B. & Fujimoto,T. (1991).*Product development performance*. Boston: Harvard Business School Press. 邦訳, 藤本隆宏, キム・B・クラーク (1993) 『製品開発力』 田村明比古訳 ダイヤモンド社

Fine, C.H. (1998). *Clock speed*. Cambridge, MA: Basic Books

藤本隆宏 (2001)「アーキテクチャの産業論」 藤本隆宏, 武石彰, 青島矢一編著 『ビジネス・アーキテクチャ』1 章, 有斐閣

藤田雅俊 (2005)「リコー―デジタル化時代を先取りした浜田広―」 三品和広編著 『経営は十年にして成らず』1 章, 東洋経済新報社

福澤光啓, 立本博文, 新宅純二郎 (2006)「ファームウェア・アーキテクチャの揺れ動きとその要因―デジタル複合機の事例―」MMRC Discussion Paper Series MMRC-J-89 東京大学ものづくり経営研究センター

Galbraith, J.R. (1973). *Designing complex organizations*. Reading, MA: Addison-Wesley. 邦訳, J・ガルブレイス(1980) 『横断組織の設計―マトリックス組織の調整機能と効果的運用―』 梅津祐良訳 ダイヤモンド社

Hannan, M.T. & Freeman, J. (1977). The population ecology of organizations. *American Journal of Sociology, 82* (5), 929-964

Henderson, R.M. and Clark, K.B.(1990).Architectural innovation: the reconfiguration of existing product technologies and the failure of established firms. *Administrative Science Quarterly, 35*, 9-30

加藤寛之(2002)「モジュラリティ・ドライバ―モジュラー化と逆流防止弁―」『赤門マネジメント・レビュー』*1*(8), 633-641

楠木建, ヘンリーW.チェスブロウ (2001)「製品アーキテクチャのダイナッミック・シフト」藤本隆宏, 武石彰, 青島矢一編著 『ビジネス・アーキテクチャ』13 章, 有斐閣

Lawrence, P.R. & Lorsch, J.W. (1967) *Organization and environment: managing differentiation and integration.* Boston, MA: Harvard University Press. 邦訳,P・R・ローレンス, J・W・ローシュ (1977)『組織の条件適応理論』 吉田博訳 産業能率短期大学出版部

柴田友厚, 玄場公規, 児玉文雄 (2002) 『製品アーキテクチャの進化論』 白桃書房

島田昌和, 石崎琢也(2005)「リコーの複写機事業の失敗と再生―プロダクト・ミックスの視点から」宇田川勝, 佐々木聡, 四宮正親編著 『失敗と再生の経営史』11 章, 有斐閣

新宅純二郎 (1994) 『日本企業の競争戦略』 有斐閣

新宅純二郎, 小川紘一, 善本哲夫 (2006) 「光ディスク産業の競争と国際的協業モデル―擦り合わせ要素のカプセル化によるモジュラー化の進展―」榊原清則, 香山晋編著『イノベーションと競争優位』4 章, NTT 出版

Sanchez, R. & Mahoney, J.T. (1996). Modularity, flexibility, and knowledge management in product and organizatuion design. *Strategic Management Journal*, *17*, 63-76

Simon (1996). *The science of the artificial* (3rd ed.). Cambridge, MA: MIT Press. 邦訳,H・A・サイモン(1999)『システムの科学』 (第三版) 稲葉元吉, 吉原英樹訳 パーソナルメディア

立本博文(2002)「ソフト開発プロセスモデルと製品属性」『赤門マネジメント・レビュー』*1*(4), 309-336

Thompson, J.D. (1967). *Organization in Action*. New York: McGraw-Hill. 邦訳, J・D・トンプソン (1987)『オーガニゼーション・イン・アクション』 高宮晋監訳 同文館

Ulrich, K. (1995). The role of product architecture in the manufacturing firm. *Research Policy*, *24*, 419-440

von Hippel, E. (1990). Task partitioning: an innovation process variable. *Research Policy*, *19*, 407-418

吉原英樹 (1992) 『富士ゼロックスの奇跡』 東洋経済新報社

参考資料

キヤノン史編集委員会編 (1987) 『キヤノン史―技術と製品の 50 年』キヤノン株式会社

経済産業省『機械統計年報』各年版

日経産業新聞編『市場占有率』各年版

OA ライフ(1995)『デジタル複写機のすべて 1996 年版』

OA ライフ(1999)『デジタル複写機のすべて No.2 (1999 年度版)』

OA ライフ(2001)『デジタル複写機のすべて No.3 (2001 年度版)』

リコー社史編集委員会編 (1996) 『IPS への道―リコー60 年技術史』株式会社リコー

『Ricoh Technical Report』 各年版 株式会社リコー

株式会社リコー ニュースリリース http://www.ricoh.co.jp/release/

株式会社リコー『有価証券報告書』1986 年度版～2004 年度版

株式会社リコー『リコー・ファクトブック』2003 年版～2006 年版