製品開発における相互依存関係の処理のための方法論のパターン

－製品アーキテクチャと組織設計の関係から－

Patterns of method for processing interdependency of New Product Development processes －a view from relationship between product architecture and organizational design－

東　　秀忠
HIGASHI, Hidetada

【概要】

本研究ノートでは、製品開発組織設計に関する諸問題を、既存研究のレビューから提起する。特に、製品アーキテクチャ論における「モジュラー化」のアプローチとその問題点についての検討から、製品開発における効果的な相互依存解決の方法論のパターン化を行う。複雑な相互依存関係の処理を実施することが求められる人工物の設計プロセスにおいて、その方法論を情報処理の必要性を節約するか、情報処理能力を向上させるかというアプローチに加え、着眼点として設計主体である組織に注目するか、設計対象である人工物そのものに注目するか、という軸を加えることでパターン化できると考えられる。

【キーワード】

製品開発・組織設計・製品アーキテクチャ・モジュラー化・技術経営

1. はじめに

本稿は、製品開発組織に関する既存研究のレビューと、製品アーキテクチャ論における「モジュラー化」の概念の再検討を通じて、製品開発組織の設計に関わる諸問題を提起することを狙いとする。具体的には、製品アーキテクチャ論が立脚する、「デザイン構造」と「タスク構造」の同形性についての再検討を行うことを通じて、製品開発における複雑性への対処のあり方を議論する。

本稿ではまず、組織設計の根底に存在する分業と協業について検討する。その上で、製品開発を複雑に相互依存した情報の処理であると定義し、そこに存在する問題解決のあり方についてのレビューを行う。その後、製品開発組織の設計に関する諸研究を概観する。そして製品アーキテクチャ論の中でも特に「モジュラー化」に関連した議論を検討する。これらの議論を通じて、製品開発における複雑性への対処のあり方のパターンを提起することを目指す。

2. 情報処理・問題解決と分業・統合に関する既存研究

2. 1「分業論」の起源

歴史上初めて「分業」について経済学の観点

から議論したのはSmith(1776)である。Smithは「国富論」の第1章から第3章において分業の効果、分業の発生原理、分業と市場の関係性について論じている。

Smith(1776)の第1章は分業の効果を大きく三つに分けて論じている。第一は技能の習熟、第二は段取り替えロスの減少、第三は専用生産設備の開発である。これらの三つの効果はともに生産性向上に大きく寄与する。これにより分業が生産性を飛躍的に向上させることを、ピン作りや釘作りのケースを紹介することで示している。

同時に、Smith(1776)は「人間の利己心に基づく交換性向」が分業を生み出すと主張している。すなわち、社会に属する各人が専門的に何かを生産し、それを自分が必要とするものと交換するほうがより多くのものを得ることができることを知っているということである。しかし分業は市場の規模と範囲によって阻害される。あまりに市場が小さいと、社会的分業が成立し得ないのである。そしてSmith(1776)の議論は交換を可能とする範囲を拡大し、社会的分業を促進するために貨幣が発生する、という方向に進展していくのである。

2.2 企業内での専門化と垂直統合

Penrose(1955)は、企業内で専門化が進む要因は、それが正当化されるほど十分な生産量の拡大であると指摘している。逆に言えば、専門化がなされるほど十分な生産量が確保されていない間は、潜在的に専門家たり得る人員も汎用的に利用されるのである。逆に専用生産設備であればそれは離散的にしか追加投入できない上、その組み合わせ上の問題から遊休が発生する可能性が非常に高い。故にこの遊休を使い果たすべく企業は成長しようとするのである。これを「倍数の原理」と呼ぶ。

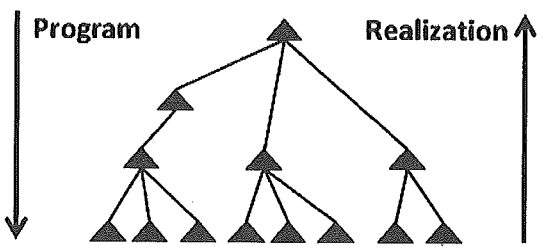
一方Chandler&Daem(1977)は、地域的な単一職能企業がいかにして垂直統合的大企業になっていったかを描写している。生産や輸送、通信といった分野での技術革新や近代的原価計算手法が「速度の経済」の達成を可能にした。速度の経済を達成しようとすると職能間の相互依存性を高める必要があり、単一企業内での統合が推し進められたのである。

2.3 情報処理のためのアルゴリズムとしての階層性

Alexander(1964)やSimon(1969)は、複雑性に対処するための方策として階層構造を提案している。階層構造が相互依存関係を分断することを利用しているのである。

Alexander(1964)はサブシステムを独立的に操作できるように設計するための方策として、ProgramとRealizationという方法論を提唱している。Programとは、設計すべきものを抽象的サブシステムへと階層的に分解していくことで独立的に操作できるようにすることである。一方、Realizationとは、分解されたサブシステム単位から実際のものを作っていき、それを合体させることで最終的に設計すべきものを完成させる作業をいう。

図1：Alexanderによる"Program"と"Realization"[i]



一方、Simon(1969)は、「準分解可能システム」という概念を提唱している。これは、階層的システムが、下位システムの内部に存在する相互作用と、下位システムのシステム間に存在

される相互作用が弁別されることによって下位システムが独立的に活動できることを指して表現したものである。そしてこのようなシステムは企業などの公的組織によく見られるものであると指摘している。

2.4 問題解決サイクルの発見

Simon(1969)は、「ジェネレータ・テスト」モデルという問題解決サイクルを提案している。これはもっとも基本的な問題解決サイクルで、代替案を生成し、それをテストするという流れを示したものである。その後サイモン(1987)で定式化された問題解決サイクルは、「問題認識→代替案のサーチ→モデル制作→シミュレーション実行→評価→選択」というルートを取る。このような、不完全な因果知識の下で探索的に問題を解くプロセスは製品開発と相性が良い。

2.5「課業分割」と「情報粘着性」

von Hippel(1990)は飛行機の設計や部屋のインテリア設計を例に取り、イノベーションに際して発生するタスク同士の相互依存関係を適切にグループ化して相互調整を減少させる方策を「課業分割」[ii]と定義した。「課業分割」を行うためにはタスク間の相互依存関係を予測する必要があるが、そのためにはタスクの定義を調整したり、タスク間で行われる問題解決の障壁を無くしたりしておくことが役に立つと主張している。

von Hippel(1994)は、誰が、どこでイノベーションを行うか？という問題に対する解として、「情報粘着性」[iii]という概念を提示している。これは「問題解決活動における情報移転の難しさ」を指すものである。情報を移転する際には(1)：情報を受け手が利用可能な状態にすることと、(2)：受け手に対して情報を移転することの二つの過程を経る必要があるが、この両過程にかかる費用の大きさが情報粘着性として定義されている。

そして、von Hippel(1994)は情報の粘着している場所と問題解決が行われる場所を関係づけた。問題解決のための情報が一カ所に粘着しているならば問題解決はその場で行われる。複数箇所に粘着しているならば、問題解決が進むにつれてそれが行われる場所が移っていく。そして、情報が複数箇所に粘着していてしかも、情報をやりとりするためのコストがあまりに高くつく場合にはその問題解決は「課業分割」されることになるのである。

表1：情報の特性と情報粘着性の関係[iv]

	コスト＝情報粘着性		
	低い	← →	高い
1：情報を受け手が利用可能な状態にする費用	明確		暗黙的
	単純		複雑
	独立的		システムに依存
	受け手の吸収能力が高い		受け手の吸収能力が低い
	送り手の信頼が高い		送り手の信頼が低い
	頻繁なコミュニケーション		限定的なコミュニケーション
2：受け手に情報を移転する費用	情報量少ない		情報量多い
	媒体が移動可能		媒体が移動不可能
	観察可能		観察不可能

2.6 部門間の「分化」と「統合」の必要性

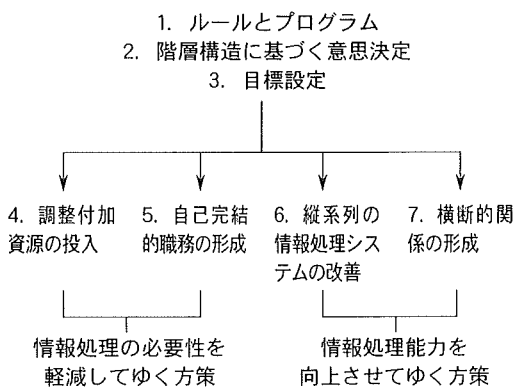
Lawrence&Lorsch(1967)は、組織内の各部門が、それぞれがさらされている環境の不確実性に応じた組織の構造化の程度や時間、目標等への指向性を取るという、「分化」の概念を提唱した。プラスチック製造企業や容器産業のケースを取って技術的不確実性や顧客からの要求の多様性といった環境要件が、研究開発部門、製造部門、営業部門など組織内の各部門の特性を「分化」させていることを示した。

そして、この「分化」された部門間を、相互調整などでもって強力に「統合」していくことが好業績のためには必要であると主張している。「分化」が不十分でも、「統合」が不十分でも環境に対する適切な適応はできないのである。

2.7「統合」のための連携調整と情報処理

Galbraith(1973)は、組織間の相互調整に伴う情報処理量が増大すると単純な機能別の階層組織では処理が追いつかなくなってしまうと主張している。そして、その解決策として４つのオプションを提示した。これは大きく分けると、組織間の調整負荷を軽減させる方法と、組織間の調整能力を向上させる方法に分類できる。

図２：情報のオーバーフローに対処するための組織設計[v]



情報処理量そのものを減少させることを通して調整負荷を軽減させる方法としては、「調整負荷資源[vi]」の投入と「自己完結的職務」の形成が論じられている。

既存の経営資源だけでは情報処理がオーバーフローしてしまう場合に、在庫拡大や人員投入、リードタイムの延長など、経営資源を追加投入したり期待業績水準を下方修正したりすることで対処することが、「調整負荷資源」の投入と定義されている[vii]。

一方、アウトプットの多様性を減少させる方法が「自己完結的職務」の形成である。機能別組織に対してたとえば事業部制の採用に見られるように、必要な資源を単位別にまとめることで部門間のアウトプット同士が相互依存関係を持たないようにする方策である。

組織間の情報処理能力を向上させる方法としては、縦系列の情報処理システムの改善によって直接的に情報処理能力を向上させる方法と、直接接触やリエゾンの導入、マトリックス組織に代表されるような横断的関係の形成が挙げられている。

前者は生産計画などを頻繁にアップデートできるようにコンピュータの処理能力を増強していくことを指している。これにより、例外事項の発生を防ぐことで調整による情報処理能力のオーバーフローを回避するのである。一方後者は、部門間の相互依存関係によって発生する問題を、階層構造の上部に持ち込まないための手法である。階層構造の上部に持ち込んでしまうと解決までに時間やコストが大きくかかってしまう。これを避けるためにインフォーマルな接触関係を構築したり、複雑な相互依存関係を統合的に処理する統合者・統合管理者を置いたり、という方法がとられるのである。

2.8 過剰分業と調整ロス

Brooks(1995)[viii]は、コンピュータのソフト

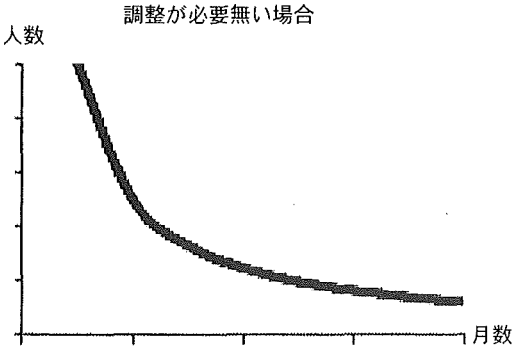
ウェア開発プロジェクトを例に取り、「Brooksの法則」を提唱した。これは「スケジュールに遅れが発生しているプロジェクトに、人員を追加投入してもリードタイムは伸びるだけである」というものである。

この法則はBrooksは自身が携わったIBMシステム360開発プロジェクトから得た経験則であるが、調整ロスに関する問題を端的に表現していると言える。プロジェクトチームの人員が増えれば増えるほど、コミュニケーションや教育、調整に食われる工数が増えるために分業による効果が相殺されてしまうのである。下に4つのグラフを提示する。これらはタスクの分割とそれに伴う調整ロスが開発リードタイムに与える影響を提示している。

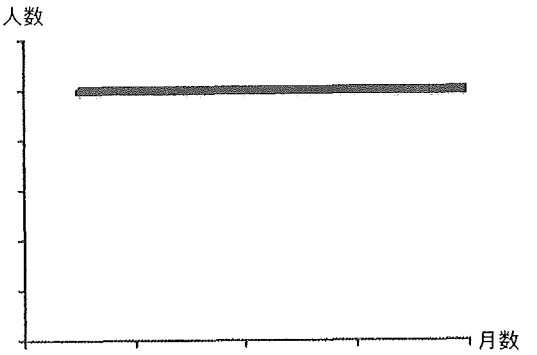
1番は調整ロスが発生しないと仮定した場合のプロジェクト人員と開発リードタイムの関係、2番はタスク分割が不可能であるためにプロジェクト人員と開発リードタイムが関係しないというケース、3番はタスクが分割可能だが、調整によるロスが開発リードタイムの短縮を妨げるケース、4番は相互依存関係が複雑で、分割されたタスク同士の調整ロスが開発リードタイムをむしろ引き延ばしてしまう、というケースである。

図3：「人月の神話」…開発リードタイムと投入人員の関係性[ix]

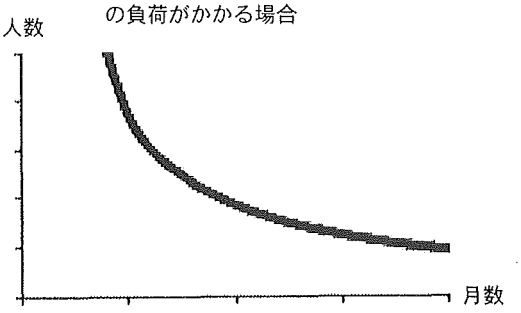
1. 開発作業細分化が可能かつ、作業間の調整が必要無い場合



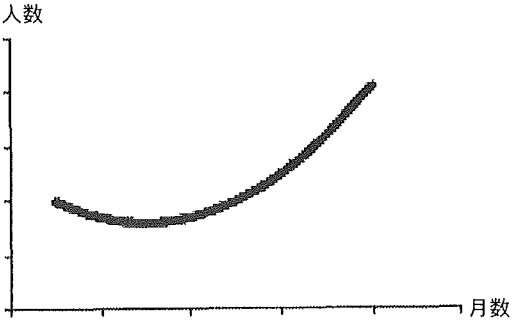
2. 開発作業を細分化できない場合



3. 作業間の調整・コミュニケーションの負荷がかかる場合



4. 作業間の調整・コミュニケーションによって負荷が増大してしまう場合



2.9 マトリックス組織としての自動車企業

Clark&Fujimoto(1991)は自動車の製品開発組織の国際的調査を行い、機能別組織が頻繁なコミュニケーションを基本にした相互横断調整を行うことで製品開発のパフォーマンスが向上することを示した。ここでは「重量級プロダクトマネジャー」という概念が提示されている。

Clark&Fujimoto(1991)によって確認された

自動車メーカーにおける製品開発組織の４類型を以下に示す。

第一のパターンはいわゆる従来型の機能別組織である。この組織形態においては、製品全体について全般的責任を負うマネジャーは存在せず、部門間の調整は規則や手続き、詳細な仕様の作成、伝統や世界観の共有、直接接触やミーティングによって担保される。

第二パターンは軽量級プロダクトマネジャー型組織である。基本的な組織構造は第一パターンと相違ないが、製品開発活動の調整者としてプロダクトマネジャーが存在している。このパターンにおけるプロダクトマネジャーはしかし製品開発における実務レベルの技術者とのパイプをもたず、調整は各機能別部門のリエゾンを通して行われる。また、このプロダクトマネジャーのタスクは主として調整であり、コンセプトや販売・マーケティングに対する責任は負っていない。これが重量級プロダクトマネジャーに対して軽量級、と呼称される要因である。

第三のパターンが「重量級プロダクトマネジャー型組織」である。組織自体は依然として機能別組織的であるが、プロダクトマネジャーの責任と影響力が非常に大きいのである。ここにおいて重量級プロダクトマネジャーは市場とも独自の接点を持った上でコンセプト創造から販売にまで責任を負った上でプロジェクトに関係する全ての部門に対して直接・間接に強い影響力を行使するのである。実質的にはこの重量級プロダクトマネジャーが製品開発のゼネラル・マネジャーとして君臨することでマトリックス組織を形成しているのである。

第４のパターンはプロジェクト実行チーム型組織である。重量級プロダクトマネジャー型よりもさらに製品主体の考え方が強い。ここでは重量級プロダクトマネジャーがプロジェクトチームを作って専属のエンジニアを配置する。チーム所属となったエンジニアは機能別部門を離脱して重量級プロダクトマネジャーに直接報告を行う。Galbraith(1973)に従えば、より「自己完結的職務の形成」に近い形である。

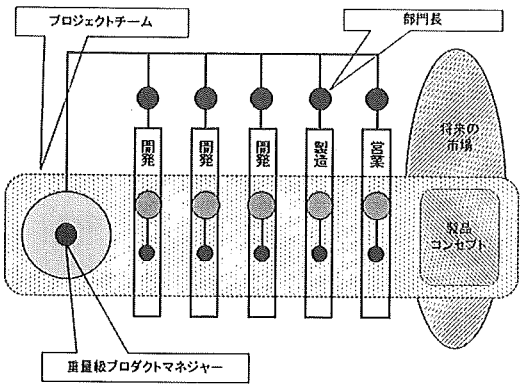
この４パターンの組織と調整の形態の中でも、高パフォーマンスなパターンは１と3,4に集中して現れていた。１は高級車専門メーカーのうちでも好業績な企業が採用し、3,4は好業績な日本の量産車メーカーに際だって現れていた。この違いは顧客の嗜好の変化の早さや違い、そしてロイヤリティの高さ、コストと性能のバランス点といった要件によって説明できる。

高級車専門メーカーにおいてはモデル間、そして時間を超えて製品のコンセプトが首尾一貫していることが高く評価される。また、圧倒的に高い製品機能を体現していることが求められる。これらを担保しているのがエンジニアリング組織の伝統や各部門の完全主義的指向である。このような条件下では各機能間での連携調整の必要性が低いのみならず、むやみな調整が妥協を呼ぶことで製品機能を落とす可能性が想定されるために、各部門の機能分化を重視するのである[x]。

一方、量産車メーカーにおいては重量級プロダクトマネジャーの存在が好業績のために重要な役割を演じている。これは多様かつ移ろいやすい顧客の嗜好・期待を先取りして把握したうえで迅速に多様な製品を市場に投入することや、性能とコストをうまくバランスさせることが重要であるためである。このためには製品開発における問題解決サイクルを、統合された形で迅速に進めることが重要になる。ここにおいて、コンセプトと製品開発組織、生産部門、営業と言った各部門とを強力に統合するための主体としての重量級プロダクトマネジャーの存在が必要になるのである。成功する重量級プロダクトマネジャーに求められる要件としては、独自に、市場と直接接触するチャネルを持つこと

や、コンセプトの技術翻訳能力、エンジニアと直接接触し、問題解決のために積極的に行動する行動力、そしてコンセプトを守護し、唱道するプロモーターとしての役割、部門間のコンフリクトをコンセプトに従って裁定する責任などが挙げられる。

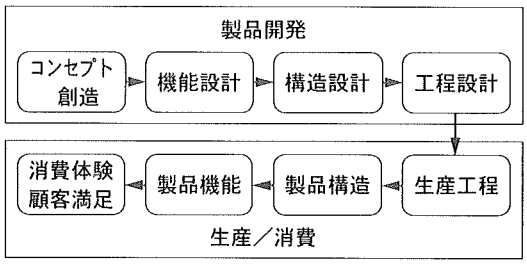
図4：重量級プロダクトマネジャー型組織[xi]



2. 10「情報創造・転写システムとしての製品開発」に存在する対称性

Clark&Fujimoto(1991)の学術的貢献は重量級プロダクトマネジャーの発見にとどまらない。製品開発・生産を「情報創造・転写システム」というパースペクティブの提供もまた重要である。この視座に立てば、製品開発とは図4で示すように「生産・消費のシミュレーション」であると考えられるのである。製品コンセプトを創造することはユーザーによる製品の消費とそれに伴う満足を定義することであり、製品計画は製品が内包している機能を定義していくことである。製品設計とは生産される製品の構造を決定することであり、工程設計とは製品を生産するための工程を設計することである。このように、製品開発プロセスと生産・消費プロセスとの間には対称性が存在しているのである[xii]。

図5：生産・消費のシミュレーションとしての製品開発[xiii]



2. 11 問題解決・分業という観点から見た「モジュラー化」[xiv]

Baldwin&Clark(2000)は、製品の機能・構造関係で規定される「製品アーキテクチャ」の「モジュラー化」について議論している。

相互依存関係が複雑に絡み合った製品において、機能間の相互依存関係をあらかじめデザイン・ルールとして解決しておくことで、機能の集合体としてのモジュールを開発する際に、他者からの干渉を排除することが可能になる。これが製品アーキテクチャのモジュラー化である。製品のモジュラー化が達成されるとそれぞれの独立したモジュールはデザイン・ルールを守っている限り自己完結的に進化を進めることが可能になるのである。

この考え方そのものは von Hippel(1990)に見られる課業分割とそう大きな違いはない。つまりは「来るべきモジュール」同士の相互依存関係をあらかじめ解決し、そのためのルールを制約条件として設定することで相互依存関係の小さい、自己完結的な組織が並行して製品開発プロセスを進めることを可能にするという手法である。

だが、Baldwin&Clark(2000)の最大の学術的貢献は、モジュラー化に対して Value という概念を織り込んだことである。すなわち、モジュラー化を行うことで統合・テストのコストが大幅に減少してシステムの進化が促進され

る。これがシステム全体のValueを飛躍的に増大させるのである。

システム全体のValueが増大すると同時に、各モジュールのValueも増大する。これが企業に対して垂直統合の必要性を削減し、水平分業を可能にするのである[xv]。そしてそれが産業のダイナミクスとしてモジュールの開発を行う独立小企業群としての「モジュラー・クラスター」が生成された点を描写しているのである。

つまり、Baldwin&Clark(2000)の議論は事前の問題解決によって自己完結的組織の連合体が製品としての統合度を担保できる、ということを主張していると読み解くことができよう。

2. 12「モジュラー化」の議論に存在する問題点

一方で、Baldwin&Clark(2000)の議論にはいくつかの点で、不明瞭な部分が残る。

まずは、「設計パラメータ」の定義が広いことが挙げられる。彼らは「設計パラメータ」を、人工物を記述するための単位であると定義し、各種の設計パラメータが適切に定義されることで人工物の設計が完了するとしている。しかしこの「記述する単位」というコンセプトには不明瞭さが残る。たとえば同書の表2－1によると、「設計パラメータ」の中に「製造プロセス」が含まれている。人工物を設計する際に、製造プロセスを決定することは必要不可欠であり、その意味で彼らの「設計パラメータ」に含まれることは間違いないところであるが、設計の対象となる人工物そのものを定義するパラメータではないという点、そして「製造プロセス」そのものもまた人工物であり、その設計を行わねばならないと言う点で問題がある。

また、設計パラメータを決定することを「設計タスク」と定義することそのものにも問題がある。というのも、設計パラメータを決定するために必要となる行為は単一ではなく、複数の活動からなっているためだ。先述の通り、製品開発における問題解決活動は「ジェネレータ・テスト」モデルで説明できるが、これに含まれる「設計・試作・実験・評価」というプロセスはそれぞれが独立した活動として捉えられていることが一般的なのである。特に、構成要素間の相互依存性が複雑な人工物においてはこれらの活動が専門化により分業されることが一般的である。

つまり、単に設計パラメータに対して特権的なデザイン・ルールを設定することだけでは詳細なタスクに踏み行った部分での相互依存関係の処理が困難な可能性があるのである。ここに至り、設計構造とタスク構造の同形性にも疑義が生じるのである。タスクから考えた場合と人工物から考えた場合で、その相互依存関係の構造に違いが生じている可能性は大いにありうるのだ。

このことは、モジュラーアーキテクチャを実現した製品といえどもタスク構造の検討に基づく組織設計を通じてその生産性を向上させることが可能だと示唆している。つまり、相互依存解決のための方法論には、「人工物ベース」で考えるか、「設計主体＝組織」ベースで考えるかというパターンが存在しうるのである。

3. まとめ：相互依存解決のための方法論のパターン化

以上をまとめると、複雑な人工物の設計に際しては、その複雑な相互依存を解決するためのプロセスが必要であることがわかる。相互依存関係を何らかの形で処理しなければ、Brooks(1975)が指摘するように、複雑性と調整負荷が爆発的に増大してしまうのである。そこで、Galbraith(1973)の議論に基づきその方法論を整理すると、「情報処理能力の向上」アプローチと、「情報処理の必要性の節約」アプローチに大別できる。

さらにもう一つ、「設計主体としての組織」

に着目するか「設計対象としての人工物」に着目するか、という分類軸を設定すると、以下のような分類が可能であろう。

表2：相互依存解決のための方法論のパターン

		着眼点：組織 or 人工物	
		組　織	人工物
方法論：情報処理節約 or 情報処理能力向上	情報処理節約	自己完結的職務	課業分割モジュラー化
	情報処理能力向上	重直統合？重量級プロダクトマネジャー	調整付加資源？

まず、左上のセルについて検討してみよう。これは、設計主体である組織が情報処理を節約しようとする行為を指す。これに当てはまるものとしてはGalbraith(1973)で指摘されている「自己完結的職務の設定」がある。右上のセルについては、設計対象である人工物にデザイン・ルールを設定することを通じて情報処理の必要性を軽減するモジュラー化や、von Hippel (1990)の課業分割が当てはまるだろう。

次に、左下のセルについては、設計主体である組織の情報処理能力を向上させる方策が当てはまる。典型例としてはClark&Fujimoto (1991)が指摘した重量級プロダクトマネジャー型組織が挙げられよう。また、Chandler &Daem(1977)が指摘した垂直統合も、このセルに当てはまるかもしれない。一方、右下のセルについては定義が難しい。ここではGalbraith(1973)の調整付加資源を挙げたが、情報処理能力が向上するような人工物の設計をどう定義するかは今後の検討課題としたい。

ここで論じている製品開発における相互依存解決のパターン化は未だその概念設計の中途にあり、説得力が不足している。今後このフレームワークをさらに堅固にすることを通じて、製品開発組織ならびにそのプロセスをよりよく描写することを目指していきたい。

参考文献

Alexander, C, (1964) *Notes on the Synthesis of Form.* Harvard University Press

Brooks, Fredelick P. Jr. (1975) *The Mythical Man－Month.* Addison－Wesley

Brooks, Fredelick P. Jr. (1995) *The Mythical Man－Month.* (1995 ed.) Addison－Wesley

Chandler, A. D. Jr. & Daems, H. (1977) Administrative Coordination, Allocation and Monitoring: Concepts and Comparisons. *Harvard University, Working Paper* NO.77－21

Clark, K. B. & T. Fujimoto, (1991) *Product Development Performance: Strategy, Organization, and Management in the World Auto Industry.Harbard 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

Galbraith, Jay R. (1973) *Designing Complex Organizations.* Addison－Wesley, 邦訳, ジェイ・R・ガルブレイス(1977)『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

Penrose, E. T. (1959) *The theory of the growth of the firm.* Oxford,

Simon, H. A. (1969) *The Art of Artificial.MIT* Press, 邦訳, ハーバート・A・サイモン(2001)『システムの科学　第3版』稲葉元吉・吉原英樹訳,　パーソナルメディア

高橋伸夫(2002)「ペンローズ『会社成長の理論』を読む」赤門マネジメントレビュー, 1,105－124

高橋伸夫（編）(2000)「超企業・組織論」有斐閣

von Hippel, E. (1990) Task Partitioning: An Innovation Process Variable. *Research Policy,* 19, 407－418

von Hippel, E. (1994) Sticky information and the locus of problem solving. *Management Science,* 40, 429－439

i　出典：Alexander(1964), p 94.　より筆者作成

ii　原文では“task partitioning”と表記。訳語は楠木(1997)、竹田(2000)を参考にした。

iii　原文では“stickiness of information”と表

記。訳語は高橋編(2000)を参考にした。

iv 出典：高橋（編）2000、p 207、表 19－1 より、一部筆者が改変した。

v 出典：ガルブレイス(1977)梅津訳、p 25、図 2－3 より筆者作成

vi 訳語。原文では“slack resource”と表記。訳語は梅津訳(1977)を参考にした。

vii 換言すれば問題を顕在化させないために経営資源を投入することであるとも言える。

viii 20 周年記念版。原著は Brooks(1975)である。

ix 出典：藤本(2001 b)、p 206、図 14.8。原典は Brooks(1995), pp.16－19, Fig 2.1、2.2、2.3、2.4 である。

x 加えて、高級車は高価格で、開発リードタイムを長く取ることが顧客より認められている。この点では調整負荷資源を投入しているといっても良いだろう。

xi 出典：藤本(2001 b)、p 281、図 16.10 より筆者作成。原典は Fujimoto(1989), p 454 a, Figure 8.12 である。

xii この対称性は、先述した Alexander(1964)が提示している“Program”と“Realization”の関係性との相似関係があるともいえる。

xiii 出典：クラーク、藤本(1993)、p 45、図 2－3 より筆者が一部改変の上作成。

xiv 原文では“modularization”と表記。“Modularization”に対する訳語としては「モジュール化」と「モジュラー化」がともに存在し、訳本であるボールドウィン、クラーク(2004)、安藤訳では「モジュール化」を使用している。本稿では、この概念が自動車産業などで多用される「モジュール化」とは一線を画すものであることを明示するため、「モジュラー化」を用いている。

xv Baldwin&Clark(2000)には「アーキテクチャのオープン化」という概念は存在していない。水平分業、モジュラークラスターの成立がモジュール化の必然的帰結であるという議論になっているのである。しかしこれは企業内部のみでの「クローズド」なモジュール化のケースも少なからず存在している。この点については藤本他（編）(2001)などで詳しく議論されている。