MMRC
DISCUSSION PAPER SERIES

MMRC-J-70

設計プロセスのシミュレーション分析に関する試論

—日本産業の比較優位特性はモデル化できるか—

東京大学 COE ものづくり経営研究センター
大隈 慎吾

東京大学経済学研究科 教授
藤本 隆宏

2006 年 6 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

設計プロセスのシミュレーション分析に関する試論
—日本産業の比較優位特性はモデル化できるか—

東京大学COEものづくり経営研究センター 特任研究員
大隈慎吾

東京大学経済学研究科 教授
藤本隆宏

2006 年 6 月

1. はじめに

ある製品の国際競争優位が、それを設計する側の組織能力（Clark and Fujimoto [1991], Fujimoto [1999]）と、設計される製品のアーキテクチャ（Ulrich [1995], Baldwin and Clark [2000]）の間の「相性」（fit）によって影響を受けるのではないか、という「アーキテクチャの比較優位仮説」は、一般的な実態観察に基づく仮説構築（藤本 [2001][2004]）、予備的な統計分析（藤本・大鹿 [2005]）などの形で展開されつつある。製品設計・工程設計に関する組織能力の地理的分布から製品ごとの国際競争優位を説明しようというこの試論は、既存の標準的な比較優位仮説、すなわち生産活動における国・製品ごとの要素生産性比（リカード）や製品の要素集約度と国の要素賦存度の相性（ヘクシャー＝オリーン＝サミュエルソン）で比較優位を説明する標準的な貿易論を補完する考え方を提供できる、と筆者らは考えている。

しかしながら、具体的に企業内における日々の設計のプロセスで起こっていることが、どのような経路で国際競争優位となって表れるのか、というミクロ的な説明は、まだ十分に展

開されてはいない。一般には、日本企業は開発チームの連携調整や開発リーダーのリーダーシップが強力であることが知られているが（Clark and Fujimoto [1991]、延岡 [1997]）、それが具体的に、どのような設計プロセスの特徴をもたらし、どのようにして製品開発の競争力として顕現しているのか、その経路は必ずしも明確ではなかった。

これに対して、藤本（2005）は、工学系の設計学、とりわけ公理系設計論（Suh [1990][2001], 中尾 [2003]、中尾・畑村・服部 [1999]）をヒントに、企業の設計活動を、「機能設計パラメータ群＝f（構造設計パラメータ群）」という連立方程式を解く問題にたとえて定式化し、これにより、統合型の開発組織がインテグラル型アーキテクチャにおいて競争優位を得る経路を、シンプルな例で素描した。この例では、まず、何らかの形で（たとえば不正確な連立方程式を解くことによって）構造設計パラメータ群の初期値を設定し、次に試行錯誤によって目標とする機能パラメータへ漸近させる、という「2 段階設計プロセス」に近い事例が現実にも多いと考え、これによって日本企業の設計開発活動に近似しようと考えた。

すなわち、日本企業の研究開発活動は、下流（開発による新製品・工程の創出）においては、統合型組織能力を活かした迅速な試行錯誤によるリードタイム短縮を特徴としているが、上流（研究による科学知識の創出）においては、オープンな科学者ネットワークによる科学知識創造を軽視する傾向がある（中馬 [2004]）、という定型的な事実（stylized fact）を出発点として、日本企業の持つアーキテクチャの優位性を「2 段階設計プロセスモデル」で説明しようとした。その課程で、上記のような組織能力を持つ日本企業は、中程度の複雑性を持ち科学知識の蓄積が十分なインテグラル・アーキテクチャ製品では国際競争力を持つが、モジュラー型アーキテクチャの製品では設計コストの比較優位がないこと、またその反面、科学知識が蓄積途上であるような科学集約的で複雑なインテグラル製品では日本が競争力を持たない可能性があることを例示した（藤本 [2005]）。以上は、実際に観察される事実をベースに組み立てた「アーキテクチャと組織能力の比較優位説」である。

しかしながら、実際に企業の設計現場で参与観察を行なって、上記のような「2 段階設計プロセス」の実証データを収集することは容易ではない。そこで本稿では、代替的なアプローチとして、上記の「公理系設計論」をベースにするシミュレーション分析を援用してみることにする。

本論で前提とする日本の製品開発に関する「定型化された事実」は以下の通りである。

① 日本企業は、戦後の生産資源が不足する中での急成長を通じて、長期雇用・長期取引をベースとする「統合型ものづくり」の組織能力を構築してきた。そうした企業では、チームによる製品開発が発達し、技術者間・チーム間の試行錯誤による設計最適化のスピ

ードも速い。

② 日本の消費者は品質にうるさいので、企業は、より高い精度で設計パラメータの最適化を行う必要がある。

③ 日本企業は、科学知識が形成途上にあるような先端商品よりも、あるていど科学知識は確立している「非ハイテク製品」で国際競争力を持つ傾向がある。

④ 日本企業は、中程度の複雑度を持つ「インテグラル」型製品で国際競争力を持つ傾向がある。

こうした「定型的事実」を、2 段階設計プロセスを基礎とするシミュレーションで再現できるかどうかが、本稿の課題である。仮に可能であるならば、国際比較優位というマクロ的なテーマに、設計プロセス論というミクロ的・工学的な基礎を与えるかもしれない。

以上のような問題意識と分析枠組みをベースに、本稿では、設計プロセスのシミュレーションを通じて、国際比較優位論に、設計論的な基礎を与える試みをしてみたいと考える。

2. シミュレーションの設定

本節では、以上の問題意識に基づいて、設計のプロセス（2 段階設計プロセス)、開発手法（コーディネーション型とコンペティション型)、開発パフォーマンス（ここではリードタイム)、そしてリードタイムに影響を与える要因である、組織の問題解決能力、科学知識のレベル、製品の複雑性（設計要素数とその相互依存関係)、市場ニーズの洗練度、のそれぞれについて、藤本（2005）に従い、シミュレーションの前提となるモデルを定義する。まず、公理系設計論の考え方にしたがって、設計プロセスを連立一次方程式の解を求めるプロセスとして近似する考え方について説明しよう。

2.1 公理的設計とは

公理系設計とは、工学系の一領域であり、設計される対象の固有技術的な違いを超えて、あらゆる人工物に共通して見られる一般的な設計プロセスを抽象的に定式化しようとの試みである。公理的設計の代表的な論者であるスー（Suh [1990][2001]）は、製品の使用者（消費者）が要求する諸機能を表すベクトル**FR**（functional requirement）と、製品の物理的な構成要素群（部品，材料等）の設計パラメータを示すベクトル**DP**（design parameter）の関係を次式のように定式化する。ここで、**A**は構造パラメータを要求機能に変換する定数群からなる行列である。単純化のため、FRベクトルもDPベクトルも要素数はｍであり、したがって行列**A**はｍ×ｍの行列であるとする。したがって、式（1）は全体として、m本の 1 次式からm個の設計パラメータ（DP_1～DP_m）の値を求める連立一次方程式の体系となる。

因果知識　　構造ﾊﾟﾗﾒｰﾀ　　要求機能

$$\mathbf{A} \quad \cdot \quad \mathbf{DP} \quad = \quad \mathbf{FR} \qquad (1)$$

情報　　物質　　現象

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & & \\ \vdots & & \ddots & \\ a_{m1} & & & a_{mm} \end{bmatrix} \qquad \mathbf{DP} = \begin{bmatrix} DP_1 \\ DP_2 \\ \vdots \\ DP_m \end{bmatrix} \qquad \mathbf{FR} = \begin{bmatrix} FR_1 \\ FR_2 \\ \vdots \\ FR_m \end{bmatrix}$$

（1）の定式化の実例として、Suh（2001）では、次のような、冷蔵庫の設計に関する事例が紹介されている。

$$\begin{bmatrix} FR_1 \\ FR_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} DP_1 \\ DP_2 \end{bmatrix}$$

FR_1：**冷蔵庫内の食料の出し入れができる，**	DP_1：**横開きのドア**
FR_2：**エネルギー損失を最小にする，**	DP_2：**ドアの断熱材**

この例では、2 つの要求機能FR_1とFR_2を実現するのに、2 つの構造パラメータDP_1とDP_2が寄与するか否かを｛「する」,「しない」｝＝｛1,0｝の 2 値で表す科学知識**A**が定義されている。しかし、行列**A**は常に 2 値で表されるわけではなく、以下のような自動車のホイールカバーの例（Suh [2001]）のように、連続値として表されることもある。

$$FR_1 = A_{11} \cdot DP_1$$

FR_1：**ホイールカバーの保持力**	DP_1：**⊿(リムの穴の径－バネの径)**
A_{11}：**バネの剛性**	

ここで、A_{11}はFR_1の変動に対するDP_1の変化率で表される係数である（FR_1はDP_1の変化に対し線形的に変化する）。もっとも、この例では要求機能が単一であるために、A_{11}は行列ではなく、連続的なスカラー量である。

また、前述の式（1）では、**FR** と **DP** の間に線形の関係 **A** が仮定されているが、現実の人工物における構造と機能の関係は線形で近似できるとは限らない。例えば、以下に示すバンのシートの組立設計事例では **A** が非線形関数となっている。

$$FR = f(DP_1, DP_2, \cdots, DP_{10})$$

FR：シートの前脚から後脚までの長さ　f：非線形関数
DP_1, …, DP_{10}：前脚と後脚のフックを連動させる部品群における10箇所のリンク

このように、公理的設計の理論一般は、実際のあらゆる設計事例に対応するため、様々な定式化を許容するものである。しかし、本稿の目的は個々の具体的事例に対する設計解を求めることではなく、製品設計の本質を十分なリアルさで近似するプロセス・モデルを提示することなので、単純化のため、上記の式（1）のような線形式で設計問題を定式化することにする[1]。

2.2　公理的設計による設計プロセスの定式化

公理設計に関する以上の基礎知識を踏まえて、設計者が現場で与えられる設計問題、そしてそれを解く設計プロセスを最も簡単な形で定式化してみよう。

前述の式（1）に戻って考えよう。公理的設計においては、式（1）の機能要件群**FR**が外部から与えられたとき、所与の因果知識**A**の下で、式（1）を満たす構造パラメータ群**DP**を求めるのが「設計プロセス」の本質であると考える。式（1）では、m本の連立一次方程式からm個の未知数（DP_1～DP_m）の値を求める形であったが、単純化のため 2 本、2 変数と仮定するならば、ある製品を設計するとは、**A**を 2×2 行列として、次のような問題を解くことに等しい。

設計者に与えられる問題例:「設計空間 **DP** において、下記を満たす **DP***を求めよ。」

$$\begin{matrix} \mathbf{FR^*} \\ \begin{bmatrix} FR_1^* \\ FR_2^* \end{bmatrix} \end{matrix} = \begin{matrix} \mathbf{A} \\ \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \end{matrix} \begin{matrix} \mathbf{DP} \\ \begin{bmatrix} DP_1 \\ DP_2 \end{bmatrix} \end{matrix} \qquad (2)$$

$$\Downarrow$$

$$\begin{cases} FR_1^* = a_{11} \cdot DP_1 + a_{12} \cdot DP_2 & (2\text{-}1) \\ FR_2^* = a_{21} \cdot DP_1 + a_{22} \cdot DP_2 & (2\text{-}2) \end{cases}$$

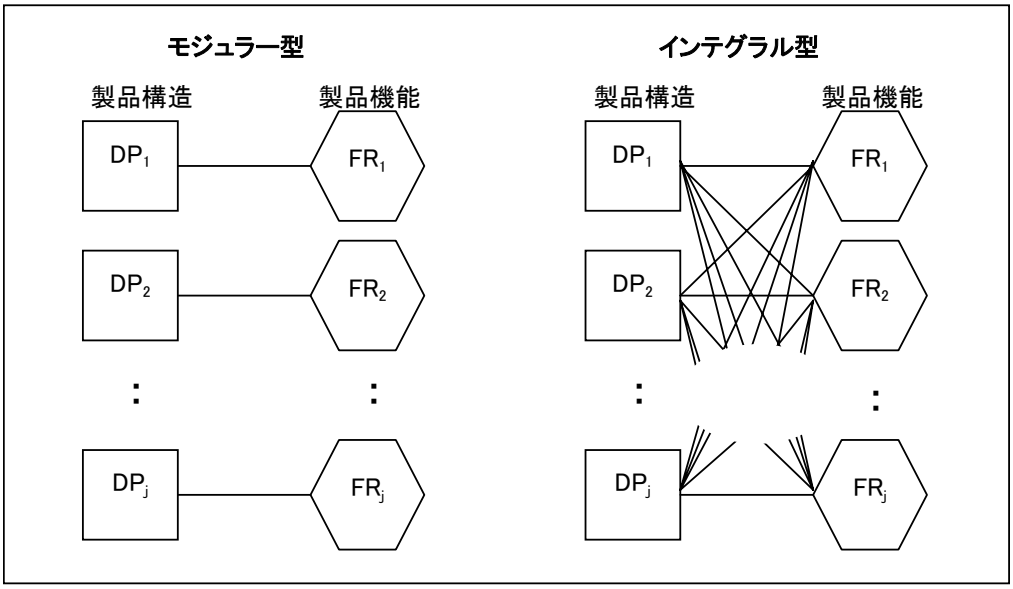
[1] ちなみに、一般に非線形として定式化される製品をそのまま設計するのは非常に困難であり、公理的設計においても「干渉設計」として忌避されるが、そうした場合、設計の現場では要求機能や構造パラメータを整理して、非線形の定式化を線形にすることが試みられる。また線形に変換することが事実上不可能な問題についても、例えば、上記したバンのシートの組立設計事例では、各DP_iを求める際に、他のDP_j (i≠j)を固定してDP_iのみを単独で設計し、これを順次繰り返すことによって真の解に漸近するという手法が採られている。これは、非線形の定式化を線形に近似したということである。したがって、設計者が直面する実際的な製品設計はほとんどが線形の定式化に還元しうるという意味で、(1) の線形的な定式化は一定の普遍性を持つといえよう。

ここで、設計空間**DP**とは、2 つの構造設計パラメータ（DP_1, DP_2）によって規定される 2 次元空間のことであり、これが製品の構造的な特性を示す。一方、機能**FR***は顧客から与えられる 2 つの機能要件を示すベクトルである。その間の関係は線形を仮定するので、行列**A**の要素a_{11}~a_{mm}は連続的なスカラー量の定数となる。

2.3 製品アーキテクチャ

次に、製品の設計思想すなわちアーキテクチャ、とりわけインテグラル（擦り合わせ）アーキテクチャと組み合わせ（モジュラー）アーキテクチャの区別を、行列 **A** の特性の違いとして規定することにしよう。

藤本（2005）などが指摘するように、アーキテクチャ理論において、いわゆる「インテグラル（擦り合わせ）型アーキテクチャ製品」[2] と、その対立的概念である「モジュラー（組み合わせ）型アーキテクチャ製品」[3] の定義は、次のような図で表すことができる。



つまり、1 つの製品機能要素FR_iを満たすのに関与する製品構造要素DP_i（部品、材料等）がただ 1 つであるのが純粋な「モジュラー型」であり、逆に、全ての製品機能要素FR_iに全ての製品構造要素DP_iが関係するのが純粋な「インテグラル型」ということになる。

これらをふまえて、公理的設計における「モジュラー型」と「インテグラル型」のアーキテクチャを定式化すると、式（1）における **A** の違いによって、両者を以下のように表現し

[2] 当該製品の設計を構成する各要素が、製品に要求される機能を媒介として、相互依存的にしか決定できない物を「インテグラル型製品」と呼ぶ。
[3] 当該製品の設計を構成する各要素が、製品に要求される機能と 1 対 1 に対応し、それぞれ独立に決定できる物を「モジュラー型製品」と呼ぶ。

分けることができる。つまり、純粋なモジュラー型は、対角線の要素以外はゼロとなる対角行列、純粋なインテグラル型は、全ての要素が非ゼロである行列となる。

実際に、(1)に上記 2 パターンの**A**を代入して連立方程式に展開すると、モジュラー型は各FR_iが単一のDP_iによって実現され、インテグラル型は各FR_iが複数のDP_iによって実現される形となることがわかるだろう。

モジュラー型　　**インテグラル型**

$$\mathbf{A}=\begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & & \\ \vdots & & \ddots & \\ 0 & & & a_{mm} \end{bmatrix} \qquad \mathbf{A}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & & \\ \vdots & & \ddots & \\ a_{m1} & & & a_{mm} \end{bmatrix} \tag{3}$$

3. 設計プロセスの定式化：2 段階モデル

3.1　2 段階設計プロセス・モデル

それでは、以上に示した公理的設計、およびアーキテクチャの諸概念を援用することによって、実際の設計活動に近似するモデルを定式化することはできるだろうか。

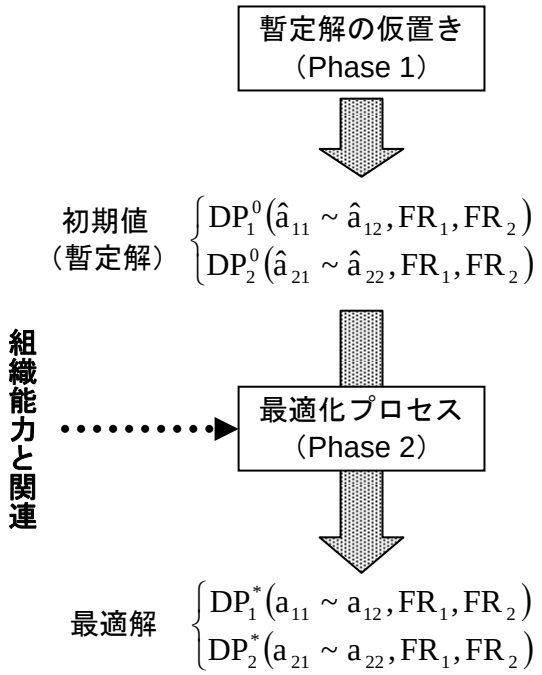
藤本（2005）では、いくつかの経験的事実から帰納することにより、実際の設計問題を解く際に用いられるアプローチ手法を「2 段階の設計プロセス・モデル」として近似的に描写している。すなわち、設計者は**図 1** に示すように、①出発点としての暫定解（初期値）の導出、②暫定解から最適解への漸近、という 2 段階を経て、例えば(2)式を解くと仮定される。

なぜ 2 段階を必要とするのか。それは、製品構造が製品機能を生み出す因果関係に関して設計者が持つ知識 **A** が、現実において不完全であるからである。したがって設計者は、現状で入手可能な、不完全な因果知識 **A'**（例えば公知の科学知識や既存製品の挙動に関する因果知識）を総動員して、「不完全な連立方程式」を解き、とりあえずの暫定解を得る。そして次に、その暫定解が目指す最適解に十分に近いことを願いつつ、試行錯誤（例えば試作品による実験）によって目指す最適解へと接近するのである。

図 1 の、$\hat{a}_{11} \sim \hat{a}_{22}$ は、不完全な因果知識**A'**の要素である。これに対して、真の因果知識を表す$\mathbf{A}^*$の要素 $a_{11} \sim a_{22}$ の値を、設計者は直接観察することができない。不完全な因果知識**A'**の精度（真の因果知識$\mathbf{A}^*$との乖離）は、設計者の持つ科学技術の知識の水準により決定されるが、一般に先端的な科学知識を多用する高度サイエンス型製品の場合、**A'**の精度は低くなりやすいし、科学知識へのアクセスのある者とない者では、**A'**の精度に大きな差が出やすい。逆に、そうした先端的科学知識を必要としない一般の製品の場合は、高度サイエンス型

製品に比べ、**A'**の精度はより高まる（真の因果知識$\mathbf{A}^*$との乖離はより小さくなる）傾向があるし、科学知識へのアクセスのある者とない者の差も出にくくなる。

図１　２段階の設計プロセス・モデル



以上に示した「2 段階設計プロセス論」による定式化を、あらためて整理しておこう。

第 1 段階：初期値、すなわち暫定設計解$\mathbf{DP}^0 = \left[DP_1^0, DP_2^0\right]^T$の導出には、過去における類似製品の設計経験や、共有知識である工学・科学の知見によって類推された不完全な因果知識**A'**が用いられるが、そうした「不完全な因果知識に基づく机上の設計」によって導かれる$\mathbf{DP}^0$と、あるべき最適設計解$\mathbf{DP}^*$の間には、当然誤差が生じることとなる。むろん、完全な因果知識$\mathbf{A}^*$が入手できていれば、$DP^0 = DP^*$となり、机上の計算のみで求めるべき最適設計解が得られるが、実際の設計作業でそのようなことは、ほとんどない。

とはいえ、科学知識へのアクセスが良いなどの理由で、他者よりも豊富な因果知識を持つ設計者は、そうでない設計者に比べて、最適設計解に比較的近い初期値を導出することが可能であろう。すなわち、**A'**と$\mathbf{A}^*$、そして$\mathbf{DP}^0$と$\mathbf{DP}^*$の乖離（設計空間上での距離）は、そうした科学知識を持った設計者の場合、より小さくて済むことが予想される。

第 2 段階：次に、試行錯誤によって最適解に漸近する最適化プロセスでは、実物試作による実験などを通じて物質や自然界に直接働きかけることにより、真の因果関係$\mathbf{A}^*$からの間接的なフィードバックが得られる。そこでは$\mathbf{A}^*$を直接観測することは出来ないが、任意の**DP**に

A*が作用した結果実現される**FR**を実験によって観察し、その**FR**と**FR***の乖離を小さくする方向に**DP**を変化させれば**DP***に漸近していくことが期待できるのである。

以上のような「2 段階設計プロセス・モデル」による定式化は、むろん、現実の設計プロセスのごく粗い近似に過ぎないが、少なくとも、設計という活動の本質的な部分を捉えているのではないかと考える。

3.2 最適解への接近法（1）：コーディネーション型

以上示した「2 段階モデル」の第 2 段階に当たる「試行錯誤による最適解への接近」は、大きく分けて、以下に示す「コーディネーション型（プロジェクト内の連携調整）」と「コンペティション型（複数案の並行開発と選抜）」の 2 つに類型化される。

まず「コーディネーション型」、つまりプロジェクト内部で設計者の連携調整によって最適解に接近する手法について、我々は式(2)のケースに関する、以下のような定式化を提案する。

（0.1）第 0 期において、因果知識 **A'**の精度に応じた初期値$\mathbf{DP}^{\mathbf{0}} = \left[\mathrm{DP}_1(0), \mathrm{DP}_2(0)\right]'$が与えられる。

（1.1）第 1 期において$\mathrm{DP}_2(0)$を所与とみなし、与えられた問題(2)の(2-1)式に代入し$\mathrm{DP}_1(1)$について解く。

（1.2）第 1 期において、上で得られた$\mathrm{DP}_1(1)$を所与とみなし、(2)の(2-2)式に代入して$\mathrm{DP}_2(1)$について解く。

（2.1）第 2 期において、第 1 期と同様に$\mathrm{DP}_1(3)$について解く

（2.2）第 2 期において、第 1 期と同様に$\mathrm{DP}_2(3)$について解く

⋮

（t.1）第t期において、上と同様に$\mathrm{DP}_1(t)$について解く

（t.2）第t期において、上と同様に$\mathrm{DP}_2(t)$について解く

⋮

以上のプロセスが想定される現実的な設計状況を考えてみよう。この場合、2 つの構造設計パラメータDP_1とDP_2についてそれぞれ専門の設計担当者、合計 2 名がおり、彼らが交互に自らの担当部分を修正し、それを全体構造に組み込んで試作を行った結果実現される機能$\mathbf{FR} = \left(\mathrm{FR}_1, \mathrm{FR}_2\right)$を観察しながら、それらを要求機能**FR***に近づけるよう協調して修正を繰り返していくケースに対応している。

このように定式化された「コーディネーション型」の理論的な解釈については、**図 2** を参照されたい。

図 2　コーディネーション型の模式図

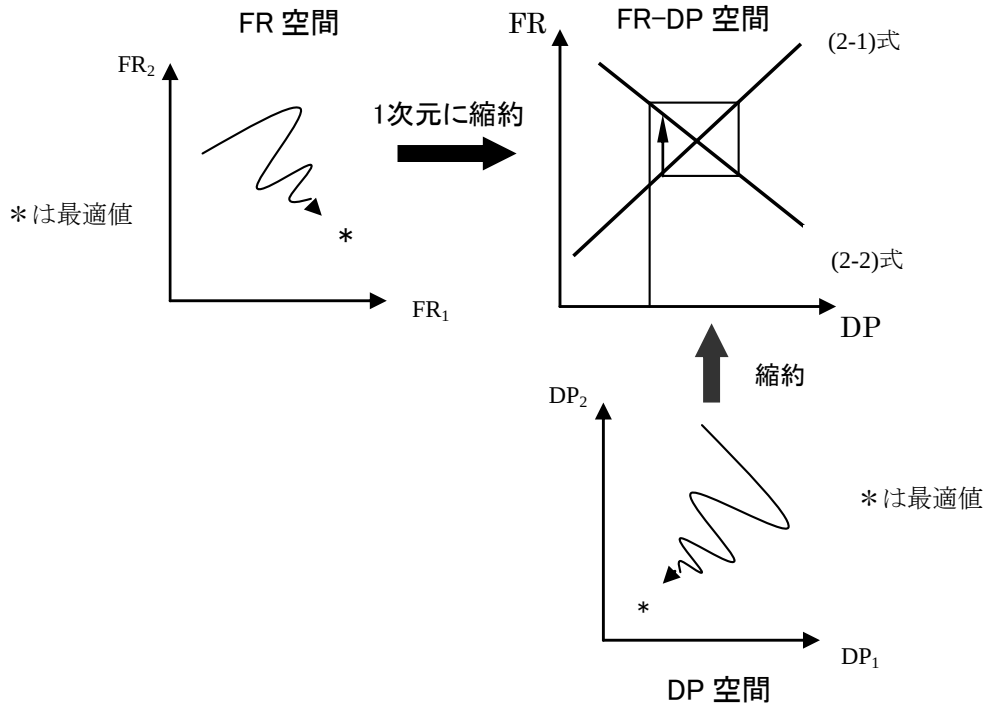


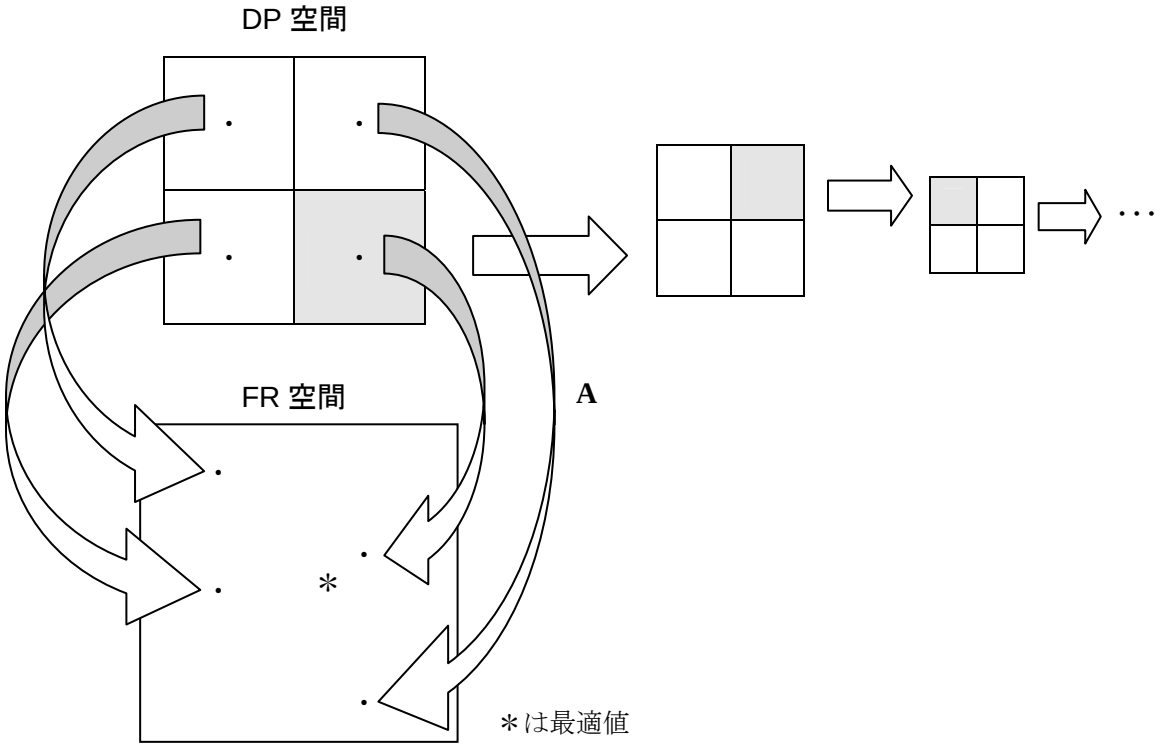
図 2 のような、**FR** ベクトルと **DP** ベクトルがスカラー変数に縮約できるケースを考えると、ミクロ経済学において、価格と数量の調整を通じて部分均衡に至るメカニズムとして紹介される「クモの巣理論」と同じ決定過程であることがわかる。

また、この場合、組織の中には、どの設計者が、どの順に、いつ設計変更に動いてよいのか、活動のシーケンスを決める上司が存在することが想定される。そうした「シーケンサー」的管理者の下で、複数の設計者が担当する設計パラメータを相互に調整し、最適解への接近を試みるのである。我々の予備的実態調査によれば、これは、実際の設計活動でもしばしば見られるパターンであるようだ。

3.2　最適解への接近法（2）：コンペティション型

次に、もう一つの最適解への接近法としてよく観察される、「複数案の並行開発と選抜」という手法を考えてみよう。この場合、設計空間上で、複数の設計案を並行開発で競わせる。その上で、要求機能に最も近い結果を出した設計案を選択する。そして、その枠の中で、さらに設計案を細かく分けた並行開発を行い、また、機能面でベストのものを選抜する。こうした、一種のトーナメントのような設計案の多段階選抜によって、最適解に漸近しようというわけである。具体的な定式化としては、以下の「4 分割法」を採用する。

図３　コンペティション（並行開発）型の模式図



（1）**DP**空間を 4 分割の部分空間に分け、その中心点の**A**による[4] **FR**空間上の写像点**FR'**を観測する。

（2）**FR'**と要求機能**FR***の乖離が最も小さかった部分空間を取り出す。

（3）取り出した部分空間を(1)と同様に、さらに 4 分割する。

（4）**FR'**と**FR***の乖離が最も小さかった部分空間を取り出す。

（5）以下、(3)～(4)の繰り返し。

現実的な設計プロセスとの対応を考えるならば、4 分割された部分空間にそれぞれ担当設計者がおり、彼らが競作で異なる設計案を並行開発している、という状況が想定できる。また、組織内には、設計を試作した結果実現した機能が要求機能に近いかどうかを判定する公平な評価者=選択者がいると暗に仮定されている。例えば、マツダの初代ロードスター開発において、FF（前輪駆動）型、MD（ミッドシップエンジン）型、FR（後輪駆動）型の設計が同時並行的に進められ、事後的にFRが採用された例が、典型的な並行開発・事後選抜に

[4] ここでの写像**A**は、不完全な因果知識**A'**ではなく真の因果知識**A***である。なぜなら、コンペティション型開発では、最初から実物試作による実験などを通じ、真の因果関係**A***からの間接的なフィードバックを得るからである。

あたる。[5]

4. パラメータの分類と整理

ここまでで、2 段階モデルの概要を説明したので、以下では、本稿のシミュレーションに対し外生的に与えられるパラメータの内容、およびその解釈と予想される結果について整理をしておく。

これまで見てきた、因果知識（ここでは科学知識に絞って考える）、製品アーキテクチャ、2 タイプの最適解への接近法（最適化プロセス）に加えて、2 つの変数を追加する。第 1 は、組織が最適解へ試行錯誤的な接近を行なう際の「試行スピード」である。こうした試行は組織的な活動であるから、一般に、情報共有が進み、チームワークの良い組織、つまり日本企業に多い「統合型ものづくり」の組織能力の高い組織において、スピードが速いと予想できる。

第 2 に、市場がどの程度洗練された製品を要求するか、つまりこの製品に関する顧客の鑑識眼がどのぐらい厳しいかを、「市場許容品質」で示す。すなわち、最適解にどこまで接近したときに顧客がその製品に「合格」の判定を下すかで、市場ニーズの洗練度を表現するのである。

4.1 事前の因果知識（とくに科学知識）レベルの指標

設計者が、前述の（2）の設計問題を与えられた時点での因果知識（とくに科学知識）のレベルは、**Aの不完全性**、つまり「2 段階の設計モデル」における、**A'**と**A***の誤差で表す。つまり、誤差が小さければ小さいほど設計者の事前の因果知識（科学知識など）のレベルは高い。

既に述べたように、一般に、先端科学知識を必要とする製品は、そうした科学知識自体がまだ発展途上であるから、因果知識のレベルは低い傾向があろう。そうした製品の場合、科学知識を豊富に持つ設計者とそうでない設計者の間で、初期値（暫定解）の出来に大きな差が出ることが予想できよう。

4.2 製品の複雑性の指標：要素数とアーキテクチャ

製品の複雑さを示すパラメータには、**A の要素数とアーキテクチャ（要素間の相互依存性）**がある。第 1 に、構造設計要素の数（公理系設計の枠組みでは機能設計要素の数に等しい）が多くなるほど複雑度は高いと仮定するのは自然であろう。第 2 に、同じ設計要素数であっ

[5] 平井他著（2003）、66～67 ページ参照。

ても、アーキテクチャがインテグラル型の場合、設計要素間の相互依存性が高いので、モジュラー型よりも複雑度が高いといえる。いずれにせよ、一般に、製品の複雑度が高くなると、設計の組織やプロセスも複雑になる。

4.3 組織プロセス・組織能力の指標

開発の組織プロセスおよび組織能力を測る指標には**最適化プロセスのタイプ**と**試行スピード**がある。そして、両者は連動する。

前述の通り、最適解に接近する開発プロセスには、コーディネーション（連携調整）型とコンペティション（並行開発）型があるが、これらが組織能力と関係するのは、各プロセスを実現するために必要な組織能力の属性が大きく異なるからである。すなわち、コーディネーション型で開発リードタイムを短縮化しようとするならば、各設計要素を担当する専門担当者が、自らの加えた設計変更に関する情報を次の担当者に迅速に伝達したり、試行結果に関する情報を全ての担当者が共有したりする必要がある等、必然的に担当者間での密なコミュニケーションが必要となる。一方、コンペティション型では、公平な裁定者がいれば、異なる設計案で競っている各担当者の間に必ずしも緊密なコミュニケーションは必要でない。つまり、統合型の組織能力の有無が大きく影響するのは、コーディネーション型の方であると予想される。

一方、試行スピードは、最適化プロセスにおいて試作品を完成させたり実験を行うスピードを表す。こちらは、担当者間の情報共有の度合いやチームワーク力が高いほどスピードが速いと考えるのが自然であり、したがって、日本企業に多い統合型の組織能力そのものを代理する変数と考えてよかろう。

4.4 市場条件の指標

製品の機能について、市場から要求される水準の厳しさを表す指標として、我々は**市場許容品質**を選んだ。本稿におけるシミュレーションでは、最適化プロセスによって実現される**FR'**と、消費者が理想的と考える要求機能**FR***との間の乖離（設計空間上での距離）が、ある一定の値以下に達した時点で要求機能を満たしたものと判定される。この場合の「ある一定の値」が、我々の言う「市場許容品質」である。[6] これを現実の市場の動きに当てはめる

[6] 工業加工では通常、設計図で指定された設計寸法に対して「寸法公差」が設定されている。例えば、厚さ 2.0mmのアクリル板を製造する場合、厳密に 2.0mmの製品を安価に大量生産するのは現実的に難しい。そこで企業は、例えば±0.2mmまでの誤差（つまり厚さ 1.8～2.2mmの製品）であれば適格品質である、というように社内のルールを定める。この上限・下限の基準を「公差」という。我々が使用する「市場許容品質」もこれと同様の考えに基づいている。すなわち、要求機能と実現機能の間で生じる誤差の許容範囲を示すのが「市場許容品質」である。ただし、公差は企業が自らの責任で設定す

ならば、この「市場許容品質」の値が小さければ、市場から要求される水準は厳しいということになり、逆に大きければ市場からの要求は緩いと解釈できる。

以上のように、我々は、公理系設計の枠組みを援用し、開発組織能力、開発手法、製品のアーキテクチャと複雑性、科学的因果知識のレベル、市場の評価の厳しさ、といった概念について、シミュレーション分析を前提とした定式化を行なった。以上で準備はできたので、次にシミュレーションの予備的な結果を示すことにする。

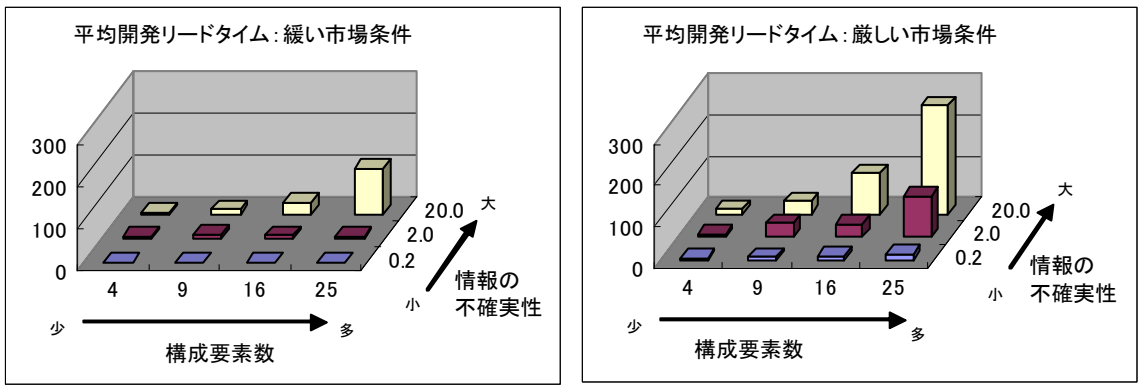
5. シミュレーション結果の素描：諸要因の平均リードタイムへの影響

シミュレーションに投入した外生パラメータの実装値や実行条件、詳細な実行結果については補論を参照されたい。それを用いた本格的な分析は今後の課題となるが、とりあえず、平均リードタイムに絞って、実行結果の予備的な考察を行なってみよう。

製品の構成要素数とアーキテクチャ：予想通り、製品の構成要素数（行列**A**の要素数）が多いほど、平均開発リードタイムが長いことが示された（**図 4**）。

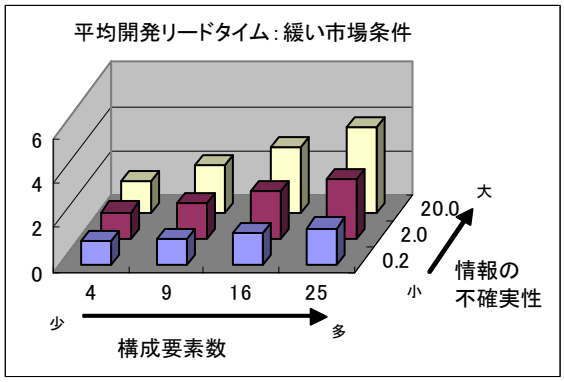
図 4

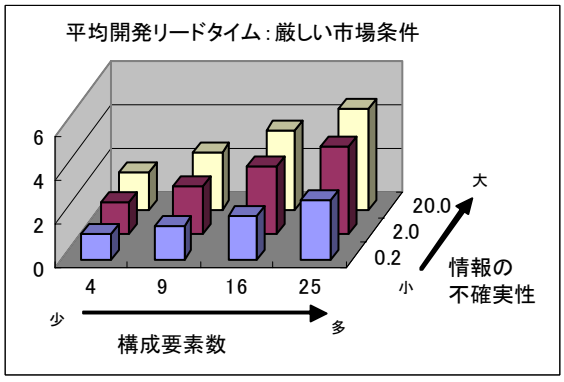
コーディネーション型ーインテグラル



るものであるが、市場許容品質は、顧客が購買の際に意思表示するものである。顧客自身が生産企業である場合は、その企業が設定する公差が市場許容品質となることもあるが、顧客が一般消費者である消費財の場合は、許容範囲に個人差が生じ、ばらつきが生じる（藤本、2001）。しかしながら本稿では、モデル化のために単純化して、「市場許容品質」の限界値は市場ごとに異なる一定値をとるものと仮定する。

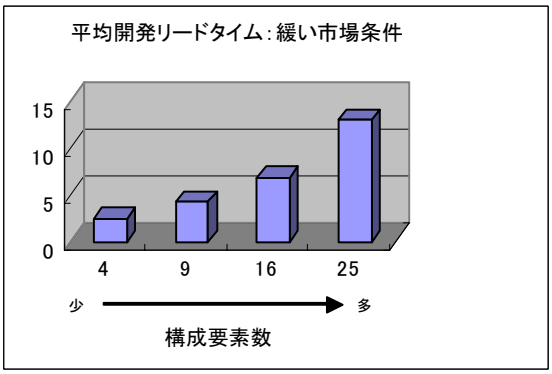
コーディネーション型－モジュラー

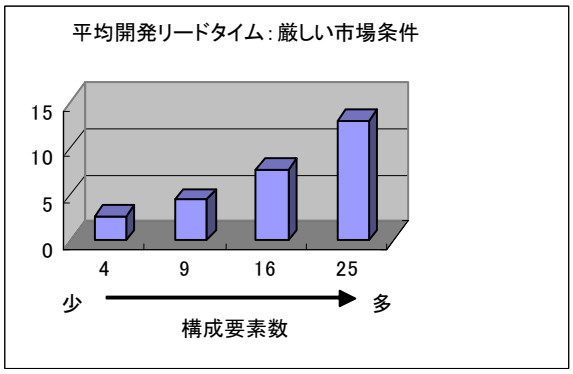




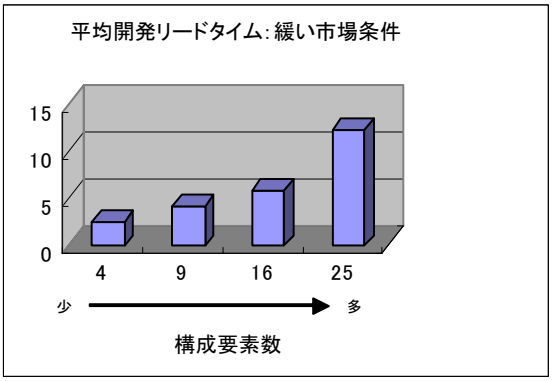
＊平均開発リードタイム（縦軸）がインテグラルの場合に比べ非常に小さいことに注意

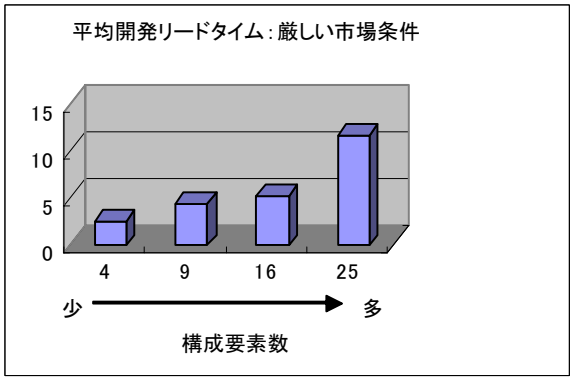
コンペティション型－インテグラル





コンペティション型－モジュラー





また、平均リードタイムは、モジュラー型製品の場合より、インテグラル製品の方がずっと長くなり、それは構成要素数が多い場合に特に顕著であることが分かった。言い換えれば、

モジュラー製品の場合、構成要素数にほぼ比例してリードタイムが長くなるのに対して、インテグラル製品の場合、要素数が増えるにしたがって、リードタイムは 2 次関数的に増加する傾向があるように見える。インテグラル製品の場合、要素間の相互依存関係の数は要素数の二乗に比例するので、この関係は予想と整合的といえよう。この結果、要素数が増えるとリードタイムのアーキテクチャ間での差が大きくなるのである。

因果知識（科学知識）の不完全性：因果知識を表す行列**A**の要素の分散で、因果知識の不完全性・不確実性を示すとするならば、因果知識が不完全なほど、平均リードタイムは長くなる傾向がある。とくに、インテグラル製品の場合、因果知識の不完全性が高まるにしたがって、リードタイムはどんどん長くなる傾向がある。

仮に、この因果知識の不完全性が科学知識の不足に起因すると考えるならば、科学知識が発展途上である科学知識集約型（ハイ・サイエンス）製品において、因果知識の不完全性が高いことになる。そして、因果知識が不完全であることがリードタイムに与える影響は、構成要素数が多いインテグラル製品で特に大きいということになる。

つまり、複雑なインテグラル製品の場合に、科学知識の多寡がリードタイムに特に大きく影響するのである。仮に科学知識に弱く試行錯誤に強い日本企業が存在すると仮定するなら、そうした企業にとっての鬼門は、むしろ複雑でインテグラルな科学知識集約型製品である、という藤本（2005）の予想と、この結果は整合的である可能性がある。

市場ニーズの厳しさ：製品に対する市場ニーズの厳しさは、理想の要求機能水準にどこまで近づいたら「適格」とみなすかを示す「市場許容品質」で代理している。シミュレーションの結果は、予想通り、市場ニーズの条件が厳しい方が平均リードタイムが長くなる傾向を示している。

最適解への接近方法：コーディネーション対コンペティション：コーディネーション型は、結果的に最適解に収束する場合には、コンペティション開発型よりも短期間で最適解に漸近する傾向が見られる。つまり、開発に成功する限り、設計リードタイムはコーディネーション型の方が短い。

しかし同時に、コーディネーション型の方が最適解から発散する（そもそも開発に失敗する）場合も多く見られる。他方、コンペティション（並行開発）型はほとんどのケースで最適解に漸近するが、その場合でも **FR** と要求機能 **FR*** のユークリッド距離はコーディネーション開発型が収束する場合に比べかなり大きい。

つまり、並行開発（コンペティション）型の場合、かなりリスクの高い開発でも確実に成

功にたどりつくが、その新製品の出来はそれほど良いものではない。他方、プロジェクト内の連携調整（コーディネーション）型の場合、開発に成功する場合はリードタイムも短いし製品の出来も良い。つまり効率的だが、反面、不確実性の高いプロジェクトの場合にはそもそも失敗してしまうリスクが馬鹿にならない。したがって、不確実性の高い荒れた状況では、並行開発型の方が、効率は悪いが手堅いといえる。これは、現実の製品開発でよくみられる状況、あるいは先行研究の知見と整合的である。[7]

これらを考えあわせると、最適化プロセスにおいては、コーディネーション型とコンペティション型のいずれか一方だけを実行するのではなく、最初にコンペティション（並行開発）型の設計を行い、次にコーディネーション型で最適解に向けより漸近するというのが最も効率的な設計プロセスといえよう。ただし逆のパターン、つまりコーディネーション型からコンペティション型という順序で実行してしまうと、最も効率が悪くなることが予想される。不確実性の高い製品開発の初期において並行開発手法が多用されるのは、現実の設計現場でもしばしば観察されることであり、シミュレーションの結果は現実の現象と整合的といえる。

6. 設計の比較優位論に関する考察と課題：「浅慮なウサギ」と「周到なカメ」の競走

6.1 「ウサギとカメ」の状況設定

以上、本稿では、公理的設計の諸概念を援用しつつ、2 段階設計プロセス・モデル（藤本[2005]）をベースにしたシミュレーション・モデルを作成し、具体的な条件を設定した上で実行した。このシミュレーション・モデルは、科学的知識のレベル、市場ニーズの厳しさ、製品の複雑さ、製品アーキテクチャ、製品開発の試行スピード（組織的問題解決能力）、開発プロセスのパターン（並行開発か連携調整か）などをモデルの中に明示的に取り込んでおり、その意味で、これまでの開発プロセスのシミュレーション・モデルに比べて、よりリアルなものになっていると考える。このモデルによるシミュレーション実行結果が、現実に観察される幾つかの現象を再現できるなら、このモデルを用いて、幾つかの予想や政策提言ができるようになるかもしれない。本稿は、そうした試論へむけた第一歩と位置づける。

そこで最後に、本稿冒頭の問題意識に立ち戻り、前節のシミュレーション結果から競争優

[7] 例えば、Iansiti（1998, p130-131）、平井他（2003, p66-67）、楠木（2001）など参照。また、藤本・安本（2000, 12 章）では、「複数の異なる要素技術を試作品で比較検討」（競争型開発・並行開発）という項目を含む多変数のアンケート統計分析を、多産業のプロジェクトを対象に行なっている。因子分析・回帰分析の結果によれば、前述の「並行開発」変数を含む「要素技術の早期集中探索」因子と「技術的な原因―結果の不確実性」因子の間には負の相関（1%水準）がみられる（p297）。つまり、不確実性と並行開発の間には明らかな相関があるという結果を得ている。

位にかかわるインプリケーションを幾つか導き出すことを試みる。

我々がこのモデルを使って最も知りたいことは、典型的な日本企業が設計開発競争、とりわけ設計スピード競争で勝ちやすいパターンと負けやすいパターンとは何であるか、ということである。そこでまず、典型的な日本のものづくり優良企業をモデル化しておく必要がある。具体的には、「統合型ものづくり」の組織能力が高いため、現場での試行錯誤のスピードが速い反面、現場のがんばりに依存しすぎる傾向があり、事前に体系的に科学知識を得るということにあまり熱心でない、という理念型としての日本企業である。中馬（2004）が描写した、現場は強いがサイエンスに弱い日本企業というイメージと重なるところが大きい。逆に、一部の欧米企業が、現場のスピードは遅いが、科学知識創造のネットワークを活用することに長けているとしたらどうか。つまり、仮に、多くの日本企業が、足は速いが事前によくものを考えない「浅慮なウサギ」であり、多くの欧米企業が、足は遅いが事前によく準備をする「周到なカメ」だとすれば、ウサギはどういうときに勝ち、どういうときに負けやすいのか。

ここで、上記の「ウサギとカメ」のアナロジーから、設計者を 2 つの類型、すなわち「浅慮なウサギ」と「周到なカメ」に分けて考えてみよう。「浅慮なウサギ」は「周到なカメ」に比べ持てる科学知識は乏しいが、試行や実験を行うスピードは「カメ」よりも速いと仮定される。すなわち、「脚の速い愚者」とでもいうべき設計者である。逆に、「周到なカメ」は「浅慮なウサギ」に比べ持てる科学知識は多いが、試行や実験を行うスピードは「ウサギ」よりも遅いと仮定される。これは、「脚の遅い賢者」と言うべき設計者である。これらを前節のパラメータを使って表現すると、因果知識（**A** の要素の分散）のレベルは「周到なカメ」の方が「浅慮なウサギ」より高いわけだから、「カメ」の方が「ウサギ」より **A** の分散は小さい。すなわち、$\sigma^2_{\mathrm{RBT}} > \sigma^2_{\mathrm{TTL}}$ である。一方、いったん試行錯誤による最適解への接近の段階に入ると、当然ながら「ウサギ」は「カメ」より速い。具体的に「カメ」は「ウサギ」の 3 分の 1 のスピードと設定しよう。すわなち、$\mathrm{s}_{\mathrm{TTL}} = (1/3)\mathrm{s}_{\mathrm{RBT}}$ である。

以上を前提に、どのような状況の時に「浅慮なウサギ」が勝ち、どのような時に「周到なカメ」が勝つのか、シミュレーションによって、予見を得たいわけである。

6.2　シミュレーションモデルの設定値と評価基準

本稿では以下において、前述の「ウサギ」と「カメ」を様々な状況下に置いて、リードタイムによる勝敗を詳細に観察するが、その前に、本稿で両者の勝敗を分析するためのフレームワークを、以下に示す**表 1** を例として、簡単に解説しよう。

表 1 は、本稿におけるシミュレーションプログラムに適当な条件を与えて実行し、そのと

きに観察された平均リードタイムから勝敗を判定した結果の簡易表である。ここで言う「適当な条件」とは、**表 1** における複雑性の指標（アーキテクチャ、構造要素数)、組織能力の指標（試行スピード、最適化プロセス)、不確実性（科学技術レベル)、市場条件（機能要求レベル）を指す。各条件は 4 節と補論 A.1 節に準じて決定されたが、以下でそれらと表中の表現との関係を説明する。

表 1

		複雑性の指標		組織の能力の指標		不確実性	市場条件		
製品名	設計組織	アーキテクチャ	構造要素数	試行ｽﾋﾟｰﾄﾞ	最適化プロセス	科学技術ﾚﾍﾞﾙ	機能要求ﾚﾍﾞﾙ	勝敗	ﾘｰﾄﾞﾀｲﾑ比
	浅慮なウサギ	インテグラル	中程度	速い	コーディネーション	低い	緩い	**勝**	1
	浅慮なカメ	インテグラル	中程度	遅い	コーディネーション	低い	緩い	**負**	3

まず、「複雑性の指標」における「アーキテクチャ」に関しては、純粋にインテグラルのケースと純粋にモジュラーのケースの 2 極端のみを設定する。次に、**A** の要素数によって決定される、「複雑性の指標」における「構造要素数」は、「多い（6 機能 6 部品)」「中程度（4 機能 4 部品)」「少ない（2 機能 2 部品)」の 3 段階を設定した。

アーキテクチャ
インテグラル
モジュラー

構造要素	A の要素数
多い	6×6=36
中程度	4×4=16
少ない	2×2=4

統合型の組織能力の強さ（チームワーク）を示す、「組織能力の指標」における「試行スピード」は、「遅い」を基準に、「速い」は 3 倍速、「非常に速い」は 9 倍速に設定し、3 段階を用意した。ちなみに、表中における「設計組織」で「ウサギ」と表現される場合は、「試行スピード」が「速い」もしくは「非常に速い」、「カメ」と表現される場合は「遅い」が必ず設定される。「組織能力の指標」における「最適化プロセス」の様式は、前述のように「コーディネーション（連携調整による単一案の修正)」と「コンペティション（複数案の並行開発の選抜)」の 2 パターンである。

試行スピード	プログラム処理スピード
非常に速い	s
速い	(1/3)s
遅い	(1/9)s

[8]

最適化プロセス
コーディネーション
コンペティション

[8] プログラム処理スピードの「ｓ」（基準スピード）については、補論A.1 を参照のこと。

次に、「不確実性」における「科学技術レベル」は、因果知識を表すマトリックス**A**の要素の分散[9] で示している。不確実性のレベルを小・中・大の 3 段階で設定し、分散の値を小で 0.2、中で 2、大で 20 とした。ここで言う「不確実性」とは、機能・構造の因果関係の不確実性のことであるが、因果関係の不確実性とは、大まかに言えば「製品設計が必要とする因果知識（情報処理必要量)」と「開発組織が有する因果知識（情報処理能力)」の差だと解釈できる（Galbraith, 1973)。したがって、製品を所与とするならば、因果知識（例えば科学知識）を多く持つ組織は低い不確実性、因果知識（科学知識）を持たない組織は高い不確実性に直面する。一方、開発組織の知識量を一定と仮定するならば、多くの科学知識を必要とするハイテク製品において不確実性が高く、新たな科学知識をあまり必要としないローテク製品において不確実性が低い、といえる。さらに、製品と組織の組み合わせを考えるならば、当然ながら、ハイテク製品と低知識組織の組み合わせが最も不確実性が高い。言い換えれば、ハイテク製品の方が、組織間で不確実性の差が大きく出やすい。以上をふまえ、本シミュレーションでは、下表で示す代表的な 3 パターンを選び[10]、それぞれに**A**の分散値を設定した。

因果知識必要量	因果知識保有量	不確実性	A の分散
多（ハイテク製品）	少（低知識組織）	高	20.0
中	中	中	2.0
少（ローテク製品）	多（高知識組織）	低	0.2

「市場条件」における「機能要求レベル」は 4 節における市場許容品質のことであるが[11]、品質にうるさいお客が多い「厳しい市場」では要求機能**FR***との間の乖離が 1.0 以内でなければならず、品質にうるさくない「緩い市場」では**FR***との乖離が 3.0 以内でもよい、として値を設定した。

機能要求レベル	市場許容品質
厳しい	1.0
緩い	3.0

本稿では以下、これらの設定値が異なる 2 つの主体をシミュレーションによって競争させ、結果としてのリードタイムのパフォーマンスを比較することにする。すなわち、**表 1** の「リードタイム比」は、シミュレーションの実行結果から得られた開発リードタイムの比（概数）

[9] 「Aの要素の分散」については、補論A.1 を参照のこと。
[10] 表中の「因果知識必要量」は「当該製品の設計が必要とする因果知識（情報処理必要量)」であり、「因果知識保有量」は「開発組織が有する因果知識（情報処理能力)」である。
[11] 「市場許容品質」については、4.4 節を参照のこと。

である。[12] また、「勝敗」については、逃げ切った方の平均開発リードタイムを基準として、負けた方のリードタイムがその 1.0 倍より大きく 2.0 倍未満であった場合は、勝った方を「辛勝」負けた方を「惜敗」、2.0 倍以上 3.0 倍以下の場合は「勝」と「負」、3.0 倍よりも大きな場合は「大勝」「大敗」と表記した。

ここで、**表 1** を改めて見てみると、試行スピード以外の設定条件が同じならば、**A** の要素値や要求機能 **FR*** に設定される乱数がランダムな影響を与えるにもかかわらず、ウサギがカメよりも 3 倍速いという試行スピードの設定がそのまま平均リードタイムに表れていることがわかる。このことから、リードタイム比に 3 倍以上の差をつけて勝った場合には、所与の条件として与えられた 3 倍のスピード格差をさらに広げることに成功しているので「大勝」と表現している。逆に、仮に勝ったとしても、当初 3 倍の格差を与えられていながら、1 倍近くまで追い上げられたケースには「辛勝」と表現することにした。

6.3 シミュレーション結果 1：インテグラル型製品の連携調整型開発

レース 1.1：「浅慮なウサギ」対「周到なウサギ」： 製品アーキテクチャはインテグラル（例えば自動車）；最適化プロセスはコーディネーション（連携調整）型；設計者はすべて「ウサギ」並に速い（$s_{RBT_1} = s_{RBT_2}$）；ウサギの中に、科学知識を深く持った「周到なウサギ（RBT_2）」と、持たない「浅慮なウサギ（RBT_1）」がいる（$\sigma^2_{RBT_1} > \sigma^2_{RBT_2}$）。これは、いわば、開発スピードの速い日本の自動車企業の中に、科学知識へのアクセスに熱心な会社と不熱心な会社がある、という状況である。

→ 結果： 「周到なウサギ」が先に最適値の近傍に到達する傾向が大きい。つまり、試行スピードが同じならば「賢者」である「周到なウサギ」が「浅慮なウサギ」より先に要求機能**FR***を満たす設計を実現する。新技術の重要性を増しつつある 21 世紀の自動車に、この状況が当てはまりそうである。つまり、自動車メーカーといえども、今後は科学知識ネットワークへのアクセスが極めて重要になる。

		複雑性の指標		組織の能力の指標		不確実性	市場条件		
製品名	設計組織	アーキテクチャ	構造要素数	試行ｽﾋﾟｰﾄﾞ	最適化プロセス	科学技術ﾚﾍﾞﾙ	機能要求ﾚﾍﾞﾙ	勝敗	ﾘｰﾄﾞﾀｲﾑ比
自動車	周到なウサギ	インテグラル	中程度	速い	コーディネーション	**高い**	厳しい	**大勝**	1
	浅慮なウサギ	インテグラル	中程度	速い	コーディネーション	**低い**	厳しい	**大敗**	3.1

レース 1.2：「浅慮なウサギ」対「周到なカメ」： 製品アーキテクチャはインテグラル（例

[12] ただし、プログラム処理スピードが(1/9)sのケースにおけるリードタイムは、(1/3)sのリードタイムがsのほぼ正確に 3 倍となることから、(1/3)sの 3 倍として推計した。

えば半導体製造装置）；最適化プロセスはコーディネーション（連携調整）型；設計者には「浅慮なウサギ」と「周到なカメ」がいる。「ウサギ」は 3 倍速い（$s_{TTL} = (1/3)s_{RBT}$）；しかし、事前の科学知識のレベルは「カメ」が上である（$\sigma^2_{RBT} > \sigma^2_{TTL}$）。これは、いわば、開発の試行スピードは速いが科学知識獲得が苦手な日本の半導体製造装置メーカーと、開発試行スピードでは日本企業に負けるが科学知識へのアクセスでは勝る欧州企業がいるという、中馬（2004）が半導体露光装置に関して示した状況に近い。つまり、周到なカメが設計プロセスの第 1 段階（科学的知識による暫定解設定）における優位を活かして逃げ切れるか、あるいは、浅慮だが足の速いウサギが、第 1 段階の遅れを跳ね返して、第 2 段階でカメを逆転できるか、という点に、この勝負の本質がある。

→ 結果： ある閾値$\sigma(>1)$に対して、ウサギのカメに対する相対的なスピード優位がその一定値以内ならば（すなわち$s_{RBT}/s_{TTL} < \sigma$ならば）、**「周到なカメ」**が先に最適値の近傍に到達する傾向が大である。つまり「カメ」（この場合は欧州企業）の「逃げ切り勝ち」となりやすい。まさに、「寓話ウサギとカメ」の教訓どおりである。一方、ウサギのスピードが一定の閾値を超えてカメより圧倒的に速いならば（すなわち$s_{RBT}/s_{TTL} > \sigma$ならば）、ウサギが先に最適値の近傍に到達する傾向が大である。つまり、ウサギの「逆転勝ち」となりやすい。ただし、製品の複雑性（= m）が増加すると、再び、「カメ」が逃げ切り勝ちする傾向が大となる。つまり、非常に複雑なインテグラル製品であって、しかも科学技術集約的な製品（例えば半導体露光装置）で、日本企業が開発競争で劣勢になる可能性がある、という中馬（2004）の指摘とラフながら整合的な結果が、シミュレーションで得られたのである。

		複雑性の指標		組織の能力の指標		不確実性	市場条件		
製品名	設計組織	アーキテクチャ	構造要素数	試行ｽﾋﾟｰﾄﾞ	最適化プロセス	科学技術ﾚﾍﾞﾙ	機能要求ﾚﾍﾞﾙ	勝敗	ﾘｰﾄﾞﾀｲﾑ比
半導体製造装置	浅慮なウサギ	インテグラル	中程度	**速い**	コーディネーション	**低い**	緩い	**惜敗**	1.4
	周到なカメ	インテグラル	中程度	**遅い**	コーディネーション	**高い**	緩い	**辛勝**	1

ただし、ウサギの試行スピードがある閾値を超えて非常に速くなると、第 1 段階のハンディにもかかわらず、ウサギによる逆転が可能になる。

		複雑性の指標		組織の能力の指標		不確実性	市場条件		
製品名	設計組織	アーキテクチャ	構造要素数	試行ｽﾋﾟｰﾄﾞ	最適化プロセス	科学技術ﾚﾍﾞﾙ	機能要求ﾚﾍﾞﾙ	勝敗	ﾘｰﾄﾞﾀｲﾑ比
半導体製造装置	浅慮なウサギ	インテグラル	中程度	**非常に速い**	コーディネーション	**低い**	緩い	**勝**	1
	周到なカメ	インテグラル	中程度	**遅い**	コーディネーション	**高い**	緩い	**負**	2.2

しかし、製品の複雑性がさらに増すと、いかに足の速いウサギでも、周到なカメに追いつけなくなる。組織的な問題解決能力の高さと、それがもたらす試行錯誤スピードの速さに頼る日本企業が、非常に複雑で先端科学集約的なインテグラル・アーキテクチャ製品で開発競争に敗れる恐れがある、という状況が、このシミュレーション結果でも再現されているのである。

		複雑性の指標		組織の能力の指標		不確実性	市場条件		
製品名	設計組織	アーキテクチャ	構造要素数	試行ｽﾋﾟｰﾄﾞ	最適化プロセス	科学技術ﾚﾍﾞﾙ	機能要求ﾚﾍﾞﾙ	勝敗	ﾘｰﾄﾞﾀｲﾑ比
半導体製造装置	浅慮なウサギ	インテグラル	**多い**	**非常に遅い**	コーディネーション	低い	緩い	**大敗**	∞
	周到なカメ	インテグラル	**多い**	**遅い**	コーディネーション	高い	緩い	**大勝**	1

[13]

6.4 シミュレーション結果２：モジュラー型製品の連携調整型開発

レース 2.1 「浅慮なウサギ」対「周到なウサギ」： 製品アーキテクチャはモジュラー型（例えばPC）；最適化プロセスはコーディネーション（連携調整）型；設計者はすべて「ウサギ」並に速い（$s_{RBT_1} = s_{RBT_2}$）；ウサギの中に、科学知識を深く持った「周到なウサギ（RBT_2）」と、持たない「浅慮なウサギ（RBT_1）」がいる（$\sigma^2_{RBT_1} > \sigma^2_{RBT_2}$）。これは、いわば、開発スピードの速いPCメーカーの中に、科学知識へのアクセスに熱心な会社と不熱心な会社がある、という状況である。仮に、「周到なウサギ（RBT_2）」はアメリカのPC企業、「浅慮なウサギ（RBT_1）」は中国のPC企業だとしよう。

[13] シミュレーション実行結果を見ると、「浅慮なウサギ」は 2 回市場許容品質を満たしている。しかし 1000 回の試行中に 2 回というのはサンプル数が極端に少ないため、このときのリードタイムの平均値は統計的に信頼できないとして除外した。これは次のような理由による。

このときの「浅慮なウサギ」のリードタイム平均は、補論 A.2 の**表 A1** における、{アーキテクチャ：インテグラル、行列 **A** の要素数：6×6、最適化プロセス：コーディネーション、**A** の不完全性：20.0} の「平均適格 STEP 数（緩）」に相当する。**表 A1** によると、確かに、この実行条件下における「適格回数（緩）（1000 回中）」（市場許容品質を満たした回数）は 2 である。ところで、この { } 内の実行条件のうち「**A** の要素数」のみを変更すると、リードタイム平均＝「平均適格 STEP 数（緩）」はどのように変化するだろうか。

A の要素数	**市場許容品質を満たした回数**	**リードタイム平均**	**標準偏差**
2×2	642	4.62	14.7
3×3	299	10.99	27.19
4×4	91	29.48	85.23
5×5	18	107.11	227.03
6×6	2	1.00	0

上の表は、補論 A.2 の**表 A1** から必要な情報だけを取り出して再構築したものであり、実行条件は、「**A** の要素数」のみを除いて、全て前述の { } 内と同じである。これを見ると、2×2 ～ 5×5 まではリードタイム平均（平均適格 STEP 数（緩））が指数的に上昇する傾向が見られ、6×6 だけがその傾向から外れていることがわかる。このときのリードタイム平均 1.00 が統計的に有意であるかどうかは、標本サイズが 2 であることから判断不能である。ちなみに、6×6 も 2×2 ～ 5×5 までの傾向に従うとするなら、リードタイム平均は 619.88（2×2 ～ 5×5 のリードタイム平均の対数をとり、A の要素数に単回帰した結果から得られた、6×6 のリードタイム平均の推定値）に近い値となるであろう。実際、浅慮なウサギのケースについて追加的に行ったシミュレーションの試行（1000SET）では、この推定値に比較的近いリードタイム 413 が得られた。したがって、リードタイムの母平均は 1.00 よりもはるかに大きいが、上記の表の試行では偶然それとかけ離れた値が出たと考え（**A** の要素に加えられる正規乱数として稀に 0 に近い値が生成された場合には、構造要素 **DP** の初期値が真の解に極めて近くなり、このような事が起こりうる）、本節における考察では除外した。つまり、「浅慮なウサギ」は 1 度も市場許容品質に達することなく、永久に最適化プロセスを実行し続けるという意味で、リードタイムを「∞」と表記した。

→ 結果： 純粋モジュラー製品の場合、「周到なウサギ」が先に最適値の近傍に到達する傾向が大きい。その点はインテグラル製品のケースであるレース 1.1 と同じである。ただし、レース 1.1 では「周到なウサギ」が「浅慮なウサギ」に大差をつけて逃げ切っていたが、ここではそこまでの差はない。また、「浅慮なウサギ」が勝つ場合も往々にして見られる。

		複雑性の指標		組織の能力の指標		不確実性	市場条件		
製品名	設計組織	アーキテクチャ	構造要素数	試行ｽﾋﾟｰﾄﾞ	最適化プロセス	科学技術ﾚﾍﾞﾙ	機能要求ﾚﾍﾞﾙ	勝敗	ﾘｰﾄﾞﾀｲﾑ比
PC	周到なウサギ	モジュラー	少ない	速い	コーディネーション	**高い**	緩い	**辛勝**	**1**
	浅慮なウサギ	モジュラー	少ない	速い	コーディネーション	**低い**	緩い	**惜敗**	**1.4**

つまり、周到な科学知識を事前に持つことのメリットは、モジュラー製品よりもむしろインテグラル製品の方が大きい、という一見パラドックス的な結果が得られたのである。これの示唆するところは大きい。つまり、仮に日本企業がインテグラル型製品に比較優位を持っており、その製品で科学知識に基づく技術進歩が見られるのであれば、日本企業は、そうした得意なインテグラル製品でこそ、科学知識の探索を周到に行い、自社の競争優位を防衛する必要があるのである。

レース 2.2 「浅慮なウサギ」対「周到なカメ」： 製品アーキテクチャはモジュラー（例えばスーパーコンピュータ）；最適化プロセスはコーディネーション（連携調整）型；設計者には「浅慮なウサギ」と「周到なカメ」がいる。「ウサギ」は 3 倍速い（$s_{TTL} = (1/3)s_{RBT}$）；しかし、事前の科学知識のレベルは「カメ」が上である（$\sigma^2_{RBT} > \sigma^2_{TTL}$）。

→ 結果： これは、既に見たレース 1.2（半導体製造装置）の製品アーキテクチャをインテグラルからモジュラーに変えた状況である。この場合は、インテグラル製品の場合（レース 1.2）とは違って、「浅慮なウサギ」が先に最適値の近傍に到達する、つまりカメに逆転勝ちする傾向が大である。製品が複雑になっても（**A**の要素数が増加しても）、この傾向は変わらない。つまり、モジュラー型のケースでは、(2)式の問題が簡単になるため知識によるパフォーマンスの差は出ないので、カメとウサギの距離差は大きくない。したがって、「脚の速い愚者」であるウサギが比較的容易に逆転できるのである。

		複雑性の指標		組織の能力の指標		不確実性	市場条件		
製品名	設計組織	アーキテクチャ	構造要素数	試行ｽﾋﾟｰﾄﾞ	最適化プロセス	科学技術ﾚﾍﾞﾙ	機能要求ﾚﾍﾞﾙ	勝敗	ﾘｰﾄﾞﾀｲﾑ比
スーパーコンピュータ	浅慮なウサギ	モジュラー	多い	**速い**	コーディネーション	**低い**	厳しい	**勝**	**1**
	周到なカメ	モジュラー	多い	**遅い**	コーディネーション	**高い**	厳しい	**負**	**2.7**

6.5 シミュレーション結果３：インテグラル型製品の並行競争型開発

レース 3.1 「浅慮なウサギ」対「周到なウサギ」： 製品アーキテクチャはインテグラル（例えば医薬品）；最適化プロセスはコンペティション（並行開発）型；設計者はすべて「ウサギ」並に速い（$s_{RBT_1} = s_{RBT_2}$）；ウサギの中に、科学知識を深く持った「周到なウサギ（RBT_2）」と、持たない「浅慮なウサギ（RBT_1）」がいる（$\sigma^2_{RBT_1} > \sigma^2_{RBT_2}$）。つまり、前述のレース 1.1 と似た状況だが、コーディネーション（連携調整）ではなく、コンペティション（並行開発競争）が最適解へ接近する方法として採用されている。

→ 結果： 「周到なウサギ（RBT_2）」、「浅慮なウサギ（RBT_1）」の双方とも、先に最適値の近傍に到達する確率は等しく 1/2 となる。このケースでは、両者の持つ属性の差は設計パフォーマンスに影響しない。どちらが勝つかは試行する度に変わる。つまり、トーナメント式の開発である場合、勝ち負けは時の運となり、強いて言えばたくさん勝負できる大企業が優位になるだろう。これは、医薬品の、とくに探索段階の状況に近いかもしれない（桑嶋[1999]）

		複雑性の指標		組織の能力の指標		不確実性	市場条件		
製品名	設計組織	アーキテクチャ	構造要素数	試行ｽﾋﾟｰﾄﾞ	最適化プロセス	科学技術ﾚﾍﾞﾙ	機能要求ﾚﾍﾞﾙ	勝敗	ﾘｰﾄﾞﾀｲﾑ比
医薬品	周到なウサギ	インテグラル	多い	速い	コンペティション	高い	厳しい	不定	1
	浅慮なウサギ	インテグラル	多い	速い	コンペティション	低い	厳しい		1

レース 3.2 「浅慮なウサギ」対「周到なカメ」：製品アーキテクチャはインテグラル（例えば半導体製造装置）；最適化プロセスはコンペティション（並行開発）型；設計者には「浅慮なウサギ」と「周到なカメ」がいる。「ウサギ」は 3 倍速い（$s_{TTL} = (1/3)s_{RBT}$）；しかし、事前の科学知識のレベルは「カメ」が上である（$\sigma^2_{RBT} > \sigma^2_{TTL}$）。前述のレース 1.2（半導体露光装置）において最適設計解への接近プロセスが連携調整から並行開発に切り替わったケースである。

→ 結果： 「浅慮なウサギ」が先に最適値の近傍に到達する傾向が大きい。しかし、製品が複雑化（**A**の要素数が増加）すると、ウサギ、カメともに最適解の近傍に到達する確率は減少する。ただしその到達確率は、概してコーディネーション型のときよりも大きい。効率は悪いが手堅い「並行開発」の特徴が出ている。要するに、コンペティション（並行開発）型では、持てる知識の差（**A**の不完全性の差）が全く影響しないので「浅慮なウサギ」（脚の速い愚者）」が常に有利となる。ただし、製品の複雑性が増すと、「賢者（カメ）」「愚者（ウサギ）」の双方ともに要求機能を最後まで満た

すことができないパターンが増えてくる（それでも、要求機能に行き着く場合は「浅慮なウサギ」の方が短期間のうちに到達するのだが)。その場合、たまたま「浅慮なウサギ」が要求機能に行き着けず永久に足踏みを続けるパターンに陥ってしまい、「周到なカメ」（賢者）の方が要求機能を満たすパターンの経路に乗っていれば「カメ」が勝つということが起こりうる。ただし、医薬品に関して言う限り、「浅慮なウサギ」に当たるのは、日本企業ではなく、むしろ力技（例えばコンビナトリアル・ケミストリー）によって開発の試行錯誤期間を短縮しようとしている、欧米の巨大医薬企業であるように見える。

		複雑性の指標		組織の能力の指標		不確実性	市場条件		
製品名	設計組織	アーキテクチャ	構造要素数	試行ｽﾋﾟｰﾄﾞ	最適化プロセス	科学技術ﾚﾍﾞﾙ	機能要求ﾚﾍﾞﾙ	勝敗	ﾘｰﾄﾞﾀｲﾑ比
半導体製造装置	浅慮なウサギ	インテグラル	中程度	速い	コンペティション	低い	厳しい	勝	1
	周到なカメ	インテグラル	中程度	遅い	コンペティション	高い	厳しい	負	3

ただし構造要素数が「多い」の場合は、ごくたまに「周到なカメ」が勝つことがある。

6.6 シミュレーション結果４：モジュラー型製品の並行競争型開発

レース 4.1 「浅慮なウサギ」対「周到なウサギ」： 製品アーキテクチャはモジュラー型（例えばPC）；最適化プロセスはコンペティション（並行開発）型；設計者はすべて「ウサギ」並に速い（$s_{RBT_1} = s_{RBT_2}$）；ウサギの中に、科学知識を深く持った「周到なウサギ（RBT_2)」と、持たない「浅慮なウサギ（RBT_1)」がいる（$\sigma^2_{RBT_1} > \sigma^2_{RBT_2}$)。競争主体や製品の特性は前述のレース 2.1 と基本的に同じだが、最適設計解への接近プロセスが連携調整から並行開発に切り替わったケースである。

→ 結果： 「周到なウサギ（RBT_2)」「浅慮なウサギ（RBT_1)」の双方とも、先に最適値の近傍に到達する（製品開発に成功する）確率は等しく 1/2。このケースでは、両者の属性の差は設計パフォーマンスに影響しない。どちらが勝つかは試行する度に変わる。つまり、レース 3.1 と同様、科学知識の差は結果に影響せず、両者の間に設計リードタイムの差はない。勝負は時の運ということになる。

		複雑性の指標		組織の能力の指標		不確実性	市場条件		
製品名	設計組織	アーキテクチャ	構造要素数	試行ｽﾋﾟｰﾄﾞ	最適化プロセス	科学技術ﾚﾍﾞﾙ	機能要求ﾚﾍﾞﾙ	勝敗	ﾘｰﾄﾞﾀｲﾑ比
PC	周到なウサギ	モジュラー	中程度	速い	コンペティション	高い	緩い	不定	1
	浅慮なウサギ	モジュラー	中程度	速い	コンペティション	低い	緩い		1

レース 4.2 「浅慮なウサギ」対「周到なカメ」：製品アーキテクチャはモジュラー（例えばスーパーコンピュータ）；最適化プロセスはコンペティション（並行開発）型；設計者には「浅慮なウサギ」と「周到なカメ」がいる。「ウサギ」は 3 倍速い（$s_{TTL} = (1/3)s_{RBT}$）；し

かし、事前の科学知識のレベルは「カメ」が上である（$\sigma^2_{\mathrm{RBT}} > \sigma^2_{\mathrm{TTL}}$）。競争主体や製品の特性は前述のレース 2.2 と基本的に同じだが、最適設計解への接近プロセスが連携調整から並行開発に切り替わったケースである。

→ 結果：「浅慮なウサギ」が先に最適値の近傍に到達する傾向が大である。製品の複雑性（**A**の要素数）が増加しても、この傾向は変わらない。また、$\sigma^2_{\mathrm{RBT}} \fallingdotseq 0$ であっても、つまり製品が極端に標準化した場合でも、この傾向は変わらない。このケースでは絶対的に「浅慮なウサギ」（脚の速い愚者）の方が有利となる。コーディネーション型の場合（レース 2.2）のように、因果知識**A**の不完全性がほとんど無いケースでも、「周到なカメ」が勝つという例外はない。1980 年代、スーパーコンピュータの開発スピードにおいて、並行開発を多用した日本の某エレクトロニクスメーカーは、開発スピードにおいて米国のライバルに対して大きな競争優位を持っていたが（Iansiti [1998]）、そのケースと整合的な結果である。

		複雑性の指標		組織の能力の指標		不確実性	市場条件		
製品名	設計組織	アーキテクチャ	構造要素数	試行ｽﾋﾟｰﾄﾞ	最適化プロセス	科学技術ﾚﾍﾞﾙ	機能要求ﾚﾍﾞﾙ	勝敗	ﾘｰﾄﾞﾀｲﾑ比
スーパーコンピュータ	浅慮なウサギ	モジュラー	多い	速い	コンペティション	低い	厳しい	勝	1
	周到なカメ	モジュラー	多い	遅い	コンペティション	高い	厳しい	負	3

以上が、本稿におけるシミュレーションの結果から得られる、暫定的なインプリケーションである。むろん、以上はあくまで暫定的な観察結果であるし、また、後追い的でアドホックな結果解釈にも留意を要する。シミュレーションは、パラメータを調整していけばどんな結果も出せるのではないか、という批判にも注意を払う必要がある。

しかしながら、今回のシミュレーションには、一定の収穫もあったと考える。第 1 に、現実の設計プロセスが持つ本質的な特徴を再現した幾つかのシミュレーション・モデルを作ることができたと考える。第 2 に、このモデルを用いたシミュレーション結果は、概ね、現実のケースで常識として言われている幾つかの傾向と整合的であった。第 3 に、上記の**「浅慮なウサギ」**（脚の速い愚者）タイプを序論で述べた日本企業と重ね合わせて考えれば、概ねケーススタディによる先行研究と整合的な結果がえられている。すなわち、序論における「定型化された事実」①～④と上のインプリケーションを比較してみると、以下の点で、シミュレーション結果と「定型的な事実」の間に、類似点が見られるのである。

① **「浅慮なウサギ」**は密なコミュニケーションを必要とするコーディネーション開発型で「周到なカメ」に対する競争優位を発揮している。緊密なコミュニケーションに基づくコーディネーション（連携調整）は、藤本（2005）の言う「統合型ものづくり」の必須

条件である。試行スピードが速いウサギと日本企業が対応するのは、ケーススタディによる観測事実とも一致している。

② **「浅慮なウサギ」**が得意なコーディネーション開発型は、厳しい市場の条件を満たすことが可能であり、日本という品質にうるさい消費者の市場で日本企業が強い理由を十分に説明している。

③ 科学知識が形成途上にあるような先端科学知識集約商品の場合は、仮に製品が複雑でインテグラルなケース（一見自動車に近い）でも、**「浅慮なウサギ」**は優位性を失う。このパラドックスは、今回のシミュレーションの結果にも度々表れている。

④ 複雑度が中程度、例えばアーキテクチャはインテグラル型だが行列 **A** の要素数が小さいようなケースでは、本シミュレーションにおいても、概して**「浅慮なウサギ」**が有利である。

このように、今回試行したシミュレーション・モデルは、およそシミュレーション・モデルが持つ弱点を全て背負っているとはいえ、それなりにかなりのリアリティを持つように仕上がったと筆者は考える。第 1 に、モデルの基礎となった「公理系設計」と「2 段階設計モデル」は、極めてシンプルなモデルとはいえ、現実の設計活動がもつ本質的な部分を描写できていると考える。第 2 に、シミュレーションの結果示された、設計リードタイムとモデルに含まれる諸変数の関係の中に、少なくとも常識に反するものは見つかっていない。

そして第 3 に、本稿の冒頭で述べた「定型的事実」を「2 段階設計プロセス」を基礎とする今回のシミュレーションで再現することが、本稿の第一義的な目的であったが、その目的を達成したとは言い切れないものの、達成へ向けた一つのめどが立ったといえよう。今後は、さらにモデルを精密化しつつ、設計活動と競争優位をシミュレーション・モデルで再現することの可能性を、さらに探っていきたい。

参考文献

[1] Baldwin,C.Y. & K.B.Clark (2000) *Design Rules: The Power of Modularity*, MIT Press, Cambridge MA.

[2] Clark, K.B.and Fujimoto, T. (1991), *Product Development Performance: Strategy, Organization, and Management in the World Auto Industry*, Boston: Harvard

[3] Fujimoto, Takahiro (1999) *The Evolution of a Manufacturing System at Toyota*, Oxford University Press.

[4] 藤本隆宏（2001）『生産マネジメント入門（Ⅰ）（Ⅱ）』日本経済新聞社

[5] 藤本隆宏（2003）『能力構築競争　日本の自動車産業はなぜ強いのか』中央公論新社

[6] 藤本隆宏（2004）『日本のものづくり哲学』日本経済新聞社

[7] 藤本隆宏（2005）「アーキテクチャの比較優位に関する一考察」『赤門マネジメント・レビュー』4 巻 11 号 pp.523-548 ; 東京大学大学院経済研究科，MMRC Discussion Paper No.24.

[8] 藤本隆宏・大鹿隆（2005）『現場発の産業立地戦略－「擦り合わせ型」を国内に』日経新聞（朝刊）経済教室 2005 年 8 月 11 日版

[9]藤本隆宏・安本雅典（2000）『成功する製品開発―産業間比較の視点』有斐閣

[10] Galbraith,J.R.(1973) Deasigning Complex Organizations. Reading, MA: Addison-Wesley.

[11] 平井敏彦他著・小早川隆治編（2003）『マツダ／ユーノスロードスター―日本製ライトウェイトスポーツカーの開発物語』三樹書房

[12] Iansiti, M.（1998）*Technology Integration*, Harvard Business School Press, Boston.

[13] 楠木建（2001）「機能分化 : 製品コンセプトのイノベーションを組織化する」『組織科学』Vol.35, No.2, 16-37.

[14] 桑嶋健一（1999）「医薬品の研究開発プロセスにおける組織能力」『組織科学』Vol.33, No.2, pp. 88-104.

[15] 中尾政之（2003）『創造設計学』丸善

[16] 中尾政之・畑村洋太郎・服部和隆（1999）『設計のナレッジマネジメント』日刊工業新聞社

[17] 延岡健太郎 (1997)「部品サプライヤーの顧客ネットワーク戦略－顧客範囲の経済性」藤本隆宏・西口. 敏広・伊藤秀史『サプライヤー・システム』有斐閣

[18] Suh,N.P.,(1990) *The Principles of Design*, Oxford University Press, New York(畑村洋太郎監訳[1992]『設計の原理―創造的機械設計論―』朝倉書店)

[19] Suh,N.P.,(2001) *Axiomatic Design - Advances and Applications*, Oxford University Press, New York(中尾政之・飯野謙次・畑村洋太郎共訳[2004]『公理的設計』森北出版)

[20] 中馬宏之（2004）「日本のサイエンス型産業が直面する複雑性と組織限界：半導体露光装置の事例から」『一橋ビジネスレビュー』52 巻 3 号　東洋経済新報社　64-85

[21] Ulrich, Karl T.（1995）"Product Architecture in the manufacturing Firm, " *Research Policy*, 24, pp.419-440.

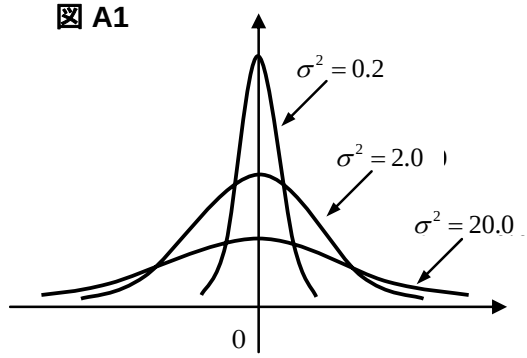
補論

A.1 シミュレーションの実行条件

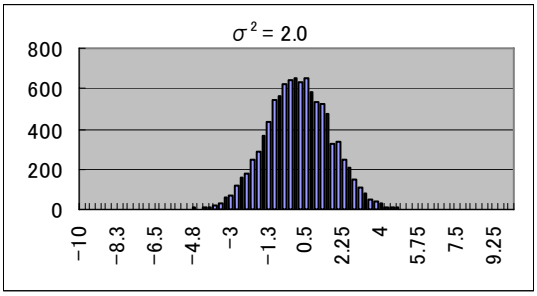
ここでは、本シミュレーションにおける外生パラメータの実装値や実行条件について解説する。

まず、シミュレーションプログラムでは(1)式の**A**と**FR***に－10 から 10 までの実数を取りうる一様乱数がセットされた。**A**については、事前の因果知識レベルを表す指標としての**A**の不完全性を実装するため、－10 から 10 までの実数を取りうる、平均 0、分散σ^2の正規乱数が各要素に加えられた。正規乱数が加えられた**A**を**A'**、元の**A**を**A***とすると、正規分布は分散σ^2が大きくなるほど分布のばらつきが大きくなるので（**図A1**）、**A'**と**A***の乖離はσ^2が大きくなるほど広がる傾向を持つ。よって、**A**の不完全性の度合いはσ^2によってコントロールされる。[14] 本シミュレーションでは、σ^2の実装値として 0.2，2.0，20.0 を試みた。

図 A1



以下に、本シミュレーションで使用した正規乱数生成アルゴリズムを用い、$\sigma^2 = 2.0$の正規乱数を 10000 回生成した結果のヒストグラムを示す。



次に、複雑性(製品特性)の指標としての設計アーキテクチャには、インテグラル型とモジ

[14] 当然のことながら、「2 段階の設計モデル」のPhase1 で用いられるのは**A'**であり、Phase2 で用いられるのは**A***である。

ュラー型のいずれかが与えられた。実装型は**A**に関する(3)の行列である。ただし、モジュラー型については、上述の正規乱数が加えられたのは対角要素のみである。同じく複雑性の指標としての**A**の要素数には、2×2 = 4, 3×3 = 9, …, 10×10 = 100 のいずれかが試みられた。組織能力の指標としての最適化プロセスについては、コーディネーション開発型とコンペティション開発型のいずれかが、3 節のアルゴリズムに忠実な形で実装された。試行スピード（組織能力の指標）については、最適化プロセス（コーディネーション開発型もしくはコンペティション開発型）を遅延なく順次実行するスピードを基準スピードsとし、それに対して、プログラムの 1STEPを実行したら次の 2STEPでは何もせず（待ち状態）、その後のSTEPから処理を再開する場合をスピード(1/3)sと定義して、sと(1/3)sのいずれかを実装して実行した。ここで言うSTEPとは、4 節のコーディネーション開発型の処理説明における(1.1), (1.2), …, (t,2)のそれぞれに対応する。一方、コンペティション開発型では(1)と(2)、(3)と(4)の組み合わせに対応する。最後に、市場条件の指標としての市場許容品質をdとすると、厳しい市場条件としてd=1.0 が、緩い市場条件としてd=3.0 がそれぞれ採用された。そして、各STEPにおいて観察される**FR**と要求機能**FR***のユークリッド距離を測定し、それが市場許容品質を下回った時点で、（厳しい、あるいは緩い）要求機能は満たされたものと判定された。

A.2　シミュレーションの実行と結果

以上のようなパラメータに関する全ての組み合わせが、本シミュレーションでは実行された。そして、コーディネーション開発型では最適解に向けて収束していくパターンと発散するパターンの両方が考えられるので[15]、各STEPにおいて観察される**FR**と要求機能**FR***のユークリッド距離が 0.5 を下回るか 10000 を超えた時点で、これを 1SETとして処理を中断し、再び**A**と**FR***に新たな乱数を投入して次のSETを行うという処理を、パラメータの各組み合わせにつき 1000SETずつ実行した。コンペティション開発型では、**DP**の要素$DP_1, DP_2, \cdots, DP_m$について、それぞれ[－1000, 1000]の区間から分割を開始し、15STEPを 1SETとして、パタメータの各組み合わせにつき 1000SETずつ実行した。等しく$1SET = 15STEP$という回数を設定したのは、例えば、**図 3** のような 2×2 分割のケースの場合、**DP**の解を探索するのは 0 を中心点とした一辺約 2000 の正方形[16] の内部であるが、その正方形を 15 回分割して出来る部分空間は 1 辺約 0.06 の正方形である。もしその中に最適解**DP***を表す点が含まれているならば、15 回分割するまでに市場許容品質$d = 1.0$に達しているはずであり、もし含まれていないならば、それ以上分割を繰り返しても最適解に漸近することは考えられない。よって

[15] ミクロ経済学における「クモの巣理論」と同じメカニズムによる。
[16] **DP**の解は 0 の近傍を最頻値として、ほぼ－1000 から＋1000 の間に分布している。

15STEP＝15分割で十分と判断した。

以上のような外生パラメータと実行条件の下でシミュレーションを 1000SET 実行した結果を集計したものを表 A1 に示す。ただし、以下に示されるのは試行スピードが s のケースのみであり、(1/3)s のケースについては後述する。

表 A1

＊表中の用語の説明：

適格回数（了）：1000SETのうち**FR**と要求機能**FR***のユークリッド距離（以下Edと表記）が 0.5 を下回ったSET数。0.5 を下回るとシミュレーションの 1SETは終了する

最大適格 STEP 数（緩）：Ed が 3.0 を下回ったときの STEP 数の最大値

最小適格 STEP 数（緩）：Ed が 3.0 を下回ったときの STEP 数の最小値

平均適格 STEP 数（緩）：Ed が 3.0 を下回ったときの STEP 数の平均値，その下の（ ）付の値は Ed が 3.0 を下回ったときの STEP 数の標準偏差

適格回数（緩）：1000SET のうち Ed が 3.0 を下回った SET 数

最大適格 STEP 数（厳）：Ed が 1.0 を下回ったときの STEP 数の最大値

最小適格 STEP 数（厳）：Ed が 1.0 を下回ったときの STEP 数の最小値

平均適格 STEP 数（厳）：Ed が 1.0 を下回ったときの STEP 数の平均値，その下の（ ）付の値は Ed が 1.0 を下回ったときの STEP 数の標準偏差

適格回数（厳）：1000SET のうち Ed が 1.0 を下回った SET 数

最大事後距離：（コンペティション型における）15 分割後の Ed の最大値

最小事後距離：（コンペティション型における）15 分割後の Ed の最小値

平均事後距離：（コンペティション型における）15 分割後の Ed の平均値，その下の（ ）付の値は 15 分割後の Ed の標準偏差

インテグラル - コーディネーション

アーキテクチャ	インテグラル	インテグラル	インテグラル	インテグラル	インテグラル
行列の要素数	2×2	2×2	2×2	3×3	3×3
最適化のプロセス	コーディネーション	コーディネーション	コーディネーション	コーディネーション	コーディネーション
分散（A の不完全性）	0.2	2.0	20.0	0.2	2.0
適格回数（了）（1000 回中）	639	534	546	343	247
最大適格 STEP 数（緩）	482	122	247	147	2611
最小適格 STEP 数（緩）	1	1	1	1	1
平均適格 STEP 数（緩）	2.23	2.73	4.62	2.70	10.56
（標準偏差）	(16.89)	(7.26)	(14.70)	(10.68)	(120.94)
適格回数（緩）（1000 回中）	858	702	642	721	470
最大適格 STEP 数（厳）	2446	222	421	492	5362
最小適格 STEP 数（厳）	1	1	1	1	1
平均適格 STEP 数（厳）	6.30	6.04	9.80	7.97	33.71
（標準偏差）	(91.59)	(16.42)	(29.66)	(32.01)	(320.06)
適格回数（厳）（1000 回中）	717	577	567	477	281

アーキテクチャ	インテグラル	インテグラル	インテグラル	インテグラル	インテグラル
行列の要素数	3×3	4×4	4×4	4×4	5×5
最適化のプロセス	コーディネーション	コーディネーション	コーディネーション	コーディネーション	コーディネーション
分散（A の不完全性）	20.0	0.2	2.0	20.0	0.2
適格回数（了）（1000 回中）	210	116	59	52	32
最大適格 STEP 数（緩）	316	196	154	622	146
最小適格 STEP 数（緩）	1	1	1	1	1
平均適格 STEP 数（緩）	10.99	2.29	7.14	29.48	2.78
（標準偏差）	(27.19)	(11.26)	(18.17)	(85.23)	(13.13)
適格回数（緩）（1000 回中）	299	519	219	91	411
最大適格 STEP 数（厳）	676	340	402	1305	231
最小適格 STEP 数（厳）	1	1	1	1	1
平均適格 STEP 数（厳）	30.36	9.36	31.38	98.17	14.12
（標準偏差）	(76.42)	(33.66)	(53.25)	(227.40)	(42.62)
適格回数（厳）（1000 回中）	222	223	79	54	92

アーキテクチャ	インテグラル	インテグラル	インテグラル	インテグラル	インテグラル
行列の要素数	5×5	5×5	6×6	6×6	6×6
最適化のプロセス	コーディネーション	コーディネーション	コーディネーション	コーディネーション	コーディネーション
分散（A の不完全性）	2.0	20.0	0.2	2.0	20.0
適格回数（了）（1000 回中）	4	10	5	1	0
最大適格 STEP 数（緩）	202	960	28	48	1
最小適格 STEP 数（緩）	1	1	1	1	1
平均適格 STEP 数（緩）	4.77	107.11	1.18	2.22	1.00
（標準偏差）	(22.05)	(227.03)	(1.60)	(6.99)	(0.00)
適格回数（緩）（1000 回中）	90	18	319	45	2
最大適格 STEP 数（厳）	612	1330	5	96	—
最小適格 STEP 数（厳）	1	1	1	1	—
平均適格 STEP 数（厳）	100.00	263.10	1.37	34.00	—
（標準偏差）	(211.01)	(387.45)	(0.95)	(53.73)	—
適格回数（厳）（1000 回中）	8	10	52	3	0

アーキテクチャ	インテグラル	インテグラル	インテグラル	インテグラル	インテグラル
行列の要素数	7×7	7×7	7×7	8×8	8×8
最適化のプロセス	コーディネーション	コーディネーション	コーディネーション	コーディネーション	コーディネーション
分散（A の不完全性）	0.2	2.0	20.0	0.2	2.0
適格回数（了）（1000 回中）	1	0	0	0	0
最大適格 STEP 数（緩）	3	4	—	5	2
最小適格 STEP 数（緩）	1	1	—	1	1
平均適格 STEP 数（緩）	1.10	1.48	—	1.14	1.13
（標準偏差）	(0.37)	(0.85)	—	(0.56)	(0.35)
適格回数（緩）（1000 回中）	219	23	0	188	8
最大適格 STEP 数（厳）	4	—	—	1	—
最小適格 STEP 数（厳）	1	—	—	1	—
平均適格 STEP 数（厳）	1.50	—	—	1.00	—
（標準偏差）	(0.89)	—	—	(0.00)	—
適格回数（厳）（1000 回中）	16	0	0	5	0

アーキテクチャ	インテグラル	インテグラル	インテグラル	インテグラル	インテグラル
行列の要素数	8×8	9×9	9×9	9×9	10×10
最適化のプロセス	コーディネーション	コーディネーション	コーディネーション	コーディネーション	コーディネーション
分散（A の不完全性）	20.0	0.2	2.0	20.0	0.2
適格回数（了）（1000 回中）	0	0	0	0	0
最大適格 STEP 数（緩）	—	5	2	—	7
最小適格 STEP 数（緩）	—	1	2	—	1
平均適格 STEP 数（緩）	—	1.10	2.00	—	1.12
（標準偏差）	—	(0.50)	(0.00)	—	(0.63)
適格回数（緩）（1000 回中）	0	145	1	0	117
最大適格 STEP 数（厳）	—	4	—	—	—
最小適格 STEP 数（厳）	—	1	—	—	—
平均適格 STEP 数（厳）	—	1.60	—	—	—
（標準偏差）	—	(1.34)	—	—	—
適格回数（厳）（1000 回中）	0	5	0	0	0

アーキテクチャ	インテグラル	インテグラル
行列の要素数	10×10	10×10
最適化のプロセス	コーディネーション	コーディネーション
分散（A の不完全性）	2.0	20.0
適格回数（了）（1000 回中）	0	0
最大適格 STEP 数（緩）	—	—
最小適格 STEP 数（緩）	—	—
平均適格 STEP 数（緩）	—	—
（標準偏差）	—	—
適格回数（緩）（1000 回中）	0	0
最大適格 STEP 数（厳）	—	—
最小適格 STEP 数（厳）	—	—
平均適格 STEP 数（厳）	—	—
（標準偏差）	—	—
適格回数（厳）（1000 回中）	0	0

モジュラー - コーディネーション

アーキテクチャ	モジュラー	モジュラー	モジュラー	モジュラー	モジュラー
行列の要素数	2×2	2×2	2×2	3×3	3×3
最適化のプロセス	コーディネーション	コーディネーション	コーディネーション	コーディネーション	コーディネーション
分散（A の不完全性）	0.2	2.0	20.0	0.2	2.0
適格回数（了）（1000 回中）	1000	1000	1000	1000	1000
最大適格 STEP 数（緩）	2	2	2	3	3
最小適格 STEP 数（緩）	1	1	1	1	1
平均適格 STEP 数（緩）	1.07	1.20	1.42	1.18	1.61
（標準偏差）	(0.25)	(0.40)	(0.49)	(0.52)	(0.82)
適格回数（緩）（1000 回中）	1000	1000	1000	1000	1000
最大適格 STEP 数（厳）	2	2	2	3	3
最小適格 STEP 数（厳）	1	1	1	1	1
平均適格 STEP 数（厳）	1.18	1.49	1.70	1.52	2.21
（標準偏差）	(0.39)	(0.50)	(0.46)	(0.78)	(0.84)
適格回数（厳）（1000 回中）	1000	1000	1000	1000	1000

アーキテクチャ	モジュラー	モジュラー	モジュラー	モジュラー	モジュラー
行列の要素数	3×3	4×4	4×4	4×4	5×5
最適化のプロセス	コーディネーション	コーディネーション	コーディネーション	コーディネーション	コーディネーション
分散（A の不完全性）	20.0	0.2	2.0	20.0	0.2
適格回数（了）（1000 回中）	1000	1000	1000	1000	1000
最大適格 STEP 数（緩）	3	4	4	4	5
最小適格 STEP 数（緩）	1	1	1	1	1
平均適格 STEP 数（緩）	2.13	1.40	2.12	2.94	1.64
（標準偏差）	(0.84)	(0.90)	(1.18)	(1.10)	(1.24)
適格回数（緩）（1000 回中）	1000	1000	1000	1000	1000
最大適格 STEP 数（厳）	3	4	4	4	5
最小適格 STEP 数（厳）	1	1	1	1	1
平均適格 STEP 数（厳）	2.66	2.03	3.11	3.59	2.74
（標準偏差）	(0.60)	(1.19)	(1.02)	(0.70)	(1.55)
適格回数（厳）（1000 回中）	1000	1000	1000	1000	1000

アーキテクチャ	モジュラー	モジュラー	モジュラー	モジュラー	モジュラー
行列の要素数	5×5	5×5	6×6	6×6	6×6
最適化のプロセス	コーディネーション	コーディネーション	コーディネーション	コーディネーション	コーディネーション
分散（A の不完全性）	2.0	20.0	0.2	2.0	20.0
適格回数（了）（1000 回中）	1000	1000	1000	1000	1000
最大適格 STEP 数（緩）	5	5	6	6	6
最小適格 STEP 数（緩）	1	1	1	1	1
平均適格 STEP 数（緩）	2.72	3.87	1.94	3.57	4.87
（標準偏差）	(1.56)	(1.18)	(1.62)	(1.85)	(1.28)
適格回数（緩）（1000 回中）	1000	1000	1000	1000	1000
最大適格 STEP 数（厳）	5	5	6	6	6
最小適格 STEP 数（厳）	1	1	1	1	1
平均適格 STEP 数（厳）	4.06	4.62	3.40	5.06	5.63
（標準偏差）	(1.14)	(0.68)	(1.89)	(1.17)	(0.70)
適格回数（厳）（1000 回中）	1000	1000	1000	1000	1000

アーキテクチャ	モジュラー	モジュラー	モジュラー	モジュラー	モジュラー
行列の要素数	7×7	7×7	7×7	8×8	8×8
最適化のプロセス	コーディネーション	コーディネーション	コーディネーション	コーディネーション	コーディネーション
分散（A の不完全性）	0.2	2.0	20.0	0.2	2.0
適格回数（了）（1000 回中）	1000	1000	999	1000	1000
最大適格 STEP 数（緩）	7	7	7	8	8
最小適格 STEP 数（緩）	1	1	1	1	1
平均適格 STEP 数（緩）	2.23	4.33	5.91	2.76	5.23
（標準偏差）	(1.98)	(2.13)	(1.30)	(2.40)	(2.33)
適格回数（緩）（1000 回中）	1000	1000	1000	1000	1000
最大適格 STEP 数（厳）	7	7	7	8	8
最小適格 STEP 数（厳）	1	1	2	1	1
平均適格 STEP 数（厳）	4.16	6.01	6.65	5.15	7.05
（標準偏差）	(2.18)	(1.27)	(0.65)	(2.38)	(1.20)
適格回数（厳）（1000 回中）	1000	1000	1000	1000	1000

アーキテクチャ	モジュラー	モジュラー	モジュラー	モジュラー	モジュラー
行列の要素数	8×8	9×9	9×9	9×9	10×10
最適化のプロセス	コーディネーション	コーディネーション	コーディネーション	コーディネーション	コーディネーション
分散（A の不完全性）	20.0	0.2	2.0	20.0	0.2
適格回数（了）（1000 回中）	999	1000	1000	999	1000
最大適格 STEP 数（緩）	8	9	9	9	10
最小適格 STEP 数（緩）	1	1	1	1	1
平均適格 STEP 数（緩）	6.88	3.21	6.07	7.92	3.61
（標準偏差）	(1.36)	(2.79)	(2.53)	(1.34)	(3.22)
適格回数（緩）（1000 回中）	999	1000	1000	999	1000
最大適格 STEP 数（厳）	8	9	9	9	10
最小適格 STEP 数（厳）	3	1	1	4	1
平均適格 STEP 数（厳）	7.62	5.98	7.99	8.64	6.87
（標準偏差）	(0.71)	(2.56)	(1.28)	(0.66)	(2.67)
適格回数（厳）（1000 回中）	999	1000	1000	999	1000

アーキテクチャ	モジュラー	モジュラー
行列の要素数	10×10	10×10
最適化のプロセス	コーディネーション	コーディネーション
分散（A の不完全性）	2.0	20.0
適格回数（了）（1000 回中）	1000	1000
最大適格 STEP 数（緩）	10	10
最小適格 STEP 数（緩）	1	2
平均適格 STEP 数（緩）	7.22	8.89
（標準偏差）	(2.51)	(1.35)
適格回数（緩）（1000 回中）	1000	1000
最大適格 STEP 数（厳）	10	10
最小適格 STEP 数（厳）	2	6
平均適格 STEP 数（厳）	9.03	9.63
（標準偏差）	(1.22)	(0.67)
適格回数（厳）（1000 回中）	1000	1000

インテグラル - コンペティション

アーキテクチャ	インテグラル	インテグラル	インテグラル	インテグラル	インテグラル
行列の要素数	2×2	2×2	2×2	3×3	3×3
最適化のプロセス	コンペティション	コンペティション	コンペティション	コンペティション	コンペティション
分散（A の不完全性）	0.2	2.0	20.0	0.2	2.0
最大事後距離（15 分割後）	1522	2004	2270	2910	2730.46
最小事後距離（15 分割後）	0.0	0.0	0.0	0.0	0.01
平均事後距離（15 分割後）	73.59	14039	167.80	185.51	201.80
（標準偏差）	(232.82)	(343.43)	410.74	458.46	497.39
最大適格 STEP 数（緩）	4	8	16	32	64.00
最小適格 STEP 数（緩）	1.0	1.0	1.0	1.0	1.00
平均適格 STEP 数（緩）	2.51	4.26	6.88	13.09	22.68
（標準偏差）	(1.12)	(2.14)	4.19	9.07	17.74
適格回数（緩）（1000 回中）	680	472	328	237	210
最大適格 STEP 数（厳）	4	8	16	32	64.00
最小適格 STEP 数（厳）	1.0	1.0	1.0	2.0	2.00
平均適格 STEP 数（厳）	2.58	4.46	7.68	13.01	22.05
（標準偏差）	(1.12)	(2.10)	4.55	9.30	18.71
適格回数（厳）（1000 回中）	475	267	113	68	41

アーキテクチャ	インテグラル	インテグラル	インテグラル	インテグラル
行列の要素数	7×7	8×8	9×9	10×10
最適化のプロセス	コンペティション	コンペティション	コンペティション	コンペティション
分散（A の不完全性）	20	20	0.2	20
最大事後距離（15 分割後）	4091.53	3410.27	3918.75	3914.38
最小事後距離（15 分割後）	0.08	0.10	0.10	0.25
平均事後距離（15 分割後）	209.30	194.89	204.52	213.22
（標準偏差）	562.46	560.94	579.26	629.29
最大適格 STEP 数（緩）	128.00	252.00	500.00	937.00
最小適格 STEP 数（緩）	2.00	2.00	2.00	4.00
平均適格 STEP 数（緩）	40.50	79.40	114.24	311.55
（標準偏差）	35.66	71.78	127.32	276.11
適格回数（緩）（1000 回中）	145	105	93	64
最大適格 STEP 数（厳）	128.00	204.00	452.00	963.00
最小適格 STEP 数（厳）	2.00	46.00	2.00	19.00
平均適格 STEP 数（厳）	47.40	130.14	144.36	392.10
（標準偏差）	38.06	58.22	148.93	374.94
適格回数（厳）（1000 回中）	25	7	14	10

モジュラー - コンペティション

アーキテクチャ	モジュラー	モジュラー	モジュラー	モジュラー	モジュラー
行列の要素数	2×2	2×2	2×2	3×3	3×3
最適化のプロセス	コンペティション	コンペティション	コンペティション	コンペティション	コンペティション
分散（A の不完全性）	0.2	0.2	0.2	0.2	2.0
最大事後距離（15 分割後）	0	0	0	6	6.71
最小事後距離（15 分割後）	0.0	0.0	0.0	0.0	0.00
平均事後距離（15 分割後）	0.00	0.00	0.00	0.02	0.01
（標準偏差）	(0.00)	(0.00)	0.00	0.27	0.18
最大適格 STEP 数（緩）	4	8	16	32	64.00
最小適格 STEP 数（緩）	1.0	1.0	1.0	1.0	1.00
平均適格 STEP 数（緩）	2.58	4.25	5.88	12.21	18.28
（標準偏差）	(1.04)	(2.20)	4.58	8.70	16.97
適格回数（緩）（1000 回中）	1000	1000	1000	999	999
最大適格 STEP 数（厳）	4	8	16	32	64.00
最小適格 STEP 数（厳）	1.0	1.0	1.0	1.0	1.00
平均適格 STEP 数（厳）	2.56	4.38	5.25	11.60	21.00
（標準偏差）	(1.06)	(2.07)	5.31	8.46	17.43
適格回数（厳）（1000 回中）	1000	1000	1000	997	997

アーキテクチャ	モジュラー	モジュラー	モジュラー	モジュラー
行列の要素数	7×7	8×8	9×9	10×10
最適化のプロセス	コンペティション	コンペティション	コンペティション	コンペティション
分散（A の不完全性）	20.0	2.0	20.0	2.0
最大事後距離（15 分割後）	6.47	6.01	4.31	7.27
最小事後距離（15 分割後）	0.00	0.00	0.00	0.00
平均事後距離（15 分割後）	0.02	0.01	0.00	0.03
（標準偏差）	0.26	0.18	0.08	0.36
最大適格 STEP 数（緩）	128.00	256.00	512.00	1017.00
最小適格 STEP 数（緩）	1.00	1.00	2.00	2.00
平均適格 STEP 数（緩）	37.42	66.36	133.67	276.65
（標準偏差）	33.96	66.18	131.47	277.16
適格回数（緩）（1000 回中）	998	998	1000	999
最大適格 STEP 数（厳）	127.00	255.00	507.00	1020.00
最小適格 STEP 数（厳）	2.00	2.00	2.00	1.00
平均適格 STEP 数（厳）	35.61	62.72	129.18	238.18
（標準偏差）	33.53	63.70	129.57	260.04
適格回数（厳）（1000 回中）	995	997	999	998

試行スピードが(1/3)s のケースについては、最大適格 STEP 数(緩)、最小適格 STEP 数(緩)、平均適格 STEP 数(緩)、最大適格 STEP 数(厳)、最小適格 STEP 数(厳)、平均適格 STEP 数(厳)が、スピードが s のケースのほぼ正確に 3 倍となる。ただし、適格回数(了)、適格回数(緩)、適格回数(厳)、最大事後距離、最小事後距離、平均事後距離についてはほとんど変わらない。これは、s のケースに対し(1/3)s のケースは単に待ち状態が 2STEP 分入るだけなので、自明の結果ともいえる。よって紙面の都合もあり割愛した。

コンペティション開発型の実行結果については、分散 σ^2 を変化させて実行しても結果に

差が見られなかった。コンペティション開発型の場合、σ^2によってコントロールされた **A'** を最適化プロセスの中で用いないのでこれも自明であるが、コーディネーション開発型との比較のために念のため分散が異なるケースも実行した。しかし、結果は予想通りであったのでここでは割愛した。また **A** の要素数についても、**A** が m×m 行列のとき、**最大適格 STEP 数（緩）**と**最大適格 STEP 数（厳）**が 2^mSTEP になるという規則性があり、これは要素数にかかわらず成り立つ。また、インテグラル型のケースにおいて最適解が最初から[－1000, 1000]にない場合は収束せず、**最大事後距離**が大きな数になるという規則も要素数にかかわらず見られた。