チーム設計の数理的考察 A mathematical study on team design work

○平井俊輔（電通国際情報サービス）

大野嘉子（電通国際情報サービス）

Shunsuke HIRAI

Yoshiko Ohno

Manufacturing Industry Business Operations, Information Services International Dentsu, LTD
2-17-1 Kounan, Minato-ku, Tokyo, 108-0075, Japan

This paper discusses the pattern model of communication on team design work aided by the framework of three dimensional CAD. In design process, designer revises the idea many times to content function. The work in progress idea can be shared other designers. Especially, it is more important to refer the work in progress for integral architecture. Because integral architecture needs adjustment of the design parameter of the components. We report the difference of design process for integral architecture and modular architecture. And, we explain behavior models to adjust rapidly the parameters within integral architecture. Designers can behave as these model autonomously, it emerge organization ability adapted to productive development.

Key Words: design theory, function decomposition, concurrent engineering, product architecture, suriawase, three dimensional CAD,

1 緒言

近年、製品がますます複雑化、大規模化する一方、開発期間は一層短縮される傾向にある。そのため、製品戦略に応じた開発組織をデザインすると共に、開発チーム全体で設計効率を上げる取り組みが重要である⁽¹⁾⁽²⁾。また、チームワークの良否が製品開発プロジェクトの成功に影響を及ぼすことも分かっている⁽³⁾。しかし、設計は概念を創造するプロセスであり、個々の設計者がチーム全体の設計プロセスの進捗や全体像を正しく把握し、各自の製品設計に応用することは非常に困難である⁽⁴⁾。むしろ、「情報共有」「緊密な連携調整」「チームワーク」といったスローガンが先行し、具体的な行動パターンが説明されることは少ない。

一方で昨今、製品のアーキテクチャに着目した研究が盛んになっているが⁽⁵⁾⁽⁶⁾⁽⁷⁾、それらは主に経営の立場から、製品アーキテクチャや組織構造、企業間取引等の因果関係によってもたらされる企業のパフォーマンスについて、マクロ的に分析されているものが多い。アーキテクチャや組織構成はある製品の開発に着目すれば企業のマネジメントレベルで設定されるスタティックな特性であり、内部で設計を行う人間のオペレーションはブラックボックスとして扱われている。

このような問題意識の下で、本発表では実際の設計の局面として議論できるミクロな視点から、チームで設計する際に、設計者間で交換される情報やタイミングについて、製品アーキテクチャごとに考察を行うものとする。また、設計のツールとして用いられている三次元CADやPDMのデータモデルをフレームワークとして活用することにする。理由としては、現時点で多くの企業に普及しており、統一的な指標で情報収集、あるいは逆にフィードバックが行いやすいこと、それらITツールが元来持っているシステム性が数理的にモデル化する際に相性がよいことがあるからである。

また、チーム設計はコンカレントエンジニアリングの一部とも言えるが、いわゆるコンカレントエンジニアリングが、企画、設計、製造などの業務プロセスを時間軸で重複させる⁽⁸⁾⁽⁹⁾のに対し、本発表では製品の構成要素に対する設計タスクを並行で行うことを意図している。そのため、分析の範囲を、製品の要求仕様が決まってから設計案が実体化するまでとし、生産準備プロセス等は含まないものとする。

2 チーム設計をモデル化するための基本概念

本発表では、チーム設計を「複数の人が、ある人工物のシステムを同時並行に設計する行為」と定義し、まずは設計プロセス、製品構成、チームについて整理する。

2.1 設計の仕掛け状態の表現

各種の設計論によれば、設計とは機能と人工物の構造の写像関係の操作と概ね解釈することができる⁽¹⁰⁾⁽¹¹⁾。ここに、ある実体構造を表す設計パラメータの集合 X があるとき、それが持つ機能の集合 F 、機能集合から実体集合への写像を f とすると、以下のような数式で表現できる。

$$F=f(X) \quad (1)$$

人工物が満足すべき機能を FR とすると、それを包含する F の逆像たる X を求めればよい。

$$FR \subseteq f(X) \quad (1)$$

しかし、現実的な知識の上では、 FR 内の要素が多くなり、 F と X の関係が複雑であるほど、一段階のプロセスでは FR を満たす X は求まらず、既知の構造やシンプルな初期案の設計パラメータから機能 F を評価し、その結果を考慮しつつ反復して変更を加えることで、所望の FR に漸近させる方法が取られる。設計開始時点から数えたときの i 番目の設計パラメータとその機能の集合を X_i 、 F_i とし、最終的に FR を満たしたときの i を n とした時の過程を Fig.1 に図示する。この図では、仕掛け状態の X_i を作成して F_i を評価し、 FR と F_i の積が大きくなるような方向性を念頭に置きながら、 $X(i+1)$ を作成する作業を繰り返す経過を表している。

Fig. 1 Progressive design process toward required function

実際には設計パラメータの取り方によって、Fig1 の X_3 のように前の案より改悪となる可能性がありうる。また、 X_i が生成されるタイミングは計画的、規則的ではなく、設計の難易度、設計者のスキル、労働環境に左右される。この2点が後に述べるチーム設計の管理を難しくしている要因となっている。式(2)の i は CAD や PDM のシステムの世界において、一般的にリビジョンやバージョンという名前で管理されているため、本論でもリビジョンと呼ぶことにする。また、人間が一度に操作できる情報量が限られているとするならば、 n は設計の難易度と考えることもできる。つまり、経験的に、多機能で新規性が高く、使用環境が厳しい部品は n が大きくなる傾向にある。 $n=1$ で済むケースとしては、ボルト・ナットなどの標準部品、過去製品からの流用設計などである。

2.2 システムの階層性と CAD のアセンブリ情報

複雑な製品のシステムは階層的な性質を持っているため、要求機能を下位機能にブレークダウンし、下位機能ごとに設計案を探索し構造を決定することが多い。そのような処理を再帰的に繰り返せば、結果的に機能と構造を階層化することができる⁽¹²⁾。設計の実務では、構造の階層を CAD でいうアセンブリで考えると理解しやすい。つまり、形状情報や材料特性を持った「部品」と、構成要素（どの部品で成り立っているか）およびそれらの相対的な配置関係の情報を保持した「アセンブリ」を用いて、ツリー状で表現するのである。

たとえば、部品 Pa 、 Pb の持つパラメータの集合を Pa 、 Pb とし、その二部品からなるアセンブリ A の持つパラメータの集合を A とすると、そのアセンブリの機能集合 FA は、先の式(1)を用いて以下のように表現できる。

$$FA = fa(A \cup (Pa \cup Pb)) \quad (3)$$

A の保持しているパラメータは、構成要素と配置情報のみであり、それらだけでは機能を発現させることはできない。また逆に、 FA は、 Pa と Pb の機能集合の和とも言えない。たとえば、重心位置や干渉の有無は、配置場所の情報に加えて各部品の形状情報がなければ算出できない属性である。

2.3 製品アーキテクチャと設計プロセス

製品の階層性を説明する枠組みとして、近年アーキテクチャの概念に注目が集まっている⁽¹³⁾⁽¹⁴⁾。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り、インテグラル型とモジュラー型に区別することができる。インテグラル型は、要求機能と構成要素の間の対応関係が絡み合っており、構成要素間の機能的相互依存性が高い製品アーキテクチャを指す。モジュラー型は、構成要素間の機能的独立性が高い製品アーキテクチャであり、構成要素同士を組み合わせた際の調整が不要である (Fig.2)。

この区別の意味を考えると、あるアセンブリがインテグラル型か、モジュラー型になるかは、式(3)のアセンブリの機能を満たそうとしたときに、下位の構成要素の設計パラメータを操作するか否かにあると言える。

インテグラル型の例として引き合いによく出される乗用車で検証してみよう。自動車を設計する際、仮にエンジンの設計パラメータが全て固定で、エンジン自体が発揮する性能が既定だとしても、エンジンの配置場所や搭載方向によって、重心位置やトラクションが変化し、車両の走行性能に影響を及ぼす。また、居住空間の確保、振動騒音対策、耐クラッシュ性等の特性にも関連してくる。これらの特性を車のコンセプトに合うように最適化しようすると、自ずとエンジンの配置場所や搭載方向に応じてエンジン内部の設計パラメータ（気筒数、吸排気方向など）にまで手を入れなければ高いレ

Fig. 2 Product architecture

ベルでのコンパクト化や走行性能向上を実現できない。つまり、下位の構成要素の設計パラメータをそれらの位置情報を含んだ上位のシステムで評価しなければ、下位の設計パラメータを求めることができない。乗用車と対比してモジュラーに近い例として挙げられるトラックを考えると、システム全体から見たエンジンの質量、サイズが相対的に小さいため、トラック全体の特性を評価した場合にエンジンの設計パラメータにまで手を加えなくても良い、つまり上位のシステムの中でエンジンのパラメータを考慮しなくて済むということである。ノート PC とデスクトップ PC のアーキテクチャの違いについても同様のことが言える。

以上のような考え方から、インテグラル型アーキテクチャで、 Pa 、 Pb を部品として持つアセンブリ A の設計手順を紐解いてみると、次のようなステップに分解できる。

- ① アセンブリ A の必要機能 FR から、部分機能 Fa と Fb を切り出す。この時点では、 Fa と Fb は仮で設定した部分機能であり、事後に調整されることが前提である。
- ② Fa と Fb の機能を満たす部品 Pa 、 Pb を個別に設計し、完成したらアセンブリ A を組む。
- ③ アセンブリの機能を評価し、 FR との差分を鑑みて、調整を含んだ機能 Fa' 、 Fb' を再定義する。
- ④ Fa' 、 Fb' を満たす Pa' 、 Pb' を個別に設計し、再度アセンブリを組んで FR を満足するか確認する。

③④がいわゆる「擦り合わせ」と呼ばれる調整作業である。

モジュラー型の場合は、①の時点でインターフェースのルールが Fa と Fb に織り込まれており、アセンブリを組んだ後の摺り合わせが不要になっている。

たとえば、モジュラー形式の一つであるスロット型では⁽¹⁵⁾、母体となるモジュールの上に据え付ける場所が限定されており、かつ接続部分が標準化された形態で、接続部分に部分機能を担う構成要素を導入するだけで全体が機能する。つまり、部品同士を擦り合わせる必要がなく、アセンブリを組む際も、配置関係が固定で構成要素のみを選択すれば良い。パソコンのマザーボードやスポーツ用自転車がこの方式の典型例である。別のモジュール形式である断面型では、モジュール群を特定のインターフェースで相互に連結することによって全体が機能する。こちらは構成要素も配置関係も自由であるが、やはりアセンブリを組んだ後に部品同士を擦り合わせる必要はない。

2.4 階層性を持つシステムの漸進的設計プロセスの表現

次に製品の階層性とリビジョンの関係について説明する。Fig.3 のように、二部品からなるアセンブリを本発表では簡略化して $A(Pa, Pb)$ と表す。さらに、アセンブリと部品にリビジョンの情報を付加した状態を、 $A_i(Pa_j, Pb_k)$ と表現することにする。

Fig. 3 Set of the revision of assembly

たとえば、当初は A.1(Pa.1, Pb.1)だったものが、Pa の設計が進化したことで Pa.2 が作成されると、その時点から、A の構成要素として Pa.2 を組み入れることが可能となる。ここで、開始時点のアセンブリの状態と識別するために、A のリビジョンも一つ上げて A.2(Pa.2, Pb.1)と表現することができる。同様に Pb.2 が生成されれば、A.3(Pa.2, Pb.2)となる。また、子部品の設計が変わらなくても、アセンブリ内で位置関係を変化させた場合は A のリビジョンが上がり A.4(Pa.2, Pb.2)となる。このように各過程での子部品のリビジョンを保持しながらアセンブリを設計することで、アセンブリの構成要素の遷移を厳密に管理することができる。たとえば、Fig.3 の例で言えば、作業後の最新の状態は A.4(Pa.2, Pb.2)であるが、必要に応じて A.2(Pa.2, Pb.1)も取り出せるということである。また、A.5(Pa.1, Pb.2)のように過去の子部品のリビジョンを特定して新たなアセンブリを構成することも出来る。その結果、過去のリビジョンとの比較や、やりなおしが行えるようになる。

2.5 チーム内における設計作業の分担と制約

製品の設計は一人の人間によって行われるわけではなく、複数の人間が関与するのが普通である。その関与の形態として、時系列ごとや構成要素ごとの分担がありえる。

いま、複数の構成要素の設計が複数の人によって、自律的かつ分散的に、非同期で進行している世界を想定する。一つのリビジョンを生成するオペレーションを一単位作業とすると、次の制約が成り立つ。

- 一つの設計作業を担当できるのは一人だけである。Fig.4(a)の一つの矢印に複数の人間を割当てられないということである。設計開始から終了まで入れ替わりで担当したとしても、瞬間的に見れば、一つの部品を設計しているのは、常に排他的に一人である。実際には、複数の人間で相談しながら設計するといったケースもあり得るが、自律的に思考する頭脳体として一人と扱うことにする。
- 一人がある時間に行える作業は一つだけである。Fig.4(a)で、縦に見た場合に同じ人間が並ばないということである。複数の部品やアセンブリを担当していたとしても、ある瞬間に設計できる対象は常に一つである。
- ある作業が終了されたことは、終了した本人はすぐさま認識できるが、本人以外では自発的に情報を確認するか、本人がその相手に伝達しなければ認識できない。

また、一つのシステムを階層化して機能・構造を分解すれば複数の人間で分担して作業することができる。たとえば、Fig.4(b)の A(Pa, Pb)の設計を分担すると考えたとき、最大3人の人間が設計に関与することができる。この図の場合、設計者 Z はアセンブリを担当するため、Pa、Pb のリビジョンの管理と両者の相対的な位置関係の調整のみを行うことになる。また、設計者 X および Y は部品の設計パラメータのみを扱うことができるが、アセンブリ A を参照することで、他の部品との位置関係を確認することは出来るものとする。

Fig. 4 The rule of team design work

3 チーム設計のモデル化

2.5 節で述べた制約に注意しながらチーム設計のプロセスを確認することにする。ただし、ここでは各設計者の問題解決能力は一定と想定し、情報のやり取りのみに着目する。

3.1 モジュラー型とインテグラル型の比較

まずシンプルなアセンブリ A(Pa, Pb)をFig4(b)のように、三人の設計者で同時並行に行う際の設計手順を確認する。モジュラー型の場合は、Pa と Pb は設計者 X、Y によって個別に設計された後に、その成果である部品を A に組み込むだけでよい(Fig.5)。ただし、Pa.1、Pb.1 が完成した時点以降に、設計者 Z に伝達するか、設計者 Z が完成しているかを能動的確認しなければならない。

インテグラル型の場合は、さらにその後に調整作業が必要である(Fig.6 の灰色部分)。モジュラー型同様、Pa、Pb の初期案は別個に作成できるが、それらは生煮え状態であるため作成後に一旦 A.1(Pa.1, Pb.1)を構成し評価した上で、設計者 X、Y に Pa、Pb の新たな機能の指示を出すことになる。設計者 X、Y は調整含みの機能を満足するように、Pa.2、Pb.2 を設計し(実際の場面では片方のみを修正することもある)、それらを設計者 Z が受け取って最終的に必要機能が満足すれば終了である。単純に一度の摺り合わせで収束すると仮定しても、モジュラー型に比較して一往復分のコミュニケーションが余分にかかることになる。

Fig. 5 Team design process of modular assembly

Fig. 6 Team design process of integral assembly

3.2 三部品以上、三階層以上への拡張

一つのアセンブリ内の部品数が増加すると、モジュラー型はアセンブリを集めて組み合わせる手間が部品点数の数に比例するだけであるが、インテグラル型の場合、相互依存関係を考慮しなければならない。あるアセンブリ内の部品間に全て相互依存関係がある場合、その組合せの数は $nC2$ となる。したがって、すべての相互依存関係を満たす解が見つかるためのステップ数は、2部品($2C2=1$)を基準とすると単純に見積もって $nC2$ 倍となる。3部品であれば3倍、4部品であれば6倍の擦り合わせ作業を行う必要があり、それに付随してコミュニケーションの手間がかかることになる。

ただし、部品点数が増えてくれば、アセンブリと部品間にサブアセンブリを作成し、階層性を増やすことである程度の相互依存関係を減少させる戦略が有効である。たとえば、6部品で構成されるアセンブリ A(Pa, Pb, Pc, Pd, Pe, Pf)の相互依存関係数は $6C2$ で15であるが、A(Ax(Pa, Pb, Pc), Ay(Pd, Pe, Pf))と、サブアセンブリごとにまとめることで、関係数を $2C2+3C2 \times 2=7$ にまで減らすことができ、チーム内のコミュニケーションのネットワークを簡素化することになる。しかし、インテグラル型のアーキテクチャの場合は、Aから見た構成要素である Ax, Ay を擦り合わせしようとした時、内部には3部品のインテグラルな構成要素が潜んでいることを忘れてはならない。したがって、システム全体での擦り合わせ回数 S は、全体の部品数を N、サブアセンブリの数を k とすると、一つのサブアセンブリ内の構成要素数は N/k 個になるため、次式で計算できる。

$$\begin{aligned}
 S &= 2 \times N / k C2 \times k C2 \\
 &= 2 \times \frac{1}{2} \times \frac{N}{k} \times \left(\frac{N}{k} - 1 \right) \times \frac{1}{2} \times k \times (k - 1) \\
 &= \frac{N(N - k)(k - 1)}{2k}
 \end{aligned}$$

(4) サブアセンブリがない状態での関係は $nC2$ であるから、一階層増やすことで、以下の削減効果がある。

$$\frac{S}{nC2} = \frac{\frac{N(N - k)(k - 1)}{2k}}{\frac{N(N - 1)}{2}} = \frac{(N - k)(k - 1)}{k(N - 1)} \quad (5)$$

3.3 摺り合わせ迅速化のメカニズム

ここまでは、2.3 節の手順を実直に守って完成した情報を伝達しプロセスを組み立ててきた。しかし、実際の業務では各作業が完了する時期は設計の難易度やスキルによってばらつく上、設計者は常に設計に専念できるわけではないため、情報の待ちが生じ、その結果アイドルタイムができてしまう。相互依存関係が複雑で、コミュニケーションによる摺り合わせが多数必要なインテグラル型では、よりその傾向が顕著である。

しかし、インテグラル型にも製品戦略上のメリットはあり、IT も含めた組織能力をインテグラル型に適応することで、設計の生産性を上げることが可能である。実際、日本の製造業はインテグラル型に強いと言われている⁽¹⁶⁾。そのような観点から、現在、3D CAD を用いてチーム設計を行っている日本企業のルールを観察すると、共通して以下のような機能を活用しているように見える。これらの適用状況は、まだ統計的な数値を取れたわけではないが、本節で述べたチーム設計のモデルに整合的であるため、以下に説明する。

Fig. 7 Referring the latest working revision

Fig. 8 Check out and modify the other component

(i) 他の設計者の仕掛け状態の参照

Fig.6 では、設計者 Z がアセンブリ A.1(Pa.1, Pb.1)を作成するのを待ってから、設計者 X、Y が調整に着手している。設計者 Z が A.1(Pa.1, Pb.1)を構成しない限り、位置関係が確定しないからである。しかし、Z がアセンブリ A.1 を構成するよりも前に、Pa.1, Pb.1 は作成されているため、設計者 X にしてみれば、先行してその情報を参照できれば、より早く Pa.2 を作成できるかもしれない。つまり、自分が参照しているアセンブリの中で A.0(Pa.1, Pb.1)が一時的に構成できればよい (Fig.7)。そうすれば、設計者 Z が Pa.1, Pb.1 を認識して A.1(Pa.1, Pb.1)を組み上げるのを待たずに設計者 X は作業を終了できる。この行動規則を、各設計者が取れば下位の構成要素の擦り合わせの収束は早くなる。ただし、設計者 Z が Pa, Pb の位置関係を調整する意図を持っている場合は、設計者 X、Y はそれを認識できないため、間違った調整を行ってしまう可能性がある。

(ii) 修正権の移動

これまで、担当の部品が変わらないことを前提としているが、他の人が担当している部品を自分で修正してしまうことで、システム全体で見たアイドルタイムを削減することもできる。つまり、擦り合わせのために相手方の部品を修正する方が適切な場合に、その担当者にその旨を伝達し修正を待つよりも、自分で修正したほうが早いことがあるケースである。ここで、Fig.8 のように設計者 X が設計者 Y の最新リビジョンを参照し、さらに自分の担当部品との調整のために Pb.1 を作成してしまえば、あとは、設計者 Z が A.2(Pa.2, Pb.2)を構成するだけでよい。ただし、設計者 Y と同時に Pb.2 を作成しないように排他制御を行わなければならない。

(iii) 特定のリビジョン構成の保持

(i) と (ii) が多用されると、設計者 Z が確認していない部品が作成されることになる。全て設計者 Z の意図と同じ方向性で作成されれば問題ないが、各人が自律的に設計している以上、時にはそうでないこともありえる。設計者 X、Y が作業する間、設計者 Z が常にアセンブリ A を監視して、Pa, Pb と組み合わせた機能を確認すればすぐに補正できるが、実際にはチームメンバー全員が完全に業務時間の同期を取るのには難しい。

その場合、下位の部品の状態に関わらず、設計者 Z がある時点で機能を確認して構成したアセンブリのリビジョンを厳密に保持しておく、その時点からやりなおしが効くので都合が良い。つまり、下位の構成要素はある程度担当者に任せて多少の間違いを許容しつつ擦り合わせの収束速度を重視し、上位のシステムの機能をやや長めのインターバルで評価して確定したリビジョンの組合せを、チーム内で正の状態として共有しておくのである。

しかし、これらの施策が全ての組織で効果を上げるわけではない。以下のような条件が揃っている必要がある。

1. メンバー間にある程度同じ設計知識がある。言い換えれば、各人が過度の専門性に陥っていない。(i)の他人の仕掛りを参照して、自分の担当部分を修正できるということは、その全体システムの中での機能の意味を理解できているということである。さらに、(ii)の修正まで行うということは、その担当者と同等の知識が必要である。
2. システムの基本的な構成が変化しない。製品の階層や構成が設計中に変化すると、下位の機能が大きく変化し（時には不要となる）、その下位構造を担当している人は自己の判断だけで調整を進めることが困難になる。逆に言えば、製品のアーキテクチャはチーム設計に移る前により少数の設計者内で十分に吟味されている必要がある。
3. 最新が最善の状態であるという暗黙的な同意がある。自己の判断で作成した構造は、それがいかなるものであれ他人に参照される可能性がある。早めに情報を出すことを意識しつつ、生煮え状態であったとしても、ある程度品質の高い案を作成する必要がある。

欧米の技術者は日本に比べて、担当・役割分けが明確で、専門性を重視していると言われている。そのようなチームアサインでは、インテグラル型の擦り合わせを高速化する行動パターン（特に 1）は取りづらい。したがって、各要素間の関係を標準化したモジュラー型の製品の開発に適合していると解釈することができる。

4 結言

本発表において、チーム設計の基本的な枠組みを提示した。今後は、事例を収集し、製品アーキテクチャ、部品点数、開発期間、設計者数、内作率等との関連を取りたいと考えている。

日本企業はインテグラル型のアーキテクチャの開発に強いと言われるが、根拠となるアーキテクチャと日本企業の組織能力との因果関係は「ツーカーの関係」や「あうんの呼吸」「以心伝心」といった曖昧な表現で納得してきた感がある。ここに、CAD や PDM をフレームワークとして分析することで、日本企業が得意とする擦り合わせの妙技の一端が垣間見えたのではないかと考えている。

一つの行動システムとして人間を見ると、それはきわめて単純なものである。その行動の経時的な複雑性は、主として彼が置かれている環境の複雑性を反映したものに他ならない⁽¹⁷⁾。ある製品の開発の経過を事後に振り返れば、そのプロセスは複雑で無秩序で計画通りに進んでいないように見えるのが大半である。チーム設計における環境とは、要求の曖昧さや開発途中での変更、製品構成の複雑さ、各人のスキルの差や労働時間等が挙げられる。その中で設計者が自律的に選択している行動パターンは意外にシンプルなものかもしれない。しかし、その行動パターンが製品アーキテクチャと組み合わせられることで、ある種の複雑系を成し組織能力が創発されるのである。

参考文献

- (1). 沼上幹、組織デザイン、(2004)、日本経済新聞社
- (2). 圓川隆夫、安達俊行、「製品開発論」、(1997)、日科技連
- (3). (株) ITID コンサルティング、2007 年度版 開発力白書、(2008)、(<http://www.itid.co.jp/projects/whitebook/index.html>)
- (4). 中沢俊彦、設計プロセスのモデリングと可視化、日本設計工学会誌 2007 年 4 月号解説文
- (5). 青木昌彦 安藤晴、モジュール化—新しい産業アーキテクチャの本質、(2002)、東洋経済新報社
- (6). 藤本隆宏、アーキテクチャとコーディネーションの経済分析に関する試論、(2008)、東京大学ものづくり経営研究センター ディスカッションペーパー
- (7). 具承桓、製品アーキテクチャのダイナミズム、(2008)、ミネルヴァ書房
- (8). 吉村允孝、モノづくりににおけるシステム設計最適化、(2007)、養賢堂
- (9). 日本機械学会、機械工学便覧デザイン編 β 1 設計工学、(2007)、丸善
- (10). 富山哲男、設計の理論、(2002)、岩波書店
- (11). Nam Pyo.Suh, "The Principles Of Design", (邦訳：畑村洋太郎、設計の原理—創造的機械設計論—、(1992)、朝倉書店)
- (12). Baldwin, Carliss Y. and Kim B. Clark [2000], Design Rules: The Power of Modularity, Cambridge, MA: MIT Press
- (13). 藤本隆宏、武石彰、青島矢一編、「ビジネス・アーキテクチャ 製品・組織・プロセスの戦略的設計」、有斐閣、(2001)
- (14). Ulrich, Karl T. and Steven D. Eppinger, Product Design and Development, New York: McGraw-Hill
- (15). Pahl, G., Beitz, W., 工学設計 体系的アプローチ、(1995)、倍風館
- (16). 藤本隆宏、生産マネジメント入門、(2001)、日本経済新聞社
- (17). Simon, Harvert A. "The Architecture of Complexity: Hierarchic Systems," The Science of Artificial, 3rd ed [1996] Cambridge, MA: MIT Press (稲葉元吉、吉原英樹訳 [1999]、「システムの科学」第 3 版、パーソナルメディア)