製品開発マネジメントの分析ツールとしての設計構造マトリックスに関する考察

広島大学大学院社会科学研究科附属地域経済システム研究センター助手　目　代　武　史*

要　旨

本稿の目的は、複雑な製品開発を分析するためのツールである設計構造マトリックス（DSM: Design Structure Matrix）について、その基本概念と最近の研究動向を整理するとともに、DSM の分析ツールとしての利点と限界点を明らかにすることである。DSM は、開発製品や開発組織などをシステムの観点から捉え、システムを構成する要素間の依存関係を簡潔に示す行列である。分析対象に応じて、コンポーネント DSM、チーム DSM、タスク DSM、パラメータ DSM などがある。

近年の研究動向を見ると、(1) 開発製品や開発組織の構造を最適化する流れと (2) 開発プロセスを最適化する流れがある。構造の最適化には主にクラスタリング、プロセスの最適化にはパーティショニングといった分析手法が用いられる。

他の製品開発手法と比較した DSM の利点は、記述する情報の簡潔性、網羅性、一覧性、操作性の良さにある。ただし、DSM は分析手法の一つに過ぎず、製品開発のすべてを分析できるわけではない。DSM には、要素間の未知の依存関係に対する脆弱性、スタティックな分析アプローチ、組織の知識蓄積に関する分析の弱さといった限界点があることに留意する必要がある。

キーワード：設計構造マトリックス、製品開発、製品アーキテクチャ

1．はじめに

本稿の目的は、複雑な製品開発を分析するためのツールとして近年注目されている設計構造マトリックス（Design Structure Matrix、以下 DSM）の基本概念と最近の研究動向を整理するとともに、DSM の分析ツールとしての利点と限界点を明らかにすることである。

DSM は、1970年代に Donald V. Steward が考案し、90年代に入ってからマサチューセッツ工科大学（MIT）の Steven D. Eppinger や A. Yassine 等によって拡張され、製品開発研究へ応用が進んでいった。とりわけ Baldwin and Clark（2000）において、製品アーキテクチャを記述するツールとして紹介されて以来、DSM が広く知られるようになった。

本研究の背景には、実践面および研究面における次のような課題がある。

まず、実践面では、製品開発マネジメントの鍵の一つは、複雑性への対処にある。製品に求められる機能や品質レベルの上昇に伴い、製品設計の複雑性はますます高まっている。また、製品が必要とする科学・技術知識は高度化や複合化が進展し、一つの製品開発プロジェクトにさまざまな個人や組織が携わるようになり、組織の複雑性も増している。さらに、市場投入のリードタイムを短縮するため、設計開発工程のオーバーラップや問題解決のフロントローディングが求められており、開発プロセスにおいても短期間で多くの問題解決をこなさなければならず、開発プロセスの複雑性も高まっている。

このような複雑性に対処するために、われわれは通常、全体システムをより単純な下位システム

* 連絡先：〒730-0053　広島市中区東千田町1-1-89
TEL（082）542-6992　FAX（082）249-4991
Email: mokudai@hiroshima-u.ac.jp

や要素へと分割していく。システムの複雑性は、構成要素の数が増えるほど、また要素間の繋がりが増えるほど、増大していく。製品開発においては、個々の要素の開発に加え、要素を統合して一つの製品にまとめ上げる必要がある。そのプロセスは、しばしば試行錯誤的であり、作業の繰り返しを伴う。このことが、製品開発マネジメントの複雑性の大きな源泉となっている。

ところが、製品開発マネジメントのツールとして知られるガントチャートや PERT などでは、こうした要素間の込み入った相互依存性から生じる開発マネジメントの複雑性を十分に捉えることができない。ガントチャートや PERT は、開発期間や所要資源を規定する重要なポイントやボトルネック、すなわちクリティカルパスを明らかにする上では有効なツールである。しかし、そうしたクリティカルパスにおいて開発作業の遅れや開発工数の増大といった問題がいかに生じるのかは十分に説明できない。なぜなら、ガントチャートや PERT では各工程の所要時間や所要工数は、少なくとも短期的には与件として扱われるからである（藤本、2001）。

DSM は、要素間やタスク間の相互依存性を直接分析の対象とすることで、従来の分析ツールが捉え切れなかった複雑性の問題をより効果的に分析できる。これが DSM を研究する実践上の背景である。ただし、DSM は他の分析ツールに対して全面的に優れているというわけではなく、他の分析ツールと相互補完的に用いられるべきである。本稿では、DSM の制約条件や限界点についても若干の考察を行う。

次に、本稿の学術研究面の背景として、製品アーキテクチャ研究の進展がある。製品アーキテクチャ研究は、製品開発における複雑性の問題を要素間の関係性の観点から分析する研究アプローチである。製品アーキテクチャとは、製品をいかに要素に分解し、そして分解した要素をいかに統合するかに関する基本的な設計構想である（Ulrich, 1995; 藤本・武石・青島編、2001）。製品アーキテクチャ研究では、様々な製品や産業を対象として、製品アーキテクチャの特性自体の分析、製品アーキテクチャが組織構造や組織関係に与える影響の分析、産業構造やイノベーションに与える影響の分析などが行われてきた。それらの先行研究の多くは、綿密な調査に基づいているものの、製品アーキテクチャ特性の記述は、研究者の主観的な観察やアンケート調査などに依拠しており、記述の客観性や再現可能性、比較可能性などに方法論上の限界があった。

DSM による製品設計特性の記述は、そうした従来の製品アーキテクチャ研究の方法論上の弱点を補うものである。2 節以降で詳しく述べるように、DSM は要素間やプロセス間の作用・被作用の関係を網羅的、体系的に記述することができる。また、DSM は情報の要約性や一覧性に優れるため、研究者と実務家との間のコミュニケーションを容易にし、より正確な製品アーキテクチャ特性の記述および分析に貢献すると考えられるのである。

本稿の構成は以下の通りである。2 節において、DSM の基本的な概念や分析手法について説明する。3 節では1990年代以降の主な研究をとりあげ、DSM を用いた最近の研究動向を整理する。そして 4 節では、製品開発研究における DSM の限界点ないし留意点について考察する。最後に 5 節において、本稿の結論を述べる。

2．設計構造マトリックスの概要

⑴ 基本概念

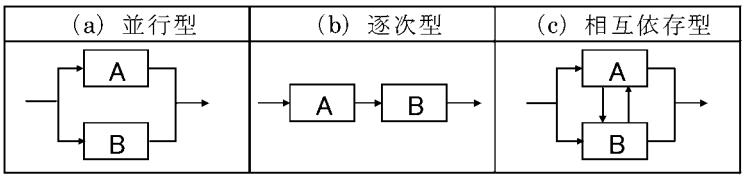
DSM は、開発製品や開発組織などをシステムの観点から捉え、システムを構成する要素間の依存関係を網羅的に示す行列である。要素間の依存関係には、物理的な関係、情報の受け渡し関係、エネルギーの交換関係、空間的な干渉関係などがある。

DSM では、システムの複雑性を要素間の依存関係に還元して記述する。要素間の関係は、作用の方向性を持った有向グラフ（directed graph; diagraph）で示すことができる。図 1 は、要素間のもっとも基本的な依存関係のパターンを示している。

並行型は、要素 A と要素 B が依存関係を持たないパターンである。依存関係を持たないとは、例えば要素 A を設計するために、要素 B からの何らかのインプットを必要としないということである。そのため、並行型では、要素 A と要素 B を互いに独立に設計することができる。

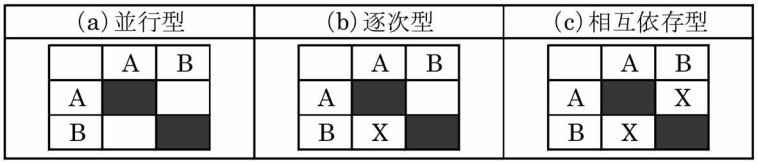
逐次型は、要素 A が要素 B に影響を与えるが、

図1　要素間の依存関係（グラフ表示）



（出所）Yassine（2004）

図2　要素間の依存関係（**DSM** 表示）



（出所）Yassine（2004）

その逆の影響はないパターンである。すなわち、要素 B を設計するためには、あらかじめ要素 A からのインプットが必要となる。

相互依存型は、要素 A と要素 B がそれぞれ影響を与え合うパターンである。すなわち要素 B を設計するためには、要素 A からのインプットが必要であり、さらに要素 B の設計の結果が要素 A の設計に影響を与える。この場合、要素間で循環的に影響の連鎖が生じるので、何らかの収斂のメカニズムが働かないと、いつまでも設計が完了しない恐れがある。

こうした要素間の依存関係のパターンを 2 × 2 の DSM で表示したのが図 2 である。

DSM は、次のように読むことができる。

1. 行および列は、システムを構成する要素を表し、行列とも同順序に配列している。
2. リストは上から下へと読んでいき、要素 A が要素 B に影響する（入力になる）場合、A 列と B 行の交点をマーク（X）する。
3. 列方向に読むと、その要素がどの要素へ影響しているかが分かる。
4. 行方向に読むと、その要素がどの要素から影響を受けているかが分かる。

図 2 に戻ると、(a) 並行型では、要素 A と要素 B との間で依存関係がないため、A 列 B 行も B 列 A 行も空欄となっている。(b) 逐次型では、A 列 B 行に印 X が入っており、要素 A が要素 B に影響を与える関係となっている。(c) 相互依存型では、A 列 B 行にも B 列 A 行にも印 X がついており、要素 B は要素 A から影響を受け、同時に要素 A は要素 B から影響を受けるケースを示す。

(2) 依存関係の記述方式

要素間の依存関係を記述するもっとも単純な方法は、依存関係の有無を 2 値的（binary）に捉えることである。すなわち、要素 A と要素 B との間に依存関係があれば、DSM の所定の欄に印（X）を記入する方式である。依存関係の有り無しで記述した DSM を 2 値 DSM（binary DSM）という。

もう一つの代表的な記述方式は、依存関係の強弱や必要性などを数値でウェート付けして記述する方法である。依存関係の数値表現には次のような尺度がある。

1. 強 0.3、中 0.2、弱 0.1、無 0（Sharman, et al. 2002; Yassine, et al. 2003など）
2. 必要 +2、望ましい +1、中立 0、望ましくない −1、有害 −2（Pimmler and Eppinger, 1994; Sosa, et al. 2003）
3. タスクの繰り返し確率（P）：0 ＜ P ＜ 1（Cronemyr, et al. 2001; Smith and Eppinger, 1997; Yassine, 2001）

第一に、依存関係の強弱を記述する場合、依存関係がない場合（0 点）を起点として、依存関係が強まるほど点数が一方向的に高まっていく。要

表1　**DSM** の基本タイプ

目的	DSM タイプ	表示対象	適用対象	分析手法
構造最適化	(a)コンポーネント DSM もしくはアーキテクチャ DSM	製品アーキテクチャにおける部品とその関係性	システム基本構造設計、エンジニアリング、等	クラスタリング
	(b)チーム DSM もしくは組織 DSM	組織における個人、グループ、チームとその関係性	組織設計、インタフェース管理、組織間統合	クラスタリング
プロセス最適化	(c)タスク DSM もしくはスケジュール DSM	プロセスにおける諸活動とそのインプット及びアウトプット関係	プロジェクトのスケジューリング、活動の順列化、サイクルタイム短縮	パーティショニング、ティアリング、シミュレーション、固有値分析
	(d)パラメータ DSM	設計におけるパラメータとその関係性	末端レベルの活動の順列化、プロセスの構築	パーティショニング、ティアリング、シミュレーション、固有値分析

（出所）Browning（2001），Yassine（2004）から引用。一部変更、加筆。

素間の依存関係を客観的もしくは記述的に捉える場合に用いられる尺度である。例えば、開発組織の DSM（チーム DSM、後述）の場合、開発チーム間のコミュニケーション頻度が毎日（強い依存関係)、週に数度（中)、月に数度（弱）などの形で依存関係を記述する。

第二に、依存関係を望ましさの観点から規範的に記述する尺度もある。例えば、部品間で物理的な依存関係が機能実現のために不可欠であれば「必要（＋2）」、依存関係があれば有用だがなくても機能は損なわなければ「望ましい（＋1）」、依存関係と機能性が中立もしくは無関係な場合は「中立（0）」、依存関係は無い方がよいのだがあっても機能を損なわない場合は「望ましくない（－1）」、依存関係があると機能を損なう場合は「有害（－2）」などと記述する。

第三に、依存関係を確率的に記述する尺度もある。この尺度は、開発作業間の依存関係を分析する場合（タスク DSM）などにしばしば用いられる。ある作業 B の結果によって、別の作業 A が影響を受ける（例えば、作業のやり直しが発生する）ような場合、その影響を受ける確率（P）を $0<P<1$ の範囲で記述するのである。

2 値的な記述方式の利点は簡便性にある。DSM の主要な目的の一つは、分析対象となるシステムの特性を、その複雑性の源泉となる構成要素間の依存関係に着目し、その他の情報は捨象して記述することである。2 値 DSM は、この目的をもっとも簡潔な形で実現するものである。

他方、数値的な記述方式は、依存関係の程度をより詳しく記述する。実際の分析では、2 値的方式と数値的方式は補完的に用いられる。すなわち、DSM 作成の第一段階では、2 値的記述により要素間の依存関係の有無をまず記述し、必要であれば、第二段階でさらに調査を行い、2 値的な依存関係を数値に置き換えるなどの方法が多くの先行研究でとられている（Sharman, et al. 2002; Yassine, et al. 2003)。

(3) **DSM** のタイプと分析手法

記述対象の違いによって、DSM にはいくつかのタイプがある（Browning, 2001)。表 1 は、DSM の基本タイプを示している。DSM はタイプによって、分析目的や分析対象、分析手法などが異なる。ここでは、DSM タイプ別に分析対象や手法について概説し、分析目的については 3 節であらためて検討する。

（**a**）コンポーネント **DSM**

コンポーネント DSM（Component-based DSM）は、製品システムを部品もしくはサブシステムに分解し、部品（もしくはサブシステム）間の依存関係を記述するものである。行列に表示される要素は、分析対象となる製品を構成する部品である。部品の間でどのような依存関係（空間、エネルギー、情報、物質など）がどのような形（並行、逐次、循環）でどの程度（強、中、弱）存在するのかを記述する。

図３　自動車用空調システムのコンポーネント **DSM**

		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Radiator	A		2 0 0 2			2 -2 0 0											
Engine fan	B	2 0 0 2				2 0 0 2								1 0 0 0			
Heater core	C				1 0 0 0			2 0 0 0	-1 0 0 0	-1 0							0 0 0 2
Heater hoses	D			1 0 0 0						0 0							
Condenser	E	2 -2 0 0	2 0 0 2				0 2 0 2		-2 2 0 2								
Compressor	F					0 2 0 2			0 2 0 2	1 0 0 2	0 0 2 0	0 0 2 0		1 0 0 0			
Evaporator case	G			2 0 0 0					2 0 0 0						2 0 0 0	2 0 0 0	2 0 0 2
Evaporator core	H			-1 0 0 0		-2 2 0 2	0 2 0 2	2 0 0 0		1 0 0 2							0 0 0 2
Accumulator	I				-1 0 0 0		1 0 0 2		1 0 0 2		1 0 0 0						
Refrigeration controls	J						0 0 2 0			1 0 0 0		0 0 2 0		1 0 0 0			
Air controls	K						0 0 2 0				0 0 2 0		0 0 2 0	1 0 0 0	0 0 2 0	0 0 2 0	
Sensors	L											0 0 2 0		1 0 0 0			
Command distribution	M		1 0 0 0				1 0 0 0				1 0 0 0	1 0 0 0	1 0 0 0		1 0 0 0	1 0 0 0	1 0 0 0
Actuators	N							2 0 0 0				0 0 2 0		1 0 0 0			
Blower controller	O							2 0 0 0				0 0 2 0		1 0 0 0			2 0 0 2
Blower motor	P			0 0 0 2				2 0 0 2	0 0 0 2					1 0 0 0		2 0 0 2	

（注）　空間 S E エネルギー
情報 I M 物質
空欄は、依存関係が存在しないこと（空間、情報、エネルギー、物質ともに０）を示す。
（出所）Pimmler and Eppinger（1994）

図４　自動車用空調システムのコンポーネント **DSM**（クラスタリング後）

		K	J	L	D	M	A	B	E	F	I	H	C	P	O	G	N
Air controls	K		0 0 2 0	0 0 2 0		1 0 0 0				0 0 2 0					0 0 2 0		0 0 2 0
Refrigeration controls	J	0 0 2 0				1 0 0 0				0 0 2 0	1 0 0 0						
Sensors	L	0 0 2 0				1 0 0 0											
Heater hoses	D						制御・接続クラスター				-1 0 0 0		1 0 0 0				
Command distribution	M	1 0 0 0	1 0 0 0	1 0 0 0				1 0 0 0		1 0 0 0				1 0 0 0	1 0 0 0		1 0 0 0
Radiator	A							2 0 0 2	2 -2 0 0	フロントエンド・エア・クラスター							
Engine fan	B					1 0 0 0	2 0 0 2		2 0 0 2								
Condenser	E						2 -2 0 0	2 0 0 2		0 2 0 2		-2 2 0 2					
Compressor	F	0 0 2 0	0 0 2 0			1 0 0 0			0 2 0 2		1 0 0 2	0 2 0 2	冷却クラスター				
Accumulator	I		1 0 0 0		-1 0 0 0					1 0 0 2		1 0 0 2					
Evaporator core	H								-2 2 0 2	0 2 0 2	1 0 0 2		-1 0 0 0	0 0 0 2		2 0 0 0	
Heater core	C				1 0 0 0							-1 0 0 0		0 0 0 2		2 0 0 0	
Blower motor	P					1 0 0 0						0 0 0 2	0 0 0 2		2 0 0 2	2 0 0 2	
Blower controller	O	0 0 2 0				1 0 0 0			内装エア・クラスター					2 0 0 2		2 0 0 0	
Evaporator case	G											2 0 0 0	2 0 0 0	2 0 0 2	2 0 0 0		2 0 0 0
Actuators	N	0 0 2 0				1 0 0 0										2 0 0 0	

（注）　空間 S E エネルギー
情報 I M 物質
空欄は、依存関係が存在しないこと（空間、情報、エネルギー、物質ともに０）を示す。
（出所）Pimmler and Eppinger（1994）

図3は、自動車用空調システムのコンポーネント DSM の事例である。事例の空調システムは、ラジエータやエンジン、ヒーターコアなど16の物理的な要素、すなわち部品から構成されている。部品間の依存関係は、空間的干渉、エネルギー交換、情報交換、物質交換の観点から、それぞれ数値尺度（必要 +2、望ましい +1、中立 0、望ましくない −1、有害 −2）で記述されている。例えば、E 列（コンデンサ）をみると、A 行（ラジエータ）とは空間上必要な依存関係があり、エネルギー面では有害な依存関係があることが分かる。また、B 列（エンジンファン）に対しては、空間面および物質面で必要な依存関係がある。なお、対角線上の点は、ある部品の自分に対する依存関係を意味するため、空欄となっている。図3をみると、DSM 全体に依存関係が散らばっていることがわかる。

行列を並び替え、依存関係にある要素を対角線の近傍にグループ化する分析手法をクラスタリング（clustering）という。一般に、クラスタリングにおいては、各クラスターの内部では部品間の依存関係が密集し、クラスター間の依存関係は最小限に抑えるように要素の並べ替えを行う。依存関係の強い部品を一つの塊、すなわちモジュールにまとめることにより、設計過程において部品間の重要な依存関係を見逃すのを防いだり、依存関係の強い部品同士を統合するなどして、問題解決を容易にするのがクラスタリングの狙いである。

図3の空調システムを、物理的な依存関係の観点からクラスタリングしたのが図4である。図4では、フロントエンド・エア（部品 A、B、E）、冷却（部品 E、F、I、H）、内装エア（部品 H、C、P、O、G、N）の3つのクラスターに加え、制御・接続クラスター（部品 K、J、L、D、M）が抽出されている。図4のケースでは、クラスター外部の依存関係は、制御・接続クラスターに集約されている。

このようにコンポーネント DSM が記述するのは、製品のアーキテクチャ特性である。クラスターをモジュールと読み替えれば、クラスタリングは、製品アーキテクチャのモジュール化の手法といえる[1)]。小さな DSM では、クラスタリングを手動で行うことも可能であるが、大きな DSM では手動でクラスタリングすることは現実的でない。クラスタリングの方法については、コンピュータで自動的に計算するためのアルゴリズムがいくつか開発されており、一般に公開されている[2)]。

（b）チーム DSM

チーム DSM（Team-based DSM）もしくは組織 DSM（Organizational DSM）は、開発組織を構成する作業チームやグループ間の依存関係を記述する DSM である。行列に表示される要素は、分析対象となる組織を構成するチームや個人である。作業チーム間でどのような依存関係がどのような形でどの程度存在するのかを記述する。

図5は、自動車エンジン開発の組織 DSM の事例である。事例の開発組織は、エンジンブロック担当、シリンダーヘッド担当、カムシャフト／バルブトレイン担当など22のチームから構成される。各チームは、CAD 設計者や生産技術者、購買担当者など数人の構成員からなっている。

チーム間の依存関係は、チーム間の情報交換の頻度により捉えられる。情報交換の頻度がほぼ毎日あれば高い依存性（大きな丸印）、週に一回程度であれば中程度の依存性（中位の丸印）、情報交換が必要ではあるがその頻度は高くない場合は低程度の依存性（小さな丸印）と判断する。図7をみると、丸印が DSM 全体に広がっており、あるチームが仕事を遂行するためには、広範なチームと情報交換が必要なことがわかる。

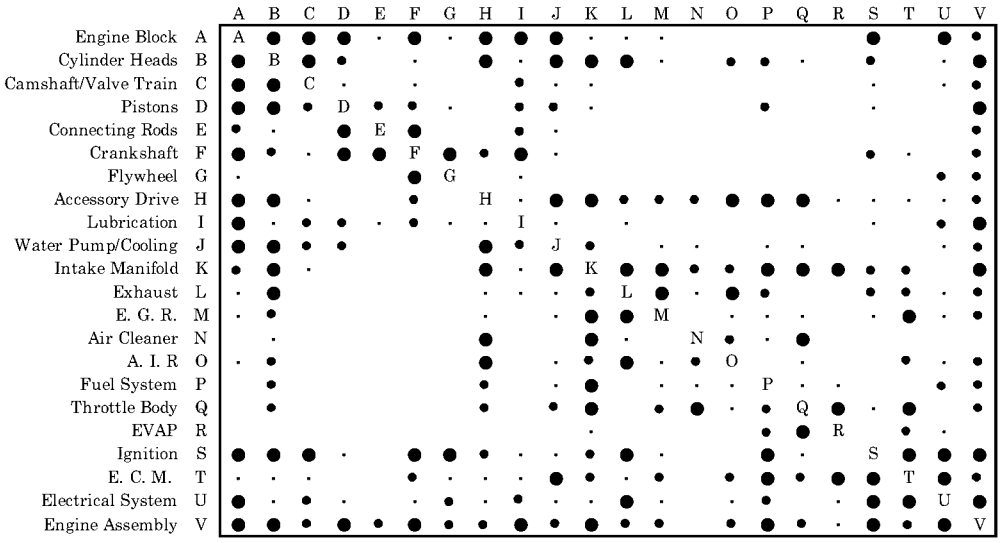
チーム DSM の分析に用いられるのは主にクラスタリングである。行列を並び替えて、依存関係の強いチーム同士は同じクラスターにまとめ、クラスター間では依存関係を抑制する。クラスタリングのアルゴリズムは、コンポーネント DSM と同じである。

図6は、図5の開発組織のクラスタリングを行った結果である。5つのチーム・クラスターが

した機能をもつように全体システムを切り分けると共に、下位システム間のインターフェースを標準化するようにアーキテクチャを変換することである。これと対をなす概念にインテグラル化がある。インテグラル化とは、全体システムを構成する下位システム同士が機能的、構造的に互いに結び付きを持つようにアーキテクチャを変化させることである。

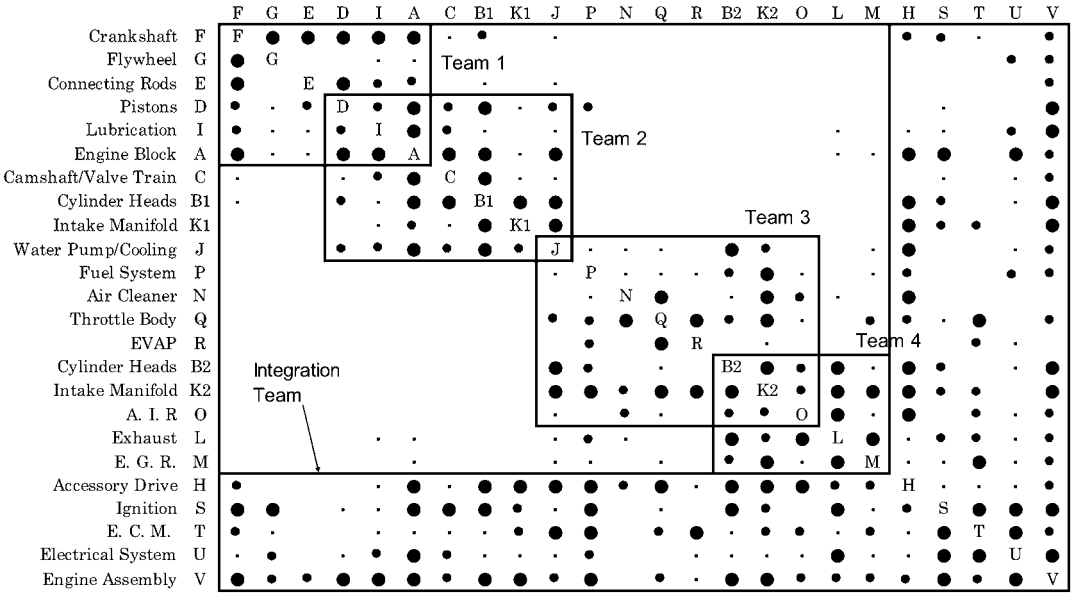
2）http://www.dsmweb.org/DSM_tools.htm

図５　自動車エンジン開発のチーム **DSM**



（注）開発チーム間の情報交換の頻度：●毎日、●毎週、・毎月
（出所）McCord and Eppinger（1993）

図６　自動車エンジン開発のチーム **DSM**（クラスタリング後）



（注）開発チーム間の情報交換の頻度：●毎日、●毎週、・毎月
（出所）McCord and Eppinger（1993）

抽出されている。クラスター１から４は、部品の関連性が高く頻繁なコミュニケーションが必要なチームの集合を示している。クラスター内ではチーム間で密なコミュニケーションがあるが、クラスター間の情報交換は最小限となっている。いくつかの開発チーム（例えば、ピストン担当、潤滑担当、エンジンブロック担当）は隣り合うクラスター（クラスター１および２）の両方に所属することで、クラスター間の情報交換チャネルが確保されている。クラスター５は、ほとんどすべてのチームと情報交換チャネルを持っており、エンジン開発の全体を統合する役割を持っている。

図７　電気自動車開発プロセスのタスク **DSM**

		2	3	4	5	7	9	11	12
Size and aerodynamics	2	⑦	0.1			0.1		0.3	0.2
Motor specifications and weight	3	0.1	②	0.2		0.1		0.1	
Total weight	4	0.2	0.2	①		0.1		0.2	0.2
Stored energy requirement	5		0.3		①		0.2		
Battery size and weight	7				0.7	①			
Speed and acceleration versus power	9	0.3		0.2			②		0.2
Speed and acceleration conformance	11						0.6	①	
Structural and suspension design	12	0.3	0.1	0.2		0.1		0.2	⑤

（注）対角線上のまるで囲まれた数字は、当該タスクの所要時間を表す。
（出所）Smith and Eppinger（1997b）

図８　電気自動車開発プロセスのタスク **DSM**（パーティショニング後）

		4	9	11	3	5	7	2	12
Total weight	4	①		0.2	0.2		0.1	0.2	0.2
Speed and acceleration versus power	9	0.2	②					0.3	0.2
Speed and acceleration conformance	11		0.6	①					
Motor specifications and weight	3	0.2		0.1	②		0.1	0.1	
Stored energy requirement	5		0.2		0.3	①			
Battery size and weight	7					0.7	①		
Size and aerodynamics	2			0.3	0.1		0.1	⑦	0.2
Structural and suspension design	12	0.2		0.2	0.1		0.1	0.3	⑤

（注）対角線上のまるで囲まれた数字は、当該タスクの所要時間を表す。
（出所）Smith and Eppinger（1997b）

チーム DSM におけるクラスタリングの狙いは、頻繁な情報交換の必要なチーム同士を一つにまとめたり、物理的に近くに配置することにより、コミュニケーションを円滑化することにある。例えば、職務の関連し合うチームの大部屋への配置はその一例である。チーム DSM の分析では、コンポーネント DSM と同様に、要素のまとまりが重要であり、要素の順序はとくに問題ではない。

（c）タスク **DSM**

タスク DSM（Task-based DSM）は、製品開発プロセス等における各種の活動（タスク）の間の依存関係を記述する DSM である。行列に表示されるのは、分析対象となるプロセスを構成するタスクである。タスク間でどのような依存関係がどのような形でどの程度存在するのかを記述する。

図７は、電気自動車開発プロセスの仮想事例である（Steward, 1981; Smith and Eppinger, 1997b）。事例の開発プロセスは、車両サイズ・空力設計、モータ仕様・重量設計、車両重量設計など８つのタスクから構成される[3)]。タスク間の依存関係とは、あるタスクが別のタスクの入力関係にある場合を意味する。例えば、車両重量設計（４列）は、モータ仕様・重量設計（３行）、加速・馬力比設計（９行）、車体構造・サスペンション設計（12行）に影響を与える。

タスク DSM では、タスクの順序に重要な意味がある。対角線より下にある印は、フィードフォーワード、すなわちタスクが前工程から後工程へと逐次的に進むことを意味する。一方、対角線より上にある印は、フィードバック、すなわち後工程から前工程へとタスクが逆流することを意味する。例えば、後工程における設計変更が前工程の設計の見直しをもたらすようなケースである。図７では、モータ仕様・重量設計から車両サイズ・空力設計への影響（３列１行）、車両重量設計

3）事例のオリジナル版は Steward（1981）で、16のタスクで構成される。同じ事例を引用した Smith and Eppinger（1997b）ではその一部を用いて分析を行っている。

からモータ仕様・重量設計への影響（4列3行）などがフィードバックを示している。

タスク間にフィードバック・ループが存在する場合、最初のタスクは、後工程のタスクからのフィードバックを推測して遂行される。例えば、図7によれば、車両サイズ・空力設計は、モータ仕様・重量設計に先立って遂行されるが、車両サイズ・空力設計を完成させるには、モータ仕様・重量設計、車両重量設計などの下流工程からのフィードバックを必要とする。すなわち、車両サイズ・空力設計を最初に行う時点では、実際のモータ仕様・重量設計や車両重量設計がこうなるだろうという予測や仮定に基づいて行われるのである。したがって、この仮定の精度を上げることが繰り返しの可能性や程度を削減する重要な鍵となる。

このような試行錯誤的なタスクの繰り返しは、革新的な製品を生み出すためには、ある程度必然となるプロセスであるが、行き過ぎると開発リードタイムの延長、開発資源の浪費などの弊害をもたらす（Eppinger, 2001）。タスクの繰り返しによる弊害を低減し、製品開発の所定の目標を達成するため、開発プロセスの最適化を図る必要がある。

パーティショニング（partitioning）もしくは順列化（sequencing）は、開発プロセスの最適化を図るための分析手法である。パーティショニングとは、DSMの行列を並び替えることによって、タスク間の依存関係を示す印Xを対角線より下に移動させる手法である。パーティショニングも小規模なDSMでは手動で行うことが可能であるが、コンピュータで自動的に行うプログラムも一般に利用可能である[4)]。

図8は、パーティショニングによってタスクの順列化を図ったDSMである。図5のDSMと比較して、対角線より上の印が減少し、フィードバック・ループが削減されていることが分かる。

しかし、パーティショニングを行っても、実際のDSMで対角線より上の印Xをすべて取り除くことはほとんど不可能である。その場合、フィードバック・ループを効率的に管理可能なレベルに制御することを考えなければならない。そのためのポイントは、対角線より上の印Xをできるだけ対角線に近づけることである。対角線の近くにフィードバック・ループがあるということは、開発プロセス上近くにあるタスクの間で繰り返しが行われるということであり、タスク間の連携がやりやすい。また、繰り返しのサイクルも短期間で回すことができる。対角線よりも離れた場所に印Xが存在する場合、仕事がかなり下流のタスクまで流れてから、上流のタスクに仕事が戻ってくることを意味し、繰り返しのサイクルが非常に長くなってしまうからである。

繰り返しが必要なタスクを識別し、それらを予め連携作業として開発プロセスに組み込むことを「計画した反復プロセス」という（Eppinger, 2001）。コンカレント・エンジニアリングは、計画した反復プロセスの典型である。

問題となるのは、対角線から離れた場所に予想外の印Xが存在する場合である。それらは「計画外の反復」であり、タスク間の予想外の依存関係によって、開発プロセスが大幅に遅延したり、問題の解決に膨大な工数を浪費させる危険性がある。このような場合、デザインレビューなどにより開発の節目の管理を徹底すること、致命的な欠陥でなければ、そのまま開発製品を市場投入し、次世代の開発プロジェクトのための教訓として扱うこと、欠陥が致命的な場合プロジェクトを放棄してしまうこと、などの対策が考えられる（Eppinger, 2001; Yassine, et al., 2000）。

（d）パラメータ DSM

パラメータDSM（Parameter-based DSM）とは、製品設計を構成するパラメータ間の依存関係を記述するDSMである。パラメータDSMは、製品開発におけるもっとも末端のレベルの意思決定に関わっている。設計パラメータ間でどのような依存関係がどのような形でどの程度存在するのかを記述する。

図9は、自動車用ブレーキシステムの事例である（Smith and Eppinger, 1997a; Yassine, 2004）。事例のブレーキシステムは、顧客ニーズ、車輪トルクなど13のパラメータで構成される[5)]。パラメータ間の依存関係とは、あるパラメータが別の

4）http://www.dsmweb.org/DSM_tools.htm

5）Smith and Eppinger（1997a）のオリジナルDSMは105の要素から構成されるが、簡単化のため、主要な依存関係を含む13の要素を抽出。

図９　自動車用ブレーキシステムのパラメータ **DSM**

		1	2	3	4	5	6	7	8	9	10	11	12	13
Customer requirements	1	1												
Wheel torque	2		2		X									
Pedal mechanical advantage	3	X		3	X	X			X		X			X
System level parameters	4	X			4									
Rotor diameter	5	X	X	X	X	5		X	X		X	X		X
ABS modular display	6		X				6			X				
Front lining coeficient of friction	7			X	X	X		7	X		X			X
Piston-rear size	8		X		X				8		X			
Caliper compliance	9			X	X					9	X			X
Piston-front size	10		X		X				X		10			
Rear lining coeficient of friction	11			X	X	X			X		X	11		X
Booster -Max. stroke	12												12	X
Booster reaction ratio	13		X	X	X	X		X	X	X	X	X	X	13

（出所）Yassine（2004）

図10　自動車用ブレーキシステムのパラメータ **DSM**（パーティション後）

		1	4	2	10	8	3	11	7	13	5	12	9	6
Customer requirements	1	1												
System level parameters	4	X	4											
Wheel torque	2		X	2										
Piston-front size	10		X	X	10	X								
Piston-rear size	8		X	X	X	8								
Pedal mechanical advantage	3	X	X		X	X	3			X	X			
Rear lining coeficient of friction	11		X		X	X	X	11		X	X			
Front lining coeficient of friction	7		X		X	X	X		7	X	X			
Booster reaction ratio	13		X	X	X	X	X	X	X	13	X			
Rotor diameter	5	X	X	X	X	X	X	X	X	X	5			
Booster -Max. stroke	12									X		12		
Caliper compliance	9		X		X		X			X			9	
ABS modular display	6												X	6

（出所）Yassine（2004）

パラメータの入力関係にある場合を意味する。例えば、ローター直径（５列）の決定は、ペダル機構アドバンテージ（３行）、フロントライニング摩擦係数（７行）、リアライニング摩擦係数（11行）、ブースター反発比率（13行）のパラメータ設定に影響を与える。

パラメータ DSM では、タスク DSM と同様、対角線より上にある依存関係が重要な意味を持つ。すなわち、対角線より下にある印 X は、パラメータ間のフィードフォーワードを意味する一方、対角線より上にある印 X は、パラメータ間のフィードバックを表す。フィードバックが存在する場合、先に決定したパラメータがその後に決定するパラメータの結果により、再度見直す必要がある。いわゆる設計変更による設計の手戻りが発生するのである。

このような設計の手戻りを最小化するために、パラメータ DSM ではパーティショニングが行われる。タスク DSM と同じ要領で、行列を並び替え、印 X を対角線より下に移動させるのである。図10は、図９をパーティションにより順列化した結果である。パーティション後に残ったフィードバックは、できるだけ対角線に近づけるとともに、コンカレント・エンジニアリングなどにより、計画的にパラメータ設計の繰り返しを管理する必要がある。

3．DSM 研究の動向

DSM を用いた研究は、1990年代以降、MIT の研究者等を中心にして、分析手法の開発が進み、応用分野も広げられていった。表 2 は、近年の DSM 研究を整理したものである。DSM の研究動向を振り返ると、大きく分けて（1）構造を最適化する流れと（2）プロセスを最適化する流れがある。

⑴ 構造の最適化

構造の最適化とは、分析対象となるシステムを構成する要素間の依存関係を整理することによって、要素間の相互作用から発生する複雑性を管理可能なレベルにコントロールしようとする研究の方向性である。コンポーネント DSM を用いた製品アーキテクチャの分析やチーム DSM を用いた組織構造の分析がこれに当る（McCord, K. R. and S. D. Eppinger, 1993; Pimmler, T. U. and S. D. Eppinger, 1994; Sharman, D. M. and A. A. Yassine, 2004; Yu, et al. 2003）。

製品アーキテクチャの構造最適化の研究は、モジュラー化の研究とほとんど同義である。構造最適化の研究では、コンポーネント DSM によってまず構成要素の間の依存関係を記述し、次にクラスタリングによって要素間の依存関係を整理するのが一般的な分析パターンである。すなわち、構成要素がクラスターの内部では緊密な依存関係を持つが、クラスターの外部との依存関係は最小化するように行列を並び替える。それにより、要素間の依存関係から生じるシステム複雑性をクラスター内に閉じ込めるのである。このクラスターをモジュールと読み換えれば、コンポーネント DSM におけるクラスタリングは、統合型アーキテクチャを持つ製品をモジュラー型に転換するための方法論を意味することが分かる。

DSM による製品アーキテクチャのモジュラー化は、ボトムアップ的である。すなわち、新規の技術や部品の導入によってモジュール化を図るのはなく、現状の部品構成とそれらの依存関係を出発点として、それをクラスタリングすることで、モジュールの形成を図るアプローチとなっている。Henderson and Clark（1990）は、このように既存の部品間のつなぎ方の見直しによってイノベーションを実現することをアーキテクチュアル・イノベーションと呼んでいる。また、Baldwin and Clark（2000）は、DSM による製品アーキテクチャの分析によって、製品アーキテクチャのモジュラー化の推進要因やプロセスを詳細に分析している。

構造の最適化は、開発組織についても様々な研究がある。例えば、McCord and Eppinger（1993）は、自動車用エンジンの開発組織を対象として、各部品の開発担当チームのコミュニケーション頻度を分析し、コミュニケーション頻度の高い開発担当チームのグループ化を行っている。これによりグループ内におけるコミュニケーション効率を改善する一方、グループ間での不必要な依存関係を削減し、組織全体の開発効率を高める組織構造を提示している。

また、組織構造の最適化問題は、しばしば製品アーキテクチャとセットで分析される。例えば、Sosa, et al.（2004）は、航空機エンジンの詳細設計フェーズについて、コンポーネント DSM とチーム DSM を記述し、両者の適合関係を分析している。分析の結果、コンポーネント間に依存関係があるにもかかわらず、それに組織が対応できていないケースが数多く見られた。そうした見逃されたコンポーネント間の依存関係は、しばしば開発担当チームの境界線上に存在するため、開発チーム同士で適切なコミュニケーションがとられていないことが原因となっていることが明らかにされた。こうした製品設計と組織との適合関係の分析は、Baldwin and Clark（2000）でも行っており、製品アーキテクチャと組織のアーキテクチャは同型化することを指摘している。

⑵ プロセスの最適化

もう一つの DSM 研究の流れは、開発プロセスの最適化である。典型的なのは、DSM 分析によって開発プロセスを構成するタスクの流れの改善を試みる研究である（Browning, 2002; Clarkson and Hamilton, 2000; Kusiak, 2002; Smith and Eppinger, 1997a; Smith and Eppinger, 1997b; Yassine and Braha, 2003; Yassine, et al. 2000; Yassine, et al. 2001）。

開発プロセス最適化の研究の一つのアプローチは、開発タスクのパーティショニングの改善を図ることである。タスク DSM により、まず現状の

表２　**DSM** を用いた製品開発研究

文献名	**DSM** タイプ	記述方式	分析手法	分析対象	研究概要
Becker, et al.（2000）	パラメータ、コンポーネント、タスク	2値	クラスタリング、パーティショニング	仮想事例	複雑なシステム設計における構成要素間のインターフェースを削減するための手法の開発。独自アルゴリズムにより、インタフェース最小化。
Browning, T. R（2001）	コンポーネント、チーム、タスク、パラメータ	2値	クラスタリング、パーティショニング、テアリング	自動車用空調システム、自動車エンジン開発組織、自動車設計、ロボットアーム / ハウジング	複雑な PD の分析手法としての DSM のレビュー。製品- 組織―プロセスのアーキテクチャの相互関係を整理。実用面での障害を明確化。
Browning, T. R（2002）	タスク（活動）	2値	パーティショニング（作業順序の順列化；手戻りの削減）	仮想例	製品開発の複雑なプロセスを記述するツールとして DSM を提示。プロセスの最適化（同期化、統合化）を図る方法論を提示。
Clarkson and Hamilton（2000）	タスク	2値	繰り返し作業の記述	ヘリコプターのローター設計	複雑な PD における最適な設計開発作業の道筋づくりのため、DSM によるタスクの依存関係の記述を提言。
Cronemyr, et al.（2001）	タスク	数値（タスク所要時間、繰り返し確率［0<P<1］）	所要時間シミュレーション	ガスタービン用ブレードの開発、航空機開発におけるサプライヤーとの協業関係	PD プロセスの改善がリードタイムに与える影響をシミュレーション分析。！ 繰り返しの削減、” タスクの遂行時間の短縮がリードタイム短縮にもたらすインパクトを定量的に計測。
Eppinger, S. D. and V. Salminen（2001）	タスク、チーム、コンポーネント	–	クラスタリング、パーティショニング	空調システム、エンジンコンパートメント、新型自動車エンジン、ジェットエンジン、電子部品	複雑な PD を製品―プロセス―組織の3つの視点から記述するアプローチの提示。
Kusiak（2002）	コンポーネント タスク（活動）	2値	クラスタリング パーティショニング	仮想例	モジュラリティを実現するための方法論の提示。製品モデル―（開発）プロセスモデル―資源モデルの相互依存性に着目。
McCord, K. R. and S. D. Eppinger（1993）	チーム	数値（情報交換：高、中、低）	クラスタリング	自動車用 V8 エンジン（GM）、ラップトップ PC	複雑な PD プロセスにおいて、プロジェクトの底辺にある技術的な構造を踏まえた上で、タスクの統合や調整を適切に行う方法を提示。
Pimmler, T. U. and S. D. Eppinger（1994）	コンポーネント	数値（必要+2、望ましい+1、中立0、望ましくない−1、有害−2）	クラスタリング	自動車用空調システム（Ford）	複雑な PD プロセスにおいて適切に要素分解し統合するための分析手法の開発。適切な製品アーキテクチャと開発組織構造の創出と選択の方法論を提示。
Sharman, D. M. and A. A. Yassine（2004）	コンポーネント（31のサブシステムで構成）	数値（4段階：0,1,2,3）	クラスタリング	産業用ガスタービン	複雑な製品アーキテクチャを記述し、その特性を掴むための方法論の確率。クラスタリングにおける恣意性の軽減、分析精度の向上。
Sharman, et al.（2002）	コンポーネント	2値→数値（高3、中2、低1、無0）	クラスタリング	産業用ガスタービン	クラスタリング・アルゴリズムの改善。モジュールの境界、経路依存性、ディメンショナリティを考慮に入れることで自動クラスタリングアルゴリズムの弱点を補完。
Smith, R. P. and S. D. Eppinger（1997a）	タスク	数値（タスク遂行所要時間、タスクやり直し確率：0.5, 0.25, 0.05）	並び替え（順列化）所要時間シミュレーション	自動車ブレーキシステム	複雑な PD における同時並行的な作業の繰り返しを記述し、作業順序の見直しなどにより、リードタイムの短縮を図るアルゴリズムを開発。
Smith, R. P. and S. D. Eppinger（1997b）	タスク	数値（タスク遂行所要時間、タスクやり直し確率）	並び替え（順列化）所要時間シミュレーション	仮想事例（Steward [1981] の電気自動車開発の仮想プロジェクト）	複雑な PD における逐次的な作業の繰り返しを記述し、作業順序の見直しなどにより、リードタイムの短縮を図るアルゴリズムを開発。
Sosa, et al.（2003）	コンポーネント、チーム	・数値（必要+2、望ましい+1、中立0、望ましくない−1、害がある-2） ・2値：開発チーム間の調整の重要性及び頻度	クラスタリング、パーティショニング（順列化）	航空機エンジンおよびその開発組織	開発製品のモジュラーな部分と統合的な部分を峻別したうえ、適切な開発チームの組織化とチーム間のインターフェースを管理。
Sosa, et al.（2004）	コンポーネント、チーム	デザイン：数値（-2、-1、0、+1、+2） 関係の性質（空間、構造、物質、エネルギ、情報） 組織：数値（6点）→2値へ変換	–	航空機エンジンの詳細設計フェーズ	製品アーキテクチャと開発組織の構造の適合性の分析。設計上・組織上の境界（boundary）が不適合（misalignment）の発生に有意に影響。
Yassine, A. and D. Braha（2003）	タスク、チーム	2値	パーティショニング（順列化）、クラスタリング	通信産業のデータサービス開発、自動車シリンダーブロック開発、自動車エンジン開発、自動車外観設計	DSM により、PD におけるタスクの繰り返し、オーバーラップ、分解と統合、問題解決の収斂について分析。適切なタスク、チームのグループ化や順列化のポイントを提示。
Yassine, et al.（2000）	タスク	数値	パーティショニング（順列化）、シミュレーション、DRFT	自動車（フード［ボンネット］設計、シートベルト設計）	複雑な PD における繰り返しタスクの削減。繰り返しを含むタスクブロックに、簡単な検証タスクを挿入することにより、DSM の単純なパーティショニングよりも大幅に繰り返しを削減。
Yassine, et al.（2001）	タスク	数値（やり直し確率：1～9）	パーティショニング（順列化）、シミュレーション	自動車（フード［ボンネット］設計）	複雑な PD におけるタスクの繰り返し確率の分析
Yassine, et al.（2003a）	チーム	数値（部品間の依存性：強・中・弱・無→0.3, 0.2, 0.1, 0）	所要工数シミュレーション	自動車（内外装）外観デザイン	複雑な PD における問題発生の記述と分析。問題解決の収束性を分析。
Yassine, et al.（2003b）	タスク、コンポーネント、パラメータ、顧客ニーズ要素	2値→数値	–	自動車用安全ベルト、自動車シャシー	PD プロセスにおける要素間（タスク、パラメータ、コンポーネントなど）の依存性を多面的に記述する手法（CM：connectivity matrix）の開発。
Yu, et al.（2003）	コンポーネント	2値 / 数値	クラスタリング	産業用ガスタービン	複雑な製品システムをより高い精度で最適なクラスタリングするアルゴリズムの開発

（出所）筆者作成

開発プロセスを記述し、次にパーティショニングによってタスクの順列化を行うのが一般的な分析手順である。タスク DSM の対角線より上にある印 X を対角線より下に移動させることにより、タスクのやり直しを低減するのである。

プロセス最適化の研究は、研究者の研究関心や専門分野などの違いによって研究アプローチにいくつかのバラエティがある。

まず、パーティショニングのアルゴリズム開発に重点を置く研究がある（Becker, et al. 2000; Smith and Eppinger, 1997a; Yu, et al. 2003）。例えば、Smith and Eppinger（1997b）は、複雑な製品開発における逐次的なタスクの反復をタスク DSM により記述し、作業順序の見直しなどにより開発リードタイムを短縮するアルゴリズムを開発している。仮想的開発プロジェクトの分析を通じて、あるタスクの実行は強い依存性のある別のタスクの後に回すこと、所要時間の長いタスクはできるだけ後にもっていくなどすることで、タスクの繰り返し確率の低減や繰り返しが生じた場合の所要時間の最小化が図れることを示した。

次に、タスクの繰り返し問題に対してより組織的な解決策を探る研究がある（Browning, 2002; Yassine, et al. 2000; Yassine and Braha, 2003）。現実の製品開発プロセスにおいては、パーティショニングによってすべての印 X を対角線より下に移動させることは不可能である。タスクの試行錯誤的な繰り返しは不可避的に生じる現象であり、タスクの繰り返しを生産的に処理する組織的な対応が必要である（Eppinger, 2001; Yassine and Braha, 2003）。

そうした組織的な解決策の一つに、ティアリング（tearing）という手法がある。タスク DSM では、対角線より上にある印 X がフィードバック・ループを表す。対角線よりも上に印 X が密集しているような場合、そこには多重的なフィードバック・ループが存在する。ティアリングは、大きなフィードバック・ループが存在するプロセスにおいて、そのフィードバックの核となっている依存関係を抽出し、その依存関係を除去することにより、フィードバック・ループを解消、もしくは経済的に管理可能なレベルに抑制する手法である（Yassine, 2004）。

どの依存関係をいかにティアリングするかは、DSM 分析だけからは導き出すことはできない。組織的な解決策とセットで考える必要がある。２つのタスクの間にフィードバック・ループが存在する場合、前工程のタスクは、後工程のタスクからのフィードバック内容を推測して、実行される。この推測の精度が高ければ、タスクのやり直しの確率や度合いが低減される。コンカレント・エンジニアリングやデザインレビューなどは、上流と下流のタスクの間で情報共有するための組織的な方法といえる。Yassine and Braha（2003）は、DSM 分析を応用して、開発プロセスにおける効果的なコンカレント・エンジニアリングの方法を提示している。また、Yassine, et al.（2000）は、開発プロセスの中に小さな検証タスクを挿入することにより、間違った仮定に基づいてタスクが進行する確率を低減させることで、タスクの繰り返しを抑制する方法を分析している。

最後に、プロセス最適化の成果をシミュレーションにより定量的に計測する研究アプローチがある（Cronemyr, et al. 2001; Smith and Eppinger, 1997a; Smith and Eppinger, 1997b; Yassine, et al. 2000; Yassine, et al. 2001; Yassine, et al. 2003）。各タスクの平均所要時間と依存関係にあるタスクのやり直し確率（０＜ P ＜１）を計測し、このデータをもとにモンテカルロ法などのシミュレーション手法を活用して、開発リードタイムや所要工数などを確率的に導き出すのである。前述のアルゴリズム研究や組織分析の多くは、シミュレーション分析を併用して、タスク DSM の改善効果の定量的な検証を行っている。

４．DSM の限界点

DSM は、製品開発において生じる複雑性を記述し分析する強力な分析ツールである。DSM は要素間の複雑な依存関係を解きほぐし、行列形式で記述することができる。記述する情報の簡潔性、一覧性、操作性に DSM の利点がある。

しかし、DSM は万能な分析手法ではない。DSM の簡潔性や操作性は、製品開発に関するある種の情報を捨象することによって得られる利点であり、そのことは同時にこの分析手法の限界にもなる。以下では、DSM 分析の限界点や留意点について若干の考察を行う。

⑴ 依存関係の予見可能性について

DSM 分析は、要素間の依存関係がある程度、特定可能あるいは予見可能であることを前提としている。基本的に、DSM の作成は既存の開発製品や開発プロジェクトをベースとして、構成要素を定義したり要素間の依存関係を特定したりする。したがって、技術的に安定し、定期的にモデルチェンジを繰り返す自動車のような製品の開発においては、部品間あるいはタスク間の依存関係は相対的に安定しており、新規開発製品の DSM を作成することは比較的容易である。

しかし、製品開発の新規性が高まるほど、部品間にどのような依存関係があるのか、あるいはタスク間でどのような繰り返しが生じるのかを特定することは難しくなる。製品開発の新規性は、新規導入の要素の多さおよび製品アーキテクチャの変化幅の大きさによって規定される。

新規に導入される技術や部品が多くなるほど、要素間に存在する依存関係を特定していくのが難しくなっていく。個々の要素は、しばしば 2 つ以上の依存関係を持つために、新規の要素の導入は、より多くの未知の依存関係を生み出すからである。

この時、新規の要素がもたらす未知の依存関係の影響は、製品アーキテクチャの特性によって異なると考えられる。開発製品の製品アーキテクチャがインテグラル型の場合、新規要素の導入に伴う未知の依存関係は、他の多くの要素に波及的に影響を与える可能性がある。一方、モジュラー型の場合、新規要素が特定のモジュールの中で採用された場合、未知の依存性があったとしてもその影響はモジュール内に止められる。しかし、新規要素がインターフェースに関わるものである場合、未知の依存関係の存在は製品システム全体に影響を与えることになる。

また、開発製品の大部分が既知の要素により構成される場合でも、要素間の関係性に多大な不確実性が生じる可能性がある。製品アーキテクチャが大幅に変更された場合である。大幅に変更された製品アーキテクチャのもとで初めて DSM を作成するような場合には、要素の間に多くの未知の依存関係が隠されることになろう。

こうした DSM の特徴から、DSM の活用においては、次の 2 点に注意する必要がある。

第一に、新規性が高く要素間の依存関係を十分に特定できないような場合においては、DSM の活用は限定的な用途に限る必要がある。新規の製品開発においても、全く新規に導入される技術や部品は限られており、その他の大部分の要素は既存のものが使われるのが一般的である。したがって、DSM の作成を通じて、既存の分かっている部分と新規の分かっていない部分とを峻別することにより、未知の依存関係が生じるかもしれない管理ポイントを割り出す目安とすることはできる。新規性の高い製品開発においても、このように DSM をある種の参照情報と活用することは有効である。

しかし第二に、新規性の高い製品開発の場合、作成される DSM はあくまでも仮説的なものである。どの部分にどのような依存関係があるかという仮説自体は、その組織やエンジニアの経験や理論的推論、予見力などによって導き出される。仮説的な依存関係や未知の依存関係があまりにも多くなると、クラスタリングやパーティショニングなどによって依存関係の最適化を行っても、現実の使用に耐える分析とはならない可能性が高い。DSM は、存在が明らかになった依存関係が製品設計や開発プロセスにどのような影響をもたらすかを分析することはできるが、依存関係自体を特定化することはできないのである。

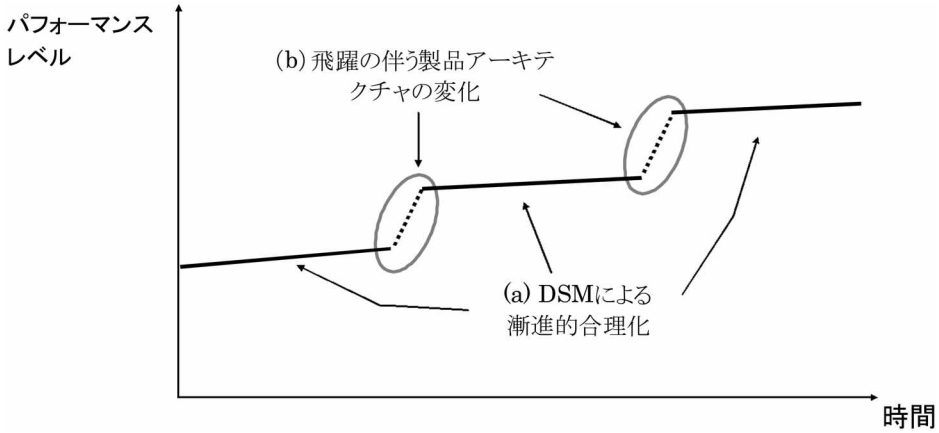
⑵ スタティックな分析アプローチ

基本的に、DSM はスタティックな分析アプローチである。現状のシステム構成の記述をベースとして、クラスタリングやパーティショニングといった手法により、構造やプロセスの改善を図るのが一般的な分析パターンである。

例えば、コンポーネント DSM を用いた製品モジュール化の分析では、現状の部品構成とそれらの依存関係を出発点として、それをクラスタリングすることで、モジュールの形成を図る。実現すべきパフォーマンスが一定である場合には、モジュール内での要素間の依存関係は密にし、モジュール間の依存関係は最小化することで、開発製品のコスト、品質、利便性などの改善、開発生産性の向上などの成果を漸進的に実現することが可能となる（図11の（a）の部分）。

しかし、実現可能なパフォーマンスのレベル自体を高めたい場合、既存のシステム構成の最適化だけでは十分でない（図11の（b）の部分）。新規

図11　**DSM** による改善とパフォーマンスとの関係



（出所）筆者作成

性の高い部品やサブシステムの導入、あるいは新しい観点からのモジュールの切り分け（クラスタリング）が必要になってくる。このような場合、既存のシステム構成に基づいた DSM の分析からは、新たなパフォーマンス・レベルに対応した製品アーキテクチャを導出することはできない。DSM とは別のロジックによる分析が必要である。要素の積み上げや改善だけでは、システム全体の最適化を必ずしも実現できないからである（柴田他、2002）。

DSM は、現状のシステムを記述し、問題のある箇所を特定するには適した手法である。クラスタリングやパーティショニングなどにより、現状のシステム構成の下で改善を積み重ねることはできる。そして、その改善が一定の閾値まで達した時、新たな製品アーキテクチャへと移行する契機となる可能性はある。しかし、DSM 自体には新たなアーキテクチャを創出するロジックは含まれておらず、むしろ漸進的なアーキテクチャの変化を強化する。基本的に DSM は、既存の要素の並び替えを中心にして構造やプロセスの最適化を図る手法であるからである。DSM によって捉えられた既存のアーキテクチャの課題や限界を乗り越えるには、DSM とは別のロジックからのアプローチが必要である[6)]。

(3) 情報フローと知識蓄積について

DSM 研究における主要な命題の一つに、製品アーキテクチャと組織構造の同型化がある。すなわち、製品開発の生産性を高めるためには、組織構造の DSM は、コンポーネント DSM でとらえられる部品やサブシステム間の依存関係と対応してなければならないということである。この命題は、開発チーム間のコミュニケーションの効率化やタスク間の情報フローの改善の面では、効果の高い方策である。

しかし、知識の蓄積面を考えると、製品アーキテクチャと同型化した組織編制が常に効果的とは限らない。コンポーネント DSM が表示するのは、あくまでも要素間の依存関係であり、要素自体の複雑性や重要性に関する情報は捨象されている[7)]。要素間の依存関係を最適化する観点からは、例えば、開発プロジェクトごとに担当チーム間のコミュニケーションを最適化するように、組織をその都度組み直すことは有効かもしれない。しかし、ある部品やサブシステムの開発に非常に深い専門性や経験蓄積を必要とするような場合には、プロジェクト・ベースのコミュニケーション効率をある程度犠牲にしても専門性や経験蓄積を重視した

6）例えば、柴田他（2002）は、「分断による学習」という概念を提起し、新たな製品アーキテクチャの創出には、実際に製品システムを要素へと分割していく設計行為の繰り返しを通じてシステムレベルの知識を組織的に積み上げていく必要があることを指摘している。

7）タスク DSM では、タスク実行の所要時間や工数の形で、個々の要素の特性が記述されることがあるが、コンポーネント DSM ではコンポーネント自体の特性について記述を与えている既存研究は、本稿でレビューした範囲では見つけられなかった。

組織編制が効果的になる可能性がある。技術の成熟性や信頼性が求められる自動車メーカーの多くが機能別組織をベースとした組織構造を採用しているのは、組織の知識蓄積を重視するという背景があるからかもしれない。

既存の DSM 研究は、やや技術決定論的な傾向があり、組織の論理は必ずしも重視されていない。組織の論理、例えば組織学習やモチベーションの観点から考えると、むしろ製品アーキテクチャの方を組織構造に適した形に変換していくことも一つの方法である。このことは DSM 自体の方法論上の問題というよりは、DSM の活用上の問題といえる。いずれにしても、今後の DSM 研究においては、技術論と組織論との間でよりバランスのとれた分析フレームワークの設計が重要である。

5．むすび

製品開発を複雑にする要因はさまざまである。例えば、顧客ニーズの複雑性や多義性、部品やサブシステム自体の技術的複雑性、個々の開発タスクの複雑性などの要因があげられる。現代の製品開発をとりわけ複雑にしているのは、これらの要因が相互に作用し合っていることである。部品と部品、タスクとタスクとの間で相互的な依存関係があり、ある一つの部品やタスクにおける変更が他の部分へと波状的に影響を与えるのである。

DSM は、こうした要素間の依存関係を網羅的かつ簡潔に記述する分析ツールである。分析対象に応じて、コンポーネント DSM、チーム DSM、タスク DSM、パラメータ DSM などがあり、構造の最適化には主にクラスタリング、プロセスの最適化にはパーティショニングといった分析手法が用いられる。製品開発の他の手法と比較した DSM の利点は、記述する情報の網羅性、簡潔性、一覧性、操作性の良さにある。

ただし、DSM は分析手法の一つに過ぎず、製品開発のすべてを分析できるわけではない。DSM には、要素間の未知の依存関係に対する脆弱性、スタティックな分析アプローチ、組織の知識蓄積に関する分析の弱さといった限界点がある。

しかし、こうした制約にもかかわらず、DSM は非常に強力な分析ツールであることに変わりはない。とりわけ、製品アーキテクチャ分析において強みを発揮する。従来の研究では、製品アーキテクチャのモジュラー度やインテグラル度の判断は、研究者自身の定性的な判断に依存する部分が大きかった。DSM を活用することにより、より客観的で反証可能な分析を行うことが可能になり、製品アーキテクチャ分析の精度の向上に大きく貢献できると考えられる。

謝　辞

論文の執筆に当っては、二人の匿名のレフェリーから本稿に対し有益なコメントを頂いた。記して感謝申し上げたい。

なお、本研究は平成17年度科学研究費補助金（若手研究（B））［課題番号：17730235］の研究成果の一部である。

参考文献

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

［2］ Becker, O., J. Ben-Asher, and I. Ackerman (2000). “A model for system interface reduction using N2 charts,” *Systems Engineering*, 3, 27–37.

［3］ Browning, T. R. (1998). “Integrative mochanisms for multiteam integration: Findings from five case studies,” *Systems Engineering*, 1, 95–112.

［4］ Browning, T. R. (2001). “Applying the design structure matrix to system decomposition and integration problems: A review and new direction,” *IEEE Transactions on Engineering Management*, 48 (3), 292–306.

［5］ Browning, T. R. (2002). “Process integration using the design structure matrix,” *Systems Engineering*, 5 (3), 180–193.

［6］ Clarkson, P. J. and J. R. Hamilton (2000). “Signposting', A parameter-driven task-based model of the design process,” *Research in Engineerign Design*, 12, 18–38.

［7］ Cronemyr, P., A. O. Ronnback and S. D. Eppinger (2001). “A decision support tool for predicting the impact of development process improvements,” *Journal of Engineering Design*, 12 (3), 177–199.

［8］ Eppinger, S. D. (2001). “Innovation at the speed of information,” *Harvard Business Review*, 79 (1), 149–158.

［9］ Eppinger, S. D. and V. Salminen (2001). “Patterns of product development interactions,” Proceedings of International Conference on Engineering Design, Glasgow, Aug. 21–23, 2001.

［10］ Henderson, R. M. and K. B. Clark (1990). “Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms,” *Administrative Science Quarterly*, 35, 9–30.
［11］ Kusiak, A. (2002). “Integrated product and process design: A modularity perspective,” *Journal of Engineering design*, 13 (3), 223–231.
［12］ McCord, K. R. and S. D. Eppinger (1993). “Managing the integration problem in concurrent engineering,” *MIT, Sloan School of Management Working Paper* #3594
［13］ Pimmler, T. U. and S. D. Eppinger (1994). “Integration analysis of product decompositions,” ASME Design Theory and Methodology Conference, Minneapolis, MN, September 1994.
［14］ Sharman, D. M. and A. A. Yassine (2004). “Characterizing complex product architectures,” *Systems Engineering*, 7 (1), 35–60.
［15］ Sharman, D. M., A. A. Yassine, and P. Carlile (2002). “Characterizing modular architectures, Proceedings of DETC ’02: ASME 2002 International Design Engineering Technical Conferences and Design Theory & Methodology Conference, Montreal, Canada, Sep. 29–Oct. 2, 2002.
［16］ Smith, R. P. and S. D. Eppinger (1997a). “Identifying controlling features of engineering design iteration,” *Management Science*, 43 (3), 276–293.
［17］ Smith, R. P. and S. D. Eppinger (1997b). “A predictive model of sequential iteration in engineering design,” *Management Science*, 43 (8), 1104–1120.
［18］ Sosa, M. E., S. D. Eppinger and C. M. Rowles (2003). “Identifying modular and integrative systems and their impact on design team interactions,” *Journal of Mechanical Design*, 125, 240–252.
［19］ Sosa, M. E., S. D. Eppinger and C. M. Rowles (2004). “The misalignment of product architecture and organizational structure in complex product development,” *Management Science*, 50 (12), 1674–1689.
［20］ Steward, D. V. (1981). *Systems Analysis and Management: Structure, Strategy and Design*, Petrocelli Books, Inc.
［21］ Ulrich, K. (1995). “The role of product architecture in the manufacturing firm,” *Research Policy*, 24, 419–440.
［22］ Ulrich, K. and S. D. Eppinger (2000). *Product Design and Development (2nd Ed.)*, The McGraw-Hill.
［23］ Von Hippel, E. (1990). “Task partitioning: An innovation process variable,” *Research Policy*, 19, 407–418.
［24］ Yassine, A. A. (2004). “An introduction to modeling and analyzing complex product development processes using the design structre matrix (DSM) method,” *Quaderni Di Management* (Italian Management Review), 9. (English translation ver.)
［25］ Yassine, A. and D. Braha (2003). “Complex concurrent engineering and the design structure matrix method.” *Concurrent Engineering*, 11 (3).
［26］ Yassine, A., D.E. Whitney, J. Lavine, and T. Zambito (2000). “Do-it-Right-First-Time (DRFT) approach to design structure matrix (DSM) restructuring,” Proceedings of DETC ’00: ASME 2000 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Baltimore, Maryland, Sep. 10–13, 2000.
［27］ Yassine, A., D.E. Whitney, J. Lavine, and T. Zambito (2001). “Assessment of rework probabilities for simulating product development processes using the design structrue matrix (DSM),” Proceedings of DETC ’01: ASME 2001 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Pittsburgh, Pennsylvania, Sep. 9–12, 2001.
［28］ Yassine, A., D. Whitney, S. Daleiden and J. Lavine (2003). “Connectivity maps: Modeling and analysing relationships in product development processes,” *Journal of Engineering design*, 14 (3), 377–394.
［29］ Yassine, A., N. Joglekar, D. Braha, S. Eppinger, and D. Whitney (2003). “Information hiding in product development: The design churn effect,” *Research in Engineerign Design*, 14, 145–161.
［30］ Yu, T., A. A. Yassine, and D. E. Goldberg (2003). “A genetic algorithm for developing modular product architectures,” Proceedings of DETC ’03: ASME 2003 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Chicago, Illinois USA, Sep. 2–6, 2003.
［31］ 足立 泰（2004）「開発プロセスの改革手法『DSM』第１回、第２回」『日経デジタルエンジニアリング』2004年２月号、３月号
［32］ 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
［33］ 藤本隆宏（2001）『生産マネジメント入門（Ⅱ）』日本経済新聞社
［34］ 藤本隆宏・武石 彰・青島矢一編（2001）『ビジネス・アーキテクチャ』有斐閣
［35］ 目代武史（2005）「広島地域における自動車部品モジュール化の動向と地場部品メーカーの対応」『地域経済研究』16号

A Review on the Design Structure Matrix as an Analytical Tool for Product Development Management

MOKUDAI Takefumi*
Research Assistant, Center for Research on Regional Economic Systems,
Graduate School of Social Sciences, Hiroshima University

Abstract

This article reviews fundamental concepts and analytical techniques of *design structure matrix (DSM)* as well as recent development of DSM studies. The DSM is a matrix representation of relationships between components of a complex system, such as products, development organizations and processes. Depending on targets of analysis, there are four basic types of DSM: Component-based DSM, Team-based DSM, Task-based DSM, and Parameter-based DSM.

There are two streams of recent DSM studies: 1) optimization of product design and organizational structure and 2) optimization of development process. The former employs *clustering*, whereas the latter does *partitioning*.

The advantages of the DSMs, compared with other analysis tools, are its conciseness, completeness, and operationality. However, DSMs are not a panacea. The article also discusses some constraints of the DSMs, such as its vulnerability to unknown dependency between components, a static nature of the analytical approach, and relatively weak implications for organizational learning.

Key words: design structure matrix, product development, product architecture

* Corresponding author: mokudai@hiroshima-u.ac.jp