企業情報システム開発における
行列を用いた情報連携表記方法（MDM）

The Matrix Description Method of Information Relationship for Developing Enterprise Information Systems (MDM)

2018 年 3 月

齊藤　哲
Tetsu SAITO

企業情報システム開発における
行列を用いた情報連携表記方法（MDM）

The Matrix Description Method of Information Relationship for Developing Enterprise Information Systems (MDM)

2018 年 3 月

早稲田大学大学院 創造理工学研究科
経営デザイン専攻　生産・流通プロセス改革研究

齊藤　哲
Tetsu SAITO

目次

第 1 章　序論

1.1 研究の背景

本論文のテーマは,「企業情報システム開発における行列を用いた情報連携表記方法」である．本論文の研究対象は，企業における業務を支援する情報システムの開発時に用いられる表記方法である．

企業内には，様々な業務を支援する複数の情報システムが存在している．これらの情報システムは，互いに情報連携している．このように，企業における業務を支援する情報システムは，複数の情報システムが互いに情報連携しながら，全体を構成する．したがって，全体を構成する情報システムを部分的に改修・更新する場合，他の情報システムとの情報連携の追加，削除，つなぎ換えの検討が必要となる．このような検討が必要となる具体例には，社外の情報システムとの連携の追加と情報連携のつなぎ換え，統合業務管理パッケージ(ERP: Enterprise Resource Plannning)導入による情報システムの置き換え，企業の経営統合に伴う情報システムの統合などがある[1].

他の情報システムとの情報連携の追加，削除，つなぎ換えの検討が不十分な場合，情報連携の抜けもれ，重複，つなぎ間違いといったエラーがおきる．このようなエラーは情報システム開発の最終段階で，ユーザが情報システムを総合的にテストする際に発見されることが多く，大きな手戻りになる[2][3]. このようなエラーがおきる原因の 1 つに，情報システム開発に用いる情報連携の表記方法のわかりづらさがある．

情報システム開発に用いる表記方法は，情報システムの特徴をとらえる必要がある．情報システムの基本形は，コンピュータを使った情報処理を機能とし，情報処理に必要なデータの入力と出力を与える IPO(Input Process Output)の形をとる．そして，情報システムは，こうした機能がデータを介して，多数連結されて，全体を構成する[4][5]. また，情報システムは，コンピュータ処理が可能なプログラムあるいはモジュールと呼ばれるレベルまで分解・展開されるという階層的構造をもつ[6]. このような IPO の形をとるという情報システムの特徴を表現するため，情報システム開発に用いられる設計書には，情報処理の機能を円や楕円で表し，入力と出力にあたるデータは機能間の情報連携として矢印や線で表す

フローチャート型の表記方法が用いられることが多い[7][8]. 一般的に用いられるフローチャート型の表記方法のひとつに，データフローダイアグラム(DFD：Data Flow Diagram)がある[9][10]．また，階層的構造をもつという情報システムの特徴を表現するため，情報システム開発に用いられる設計書には，階層を使って機能をサブ機能に分解・展開した結果を記述する表記方法が用いられる．DFD は階層を使って，機能の分解・展開を表現することができる．この階層を使った DFD はストラクチャード DFD(SDF：Structured DFD)と呼ばれる[11][12]. DFD のようなフローチャート型表記方法は, 全体を構成する情報システムが部分的に改修・更新される場合にも用いられることが多い．このような場合，DFD で記述された設計書に対し, 情報システムの部分的な改修･更新に伴って発生する，機能の追加・削除や情報連携の追加・削除，つなぎ換えを記述する必要がある．

この際，DFD を用いて機能や情報連携を記述しようとすると，

(1) 機能を記述する位置が決まっていないため，情報連携をたどるための線と方向を示す矢印が必要となる．そのため，機能を記述する位置によっては，矢印線が交差して，情報連携がわかりにくくなる．このわかりにくさを軽減するため，1 枚のシートに記述する機能数，情報連携数を制限する．

(2) 機能をサブ機能に分解・展開することにより，記述する機能数や情報連携数が増えていくため．上記(1)と同様，矢印線が交差して，情報連携がわかりにくくなる. また, 1 枚のシートに記述する機能数, 情報連携数を制限するため，分解・展開後のサブ機能は元の機能と別シートに記述する．そうすると，分解・展開後のサブ機能と元の機能との階層関係がわかりにくくなる．また，サブ機能間の情報連携がたどりにくくなる．

という表記方法上の 2 つの問題がある．なお，この後，サブ機能に分解・展開することを，本論文では単に分解すると表現する．

1.2 研究の目的

本研究では，情報システム開発に広く用いられているフローチャート型の表記方法の DFD を，全体を構成する情報システムの部分的な改修・更新に活用するときに発生する表記方法上の問題と課題を明らかにする．そして，この課題を解決するために，行列を用いた表記方法 MDM(Matrix Description Method)を提案し，その有効性を確認することを目的とする[13][14][15][16].

情報システムの改修・更新に DFD を活用するときに発生する表記方法上の課題は，1.1 節で示した 2 つの問題から，次の 2 点に集約できる．

(1) 連携図を記述する際の課題

具体的には，機能の配置が決まっており，情報連携をたどるための線や矢印を必要としない収容能力の高い表記方法の実現である．

(2) 機能を分解した結果を記述する際の課題

具体的には，機能の分解後も階層の上下関係が把握でき，同時に分解後の下位機能同士の情報連携が一覧できる表記方法の実現である．

本論文で提案する行列を用いた表記方法 MDM は，機能を二次元正方行列の対角上に配置し，機能と機能をつなぐ情報連携は 2 つの機能の行と列の交点に位置するように配置する．この表記により，MDM は機能を配置する位置が決まっており，情報連携をたどるための線や矢印を必要としない収容能力の高い表記方法となる．これにより，連携図を記述する際の課題が解決できる．

また，MDM は階層を使って分解前の機能と分解後の機能との上下関係をフラットに表現する．この表記により，MDM は機能の分解後も階層関係が把握でき，同時に分解後の下位機能同士の情報連携が一覧できる表記方法となる．これにより，機能を分解した結果を記述する際の課題が解決できる．

MDM の有効性の確認は，次の 3 段階で実施する．

まず，DFD の問題指摘に用いた例題で発生している問題の解決の確認である．DFD の問題を指摘する際に用いた連携図を，同じように MDM を使って作成し，指摘した問題が解決していることを確認する．次に，初学者を対象に DFD と MDM を用いて連携図を作成する実験を実施し，MDM の効果を確認する．実験の評価は，機能，情報連携の正答率と連携図の作成時間の比較による．この実験により，MDM が連携図を記述する際の課題に有効であることを確認する．最後に，情報システム開発経験者を対象に DFD と MDM を用いて機能を分解した連携図を作成する実験を実施し，MDM の効果を確認する．実験の評価は，あらかじめ決めた回答時間内に回答できた情報連携の正答率と連携図を記述したシート枚数の比較による．この実験により，MDM が機能を分解した結果を記述する際の課題に有効であることを確認する．

1.3 本論文の構成

本論文の構成は以下のとおりである.

第 1 章では，研究の背景を述べ，本研究の目的を示す.

第 2 章では，全体を構成する情報システムを部分的に改修・更新する際に，フローチャート型表記方法の DFD を用いた例を示し，表記方法上の問題を指摘する．問題指摘に使用する例題は，すでに複数の情報システムが存在する中で，新たな情報システムを追加開発する事例である.

第 3 章では，第 2 章で明らかになった問題を分析し，連携図を読みやすく，理解しやすいように記述するという観点で,(1)連携図を記述する際の課題と(2)機能を分解した結果を記述する際の課題を設定する.

第 4 章では，第 3 章で設定した課題を解決しようとする表記方法として，線や矢印を使用せずに機能間の関係を示すことができる行列を用いた表記方法を中心に，従来手法を確認する.

第 5 章では，第 3 章で提起した課題を解決する表記方法として，行列を用いた表記方法 MDM を提案する．MDM は機能を二次元正方行列の対角上に配置し，情報連携は 2 つの機能の行と列の交点に位置するように配置する．また，MDM は階層を使って分解前の機能と分解後の機能との上下関係をフラットに表現する.

第 6 章では，MDM の有効性を 3 段階で確認する．まず，DFD の問題指摘に用いた例題で発生している問題の解決の確認である．次に，初学者を対象に DFD と MDM を用いて連携図を作成する実験を実施し，MDM の効果を確認する．最後に，情報システム開発経験者を対象に DFD と MDM を用いて機能を分解した連携図を作成する実験を実施し，MDM の効果を確認する.

第 7 章では，情報システムを改修・更新するプロジェクトに MDM を適用した 3 社の事例を示し，実際のプロジェクトに適用可能であることを確認する．取り上げた A 社は，社外の情報システムとの連携の追加と情報連携のつなぎ換えを行う事例である．B 社は，統合業務管理パッケージ導入による情報システムの置き換えを行う事例である，C 社は，企業の経営統合に伴う情報システムの統合を行う事例である

第 8 章では，本研究で得られた成果を要約し，今後の課題について述べる.

第 2 章　フローチャート型表記方法の問題

2.1 フローチャート型表記方法の概要

フローチャート型表記方法は，機能を円や楕円で表す．また，機能間の情報連携は機能と機能をつなぐ線で表し，機能への入力・出力を示す情報連携の方向は矢印で表す．ここでは，これらの特徴をもつフローチャート型表記方法として，広く情報システム開発に用いられている機能情報関連図(DFD：Data Flow Diagram)を取り上げる．DFD は 1978 年に Tom DeMarco が発表した“Structured Analysis and System Specification”が起源とされる[9]．　DFD は，表 2.1 に示すように，4 つ基本要素から構成され，それぞれ表記上のシンボルが使われている．名前が付いた矢印線はデータの経路を示す．円や楕円はデータの変換を表す．直線はファイルあるいはデータベースを表す．四角形はデータの発信者あるいは受信者を表す．

表 2.1 DFD の凡例

No.	表記	記号	表記の説明
1	データフロー名 →	データフロー	要素間のインターフェースを表す.
2	処理名	処理	入力データフローから出力データフローへの変換を表す.
3	ファイル名	ファイル	一時的なデータの貯蔵庫を表す.
4	データ発生源名または行先名	データの発生源または行先	対象システム外にあり，データの発信地あるいは受信地を表す.

この DFD を用いて IPO の基本形を表現した例と，表 2.1 に示した 4 つの要素をすべて使用している連携図の例を図 2.1 に示す．

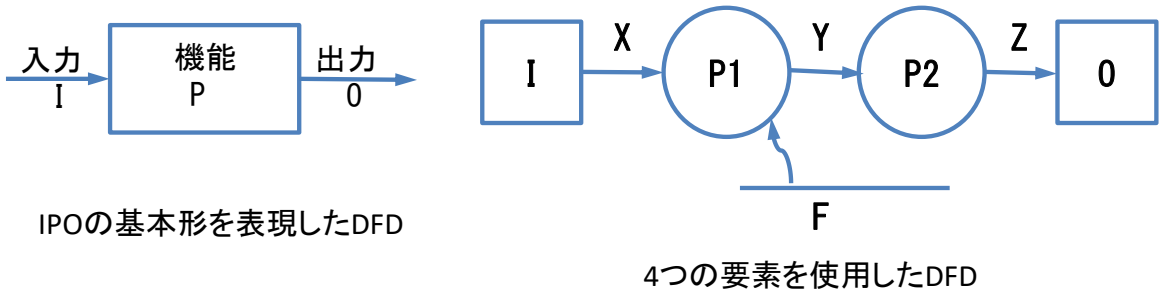


図 2.1　DFD の一般的な表現

DFD は階層を用いて機能を分解することができる．DFD を使って記述した機能を分解し，そのひとつひとつを記述していくと，階層化した一連の DFD ができあがる．図 2.2 は，階層を用いて A の機能を A1，A2，A3 のサブ機能に分解した DFD のイメージである．階層を用いて分解した DFD は SDF(Structured DFD)と呼ばれる．

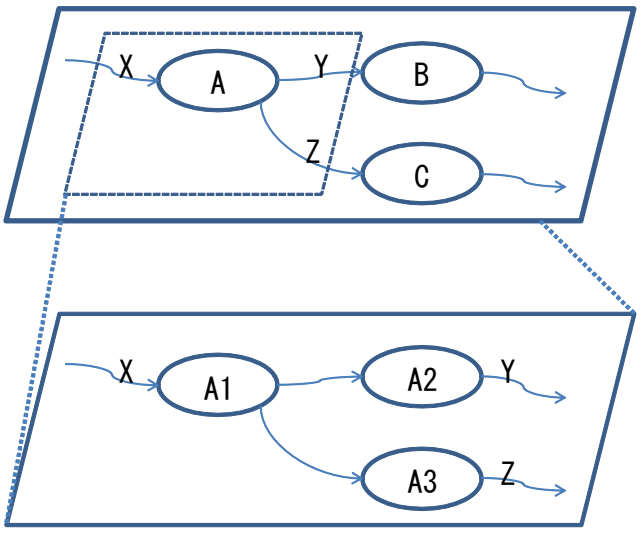


図 2.2　階層を用いた機能分解の例(DFD)

なお，フローチャート型の特徴をもつ表記方法には，DFD の他，IDEF 0 (Integration DEFinition 0)[17][18][19], BPMN(Business Process Modeling and Notation)[20]，業務流れ図(WFA: Work Flow Architecture)[21]などがある．

2.2 DFD を用いた情報システム開発の例と問題

2.2.1 問題指摘に使用する例題

複数の情報システムが存在する中で，情報システムを追加開発する際に，フローチャート型表記方法の DFD を用いた例を示し，表記方法上の問題を指摘する．問題指摘に使用する例題は，「あるレストランチェーン A 社において，食材発注に電子取引を導入するための情報システム改修」である．

すでに複数の情報システムが存在する中で，情報システムを追加開発する際の進め方を図 2.3 に示す．

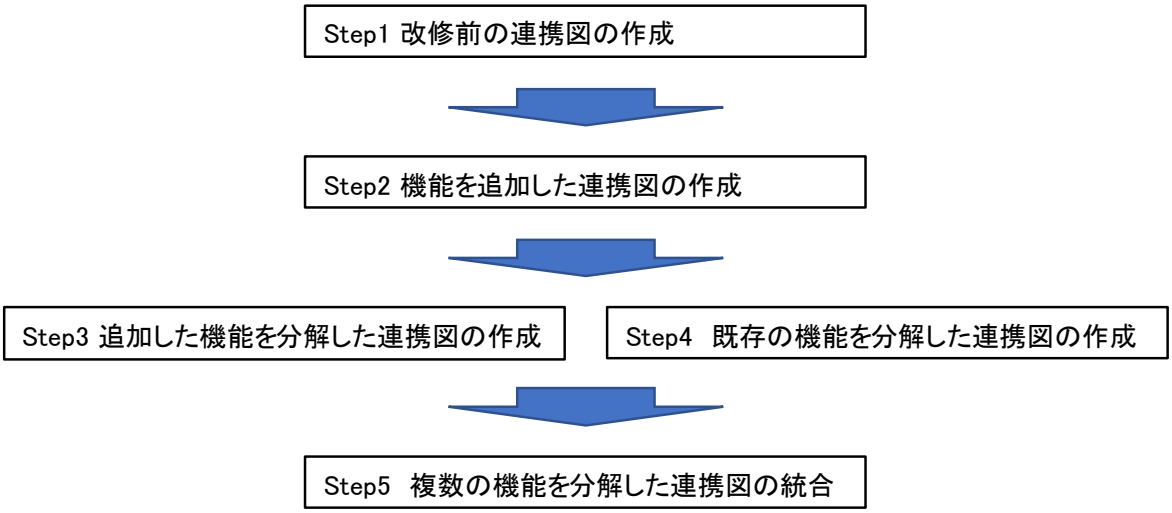


図 2.3　情報システムを追加開発する進め方

最初に，Step 1 では，情報システム改修前の連携図を作成する．

次に，Step 2 では，Step 1 で作成した連携図を参考にしながら，新たな機能を追加した連携図を作成する．

さらに，Step 2 で作成した連携図に記述された機能の分解とそれに伴う情報連携のつなぎ換えを行う．Step 3 では Step 2 で追加した機能を分解した連携図を作成する．また，Step 4 では既存の機能を分解した連携図を作成する．

最後に，Step 5 では，それまでに作成した複数の連携図を統合し，機能や情報連携の抜けもれ，重複，つなぎ換えによる矢印線のつなぎ間違いなどのエラーがないことを検証する．

2.2.2 改修前の連携図(DFD)の作成

最初に，図 2.3 の「Step1 改修前の連携図の作成」を行う．

連携図を作成する前に，電子取引導入前の食材発注に関する機能と情報連携を定義する．このような機能と情報連携は，情報システムに関連するステークホルダへのインタビューやブレインストーミングなどを通じて定義するため，通常，図 2.4 に示すように，自然言語を用いて，定義文の形で表現する．

(1) 店舗(発注・検収)が食材発注を行う場合，仕入先(食材提供)にFAXをすることにより，食材が納品書とともに店舗に届く．
(2) 月末になると，納品書が店舗(発注・検収)から商品本部(食材調達)に集められる．
(3) 商品本部(食材調達)はこの店舗(発注・検収)からの納品書と仕入先(食材提供)からの請求書を照合することにより，経理部(出納)に支払依頼を行う．
(4) 経理部(出納)は，支払依頼に基づいて，仕入先(食材提供)に支払いを行う．
(5) 支払後に，仕入先(食材提供)から領収書を受領する．

図 2.4 電子取引導入前の定義文

改修前の連携図を記述する際，まず，図 2.4 の定義文から，機能と情報連携を取りだす．そして，取りだした機能と情報連携の配置を考えながら，図 2.5 のように記述する．DFD では，機能は楕円で表し，楕円の中に機能名を記述する．また，機能間の情報連携は機能と機能をつなぐ線で表し，機能への入力・出力を示す情報連携の方向は矢印で表す．情報名は，この矢印線の周辺に記述する．図 2.5 の例では，機能数は 4 個，情報連携数は 7 個ある．

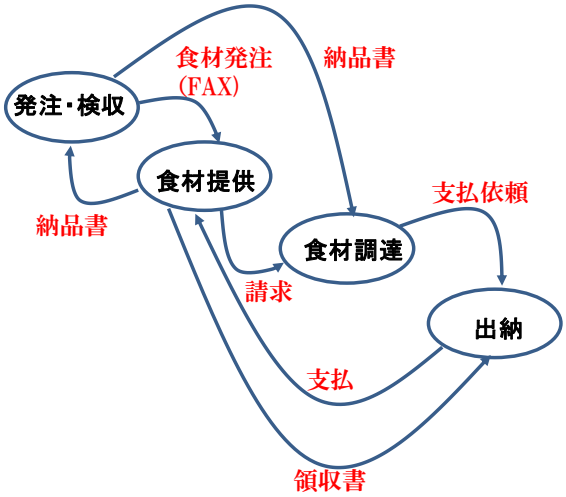


図 2.5 電子取引導入前の連携図(DFD)

2.2.3 機能を追加した連携図(DFD)の作成

次に，図 2.3 の「Step2 機能を追加した連携図の作成」を行う．Step 1 で作成した電子取引導入前の連携図に対し，新たな機能を追加し，情報連携のつなぎ換えを行う．

電子取引を導入すると，食材発注は店舗から仕入先に対して，EDI 会社を経由して行われるため，EDI 会社で実施する電子取引機能を新たに追加する．電子取引導入後の機能および情報連携の定義は，図 2.6 のような自然言語で，定義文の形で表現する．

(1) 店舗(発注・検収)が食材発注を行う場合，仕入先(食材提供)に対しEDI会社(電子取引)を経由して行なう．
食材は仕入先(食材提供)から納品書とともに店舗(発注・検収)に届く．
食材の検収は．仕入先(食材提供)に対して，EDI会社(電子取引)を経由して行う．
(2) 仕入先(食材提供)は，店舗(発注・検収)に納品すると同時に納品情報をEDI会社(電子取引)に送る．
(3) 月末になると，商品本部(食材調達)は仕入先(食材提供)からの請求に基づき，経理部(出納)に支払依頼を行う．
(4) 経理部(出納)は，支払依頼に基づいて，仕入先(食材提供)に支払いを行う．
支払後に，仕入先(食材提供)から領収書を受領する．
また，経理部(出納)は，EDI会社(電子取引)に対し，手数料を支払う．
支払後に，EDI会社(電子取引)から領収書を受領する．
(5) 商品本部(食材調達)は，EDI会社(電子取引)の発注・納品・検収に関する情報を参照できる．

図 2.6　電子取引導入後の定義文

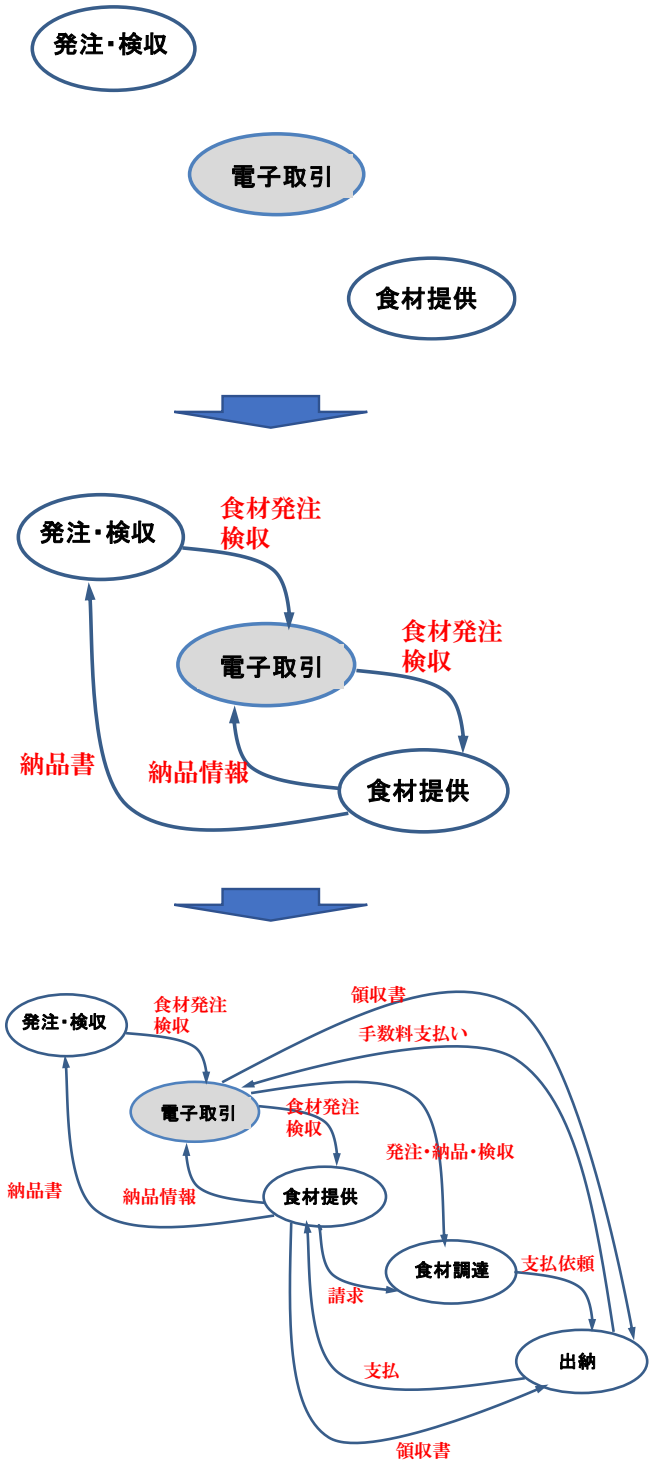
この定義文から，機能と情報連携を取りだし，図 2.5 で示した電子取引導入前の連携図を参考にしながら，新たな連携図を記述する．新たな連携図の作成は，次の手順で行う．

(1) 新たな機能である電子取引機能を配置する位置を考えながら，追加する．

(2) 追加した電子取引機能との間で発生する食材の発注・納品，検収に関する情報連携を追加する．

(3) 手数料支払いなどの電子取引機能との情報連携を追加する．電子取引導入前に行われていた情報連携は，削除，つなぎ換え，内容の変更の有無を確認して記述する，

連携図の作成手順を図 2.7 に示す．完成した連携図をみると，機能数は 5 個，情報連携数は 13 個である．



(1) 新たな機能である電子取引機能を配置する位置を考えながら，追加する．

(2) 追加した電子取引機能との間で発生する食材の発注・納品、検収に関する情報連携を追加する．

(3) 手数料支払いなどの電子取引機能との情報連携を追加する。電子取引導入前に行われていた情報連携は、削除，つなぎ換え，内容の変更の有無を確認して記述する．

図 2.7 電子取引導入後の連携図の作成手順(DFD)

図 2.7 で示した電子取引導入後の連携図を図 2.5 で示した電子取引導入前の連携図と図 2.8 のように比較してみると，機能数 5 個のうち，新規の機能は 1 個，従来から存在する機能は 4 個である．情報連携数 13 個のうち，新規の情報連携は 6 個，つなぎ換えが行われた情報連携は 2 個，従来から変化のない情報連携が 5 個ある．また，削除された情報連携は 1 個である．機能数が 1 個の増加に対して，情報連携数は 6 個増加しており，煩雑さが増して見える．また，従来から変化のない情報連携が 5 個あるにもかかわらず，書き直しが必要となっている．

このように，DFD を用いて機能を追加した連携図を作成するとき，既存の連携図から変化がない機能や情報連携が存在するにもかかわらず，書き直しが必要となり手間がかかることがある．

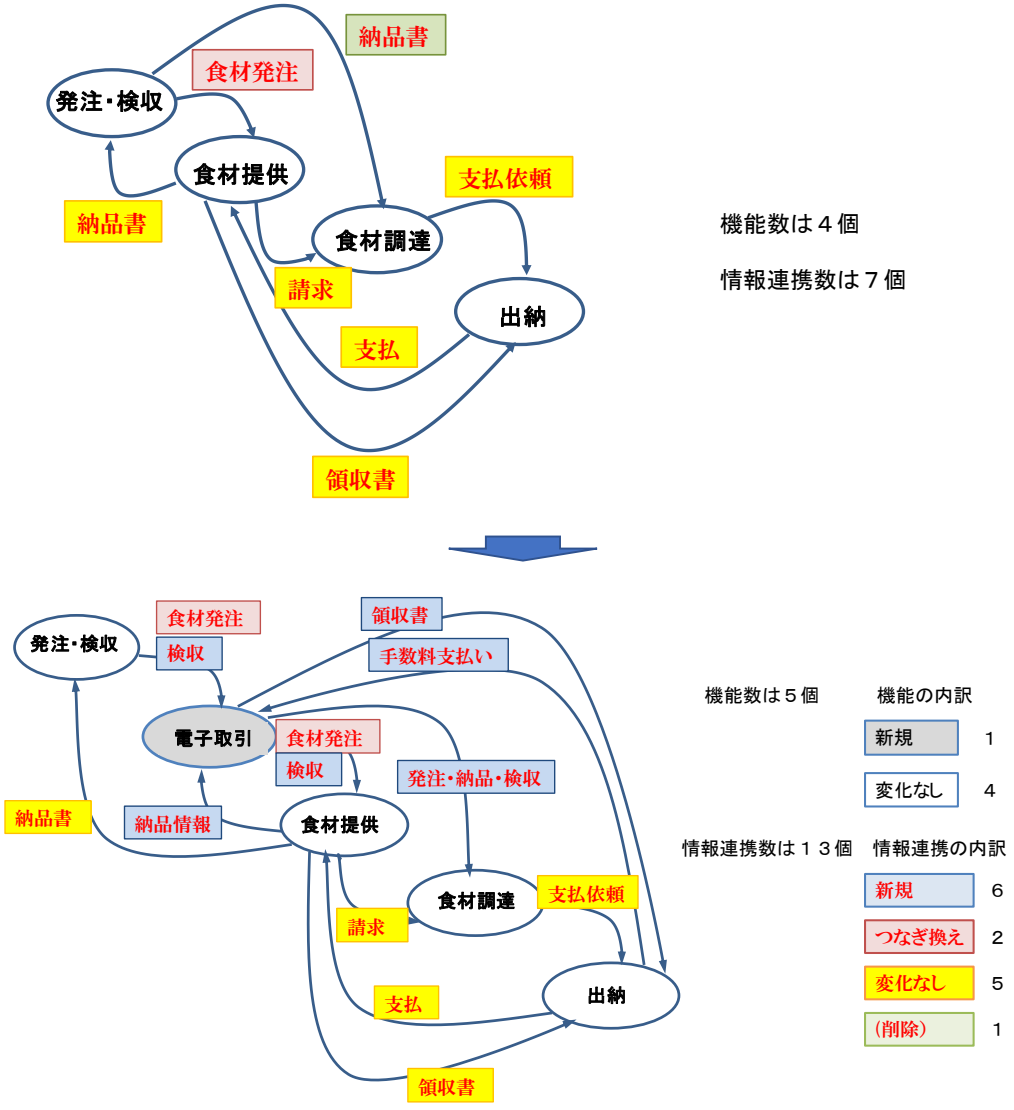


図 2.8 電子取引導入前後の連携図の比較

DFD を用いて新たな連携図を作成する際，図 2.7 のように，追加する機能の位置を考えて配置すると，機能と機能をつなぐ線の交差は見られない．しかし，追加する機能の配置によっては，図 2.9 のように，線の交差が起こる．そうすると，連携図が煩雑となり，情報連携をたどる際に見づらくなる．

このように，DFD を用いて機能を追加した連携図を作成する場合，追加する機能の位置を考えずに記述すると，情報連携の増加に伴い，矢印線が交差することがあり，情報連携がたどりにくくなる．したがって，矢印線が交差しないように，追加する機能の記述する位置を考える必要がある．また，追加する機能だけではなく，すでに記述されている機能を含めて矢印線が交差しないように，連携図全体の機能の配置を変える場合がある．この場合，連携図全体の書き直しが必要となり，手間がかかることになる．

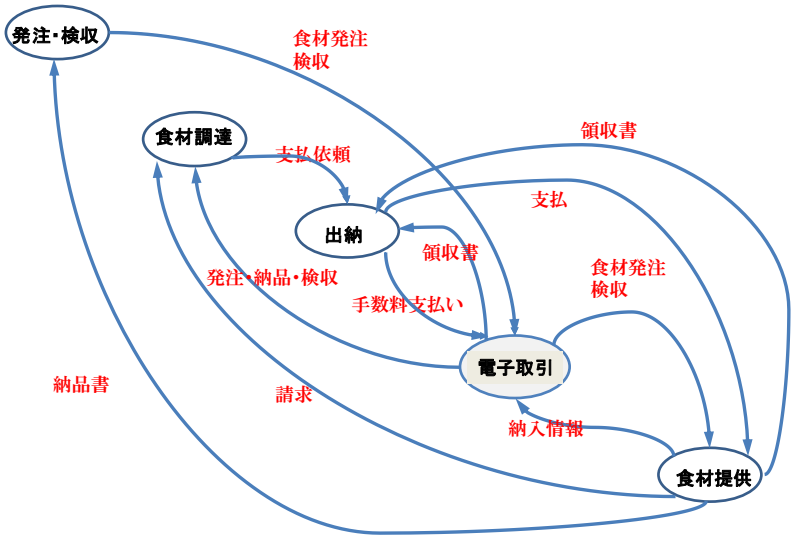


図 2.9　線の交差がある連携図の例

以上，Step 2 のケースから，DFD を用いて新たな連携図を作成する際の表記方法上の問題を煩雑性-1 としてまとめることができる．

煩雑性-1

現象　情報連携の増加に伴い，矢印線が交差する．

原因　フローチャート型表記方法は情報連携を矢印線で表す．

結果　情報連携がたどりにくくなる．

2.2.4 追加した機能を分解した連携図(DFD)の作成

次に，図 2.3 の「Step3 追加した機能を分解した連携図の作成」を行う．Step 2 で追加した電子取引機能を分解し，情報連携のつなぎ換えを行う．

電子取引機能を利用するには，食材マスタの登録が必要なため，電子取引の機能をデータ交換機能と食材マスタ登録機能に分解する．食材マスタ登録に関する機能および情報連携の定義は，図 2.10 のような自然言語で，定義文の形で表現する．

(1) 商品本部(食材調達)は仕入先(食材提供)に商品基礎情報を提供する.
(2) 仕入先(食材提供)は，EDI会社(電子取引)の食材マスタに対して，食材情報を登録する.
(3) 店舗(発注・検収)は食材マスタを参照して，発注を行う.
(4) 商品本部(食材調達)は，いつでも食材情報を参照できる.

図 2.10 食材マスタ登録に関する定義文

この定義文から，機能と情報連携を取りだし，電子取引導入後の連携図を参考にしながら，電子取引機能を分解した新たな連携図を作成する．

DFD を用いて，機能を分解する手順は以下のとおりである．

(1) 分解する機能の枠を作成する．

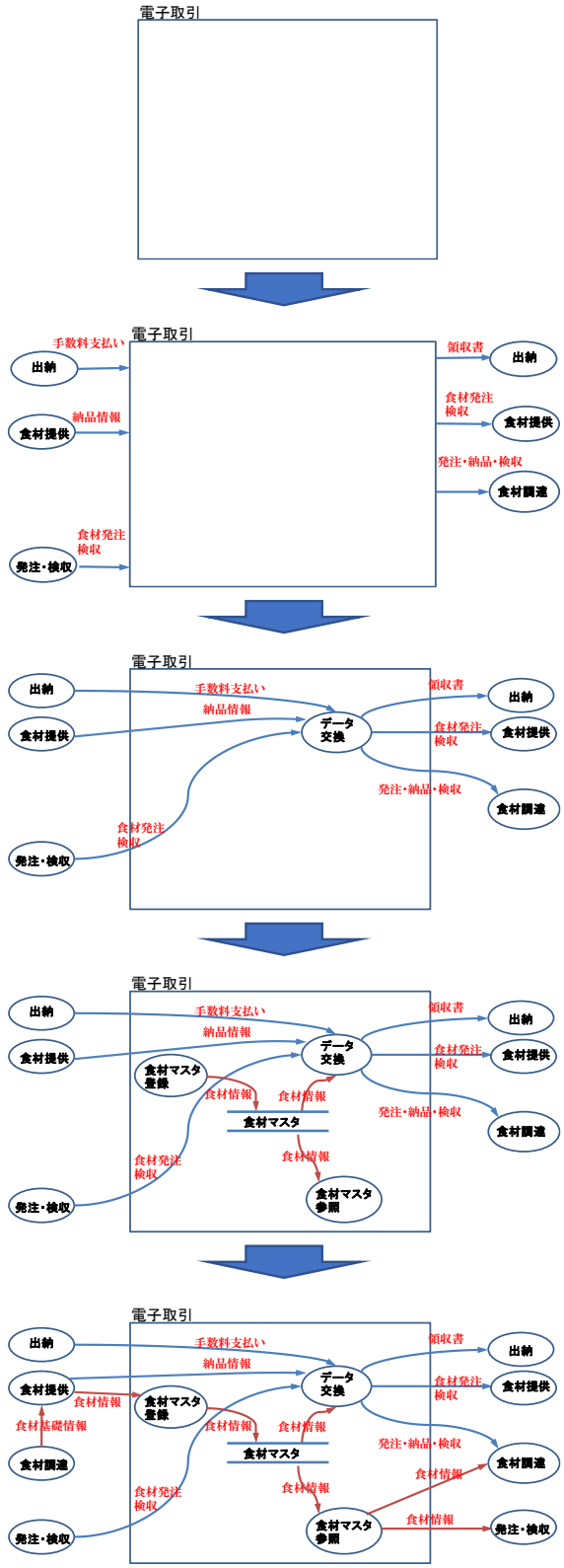
(2) 分解前の連携図を参照し，分解する機能と情報連携している機能を枠の外側に配置して，それらの機能と枠との間に情報名を転記する．

(3) 枠外の機能との入出力に対応する分解後の機能を枠内に記述し，情報連携のつなぎ換えを行う．

(4) 枠内の分解後の機能同士の情報連携を記述する．

(5) 枠内の分解後の機能と枠外の機能との新たな情報連携を記述する．

この手順を用いて，図 2.11 のように電子取引機能を分解した連携図を作成する．



(1) 電子取引機能を分解する枠を作成する.

(2) 電子取引機能と情報連携している機能を枠外に配置し,情報名を転記する.

(3) 分解後の機能として,データ交換機能を枠内に配置し,情報連携のつなぎ換えを行う.

(4) 枠内に食材マスタ登録機能,食材マスタ参照機能を新たに配置し,枠内の機能同士の情報連携を記述する.

(5) 枠内の分解後の機能と枠外の機能との新たな情報連携を記述する.
枠外の機能同士の連携が見つかった場合は,覚え書きとしてメモする.

図 2.11 電子取引機能を分解する手順(DFD)

図 2.11 で作成した分解後の電子取引機能と図 2.7 で作成した連携図の中の電子取引機能の間に階層の上下関係があることを示すため，図 2.12 のように，1 枚のシート上に 2 つの図を記述し，引出し線でガイドする表記方法を用いることがある．このような表記方法を用いると，結果として 1 枚のシートに記述する機能数と情報連携数が増えることになり，煩雑となる．この機能数や情報連携数の増加による煩雑さを避けようとすると，階層別に作成した 2 つの連携図を 2 枚のシートに分けて記述することになる．連携図が 2 枚のシートに分かれると，上位の機能と下位の機能の上下関係がわかりにくくなる．そのため，上位の機能と下位の機能をつなぐ情報連携をたどろうとすると手間がかかる．

なお，連携図を見やすくするため，1 枚のシートに記述する機能数を制限する考え方は，フローチャート型表記方法を用いた図の書き方として広く知られている．たとえば，IDEF 0 では 1 枚のシートに記述する機能を 3 個以上 6 個以内とすることが推奨されている[26].

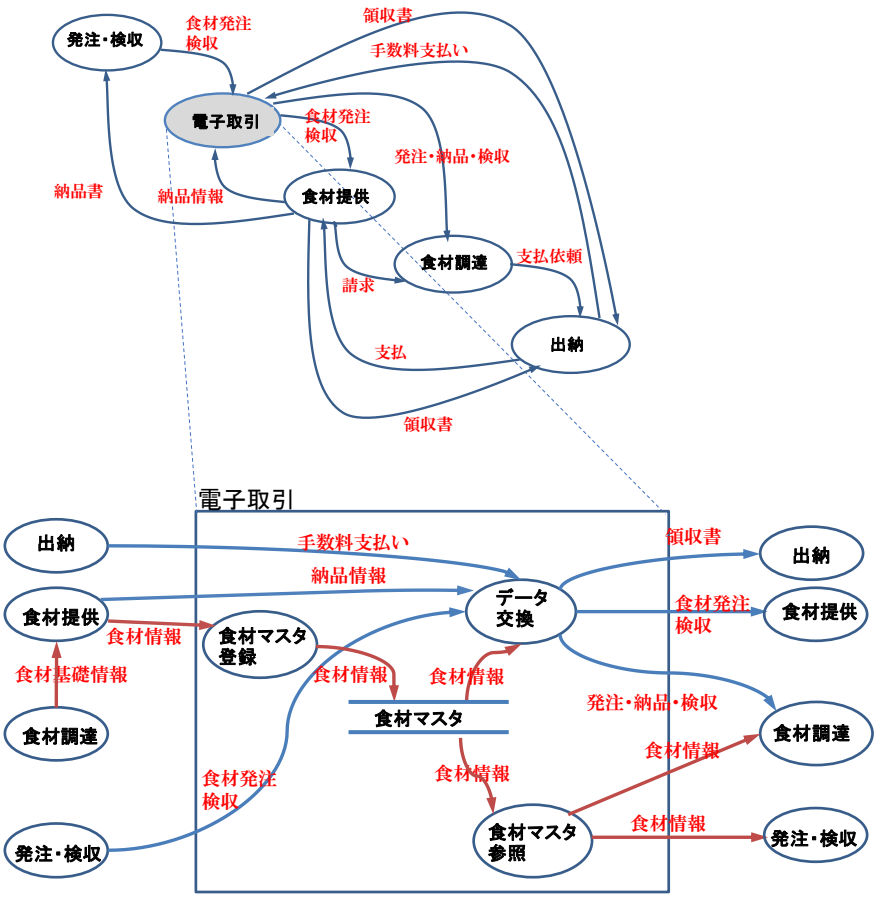


図 2.12　電子取引機能の階層関係を表す表記方法

以上の Step 3 のケースから，DFD を用いて機能を分解した結果を記述する際の表記方法上の問題を煩雑性-2 としてまとめることができる．

煩雑性-2

現象　機能を分解すると，機能数と情報連携数が増える．

原因　上位の機能と下位の機能にシートが分かれる．

結果　上位の機能と下位の機能の関係がわかりにくくなる．

2.2.5 既存の機能を分解した連携図(DFD)の作成

次に，図 2.3 の「Step4 既存の機能を分解した連携図の作成」を行う．Step 2 で作成した連携図に記述されている既存の発注・検収機能を分解し，情報連携のつなぎ換えを行う．

既存の発注・検収機能は新たに電子取引への検収が必要となるため，発注と検収の 2 つの機能に分解する．電子取引への検収に関する機能および情報連携の定義は，図 2.13 のような自然言語で，定義文の形で表現する．

(1) 店舗(発注・検収)はEDI会社(電子取引)の食材マスタ参照して，発注を行う．
その際，発注数量を決めるため，在庫管理に食材数量を問い合せる．
(2) 店舗(発注・検収)は仕入先(食材提供)の納入書を見て，EDI会社(電子取引)に対して，
検収を行う．
検収により，食材数量分，在庫をプラスする．

図 2.13　電子取引への検収に関する定義文

この定義文から，機能と情報連携を取りだし，電子取引導入後の連携図を参考にしながら，発注・検収機能を分解した新たな連携図を作成する．

DFD を用いて，機能を分解する手順は，Step 3 で述べた 5 つの手順と同様である．

(1) 分解する機能の枠を作成する．

(2) 分解前の連携図を参照し，分解する機能と情報連携している機能を枠の外側に配置して，それらの機能と枠との間に情報名を転記する．

(3) 枠外の機能との入出力に対応する分解後の機能を枠内に記述し，情報連携の

つなぎ換えを行う.

(4) 枠内の分解後の機能同士の情報連携を記述する.

(5) 枠内の分解後の機能と枠外の機能との新たな情報連携を記述する.

この手順を用いて，図 2.14 のように発注・検収機能を分解した連携図を作成する.

また，図 2.14 で作成した分解後の発注・検収機能と図 2.7 で作成した連携図の発注・検収機能の間に階層の上下関係があることを示すため，図 2.15 のように，2 つの図を引出し線でガイドする表記方法が用いられることがある.

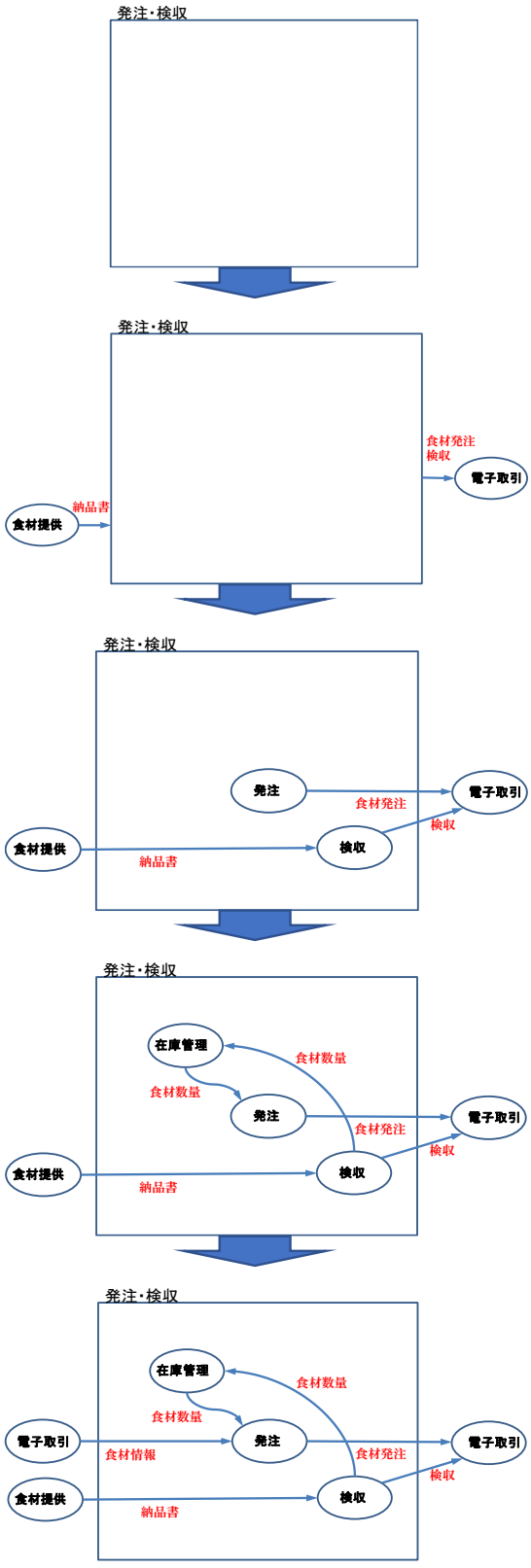
ここでの問題は，Step 3 で指摘した DFD を用いて機能を分解した結果を記述する際の表記方法上の煩雑性-2 と同様である.

煩雑性-2(再掲)

現象　機能を分解すると，機能数と情報連携数が増える.

原因　上位の機能と下位の機能にシートが分かれる.

結果　上位の機能と下位の機能の関係がわかりにくくなる.



(1) 発注・検収機能を分解する枠を作成する.

(2) 発注・検収機能と情報連携している機能を枠外に配置し，情報名を転記する.

(3) 分解後の機能として，発注機能と検収機能を枠内に配置し，情報連携のつなぎ換えを行う.

(4) 枠内に在庫管理機能を新たに配置し，枠内の機能同士の情報連携を記述する.

(5) 枠内の分解後の機能と枠外の機能との新たな情報連携を記述する．枠外の機能同士の連携が見つかった場合は，覚え書きとしてメモする.

図 2.14 発注・検収機能を分解する手順(DFD)

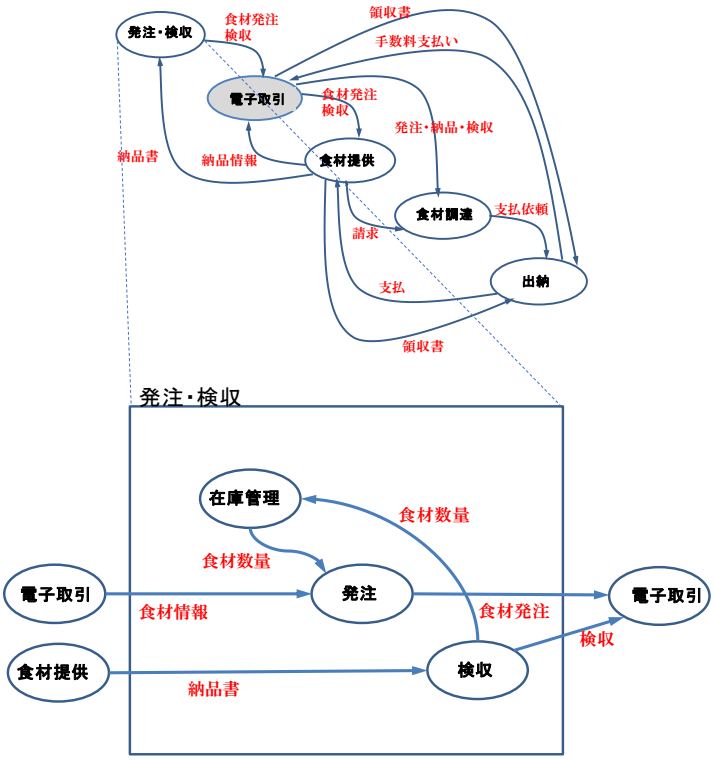


図 2.15　発注・検収機能の階層関係を表す表記方法

2.2.6 複数の機能を分解した連携図(DFD)の統合

最後に，図 2.3 の「Step5 複数の機能を分解した連携図の統合」を行う．Step 3,Step 4 で作成した分解後の機能が記述されている連携図を統合し，機能や情報連携の抜けもれ，重複，つなぎ換えによる矢印線のつなぎ間違いなどのエラーがないことを確認する．

Step 3 で電子取引機能,Step 4 で発注・検収機能という 2 つの機能を分解したため, 分解後の下位機能同士で情報連携のエラーがないか確認する. 具体的には, 分解後の電子取引機能と発注・検収機能を記述した連携図を見比べながら，次の 2 点を確認する．

(1) 発注・検収機能の枠外に記述した電子取引という機能名をたよりに，発注・検収の発注機能や検収機能から電子取引のデータ交換機能に情報連携していることを確認する．

(2) 電子取引機能の枠外に記述した発注・検収という機能名をたよりに，電子取

引機能の食材マスタ参照が発注・検収の発注に情報連携していることを確認する.

図 2.12 で作成した分解後の電子取引機能や図 2.15 で作成した分解後の発注・検収機能と，図 2.7 で作成した連携図の電子取引機能や発注・検収機能との間に階層の上下関係があることを示すため，図 2.16 の連携図のように，分解後の図を引出し線でガイドする表記方法が用いられることがある．また，分解後の機能同士の情報連携を示すため，図 2.16 の連携図のように，分解後の機能の枠外に記述した機能名同士を矢印線でつなぐ表記方法が用いられることがある.

図 2.16 の連携図は，図 2.12 の連携図と図 2.15 の連携図を 1 枚のシートにまとめたものであり，3 つの連携図を 1 枚のシートに記述している．1 枚のシートに記述する機能数と情報連携数が，図 2.12 の連携図や図 2.15 の連携図より，さらに増えることになり，煩雑さが増している．また，分解後の機能同士を結ぶため，新たな矢印線が付け加えられており，煩雑性-1 で指摘した矢印線に起因する煩雑さが増している.

この機能数や情報連携数の増加による煩雑さを避けようとすると，3 つの連携図を上位の連携図が記述された 1 枚のシートと下位の連携図が記述された 2 枚のシートの合計 3 枚のシートに分けて記述することになる．上位の連携図が記述されたシートと下位の連携図が記述されたシートが分かれると，煩雑性-2 で指摘した上位の機能と下位の機能の上下関係がわかりにくくなることがおこる.

また，上位の連携図が記述された 1 枚のシートと下位の連携図が記述された 2 枚のシートの合計 3 枚のシートに連携図が分かれることから，分解後の機能同士の情報連携をたどろうとすると，次のようなシートの参照手順となる.

(1) 上位の連携図が記述されたシートを参照し，情報連携の確認が必要な発信元の機能名と受信先の機能名を確認する.

(2) 発信元機能の下位の連携図が記述されたシートを参照し，確認が必要な分解後の機能名を確認する．また，その分解後の機能から情報連携している枠外の機能名が受信先の上位の機能名と一致していることを確認する.

(3) 受信先機能の下位の連携図が記述されたシートを参照し，確認が必要な分解後の機能名を確認する．また，その分解後の機能へ情報連携している枠外の機能名が発信元の上位の機能名と一致していることを確認する.

(4) 下位の連携図が記述された 2 枚のシートを参照し，分解後の発信元機能と枠外の受信先機能をつなぐ情報名と分解後の受信先機能と枠外の発信元機能をつなぐ情報名が一致していることを確認する．

(5) 分解後の下位機能同士がつながるように，枠外に書かれた相手先の上位の機能名を下位の機能名に変更する．この変更により，各機能の階層を合わせることができるが，上位の機能名が消える．そのため，上位機能をたどろうとすると，上位の連携図が記述されたシートを参照する必要がある．

この手順を用いて，図 2.17 のように，分解後の下位機能同士の発注機能からデータ交換機能へ情報連携していることを確認する．

このように，分解後の下位機能同士の情報連携をたどろうとすると，下位の連携図が記述されたシートだけではなく，上位の連携図が記述されたシートの参照も必要となる．そのため，分解後の下位機能同士をつなぐ情報連携がわかりにくくなる．また，分解後の下位機能同士の情報連携をたどろうとすると手間がかかる．

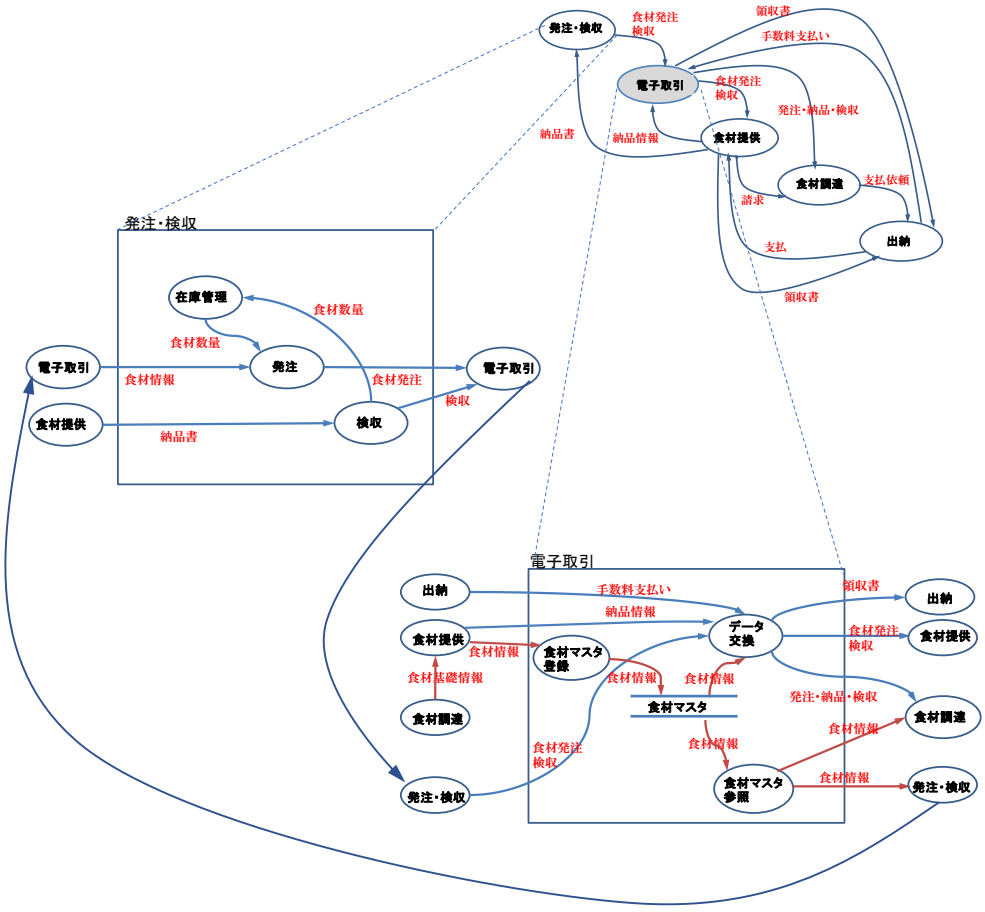
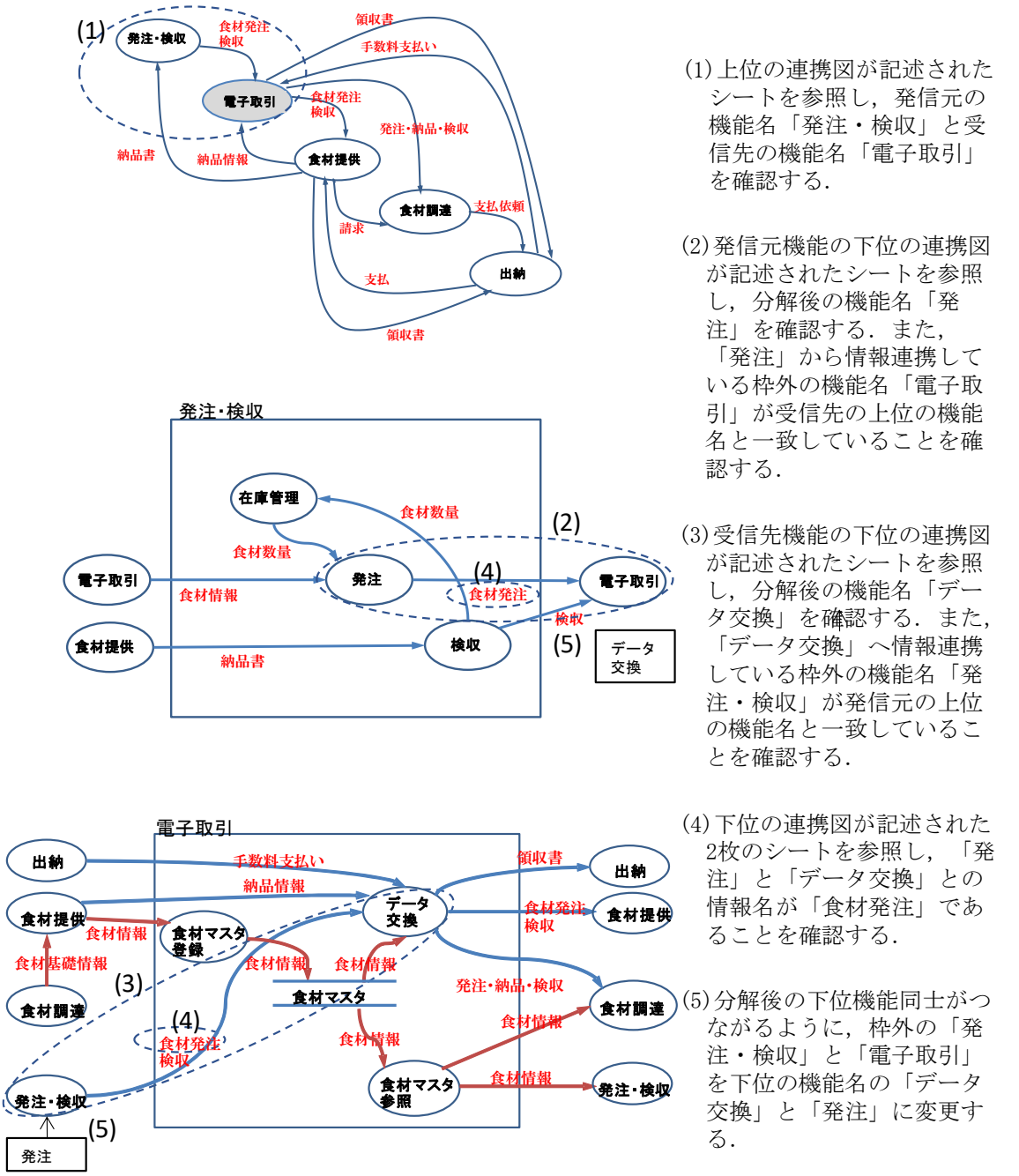


図 2.16 複数の機能を分解した後の連携図(DFD)



(1) 上位の連携図が記述されたシートを参照し，発信元の機能名「発注・検収」と受信先の機能名「電子取引」を確認する．

(2) 発信元機能の下位の連携図が記述されたシートを参照し，分解後の機能名「発注」を確認する．また，「発注」から情報連携している枠外の機能名「電子取引」が受信先の上位の機能名と一致していることを確認する．

(3) 受信先機能の下位の連携図が記述されたシートを参照し，分解後の機能名「データ交換」を確認する．また，「データ交換」へ情報連携している枠外の機能名「発注・検収」が発信元の上位の機能名と一致していることを確認する．

(4) 下位の連携図が記述された2枚のシートを参照し，「発注」と「データ交換」との情報名が「食材発注」であることを確認する．

(5) 分解後の下位機能同士がつながるように，枠外の「発注・検収」と「電子取引」を下位の機能名の「データ交換」と「発注」に変更する．

図 2.17 下位同士の情報連携を確認する手順

以上の Step 5 のケースから，DFD を用いて，分解後の下位機能同士で情報連携のエラーがないか確認する際の表記方法上の問題を煩雑性-3 としてまとめることができる．

煩雑性-3

現象　下位機能同士の情報連携を確認する際，上位の連携図を参照する必要がある．

原因　下位機能同士の情報連携の相手先機能が上位機能名で表現されている.

結果　下位機能同士の情報連携は直接たどれない.

以上，第 2 章では，煩雑性-1 から煩雑性-3 の 3 つの情報連携の表記方法上の問題を指摘した.

次章では，これらの 3 つの煩雑性を整理し，情報連携の表記に関する課題を導出する.

第３章　情報連携の表記に関する課題

3.1 情報連携表記方法上の問題整理

本節では，第 2 章で指摘した情報連携の表記方法上の煩雑性を入力とし，連携図を読みやすく，理解しやすいように記述するという観点から問題を整理する．

第 2 章で指摘した煩雑性は次の 3 点である．

煩雑性-1

現象　情報連携の増加に伴い，矢印線が交差する．

原因　フローチャート型表記方法は情報連携を矢印線で表す．

結果　情報連携がたどりにくくなる．

煩雑性-2

現象　機能を分解すると，機能数と情報連携数が増える．

原因　上位の機能と下位の機能にシートが分かれる．

結果　上位の機能と下位の機能の関係がわかりにくくなる．

煩雑性-3

現象　下位機能同士の情報連携を確認する際，上位の連携図を参照する必要がある．

原因　下位機能同士の情報連携の相手先機能が上位機能名で表現されている．

結果　下位機能同士の情報連携は直接たどれない．

これらの 3 つの煩雑性について，2.2 節に示した連携図を記述する作業手順にしたがって，問題を分析する．

煩雑性-1 は，2.2.3 項 機能を追加した連携図(DFD)の作成の中で「機能の追加し，情報連携のつなぎ換えを行う」際におこると指摘した．したがって，煩雑性-1 を入力とした問題分析では，「機能を追加し，情報連携のつなぎ換えを行う」作業を分析する．

煩雑性-2 は，2.2.4 項 追加した機能を分解した連携図(DFD)の作成と 2.2.5 項 既存の機能を分解した連携図(DFD)の作成の中で「機能を分解し，情報連携のつ

なぎ換えを行う」際におこると指摘した．したがって，煩雑性-2 を入力とした問題分析では，「機能を分解し，情報連携のつなぎ換えを行う」作業を分析する．

煩雑性-3 は，2.2.6 節 複数の機能を分解した連携図(DFD)の統合の中で「分解後の下位機能同士で情報連携のエラーがないか確認する」際におこると指摘した．したがって，煩雑性-3 を入力とした問題分析では，「分解後の下位機能同士で情報連携のエラーがないか確認する」作業を分析する．

3.2 情報連携表記方法上の問題分析

本研究では，情報連携表記方法上の問題を分析する際，作業分析に使われる動作分析を応用して実施する．具体的には，3 つの煩雑性を入力として，このような煩雑さが発生する動作を分析する．動作分析を応用した情報連携表記方法上の問題分析結果を図 3.1 に示す．

縦軸に，3 つの煩雑性を列挙する．横軸は，「記述作業」，「記述順序」，「現象」，「DFD 表記法上の制約」，「原因」とする．

最初に，分析する「記述作業」を列挙する．記述作業は，3.1 節で指摘した「機能を追加し，情報連携のつなぎ換えを行う」，「機能を分解し，情報連携のつなぎ換えを行う」，「分解後の下位機能同士で情報連携のエラーがないか確認する」作業とする．次に，この記述作業に対応する連携図の「記述順序」を洗い出す．たとえば，「機能を追加し，情報連携のつなぎ換えを行う」という記述作業に対応して，順番に「追加する機能を書く位置を決める」，「追加する機能名を書く」，「追加する矢印線を書く」，「矢印線の近くに情報名を書く」，「追加機能と既存機能の情報連携をつなぎ換える」，「変更のない情報名と矢印線を書き換える」といった記述順序を洗い出す．同様に，「機能を分解し，情報連携のつなぎ換えを行う」作業や「分解後の下位機能同士で情報連携のエラーがないか確認する」作業についても，記述順序を洗い出す．

次に，3 つの記述作業に対応して洗い出した記述順序の全体を俯瞰し，記述順序によっておこる表記上の「現象」を確認する．ここでは，1 つの現象が記述作業の違う複数の記述順序から導き出される．たとえば，「機能数が増加する」という現象は，「追加する機能名を書く」という記述順序と「分解後の機能名を書く」という記述順序の両方から導き出される．同様に，「情報名が増加する」「矢印線

が交差する」「上下関係をたどる時に別シートの参照が必要となる」という現象が導き出される.

次に，その現象がおこる「DFD 表記方法上の制約」を導き出す. たとえば,「機能数が増加する」という現象は，1 枚のシートに書く機能数が 3 個以上 6 個以内という表記方法上の制約によるものと考えられる. そのため，DFD 表記方法上の制約として,「1 枚のシートに書ける機能数は限られる」という制約を導き出す. 同様に,「1 枚のシートに書ける情報名数は限られる」「矢印線で情報連携を表現しているため，線が交差する」「分解した機能の連携先は相手の上位機能しかわからない」という DFD 表記方法上の制約を導き出す.

最後に，この DFD 表記方法上の制約が煩雑性を起こしていると考えられる場合に「原因」とする.

この分析の結果，煩雑性を起こしている原因は,「1 枚のシートに書ける機能数と情報名数は限られる」「矢印線で情報連携を表現しているため，線が交差する」「分解した機能の連携先は相手の上位機能しかわからない」という 3 つになることがわかる.

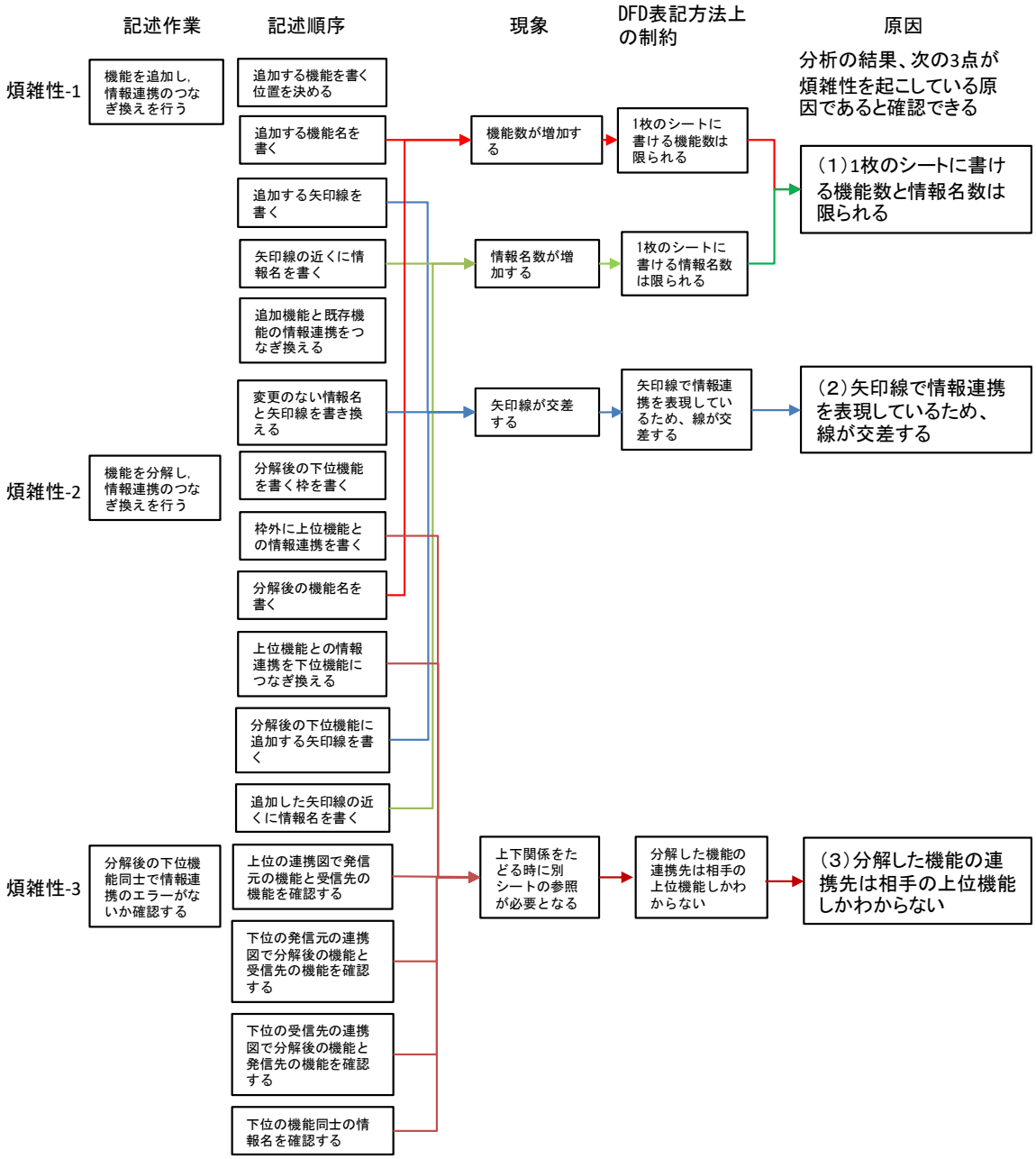


図 3.1 情報連携表記方法上の問題分析

3.3 情報連携表記方法上の課題

前節 3.2 節の問題分析の結果，煩雑性を起こしている 3 つの原因が確認できた．ここでは，この 3 つの原因から，図 3.2 に示すように情報連携表記方法上の課題を導き出す．

最初に，3 つの原因を取り除くという観点で，「着眼」を整理する．

3 つの原因に対応する着眼は次のとおりである．

(1)「1 枚のシートに書ける機能数と情報名数は限られる」を取り除く着眼

着眼-1 機能を記述する位置を決める．

着眼-2 情報連携を記述する位置を決める．

着眼-3 情報名をすべて書けるようにする．

(2)「矢印線で情報連携を表現しているため，線が交差する」を取り除く着眼

着眼-4 情報連携を線がなくても表現できる．

着眼-5 情報連携の向きを矢印がなくても表現できる．

(3)「分解した機能の連携先は相手の上位機能しかわからない」を取り除く着眼

着眼-6 分解した機能の上下関係がわかる．

着眼-7 分解した下位機能と別の上位機能との情報連携がわかる．

着眼-8 分解した下位機能同士の情報連携がわかる．

次に，これらの着眼から，提案する表記方法が具備すべき「狙い」を定義する．着眼-1～着眼-3 から，「機能と情報連携の収容能力を高める」という狙いが考えられる．着眼-1～着眼-5 から，「矢印線がなくても情報連携を表現できる」という狙いが考えられる．また，着眼-6～着眼-8 から，「引出し線がなくても階層の上下関係を表現できる」という狙いが考えられる

最後に，提案する表記方法が具備すべき狙いをまとめ，「情報連携表記方法上の課題」を求める．

以上より，情報連携表記方法上の課題は，次の 2 点に集約できる．

(1) 連携図を記述する際の課題

具体的には，機能の配置が決まっており，情報連携をたどるための線や矢印を必要としない収容能力の高い表記方法の実現である．

(2) 機能を分解した結果を記述する際の課題

具体的には，機能を分解後も階層の上下関係が把握でき，同時に分解した下位機能同士の情報連携が一覧できる表記方法の実現である．

これらの 2 つの課題から，情報連携表記方法に求められる具備要件は，次の 2 点に集約できる．

(1) 機能と情報連携を記述する位置が明示的に示せること

(2) 階層の上下関係が一覧できること

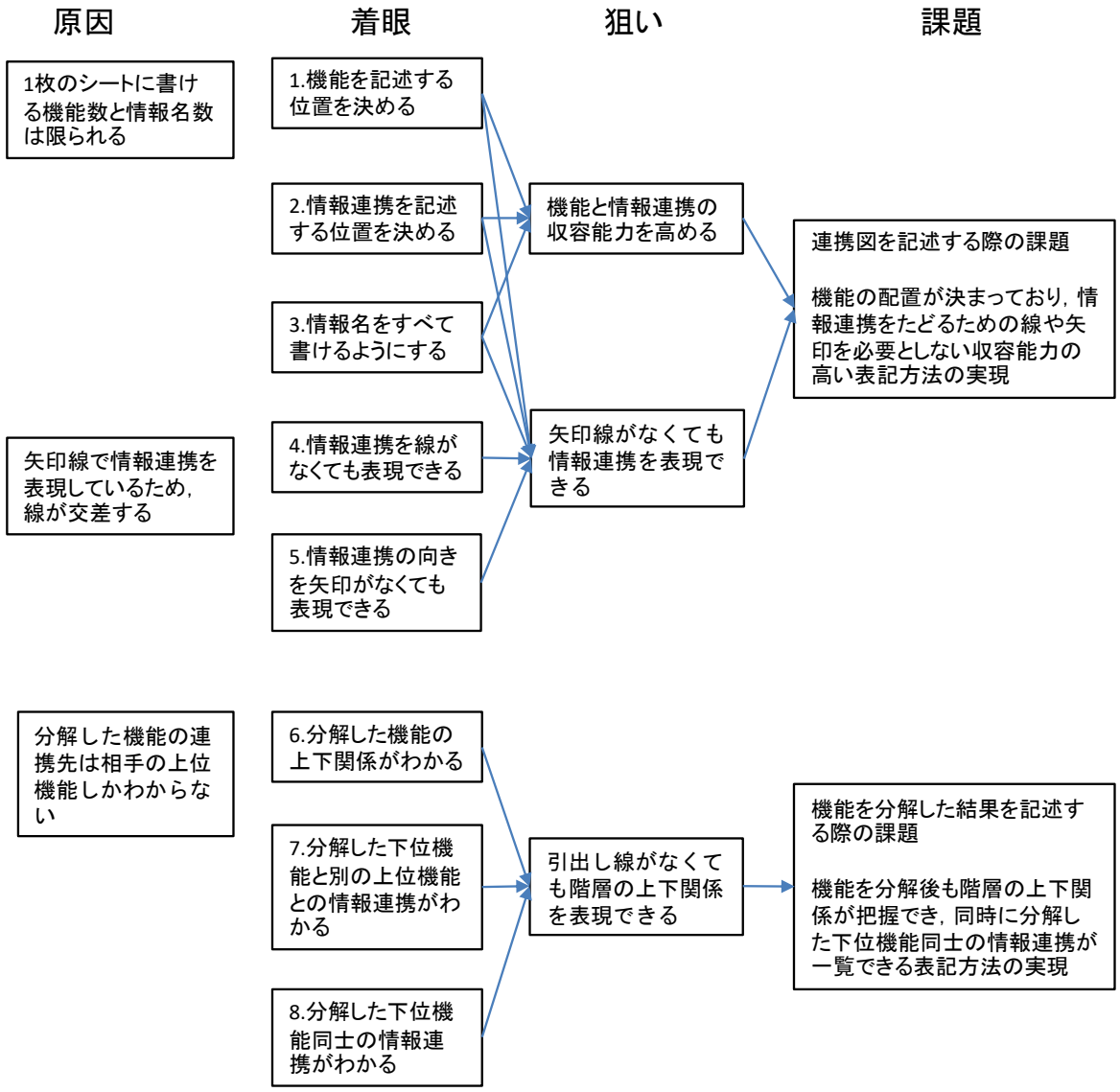


図 3.2　問題解決の着眼と課題

次章では，この 2 つの課題を解決しようとする，機能と情報連携の表記に関する従来手法を確認する．

第４章　情報連携の表記に関する従来手法

4.1 連携図を記述する際の課題解決の表記方法

第 3 章で導出した「連携図を記述する際の課題」に対応する表記方法として，線や矢印を使用せずに機能間の関係を示すことができる，行列を用いた表記方法がある．行列は，コンパクトに機能や機能間の関係を表現できるだけではなく，容易に拡張でき，また，直感的にわかりやすいという特徴がある．

本節では，線や矢印を使用しないという特徴をもつ，行列を用いた表記方法を中心に従来手法を調査する．また，従来手法を用いて図 2.5 に示した連携図を記述することにより，その従来手法が連携図を記述する際の課題を解決できるかどうかを確認する．

4.1.1 構造設計マトリクス(DSM)

構造設計マトリクス（DSM: Design Structure Matrix）は，後戻りや反復作業を含む複雑なプロセスの構造を簡潔に表現するため，タスク間の依存関係を網羅的に示すマトリクスである[22][23][24]．DSM は，タスクを記述する位置が決まっている行列を用いた表記方法である． DSM は当初，主に製品設計に用いられていたが，工程設計，組織編成や業務機能の設計にも応用されており，事例も多数報告されている．プロセス設計に応用した DSM の特徴として，(1)プロセスを構成するタスクを実行順にマトリクスの対角線上に上から下に記述する，(2)マトリクス上の対角線の下半分のマークはフィードフォワードを示し，上半分のマークはフィードバックを表している，(3)対角線上の要素はある目的を実現する 1 つのプロセスに限定したタスクであり，原則として複数のプロセスは記述しない，がある．DSM は複雑なプロセスの分析に有効であり，タスク順序を最適化することに効果がある[25]. 図 4.1 の左図はタスク間の依存関係を示した DSM であり，煩雑に見える右図の有向グラフと等価である[22].

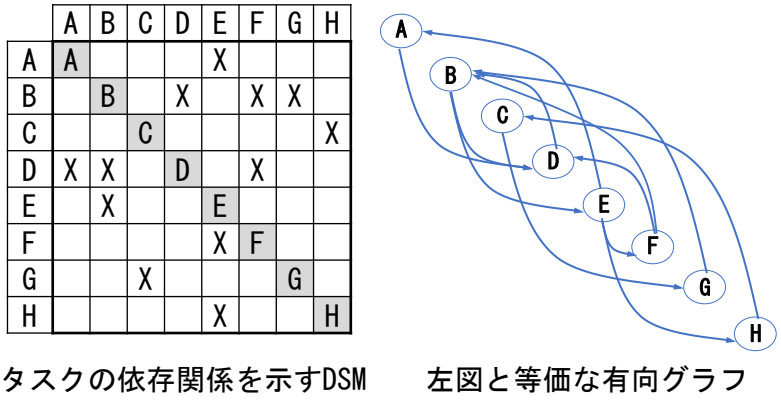


図 4.1 DSM の凡例

DSM を用いて, 図 2.5 に示した連携図を記述すると図 4.2 のようなマトリクスになる.

DSM 上のマークは機能間に依存関係があることを示しており，機能間に情報の流れがあることを示唆しているが, DFD の矢印線の近くに書かれる情報のように機能間でやりとりされる情報名を記述することはない．したがって，情報名は別の図表を使って記述する必要があるという問題がある. DSM は, 原則として, 1 つのマトリクスに複数のプロセスを記述することはない．そのため，情報システム開発に DSM を適用した場合，プロセスごとに複数のマトリクスが必要となる．また，複数のマトリクスが必要となると，マトリクス間の情報連携が記述できないという問題がある.

DSM の応用事例として，製品からその製品を構成するモジュールに分解する適用例が見られるが，モジュール間にインターフェースが存在することを示唆するにとどまっている[22].

	発注・検収	食材提供	食材調達	出納
発注・検収	発注・検収	X	X	
食材提供	X	食材提供	X	X
食材調達			食材調達	X
出納		X		出納

図 4.2 DSM による連携図

4.1.2 フロムツウチャート(from-to chart)

行列を用いて，矢印線が不要な連携図としてフロムツウチャートがある[26][27]．フロムツウチャートは，流出流入図表ともよばれ，多品種少量生産の工程を分析する際に用いられる．フロムツウチャートでは，最初に，設備を列挙し，これらを表の一番上の行に記入する．この行は，設備間を流れる物の行き先を示すので，“To”とする．“To”の行に書かれている同じ順序で，設備を表の一番左の列に記入する．この列は，設備間を流れる物の送り元を示すので“From”とする．設備間を流れる物がどの設備からどの設備へ移動するかを把握し，その物流量をこの表に記述する．たとえば，図 4.3 の設備 A から設備 E に 8 単位の物が流れていることがわかる．

From \ To	A	B	C	D	E
A		3		2	8
B	6				
C		10			
D					4
E				7	

図 4.3 from-to chart の凡例

フロムツウチャートを用いて，図 2.5 に示した連携図を記述すると，図 4.4 のような表になる．最初に，情報連携の受信先の機能を示す行を“To”として，表の一番上の行に記入する．次に，情報連携の発信元の機能を示す列を“From”として，“To”の行に書かれている同じ順序で，表の一番左の列に記入する．その後，“From”に書かれた機能から“To”に書かれた機能への情報連携を“From”に書かれた機能の行と“From”に書かれた機能の列の交点に記述する．このように，フロムツウチャートを活用すると，図 4.4 のように機能と機能間の情報連携を表現することができる．

フロムツウチャートでは機能が 2 か所に記述されているため，情報連携をたどる時に，“To”の機能から“From”の機能を見つけ出し，次の行をたどりなおさなければならないという問題がある．また，機能が“From”の列と“To”の行の 2 か所に書かれているため，機能の分解が難しいという問題がある．

From　To	発注・検収	食材提供	食材調達	出納
発注・検収		食材発注	納品書	
食材提供	納品書		請求	領収書
食材調達				支払依頼
出納		支払		

図 4.4 フロムツウチャートによる連携図

4.2 機能を分解した結果を記述する際の課題解決の表記方法

第 3 章で導出した「機能を分解した結果を記述する際の課題」に対応する表記方法として，機能の分解後も階層関係を把握でき，分解した下位機能同士の情報連携が把握できる表記方法に UML アクティビティ図がある．また，階層を使って機能を分解する表記方法に機能構成図がある．

本節では,機能を分解した結果を記述する際の課題に対応する従来手法として,UML アクティビティ図と機能構成図を調査する.また,従来手法を用いて図 2.12 や図 2.15 に示した機能を分解した連携図を記述することにより,その従来手法が機能を分解した結果を記述する際の課題を解決できるかどうかを確認する.

4.2.1 UML アクティビティ図

UML は 1997 年にオブジェクト指向技術標準化団体(OMG: Object Management Group)によって，標準モデリング技法として採択された[28]．UML の中でアクティビティ図(Unified Modeling Language Activity Diagram)は，業務の処理手順を表現することを狙いとしている．アクティビティ図は，表 4.1 に示すような複数のアクティビティとよばれる処理要素を分岐や結合などの制御要素を使って順序づけることにより，より現実の処理に近い業務の流れを表現する[8].

UML アクティビティ図では，図 4.5 に示すようにスイムレーンを用いて，機能を分解することができる．スイムレーンでは，垂直の線で，区分けしたいいくつかのゾーンを表し，各ゾーンに上部に分解前の機能名を記入する．そして，分解後の機能は，分解前の機能名を記入したゾーン内に，処理順に上から下に並べて記述する．分解後の機能同士を結ぶ情報連携は矢印線で記述し，情報名は矢印線の近くに記述することができる．

このように，スイムレーンでは，機能を分解したあとも，下位機能同士の情報

連携を一覧できる工夫がなされている．また，分解前の機能名を記入した同じゾーン内に分解後の機能が記述されるため，機能の階層関係が把握しやすい．

一方，1 つの処理手順を表すために，その処理手順をスイムレーンで記述した 1 枚のシートが必要となる．そのため，処理手順が多くなると，それに応じてシート枚数も多くなるという問題がある．たとえば，図 4.5 では発注・検収に関する処理手順は示しているが，請求・支払に関する処理手順を示すことは難しい．また，シート枚数が多くなると，別シートに記述された機能間の情報連携を把握しにくいという問題がある．

表 4.1 UML アクティビティ図の凡例

No.	表記	記号	表記の説明
1		開始	業務の流れの開始を表す.
2		終了	業務の流れの終了を表す.
3		アクティビティ， アクション	業務の処理を表す.
4		分岐/合流	条件による分岐/合流を表す.
5		フォーク/ジョイン	並行(複数処理の開始)/同期(複数処理の終了)を表す.

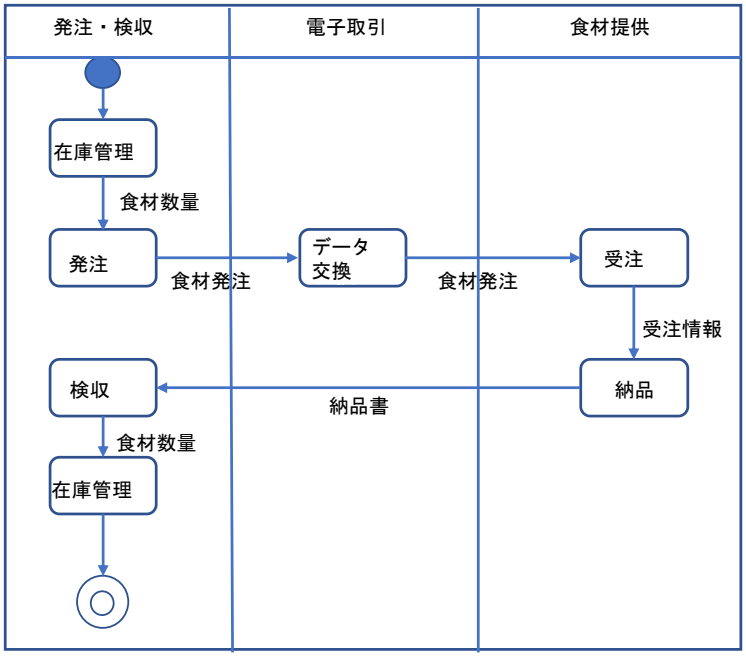


図 4.5 スイムレーンの例

4.2.2 機能構成図(DMM)

機能構成図 DMM(Diamond Mandara Matrix)は，経済産業省が主導して構築した EA(Enterprise Architecture)[29][30]の中でビジネスアーキテクチャの成果物として定義されている[31]．DMM は図 4.6 に示すように，機能を階層的に 3×3 のマトリクスで表現する表記方法である．DMM は機能を配置する位置がマトリクスのセル内と決まっている．また，階層を使って機能の分解を表現することができる．しかし，DMM は 1 つの階層に記述できる機能が 8 個以内に限定され，多様な企業情報システムにおける機能を表現するには少ない場合がある．

DMM のもつ機能を分解できる特徴を用いて，2 つの機能を分解した例を図 4.7 に示す．

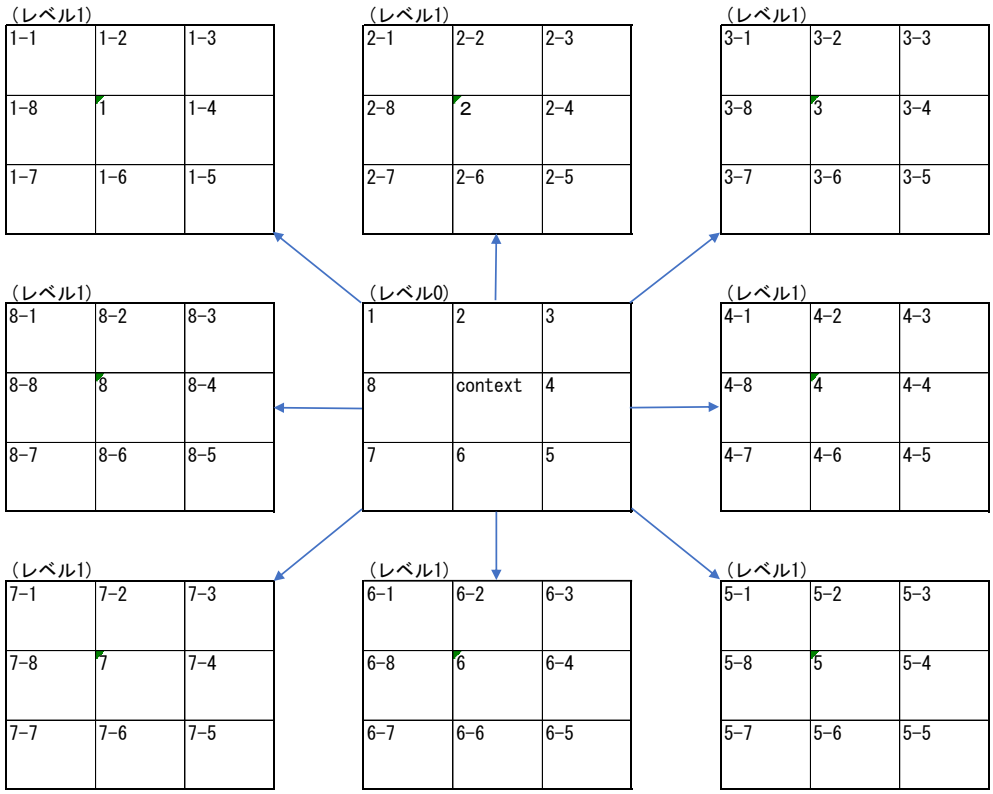


図 4.6　DMM の凡例

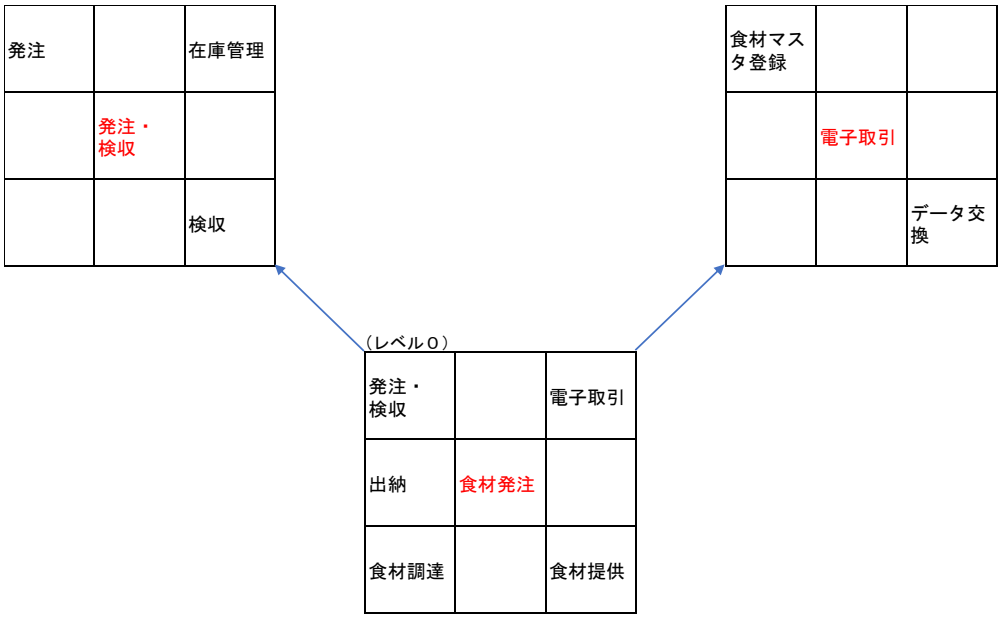


図 4.7　DMM の例

図 4.7 の例でもわかるように，DMM では，機能の分解はできるが情報名の記述はできないという問題がある．

4.3 課題に対する従来手法のまとめ

4.1 節で述べた連携図の記述に関する課題と 4.2 節で述べた機能の分解結果の記述に関する課題に対する従来の表記方法の対応を表 4.2 にまとめる.

行列を用いた表記方法である DSM, DMM は，機能を記述する位置は決まっているが，情報名を記述することはできないことがわかる．それに対し，from-to chart は情報名を記述することはできるが，機能の分解を表現することが難しいことがわかる．一方，行列を用いない表記方法であるスイムレーンは，機能の分解を表現することはできるが，矢印線を使うためフローチャート型と同様の問題を抱えていることがわかる.

その結果，従来の表記方法は，連携図の記述に関する課題と機能の分解結果の記述に関する課題の両方には対応できていないことがわかる.

表 4.2 表記方法の課題に対する対応

本研究の課題	着眼No	従来の表記方法				従来手法の問題
		DSM	from-to chart	スイムレーン	DMM	
連携図の記述に関する課題	1. 機能を記述する位置を決める.	○	○	○	○	・DSM, DMMは，情報名を記述することはない. ・スイムレーンは，矢印線を使うため，フローチャート型と同様の問題を持つ.
	2. 情報連携を記述する位置を決める.	○	○			
	3. 情報名をすべて書けるようにする.		○			
	4. 情報連携を線がなくても表現できる.	○	○			
	5. 情報連携の向きを矢印がなくても表現できる.	○	○			
機能の分解結果の記述に関する課題	6. 分解した機能の上下関係がわかる.	○		○	○	・DSM, DMMは，情報名を記述することはない. ・from-t chartは，機能の分解が難しい.
	7. 分解した下位機能と別の上位機能との情報連携がわかる.			○		
	8. 分解した下位機能同士の情報連携がわかる.			○		

次章では，連携図を記述する際の課題と機能を分解した結果を記述する際の課題の両方に対応する，行列を用いた表記方法を提案する.

第 5 章　提案する行列を用いた表記方法(MDM)

5.1 提案する表記方法の考え方

DFD では，図 2.5 に示したように，機能は複数の楕円で記述し，どの機能(楕円)からどの機能(楕円)に情報を連携(伝達)しているかは，矢印線(→)で記述する．このことから,一般的な DFD の表記方法には 3 つのルールがあると考えられる．

(1) 機能を示す楕円は他の楕円から情報を入力(受信)する．

(2) 機能を示す楕円は他の楕円へ情報を出力(発信)する．

(3) 受発信される情報はどの楕円からどの楕円へという発信元と受信先の機能がわかっている．

このように，一般的な DFD の表記方法は機能を中心に表現するルールとなっている．

本研究では，この表記方法を図 5.1 で示すように新たな 2 つのルールを用いて表現しなおす．

(1) 機能を示す楕円は箱で示し,これを機能 E とし図中のすべての機能に 1～n の番号を付ける．これにより，各機能は E(1)～E(n)と示すことができるようになる．この機能 E に付ける 1～n の番号は，機能を記述する場所(アドレス)を示すことになる．

(2) 受発信される情報は情報連携 R とし，図中のすべての情報連携にどの機能からどの機能に向かって情報連携するかという方向(ベクトル)を示す機能番号(アドレス)を付ける．具体的には，図 5.2 の右図で示すように，ベクトルは,1 次元 y が受信先の機能番号で,2 次元 x が発信元の機能番号とする．たとえば，R(3,2)は機能 E(3)が受信先で機能 E(2)が発信元の情報連携である．R(2,3)は機能 E(2)が受信先で機能 E(3)が発信元の情報連携である．

この新たな 2 つのルールを用いると，DFD を表現しなおすことができる．一般的な DFD では，図 5.2 の左図のように，機能 P を中心に情報連携は入力 I，出力 O と表現する．一方，機能にアドレスをもった DFD では，図 5.2 の右図のように，アドレスをもった機能と，機能間をつなげるベクトルをもった情報連携で表現する．

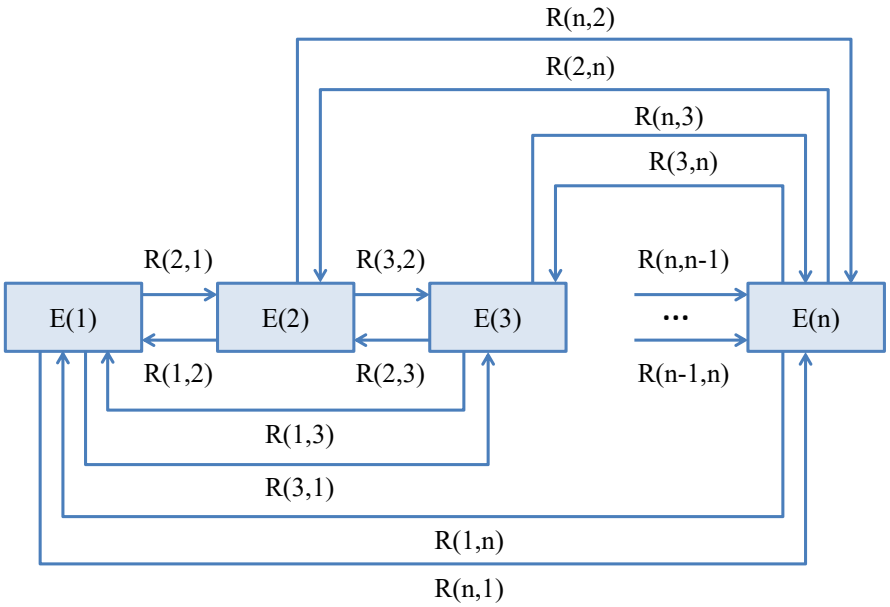


図 5.1 機能にアドレスをもった DFD の表記ルール

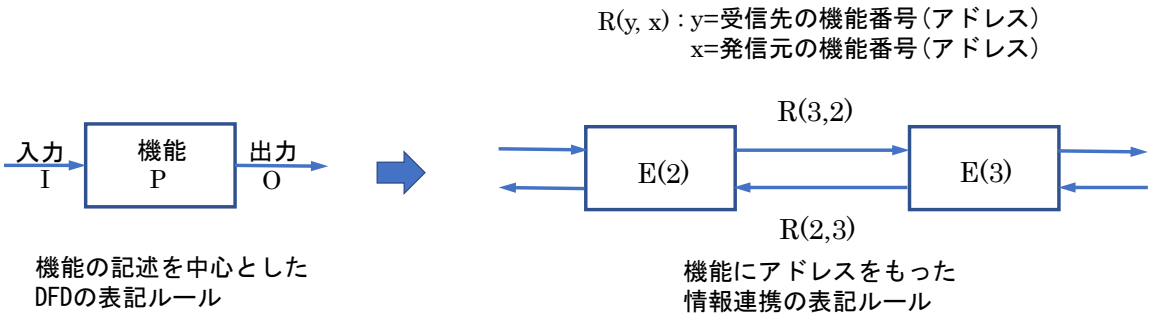


図 5.2　情報連携の表記ルール

このように表現しなおした DFD の，ある機能 E (x)に注目すると，機能 E (x)は他のある機能 E(y)に情報連携 R(y,x)を発信する．また，ある機能 E(y)に注目すると，機能 E(y)は他のある機能 E(x)から情報連携 R(y,x)を受信するという関係を示すことができる．ここで，情報連携 R(y,x)の値は，情報連携が 1 個以上存在する場合に「情報連携(矢印線)あり」という意味を示す値 1 を設定し，1 個も存在しない場合に「情報連携(矢印線)なし」という意味を示す値 0 を設定する．

そこで，情報連携 R(y,x)から受信先機能 E(y)と発信元機能 E(x)を分離し，ある機能 E(y)が受信する情報連携 R(y,x)の数(受信している矢印線の有無)と，ある機能 E (x)が発信する情報連携 R (y,x)の数(発信している矢印線の有無)をカウントする．

受信先機能 E (y)が受信する情報連携の数を Ri (y)個とすると Ri (y)の個数は (1) 式で示すことができる．この式から受信先機能 E(y)が受信する情報連携数の最大は機能数と等しい n 個であることがわかる．

$$Ri(\mathrm{y}) = R(\mathrm{y},1) + R(\mathrm{y},2) + \cdots + R(\mathrm{y},\mathrm{n}) = \sum_{x=1}^{n} R(y,x) \quad \cdots\cdots (1)$$

同様に，発信元機能 E(x)が発信する情報連携数を Ro(x)個とすると Ro(x)の個数は (2) 式で示すことができる．この式から発信元機能 E (x)が発信する情報連携数の最大は機能数と等しい n 個であることがわかる．

$$Ro(\mathrm{x}) = R(1,\mathrm{x}) + R(2,\mathrm{x}) + \cdots + R(\mathrm{n},\mathrm{x}) = \sum_{y=1}^{n} R(y,x) \quad \cdots\cdots (2)$$

また，情報連携 R (y,x)は発信元機能 E (x) から受信先機能 E (y)へと方向を示すので，その方向を発信元ベクトル x と受信先ベクトル y とに分けて考えることができる．これにより，(1) 式の受信先機能 E(y)が受信する情報連携数 Ri(y)と，(2) 式の発信元機能 E (x)が発信する情報連携数 Ro (x)の積は，DFD 中に存在する情報連携数の最大個数になるといえる．そこで，ある機能 E(y)が受信する情報連携 R (y,x)の(x=1～n)を行ベクトル，ある機能 E (x)が発信する情報連携 R (y,x)の(y=1～n)を列ベクトルとすると，DFD 中の最大情報連携可能数 N は (3) 式で求めることができる．

$$\mathrm{N} = Ri(\mathrm{y}) \times Ro(\mathrm{x}) = \sum_{y=1}^{n}\sum_{x=1}^{n} R(y,x)$$

$$= |R(\mathrm{y},1) \quad \ldots \quad R(\mathrm{y},\mathrm{n})| \times \begin{vmatrix} R(1,\mathrm{x}) \\ \vdots \\ R(\mathrm{n},\mathrm{x}) \end{vmatrix}$$

$$= \begin{vmatrix} R(1,1) & \cdots & R(1,\mathrm{n}) \\ \vdots & & \vdots \\ R(\mathrm{n},1) & \cdots & R(\mathrm{n},\mathrm{n}) \end{vmatrix} \quad \cdots\cdots (3)$$

この行列式は，図 5.3 のように，機能数 n の二次元正方行列(n,n)上の行×列の位置に情報連携 R(1,1)～R(n,n)が存在することを示している．これは，機能数 n

の DFD 中で発生する可能性がある情報連携のすべてを網羅することを意味する．また，対角は受信先と発信元の機能が同じであることを意味するため，他の機能との情報連携は存在しない．そのため，対角上の情報連携数はゼロ個である．これらのことから，機能数が n 個の DFD 中において情報連携数(矢印線の本数)の最大数は，(n^2-n)個になるということがわかる．

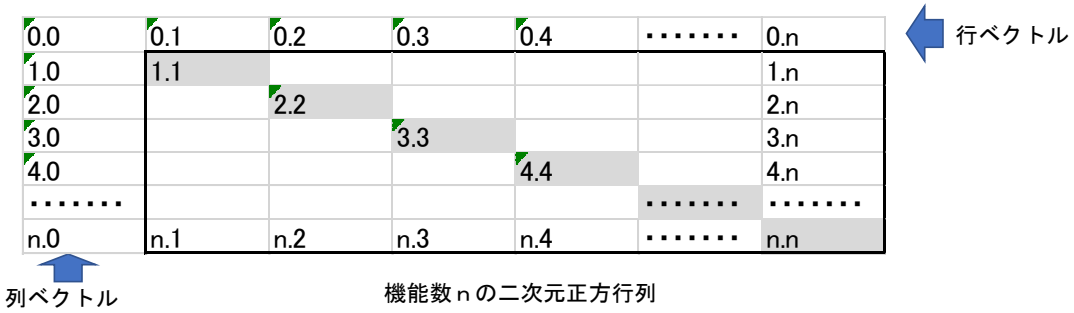


図 5.3 機能数 n の二次元正方行列と情報連携数

そこで，情報連携が存在するベクトル位置(y,x)において，情報連携が存在するか，または，存在しないかという状態を示す値(0 or 1)の代わりに，情報連携が存在しない場合は空値(null)の値を設定し，情報連携が存在する場合に情報名 R を値として設定することができれば，DFD は機能数 n の二次元正方行列を用いて表現可能になる．機能と情報連携の両方にアドレスをもつ図 5.4 の左図の DFD は，右図のように機能と情報連携を記述する位置にアドレスをつけた二次元正方行列として表現できるようになる．この左右の 2 つの図は等値である．この表記ルールにより，一般的な DFD の機能を中心に表現するルールは，二次元正方行列を用いた機能と情報連携を対等に表現するルールとなる．

機能と情報連携を記述する位置にアドレスをつけた二次元正方行列を用いることにより，図 3.2 で指摘した連携図を記述する際の課題の 1 つである「機能の配置が決まっており」を実現することができる．したがって，3.3 節で指摘した情報連携表記方法の具備要件の 1 つである「機能と情報連携を記述する位置が明示的に示せること」を満たすことができる．

このように，二次元正方行列を用いて，機能と情報連携を表記する方法を，MDM(Matrix Description Method)と呼ぶ．

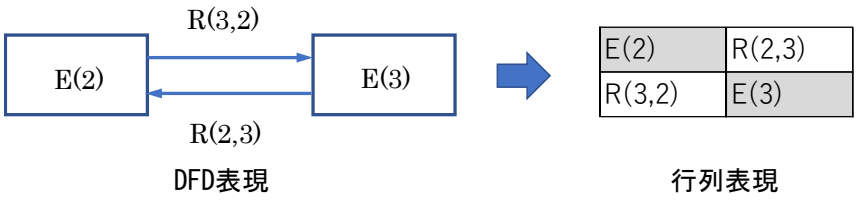


図 5.4 等値な DFD 表現と行列表現の例

5.2 行列を用いた表記方法(MDM)

5.2.1 行列を用いた表記方法(MDM)の考え方

MDM では機能 E を二次元正方行列の対角上に配置し，機能と機能を結ぶ情報連携 R(y,x)は 2 つの機能 E(y)の行と E(x)の列の交点に位置するように配置する．これにより，その交点の座標は発信元機能 E(x)，受信先機能 E(y)を同時に示すことになる．その際，図 5.5 で示すように情報連携の方向は行列交点の上下(垂直)方向が発信，左右(水平)方向が受信と定義する．すると，情報連携は列番号(x)で示す機能が発信元で，行番号(y)で示す機能が受信先になる．これにより，情報連携は反時計回りの一方通行で表現することが可能になり，矢印と線が不要になる．

これらの考え方による MDM を用いた基本的な連携図は，二次元行列表，機能，情報連携の 3 つを用いて表現する．MDM を用いた連携図では，機能名を二次元行列表の対角上のセルに記述し，情報名は 2 つの機能の行と列の交点のセル中に記述する．

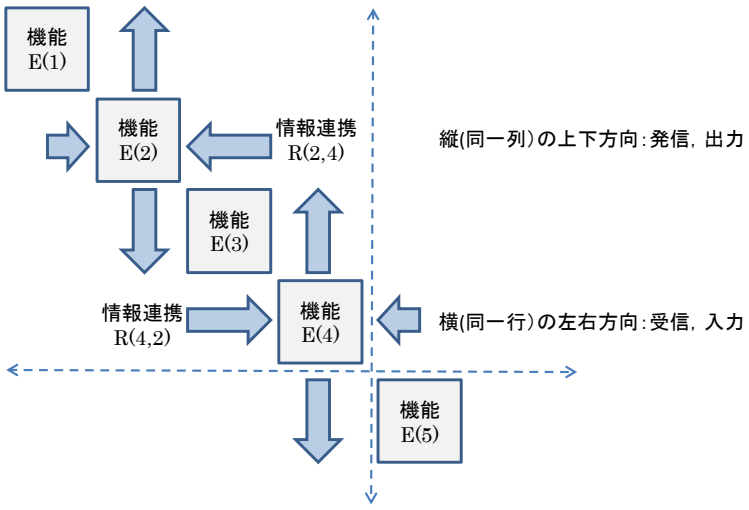


図 5.5　行列を用いた表記方法

なお，本論文中の MDM を用いた連携図上の矢印線は，読みやすさの便宜のために付したものである．

5.2.2 二次元行列表

二次元行列表は，MDM を用いた連携図において，機能名や情報名を記述するワークシートの役割を果たす．

二次元行列表：M(y,x)

y, x は y=x，y=1～n，x=1～n の正方行列である．また，機能数を n とすると行・列の最大値は n と等しい．

連携図は，図 5.6 に示すように二次元の行列表 M(y,x)で示すことができる．この，行列上の行を y 軸，列を x 軸と呼び，行列表上の任意の交点をセル(cell) C と呼ぶ．セルは C(y=1, x=1), C(y=2, x=2),…, C(y=n, x=n) のように y 軸と x 軸の座標軸でその位置を表す．

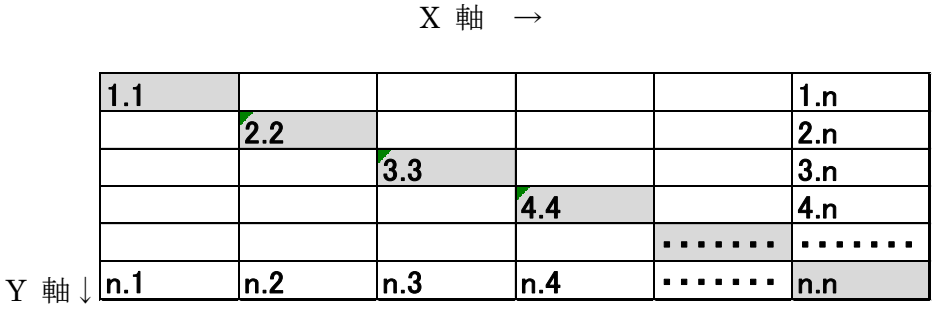


図 5.6　二次元行列表の例 M(y,x)

5.2.3 機能

機能は，次のように二次元行列表上に表現される．

機能：E(e), e=1, 2, …, n

機能につける番号(e)は表記上の位置を示すものであって，機能の意味を示すものではない．また，n は機能の最大数で利用者が任意に定義する．

機能は，図 5.7 に示すように二次元行列表の，y 軸，x 軸の対角上の交点（たとえば，C(y=1, x=1), C(y=2, x=2), …, C(y=n, x=n)の位置）に配列する．そこで，

この配列されたセルに，機能名を C(y=1, x=1)=E(1), C(y=2, x=2)=E(2), C(y=3, x=3)=E(3), …, C(y=n, x=n)=E(n)の順に格納する．

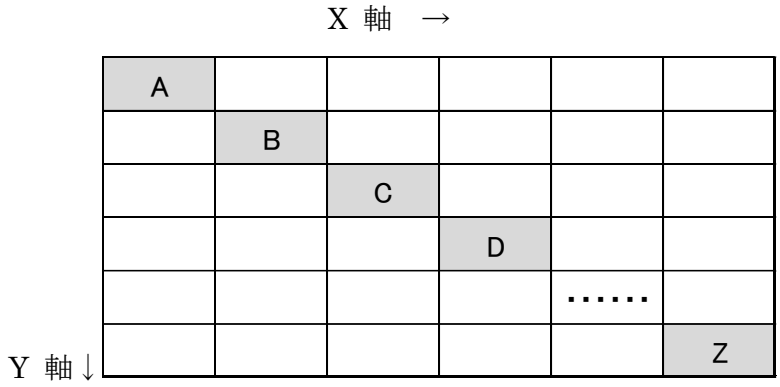


図 5.7　機能の配列の例

5.2.4 情報連携

情報連携は，機能間の情報の関係を表す．情報連携は，次のように二次元行列表上に表現される．

情報連携：R(r), r=1, 2, …, m

情報連携につける番号(r)は表記上の唯一性（重複性排除）を示すものであって，情報連携の意味を示すものではない．また，m は情報連携の最大数で利用者が任意に定義する．情報連携には，発信元機能と受信先機能が存在する．

情報連携の記述ルールは y 軸(同一列の上下)方向が情報連携の発信元の機能，x 軸（同一行の左右）方向が情報連携の受信先の機能とする．これにより情報連携は反時計回り（左回り）に一方通行で表現できる．情報連携を中心に機能を表現する場合，情報名は二次元行列表の対角上に配置済みの発信元機能の y 軸方向と受信先機能の x 軸方向が交わるセル中に記述する．機能を中心に情報連携を表現する場合，機能 E(x)から機能 E(y)への情報連携 R(m)はセル C(y, x)に配置する．ある機能からの情報連携の発信は y 軸方向のセル（同一列の上下方向）を用いて他の機能への発信を表現する．ある機能への情報連携の受信は x 軸方向のセル(同一行の左右方向）を用いて他の機能からの受信を表現する．

具体的には，図 5.8 に示すように機能 E(3)を配置するセルは C(y=3, x=3) である．機能 E(6)を配置するセルは C(y=6,x=6) である．機能 E(3)=C から機能 E(6)=F

への情報連携 R(1)の情報名はセル C(y=6, x=3)に記述する．同様に機能 E(6)=F から機能 E(3)=C への情報連携 R(2)の情報名はセル C(y=3, x=6)に記述する．

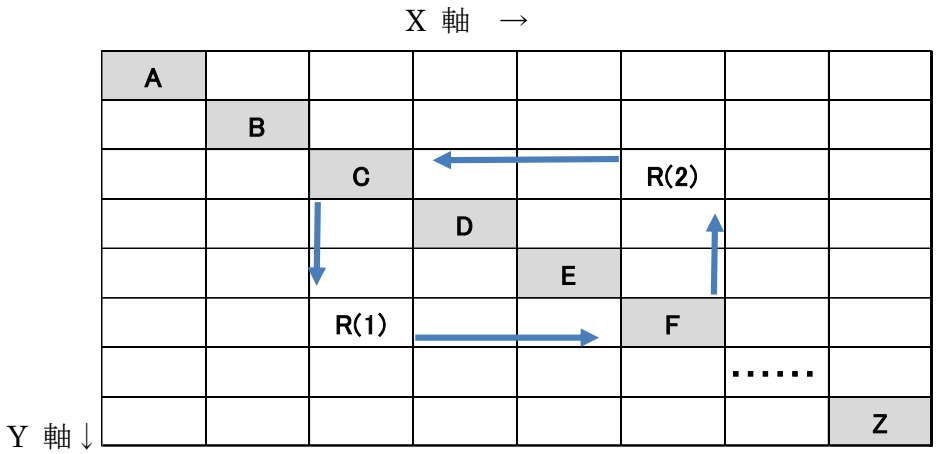


図 5.8　情報連携の例（1）

情報名を記述するセル C(y,x)中に異なる意味の複数の情報連携の情報名が存在してもよい．また，同じ情報名 R(r)が異なる複数の機能 E(e)と情報連携しても良い．たとえば，図 5.9 に示すように，セル C(3,2)には情報連携 R(3)と情報連携 R(5)の二つの情報名が記述されているが，いずれの情報名も機能 B から機能 C へ情報連携していることを示している．また，機能 F から機能 B と機能 C の二つの機能に対して同一内容の情報名 R(2)が情報連携していることを示している．

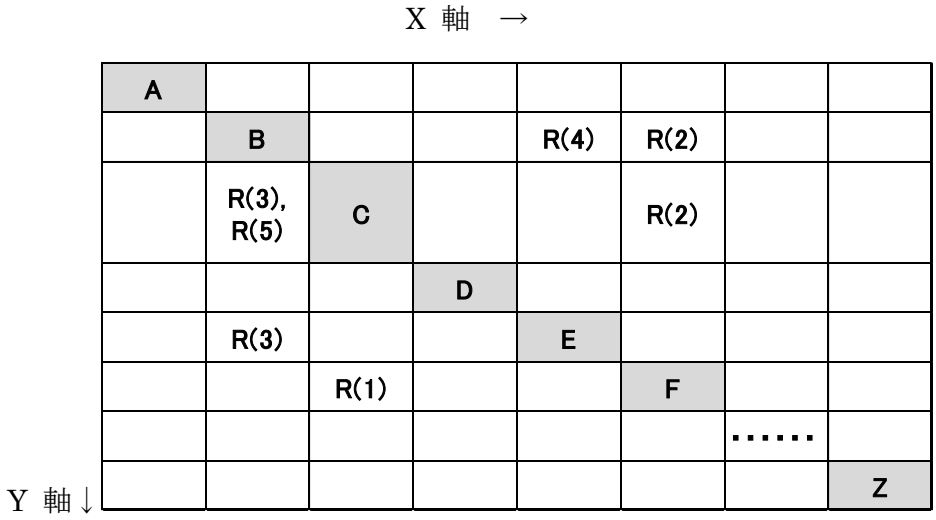


図 5.9　情報連携の例（2）

情報連携の記述ルールにより，ある y 軸（同一列の上下方向）を取りだすと，その列の対角位置に記されている機能が発信元になっている情報連携のすべてを一覧として表現することになる．また，ある x 軸（同一行の左右方向）を取りだすと，その行の対角位置に記されている機能が受信先になっている情報連携のすべてを一覧として表現することになる．

したがって，MDM は，機能の数が決まると，その機能間の情報連携の可能性のすべてを一覧として表現することができるという特徴をもつ．

5.2.5 MDM による連携図の表記に関する解決

第 3 章で，DFD の表記方法上の制約から煩雑性を起こしている原因を確認し，その原因を取り除く着眼を示した．ここでは，連携図を記述する際の課題を求める着眼-1 から着眼-5 を再掲し，それぞれの着眼に対応する MDM の解決を示す．

最初に，「1 枚のシートに書ける機能数と情報名数は限られる」を取り除く着眼としてあげた，着眼-1 から着眼-3 に対応する MDM の解決を示す．

着眼-1 機能を記述する位置を決める．

解決-1 機能は図 5.7 に示すように，二次元正方行列の対角上に配置する．

着眼-2 情報連携を記述する位置を決める．

解決-2 情報連携は図 5.8 に示すように，発信元機能の列と受信先機能の行の交点に配置する．

着眼-3 情報名をすべて書けるようにする．

解決-3 図 5.8 に示すように，行列のセルを情報名の記入欄とするため，機能の数が決まると，発生する可能性のあるすべての情報連携を網羅することができる．

このように，MDM のもつ機能と情報連携を記述する位置が決まるという特徴と機能間の情報連携のすべてを表現することができるという特徴により，連携図を記述する際におこる「1 枚のシートに書ける機能数と情報名数は限られる」原因を取り除くことができる．

次に，「矢印線で情報連携を表現しているため，線が交差する」を取り除く着眼としてあげた，着眼-4,着眼-5 に対応する MDM の解決を示す．

着眼-4 情報連携を線がなくても表現できる.

解決-4 図 5.8 に示すように，情報名を記述する位置が決まるため，矢印線が不要となる.

着眼-5 情報連携の向きを矢印がなくても表現できる.

解決-5 情報連携の向きは図 5.8 に示すように, 反時計回りの一方通行で表現する.

このように，MDM のもつ情報名を記述する位置が決まるという特徴と情報連携の方向が反時計回りの一方通行となるという特徴により，連携図を記述する際におこる「矢印線で情報連携を表現しているため，線が交差する」原因を取り除くことができる.

なお，フローチャート型表記方法では連携図を見やすくするため，1 枚のシートに記述する機能数を制限する．たとえば，IDEF 0 では 1 枚のシートに記述する機能を 3 個以上 6 個以内とすることが推奨されている．これに対して，MDM を用いると，機能名を文字サイズ 9 ポイント，6 文字で表現する場合，A4 サイズのシートに 18 個の機能, 306 個の情報連携が収容可能である. したがって, MDM の収容可能な機能数を IDEF 0 と比較すると，3 倍から 6 倍の収容能力をもつことがわかる.

加えて, MDM の二次元正方行列の特性を活かすと, 行・列を追加, あるいは, 入れ替えても，情報連携は保たれるという特徴がある.

この MDM のもつ特徴を活用すると，機能の分解や並び順変更といった操作ができるようになる.

5.3 MDM の操作例

5.3.1 階層を用いた機能の分解

MDM を使うと，機能と情報連携は階層により分解できる．階層の種類には機能の階層と情報連携の階層がある.

機能×階層：E(e, le)

情報連携×階層：R(r, lr)

機能の階層（le)：le=1, 2, …, j

情報連携の階層（lr)：lr=1, 2, …, k

階層につける番号(le),(lr)は階層の相対的な深さを示す．また，階層の意味それ自体を示すものではない．

ある機能 E(e)の階層(le)と，ある情報連携 R(r)の階層(lr)が同じ階層番号を用いていても，階層の深さの意味は，ある機能 E(e)と，ある情報連携 R(r)のそれぞれの用い方によって異なるので,階層の最大の深さ j,k は利用者が任意に定義する．しかし，ある機能 E(e)内における階層番号(le)は，統一した階層の深さと定義で用いることが望ましい．同様に，ある情報連携 R(r)内における階層番号(lr)は，統一した階層の深さと定義で用いることが望ましい．

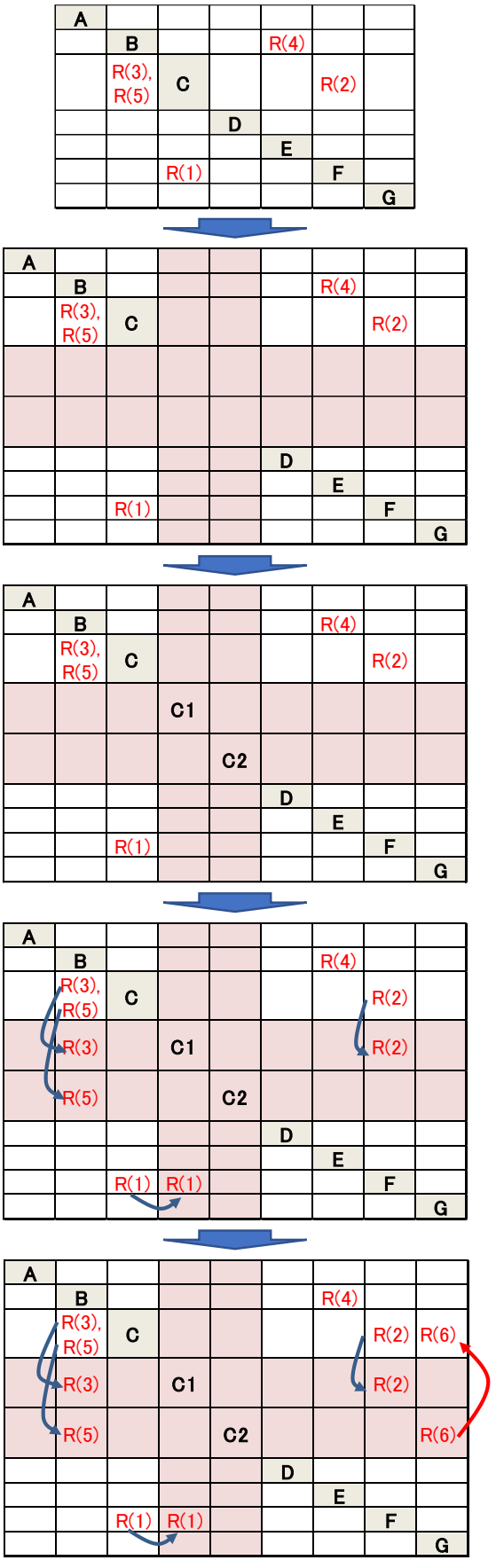
機能 E(e)を階層(le)により分解する例を図 5.10 に示す．

この例では機能“C”の階層を 1 階層だけ分解する．まず，機能“C”の後の行と後の列に分解後の機能数分の行と列を追加する．この例では 2 行，2 列を追加している．次に，追加した行列の対角部分に分解後の機能名“C1”,“C2”を追加する．そして，機能“C”が受信している情報連携 R(2), R(3), R(5)を受信先となる分解後の機能の行に複写する．同様に機能“C”が発信している情報連携 R(1)を発信元となる分解後の機能の列に複写する．こうして，1 つの機能“C”を分解することができる．

次に，分解後の連携 R(r)の見直しを行う．見直した結果，分解後の機能と他の機能の間に新たな情報連携が追加される場合がある．この例では，分解後の機能“C2”と機能“G”の間に新たな情報連携 R(6)が追加されている．追加した情報連携 R(6)は，上位階層の機能“C”にも同時に反映する．

さらに，別の機能を分解する例を図 5.11 に示す．この例では，機能“F”の階層を 1 階層だけ分解する．この例では 2 行，2 列を追加している．次に，追加した行列の対角部分に分解後の機能名“F1”,“F2”を追加する．そして，機能“F”が受信している情報連携 R(1)を受信先となる分解後の機能“F1”の行に複写する．同様に機能“F”が発信している情報連携 R(2)を発信元となる分解後の機能“F2”の列に複写する．こうして，1 つの機能“F”を分解することができる．この図 5.11 の連携図をみると，情報連携 R(1)は“C1”と“F1”の行・列の交点に記述し，分解後の下位機能同士を直接つないでいることがわかる．同様に，情報連携 R(2)は“F2”と“C1”の行・列の交点に記述し，分解後の下位機能同士を直接つないでいることがわかる．

以上の操作を繰り返して機能 E(e)の分解を行う．



(1) 分解前の連携図

(2) 分解後の機能数分の行・列を追加

(3) 分解後の機能名を追加

(4) 上位の機能が受発信している情報連携を分解後の機能の行・列に複写

(5) 追加した情報連携を上位の機能に複写

図 5.10 階層による分解の記述例

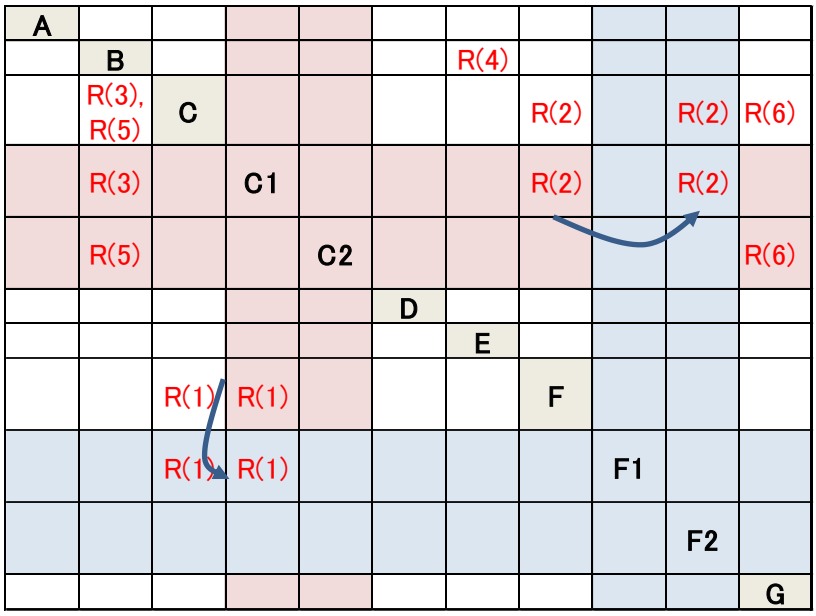


図 5.11　2 つの機能を分解した例

5.3.2 フラットな階層表現

階層を使って分解前の機能と分解後の機能に上下関係があることを示すため，MDM では，分解前の機能と右下の分解後の機能を含む（行×列)範囲を明示的に枠で囲む操作がある．

図 5.11 で示した分解後の連携図において，図 5.12 のように，分解前の機能“C”と右下の分解後の機能“C1”，“C2”を含む（行×列)範囲を明示的に枠で囲む．この枠によって，左上の機能“C”と右下の機能“C1”，“C2”に上下関係があることがわかる．同様に，分解前の機能“F”と右下の分解後の機能“F1”，“F2”を含む（行×列)範囲を明示的に枠で囲む．この枠によって，左上の機能“F”と右下の機能“F1”，“F2”に上下関係があることがわかる．

このように，MDM では，階層を使った機能の上下関係をフラットに表現することができる．したがって，3.3 節で指摘した情報連携表記方法の具備要件の 1 つである「階層の上下関係が一覧できること」を満たすことができる．

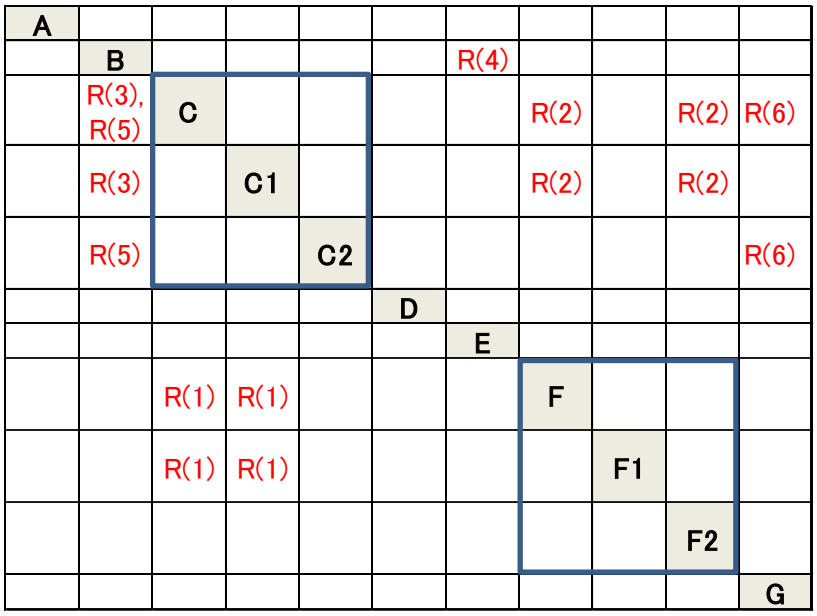


図 5.12 枠によるフラットな階層表現

5.3.3 機能の並び順変更

MDM がもつ，機能は二次元正方行列の対角上に配置するという特徴を活用すると機能の並び順を変更することが容易にできる．たとえば，図 5.8 で示した連携図の機能“F”を機能“E”の前に並べ替える例を図 5.13 に示す．この例では，機能“E”の前の行と列にそれぞれ 1 行 1 列を追加する．そして，元の機能“F”の行を追加した新しい行に複写する．続いて，元の機能“F”の列を追加した新しい列に複写する．その後，元の機能“F”の行と列を削除する．こうして，機能“F”を機能“E”の前に並べ替えることができる．

MDM がもつ二次元正方行列の特性を活かすと，行・列の追加や入れ替えを行っても，機能間の情報連携は保たれる．この特徴を活用すると，機能間の情報連携を保ったまま，機能の並び順変更ができるようになる．

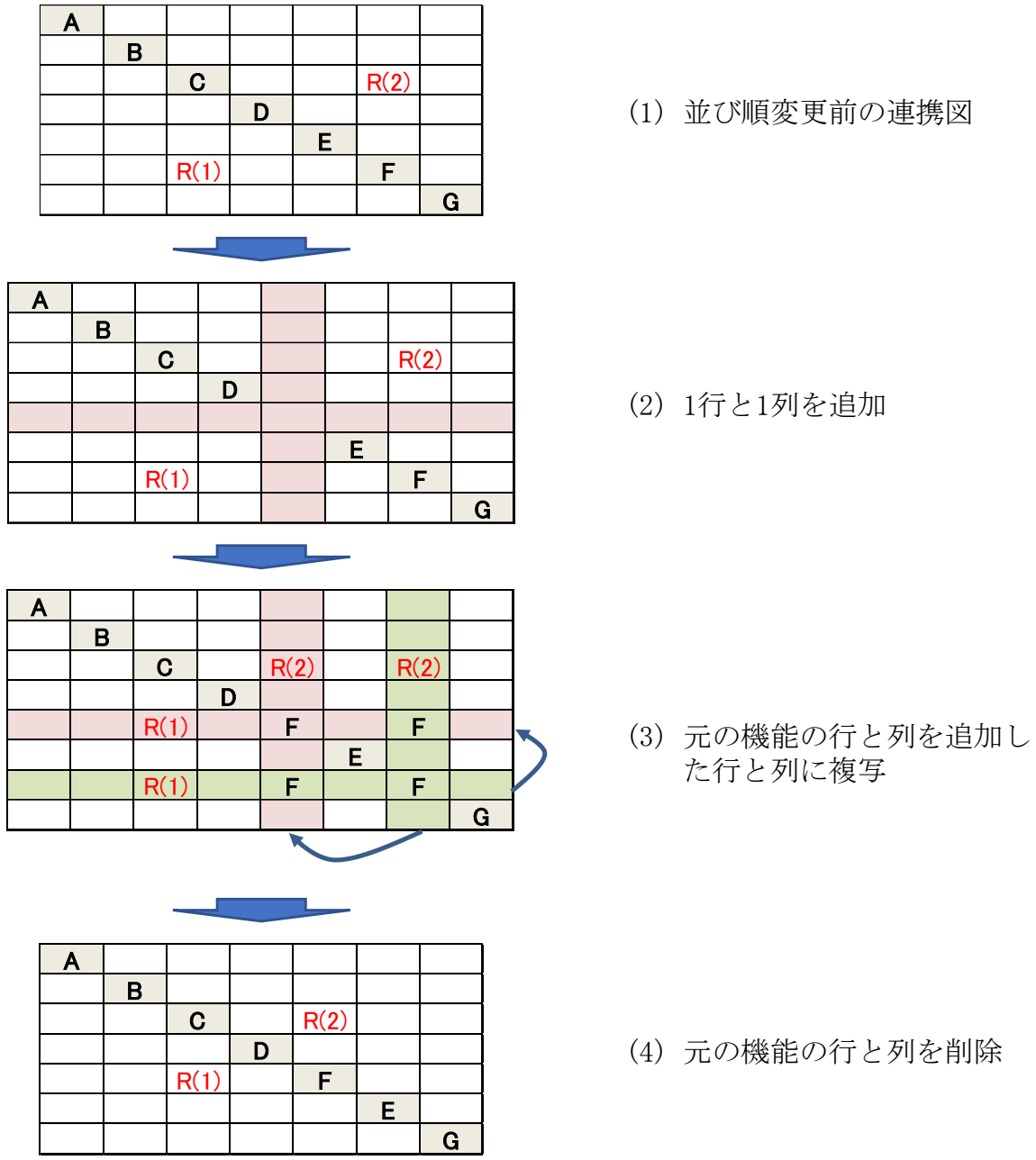


図 5.13　機能の並び順変更の例

5.3.4 複数連携図の結合

MDM を用いた連携図は二次元正方行列で作成されている．その二次元行列表の右下方の対角(y=n+1, x=n+1)を起点に，図 5.14 に示すように追加したい連携図を右下方に追加していくと複数の連携図を同一連携図として結合できる．

これにより，追加前の連携図と追加後の連携図に新たな情報連携を追加する行列領域が生まれる．また，5.3.3 項で示す機能の並び順変更を活用することにより

新しくできた連携図を 1 つの連携図として扱うことが可能になる.

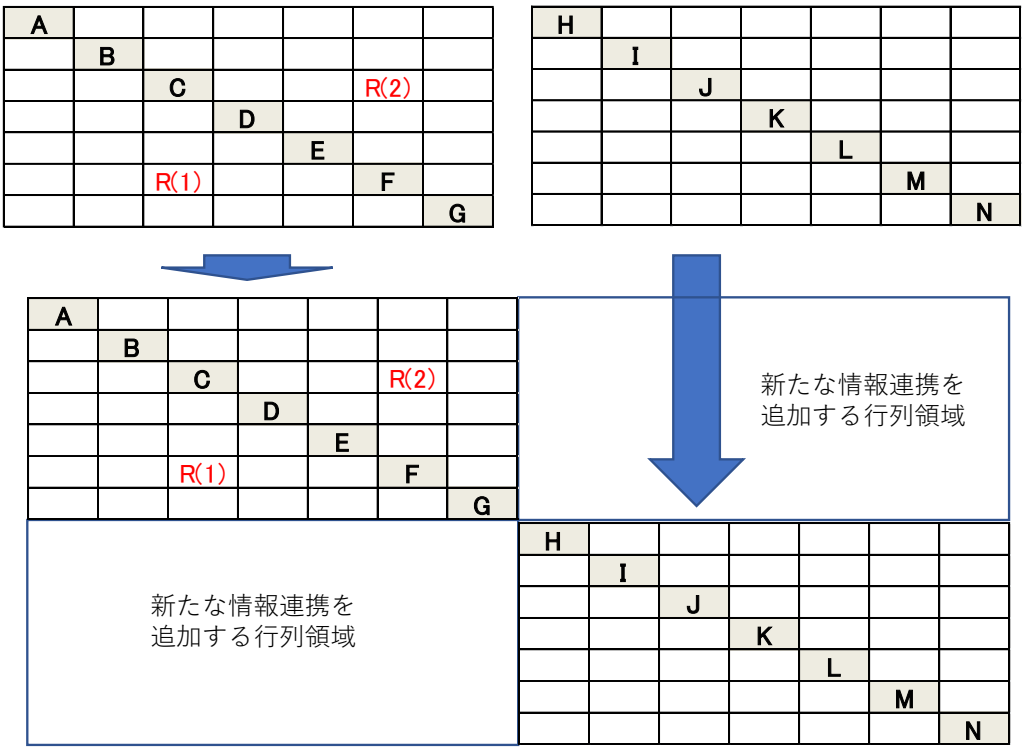


図 5.14　複数連携図の結合の例

5.3.5 MDM による分解結果の表記に関する解決

第 3 章で，DFD の表記方法上の制約から煩雑性を起こしている原因を確認し，その原因を取り除く着眼を示した．ここでは，機能を分解した結果を記述する際におこる「分解した機能の連携先は相手の上位機能しかわからない」原因を取り除く着眼-6 から着眼-8 を再掲し，それぞれの着眼に対応する MDM の解決を示す.

着眼-6　分解した機能の上下関係がわかる.

解決-6　枠を用いたフラットな階層表現により，図 5.12 で示すように分解した機能の上下関係が把握できる.

着眼-7　分解した下位機能と別の上位機能との情報連携がわかる.

解決-7　分解した下位機能と別の上位機能との情報名は，図 5.10 で示すように 2 つの機能の行・列の交点のセル中に記述できる.

着眼-8　分解した下位機能同士の情報連携がわかる.

解決-8 分解した下位機能同士の情報名は，図 5.11 で示すように 2 つの機能の行・列の交点のセル中に記述できる．

このように，MDM のもつ機能の階層をフラットに表現できる特徴と情報名は 2 つの機能の行・列の交点のセル中に記述できる特徴により，機能を分解した結果を記述する際におこる「分解した機能の連携先は相手の上位機能しかわからない」原因を取り除くことができる．

5.4 MDM による課題解決のまとめ

MDM を活用すると，第 3 章で指摘した情報連携表記方法上の課題は，8 つの着眼にそれぞれ対応した「MDM による課題解決の仕組み」により，表 5.1 のように解決できることがわかる．

連携図の記述に関する課題に対応して，次の 5 つの仕組みで課題を解決する．

(1) 機能は二次元正方行列の対角上に配置する．

(2) 情報連携は発信元機能の列と受信先機能の行の交点に配置する．

(3) 行列のセルを情報名の記入欄とするため，発生する可能性のあるすべての情報連携を網羅することができる．

(4) 情報名を記述する位置が決まるため，矢印線が不要になる．

(5) 情報連携は反時計回りの一方通行で表現する．

機能の分解結果の記述に関する課題に対応して，次の 3 つの仕組みで課題を解決する．

(6) 枠を用いたフラットな階層表現により，分解した機能の上下関係が把握できる．

(7) 分解した下位機能と別の上位機能との行・列の交点に情報名を記述できる．

(8) 分解した下位機能同士の行・列の交点に情報名を記述できる．

また，MDM を活用すると，第 3 章で示した情報連携表記方法に求められる 2 つの具備要件を満たしていることが確認できる．

１つめの「機能と情報連携を記述する位置が明示的に示せること」という具備要件は,「機能と情報連携を記述する位置にアドレスをつけた二次元正方行列を用いる」ことにより満たすことができる．

２つめの「階層の上下関係が一覧できること」という具備要件は,「階層を使った機能の上下関係をフラットに表現する」ことにより満たすことができる.

表 5.1　表記方法の課題と MDM による解決

本研究の課題	着眼No	従来の表記方法				提案の表記方法	
		DSM	from-to chart	スイムレーン	DMM	MDM	MDMによる課題解決の仕組み
連携図の記述に関する課題	1. 機能を記述する位置を決める.	○	○	○	○	○	機能は二次元正方行列の対角上に配置する.
	2. 情報連携を記述する位置を決める.	○	○			○	情報連携は発信元機能の列と受信先機能の行の交点に配置する.
	3. 情報名をすべて書けるようにする.		○			○	行列のセルを情報名の記入欄とするため，発生する可能性のあるすべての情報連携を網羅することができる.
	4. 情報連携を線がなくても表現できる.	○	○			○	情報名を記述する位置が決まるため，矢印線が不要になる. .
	5. 情報連携の向きを矢印がなくても表現できる.	○	○			○	情報連携は反時計回りの一方通行で表現する.
機能の分解結果の記述に関する課題	6. 分解した機能の上下関係がわかる.	○		○	○	○	枠を用いたフラットな階層表現により，分解した機能の上下関係が把握できる.
	7. 分解した下位機能と別の上位機能との情報連携がわかる.			○		○	分解した下位機能と別の上位機能との行・列の交点に情報名を記述できる.
	8. 分解した下位機能同士の情報連携がわかる.			○		○	分解した下位機能同士の行・列の交点に情報名を記述できる.

次章では，これらの「MDM による課題解決の仕組み」の有効性を，情報システム開発に対する実用性という観点から確認する.

第６章　MDM の有効性確認

6.1 例題を用いた問題解決の確認

6.1.1 問題解決確認の進め方

MDM によって，フローチャート型表記方法の DFD の問題が解決されることを確認する．確認方法は，第 2 章の問題指摘で用いた「あるレストランチェーン A 社において，食材発注に電子取引を導入するためのシステム改修」の例題を，MDM を用いて記述することによる．その際，図 2.3 に示した進め方と同じ手順で，MDM を用いた連携図を作成する．

6.1.2 改修前の連携図(MDM)の作成

最初に，図 2.3 の「Step1 改修前の連携図の作成」を行う．

図 2.4 に示した定義文から，機能と情報連携を取りだし，MDM を用いて図 6.1 のような電子取引導入前の連携図を作成する．

発注・検収	納品書		
食材発注	**食材提供**		支払
納品書	請求	**食材調達**	
	領収書	支払依頼	**出納**

図 6.1 電子取引導入前の連携図(MDM)

6.1.3 機能を追加した連携図(MDM)の作成

次に，図 2.3 の「Step2 機能を追加した連携図の作成」を行う．

図 2.6 に示した定義文から，機能と情報連携を取りだし，図 6.1 で示した電子取引導入前の連携図に追記する．この際，連携図を新たに書き直す必要はない．

電子取引導入後の連携図は，図 6.2 のような手順で作成する．

(1) 図 6.1 の電子取引導入前の連携図の右下に行・列を追加し，対角上に新たな機能である電子取引機能を追加する．

(2) 食材発注を電子取引機能経由に見直し，電子取引機能と発注・検収機能間，および，電子取引機能と食材提供機能間の食材発注に関する情報連携の追加，つなぎ換えを行う．

(3) 検収，納品に関する情報連携の追加，削除を行う．

(4) 手数料支払いなどの電子取引機能との情報連携を追加する．

作成した連携図をみると，電子取引導入前に行われていた情報連携に変更がない場合，その情報連携はそのまま保たれていることがわかる．

この Step 2 のケースから，第 2 章では煩雑性-1 として指摘した，DFD を用いて新たな連携図を作成する際の表記方法上の問題を再掲する．

煩雑性-1（再掲）

現象　情報連携の増加に伴い，矢印線が交差する．

原因　フローチャート型表記方法は情報連携を矢印線で表す．

結果　情報連携がたどりにくくなる．

この煩雑性-1 に対して，MDM では次のように解決できていることが確認できた．

- 矢印線を使わないため，連携図における矢印線の交差を考慮する必要がない．
- 機能を追加する位置が二次元正方行列の対角上と決まっているため，機能の追加によって連携図全体の機能の配置を変える必要がない．
- 既存の連携図から変化がない機能や情報連携は書き直す必要がない．

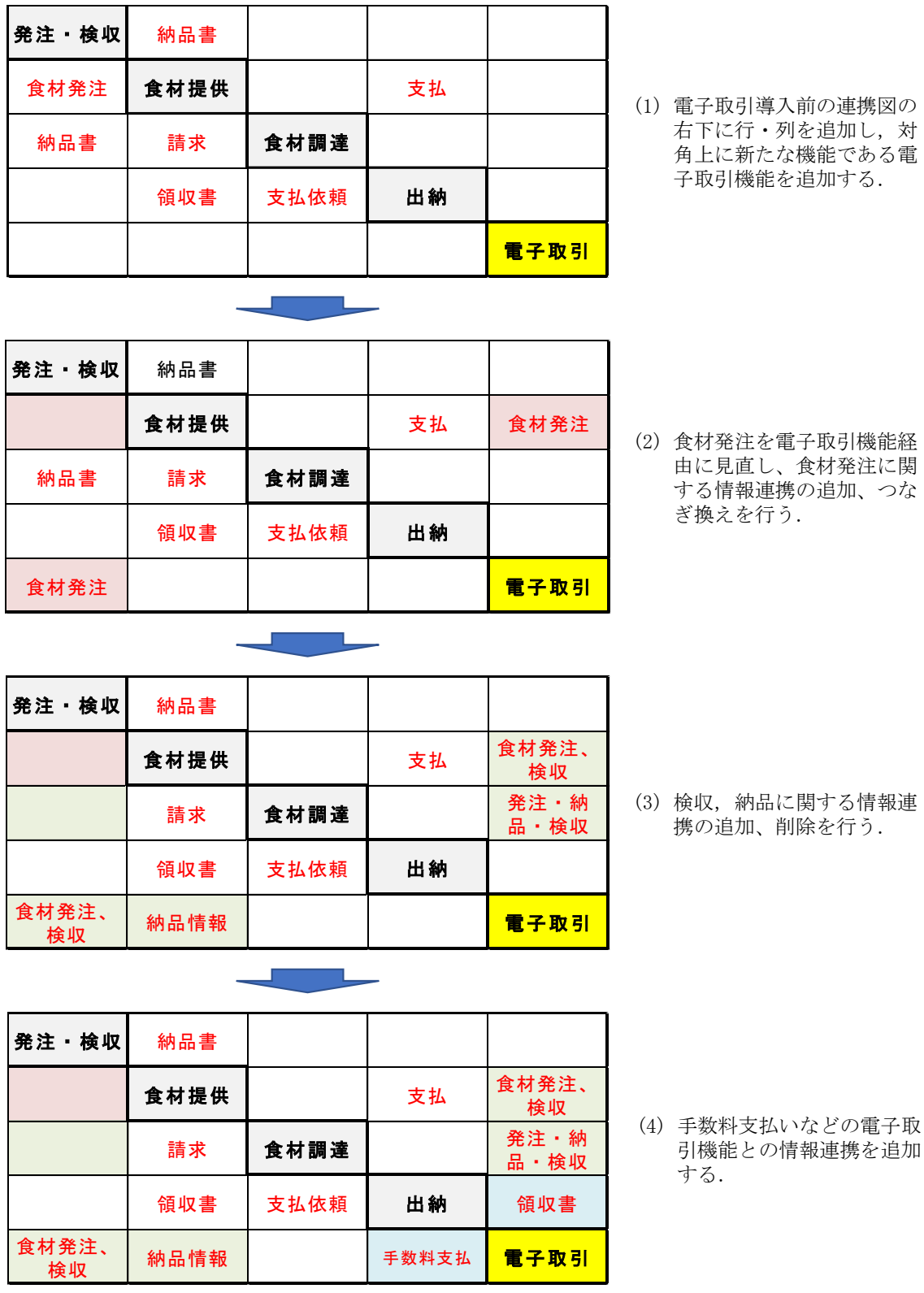


図 6.2 電子取引導入後の連携図の作成手順(MDM)

6.1.4 追加した機能を分解した連携図(MDM)の作成

機能を追加した連携図に対して，図 2.3 の「Step3 追加した機能を分解した連携図の作成」を行う．

MDM を用いて，電子取引導入後の連携図の電子取引機能を分解した連携図は，図 6.3 のような手順で作成する．

(1) 電子取引機能の右下に行・列を追加し，対角上にデータ交換機能，食材マスタ登録を追加する．また，階層をフラットに表現するため，分解前の電子取引機能，分解後のデータ交換機能，食材マスタ登録機能を含む(行×列)範囲を枠で囲む．

(2) 電子取引機能に関する情報連携をデータ交換機能の情報連携に複写する．

(3) 食材マスタ登録に関する情報連携を追記する．

(4) 食材マスタ登録機能との情報連携を上位階層の電子取引機能に複写する．

この Step 3 のケースから，第 2 章では煩雑性-2 として指摘した，機能を分解した結果を記述する際の表記方法上の問題を再掲する．

煩雑性−２（再掲）

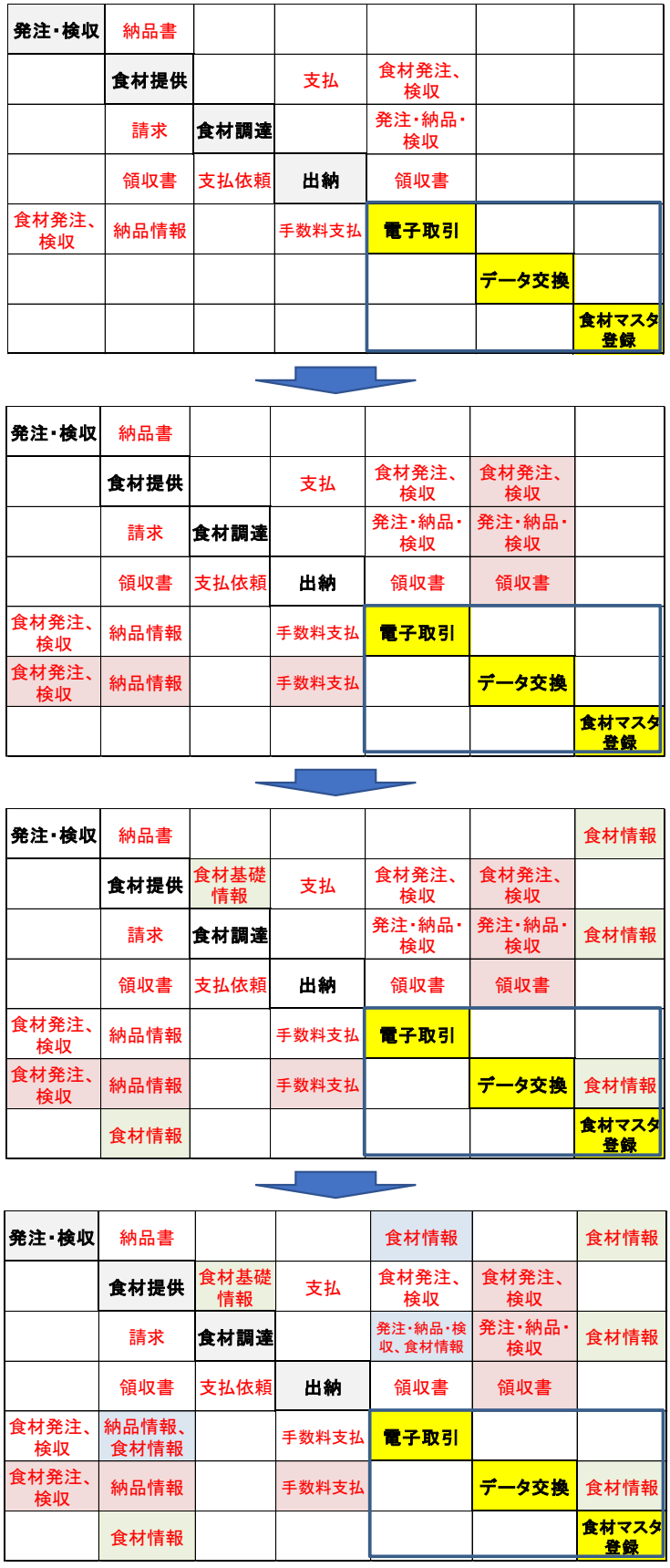
現象　機能を分解すると，機能数と情報連携数が増える．

原因　上位の機能と下位の機能にシートが分かれる．

結果　上位の機能と下位の機能の関係がわかりにくくなる．

この煩雑性-2 に対して，MDM では次のように解決できていることが確認できた．

- 機能の上下関係はフラットに表現されるため，上下関係を示す引出し線などのガイドを必要としない．
- 機能の上下関係は元の 1 枚のシートを拡張することで表現できるため，シート間の参照を必要としない．



(1) 電子取引機能の右下に行・列を追加し，データ交換機能，食材マスタ登録を追加する．また，電子取引機能，データ交換機能，食材マスタ登録機能を含む(行×列)範囲を枠で囲む．

(2) 電子取引機能に関する情報連携をデータ交換機能の情報連携に複写する．

(3) 食材マスタ登録に関する情報連携を追記する．

(4) 食材マスタ登録機能との情報連携を上位階層の電子取引機能に複写する．

図 6.3 電子取引機能を分解する手順(MDM)

6.1.5 既存の機能を分解した連携図(MDM)の作成

追加した機能を分解した連携図に対して，図 2.3 の「Step 4 既存の機能を分解した連携図の作成」を行う．

MDM を用いて，機能を分解した連携図を作成する手順は Step 3 で述べた手順と同様，図 6.4 のような手順で作成する．

(1) 発注・検収機能の右下に行・列を追加し，対角上に発注機能，検収登録，在庫管理機能を追加する．また，階層をフラットに表現するため，分解前の発注・検収機能，分解後の発注機能，検収登録，在庫管理機能を含む(行×列)範囲を枠で囲む．

(2) 発注・検収機能に関する情報連携を分解後の発注機能，検収機能に複写する．

(3) 新たな機能である在庫管理機能に関する情報連携を追記する．

この既存の機能を分解する手順において，(2)発注・検収機能に関する情報連携を分解後の発注機能，検収機能に複写する際，食材発注，検収，食材情報といった情報連携は，DFD を使った連携図では図 2.14 に示したように電子取引機能との情報連携としかわからない．ところが，MDM を使った連携図では，食材発注，検収はデータ交換機能との情報連携，食材情報は食材マスタ登録機能との情報連携というように，電子取引機能の分解後の機能との情報連携を直接把握することができる．このことは，機能の分解と同時に情報連携の分解ができるという MDM の特徴を示すものである．

この Step 4 のケースから，第 2 章では煩雑性-2 として指摘した，機能を分解した結果を記述する際の表記方法上の問題を再掲する．

煩雑性－２（再掲）

現象　機能を分解すると，機能数と情報連携数が増える．

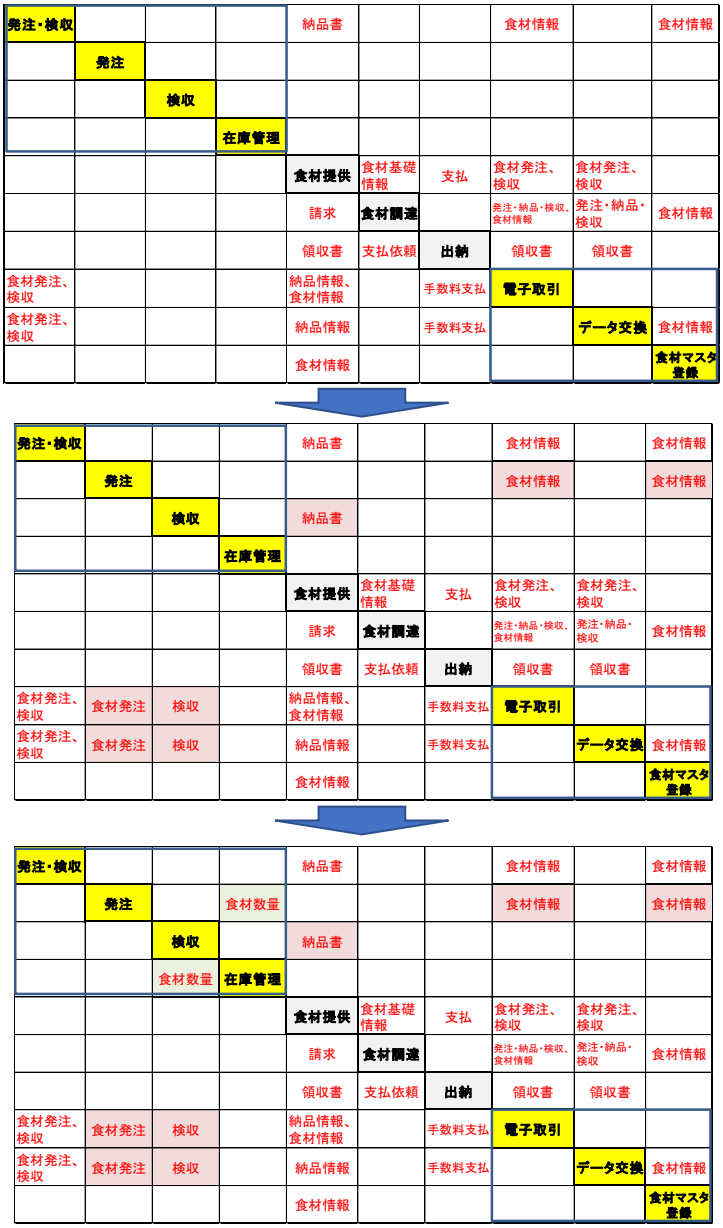
原因　上位の機能と下位の機能にシートが分かれる．

結果　上位の機能と下位の機能の関係がわかりにくくなる．

この煩雑性-2 に対して，6.1.4 項と同様に，MDM では次のように解決できていることが確認できた．

- 機能の上下関係はフラットに表現されるため，上下関係を示す引出し線などのガイドを必要としない．

・ 機能の上下関係は元の 1 枚のシートを拡張することで表現できるため，シート間の参照を必要としない．



(1) 発注・検収機能の右下に行・列を追加し，対角上に発注機能，検収登録，在庫管理を追加する．また，発注・検収機能，発注機能，検収登録，在庫管理を含む(行×列)範囲を枠で囲む．

(2) 発注・検収機能に関する情報連携を分解した発注機能，検収機能に複写する．

(3) 新たな機能である在庫管理機能に関する情報連携を追記する．

図 6.4 発注・検収機能を分解する手順(MDM)

6.1.6 複数の機能を分解した連携図(MDM)の確認

DFD を用いて，複数の機能を分解した連携図に情報連携のエラーがないか確認する場合，図 2.3 の「Step5 複数の機能を分解した連携図の統合」を行う必要がある．一方，MDM を用いると，図 6.1 から図 6.4 に示した手順を経て，図 6.5 に示すような複数の機能を分解した連携図を作成することができる．この図 6.5

に示す連携図は，図 6.4 に示す連携図の最終形と同じものである．したがって，MDM を用いて，複数の機能を分解した連携図に情報連携のエラーがないか確認する場合，DFD を用いた場合に必要であった図 2.16 に示したような連携図の統合は不要となる．

また，MDM を用いた図 6.5 では，分解後の下位機能同士の行・列の交点に情報名が記述されていることがわかる．たとえば，発注機能とデータ交換機能との行・列の交点に食材発注という情報名が記述されている，同様に，検収機能とデータ交換機能との行・列の交点に検収，食材マスタ登録機能と発注機能との行・列の交点に食材情報という情報名が記述されている，そのため，MDM を用いると，下位機能同士の情報連携は，上位の連携図が記述されたシートや下位の連携図が記述された別のシートを参照することなく，直接たどることができる．

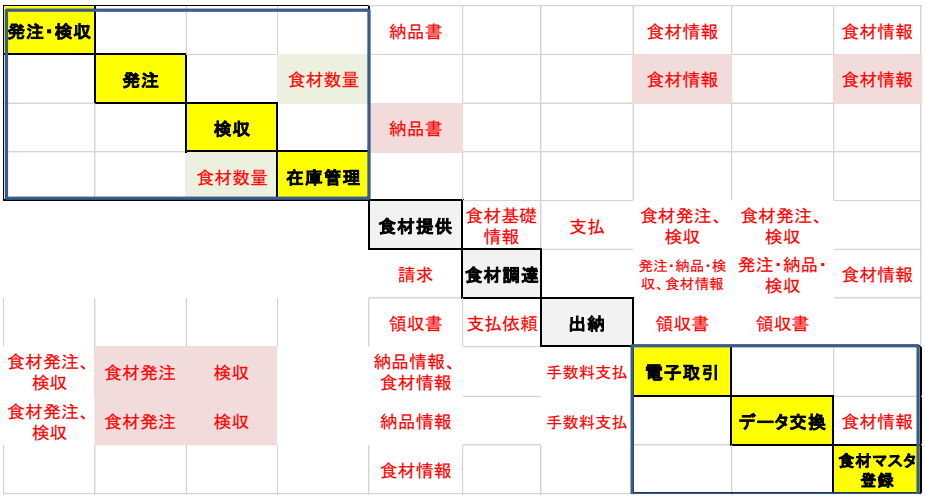


図 6.5 複数の機能を分解した連携図(MDM)

この Step 5 のケースから，第 2 章では煩雑性-3 として指摘した，分解後の下位機能同士で情報連携のエラーがないか確認する際の表記方法上の問題を再掲する．

煩雑性-3(再掲)

現象　下位機能同士の情報連携を確認する際，上位の連携図を参照する必要がある．

原因　下位機能同士の情報連携の相手先機能が上位機能名で表現されている．

結果　下位機能同士の情報連携は直接たどれない.

この煩雑性-3 に対して，MDM では次のように解決できていることが確認できた.

・フラットな階層表現により，分解前の機能と分解後の機能を 1 枚のシート上に記述することができる.

・分解した下位機能同士の行・列の交点に情報名を記述できる．そのため，下位機能同士の情報連携を直接たどることができる.

以上より，MDM によって，フローチャート型表記方法の DFD の煩雑性-1 から煩雑性-3 のすべてが解決することを確認できた.

6.1.7 分解した機能の折りたたみ

MDM は行列を用いているため，DFD に比べ，1 枚のシートに多くの機能や情報連携を記述することができる．ところが，図 6.5 のように分解した機能や情報連携をすべて表示すると煩雑に見えることがある．また，MDM を用いて機能や情報連携の分解を進めていくと，階層が深くなる．それと同時に，機能や情報連携を記述する二次元行列表が大きくなり，情報連携がたどりにくくなるという DFD と同じような問題が起こる．この問題は，スプレッドシートのグループ化機能を活用して，解決する．たとえば，図 6.6 のように，図 6.5 で示した分解した機能を折りたたむことが可能になる．なお，図 6.6 中の矢印(←→)はグループ化する行と列の範囲を表す.

食材マスタ登録機能と発注機能との間の食材情報のように下位階層で新たに追加した情報連携は，上位階層の電子取引機能と発注・検収機能との間の情報連携としても反映されている．したがって，分解した機能を折りたたんでも，下位機能同士の情報連携を把握することが可能になる.

このように，分解した機能と情報連携の行・列を折りたたむことにより，情報量を少なくして，見やすくすることができる．なお，さらに機能や情報連携の分解を進める場合は，グループ化を解除し，図 6.5 の状態にもどす.

DFD を用いた連携図では，図 2.17 の連携図のように，上位の連携図が記述されたシートと下位の連携図が記述されたシートに分かれる．そのため，図 2.17 の(1)のように，上位の連携図をみても，下位機能同士の情報名はわからない．ま

た，図 2.17 の(5)のように機能名を変更すると，下位機能同士の情報連携をたどる際に，上位の機能名はわからない．一方，MDM を用いた連携図では，分解した機能を折りたたんでも，下位機能同士の情報名を把握することができる．また，下位機能同士の情報連携をたどる際には，グループ化を解除するため，同一枠内の上位の機能名を把握することができる．

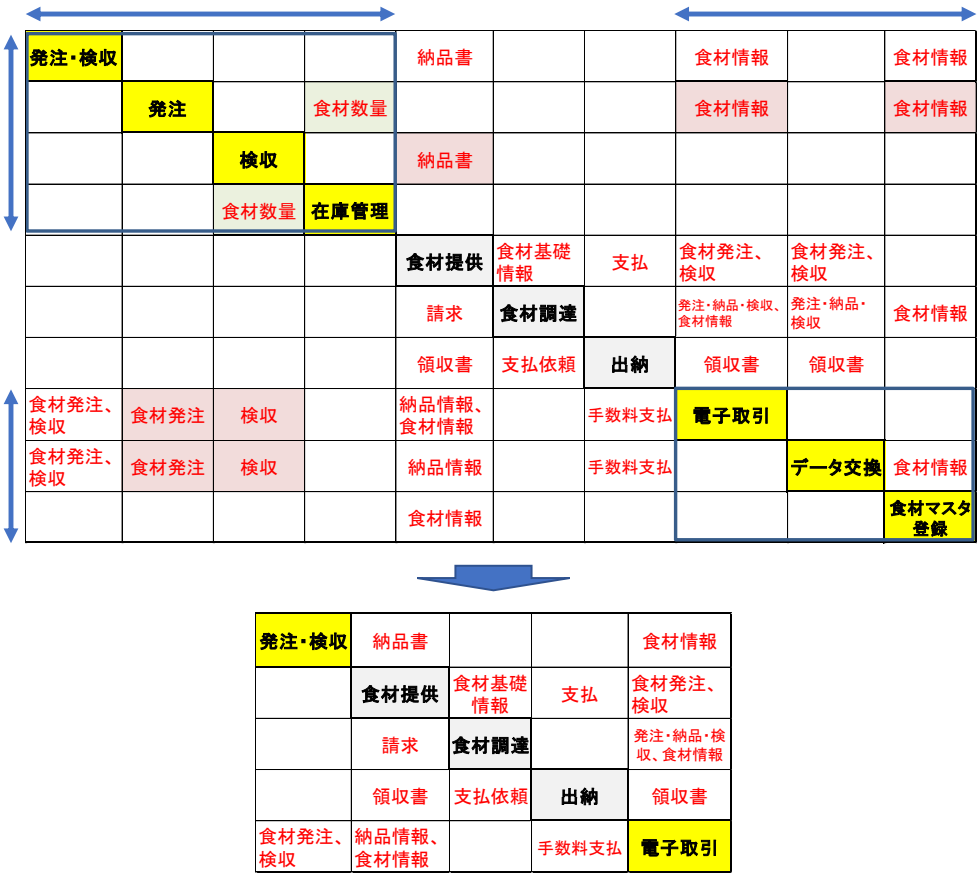


図 6.6　分解した機能の折りたたみ

6.1.8 機能の並び順変更

MDM を使うと，見やすくするために，連携図を書き直すことなく，機能の並び順を変更することができる．

機能の並び順を変更した例を図 6.7 に示す．図 6.7 では，電子取引機能を，食材発注に関連がある発注・検収機能と食材提供機能の間に配置するように，機能の並び順変更を行っている．なお，変更後の機能の並び順は図 2.7 と同じである．

図 6.7 機能の並び順変更

6.2 初学者対象の実験による効果測定

6.2.1 初学者を対象とした実験の目的

初学者を対象とした連携図を記述する実験により，MDM の表記方法上の特徴を明らかにする．また，DFD との比較実験により，連携図を記述する際の MDM の効果を確認する．MDM は二次元正方行列を用いていることから，機能の数が決まると，その機能間で発生する可能性のあるすべての情報連携を一覧として表現することができるという特徴をもつ．この特徴を活かすと，MDM は DFD に比べて，より正確に機能ならびに機能間の情報連携を抽出・記述することができると考える．また，MDM は機能や情報連携を配置する位置が決まっており，情報連携を表現するための矢印線が不要であるという特徴をもつ．したがって，連携図を見やすくするため，機能や情報連携を配置する位置を変えたり，交差している矢印線を書き直したりする手間がかからない．この特徴を活かすと，MDM は DFD に比べて，より短い時間で連携図を記述することができると考える．

本実験の目的は， MDM が機能ならびに機能間の情報連携をより正確に抽出・記述できることやより短い時間で連携図を記述することができることを, DFD との比較実験を通して明らかにすることである[32].

6.2.2 初学者を対象とした実験の方法

(1) 被験者と題材を選定する考え方

実験の被験者は，大学でプロジェクトマネジメントを専攻する大学生とする．情報システム開発の流れは理解しているが，様々な業務プロセスを分析した経験は乏しい．被験者は，同じ研究室に所属する大学生を選定しており，能力差は少ないと考えられる．

実験の題材には，初学者でもイメージしやすいことを考慮して，外食チェーンの業務プロセスを選定する．この題材では，仕入先，本社，店舗，顧客といったサプライチェーンの各段階において，相互に情報連携が発生する．

(2) 実験手順の考え方

実験で MDM と比較する表記方法として，DFD を採用する．DFD は，情報システム開発に用いられる表記方法として，広く知られた実績ある手法である．また，DFD は MDM の行列型とは異なるフローチャート型の表記方法であり，特

徴の違いを比較可能であると考える．また，被験者である大学生が講義ですでに学習している点も考慮する．

6.2.3 初学者を対象とした実験条件

初学者を対象とした実験の実施条件を表 6.1 に示す．

被験者の母集団はプロジェクトマネジメントを専攻する大学 3,4 年生 12 名である．この 12 名を 2 名 1 組の 6 班に分け，3 班ずつ DFD，MDM を使って，連携図を記述する．DFD，MDM を使った 3 班のうち，機能名，情報名が書かれた定義文の読取り能力が低い各々1 班を実験の評価対象から除外することにした．理由は，機能名や情報名を読み取ることができないと，記述すべき機能や情報連携が連携図からもれることになり，表記方法の比較にならないためである．このような被験者の絞込みを実施することにより，被験者の能力差が表記方法の評価に与える影響を極力小さくすることができると考える．

表 6.1 初学者対象の実験の実施条件

日時	2017.06.23(金)　16:30-18:00
場所	千葉県内　大学演習室
実験方法	定義文から，機能名,情報名を読取り，連携図を作成する.
評価方法	機能，情報連携の正答率と連携図の作成時間の比較
被験者の母集団	12名を2名1組の6班に分け，DFD,MDM各3班とする.
被験者の絞込条件	定義文の読取り能力が低い各々1班を評価対象外とする.
評価対象	DFD,MDM各2班の4班

このような被験者の絞込みを行ったため，評価対象の被験者は DFD と MDM 各 2 班の 4 班に分けられた大学 3,4 年生８名である．

被験者は，事前知識として DFD による連携図の作成方法を講義で学習している．MDM による連携図の作成方法については，講義と簡単な演習を実施することで習得させる． DFD や MDM を使って連携図を記述する際，時間の制約は設けず，各グループが完了したと判断した時点で終了とする．

実験では，最初に定義文を配布し，次に 2 つの表記方法で連携図を記述させ，最後に 2 つの表記方法の優れている点と劣っている点を考察させた．

実験の評価は機能，情報連携の正答率と連携図の作成時間を比較する．

実験で使用する機能名，情報名は定義文ですべて与える．実験で使用した定義文を図 6.8 に示す．内容は，食材の調達，顧客の予約と苦情に関する機能と情報連携である．

(1) A社は３点（B店，C店，D店）の店舗を展開する外食チェーン店である．
A社の業務の流れはいかのようになっている．
(2) 月末に営業本部で営業会議が開催され，翌月の各店舗の販売目標が決定される．
各店舗では，販売目標をもとに，各店舗の在庫を確認して，必要な食材を洗い出す．
(3) 物流部門では，各店舗の必要な商材をとりまとめて，生産者や小売業者に発注する．
野菜は無農薬の農家と直接取引をしている．
肉，魚はそれぞれ精肉店，鮮魚店に発注する．
ただし，緊急の場合には，店舗間で連絡し合い，自社トラックで食材を届ける場合もある．
(4) 材料は契約している物流企業に依頼して，仕入先から各店舗に直接輸送してもらう．
ただし，店舗に在庫の置き場が少ないため，店舗から物流業者に配送の連絡をしてから，配送してもらう．
(5) 顧客は直接各店舗に予約をする．
予約が当日の３日より前の場合には，確認の連絡をいれる．
(6) 顧客は，各店舗の他，本社の顧客窓口にサービスの苦情を言う．
苦情は営業本部を経由して，本社，全支店で共有する．

図 6.8　実験で使用した定義文

この定義文から読み取れる機能数は表 6.2 に示すように 12 種類(内訳は，顧客 1，企業内 7，協業事業者 4)である．同様に，情報連携数は表 6.3 に示すように 11 種類(内訳は，食材調達５，予約２．苦情４)である．MDM を用いて，これらの機能と情報連携を記述した連携図の正解例を図 6.9 に示す．表 6.2 に示した 12 種類の機能は，図 6.9 に示すように連携図の対角上に配置する．表 6.3 に示した 11 種類の情報連携を表す情報名は，発信元の機能の列と受信先の機能の行の交点のセルに記述する．なお，連携図内の括弧で囲んだ連携は物品の連携を示しており，情報連携ではないが，定義文を理解するために必要と考えられるため，記述してある．

表 6.2 機能の内容

No.	内訳	内容
1	協業事業者	農家
2		精肉店
3		鮮魚店
4		物流企業
5	企業内	A社本社
6		営業本部
7		物流部門
8		顧客窓口
9		B店
10		C店
11		D店
12	顧客	顧客

表 6.3　情報連携の内容

No.	内訳	内容	発信元	受信先
1	食材調達	販売目標の伝達	営業本部	各店舗
2		各店舗から発注依頼	各店舗	物流部門
3		食材発注	物流部門	各仕入先
4		各店舗から配送依頼	各店舗	物流企業
5		各店舗間の食材融通	各店舗	各店舗
6	予約	顧客の予約	顧客	各店舗
7		3日前以前の確認	各店舗	顧客
8	苦情	店舗へ	顧客	各店舗
9		顧客窓口へ	顧客	顧客窓口
10		営業本部まとめ	顧客窓口	営業本部
11		全社共有	営業本部	本社、各店舗

農家			（集荷）			食材発注					
	精肉店		（集荷）			食材発注					
		鮮魚店	（集荷）			食材発注					
（出荷）	（出荷）	（出荷）	物流企業					配送依頼	配送依頼	配送依頼	
				A社本社	苦情						
					営業本部		苦情				
						物流部門		発注依頼	発注依頼	発注依頼	
							顧客窓口				苦情
			（配送）		販売目標 苦情			B店	食材融通	食材融通	予約 苦情
			（配送）		販売目標 苦情			食材融通	C店	食材融通	予約 苦情
			（配送）		販売目標 苦情			食材融通	食材融通	D店	予約 苦情
								予約確認	予約確認	予約確認	顧客

図 6.9 MDM を用いた連携図の正解例

6.2.4 初学者を対象とした実験結果

初学者を対象とした実験結果の要約を表 6.4 に示す.

表 6.4　初学者を対象とした実験結果の要約

			MDM			DFD		
No.	評価項目	評価値(単位)	A班	B班	平均	C班	D班	平均
1	機能	正答数(個)	12	12	12.0	12	11	11.5
		誤答数(個)	0	0	0.0	0	1	0.5
		正答率(%)	100.0	100.0	100.0	100.0	91.7	95.8
2	情報連携	正答数(個)	8	7	7.5	8	5	6.5
		誤答数(個)	3	4	3.5	3	6	4.5
		正答率(%)	72.7	63.6	68.2	72.7	45.5	59.1
3	作成時間	時間(分)	25	13	19.0	30	30	30.0

この実験結果を整理すると，次の 2 点がわかる.

(1) 機能の正答率や情報連携の正答率は MDM のほうが高いが，大きな差はな

い．この結果は，被験者の絞込みを行ったためと考えられる．

(2) 作成時間の平均は MDM のほうが DFD より 11 分短い．

次に，DFD を用いた連携図の作成時間が長くなった要因について，DFD を用いて連携図を作成した被験者に対し，個別に聞き取り調査を実施した．その結果，DFD では，連携図を記述する過程で何度か手戻りが発生したことがわかった．連携図を作成するため，定義文を順番に読み進めながら，先に作成した連携図に機能や情報連携を追記していく．その際，情報連携のつなぎ間違いなどのエラーや機能の配置についての改善の必要性など，先に作成した連携図の修正点や改善点が後から見つかることがある．このような場合，どのように修正・改善していくか議論する過程で，何度か手戻りが発生し，最終的に連携図を書き直したと述べている．一方，MDM を用いて連携図を作成した被験者に対する聞き取り調査では，機能の配置を終えると，情報連携を記述する位置が一意に決定するため，書き終えた後に手戻りが発生することは少なかったと述べている．この結果，定義文を順番に読み進めながら，連携図を作成する際，混乱することがなかったとのことである．

以上の聞き取り調査の結果は，実験の最後に 2 つの表記方法の優れている点と劣っている点を考察させた結果とも一致している．被験者に実験の感想を求めたところ，ほぼ共通した回答が得られ，MDM と DFD の優れている点，劣っている点は相反する結果となった．MDM が優れている点として，機能や情報連携を記述する位置が固定しているため，連携図を作成しやすい．また，他人が作成した連携図を理解しやすい点を指摘した．劣っている点として，情報連携の向きを左上や右下という機能の位置と関係づけて解釈するルールとなっているため，情報連携が直感的にわかりにくい点を指摘した．反対に，DFD が優れている点として，情報連携の方向は矢印で表記するため，直感的に情報連携を把握しやすい点を指摘した．劣っている点として，機能を配置する自由度が高いため，うまく配置できた時には，情報連携の理解は容易であるが，配置に失敗した場合は情報連携を表す矢印線が交差して，情報連携の理解が困難となることを指摘した．また，他人が書いた DFD を理解することに時間がかかり，情報伝達の面で劣っていることも指摘した．

6.2.5 初学者を対象とした実験のまとめ

実験結果より，MDM が DFD に比べて，より短い時間で連携図を記述することができることが確認できた．この実験結果より，業務プロセスを分析することに慣れていない初学者から見た，MDM の表記方法としての特徴をまとめる．

MDM は機能や情報連携を記述する位置が固定しているため，定義文の内容を理解できれば，機械的に機能や情報連携を記述することができる．今回の実験で比較した DFD のように，機能の配置や情報連携を示す矢印線の交差など，本来は二次的であるはずの表記方法に気を取られることがなく，連携図の書き直しなどの手戻りが発生することが少ない点が，初学者にとって好適である．一方で，情報連携の向きを左上や右下という機能の位置と関係づけて解釈するルールとなっているため，直感的にわかりにくいという指摘がある．この点は，演習を含めた学習により MDM に慣れることや矢印線を補助的に活用することで克服することが可能であると考える．

以上のことから，MDM を使うことにより，学生のような初学者でも，短い時間で効率的に連携図を作成できることが確認できた．

6.3 情報システム開発経験者対象の実験による効果測定

6.3.1 情報システム開発経験者を対象とした実験の目的

情報システム開発経験者を対象とした，機能を分解した連携図を作成する実験により，MDM の表記方法上の特徴を明らかにする．また，DFD との比較実験により，上位機能同士の情報連携を下位機能同士の情報連携につなぎ換える際の MDM の効果を確認する．MDM は，階層を使った機能の上下関係をフラットに表現することができる．この表記により，MDM は，分解した機能の上下関係が把握できると同時に，分解した下位機能同士の情報連携が一覧できるという特徴をもつ．この特徴を活かすと，MDM は DFD に比べて，情報連携のつなぎ換えの際に発生する情報連携の抜けもれやつなぎ間違いといったエラーを少なくできると考える．

また，MDM の収容可能な機能数を DFD と比較すると，3 倍から 6 倍の収容能力をもつ．この特徴を活かすと，MDM は DFD に比べて，少ないシート枚数で連携図を記述できると考える．

本実験の目的は，機能を分解した連携図の記述に関して，MDM が情報連携のつなぎ換えのエラーを防ぐ効果があることや少ないシート枚数で連携図を記述できることを，DFD との比較実験を通して明らかにすることである．

6.3.2 情報システム開発経験者を対象とした実験の方法

(1) 被験者と題材を選定する考え方

実験の被験者は，情報システム開発を経験している 20 才代から 30 才代の若手のシステムエンジニアとする．いずれの被験者も業務プロセスを分析するような設計業務の経験がある．さらに，同じような年代のシステムエンジニアを選定しており，業務経験による能力差は少ないと考えられる．

実験の題材には，第 2 章の問題指摘で使用した「あるレストランチェーン A 社において，食材発注に電子取引を導入するための情報システム改修」を用いる．演習問題は 3 問あり，演習問題 1「電子取引導入前の連携図の作成」では，機能数 4 個，情報連携数 7 個の簡単な連携図を作成する．演習問題 2「電子取引導入後の連携図の作成」では，演習問題 1 で作成した連携図に，新たな機能と情報連携を追加し，機能数 5 個，情報連携数 13 個の連携図を作成する．演習問題 3「電子取引機能と発注・検収機能を分解した連携図の作成」では，演習問題 2 で作成した連携図のうち,電子取引機能と発注･検収機能の 2 つの機能を同時に分解し，分解した結果を記述した連携図を作成する．

(2) 実験手順の考え方

実験の目的は，機能を分解した連携図の記述に関して表記方法を比較することである．しかし，いきなり機能の分解した連携図を記述することはせず，図 2.3 に示した連携図を作成する手順を踏んで，演習を進めることとする．最初に，簡単な連携図を作成する演習問題 1 や演習問題 2 を行うことで，被験者に連携図の作成に慣れてもらう，その後，機能を分解した連携図を記述する演習問題 3 を実施することで，勘違いや考慮不足による手戻りなど表記方法以外で実験結果に差がでる要素を極力排除する．

6.3.3 情報システム開発経験者を対象とした実験条件

情報システム開発経験者を対象とした実験の実施条件を表 6.5 に示す．

表 6.5 情報システム開発経験者対象の実験の実施条件

日時	2017.12.19(火)　10:00-12:00
場所	東京都内　情報関連企業会議室
実験方法	定義文から，2つの機能を分解した連携図を作成する.
評価方法	情報連携の正答率と連携図を記述したシート枚数の比較
被験者の母集団	20名を10名ずつ2グループに分け，DFD,MDM各10名とする.
被験者の絞込条件	事前に連携図を作成する演習を行い，情報連携の正答数が多い上位4名ずつを評価対象とする.
評価対象	DFD,MDM各4名の8名

被験者の母集団は情報システム開発を経験している 20 才代から 30 才代の若手のシステムエンジニア 20 名である. いずれの被験者も DFD を使って業務プロセスを分析するような設計業務の経験がある．MDM による連携図の作成方法については，講義と簡単な演習を実施することで習得させる.

最初に，20 名の被験者を 10 名ずつ 2 つのグループに分ける．その各々のグループが，DFD，MDM を使って演習問題 1 の連携図を作成する．連携図を作成する時間は最初に定義文を配布してから 7 分とし，作成の途中であっても演習を打ち切ることとする．配布した定義文は，図 2.4 に示した電子取引導入前の定義文である.

次に，演習問題 2 も，20 名の被験者を 10 名ずつ 2 つのグループに分けて実施する．その各々のグループが，DFD，MDM を使って演習問題 2 の連携図を作成する．連携図を作成する時間は最初に定義文を配布してから 10 分とし，作成の途中であっても演習を打ち切ることとする．配布した定義文は，図 2.6 に示した電子取引導入後の定義文である.

演習問題 1 と演習問題 2 の結果，機能をすべて正答し，情報連携の正答数が多い上位 4 名ずつ合計 8 名の被験者を演習問題 3 の比較対象とする．このような実験手順を踏むことで，比較実験を行う被験者の能力差を極力小さくすることができると考える.

演習問題 3 の連携図を作成する時間は最初に定義文を配布してから 25 分とし，作成の途中であっても演習を打ち切ることとする．演習問題 3 で配布した定義文

は，図 2.10 に示した食材マスタ登録に関する定義文と図 2.13 に示した電子取引への検収に関する定義文である．正解例は，DFD の場合は図 2.16，MDM の場合は図 6.5 に示した連携図である．

実験の主な目的は，MDM が情報連携の抜けもれやつなぎ間違いといったエラーを防ぐ効果があることを，DFD との比較実験で明らかにすることである．情報連携のエラーは，上位機能同士の情報連携を下位機能同士の情報連携へつなぎ換える際に発生することが考えられる．そのため，情報連携をつなぎ換える過程で発生する情報連携の種類を明らかにし，その種類ごとにエラーの発生を比較することとする．情報連携には，機能間のつなぎかたによって 4 種類の情報連携がある．4 種類の情報連携の内訳は，(1) 1 つの機能を分解した下位機能間の情報連携，(2)2 つの機能を分解した下位機能間の情報連携，(3)分解後の下位機能と分解していない上位機能との情報連携，(4)分解していない上位機能間の情報連携，である．この実験では機能間のつなぎかたが違う 4 種類の情報連携について，DFD と MDM の正答率を評価することにする．

機能数は 10 個あるが，分解前の機能 5 個，分解後の機能 5 個である．分解前後の機能の内訳を表 6.6 に示す．

また，情報連携数は 20 個あるが，(1) 1 つの機能を分解した下位機能間の情報連携が 3 個，(2)2 つの機能を分解した下位機能間の情報連携が 3 個，(3)分解後の下位機能と分解していない上位機能との情報連携 9 個，(4)分解していない上位機能間の情報連携 5 個である．4 種類の情報連携の内訳を表 6.7 に示す．

表 6.6 分解前後の機能の内訳

No.	分解前の機能	分解後の機能
1	(1)発注・検収	
2		(1)発注
3		(2)検収
4		(3)在庫管理
5	(2)食材提供	
6	(3)食材調達	
7	(4)出納	
8	(5)電子取引	
9		(4)データ交換
10		(5)食材マスタ

表 6.7 情報連携の内訳

No.	分解の仕方	情報連携名	発信元	受信先
1	1つの機能を分解した下位機能間	(1)食材数量	在庫管理	発注
2		(2)食材数量	検収	在庫管理
3		(3)食材情報	食材マスタ	データ交換
4	2つの機能を分解した下位機能間	(1)食材発注	発注	データ交換
5		(2)検収	検収	データ交換
6		(3)食材情報	食材マスタ	発注
7	分解後の下位機能と分解していない上位機能間	(1)納品書	食材提供	検収
8		(2)納品情報	食材提供	データ交換
9		(3)食材情報	食材提供	食材マスタ
10		(4)手数料支払	出納	データ交換
11		(5)領収書	データ交換	出納
12		(6)食材発注	データ交換	食材提供
13		(7)検収	データ交換	食材提供
14		(8)発注・納品・検収	データ交換	食材調達
15		(9)食材情報	食材マスタ	食材調達
16	分解していない上位機能間	(1)請求	食材提供	食材調達
17		(2)領収書	食材提供	出納
18		(3)支払依頼	食材調達	出納
19		(4)支払	出納	食材提供
20		(5)食材基礎情報	食材調達	食材提供

もう 1 つの実験の目的は，MDM の機能や情報連携の収容能力の高さを DFD との比較により，明らかにすることである．収容能力の比較のため，実験では MDM と DFD を用いて，それぞれ同じ大きさのシートに連携図を記述する．そして，MDM と DFD で，連携図を記述するために使用したシート枚数を比較することとする．

6.3.4 情報システム開発経験者を対象とした実験結果

情報システム開発経験者を対象とした実験結果の要約を表 6.8 に示す．

情報連携の正答率は，MDM が 76.3%，DFD が 53.8%であり，MDM の正答率が高いことがわかる．情報連携について，詳しくみると，4 種類の情報連携のいずれも MDM の正答率が高いことがわかる．特に，2 つの機能を分解した下位機能間の情報連携において，MDM は正答率 83.3%に対して，DFD は正答率 50.0%と 33.3%の差がある．このことは，DFD では分解後の下位機能同士の情報連携の記述が難しいことを示しているといえる．

次に，分解していない上位機能間の情報連携をみると，MDM は正答率 80.0%

に対して，DFD は正答率 35.0%と 45.0%の差がある．事前に実施した演習問題 2 で正解が出ており，比較的回答が容易であると考えたが，DFD で記述した被験者に聞き取り調査をしたところ，制限時間内に情報連携を記述し切れなかったとの回答を得た．このことは，DFD では機能を分解した連携図の記述に手間がかかることを示しているといえる．

また，今回の実験では，機能と情報連携を記述するスペースを十分確保できるよう，A2(A4×4)サイズのシートを用意した．1 枚のシートで全員が連携図を記述することを期待したが，実際には，MDM で記述した 4 名が全員 1 枚のシートに記述できたのに対し，DFD で記述した 2 名が 2 枚のシートを使っている．このことは，MDM の機能と情報連携の収容能力が，DFD より優れていることを示しているといえる．

表 6.8 情報システム開発経験者を対象とした実験結果の要約

			MDM					DFD				
No.	情報連携の種類	評価値(単位)	A	B	C	D	平均	E	F	G	H	平均
1	(1)1機能を分解した下位機能間	正答数(個)	2	2	3	3	2.5	2	2	2	1	1.8
		誤答数(個)	1	1	0	0	0.5	1	1	1	2	1.3
		正答率(%)	66.7	66.7	100.0	100.0	83.3	66.7	66.7	66.7	33.3	58.3
2	(2)2機能を分解した下位機能間	正答数(個)	1	3	3	3	2.5	0	3	0	3	1.5
		誤答数(個)	2	0	0	0	0.5	3	0	3	0	1.5
		正答率(%)	33.3	100.0	100.0	100.0	83.3	0.0	100.0	0.0	100.0	50.0
3	(3)下位機能と上位機能間	正答数(個)	7	4	9	5	6.3	8	5	3	7	5.8
		誤答数(個)	2	5	0	4	2.8	1	4	6	2	3.3
		正答率(%)	77.8	44.4	100.0	55.6	69.4	88.9	55.6	33.3	77.8	63.9
4	(4)上位機能間	正答数(個)	5	1	5	5	4.0	1	1	1	4	1.8
		誤答数(個)	0	4	0	0	1.0	4	4	4	1	3.3
		正答率(%)	100.0	20.0	100.0	100.0	80.0	20.0	20.0	20.0	80.0	35.0
5	合計	正答数(個)	15	10	20	16	15.3	11	11	6	15	10.8
		誤答数(個)	5	10	0	4	4.8	9	9	14	5	9.3
		正答率(%)	75.0	50.0	100.0	80.0	76.3	55.0	55.0	30.0	75.0	53.8
6	シート枚数	枚数（枚）	1	1	1	1	1.0	1	2	1	2	1.5

なお，今回の実験は，白紙のシートを配布して，そのシートに回答を記述するかたちで実施した．また，演習問題が終了するごとに回答を記述したシートを回収することにした．この実験方法では，演習問題 3 は演習問題 2 の回答を記述したシートと別のシートに記述し直すことになる．一方，実際に MDM を適用するときには，スプレッドシートを活用し，演習問題 2 の連携図に対し，図 6.3 や図 6.4 に示した手順で機能の分解を実施することで，演習問題 3 の連携図を得ることができる．この場合，分解していない機能同士の情報連携は，引き継がれるため，今回の実験のような連携図の書き直しによる情報連携のエラーを防ぐことができる．

6.3.5 情報システム開発経験者を対象とした実験のまとめ

情報システム開発経験者からみた, MDM の表記方法としての特徴をまとめる. MDM は，階層を使って分解した機能の上下関係をフラットに表現することができる．この表記により，MDM は機能を分解した後も階層関係が把握できるだけではなく，分解した下位機能同士の情報連携を一覧できるという特徴をもつ．この特徴を活かすと，MDM は DFD に比べて，機能を分解する際に発生する情報連携のつなぎ換えのエラーを少なくできると考えられる.

今回の実験では，DFD との比較において，MDM は機能を分解する際に発生する情報連携の抜けもれやつなぎ間違いといったエラーを防ぐ効果があることを確認できた．このことは， MDM が階層を使って分解した機能の上下関係をわかりやすく表記できることを示している．特に，情報連携の種類の中でも記述が難しいと考えられる，分解した下位機能同士の情報連携の記述において，エラーを防ぐ効果があることを確認できた．このことは， MDM が階層を使って分解した下位機能同士の情報連携をわかりやすく表記できることを示している.

また，MDM は，連携図を記述するシート枚数を少なくできることが確認できた．このことは，MDM の機能と情報連携の収容能力が， DFD より優れていることを示している.

以上のことから，情報システム開発経験者が機能を分解した連携図を記述する際の MDM の効果を確認できた.

6.4 MDM の有効性確認のまとめ

MDM は，表 5.1 で示したように，情報連携表記方法上の課題を解決する仕組みをもつ．その課題を解決する仕組みの有効性を，情報システム開発に対する実用性という観点から，次の 3 段階で確認した.

最初に，MDM によって，DFD を用いて連携図を作成する際に発生する 3 つの煩雑性が解決していることを確認した．確認方法は，第 2 章の問題指摘で用いた例題を，MDM を用いて記述することによる．その際，図 2.3 に示した進め方と同じ手順で，MDM を用いた連携図を作成した.

新たな連携図を作成する際に発生する「情報連携の増加に伴い矢印線が交差する」という煩雑性-1 に対して，MDM では，矢印線を使わないため，連携図にお

ける矢印線の交差を考慮する必要がないことを確認できた．加えて，機能を追加する位置が二次元正方行列の対角上と決まっているため，機能の追加によって連携図全体の機能の配置を変える必要がないことや，既存の連携図から変化がない機能や情報連携は書き直す必要がないことを確認できた．

機能を分解した結果を記述する際に発生する「機能を分解すると機能数と情報連携数が増えるため，上位の機能と下位の機能にシートが分かれる」という煩雑性-2 に対して，MDM では，機能の上下関係は元の 1 枚のシートを拡張することで表現できるため，シート間の参照を必要としないことを確認できた．加えて，機能の上下関係はフラットに表現されるため，上下関係を示す引出し線などのガイドを必要としないことを確認できた．

分解後の下位機能同士の情報連携をたどる際に発生する「下位機能同士の情報連携を確認する際，上位の連携図を参照する必要がある」という煩雑性-3 に対して，MDM では，フラットな階層表現により，分解前の機能と分解後の機能を 1 枚のシート上に記述することができることを確認できた．加えて，分解した下位機能同士の行・列の交点に情報名を記述できるため，下位機能同士の情報連携を直接たどることができることを確認できた．

以上のことから，MDM により，DFD の煩雑性-1 から煩雑性-3 のすべてが解決することを確認できた．

次に, 初学者を対象に DFD と MDM を用いて連携図を作成する実験を実施し，連携図を記述する際の MDM の効果を確認した．実験の評価は，機能，情報連携の正答率と連携図の作成時間を比較する．実験の結果，機能の正答率や情報連携の正答率に大きな差はなかった．一方，MDM が DFD に比べて，より短い時間で連携図を記述することができることを確認できた．また，MDM を用いて連携図を作成した被験者に対する聞き取り調査では，機能の配置を終えると，情報連携を記述する位置が一意に決定するため，書き終えた後に手戻りが発生することは少なかったとの回答を得た．このことは，MDM は機能や情報連携を記述する位置が固定しているため，定義文の内容を理解できれば，機械的に機能や情報連携を記述することができることを示している．また，機能の配置や情報連携を示す矢印線の交差など，本来は二次的であるはずの表記方法に気を取られることがないため，連携図の書き直しなどの手戻りの発生が少ないことを示している．

以上のことから，MDM を使うことにより，初学者でも短い時間で効率的に連

携図を作成できることを確認できた.

最後に，情報システム開発経験者を対象に DFD と MDM を用いて機能を分解した連携図を作成する実験を実施し，上位機能同士の情報連携を下位機能同士の情報連携につなぎ換える際の MDM の効果を確認した．実験の評価は，あらかじめ決めた回答時間内に回答できた情報連携の正答率と連携図を記述したシート枚数を比較する．実験の結果，情報連携の正答率は DFD に比べ，MDM の正答率が高いことを確認できた．特に，情報連携の種類の中でも記述が難しいと考えられる，分解した下位機能同士の情報連携の記述においても，MDM の正答率が高いことを確認できた．このことは，MDM は機能を分解する際に発生する情報連携の抜けもれやつなぎ間違いといったエラーを防ぐ効果があることを示している．また, MDM は, 連携図を記述するシート枚数を少なくできることが確認できた．このことは，MDM の機能と情報連携の収容能力が，DFD より優れていることを示している．

以上のことから，情報システム開発経験者が機能を分解した連携図を記述する際の MDM の効果を確認できた.

このように，情報システム開発に対する実用性という観点から，MDM の有効性を 3 段階で確認することができた.

次章では，MDM の実システム開発への適用事例を示し，MDM が実際の情報システム開発プロジェクトに適用可能であることを確認する.

第 7 章　MDM の実システム開発への適用事例

7.1 情報連携の確認に MDM を適用した事例

7.1.1 A 社 電子取引導入プロジェクトの概要

A 社は，日本料理レストランを中心に，居酒屋，焼き肉店などを多店舗展開している中堅のレストランチェーンである．A 社は東京，大阪，名古屋の大都市経済圏を中心に店舗網を形成しており，着実な新規店舗の拡大や不採算店舗の撤退を実施することにより，事業基盤を強固なものにしている．A 社では，営業店改善プロジェクト，セントラルキッチン(工場)改善プロジェクトといった実務レベルの改善プロジェクトを継続的に実施しており，一定の成果をあげている．しかし，実務レベルのプロジェクトの推進過程において，本社業務を含めた全社的な業務の抜本的な見直しや情報システムの活用による合理化の必要性が認識されている．また，将来の業容拡大を実現するには，管理間接業務の生産性を向上し，全社の経営資源を顧客に価値を提供する業務に集中させる必要がある．そこで，店舗での仕入・販売，調達，会計といった会社のビジネスプロセスの見直し(BPR: Business Process Reengineering)を行い[33][34]，業務のアウトソーシングの検討を含めて，生産性の向上を図るためのプロジェクトがスタートした．

そのプロジェクトの中で，電子取引導入により，食材発注業務を社外にアウトソーシングすることが検討された．また，電子取引導入に伴って，食材発注業務だけではなく，商品本部の食材調達業務や経理部の出納業務などのビジネスプロセスの変更を検討することになる．したがって，これらの業務を支援する情報システムについても改修を検討する必要があった．

7.1.2 A 社 電子取引導入プロジェクトの問題

A 社の情報システム改修の検討は，DFD を用いて行われている．情報システムの改修は，すでに複数の情報システムが存在する中で，新たな情報システムを追加開発する形で進められる．第 2 章の問題指摘で使用した「食材発注に電子取引を導入するための情報システム改修」は，A 社の情報システム改修の一部である．情報システム改修の検討は，階層を用いた機能の分解によって実施する．

最初に, DFD を用いて全社の情報システムを俯瞰する大分類の連携図を作成する．次に，大分類の連携図に記述した機能を分解する．分解した結果を中分類の連携図として，機能ごとに複数のシートに記述する．さらに，中分類の連携図に記述した機能を分解する．分解した結果を小分類の連携図として，機能ごとに複数のシートに記述する．この小分類の連携図を作成した段階で，連携図のシート枚数は，表 7.1 に示すように A4 サイズ 62 枚になっている．

表 7.1　A 社の連携図のシート枚数

No.	分類	業務名	シート枚数
1	大分類	全社	1
2	中分類	本社業務	1
3	小分類	本社業務	6
4	中分類	商品開発	1
5	小分類	商品開発	3
6	中分類	購買	1
7	小分類	購買	4
8	中分類	製造	1
9	小分類	製造	5
10	中分類	物流	1
11	小分類	物流	4
12	中分類	外販	1
13	小分類	外販	1
14	中分類	営業	1
15	小分類	営業	8
16	中分類	エリア業務	1
17	小分類	エリア業務	4
18	中分類	店舗	2
19	小分類	店舗	16
20	合計		62

次に，それまでに作成した連携図を統合し，情報連携をたどることにより，情報連携の抜けもれ，重複，つなぎ間違いなどのエラーがないことを確認する必要がある．ところが，A 社では，DFD で記述した連携図が A4 サイズ 62 枚にも達したため，情報連携をたどることが難しくなった．その結果，情報連携の検証が進まなかった．

7.1.3 A 社 電子取引導入プロジェクトへの MDM の適用

A 社では，MDM を用いて，DFD では難しかった情報連携の検証を実施した．

最初に DFD で記述された大分類の連携図から，機能と情報連携を取りだし，二次元正方行列に配置した．次に，階層を用いた機能の分解を行い，中分類の機能と情報連携が書かれた連携図を作成した．最後に，もう一度，階層を用いた機能の分解を行い，小分類の機能と情報連携が書かれた連携図を作成した．

このように，MDM を用いて，階層をフラットに表現した連携図を作成することにより，上位の機能から下位の機能につなぎ換えた情報連携や分解後の下位機能同士の情報連携を直接たどることができるようになる．その結果，情報連携の検証を進めることができた．

7.1.4 A 社 電子取引導入プロジェクトへの MDM の適用の効果

A 社で作成した MDM を用いた連携図の大きさは，A0 サイズのシート 1 枚である．A0 サイズ(模造紙のサイズ)は，A4 サイズに換算するとシート 16 枚となる．

DFD で記述した連携図は表 7.1 に示したように，A4 サイズのシート 62 枚である．したがって，MDM は DFD に比べて機能と情報連携の収容能力が約 4 倍あることがわかる．このことから，MDM の「機能と情報連携の収容能力を高める」という表記方法の狙いが実現されていることが確認できた．

また，MDM を用いて，機能を中分類，小分類に分解することにより，機能の上下関係をフラットに表現する連携図を作成することができた．この連携図を使うと，上位の機能から下位の機能につなぎ換えた情報連携や分解後の下位機能同士の情報連携を直接たどることができるようになる．このことから，MDM の「引出線がなくても階層の上下関係を表現できる」という表記方法の狙いが実現されていることが確認できた．

このように，機能の上下関係をフラットに表現する連携図を模造紙サイズ 1 枚に集約することで，プロジェクトメンバ全員が情報連携の変更箇所を一斉に，同時に確認できることになる．このことが，情報連携の抜けもれ，重複，つなぎ間違いなどのエラーを減らしていると考えられる．

こうして，A 社では，MDM を活用することにより，BPR に伴う機能の追加と情報連携のつなぎ換えを確認することができ，その後の情報システム開発を円滑に進めることができた．

7.2 同一プロジェクトに 2 つの表記方法を適用した事例

7.2.1 B 社 ERP 導入プロジェクトの概要

工作機械製造を行う B 社のプロジェクトは，海外を含めた製品の販売管理を中心とした基幹業務の情報システム更新プロジェクトである．B 社のメインフレーム上で稼動する情報システムは，長年エンハンスを続けてきたことにより，メンテナンス性が悪くなっている．そのため，情報システムのメンテナンスにかかる運用コストを削減することが，大きな課題であった．この課題を解決するため，新たに統合業務管理パッケージ ERP(Enterprise Resource Plannning)を導入し，従来のメインフレーム上で稼動する情報システムを ERP に置き換えるプロジェクトがスタートした[35][36]．B 社の情報システムは，メインフレーム上の情報システムと連携する 40 個の周辺情報システムで構成されている．表 7.2 に従来のメインフレーム以外の周辺情報システムのうち，主な周辺情報システムの機能とその内容を示す．

表 7.2　B 社の主な周辺情報システムの機能

No.	機能名	内容
1	海外販売	海外現地法人の販売管理機能
2	受注入力	受注情報をEDIを活用して入力する機能
3	特殊品管理	顧客の個別仕様にあわせた受注生産製品の管理機能
4	標準品管理	在庫を持って受注に対応する標準規格製品の管理機能
5	倉庫管理	在庫品の入出庫やロケーション管理
6	購買申請	原材料の発注を依頼する機能
7	設計情報管理	製品設計に関するマスタ管理や原価管理を行う機能
8	経費精算	旅費など経費精算を行う機能
9	連結会計	子会社を含めた連結会計機能

このプロジェクトに用いられた情報連携の表記方法は図 7.1 に示す DFD と SDF である．DFD で記述した B 社の情報システム連携図をみると，ERP を中心に，表 7.2 に示した周辺の情報システムが複雑に情報連携していることがわかる．

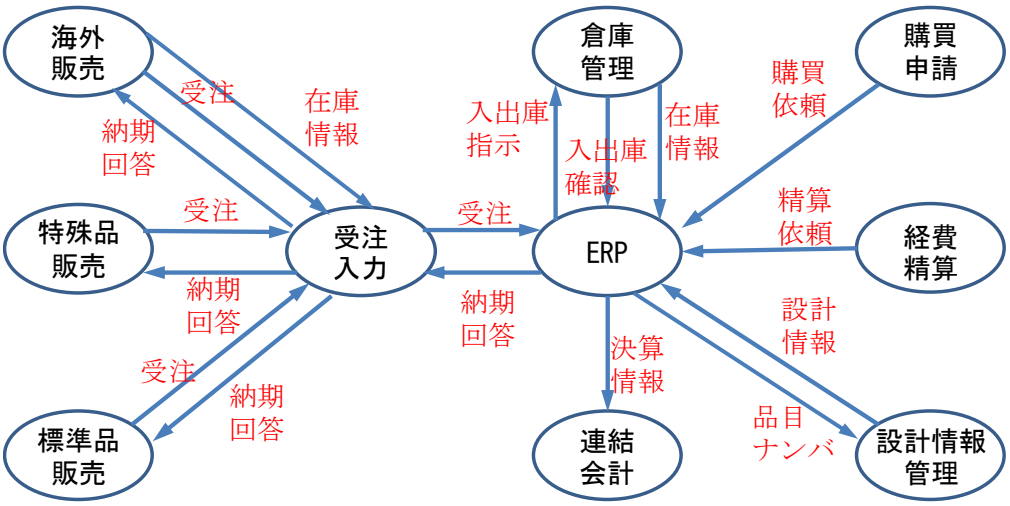


図 7.1　B 社の情報システム連携図(一部)(DFD)

7.2.2 B 社 ERP 導入プロジェクトの問題

B 社の ERP 導入プロジェクトは，ユーザが情報システムを総合的にテストする段階で，中断を余儀なくされた．中断の理由は，入出力情報やデータベースの情報項目の抜けもれや情報連携のつなぎ間違いによるエラーが多発したことによる．これらの情報連携のエラーの原因は，それらを記述すべき設計書に正しく記載されていないことであった．そのため，その設計書に基づいて作成するプログラムやモジュールにはエラーが内在されることになる．これらの情報項目や情報連携のエラーは，プログラムやモジュールのアルゴリズムが正しいかどうかを検証する機能中心のテストでは抽出が難しい．そのため，情報連携のエラーは B 社のようにプログラムやモジュールを結合して検証する総合テストの段階に至るまで発見されないことが多い．そこで， B 社では，DFD や SDF で機能や情報連携が書かれた設計書を調査し，エラーの原因を調査することになった．

7.2.3 B 社 ERP 導入プロジェクトへの MDM の適用

B 社では，MDM を用いて，機能と情報連携の整理を行なった．その際，DFD や SDF で記述した連携図から機能や情報連携をもれなく抜き出すため, MDM で連携図を記述する前に E-R(Entity Relation)図を作成した．DFD で記述した図 7.1 の情報連携は，表 7.3 のように E-R 図で記述することが可能である．

E-R 図は，情報連携を中心に，その情報の入力元機能と出力先機能を記述で

きる．従来手法の確認で述べたフロムツウチャートと同様，情報連携をたどる際に，出力先機能から入力元機能を見つけ出し，次の行をたどりなおさなければならないが，機能間の情報連携を抜き出し，つなぎ間違いがないか検証する際には有用な表記方法である．しかし，E-R 図では情報連携の全体像はわかりにくい．

この E-R 図をもとに, MDM を用いて, B 社の情報システム連携図を作成した．その結果，図 7.1 に示した ERP と周辺の情報システムとの情報連携は，図 7.2 に示すように MDM で記述することが可能になった．

表 7.3 B 社の E-R 図

No.	IN	情報連携	OUT
1	海外販売	在庫情報	受注入力
2	海外販売	受注	受注入力
3	受注入力	納期回答	海外販売
4	特殊品販売	受注	受注入力
5	受注入力	納期回答	特殊品販売
6	標準品販売	受注	受注入力
7	受注入力	納期回答	標準品販売
8	受注入力	受注	ERP
9	ERP	納期回答	受注入力
10	倉庫管理	入出庫確認	ERP
11	倉庫管理	在庫情報	ERP
12	ERP	入出庫指示	倉庫管理
13	購買申請	購買依頼	ERP
14	経費精算	精算依頼	ERP
15	設計情報管理	設計情報	ERP
16	ERP	品目ナンバ	設計情報管理
17	ERP	決算情報	連結会計

海外販売			納期回答						
	特殊品販売		納期回答						
		標準品販売	納期回答						
受注在庫情報	受注	受注	受注入力						納期回答
				倉庫管理					入出庫指示
					購買申請				
						設計情報管理			品目ナンバ
							経費精算		
								連結会計	
			受注	入出庫確認在庫情報	購買依頼	設計情報	精算依頼	決算情報	ERP

図 7.2　修正前の B 社の情報システム連携図(一部)(MDM)

次に，この MDM を活用して，情報連携の検証を実施する．ここでは，受注と納期回答に関する情報連携を取り上げる．受注には，海外販売と国内販売に対応する機能があるが，受注入力機能として一括処理されていた．そのため，海外との受注に関して，機能を分解し，情報連携を詳細に検討することが行われていないことがわかった．これに対し，MDM を活用して，機能と情報連携を検討した結果，海外との受注に関して，船積希望日付という情報項目がもれていることがわかった．同様に，納期回答の情報連携に船積日付という情報項目がもれていることがわかった．

また，MDM を活用した受注や納期回答の検討において，受注入力機能は海外販売と国内の特殊品販売，標準品販売と ERP の間に位置しているが，情報連携の加工を行っていないことがわかった．そのため，受注入力機能を経由せず，海外販売と国内の特殊品販売，標準品販売と ERP を直接情報連携しても問題がないことがわかった．こうして，受注や納期回答に関して，不必要な情報連携の重複を発見することができた．

このように，MDM の活用により，情報連携の抜けもれ，重複などのエラーを発見することができた．情報連携のエラーを修正した連携図を図 7.3 に示す．

海外販売									納期回答 船積日付
	特殊品 販売								納期回答
		標準品 販売							納期回答
			受注入力						
				倉庫管理					入出庫指示
					購買申請				
						設計情報 管理			品目 ナンバ
							経費精算		
								連結会計	
受注 船積希望日付 在庫情報	受注	受注		入出庫確認 在庫情報	購買依頼	設計情報	精算依頼	決算情報	ERP

図 7.3 修正後の B 社の情報システム連携図(一部)(MDM)

7.2.4 修正箇所の設計書への反映

B 社では，連携図の表記方法として DFD を用いた場合，機能を分解する前の連携図が書かれた設計書に対して，機能分解後の設計工程で見つかった情報連携の抜けもれ，重複などのエラーの修正箇所の反映は，設計書の書き直しになり，手間がかかるため，実施されてこなかった．

MDM の活用により，機能分解後の設計工程で見つかった情報連携の抜けもれ，重複などのエラーの修正箇所は，分解前の連携図を書き直しすることなく，設計書に反映することができるようになる．そのため，各設計工程で活用する情報連携に関する設計書に矛盾がなくなり，情報システムの運用・保守の局面で設計書を参照する際の効率向上につながっている．

7.2.5 B 社 ERP 導入プロジェクトへの MDM 適用の効果

B 社の DFD で記述された連携図の機能数，情報数はそれぞれ 40 件，300 件であった．この連携図をもとに 7.2.3 項で示した手順で，MDM を用いて連携図を記述し直した．その後，MDM で記述された連携図を用いて，情報連携をたどり直した結果，情報連携の抜けもれを 10 件発見することができた．また，抜けも

れを発見するときに，情報連携をたどり直すことになるため，その際，不必要な情報連携の重複を 7 件発見することができた．情報連携の抜けもれ 10 件と合わせて，計 17 件 5.7%の改善が見られる．この事例は，DFD に比べて，MDM が情報連携の全体像を把握しやすいこと，また，情報連携をたどりやすいことを示す好例である．このことから，MDM の「矢印線がなくても情報連携を表現できる」という表記方法の狙いが実現されていることが確認できた．

こうして，B 社では，MDM を活用することにより，ERP 導入に伴う機能の置き換えと情報連携の変更を実現し，情報システム開発を軌道に乗せることができた．

7.3 情報システム統合に MDM を適用した事例

7.3.1 C 社 情報システム統合プロジェクトの概要

C 社は，小売業の W 社，X 社，Y 社の 3 事業会社と，3 社の共同仕入れを担当する Z 社の計 4 社を傘下にもつホールディング会社である．W 社，X 社，Y 社は，同一形態の小売業である．この 3 社は業務提携の段階を経て，経営統合によって新たに C 社が設立された．経営統合前後の C 社と各社の関連を図 7.4 に示す．

C 社の傘下の小売事業会社 W 社，X 社，Y 社では，経営統合後も，それぞれ別の情報システムが稼働している．そのため，機能の重複による情報システムの非効率さが目立つようになっている．そこで，C 社では，傘下の小売事業会社 3 社の情報システムを統合し，経営統合後の情報システムを再設計するプロジェクトがスタートした．

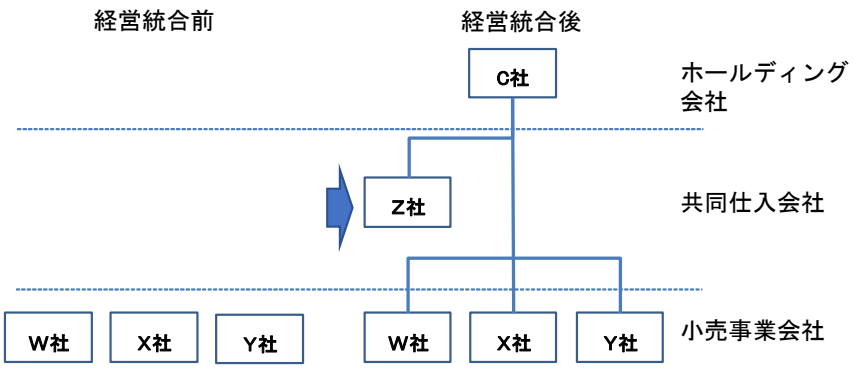


図 7.4 C 社と事業会社の関連

7.3.2 C 社 情報システム統合プロジェクトの問題

小売事業会社の既存の情報システムは，事業会社ごとの本部システム，店舗システム，物流システムといった業務個別に対応する情報システムとそれらの情報システム間の情報連携を示す連携図として把握されている．したがって，業務統合前の 3 社の情報システムを，3 社別々の連携図として記述することはできる．しかし，3 社別々の連携図から機能や情報連携の差異を吸収して，業務統合後の情報システム連携図を記述することはできなかった．その理由は，機能数が 3 社分あることから，1 社の 3 倍に膨れあがり，検討すべき機能や情報連携の組み合わせが検討可能な作業量を超えたためと思われる．さらに，共同仕入会社 Z 社の機能を，各事業会社のどの機能からどう取りだすか，また，Z 社と各事業会社との情報連携をどうするかについても，各事業会社の考え方が一致しない．そのため，小売事業会社の情報システム統合の検討に着手してから 3 年が経過しても，統合後の情報システムの要件定義ができないままであった[37][38][39].

統合後の情報システムの要件定義ができない理由として，次のような問題が解決されなかったことがあげられる.

(1) 検討すべき機能数が 1 社 30 個程度であったが，3 社で 3 倍の 87 個に膨れあがったため，検討可能な作業量を超えている.

(2) 3 社別々の業務手続きの差異を整理しているが,その差異を吸収する情報システムの機能を特定することが難しい.

(3) 機能の特定が難しいことから，その機能に関連する入出力情報やデータベースの情報項目の改修箇所を特定することが難しい.

(4) ある機能を改修すると，その影響が他の事業会社の機能にどのような影響を及ぼすのかがわからない.

(5) 3 年経過すると,情報システムに影響を及ぼすような新しい業務手続きが追加される.

この事例から，情報システム統合を進める際の表記方法上の問題として，統合後の情報システムの機能と情報連携を定義するため，複数枚の連携図を 1 枚の連携図に書き換えることが難しいことがあげられる.

7.3.3 C 社 情報システム統合プロジェクトへの MDM の適用

情報システムを統合するプロジェクトが進まないため，C 社では MDM を用いて統合後の情報システムの機能と情報連携を定義することになった[40][41]．このような情報システム統合における MDM を用いた連携図の作成手順を図 7.5 に示す.

(1) 各社の DFD を参考に連携図を MDM で書き直し，事業会社別に計 3 枚の連携図を作成する．ここでは，書き直し後の W 社の例を示す.

(2) 別々に作成した各社の連携図を，同一の連携図上に結合する．MDM のもつ行列の特性を活かすと，このように 3 枚の連携図を 1 枚に集約することができる.

(3) 類似機能が集まるように,機能の並び順を変更する．これにより，類似機能が連携図上の同じ場所に並び，かつ，それらの情報連携は維持されていることから，改修すべき機能と情報連携が特定できるようになる.

(4) 集めた類似機能を，1 つの機能に集約する．統合する情報システムが明らかになり，情報システムの詳細検討がやりやすくなる.

(5) 新たな機能を追加し，情報連携のつなぎ換えを行う.

この結果，機能数を 87 個から 45 個に集約することができた．また，情報連携数は 368 個から 239 個に集約することができた.

この設計手順にしたがって，C 社では DFD で記述された各社別々の連携図を MDM で記述された 1 枚の連携図に統合することができた．その結果，機能数は検討可能な 45 個になった．連携図を記述するシート枚数は A4 サイズ 9 枚分相当である.

図 7.5 情報システム統合の連携図の作成手順(MDM)

7.3.4 C 社 情報システム統合プロジェクトへの MDM 適用の効果

C 社では，複数連携図の結合と機能の並び順変更という MDM の特徴を活かして, 図 7.5 に示した手順で, 複数の連携図の統合を実施した. 3 社で機能数 87 個, 情報連携 368 個が書かれた連携図を，機能数 45 個，情報連携数 239 個の連携図に統合することができた.

このことから，MDM の「矢印線がなくても情報連携を表現できる」という表記方法の狙いが実現されていることが確認できた．また，MDM の「複数連携図の結合や機能の並び順変更といった操作性の高さ」を確認できた.

こうして，C 社では，MDM を活用することにより，会社統合後 3 年経過しても実現できなかった情報システム統合を，設計後 1.5 年で成功に導くことができた.

7.4 情報システム開発への適用事例のまとめ

この章で取り上げた MDM を適用した 3 社の情報システム開発事例の概要と MDM 適用の効果を表 7.4 にまとめる.

表 7.4 MDM を適用した情報システム開発事例

No	企業名	事業名	情報システム名	機能数	情報連携数	MDM適用の特徴	MDM適用の効果
1	Ａ社	和食チェーン	店舗システム	61	415	機能の追加による情報連携の変更	シート枚数の減少（A4換算62枚から16枚に減少）
2	Ｂ社	工作機械製造	需給管理システム	40	300	機能の置き換えによる情報連携の変更	情報連携のエラーの発見（300件の情報連携から17件のエラーの発見）
3	Ｃ社	ホームセンタ	物流システム	45	239	複数連携図の結合と機能の並び順変更	連携図の統合による機能数・情報連携数の削減（機能数87個→45個，情報連携数368個→239個）

A 社のプロジェクトは，ビジネスプロセスの見直し(BPR: Business Process Reengineering)による電子取引機能の追加とそれに伴う情報連携のつなぎ換えを行うプロジェクトである．連携図を記述する表記方法上の特徴としては，機能の追加による情報連携の変更がある.

A 社のプロジェクトでは，機能の追加と分解により DFD で記述された連携図

のシート枚数が多くなり，つなぎ換えた情報連携がたどりにくくなっている．そこで，MDM を用いて，機能を中分類，小分類に分解することにより，機能の上下関係をフラットに表現する連携図を作成した．DFD で記述した連携図が A4 サイズのシート 62 枚に対し，MDM では連携図を A0 サイズのシート 1 枚で記述することができた．A0 サイズは，A4 サイズに換算すると 16 枚であり，DFD の A4 サイズ 62 枚と比べると，MDM は約 4 倍の収容能力があることがわかる．

A 社プロジェクトの事例より，MDM の「機能と情報連携の収容能力を高める」という表記方法の狙いが実現されていることが確認できる．また，MDM の「引出線がなくても階層の上下関係を表現できる」という表記方法の狙いが実現されていることが確認できる．

B 社のプロジェクトは，海外を含めた製品の販売管理を中心とした基幹業務の情報システムを ERP 導入により置き換えるプロジェクトである．連携図を記述する表記方法上の特徴としては，機能の置き換えによる情報連携の変更がある．

B 社プロジェクトでは，機能の置き換えによる情報連携の抜けもれやつなぎ間違いによるエラーがあり，総合テストが進まない状況であった．そこで，MDM を適用して，機能の分解を行い下位機能同士の情報連携の検証を行った．その結果，DFD では見つからなかった情報連携のエラーを 17 件発見することができた．この B 社プロジェクトの事例は，DFD に比べて，MDM が情報連携の全体像を把握しやすいこと，また，情報連携をたどりやすいことを示している．

B 社プロジェクトの事例より，MDM の「矢印線がなくても情報連携を表現できる」という表記方法の狙いが実現されていることが確認できる．

C 社のプロジェクトは，企業統合に伴う情報システムの統合プロジェクトである．連携図を記述する表記方法上の特徴としては，複数連携図の結合と機能の並び順変更がある．

C 社のプロジェクトでは，企業の経営統合に伴う情報システム統合の検討が進まない状況であった．情報システム統合では，3 社別々の情報システム連携図から機能や情報連携の差異を吸収して，業務統合後の情報システム連携図を作成することが求められる．そこで，MDM を適用して，まず，3 社の統合前の情報システム連携図を作成した．その後，MDM で記述した複数連携図の結合と機能の並び順変更により，業務統合後の情報システム連携図に統合した．その結果，3 社で機能数 87 個，情報連携 368 個が書かれた連携図を，機能数 45 個，情報連携

数 239 個の連携図に統合することができた.

そして，会社統合後 3 年経過しても実現できなかった情報システム統合を，設計後 1.5 年で成功に導くことができた.

C 社プロジェクトの事例より，MDM の「矢印線がなくても情報連携を表現できる」という表記方法の狙いが実現されていることが確認できる．また，MDM の「複数連携図の結合や機能の並び順変更といった操作性の高さ」を確認できる.

以上より，MDM は，図 3.2 に示した「機能と情報連携の収容能力を高める」,「矢印線がなくても情報連携を表現できる」,「引出線がなくても階層の上下関係を表現できる」という表記方法の 3 つの狙いを実現していることがわかる．さらに，MDM の「複数連携図の結合や機能の並び順変更といった操作性の高さ」を実現していることがわかる.

このことから，MDM は，実際の情報システム開発プロジェクトに適用可能であることが確認できた.

第 8 章　結論

8.1 本論文のまとめ

本論文のテーマは,「企業情報システム開発における行列を用いた情報連携表記方法」である．本論文の研究対象は，企業における業務を支援する情報システムの開発時に用いられる表記方法である．

第 1 章では，研究の背景を述べ，本研究の目的を示した．

企業内には，様々な業務を支援する複数の情報システムが存在している．これらの情報システムは，互いに情報連携している．全体を構成する情報システムを部分的に改修・更新する場合，他の情報システムとの情報連携の追加，削除，つなぎ換えの検討が必要となる．この検討が不十分な場合，情報連携の抜けもれ，重複，つなぎ間違いといったエラーがおきる．このようなエラーは情報システム開発の最終段階で，ユーザが情報システムを総合的にテストする際に発見されることが多く，大きな手戻りになる．このようなエラーがおきる原因の 1 つに，情報システム開発に用いる情報連携の表記方法のわかりづらさがある．情報システム開発に用いる表記方法は，コンピュータを使った情報処理を機能とし，情報処理に必要なデータの入力と出力を与える IPO(Input Process Output)の形をとる．そして，情報システムは，こうした機能がデータを介して，多数連結されて，全体を構成する．また，情報システムは，コンピュータ処理が可能なプログラムあるいはモジュールと呼ばれるレベルまで分解・展開されるという階層的構造をもつ．情報システム開発に用いられる設計書には，情報処理の機能を円や楕円で表し，入力と出力は機能間の情報連携として矢印や線で表すフローチャート型の表記方法が用いられることが多い．

そこで，従来から広く用いられているフローチャート型表記方法の 2 つの問題を指摘した．1 点目は，機能を記述する位置が決まっていないため，情報連携をたどるための線と方向を示す矢印が必要となる．そのため，機能を記述する位置によっては，矢印線が交差して，情報連携がわかりにくくなることである．2 点目は，機能を分解する際に，分解後の機能を元の機能と別シートに記述すると，分解後の機能と元の機能との階層関係がわかりにくくなる．また，分解後の機能間の情報連携がたどりにくくなるということである．

次に，この 2 つの問題から，「機能の配置が決まっており，情報連携をたどるための矢印線を必要としない収容能力の高い表記方法の実現」と「機能を分解後も階層の上下関係が把握でき，同時に分解した下位機能同士の情報連携が一覧できる表記方法の実現」という 2 つの課題を設定した．

そして，研究の目的は，この 2 つの課題を解決する「行列を用いた表記方法 MDM(Matrix Description Method)」を提案し，その有効性を確認することである．

第 2 章では，情報システムを改修する例題を示し，その例題にフローチャート型表記方法の DFD(Data Flow Diagram)を適用することでおこる 3 つの煩雑性を明らかにした．煩雑性-1 は，新たな連携図を作成する際に発生する「情報連携の増加に伴い矢印線が交差するため，情報連携がたどりにくくなる」ことである．煩雑性-2 は，機能を分解した結果を記述する際に発生する「上位の機能と下位の機能にシートが分かれるため,機能の上下関係がわかりにくくなる」ことである．煩雑性-3 は，分解後の下位機能同士の情報連携をたどる際に発生する「連携先機能が上位機能名で表現されているため，下位機能同士の情報連携は直接たどれない」ことである．

第 3 章では，第 2 章で指摘した煩雑性の原因について，連携図を記述する手順にしたがって，作業分析を行なった．その結果，煩雑性の原因は，「1 枚のシートに書ける機能数と情報名数は限られる」,「矢印線で情報連携を表現しているため，線が交差する」,「分解した機能の連携先は相手の上位機能しかわからない」の 3 つであることが明らかになった．次に，この 3 つの原因を取り除く着眼を整理した．原因-1 を取り除く着眼は，着眼-1 機能を記述する位置を決める，着眼-2 情報連携を書く位置を決める，着眼-3 情報名をすべて書けるようにする，である．原因-2 を取り除く着眼は，着眼-4 情報連携を線がなくても表現できる，着眼-5 情報連携の向きを矢印がなくても表現できる，である．原因-3 を取り除く着眼は，着眼-6 分解した機能の上下関係がわかる，着眼-7 分解した下位機能と別の上位機能との情報連携がわかる，着眼-8 分解した下位機能同士の情報連携がわかる，である．次に，これらの着眼から，「機能と情報連携の収容能力を高める」,「矢印線がなくても情報連携を表現できる」,「引出し線がなくても階層の上下関係が表現できる」という，提案する表記方法の狙いを定義した．最後に，これらの狙いをまとめ,「機能の配置が決まっており，情報連携をたどるための線や矢印を必要

としない収容能力の高い表記方法の実現」，および，「機能を分解後も階層の上下関係が把握でき，同時に分解した下位機能同士の情報連携が一覧できる表記方法の実現」という情報連携表記方法上の 2 つの課題を提起した．これらの 2 つの課題から，情報連携表記方法に求められる具備要件は，「機能と情報連携を記述する位置が明示的に示せること」，および，「階層の上下関係が一覧できること」の 2 つに集約できることを示した．

第 4 章では，第 3 章で設定した課題を解決する表記方法として，線や矢印を使用せずに機能間の関係を示すことができる行列を用いた表記方法を中心に，従来手法を確認した．具体的には，(1)構造設計マトリクス(DSM:Design Structure Matrix)，(2)フロムツウチャート(from-to chart)，(3)UML アクティビティ図(Unified Modeling Language Activity Diagram)のスイムレーン，(4)機能構成図(DMM: Diamond Mandara Matrix)を調査し，情報連携表記方法上の 2 つの課題に対応している表記方法が見つからないことを確認した．

第 5 章では，第 3 章で指摘した課題を解決する表記方法として，行列を用いた表記方法 MDM を提案した．MDM の考え方は，連携図の記述に関する課題に対して，(1)機能は二次元正方行列の対角上に記述する，(2)情報連携は発信元の機能の列と受信先の機能の行の交点に配置する，(3)行列のセルを情報名の記入欄とすることにより，発生する可能性のあるすべての情報連携を網羅することができる，(4)情報名を記述する位置が決まるため，矢印線が不要になる，(5)情報連携は反時計回りの一方通行で表現する，という 5 つの仕組みで解決する．また，機能の分解結果の記述に関する課題に対して，(6)分解した機能の上下関係が把握できる，(7)分解した下位機能と別の上位機能との行・列の交点に情報名を記述できる，(8)分解した下位機能同士の行・列の交点に情報名を記述できる，という 3 つ仕組みで解決する．また，MDM が，第 3 章で示した情報連携表記方法に求められる具備要件について，１つめの「機能と情報連携を記述する位置が明示的に示せること」は，「機能と情報連携を記述する位置にアドレスをつけた二次元正方行列を用いる」ことにより満たす．２つめの「階層の上下関係が一覧できること」は，「階層を使った機能の上下関係をフラットに表現する」ことにより満たす．

第 6 章では，情報システム開発に対する実用性という観点から，MDM の有効性確認を 3 段階で行った．最初に，第 2 章の問題指摘で用いた例題を，MDM を用いて記述することにより，フローチャート型表記方法の DFD の問題が解決さ

れることを確認した．次に，初学者を対象に，MDM と DFD を用いて連携図を作成する実験を実施した．その実験結果より，MDM を使うことで，初学者でも短い時間で効率的に連携図を作成できる効果があることが確認できた．最後に，情報システム開発経験者を対象に，MDM と DFD を用いて機能を分解した連携図を作成する実験を実施した．その実験結果より，MDM には機能を分解する際に発生する情報連携の抜けもれやつなぎ間違いといったエラーを防ぐ効果があることが確認できた．また，MDM の機能と情報連携の収容能力が，DFD より優れていることを確認できた．

第 7 章では，情報システムを追加開発するプロジェクトに MDM を適用した 3 社の事例を示し，MDM が実際の情報システム開発に適用可能であることを確認した．A 社では，DFD で記述された連携図のシート枚数が多くなり，情報連携がたどりにくくなっていたため，MDM を用いて，機能の上下関係をフラットに表現する連携図を作成した． DFD で記述した連携図が A4 サイズのシート 62 枚に対し, MDM では連携図を A0 サイズのシート 1 枚(A4 サイズ換算で 16 枚)で記述することができた．この事例より，MDM の「機能と情報連携の収容能力を高める」という表記方法の狙いが実現されていることが確認できた．また，MDM の「引出線がなくても階層の上下関係を表現できる」という表記方法の狙いが実現されていることが確認できた．

B 社では，情報連携の抜けもれやつなぎ間違いによるエラーがあり，総合テストが進まない状況であったため, MDM で記述した連携図を作成した. その結果, 300 件の情報連携から，DFD では見つからなかったエラーを 17 件発見することができた．この B 社の事例は，DFD に比べて，MDM が情報連携の全体像を把握しやすいこと，また，情報連携をたどりやすいことを示している．この事例より，MDM の「矢印線がなくても情報連携を表現できる」という表記方法の狙いが実現されていることが確認できた．

C 社では，企業の経営統合に伴う情報システム統合の検討が進まない状況であった．C 社では，複数連携図の結合と機能の並び順変更という MDM の特徴を活用して，複数の連携図の統合を実施した．3 社で機能数 87 個，情報連携 368 個が書かれた連携図を，機能数 45 個，情報連携数 239 個の連携図に統合することができた．この事例より，MDM の「矢印線がなくても情報連携を表現できる」という表記方法の狙いが実現されていることが確認できた．また，MDM の「複

数連携図の結合や機能の並び順変更といった操作性の高さ」を確認できた．

以上より，研究の目的である「行列を用いた表記方法 MDM を提案し，その有効性を確認すること」が実現できていることを確認した．

8.2 今後の課題

今後の課題としては，提案した MDM をより広く浸透させ，より多くの情報システム開発に適用するための，MDM のツール化があげられる．また，MDM を情報システム開発に活用できるシステムエンジニアの教育と MDM の普及が考えられる．

(1) MDM のツール化

MDM をさらに実用的に情報システム開発に活用するためには，MDM のツール化が課題である．たとえば，行列の操作性を向上させるため，機能の分解や並び順変更を簡単に行うことができるツールが考えられる．また，情報システム開発で汎用的に使われる IPO (Input Process Output)図，E-R (Entity Relation)図などへの変換ツールがあると，個々の機能のアルゴリズム設計に役立つ[42]．逆に，IPO 図や E-R 図から MDM を作成する変換ツールがあると，個々の機能を設計した後の情報連携の検証に役立つと考えられる．

(2) MDM の教育と普及

MDM を活用できるシステムエンジニアの教育と MDM の普及が課題である．MDM を数多くの情報システム開発に適用し，これまで以上に普及させていくためには，MDM を活用できるシステムエンジニアの育成が必要である．そのためには，今回の演習に用いたような例題を整備し，MDM を体系的に学べるような教材を整備・開発する必要がある．

参考文献

[1] 南波幸雄：「企業情報システムアーキテクチャ」，翔泳社(2009)

[2] 情報処理推進機構　ソフトウェア・エンジニアリング・センター編：「経営者が参画する要求品質の確保：超上流から攻める IT 化の勘所」，オーム社(2005)

[3] Boehm B. W.: “Software Engineering”, IEEE Transaction on Computers, Vol.C-25, No.12, pp.1226-1241(1976-12)

[4] 高橋真吾：「システム学の基礎」，培風館(2007)

[5] 北原貞輔：「システム科学入門」，有斐閣(1986)

[6] Simon, H. A.: “The sciences of the Artificial Third edition”, The MIT Press, (1996) (稲葉元吉，吉原英樹訳：「システムの科学　第 3 版」，パーソナルメディア)

[7] Thayer, R. H., Dorfman, M., Davis, A. M. :　“Software Requirements Engineering”, IEEE (2003)

[8] 小林隆:「ビジネスプロセスのモデリングと設計」,コロナ社 (2005)

[9] DeMarco, T.: “Structured Analysis and System Specification”, Yourdon Press, New York (1978) (高梨智宏，黒田純一郎訳：「構造化分析とシステム仕様」，日経 BP 出版センター)

[10] Svobada, C. P.: “Structured Analysis”, Software Requirements Engineering Second Edition, pp.303-322, IEEE Computer Society (2000)

[11] 葉木陽一,末宗英利,前澤裕行,印東功,関沢智: “SEWB の開発思想と機能”，日立評論,Vol.70,No.2,pp7-14(1988.2)

[12] Kodosky, J., MacCrisken, J., Rymar, G.: “Visual Programming Using Strctured Data Flow”, Proceedings 1991 IEEE Workshop on Vissual Languages, pp.34-39, IEEE (1991)

[13] Saito, T., Udagawa, K., Mitsukuni, K.: “A Simultaneous Business Design Method Utilizing G-RD”, MIS Review, Vol.18, No.2, pp.51-79 (2013-3)

[14] 齊藤哲，光國光七郎 ：“G-RD を活用した同時的業務設計方法の提言”，日本経営工学会 論文誌, Vol.64, No.3, pp.428-437　(2013-10)

[15] 齊藤哲，渡部顕一郎，玉樹正人，光國光七郎: “機能と境界・全体連携図の表記方法に関する考察”,電気学会論文誌 C(電子・情報・システム部門誌), Vol.134, No.5, pp.737-746 (2014-5)

[16] 齊籐哲，光國光七郎: “機能と責任境界・管轄の全体連携図(G-RD)を活用したステークホルダ要求獲得の提案”, プロジェクトマネジメント学会誌, Vol.18, No.1, pp.8-13 (2016-2)

[17] Draft Federal Information Processing Standards Publication 183: “Announcing the Standard for INTEGRATION DEFINITION FOR FUNCTION MODELING (IDEF0)” (1993)

[18] 伊藤潔,廣田豊彦,冨士隆,熊谷敏,川端亮:「ソフトウェア工学演習」，オーム社,(2002)

[19] 古川善吾,廣田豊彦:「ソフトウェア工学の基礎Ⅶ」,近代科学社,(2001)

[20] OMG Document Number: formal/2011-01-03: “Business Process Modeling and Notation (BPMN) Version 2.0”: http://www.omg.org/spec/BPMN/2.0/

[21] 戸田保一，飯島淳一，速水治夫，堀内正博 :「ワークフロー」，日科技連出版社，(1998)

[22] Steven, D.E., Tyson, R. B.: “Design Structure Matrix Methods and Applications (Engineering Systems)”, The MIT Press, (2012) (西村秀和監訳，大富浩一，関研一訳 :「デザイン・ストラクチャー・マトリクス DSM」，慶應義塾大学出版会)

[23] Yassine, A.A.: “An Introduction to Modeling and Analyzing Complex Product Development Processes Using the Design Structure Matrix (DSM) Method”, *Quaderni di Management (Italian Management Review)*, Vol.9, (English translation ver.) (2004)

[24] 目代武史 :“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”，地域経済研究，第 17 号, pp. 25-42 (2006)

[25] 森俊樹 :“工程・組織効率化のための設計手法”，東芝レビュー, Vol. 60, No.1, pp. 44-47 (2005)

[26] 吉本一穂,大成尚,渡辺健 :「メソッドエンジニアリング」,朝倉書店,(2001)

[27] 日本経営工学会編：「生産管理用語辞典」，日本規格協会，(2002)

[28] OMG Document Number: ptc/2013-09-05: “OMG Unified Modeling Language™ (OMG UML) Version 2.5”: http://www.omg.org/spec/UML/2.5/

[29] Zachman, J. A.: “A Framework for information systems architecture”, IBM SYSTEM JOURNAL, vol.26, No.3, pp.276-292

[30] Sowa. J. F., Zachman, J. A.: “Extending and formalizing the framework for information systems architecture”, IBM SYSTEM JOURNAL, vol.31, No.3, pp.590-616

[31] 経済産業省 EA 策定ガイドライン ver.1.1.: http://www.meti.go.jp/policy/it_policy/itasociate/it.associate.htm/

[32] Saito, T., Shimoda, A., Mitsukuni, K.: “Comparative Study of Description Method for Business Process Visualization”, Proceedings of the 11th International Conference on Project Management (ProMAC2017), pp.984-992, The Society of Project Management, (2017)

[33] Hammer, M., Champy, J.: “Reengineering the Corporation: A Manifesto for Business Revolution”, Harper Business (1993) (野中郁次郎監訳：「リエンジニアリング革命」，日本経済新聞社)

[34] Davenport, T. H.: “Process Innovation: Reengineering Work through Information Technology”, Harvard Business School Press (1993) (卜部正夫，伊藤俊彦，杉野周，松島桂樹訳：「プロセス・イノベーション」，日経 BP 出版センター)

[35] 手島歩三，根来龍之，杉野周編：「ERP とビジネス改革」，日科技連出版社，(1998)

[36] 林寛，福田拓生：「業務改革と ERP」，日本図書刊行会，(1998)

[37] 一般社団法人情報サービス産業協会 REBOK 企画 WG 編：「要求工学知識体系　第 1 版」，近代科学社，(2011)

[38] 一般社団法人情報サービス産業協会 REBOK 企画 WG 編：「要求工学実践ガイド」，近代科学社，(2014)

[39] NTT ソフトウェアイノベーションセンタ編，飯村結香子，斎藤忍：「REBOK に基づく要求工学実践ガイド」，近代科学社，(2015)

[40] 経営情報学会システム統合特設研究部会 :「成功に導くシステム統合の論点」, 日科技連出版社, (2005)

[41] Saito, T., Mitsukuni, K.: "A Case Study of Application of G-RD to Business Integration after M&A", Proceedings of the 19th Asia-Pacific Decision Sciences Institute Conference (APDSI2014), Yokohama, Japan, (2014)

[42] 佐々木宏 :「図解　経営情報システム」, 同文館出版, (1997)

謝辞

本論文は，システムエンジニアとして長年情報システム開発に携わる中で，ずっと感じていた問題意識をテーマに，早稲田大学大学院　創造理工学研究科に在学中に行った研究をまとめたものです.

早稲田大学　理工学術院　創造理工学研究科　経営デザイン専攻　光國光七郎教授には，長期間にわたる懇切なご指導をいただきました．ここに深く感謝し，心から御礼申し上げます.

本論文をまとめるにあたり，副査を引き受けていただいた早稲田大学　理工学術院　創造理工学研究科　経営デザイン専攻の大成尚教授，髙田祥三教授，吉本一穂教授には,貴重なお時間を割いていただき,丁寧なご指導をいただきました．ここに深く感謝の意を表します.

千葉工業大学　社会システム科学研究科　マネジメント工学専攻　下田篤教授には，実験に関する具体的なアドバイスをたくさん頂戴しました．ここに厚く御礼を申し上げます．また，実験に参加くださった下田研究室の学生のみなさん，ならびに，日立産業制御ソリューションズの若手のシステムエンジニアのみなさんには，多大なご支援とご協力をいただきました．ここに謝意を表します.

本論文のテーマである表記方法の適用プロジェクトに参画いただいた日立製作所や日立コンサルティングの皆様に感謝の意を表します．また，事例研究の対象企業でお世話になった数多くの皆様にも感謝の意を表します.

最後に，陰ながら応援してくれた妻や家族に心から感謝いたします.

2018 年 3 月　齊籐哲

研究業績書

齊籐　哲
(2017 年　12 月現在)

種 類 別	題名,　発表・発行掲載誌名,　発表・発行年月,　連名者（申請者含む）
論文	（査読論文）
○	[1] A Simultaneous Business Design Method Utilizing G-RD, MIS Review, Vol.18, No.2, pp.51-79 (2013-3) Tetsu Saito, Kingo Udagawa, Koshichiro Mitsukuni
○	[2] G-RD を活用した同時的業務設計方法の提言, 日本経営工学会 論文誌, Vol.64, No.3, pp.428-437 (2013-10) 齊藤哲, 光國光七郎
○	[3] 機能と境界・全体連携図の表記方法に関する考察, 電気学会論文誌 C(電子・情報・システム部門誌), Vol.134, No.5, pp.737-746 (2014-5), 齊藤哲, 渡部顕一郎, 玉樹正人, 光國光七郎
講演	（国際会議）
	[1] A New Proposal for the Description Method of the Relations Between Businesses, 21st International Conference on Production Research (2011-8) Tetsu Saito, Kingo Udagawa, Koshichiro Mitsukuni
	[2] A Proposal of Simultaneous Business Design Method Utilizing G-RD, 17th Asia-Pacific Decision Sciences Institute Conference (2012-7) Tetsu Saito, Kingo Udagawa, Koshichiro Mitsukuni
	[3] Case Study of a Business Design Method to Define Requirements about Relations between Businesses, 7th International Conference on Project Management (2013-11) Tetsu Saito, Koshichiro Mitsukuni
	[4] A Case Study on Application of G-RD to Business Integration after M&A 19th Asia-Pacific Decision Sciences Institute Conference (2014-7) Tetsu Saito, Koshichiro Mitsukuni
	[5] The Proposal of Stakeholder Requirement Utilizing G-RD in Business Process Information System, 10th International Conference on Project Management (2016-11) Tetsu Saito, Koshichiro Mitsukuni

研究業績書

齊籐　哲
(2017 年　12 月現在)

種 類 別	題名，　発表・発行掲載誌名，　発表・発行年月，　連名者（申請者含む）
	[6] Comparative Study of Description Method for Business Process Visualization 11th International Conference on Project Management (2017-11) Tetsu Saito, Atsushi Shimoda, Koshichiro Mitsukuni (国内会議) [7] 業務間連携の表記方法に関する提言 日本経営工学会平成 23 年秋季研究大会(2011-11) 齊藤哲，光國光七郎 [8] 階層を用いた連携詳細化による業務設計方法の研究 日本経営工学会平成 24 年春季大会(2012-4) 齊藤哲，光國光七郎 [9] 機能と境界・全体連携図の表記方法に関する提案 電気学会平成 24 年 C 部門大会(2012-9) 光國光七郎，齊藤哲，渡部顕一郎，玉樹正人 [10] 連携を中心とした業務設計方法の事例研究 電気学会平成 25 年 C 部門大会(2013-9) 齊藤哲 ,光國光七郎 [11] G-RD を活用した業務ニーズ分析方法の事例 電気学会平成 25 年 C 部門大会(2013-9) 光國光七郎，齊藤哲 [12]企業合併に伴う業務統合における G-RD 活用事例の考察 日本経営工学会 2014 年春季大会(2014-5) 齊藤哲，光國光七郎 [13]論理的枠組みによる業務要素の定義方法 電気学会平成 26 年 C 部門大会(2014-9) 光國光七郎，齊藤哲 [14]複数の管理単位にまたがる業務を対象とした業務要件分析方法 電気学会平成 27 年 C 部門大会(2015-9) 光國光七郎，戸田雅之，齊藤哲 [15]企業業務処理システム開発における要求獲得に関する一考察 電気学会平成 28 年 C 部門大会(2016-9) 齊藤哲，戸田雅之，光國光七郎

研究業績書

齊籐　哲

(2017 年　12 月現在)

種 類 別	題名,　　発表・発行掲載誌名,　　発表・発行年月,　　連名者（申請者含む）
	[16] 企業情報システム開発における機能の分解・統合と連携の表記方法に関する考察 電気学会平成 29 年 C 部門大会(2017-9) 齊藤哲, 光國光七郎
解説記事	[1] 機能と責任境界･管轄の全体連携図(G-RD)を活用したステークホルダ要求獲得の提案 プロジェクトマネジメント学会誌, Vol.18, No.1, pp.8-13 (2016-2) 齊藤哲, 光國光七郎　（プロジェクトマネジメント学会 2017 年度文献賞受賞）